AIRO Springer Series 3

Massimo Paolucci
Anna Sciomachen
Pierpaolo Uberti *Editors*

Advances in Optimization and Decision Science for Society, Services and Enterprises

ODS, Genoa, Italy, September 4-7, 2019

AIRO
ASSOCIAZIONE ITALIANA DI RICERCA OPERATIVA
OPTIMIZATION AND DECISION SCIENCE

Springer

AIRO Springer Series

Volume 3

The **AIRO Springer Series** focuses on the relevance of operations research (OR) in the scientific world and in real life applications.

The series publishes contributed volumes, lectures notes, and monographs in English language resulting from workshops, conferences, courses, schools, seminars, and research activities carried out by AIRO, Associazione Italiana di Ricerca Operativa - Optimization and Decision Sciences: http://www.airo.org/index.php/it/.

The books in the series will discuss recent results and analyze new trends focusing on the following areas: Optimization and Operation Research, including Continuous, Discrete and Network Optimization, and related industrial and territorial applications. Interdisciplinary contributions, showing a fruitful collaboration of scientists with researchers from other fields to address complex applications, are welcome.

The series is aimed at providing useful reference material to students, academic and industrial researchers at an international level.

More information about this series at http://www.springer.com/series/15947

Massimo Paolucci • Anna Sciomachen •
Pierpaolo Uberti
Editors

Advances in Optimization and Decision Science for Society, Services and Enterprises

ODS, Genoa, Italy, September 4-7, 2019

AIRO
ASSOCIAZIONE ITALIANA DI RICERCA OPERATIVA
OPTIMIZATION AND DECISION SCIENCE

Springer

Editors
Massimo Paolucci
Department of Informatics, Bioengineering, Robotics, and Systems Engineering (DIBRIS)
University of Genoa
Genoa, Italy

Anna Sciomachen
Department of Economics and Business Studies (DIEC)
University of Genoa
Genoa, Italy

Pierpaolo Uberti
Department of Economics and Business Studies (DIEC)
University of Genoa
Genoa, Italy

ISSN 2523-7047 ISSN 2523-7055 (electronic)
AIRO Springer Series
ISBN 978-3-030-34959-2 ISBN 978-3-030-34960-8 (eBook)
https://doi.org/10.1007/978-3-030-34960-8

This Springer imprint is published by the registered company Springer Nature Switzerland AG.
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This volume is the third in the AIRO Springer Series. It contains very recent results in the field of optimization and decision science, aimed at solving complex problems.

The volume is mainly addressed to the operations research (OR) community but, due to its interdisciplinary contents, it will also be of great interest for scholars and researchers from many scientific disciplines, including computer sciences, economics, mathematics, and engineering. OR is known as the discipline of optimization applied to real-world problems and to complex decision-making fields. The focus is on mathematical and quantitative methods aimed at determining optimal or near-optimal solutions in acceptable computation times. Moreover, the volume contains real industrial applications, making it interesting for practitioners facing decision problems in logistics, manufacturing production, and services. Readers will surely find innovative ideas, from both a methodological and an applied perspective.

This volume includes both applications and theoretical results that represent state-of-the-art research contributions in the following fields and areas: smart port terminal operations; data exploitation: methods and applications; financial modeling and portfolio optimization; optimization in public transport; optimization in machine learning; support to Industry 4.0 and smart manufacturing; health care management and planning; equilibrium problems, variational models, and applications; mixed integer programming; transportation networks performance and reliability; vehicle routing problems; stochastic programming: optimization under uncertainty and applications; rail port operations; combinatorial optimization; OR applications in routing; inventory; graphs; traveling salesman and arc touting problems; and optimization in telecommunication networks and queuing systems.

The results published in this volume have been accepted for publication after a double-blind peer review process by experts in OR that had the aim of guaranteeing the quality of the contents. The editors want to thank the referees for their interesting comments, for their suggestions on how to improve the contents, and for completing the review process within the allowed time.

The editorial choice for this volume was to force the authors to focus on the central results obtained in their research. For this reason, the length of the contributions has been limited, with only a few exceptions. As a consequence, the volume provides very interesting results. As editors, we expect and hope that most of the papers included in the volume will be further developed so that they are suitable for publication in top-rated international journals. Indeed, the fact that some of the contributions focus on theoretical results while others deal with applications leaves much scope for empirical and theoretical aspects, respectively, to be developed in future research. The adopted approach was also designed to encourage young researchers with little experience in academic research, giving them the possibility to publish preliminary results and to experience the review process.

The contributions included in this volume are a subset of the works presented at ODS2019—International Conference on Optimization and Decision Science. The conference was the 49th annual meeting organized by the Italian Operations Research Society (AIRO) and took place in Genoa, Italy, on September 4–7, 2019.

Genoa, Italy — Massimo Paolucci
Genoa, Italy — Anna Sciomachen
Genoa, Italy — Pierpaolo Uberti
September 2019

Contents

About the Editors

Massimo Paolucci received a PhD in electronic and computer science in 1990. He is Associate Professor in Operations Research at the Department of Informatics, Bioengineering, Robotics, and System Engineering (DIBRIS) of the University of Genoa. His research activities are focused on metaheuristic and matheuristic algorithms for combinatorial optimization problems, planning and scheduling, decision support systems, and multi-criteria methods. Reference fields of application are intermodal logistics and shipping and manufacturing.

Anna Sciomachen is Full Professor of Operations Research at the Department of Economics and Business Studies, University of Genoa, where she is Coordinator of the Master of Science in Management of Maritime and Port Enterprises and teaches optimization and simulation methods for logistics. She is a past President of the Italian Society of Operations Research. Her main research fields are optimization models and heuristic methods in distributive logistics and multimodal transportation networks, liner problems, stowage planning, simulation techniques for performance analysis, and location-routing problems.

Pierpaolo Uberti received his PhD in Mathematics for Financial Markets from the University of Milano-Bicocca in 2010 for a dissertation on "Higher Moments Asset Allocation." Since 2011, he has been a researcher at the University of Genoa. His research interests cover the fields of quantitative finance, optimization, portfolio selection, and risk measures.

A Receding Horizon Approach for Berth Allocation Based on Random Search Optimization

Cristiano Cervellera, Mauro Gaggero, and Danilo Macciò

Abstract An approach to address the berth allocation problem is presented that is based on the receding horizon paradigm. In more detail, berthing decisions are computed by solving an optimization problem at each time step aimed at minimizing the waiting times of vessels, exploiting predictions on the ship arrivals and berth occupancy over a moving window starting from the current time instant. A discrete time dynamic model is devised to forecast the state of the terminal in the forward window, and a computationally-efficient approximate solution method based on random search is proposed. The considered framework can be used either for real time planning or scheduling in advance. Simulation results are reported to show the effectiveness of the method in different terminal configurations, forward horizons, and traffic intensities, in comparison with state-of-the-art approaches.

Keywords Berth allocation · Receding horizon · Random search optimization

1 Introduction

Berth allocation is a classic problem related to the optimization of container terminals. It consists in choosing the berthing positions for ships arriving at a terminal within a certain time horizon. A comprehensive collection of the available works in the literature is reported in [3, 4], together with a systematic classification of the various problem instances. In general, the most diffused approach formalizes the problem as a mixed-integer programming one with several variables and constraints, according to an *a priori* scheduling paradigm. However, the solution of this problem is very difficult, especially for many vessels and berths [9, 10]. Thus, researchers have devoted lots of efforts to find approximate solutions in a reduced amount of time, also for large instances of the problem, using heuristic approaches.

C. Cervellera · M. Gaggero (✉) · D. Macciò
National Research Council of Italy, CNR-INM, Genoa, Italy
e-mail: cristiano.cervellera@cnr.it; mauro.gaggero@cnr.it; danilo.maccio@cnr.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_1

Techniques relying on evolutionary or genetic algorithms, as well as methods based on tabu search and simulated annealing, are noticeable examples (see, among others, [6, 8, 11, 13, 14]). The most important issue is the assumption that the information on service times and arrivals of vessels will hold true in the future, so that static scheduling plans can be provided in advance. Rescheduling is needed in the case of large deviations from the forecast values or in a dynamic environment where ships arrive continuously, which may require to solve complex optimization problems in real time or exploit adaptation techniques not easily generalizable [15, 17].

In this work, we solve the berth allocation problem through a multistage decision approach, which consists in splitting the scheduling of a certain amount of ships within a given horizon into several smaller problems at various time steps, where we have to choose the berthing positions for ships that have just arrived at the terminal and are waiting to be served. The solution of the various problems is addressed through an approximate method based on random search, which guarantees a reduction of the overall computational effort and therefore the possibility of dealing with scenarios characterized by many vessels and berths. The proposed approach can be used either for planning in advance or real time berth allocation to face, for instance, unexpected deviations from the planned situation.

In more detail, we present a discrete time dynamic model of the terminal with state variables representing berth occupancy and decision variables accounting for the choices of ships that enter the terminal in free berths, among those that are ready to be served. Decisions are taken by solving optimization problems to minimize the waiting times of vessels. We adopt the receding horizon paradigm [12], which consists in optimizing over a time window starting from the current time instant up to a certain horizon in the future, exploiting predictions on the arrival times of ships and evolution of the terminal given by the considered dynamic model. Such an approach has been successfully employed in various contexts, including container terminals [1, 2, 16]. Its use for berth and quay crane allocation has been studied in [5], where an event-based formalization with Petri nets is adopted, a fixed horizon of ships (and not of time steps) is considered in the future, and a quasi-exhaustive search is adopted to perform optimization, which may limit the application to a reduced number of vessels. Instead, here we present a dynamic system representation of the berthing environment and focus on the berth allocation problem, even if the extension to consider also the quay crane allocation is straightforward. An optimization based on random search is presented that avoids using an enumerative approach, thus allowing to consider longer horizons in the future. In this perspective, choosing the ships that enter the terminal at each time step among those that are waiting to be served may be suboptimal, but it allows to have smaller search spaces with respect to scheduling all the vessels arriving in a given time window, including those not yet arrived at the terminal. This enables dealing with scenarios characterized by large amount of ships and berths with a reduced computational effort. In particular, sequences of feasible decisions are randomly generated, and the optimal solution is constructed iteratively exploiting a pruning mechanism to reduce the required burden.

The main advantages of the proposed approach can be summarized as follows: 1) owing to the reduced computational effort, the problem can be solved in real time. However, the proposed framework can be still used to perform planning in advance by computing a sequence of optimal actions for any desired horizon length in the future; 2) uncertainties on the arrival or handling times of ships can be easily taken into account by minimizing expected costs averaged over uncertainties; 3) decisions are taken as a function of the current state of the terminal, hence it is possible to react to deviations from the planned situation straightforwardly, without the need of adaptation mechanisms; 4) any performance goal can be considered with no conceptual difficulties, taking into account several aspects, such as service priorities, fuel consumption, and CO_2 emissions; 5) the extension to more complex scenarios, such as the simultaneous berth and quay crane allocation, as well as the account for tides or time-dependent limitations on berths and vessels, is straightforward.

The rest of this work is organized as follows. Section 2 reports the dynamic model of the berthing process. Section 3 presents the proposed receding horizon approach and the technique used to find suboptimal solutions. Preliminary simulation results are shown in Sect. 4, while conclusions are drawn in Sect. 5.

2 Dynamic Model of the Terminal

We present a model of the berthing process in terms of a discrete time dynamic system for $t = 0, 1, \ldots$. More specifically, we focus on a terminal with a quay partitioned in B discrete berthing positions. Ships to be served are contained in a queue (potentially of infinite length) denoted by $\bar{Q}$. Each vessel is indexed by a positive identifier j, and is characterized by a known time $t_j^{\text{arr}} \in \mathbb{N}$ when it arrives at the terminal and handling times $b_{ij} \in \mathbb{N}$, $i = 1, \ldots, B$, which represent the amount of time steps needed to complete the loading/unloading operations of ship j when served at berth i. To avoid burdening the notation in the following, we consider also a fictitious vessel with $j = 0$, $t_j^{\text{arr}} = 0$, and $b_{i0} = 0$ for all i. Let $A(t)$ be the set of vessels arriving at the terminal at time t, i.e., $A(t) := \{j \in \bar{Q} : t_j^{\text{arr}} = t\}$.

To keep track of the ships served at the various berths, we introduce B state variables at each time $t = 0, 1, \ldots$, denoted by $s_i(t) \in \mathbb{Z}$, $i = 1, \ldots, B$, which represent the remaining service time of berth i at time t. If $s_i(t)$ is positive, the berth i is serving a ship, and loading/unloading operations are expected to be concluded in $s_i(t)$ time steps. If it is equal to zero, the operations are just finished and a new ship can be berthed. If it is negative, the berth is idle from $|s_i(t)|$ time steps, and therefore it is ready to serve a new vessel. Besides, at each time $t = 0, 1, \ldots$, we define a subset of the queue $\bar{Q}$, denoted by $Q(t)$, made up by the vessels that are arrived and are waiting to enter the terminal and start loading/unloading operations.

To decide the ships that are berthed, we consider B decision variables for $t = 0, 1, \ldots$, denoted by $a_i(t) \in \mathbb{N}$, $i = 1, \ldots, B$, which represent the identifier of the vessel that is let into berth i at time t. If $a_i(t) = 0$, no ship enters berth i. Let us collect all the decision variables in the vector $\underline{a}(t) := (a_1(t), \ldots, a_B(t))$.

The state variables $s_i(t)$ and $Q(t)$ evolve according to the following dynamic equations depending on the decisions that are taken at each time step $t = 0, 1, \ldots$:

$$s_i(t+1) = (s_i(t) - 1)\,(1 - \chi(a_i(t))) + b_{i a_i(t)}, \quad i = 1, \ldots, B, \tag{1a}$$

$$Q(t+1) = Q(t) \cup A(t) \setminus \{a_i(t), i = 1, \ldots, B\}, \tag{1b}$$

where the function $\chi(\cdot)$ is such that $\chi(z) = 1$ if $z \neq 0$ and $\chi(z) = 0$ otherwise. If no ship enters berth i at time t, we have $a_i(t) = 0$ and therefore $\chi(a_i(t)) = 0$, and the remaining service time is reduced by one since $b_{i0} = 0$ for all i. If a new vessel is let into berth i, we have $a_i(t) \neq 0$ and therefore $\chi(a_i(t)) = 1$, and the service time is updated with the handling time of the corresponding vessel $b_{i a_i(t)}$. As regards $Q(t)$, we have a simple set dynamics: the set of vessels waiting to be served at time $t+1$ is equal to the same set at time t plus the new arrivals $A(t)$ minus the ships that enter the terminal at time t, which depend on the decisions $\underline{a}(t)$.

The goal is to choose, among the vessels that are ready to berth, the ones that enter the terminal at each time step and the corresponding destination berth, according to a suitable criterion. The chosen criterion and the technique used to perform decisions will be detailed in the next section. Now, we define a feasibility set where the decision variables can take their values. First, we assume to be able to berth only ships that are contained in the set $Q(t) \cup \{0\}$, as it includes vessels that are arrived at the terminal and are waiting to be served together with the fictitious ship. Therefore, we impose $a_i(t) \in Q(t) \cup \{0\}$ for all $i = 1, \ldots, B$ and $t = 0, 1, \ldots$. The choice that no ship enters a free berth even if there are vessels ready to berth is a feasible action. This may be helpful to keep a berth free until the arrival of another ship, characterized, for instance, by a higher priority or a larger amount of containers to load/unload. Moreover, to avoid that a new ship is let into a busy berth, we impose the following for $t = 0, 1, \ldots$:

$$a_i(t) \leq M\,(1 - \sigma(s_i(t))), \quad i = 1, \ldots, B, \tag{2}$$

where $\sigma(\cdot)$ is the unit step function, i.e., $\sigma(z) = 1$ if $z > 0$ and $\sigma(z) = 0$ otherwise, and M is a large positive constant. If berth i is busy, i.e., $s_i(t) > 0$, (2) is equivalent to $a_i(t) \leq 0$. Together with $a_i(t) \in Q(t) \cup \{0\}$ and the positive ship identifiers, this means that $a_i(t)$ is constrained to be equal to 0, i.e., no ship is let into berth i. If a berth is free, i.e., $s_i(t) \leq 0$, constraint (2) is trivially satisfied. Further, a ship cannot enter the terminal in more than one berth, i.e., for $t = 0, 1, \ldots$ we consider the following constraint, ensuring that at most one of the decision variables $a_1(t), \ldots, a_B(t)$ is equal to j at a certain time t:

$$\sum_{i=1}^{B} 1 - \chi\,(a_i(t) - j) \leq 1, \quad j \in Q(t). \tag{3}$$

Constraints (2), (3), and $a_i(t) \in Q(t) \cup \{0\}$, $i = 1, \ldots, B$, define a time-varying feasibility set denoted by $\mathscr{A}_t$, where the decision vector $\underline{a}(t)$ can take its values.

3 Receding Horizon Berth Allocation

In this section, we describe the receding horizon approach to select at each time step the best ships to be berthed according to a chosen performance criterion. Such an approach requires the solution of a sequence of optimization problems, which can be found either for planning in advance or real time scheduling. The proposed technique is general, and any performance index for the berthing policy can be considered by defining a suitable cost function (4). For instance, we may choose to minimize berth idle times, fuel consumption, total waiting and service times of ships, and so on. Without loss of generality, here we minimize the waiting times of vessels, i.e., the difference between the time instants when they arrive at the terminal and the ones when they are berthed and start loading/unloading operations. Hence, at each $t = 0, 1, \ldots$, we define the cost

$$c(t) = \sum_{j \in Q(t)} \left(t - t_j^{\text{arr}}\right) \prod_{i=1}^{B} \chi\left(j - a_i(t)\right), \tag{4}$$

where the first term is proportional to the waiting time of ships, while the second one avoids to penalize ships entering the terminal at time t, i.e., such that $a_i(t) = j$ for a certain i.

According to the receding horizon paradigm, we define at each time $t = 0, 1, \ldots$ a window of length $T > 0$ starting from the current time step, in which we decide the vessels that enter the terminal among those that are ready and the corresponding berthing positions. To this purpose, we exploit predictions on future arrivals of ships and evolution of the berth occupancy within this window, based on the dynamic model (1). Thus, we have to solve the following receding horizon berth allocation problem, referred to as RHBA-T, where T is the length of the forward horizon.

Problem RHBA-T At each time $t = 0, 1, \ldots$, solve

$$\min_{\underline{a}(t) \in \mathscr{A}_t, \ldots, \underline{a}(t+T-1) \in \mathscr{A}_{t+T-1}} \sum_{k=t}^{t+T-1} c(k), \tag{5}$$

subject to the dynamic equation (1), which represents a further constraint. □

Once the optimal values $\underline{a}^\circ(t), \ldots, \underline{a}^\circ(t+T-1)$ have been computed, only the first decision $\underline{a}^\circ(t)$ is retained and applied. The procedure is iterated for the other time steps, with a one-step-forward shift of the time window $[t, t+T-1]$. Figure 1 sketches the considered approach, where the forward horizons at times t and $t+1$ are reported. Problem RHBA-T is a nonlinear integer optimization problem involving non-smooth functions. The number of unknowns is given by $B \cdot T$, i.e., it is proportional to the number of berths and to the length of the decision horizon. Since all the involved quantities are integers, the problem is purely combinatorial. At least in principle, it is possible to find an exact solution by resorting to an

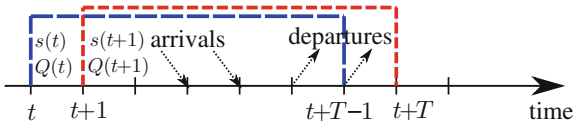


Fig. 1 Sketch of the receding horizon approach with the forward windows at times t and $t+1$

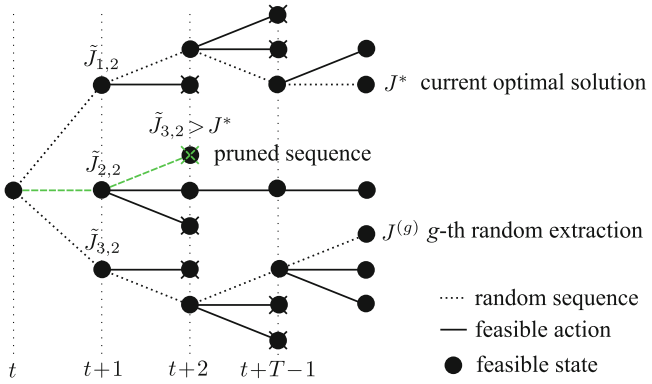


Fig. 2 Example of enumeration tree of the solutions of Problem RHBA-T at time t

enumeration procedure. However, for large horizons T and terminals with many berths, such an approach requires a too large computational effort. Hence, we must resort to approximation techniques. In this work, we focus on an approximate solution method based on random sampling (see, e.g., [7, chap. 3]).

In more detail, at each time t, we randomly extract a value for $\underline{a}(t)$ from the corresponding feasibility set $\mathscr{A}_t$. Then, we let the model evolve at time $t+1$ according to (1) with the considered decision vector. At time $t+1$, a new random extraction from the set $\mathscr{A}_{t+1}$ is done to select a value for $\underline{a}(t+1)$. Such a procedure is iterated up to the end of the forward window, i.e., up to the time $t+T-1$. To guarantee a "good" coverage of the set $\mathscr{A}_t \times \cdots \times \mathscr{A}_{t+T-1}$, we extract a total of G random sequences $\underline{a}^{(g)}(t), \ldots, \underline{a}^{(g)}(t+T-1)$, $g = 1, \ldots, G$, where G is a positive constant. Then, we compute the corresponding cost $J^{(g)}$ as in (5) for each sequence. The sequence corresponding to the minimum cost is selected as the best approximation of the true optimal solution of Problem RHBA-T. Clearly, the larger is G, the better is the quality of the approximation, but the greater is the computational effort. Thus, a tradeoff between accuracy and computational burden is needed. To this end, notice that all the possible decisions in Problem RHBA-T at a certain time t can be represented through an enumeration tree, as depicted in Fig. 2, where nodes and arcs are feasible states and actions in the interval $[t, t+T-1]$, respectively. The root of the tree corresponds to the state of the terminal at the beginning of each window of length T. The tree is made up of T "levels", where the decisions at each level depend on the actions taken at the previous ones. The extraction of random sequences corresponds to the exploration of paths in the tree

Algorithm 1 Solution of Problem RHBA-T

Inputs: t, T, G, $s_i(t)$ for all i, $Q(t)$
$J^* \leftarrow \infty$; $g^* \leftarrow 0$ *// initialization of the optimal solution*
for g **from** 1 **to** G **do** *// loop over the G random extractions*
  $J^{(g)} \leftarrow 0$ *// initialization of the cost of the g-th random sequence*
  $\tilde{s}_i(t) \leftarrow s_i(t)$ for all i; $\tilde{Q}(t) \leftarrow Q(t)$ *// initialization of state variables for receding horizon*
  for k **from** t **to** $t+T-1$ **do** *// loop over the time horizon*
    $\tilde{B}(k) \leftarrow$ berths that are free, i.e., such that $\tilde{s}_i(k) \leq 0$
    $a_i^{(g)}(k) \leftarrow 0$ for all i *// initialization of the g-th random sequence*
    if length$(\tilde{B}(k)) > 0$ and length$(\tilde{Q}(k)) > 0$ **then** *// there are free berths and ready ships*
      $(i, j) \leftarrow$ random pair extracted from the set $\tilde{B}(k) \times [\tilde{Q}(k) \cup \{0\}]$
      $a_i^{(g)}(k) \leftarrow j$ *// random decision*
    end if
    $\tilde{s}_i(k+1)$, $\tilde{Q}(k+1) \leftarrow$ update with the state equation (1) and decisions $a_i^{(g)}(k)$
    $J^{(g)} \leftarrow J^{(g)} + c(k)$ *// update of the intermediate cost*
    if $J^{(g)} > J^*$ **then** *// pruning: no good solution can be found with the g-th sequence*
      $J^{(g)} \leftarrow \infty$; break
    end if
  end for
  if $J^{(g)} \leq J^*$ **then** *// update the current optimal solution*
    $J^* \leftarrow J^{(g)}$; $g^* \leftarrow g$
  end if
end for
$\underline{a}^*(t), \ldots, \underline{a}^*(t+T-1) \leftarrow \underline{a}^{(g^*)}(t), \ldots, \underline{a}^{(g^*)}(t+T-1)$ *// retrieve the optimal sequence*
Output: $\underline{a}^*(t)$ *// only the first action is retained and applied*

from the root node up to one of the leaves (see the dashed paths in Fig. 2). Each node l of the k-th level, $k = 1, \ldots, T$, can be labeled with an intermediate cost $\tilde{J}_{l,k}$ given by the sum of the term $c(t)$ in (4) from t to $t+k-1$. Such costs always increase as we approach the leaves, and the intermediate cost of a leaf equals the overall cost in (5) associated with a certain random sequence of decisions. To reduce the computational burden, we adopt a pruning mechanism that consists in aborting the generation of a random sequence if the intermediate cost at a certain node is greater than the current minimum overall cost (see the green, dashed path in Fig. 2), which is iteratively updated starting from an initial value equal to $+\infty$. In fact, since the costs always increase as we approach the leaves, it is useless to proceed generating additional random actions in that sequence, as a greater cost with respect to the current minimum one will be obtained. The pseudo-code of the considered approximate solution technique is reported in Algorithm 1.

4 Simulation Results

In this section, we report the results of preliminary simulations to verify the effectiveness of the proposed approach. We focused on three scenarios characterized by different numbers of berths, served ships, and traffic intensity. All the tests were

done in Matlab on a PC with a 3.7 GHz Intel i7 CPU and 16 GB of RAM. In all the cases, we compared the results with a method that consists in berthing, among all the ships that are ready at time t, those that are waiting from the larger amount of time in the available berth that guarantees the lowest handling time. This technique, in contrast to the proposed RHBA, has no predictive flavor, and therefore we will refer to it as greedy (GRD) in the following. As a further comparison, we implemented an approach based on [5], where a fixed amount of 7 ships was scheduled at each time step. An exhaustive search enumerating all the possible scheduling combinations of such vessels was considered, including also those not yet arrived at the terminal, and the combination guaranteeing the lowest cost as in (5) was selected as the optimal one. We refer to this approach as predictive exhaustive search (PES).

Scenario A We focused on a terminal with 3 berths and a total of 20 ships to be served in an interval of 1 week. Since the size of the problem is small, an optimal scheduling in advance of vessels obtained by solving a classic mixed-integer formulation can be found and used as reference solution. Specifically, we adopted the approach of [9] and denoted it as MIF (mixed-integer formulation). Two levels of traffic intensities were considered, i.e., ship arrivals were modeled via Poisson distributions with rates 0.2 and 0.25 for "low" and "high" traffic, respectively. For the sake of simplicity, only one new arrival per time step was considered.

Scenario B We focused on a terminal with 5 berths and a total of 100 ships to be served in an interval of 3 weeks. The size of the problem makes it very difficult the use of exact mixed-integer formulations to find an optimal solution. As in Scenario A, two levels of traffic intensities were considered, i.e., "low" and "high", with ship arrivals modeled through Poisson distributions with rates 0.2 and 0.25, respectively.

Scenario C We focused on a terminal with 10 berths and a total of 100 ships to be served in an interval of 1 week. Due to the large number of berths and vessels, it is hard to find an optimal solution with the MIF approach. As in the previous cases, two levels of traffic intensities were considered, with ship arrivals modeled through Poisson distributions with rates 1.5 and 2 (increased values with respect to the previous scenarios were used to balance the larger number of berths).

In all the scenarios, a sampling time of 1 h was considered, and the handling times of ships and the initial remaining service time of berths were randomly generated based on uniform distributions in the ranges $[10, 45]$ h and $[0, 45]$ h, respectively, with a rounding to the nearest integer. The initial queue $Q(0)$ of vessels waiting to be served was fixed equal to the empty set. To have statistically-significant results, we repeated 100 times the simulations with different realizations of the arrivals. The RHBA was applied with three horizon lengths, i.e., $T = 10$ h, 30 h, 50 h, and $G = 1000$ random decision extractions. The value of G was chosen via a trial and error procedure: we started with a small value and increased it until no noticeable improvement in the cost of (5) could be observed.

The boxplots of the mean total times of ships (defined as the sum of the waiting and handling times) over the 100 simulations are reported in Fig. 3. Table 1 reports the average of the waiting and total times of vessels over the 100 simulation runs,

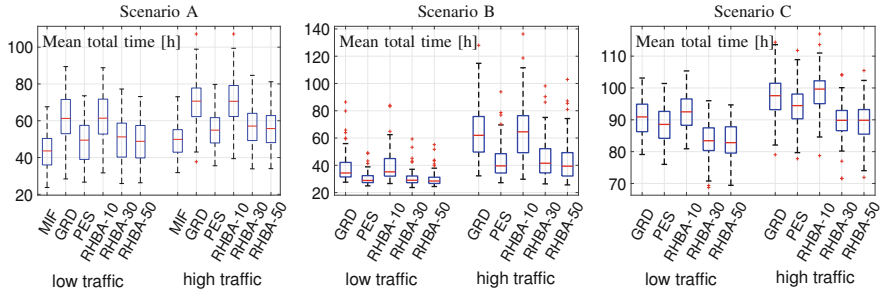


Fig. 3 Boxplots of the mean total time of ships over 100 simulation runs

together with the average computational effort required to find the berth allocation for all the vessels to be served. As expected, the MIF approach provides the best results in Scenario A, at the price of the highest computational burden. The proposed RHBA for $T = 50$ guarantees satisfactory results that represent a good tradeoff between accuracy and required effort, both in low and high traffic conditions. In particular, the RHBA can be used also for large instances of the problem, while the MIF is limited to the low-dimensional case of Scenario A. The decay in the performance is about 10% on the average, but a saving of 2 orders of magnitude in the computational time is guaranteed. As compared to the GRD approach, much lower waiting and total times of ships are obtained by the RHBA in all the scenarios. In particular, a reduction of about 20% of the average times of vessels is experienced. Clearly, a higher computational effort is needed that increases with the length of the horizon T. However, in the worst case the time needed to find a solution is about 5 min, which enables the application of the RHBA for real time planning. Also the comparison with the PES method based on [5] is successful, as lower waiting and total times of ships can be obtained in all the cases. The main advantage of the proposed approach lies in limiting the choice of the vessels entering the terminal to those that are waiting instead of scheduling a fixed number of ships like in PES. The result is a smaller search space, which can be efficiently explored through random sampling rather than with exhaustive search. As a consequence, longer horizons can be considered, with better performance and a much lower required computational effort. In particular, the advantages of the RHBA increase if the number of berths grows. In this case, having longer horizons is very important to reduce waiting and total times of vessels, while the proposed random search procedure to find a solution is very efficient as compared to exhaustive search. In fact, in Scenario C the RHBA guarantees a 6% reduction on the average waiting and total times of vessels and a saving of 1 order of magnitude in the required computational effort.

Table 1 Summary of the simulation results averaged over 100 simulation runs

	Traffic		MIF	GRD	PES	RHBA-10	RHBA-30	RHBA-50
A	Low	Mean waiting time [h]	22.19	36.95	27.27	37.67	27.97	27.05
		Mean total time [h]	43.85	61.46	49.21	62.25	50.08	48.80
		Simulation time [s]	$2.81 \cdot 10^{3}$	$5.15 \cdot 10^{-3}$	$6.04 \cdot 10^{2}$	$1.77 \cdot 10^{1}$	$3.09 \cdot 10^{1}$	$4.21 \cdot 10^{1}$
	High	Mean waiting time [h]	28.18	45.98	34.65	46.18	35.33	34.20
		Mean total time [h]	49.72	70.37	56.22	70.56	57.22	55.78
		Simulation time [s]	$1.84 \cdot 10^{3}$	$5.15 \cdot 10^{-3}$	$6.15 \cdot 10^{2}$	$1.81 \cdot 10^{1}$	$3.16 \cdot 10^{1}$	$4.25 \cdot 10^{1}$
B	Low	Mean waiting time [h]	–	13.74	7.87	14.79	7.46	6.69
		Mean total time [h]	–	38.47	31.24	39.66	30.68	29.68
		Simulation time [s]	–	$1.84 \cdot 10^{-2}$	$2.09 \cdot 10^{3}$	$8.72 \cdot 10^{1}$	$1.63 \cdot 10^{2}$	$2.43 \cdot 10^{2}$
	High	Mean waiting time [h]	–	40.20	23.04	40.41	22.41	20.49
		Mean total time [h]	–	64.96	45.58	65.07	44.83	42.63
		Simulation time [s]	–	$1.35 \cdot 10^{-2}$	$1.77 \cdot 10^{3}$	$1.09 \cdot 10^{2}$	$2.12 \cdot 10^{2}$	$3.16 \cdot 10^{2}$
C	Low	Mean waiting time [h]	–	67.16	66.45	68.73	61.12	60.89
		Mean total time [h]	–	90.96	89.63	92.59	83.41	83.22
		Simulation time [s]	–	$1.18 \cdot 10^{-2}$	$1.37 \cdot 10^{3}$	$4.82 \cdot 10^{1}$	$9.77 \cdot 10^{1}$	$1.43 \cdot 10^{2}$
	High	Mean waiting time [h]	–	73.53	72.65	74.87	67.43	67.51
		Mean total time [h]	–	97.35	95.83	98.71	89.71	89.86
		Simulation time [s]	–	$9.02 \cdot 10^{-3}$	$1.45 \cdot 10^{3}$	$4.74 \cdot 10^{1}$	$9.55 \cdot 10^{1}$	$1.32 \cdot 10^{2}$

5 Conclusions and Outline of Future Works

An approach based on the receding horizon paradigm has been proposed to face berth allocation that consists in solving a sequence of optimization problems to choose, at each time step, the vessels that enter the terminal among those waiting to be served together with the corresponding berthing positions, exploiting future predictions within a forward time window of a certain length. An approximate solution method based on random search has guaranteed satisfactory results if compared to alternative approaches, as a tradeoff between accuracy and required effort.

Future works include the extension to simultaneous quay crane and berth allocation, the account for service priorities and time-dependent availability of berths and ships, and a comparison with state-of-the-art heuristics for scheduling.

References

1. Alessandri, A., Cervellera, C., Cuneo, M., Gaggero, M., Soncin, G.: Management of logistics operations in intermodal terminals by using dynamic modelling and nonlinear programming. Maritime Econ. Logist. **11**(1), 58–76 (2009)
2. Alessandri, A., Cervellera, C., Gaggero, M.: Predictive control of container flows in maritime intermodal terminals. IEEE Trans. Control Syst. Technol. **21**(4), 1423–1431 (2013)
3. Bierwirth, C., Meisel, F.: A survey of berth allocation and quay crane scheduling problems in container terminals. Eur. J. Oper. Res. **202**(3), 615–627 (2010)
4. Bierwirth, C., Meisel, F.: A follow-up survey of berth allocation and quay crane scheduling problems in container terminals. Eur. J. Oper. Res. **244**(3), 675–689 (2015)
5. Cahyono, R.T., Flonk, E.J., Jayawardhana, B.: Dynamic berth and quay crane allocation for multiple berth positions and quay cranes. Proc. Eur. Control Conf. 3262–3267 (2015). https://doi.org/10.1109/ECC.2015.7331037.
6. Dulebenets, M.A.: Application of evolutionary computation for berth scheduling at marine container terminals: parameter tuning versus parameter control. IEEE Trans. Intell. Transport. Syst. **19**(1), 25–37 (2018)
7. Fang, K.T., Wang, Y.: Number-Theoretic Methods in Statistics. Chapman & Hall, London (1994)
8. Giallombardo, G., Moccia, L., Salani, M., Vacca, I.: Modeling and solving the tactical berth allocation problem. Transport. Res. B-Meth. **44**(2), 232–245 (2010)
9. Hansen, P., Oguz, C.: A note on formulations of the static and dynamic berth allocation problems. Les Cahiers du Gerad **30**, 1–20 (2003)
10. Imai, A., Nishimura, E., Papadimitriou, S.: Berth allocation with service priority. Transport. Res. B-Meth. **37**(5), 437–457 (2003)
11. Lalla-Ruiz, E., Voss, S., Exposito-Izquierdo, C., Melian-Batista, B., Moreno-Vega, J.M.: A POPMUSIC-based approach for the berth allocation problem under time-dependent limitations. Ann. Oper. Res. **253**(2), 871–897 (2017)
12. Mayne, D.Q., Rawlings, J.B., Rao, C.V., Scokaert, P.O.M.: Constrained model predictive control: stability and optimality. Automatica **36**(6), 789–814 (2000)
13. Nishi, T., Okura, T., Lalla-Ruiz, E., Voss, S.: A dynamic programming-based matheuristic for the dynamic berth allocation problem. Ann. Oper. Res. 1–20 (2017). https://doi.org/10.1007/s10479-017-2715-9

14. Umang, N., Bierlaire, M., Vacca, I.: Exact and heuristic methods to solve the berth allocation problem in bulk ports. Transport. Res. E-Log. **54**, 14–31 (2013)
15. Umang, N., Bierlaire, M., Erera, A.L.: Real-time management of berth allocation with stochastic arrival and handling times. J. Scheduling **20**(1), 67–83 (2017)
16. Xin, J., Negenborn, R., Lodewijks, G.: Event-driven receding horizon control for energy-efficient container handling. Control Eng. Pract. **39**(6), 45–55 (2015)
17. Zhen, L.: Tactical berth allocation under uncertainty. Eur. J. Oper. Res. **247**(3), 928–944 (2015)

Integrating Ship Movement Scheduling and Tug Assignment Within a Canal Harbor

Giacomo di Tollo, Raffaele Pesenti, and Matteo Petris

Abstract In this paper we address the in-port ship scheduling and tug assignment problem. This problem aims to determine a schedule of ship movements, and their escorting tugs, within a canal harbor. We formulate the problem as a Boolean satisfiability problem. In particular, we deal with canal-harbors, as this kind of harbors present strict constraints, e.g., on safety distance. We consider the Port of Venice, a medium size Italian harbor, as a case study.

Keywords Ship scheduling · Tug assignment · Locomotive scheduling problem · Boolean satisfiability

1 Introduction

This work introduces the in-Port ship Scheduling and tug Assignment Problem (PSAP), whose aim is to determine simultaneously a schedule of ship movements, and their escorting tugs, within a canal harbor. This work was motivated by a collaboration with the Port Authority of the Port of Venice in Italy. The Port of Venice is a medium Italian harbor located in the Venetian Lagoon. The lagoon shallow waters impose that ships sail only along narrow canals that are constantly dredged. In addition, the historical value and structural fragility of the city of Venice make imperative that for no reason a ship strands on one of the lagoon small islands. In this canal harbor, navigation undergoes to strict regulations, e.g., ships are required to maintain a safety distance and are forbidden to pass or overtake each other. Also, ships must be escorted by one or more tugs throughout the navigation

G. di Tollo
Department of Economics, Ca' Foscari University of Venice, Venice, Italy
e-mail: giacomo.ditollo@unive.it

R. Pesenti · M. Petris (✉)
Department of Management, Ca' Foscari University of Venice, Venice, Italy
e-mail: pesenti@unive.it; matteo.petris@unive.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_2

if they exceed a certain tonnage or, for any reason, are restricted in their ability to maneuver on their own.

The PSAP is of interest for port authorities and port operators, who have to manage daily the ship movements in the harbor, and for ship companies, which are interested in performing the in-port activities according the planned timing.

The PSAP falls within the rich literature on in-port or canal ship scheduling and routing problems, such as: the berth allocation and quay crane scheduling problems [1, 4, 6], the problem of coordinating the movements of tankers that may require to visit different berths [13], the channel-berth coordination problems [15, 16].

Our problem generalizes the problem of scheduling ship movements within a canal harbor (PSP) introduced in [9], there the tug assignment is not considered. In particular, Pellegrini et al. [9] proposes a *Mixed Integer Linear Programming* model that is motivated by the analogies between the PSP and the problem of scheduling trains on a single track railway network highlighted in [3] and it is inspired by the RECIFE-MILP [8] model for the railway scheduling problem.

The problem of assigning tugs to a set of pre-scheduled ship movements shows similarities with another railway management problem, the Locomotive Scheduling Problem (LSP). According to [12] the aim of the LSP is to assign locomotives to a set of pre-scheduled trains in order to deliver the trains in the schedule at minimum cost. The LSP has been widely studied through the years hence many variants has been proposed: an exhaustive survey on the subject is [10]. Although the similarities, some differences arise as well: locomotives cannot travel freely if tracks are occupied by trains whereas tugs are allowed to sail freely in canals occupied by ships except in some extremely narrow canals; a train pulled by a locomotive may stop at intermediate points along its route in a station whereas a ship must be escorted to its destination berth without intermediate stops.

The sequential approach among train scheduling and locomotives assignment presents some drawbacks as pointed out in [2]. For this reason, in recent years, interest has start growing for integrated scheduling and assignment problems. An example of this type of problems is proposed in [14], where the authors' aim is to determine simultaneously a train timetabling and an assignment of the locomotives to the trains.

The PSAP is also an integrated scheduling and assignment problem. In contrast with [14], PSAP does not allow ship movements to be canceled and permit the assignment of more than one tug per ship.

In this paper, we formulate the PSAP as a Boolean satisfiability problem (SAT) [5] in the assumption that we can considered the time discretized. This approach allows us to manage the real world PSAP instances of the Port of Venice. It differs both from the RECIFE-MILP formulation for PSP in [9] and the minimum cost multi-commodity network flow formulation with incompatible arcs and flow restriction for the locomotive assignment problem in [14]. We solve the PSAP by means of *GASAT*, a Genetic Local Search Algorithm for the Satisfiability Problem proposed in [7], whose performances have been discussed and enhanced by the reactive search strategies introduced in [11].

The reminder of the paper is organized as follows. In Sect. 2, we formally define the problem. In Sect. 3 we present a formulation of the PSAP as a Boolean satisfiability problem. In Sect. 4, we discuss a case study. Finally, we draw some conclusions in Sect. 5.

2 Problem Statement

In this section we provide a formal statement of the PSAP. To this end, first we need to introduce some notation and basic assumptions.

We represent a canal harbor as network $\mathcal{G} = (V, E)$ of waterways. Specifically, each vertex $i \in V$ corresponds to a *navigation point*, i.e., a point of interest for the navigation of ships. Each edge $e = (i, j) \in E$ corresponds to a canal segment that joins the navigation points i and j and includes no other navigation point in between. A navigation point can be:

- a *berth* or a *roadstead*;
- a *connection point* between two waterways or between a berth and a waterway;
- an other type of *relevant point* where a particular operation, such as a turn or the beginning or the end of a tug service, is carried out or some harbormaster constraint holds, e.g., it delimits an area where particular speed limits are imposed.

We denote by M the set of ship movements which should start within a time interval T. For all movements $m \in M$ we denote by: (1) t_m the starting time of movement m, (2) r_m the route of movement m, i.e. the path in $\mathcal{G}$ connecting the departure berth and arrival berth of m (with abuse of terminology we consider the roadsteads as berths); (3) i_m, j_m the departure and the arrival berth of m, respectively; (4) $T_m = [init_m, dead_m]$ the departure time window of movement m, where $init_m$ and $dead_m$ are respectively the earliest and latest starting time of movement m; (5) s_m the ship involved in movement m; (6) st_m the time needed to complete movement m; (7) N_m the number of tugs required to perform movement m; (8) Ω_m the set of tugs compatible with the ship involved in movement m.

For all movements $m \in M$ such that $N_m > 0$ we denote by pu_m and do_m the service time for a tug to respectively pick up and drop off ship s_m.

For all movements $m \in M$ and for all $k = 1, \ldots, N_m$ we denote by f_m^k and l_m^k respectively the initial and the final navigation points in r_m of service of the k-th tug on movement m.

For all pairs of movements $m, m' \in M$ we denote by: (1) $\mathbb{1}(m, m')$ the indicator function which takes value 1 if m and m' are operated by the same ship, 0 otherwise; (2) $ms_{m,m'}$, $ns_{m,m'}$ the minimum, respectively maximum, separation time between the end of a movement m and the start of the subsequent movement m' of the same ship, i.e., when $\mathbb{1}(m, m') = 1$; (3) $mh_{m,m'}$ the minimum headway time between m and m' starting times required to guarantee that s_m and $s_{m'}$ will maintain a minimal safety distance during their movements. In particular, $mh_{m,m'}$ is assumed equal

to infinity, if movement m should not be scheduled before movement m'. For all movements m and navigation points i we denote by $st_{i,m}$ the sailing time of s_m from i_m to i.

Finally, we denote Ω the set of the available tugs. For all tugs $\omega \in \Omega$ and for all pairs of navigation points $i, j \in V$, we say that ω *is traveling light from* i *to* j if it is sailing from i to j without escorting any ships; we suppose that the tugs sail at the same constant speed in the canal harbor and denote by $lt_{i,j}$ the sailing time of a tug traveling light from i to j. For all tugs $\omega \in \Omega$ we denote by $M_\omega \subseteq M$ the subset of M containing the movements that can be operated by ω.

We understand that the following assumptions on ship and tug movements hold.

Assumptions 1 (Ship Movements) Each movement $m \in M$:

1.1 has both its route r_m and sailing time a-priori fixed. In particular, a ship cannot stop along its route once it has started moving.
1.2 has its starting time t_m to be scheduled within a given time window T_m.
1.3 has to respect a separation time $ns_{m,m'}$ between its own starting time and the starting time of other possible movements m' performed by the same ship s_m.
1.4 has its starting time t_m to maintain an headway $mh_{m,m'}$ with the starting time of any other movements m' whose route may interfere with its own.
1.5 should be escorted by N_m compatible tugs, if $N_m > 0$.

We remark that Assumptions 1.2 and 1.3 are necessary to guarantee that in-port operations, such as loading and unloading, can be performed. Assumption 1.4 is necessary to guarantee that each ship maintains a minimal safety distance from other ships while sailing along the canals. Indeed, given Assumption 1.1 and the results in [9], a time headway between the starting times of the movements of two ships guarantees that the two ships never gets too close during their movements.

Assumptions 2 (Tug Movements) Each tug $\omega \in \Omega$:

2.1 cannot serve more than one movement simultaneously.
2.2 can cross or overtake a ship sailing in the same canal.

We are now ready to formally state the PSAP.

Problem 1 (In-Port Ship Scheduling and Tug Assignment Problem) Let a set of movements M within canal harbor $\mathcal{G}$ and a set of tugs Ω be given. Under Assumptions 1 and 2, determine a feasible starting time $t_m \in T_m$, for each movement $m \in M$, and to m the required N_m tugs in Ω_m.

Hereinafter, we denote a PSAP instance by $\pi(M, \Omega, \mathcal{G})$. In the next section, we formulate the above problem as a Boolean satisfiability problem. In particular, we assume the time frame T as discretized, i.e., T is a set of successive time instants. In the practice of the Port of Venice, T covers 24 h and can be discretized with a step equal to 5 min. The length of the routes to sail and the inevitable uncertainties afflicting navigation make a higher time resolution meaningless, at least in the planning phase of movements.

3 Problem Formulation

In this section, initially, we recall the main concepts concerning Boolean satisfiability. Then, we show how to express the conditions and assumptions that define the PSAP in terms of logical clauses.

A *propositional formula* ϕ is a formula defined over Boolean variables that take values in the set $\{true, false\}$. We say that ϕ is *satisfiable* if there exists a truth assignment to the variables occurring in ϕ which makes it evaluate $true$, otherwise we say that ϕ is *unsatisfiable*.

A propositional formula ϕ is said to be in *Conjunctive Normal Form* (CNF) if it is a conjunction of *clauses*, i.e., $\phi = c_1 \wedge c_2 \wedge \cdots \wedge c_m$, where a clause c_i is a disjunction of literals, i.e., $c_i = l_{i,1} \vee l_{i,2} \vee \cdots \vee l_{i,m_i}$, and a *literal* is either a variable or its negation.

Accordingly, the *Boolean satisfiability problem* (SAT) [5] can be stated as: given a CNF propositional formula ϕ, decide if it is satisfiable.

From now on we say that a SAT problem has a *feasible solution* if it exists a truth assignment to the variables occurring in ϕ which makes it evaluate $true$.

Preliminarily, we prove the following lemma which guarantees that there is no loss of generality in assuming that each movement requires to be escorted by at most one tug along all its route. To this end, let us first define two PSAP instances as *equivalent* if each feasible solution of one instance, in term of movement schedule and tug assignments, allows to derive a feasible solution of the other instance in $O(|M||\Omega|)$ operations.

Lemma 1 *A PSAP instance $\pi_1(M_1, \Omega, \mathcal{G})$ can always be reduced to an equivalent PSAP instance $\pi_2(M_2 \cup D, \Omega, \mathcal{G})$, where $|D| \leq \sum_{m \in M_1} N_m$ and each movement $m \in M_2 \cup D$ requires to be escorted by at most one tug along all its route.*

Proof We prove the Lemma constructively by indicating how to generate the movement set $M_2 \cup D$ of instance π_2.

The set M_2 includes the same movements of M_1. However, each movement $m \in M_2$ is stripped of all its tug services, with the only exception of the first service that escorts the movement along all its route r_m.

The set D includes dummy movements to handle the situations in which a movement $m \in M_1$ either requires more than one tug, $N_m \geq 2$, or requires tugs only for a portion of its route, $f_m^k \neq i_m$ or $l_m^k \neq j_m$, for some $k = 1, \ldots, N_m$.

Consider the generic k-th tug service requested by movement $m \in M_1$ of π_1. Assume that either $k \geq 2$ or $f_m^k \neq i_m$ or $l_m^k \neq j_m$. In this situation, we generate a dummy movement $\bar{m} \in D$ that shares the same route of m from the navigation point f_m^k to the navigation point l_m^k and such that

- $T_{\bar{m}} = \{init_m + st_{f_m^k,m}, \ldots, dead_m + st_{f_m^k,m}\}$;
- $ms_{m,\bar{m}} = ns_{m,\bar{m}} = mh_{m,\bar{m}} = -mh_{\bar{m},m} = st_{f_m^k,m}$;
- $ser_{\bar{m}} = st_{l_m^k,m} - st_{f_m^k,m}$, i.e., the service time of the required tug on m;
- $i_{\bar{m}} = f_m^k$ and $j_{\bar{m}} = l_m^k$;

- $N_{\bar{m}} = 1$;
- $\Omega_{\bar{m}} = \Omega_m$;
- $pu_{\bar{m}} = pu_m$ and $do_{\bar{m}} = do_m$.

We observe that the first four of the above conditions impose that the movement $\bar{m}$ occurs parallel to m from f_m^k to l_m^k. The remaining conditions impose that movement $\bar{m}$ uses a tug compatible with m.

Let $D(m) \subseteq D$ be the subset of dummy movement induced by a movement $m \in M_1$. By construction $|D(m)| \leq N_m$, hence $|D| \leq \sum_{m \in M_1} N_m$. Finally, consider a feasible solution for π_2, the corresponding feasible solution for π_1 is obtained by scheduling each movement $m \in M_1$ as a the corresponding movement in M_2 and by assigning to m also the tugs of the movements in $D(m)$. Inverse argument allows to derive a feasible solution for π_2 given a feasible solution for π_1. □

Hereinafter we consider instances where each movement requires to be escorted by at most one tug along all its route.

We are now ready to introduce the clauses that define the PSAP SAT formulation.

Given an instance $\pi(M, \Omega, \mathcal{G})$, we consider the following Boolean variables. For all movements $m \in M$ and all time instants $t \in T_m$:

$$x_{m,t} = \begin{cases} true & \text{if } m \text{ is scheduled to begin at time } t, \\ false & \text{otherwise;} \end{cases}$$

for all movements $m \in M$ such that $N_m > 0$ and all tugs $\omega \in \Omega_m$:

$$z_{m,\omega} = \begin{cases} true & \text{if } m \text{ is assigned with tug } \omega, \\ false & \text{otherwise;} \end{cases}$$

finally, for all pairs of movements $m, m' \in M$ such that $N_m, N_{m'} > 0$ and $\Omega_m \cap \Omega_{m'} \neq \emptyset$:

$$y_{m,m'} = \begin{cases} true & \text{if } m \text{ and } m' \text{ are escorted by the same tug,} \\ false & \text{otherwise.} \end{cases}$$

Then, we introduce the clauses in CNF whose satisfaction guarantees that Assumptions 1 and 2 hold true.

C1 Movement operation clauses:

$$\bigvee_{t \in T_m} x_{m,t} \qquad \forall m \in M \qquad (1)$$

$$\neg x_{m,t} \vee \neg x_{m,t'} \qquad \forall m \in M, \forall t, t' \in T_m : t \neq t' \qquad (2)$$

$$\bigvee_{\omega \in \Omega_m} z_{m,\omega} \qquad \forall m \in M : N_m > 0 \tag{3}$$

$$\neg z_{m,\omega} \vee \neg z_{m,\omega'} \qquad \forall m \in M : N_m > 0, \forall \omega, \omega' \in \Omega_m : \omega \neq \omega' \tag{4}$$

Clauses (1) and (2) require that $m \in M$ are scheduled at one and only one time instant $t \in T_m$; clauses (3) and (4) require, if $N_m > 0$, that m is assigned with one and only one compatible tugs belonging to Ω_m.

C2 Separation time clauses:

$$\neg x_{m,t} \bigvee_{t' \in \{t+ms_{m,m'},\ldots,t+ns_{m,m'}\}} x_{m',t'} \qquad \forall m, m' \in M : \mathbb{1}(m, m') = 1, \forall t \in T_m \tag{5}$$

These clauses require that schedules of two movements m and m', with $m < m'$, operated by the same ship have to be separated by an interval of length between $ms_{m,m'}$ and $ns_{m,m'}$

C3 Headway clauses:

$$\neg x_{m,t} \bigvee_{t' \in H_{m,m'}(t) \cap T_{m'}} x_{m',t'} \tag{6}$$

$$\forall m, m' \in M : m < m', \mathbb{1}(m, m') = 0, \forall t \in T_m,$$

where $H_{m,m'}(t) = \{init_{m'}, \ldots, t - mh_{m',m}\} \cup \{t + mh_{m,m'}, \ldots, dead_{m'}\}$.

These clauses require that schedules of two movements m and m' operated by different ships have to be separated by a minimal headway $mh_{m,m'}$ if m is scheduled before m' or by $mh_{m',m}$ otherwise.

C4 Tug usage clauses:

$$\neg z_{m,\omega} \vee \neg z_{m',\omega} \vee y_{m,m'} \qquad \forall \omega \in \Omega, \forall m, m' \in M_\omega : m < m' \tag{7}$$

$$\neg x_{m,t} \vee \neg y_{m,m'} \bigvee_{t' \in U_{m,m'}(t) \cap T_{m'}} x_{m',t'} \quad \forall m, m' \in M : m < m', \Omega_m \cap \Omega_{m'} \neq \emptyset, \forall t \in T_m, \tag{8}$$

where $U_{m,m'}(t) = \{init_{m'}, \ldots, t - ut_{m',m}\} \cup \{t + ut_{m,m'}, \ldots, dead_{m'}\}$ with $ut_{m',m} = ser_{m'} + do_{m'} + lt_{j_{m'},i_m} + pu_m$ and $ut_{m,m'} = ser_m + do_m + lt_{j_m,i_{m'}} + pu_{m'}$.

Clauses (7) require that $y_{m,m'}$ assumes value $true$, if m and m' share a same tug.

Clauses (8) require schedules of two movements m and m' served by a same tug have to be separated so that the tug can complete its service with one movement before starting its service with the other movement. Indeed, the

terms $ut_{m',m}$ and $ut_{m,m'}$ include the service and drop off time of the tug on the first movement, the sailing time of the tug from the first to the second service, finally, the pick up time. We observe that if movement m is scheduled at t and there are no time instants $t' \in T_{m'}$ before $t - ut_{m',m}$ or after $t + ut_{m,m'}$, then the clause imposes that m and m' are assigned with different tugs.

4 Computational Experiments

The Port of Venice is a medium-size Italian port (about 3500 calls in 2018, for a total 81,000,000 gross tonnage, 632,000 containers—in TEU, and 1,500,000 passengers) situated in the Venetian Lagoon. The access to the port is guaranteed by two inlets which connect the roadsteads respectively with the passenger terminals in Marittima (old town centre) and with the commercial terminals in Marghera (mainland).

The topological layout of the port can be described by a tree with 281 vertexes, among which we distinguish 163 active berths, 2 roadsteads, 97 junction and 19 other navigation points.

The computational experiments reported in this section consider the movements occurring in 10 among the most congested days in the period 2011–2016. For all $i = 1, \ldots, 10$, we denote by $\pi_i := \pi_i(M_i, \Omega, \mathcal{G})$ the i-th instance and by $\sigma_i := \sigma_i(M_i, \Omega, \mathcal{G})$ its solution. In all the instances the set Ω includes all the 13 tugs which operate in the Port of Venice.

We use the solver *GASAT*, presented in [7] with the reactive search strategies introduced in [11] to solve the instances. We run the experiments on a laptop PC Dell XPS 15 9560, with an Intel Core i7-7700HQ at 2.80GHz and 16.0GB of installed RAM.

For each instance, we report in Table 1 the number of movements to be scheduled, the number of variables and clauses characterizing the SAT formulation and the computational time needed to solve the instance with the solver *GASAT*. First we

Table 1 Computational results

Instance	# Movements	# Variables	# Clauses	CPU time (s)
π_1	69	6303	148,305	1.52
π_2	70	6450	153,092	1.62
π_3	69	6315	145,070	1.55
π_4	58	5027	120,176	1.17
π_5	59	5120	118,441	1.03
π_6	61	5309	115,565	1.02
π_7	68	6608	171,672	5.59
π_8	57	4910	120,362	1.11
π_9	65	5780	125,387	1.09
π_{10}	54	4523	101,921	0.94

report that the width of the movement time windows is an important parameter to obtain feasible solutions and to maintain the size of the instances reasonable. For all instances π_i, $i \neq 7$, we found a feasible solution by setting the time windows as 4 h time windows centered in the rounding to the closest second hour of the movement expected departure times. Instance π_7 is a rather complicated instance which required an enlargement of the time windows width to 4.5 h.

All the considered instances were solved in less the 6 s, hence we decided to investigate the problem of finding the minimum movement time windows width which lead to a feasible solution. We do that by means of an iterative algorithm based on the bisection method. We initialize the first interval to $[0, \delta_i]$, where δ_i is the width of the movement time windows which leads to the feasible solution σ_i and width 0 leads to an unfeasible solution of π_i. We consider the midpoint of the interval as a new width, we build the new instance and we call the solver. If after 5 min of computation a solution is found we update the interval such that the two extremes correspond to width of the time windows which lead respectively to a feasible and an unfeasible solution. We stop the iteration if the length of the interval is less or equal to 10 min or if after 5 min of computation the solver has not found a solution. As an example, we ran the algorithm on instance π_1 and we found, in 4 iterations, that the width can be reduced from 4 to 2.5 h. The algorithm running time was 256.24 s, 254.82 s to generate the instances and 1.42 s to solve them with the solver *GASAT*.

5 Conclusions

We have introduced the in-Port ship Scheduling and tug Assignment Problem and proposed a SAT formulation for it. The PSAP presents may resemblances with single track train scheduling problems and locomotive scheduling problems and shares their NP-hard complexity. However, we have shown that the PSAP instances for a medium Italian port can be solved in a reasonable time, provided that tugs can pass or overtake ships that may encounter along their routes (Assumption 2.2).

The PSAP instances enjoy the same characteristics that make them solvable and are connected to the specificity of maritime transportation. They are: the possibility of considering time discretized with a step equal to 5 min. and the fact that a ship cannot stop along its route once it has started moving (Assumption 1.1). In train scheduling, one should consider a discretized time step no longer than 1 min. and take into account that a train may stop along its path. The assignment of tugs to movements presents some similarity to the assignment of locomotives to trains. However, in the case of train scheduling an assumption equivalent to Assumption 2.2 cannot hold since locomotives and trains share the same tracks. Indeed, also in the PASP case, if the canals in a harbor are so narrow that the Assumption 2.2 does not hold, a more complicated formulation is required and much more computational time is necessary to solve the problem instances, to the point that possibly heuristic approaches have to be considered and are current object of study by the authors of this paper.

Acknowledgements This study was partially funded by the "Smart PORt Terminals - SPORT" MIUR-PRIN project (grant number: 2015XAPRKF) financed by the Italian state. The study was partially developed within the *Centro Studi su Economia e Management della Portualità* of Università Ca' Foscari, Venezia.

References

1. Bierwirth, C., Meisel, F.: A survey of berth allocation and quay crane scheduling problems in container terminals. Eur. J. Oper. Res. **202**(3), 615–627 (2010)
2. Bussieck, M.R., Winter, T., Zimmermann, U.T.: Discrete optimization in public rail transport. Math. Program. **79**(1), 415–444 (1997)
3. Canestrelli, E., Corazza, M., De Nadai, G., Pesenti, R.: Managing the ship movements in the port of Venice. Netw. Spat. Econ. **17**(3), 861–887 (2017)
4. Carlo, H.J., Vis, I.F.A., Roodbergen, K.J.: Seaside operations in container terminals: literature overview, trends, and research directions. Flex. Serv. Manuf. J. **27**(2), 224–262 (2015)
5. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York (1979)
6. Gharehgozli, A.H., Roy, D., de Koster, R.: Sea container terminals: new technologies and or models. Maritime Econ. Logist. **18**(2), 103–140 (2016)
7. Lardeux, F., Saubion, F., Hao, J.K.: GASAT: A genetic local search algorithm for the satisfiability problem. Evol. Comput. **14**(2), 223–253 (2006)
8. Pellegrini, P., Marlière, G., Pesenti, R., Rodriguez, J.: Recife-milp: An effective milp-based heuristic for the real-time railway traffic management problem. IEEE Trans. Intell. Transp. Syst. **16**(5), 2609–2619 (2015)
9. Pellegrini, P., di Tollo, G., Pesenti, R.: Scheduling ships movements within a canal harbor. Soft Comput. (2018). https://doi.org/10.1007/s00500-018-3469-2
10. Piu, F., Speranza, M.G.: The locomotive assignment problem: a survey on optimization models. Int. Trans. Oper. Res. **21**(3), 327–352 (2014)
11. di Tollo, G., Lardeux, F., Maturana, J., Saubion, F.: An experimental study of adaptive control for evolutionary algorithms. Appl. Soft Comput. **35**, 359–372 (2015)
12. Vaidyanathan, B., Ahuja, R.K., Liu, J., Shughart, L.A.: Real-life locomotive planning: new formulations and computational results. Transp. Res. B Methodol. **42**(2), 147–168 (2008)
13. Wang, X., Arnesen, M.J., Fagerholt, K., Gjestvang, M., Thun, K.: A two-phase heuristic for an in-port ship routing problem with tank allocation. Comput. Oper. Res. **91**, 37–47 (2018)
14. Xu, X., Li, C.L., Xu, Z.: Integrated train timetabling and locomotive assignment. Transp. Res. B: Methodol. **117**, 573–593 (2018)
15. Zhang, X., Lin, J., Guo, Z., Liu, T.: Vessel transportation scheduling optimization based on channel-berth coordination. Ocean Eng. **112**, 145–152 (2016)
16. Zhen, L., Liang, Z., Zhuge, D., Lee, L.H., Chew, E.P.: Daily berth planning in a tidal port with channel flow control. Transp. Res. B Methodol. **106**, 193–217 (2017)

The Maximum Nearby Flow Problem

Gennaro Auricchio, Stefano Gualandi, and Marco Veneroni

Abstract We present a new Linear Programming model that formulates the problem of computing the Kantorovich-Wasserstein distance associated with a truncated ground distance. The key idea of our model is to consider only the quantity of mass that is transported to nearby points and to ignore the quantity of mass that should be transported between faraway pairs of locations. The proposed model has a number of variables that depends on the threshold value used in the definition of the set of nearby points. Using a small threshold value, we can obtain a significant speedup. We use our model to numerically evaluate the percentage gap between the true Wasserstein distance and the truncated Wasserstein distance, using a set of standard grey scale images.

Keywords Optimal transport · Wasserstein distance · Network simplex

1 Introduction

In Machine Learning, the computation of a measure of similarity (or dissimilarity) between pairs of objects is a crucial subproblem. For instance, in Clustering problems we want to find a partition of a given set of objects in different clusters, in such a way that the objects that belong to the same cluster are pairwise *similar* and those that are in different clusters are as *dissimilar* as possible [8, 15]. In Classification problems, in order to classify a new data point, the popular k-nearest-neighbour heuristics are based on the computation of the k nearest (already classified) points: *nearest* with respect to a given metric. While in literature several different metrics were proposed (e.g., see Chap. 11 in [11]), the use of Kantorovich-Wasserstein distances has recently proved to be a valuable option in several

G. Auricchio · S. Gualandi (✉) · M. Veneroni
Department of Mathematics, University of Pavia, Pavia, Italy
e-mail: gennaro.auricchio01@universitadipavia.it; stefano.gualandi@unipv.it; marco.veneroni@unipv.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_3

application domains, as for instance, in Computer Vision [19–21], Computational Statistic [16], Probability [5, 6], and Machine Learning [3, 10, 13, 24]. However, since computing Kantorovich-Wasserstein distances is computationally demanding and requires the solution of a standard Hitchcock-Koopmans transportation problem [12, 23], a number of researchers are focusing in designing new efficient algorithms that can well approximate the true distance in a short computational time. The work we present in this paper is along this line of research.

The optimization problem that yields the Kantorovich-Wasserstein distance can be solved with different numerical methods. Nowadays, the most popular methods are based on (1) the Sinkhorn's algorithm [1, 9, 25], which solves (approximatively) a regularized version of the basic optimal transport problem, and (2) Linear Programming-based algorithms [2, 4, 7, 17], which exactly solve the basic optimal transport problem by formulating and solving an equivalent uncapacitated minimum cost flow problem [14, 18]. One of the fastest heuristic algorithms used in the literature is the FastEMD algorithm, introduced in [19]. In their work, the authors studied the case when the ground distance between any pair of points is saturated (i.e, truncated) at some parameter t. Using truncated ground distances, it is possible to formulate the Hitchcock-Koopmans transportation problem on a reduced network, and, as a consequence, to significantly speed up the computation time. Indeed, since the ground distance is truncated, their approach provides only a lower bound to the true Kantorovich-Wasserstein distance.

In this paper, we propose a new model to compute the Kantorovich-Wasserstein distances with truncated ground distances, and we formally prove that our reformulation is correct. Our reformulation gives a maximization problem defined only over nearby points, which is different from the problem introduced in [19]. In addition, our computational results show numerically the tradeoff between the percentage gap and the runtime of using truncated Kantorovich-Wasserstein distances versus the exact distances.

The outline of this paper is as follows: Sect. 2 introduces the foundations of Optimal Transport and fixes the notation. Section 3 presents in Theorem 1 our main result, which is based on our new Linear Programming formulation of the truncated transportation problem. In Sect. 4, we report our computational results that show the benefits of using truncated distances in terms of computational speed versus solution quality.

2 Optimal Transport

Let G_d be a d-dimensional regular grid composed of n^d points, and let

$$c : G_d \times G_d \to [0, \infty),$$

be a cost function between points in G_d. We denote $c_{a,b} = c(a, b)$, where $a = (a_1, \ldots, a_d) \in G_d$ and $b = (b_1, \ldots, b_d) \in G_d$.

Given two probability measures $\mu, \nu : G_d \to [0, 1]$ such that $\sum_{a \in G_d} \mu_a = \sum_{b \in G_d} \nu_b = 1$, we define the set

$$\Pi_{\mu,\nu} := \left\{ \pi_{a,b} \geq 0 : \sum_{a \in G_d} \pi_{a,b} = \nu_b; \sum_{b \in G_d} \pi_{a,b} = \mu_a \right\}.$$

Any $\pi \in \Pi_{\mu,\nu}$ is called a transportation plan. We can then define the functional $\mathbb{T}_c : \Pi_{\mu,\nu} \to [0, \infty)$ as

$$\mathbb{T}_c(\pi) := \sum_{(a,b) \in G_d \times G_d} c_{a,b} \pi_{a,b}.$$

The Wasserstein distance between μ and ν associated to the cost function c is

$$W_c(\mu, \nu) := \inf\{\mathbb{T}_c(\pi) : \pi \in \Pi_{\mu,\nu}\}. \tag{1}$$

In this paper, we consider Wasserstein distances where the cost $c_{a,b}$ is induced by standard Minkowsky p norms, as for instance the squared Euclidean norm

$$c_{a,b} = \sum_{i=1}^{d} (a_i - b_i)^2. \tag{2}$$

Since we are considering d-dimensional regular grids composed of a finite number of n^d points, the inf in (1) is a minimum, which can be found by solving the following geometric Hitchcock-Koopmans transportation problem [12]:

$$\begin{aligned}
(\textbf{EMD}) \quad W_c(\mu, \nu) := \min \quad & \sum_{(a,b) \in G_d \times G_d} c_{a,b} \pi_{a,b} && && (3)\\
\text{s.t.} \quad & \sum_{a \in G_d} \pi_{a,b} = \nu_b, && \forall b \in G_d, && (4)\\
& \sum_{b \in G_d} \pi_{a,b} = \mu_a, && \forall a \in G_d, && (5)\\
& \pi_{a,b} \geq 0 && \forall (a, b) \in G_d \times G_d. && (6)
\end{aligned}$$

While this problem is polynomially solvable using uncapacitated min cost flow algorithms [14, 18], the size of instances appearing in Machine Learning applications makes the solution of this problem a very interesting computational challenge.

In the following section, we show how we can reduce the size of problem (3)–(6) by introducing t-truncated ground distances, in place of $c_{a,b}$), as defined in (2). In addition, we show how this problem can be reformulated as a maximization linear programming problem defined on a slightly different set of constraints.

3 The Maximum Nearby Flow Problem

In this section, we present a new formulation to the problem of computing truncated Wasserstein distances. The advantage with respect to [19] is that our formulation requires fewer variables. In the following, we introduce the basic definitions used in our model and we formally state our main result in Theorem 1.

Definition 1 Given a positive threshold $t \in \mathbb{R}^+$ we define the *t-truncated* cost function $c^{(t)}$ as

$$c^{(t)}_{a,b} := \min\{c_{a,b}, t\} \qquad \forall (a,b) \in G_d \times G_d.$$

Definition 2 Given a cost function c and a fixed parameter $t > 0$, we define the set of near points

$$N^{(t)}_c := \{(a,b) \in G_d \times G_d : c_{a,b} < t\}.$$

Similarly, we define the set of points near to a and b, respectively, as

$$O^{(t)}_c(a) := \{b \in G_d : c_{a,b} < t\},$$
$$I^{(t)}_c(b) := \{a \in G_d : c_{a,b} < t\}.$$

Indeed, we are implicitly considering a bipartite graph with two node partitions V_1 and V_2, where each of the two vertex sets has a node for each point of the grid G_d. The arc set corresponds to the set $N^{(t)}_c$, a subset of $V_1 \times V_2$. The set of outgoing arcs from a node of the first partition is $O^{(t)}_c$, while the set of incoming arcs to a node of the second partition is $I^{(t)}_c(b)$.

Definition 3 Given two probability measures μ and ν on G_d, the collection of positive values $\eta := \{\eta_{a,b}\}_{(a,b)\in N^{(t)}_c}$ is a **nearby flow** if

$$\sum_{a \in I^{(t)}_c(b)} \eta_{a,b} \le \nu_b, \quad \forall b \in G_d, \tag{7}$$

$$\sum_{b \in O^{(t)}_c(a)} \eta_{a,b} \le \mu_a, \quad \forall a \in G_d. \tag{8}$$

We denote by $\mathcal{N}_{\mu,\nu}$ be the set of all nearby flows between μ and ν.

Definition 4 The *nearby flow transport functional* $\mathbb{B}^{(t)}_c$ is defined as

$$\mathbb{B}^{(t)}_c(\eta) := \sum_{N^{(t)}_c} s_{a,b}\eta_{a,b}$$

where $s_{a,b} := (t - c_{a,b})$.

We are now ready to present our reformulation of the truncated Wasserstein distance. The maximization problem that appears in the following theorem is called the **Maximum Nearby Flow Problem** associated with the threshold value t.

Theorem 1 *Given a threshold value $t > 0$, a cost function c on $G_d \times G_d$, two probability measures μ and ν defined over G_d, the following relation holds*

$$W_{c^{(t)}}(\mu, \nu) := \inf_{\Pi_{\mu,\nu}} \mathbb{T}_{c^{(t)}}(\pi) = t - \max_{\mathcal{N}_{\mu,\nu}} \mathbb{B}_c^{(t)}(\eta).$$

Proof In order to prove the theorem, we first show how given a flow π we can get a feasible nearby flow η, and, later, we show the contrary, that is, how to get a feasible flow π given a nearby flow η.

Let us start with a given $\pi \in \Pi_{\mu,\nu}$. Then, we can define

$$\eta_{a,b} := \pi_{a,b,}, \qquad \forall (a, b) \in N_c^{(t)},$$

for which we can easily show that

$$\sum_{a \in I_c^{(t)}(b)} \eta_{a,b} = \sum_{a \in I_c^{(t)}(b)} \pi_{a,b} \leq \sum_{a \in G_d} \pi_{a,b} = \nu_b, \quad \forall b \in G_d,$$

$$\sum_{b \in O_c^{(t)}(b)} \eta_{a,b} = \sum_{b \in O_c^{(t)}(b)} \pi_{a,b} \leq \sum_{b \in G_d} \pi_{a,b} = \mu_a, \quad \forall a \in G_d.$$

Hence, we have that $\eta \in \mathcal{N}_{\mu,\nu}$.

By a simple computation, we get

$$\begin{aligned}
\sum_{G_d \times G_d} c_{a,b}^{(t)} \pi_{a,b} &= \sum_{N_c^{(t)}} c_{a,b}^{(t)} \pi_{a,b} + t \sum_{G_d \times G_d \setminus N_c^{(t)}} \pi_{a,b} \\
&= \sum_{N_c^{(t)}} c_{a,b}^{(t)} \pi_{a,b} + t \left(\sum_{G_d \times G_d} \pi_{a,b} - \sum_{N_c^{(t)}} \pi_{a,b} \right) \qquad (9) \\
&= \sum_{N_c^{(t)}} c_{a,b}^{(t)} \pi_{a,b} + t \left(1 - \sum_{N_c^{(t)}} \pi_{a,b} \right) = t - \sum_{N_c^{(t)}} \left(t - c_{a,b}^{(t)} \right) \pi_{a,b} \\
&= t - \mathbb{B}_c^{(t)}(\eta).
\end{aligned}$$

In (9), since π is defined as a (joint) probability, we have $\sum_{G_d \times G_d} \pi_{a,b} = 1$.

We have shown that for every $\pi \in \Pi_{\mu,\nu}$ we can define a nearby flow η such that

$$\mathbb{T}_{c^{(t)}}(\pi) = t - \mathbb{B}_c^{(t)}(\eta).$$

However, it could exist a nearby flow η with a larger value of $\mathbb{B}_c^{(t)}(\eta)$, and, hence, so far we have only proved

$$\min_{\pi \in \Pi_{\mu,\nu}} \mathbb{T}_{c^{(t)}}(\pi) \geq t - \max_{\eta \in \mathcal{N}_{\mu,\nu}} \mathbb{B}_c^{(t)}(\eta). \tag{10}$$

Now we want to show that the previous relation holds with equality. We need to show that given a nearby flow $\eta \in \mathcal{N}_{\mu,\nu}$ we can get a feasible π. We start by introducing the slack variables of constraints (7) and (8):

$$\tilde{\mu}_a := \mu_a - \sum_{b \in O_c^{(t)}(a)} \eta_{a,b}, \quad \forall a \in G_d,$$
$$\tilde{\nu}_b := \nu_b - \sum_{a \in I_c^{(t)}(b)} \eta_{a,b}, \quad \forall b \in G_d.$$

We have $\tilde{\mu}_a \geq 0$ and $\tilde{\nu}_b \geq 0$ for each $a, b \in G_d$. Since μ and ν are probabilities (i.e., $\sum_{a \in G_d} \mu_a = \sum_{b \in G_d} \nu_b = 1$), we have that

$$\begin{aligned}\sum_{a \in G_d} \tilde{\mu}_a &= \sum_{a \in G_d} \left(\mu_a - \sum_{b \in O_c^{(t)}(a)} \eta_{a,b} \right) \\ &= 1 - \sum_{a \in G_d} \sum_{b \in O_c^{(t)}(a)} \eta_{a,b} \\ &= \sum_{b \in G_d} \nu_b - \sum_{b \in G_d} \sum_{a \in I_c^{(t)}(b)} \eta_{a,b} = \sum_{b \in G_d} \tilde{\nu}_b.\end{aligned}$$

We introduce the value $M = \sum_{b \in G_d} \tilde{\nu}_b = \sum_{a \in G_d} \tilde{\mu}_a$, which is used to define

$$\pi_{a,b} := \begin{cases} \dfrac{\tilde{\mu}_a \tilde{\nu}_b}{M} & \text{if } (a,b) \in (G_d \times G_d) \backslash N_c^{(t)}, \\ \eta_{a,b} + \dfrac{\tilde{\mu}_a \tilde{\nu}_b}{M} & \text{if } (a,b) \in N_c^{(t)}. \end{cases} \tag{11}$$

We now show that $\pi_{a,b} \in \Pi_{\mu,\nu}$. For all $b \in G_d$ it holds

$$\begin{aligned}\sum_{a\in G_d} \pi_{a,b} &= \sum_{a\in G_d\setminus I_c^{(t)}(b)} \pi_{a,b} + \sum_{a\in I_c^{(t)}(b)} \pi_{a,b} \\ &= \sum_{a\in G_d\setminus I_c^{(t)}(b)} \frac{\tilde{\mu}_a\tilde{\nu}_b}{M} + \sum_{a\in I_c^{(t)}(b)} \eta_{a,b} + \sum_{a\in I_c^{(t)}(b)} \frac{\tilde{\mu}_a\tilde{\nu}_b}{M} \\ &= \sum_{a\in G_d} \frac{\tilde{\mu}_a\tilde{\nu}_b}{M} + \sum_{a\in I_c^{(t)}(b)} \eta_{a,b} = \tilde{\nu}_b + \sum_{a\in I_c^{(t)}(b)} \eta_{a,b} = \nu_b.\end{aligned}$$

Similarly, we can show that $\sum_{b\in G_d} \pi_{a,b} = \mu_a$, and, hence, the π defined in (11) belongs to $\Pi_{\mu,\nu}$. Regarding its cost, we have

$$\begin{aligned}\sum_{(a,b)\in G_d\times G_d} c_{a,b}^{(t)}\pi_{a,b} &= \sum_{(a,b)\in G_d\setminus N_c^{(t)}} c_{a,b}^{(t)}\pi_{a,b} + \sum_{(a,b)\in N_c^{(t)}} c_{a,b}^{(t)}\pi_{a,b} \\ &= t \sum_{(a,b)\in G_d\setminus N_c^{(t)}} \frac{\tilde{\mu}_a\tilde{\nu}_b}{M} + \sum_{(a,b)\in N_c^{(t)}} c_{a,b}^{(t)}\eta_{a,b} + \sum_{(a,b)\in N_c^{(t)}} c_{a,b}^{(t)}\frac{\tilde{\mu}_a\tilde{\nu}_b}{M} \\ &\le t \sum_{(a,b)\in G_d\times G_d} \frac{\tilde{\mu}_a\tilde{\nu}_b}{M} + \sum_{(a,b)\in N_c^{(t)}} c_{a,b}^{(t)}\eta_{a,b} \\ &= tM + \sum_{(a,b)\in N_c^{(t)}} c_{a,b}^{(t)}\eta_{a,b}.\end{aligned}$$

By definition, the constant $M = \sum_{b\in G_d} \tilde{\nu}_b$ can be rewritten as

$$M = \sum_{b\in G_d}\left(\nu_b - \sum_{a\in I_c^{(t)}(b)} \eta_{a,b}\right) = \sum_{b\in G_d} \nu_b - \sum_{(a,b)\in N_c^{(t)}} \eta_{a,b} = 1 - \sum_{(a,b)\in N_c^{(t)}} \eta_{a,b}.$$

Since $s_{a,b} = t - c_{a,b}$, we can write

$$\begin{aligned}\sum_{(a,b)\in G_d\times G_d} c_{a,b}^{(t)}\pi_{a,b} &\le tM + t \sum_{(a,b)\in N_c^{(t)}} \eta_{a,b} - \sum_{(a,b)\in N_c^{(t)}} s_{a,b}\eta_{a,b} \\ &= t - \sum_{(a,b)\in N_c^{(t)}} s_{a,b}\eta_{a,b}.\end{aligned}$$

Thus, we showed that for each nearby flow $\eta \in \mathcal{N}_{\mu,\nu}$ there exists a $\pi \in \Pi_{\mu,\nu}$ such that

$$\mathbb{T}(\pi)_{c^{(t)}}(\pi) \le t - \mathbb{B}(\eta)_c^{(t)}(\eta),$$

and, hence

$$\min_{\pi \in \Pi_{\mu,\nu}} \mathbb{T}_{c^{(t)}}(\pi) \leq t - \max_{\eta \in \mathcal{N}_{\mu,\nu}} \mathbb{B}_c^{(t)}(\eta),$$

which together with the inequality (10) completes the proof. □

The main consequence of Theorem 1 is that whenever we are using a truncated ground distance $c_{a,b}^{(t)}$, the transportation problem (3)–(6) can be reformulated with the following **Maximum Nearby Flow Problem (MNF)**:

$$\begin{aligned}
(\textbf{MNF}) \quad W_{c^{(t)}}(\mu,\nu) := t - \max \quad & \sum_{(a,b)\in N_c^{(t)}} s_{a,b}\eta_{a,b} \\
\text{s.t.} \quad & \sum_{a\in I_c^{(t)}(b)} \eta_{a,b} \leq \nu_b, && \forall b \in G_d, \\
& \sum_{b\in O_c^{(t)}(b)} \eta_{a,b} \leq \mu_a, && \forall a \in G_d, \\
& \eta_{a,b} \geq 0 && \forall (a,b) \in N_c^{(t)}.
\end{aligned}$$

Depending on the type of cost function c and on the value of the threshold parameter t, the number of variables $\eta_{a,b}$ can be reduced to a small fraction of the number of variables appearing in (3)–(6). Indeed, when $t = \max_{a,b} c_{a,b}$, then the previous problem gives the optimal value of the original problem; for smaller values of t, it provides a lower bound. Unfortunately, from an optimal solution $\eta_{a,b}^*$ of the maximum nearby flow problem, we cannot obtain an optimal solution π (i.e., a transportation plan) for the Wasserstein distance.

4 Computational Results

We run several numerical tests with the objective of assessing the impact of the threshold value t on the gap between the value of the lower bound given by problem (MNF) with respect to the optimal solution value of problem (EMD), that is, the ratio $\frac{W_{c^{(t)}}}{W_c}$. In addition, we compute the ratio between the runtime of solving the two corresponding linear problems using the commercial solver Gurobi v8.0.

As problem instances, we use a collection of ten different gray scale images with 32×32 pixels, which belong to the DOTmark benchmark [22]. We compute the distance between every pair of images, for a total of 45 pairs. For each pair of images, first, we solve once problem (EMD), and then, we solve problem (MNF) with the threshold value t ranging from 1 up to $\lceil 32\sqrt{2} \rceil$, increasing t by 1 each time.

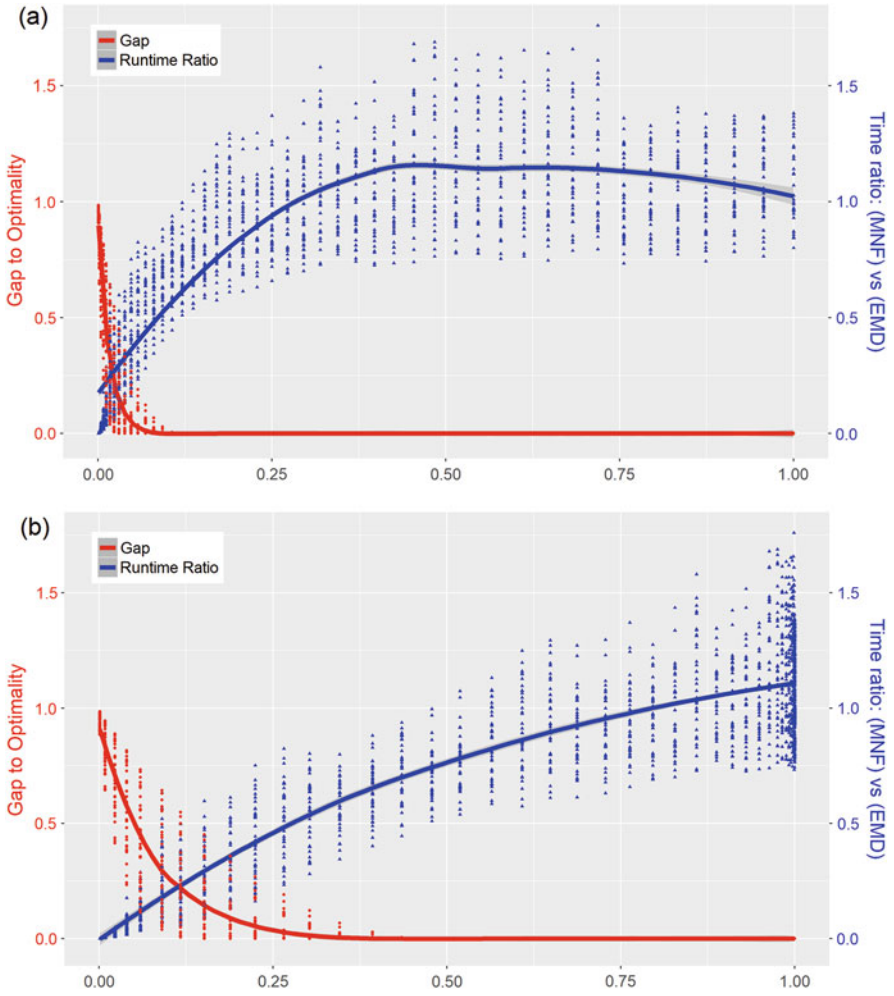


Fig. 1 Gap to optimality (left y-scale) and time ratio between solving problem (MNF) and (EMD) (right y-scale): in (**a**) as a function of the threshold $\frac{t}{C_{max}}$, where t is the distance threshold and C_{max} is the maximum distance. In (**b**) as a function of the number of arc variables in problem (MNF) against the number of arc variables in problem (EMD), that is, the ratio $\frac{|N_C^{(t)}|}{|G_d \times G_d|}$. (**a**) Threshold ratio: t/Cmax. (**b**) Ratio of the number of arcs variables

Figure 1a shows the results as function of the threshold value t over the maximum distance $C_{max} = 32\sqrt{2}$, while Fig. 1b shows the same results, but as a function of the number of the arc variables in (MNF) over (EMD), that is, the ratio $\frac{|N_C^{(t)}|}{|G_d \times G_d|}$. Note that in (EMD) we have a complete bipartite graph, while in (MNF) we have a subset of the complete arc set. Both figures show in the left y-axis the gap value, and on the right y-axis the ratio of the runtime for solving the problems.

Clearly, we can remark two main features for this type of instances. First, a small value of the threshold t permits to obtain already a gap close to zero (see Fig. 1a). However, as expected, the complexity of the problem does not really depend on the value of the threshold t, but on the number of arc variables that a given value of t implies, that is, the cardinality of the set $N_c^{(t)}$ (see Fig. 1b).

Acknowledgements This research was partially supported by the Italian Ministry of Education, University and Research (MIUR): Dipartimenti di Eccellenza Program (2018–2022)—Dept. of Mathematics "F. Casorati", University of Pavia.

References

1. Altschuler, J., Weed, J., Rigollet, P.: Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration. Adv. Neural Inf. Proces. Syst. 1961–1971 (2017)
2. Amaldi, E., Coniglio, S., Gualandi, S.: Coordinated cutting plane generation via multi-objective separation. Math. Program. **143**(1), 87–110 (2014)
3. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein GAN. arXiv preprint. arXiv:1701.07875 (2017)
4. Auricchio, G., Bassetti, F., Gualandi, S., Veneroni, M.: Computing Kantorovich-Wasserstein distances on d-dimensional histograms using $(d + 1)$-partite graphs. Adv. Neural Inf. Proces. Syst. 5793–5803 (2018)
5. Bassetti, F., Regazzini, E.: Asymptotic properties and robustness of minimum dissimilarity estimators of location-scale parameters. Theory Probab. Appl. **50**(2), 171–186 (2006)
6. Bassetti, F., Bodini, A., Regazzini, E.: On minimum Kantorovich distance estimators. Stat. Probab. Lett. **76**(12), 1298–1302 (2006)
7. Bassetti, F., Gualandi, S., Veneroni, M.: On the computation of Kantorovich-Wasserstein distances between 2D-histograms by uncapacitated minimum cost flows. arXiv preprint, arXiv:1804.00445 (2018)
8. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, New York (2006)
9. Cuturi, M.: Sinkhorn distances: Lightspeed computation of optimal transport. Adv. Neural Inf. Proces. Syst. 2292–2300 (2013)
10. Cuturi, M., Doucet, A.: Fast computation of Wasserstein barycenters. In: International Conference on Machine Learning, pp. 685–693 (2014)
11. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley, New York (2012)
12. Flood, M.M.: On the Hitchcock distribution problem. Pac. J. Math. **3**(2), 369–386 (1953)
13. Frogner, C., Zhang, C., Mobahi, H., Araya, M., Poggio, T.A.: Learning with a Wasserstein loss. Adv. Neural Inf. Proces. Syst. 2053–2061 (2015)
14. Goldberg, A.V., Tardos, É., Tarjan, R.: Network flow algorithm. Technical report, Cornell University Operations Research and Industrial Engineering (1989)
15. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics. Springer, New York (2009)
16. Levina, E., Bickel, P.: The Earth mover's distance is the Mallows distance: some insights from statistics. In: Proceedings of the Eighth IEEE International Conference on Computer Vision, 2001. ICCV 2001, vol. 2, pp. 251–256. IEEE (2001)
17. Ling, H., Okada, K.: An efficient earth mover's distance algorithm for robust histogram comparison. IEEE Trans. Pattern Anal. Mach. Intell. **29**(5), 840–853 (2007)
18. Orlin, J.B.: A faster strongly polynomial minimum cost flow algorithm. Oper. Res. **41**(2), 338–350 (1993)

19. Pele, O., Werman, M.: Fast and robust earth mover's distances. In: 2009 IEEE 12th International Conference on Computer Vision, pp. 460–467. IEEE (2009)
20. Rubner, Y., Tomasi, C., Guibas, L.J.: A metric for distributions with applications to image databases. In: Sixth International Conference on Computer Vision, 1998, pp. 59–66. IEEE (1998)
21. Rubner, Y., Tomasi, C., Guibas, L.J.: The earth mover's distance as a metric for image retrieval. Int. J. Comput. Vis. **40**(2), 99–121 (2000)
22. Schrieber, J., Schuhmacher, D., Gottschlich, C.: Dotmark–a benchmark for discrete optimal transport. IEEE Access **5**, 271–282 (2017)
23. Schrijver, A.: On the history of the transportation and maximum flow problems. Math. Program. **91**(3), 437–445 (2002)
24. Solomon, J., Rustamov, R., Guibas, L., Butscher, A.: Wasserstein propagation for semi-supervised learning. In: International Conference on Machine Learning, pp. 306–314 (2014)
25. Solomon, J., De Goes, F., Peyré, G., Cuturi, M., Butscher, A., Nguyen, A., Du, T., Guibas, L.: Convolutional Wasserstein distances: efficient optimal transportation on geometric domains. ACM Trans. Graph. (TOG) **34**(4), 66 (2015)

Linear Models for Portfolio Selection with Real Features

Thiago Alves de Queiroz, Leandro Resende Mundim, and André Carlos Ponce de Leon Ferreira de Carvalho

Abstract An efficient investment portfolio would have maximum return or minimum risk. Several approaches based on the "expected returns - variance of returns" rule seek for a good balance between yield and risk. These approaches may differ in either how to measure risk or how to estimate expected yields. In this work we consider linear programming models found in the literature to estimate risks, like mean absolute deviation and Gini's mean difference. Thus, two mixed integer programming models are investigated in a portfolio optimization problem for a given expected return. For such, we add real features, including transaction lots, cardinality, and investment threshold. Experiments using data from the Dow Jones stock market demonstrate the superiority of the investigated models in the presence of these real features when compared with a market average indicator of return.

Keywords Portfolio optimization · Mean-risk models · Mean absolute deviation · Gini's mean difference · Real features

1 Introduction

In [9], we can find the "expected returns - variance of returns" (E-V) rule that has been used as the basis for many mean-risk models in portfolio optimization. In this rule, an investor wants to place his/her budget in a set of securities (or assets) in order to minimize risk (e.g., variance of return) or maximize yield (e.g., expected return). These two objectives are usually conflicting, since when one increases, the

T. A. de Queiroz
Institute of Mathematics and Technology, Federal University of Goiás, Catalão, GO, Brazil
e-mail: taq@ufg.br

L. R. Mundim (✉) · A. C. P. L. F. de Carvalho
Institute of Mathematics and Computer Sciences, University of São Paulo, São Carlos, SP, Brazil
e-mail: mundim@icmc.usp.br; andre@icmc.usp.br

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_4

other decreases. Usually, the higher the risk, the higher the yield, so an investor would choose the portfolio with the smallest risk.

The Markowitz's mean-variance model described in [9] considers n assets with expected return values $\mu \in \mathbb{R}^n$ and covariance matrix $\sigma \in \mathbb{R}^{n \times n}$. Continuous variables $x \in \mathbb{R}^n$ indicate the relative amount invested in each asset, for a single period of investment. Moreover, the sum of all relative amount invested must be 1. Thus, the maximum expected return for a given risk (in this case, the variance of returns) and the minimum risk that can be achieved for a given expected return can be modelled.

Different formulations have been proposed to linearize the risk measure, since in [9] it is quadratic computable. One of them is the Gini mean difference (GMD) model, where the risk is linear programming computable, proposed in [12]. In the GMD model, the risk is modeled by the Gini's mean difference. Another is the Mean Absolute Deviation (MAD) model, proposed in [5], which defines the risk measure as the mean absolute deviation from the mean.

On the other hand, in [13], the risk was modeled based on the minimum return instead of variance, resulting in a minimax portfolio problem. In [10], the risk measure was tackled as a conditional value at risk (CVaR) instead of value at risk (VaR), since the latter lacks characteristics such as subadditivity and convexity. The above mentioned models have opened new opportunities in different application contexts, specially in the financial field. For example, in [3], a mean-variance model with CVaR constraints was investigated for a pension fund asset allocation. In [14], a continuous-time sticky-price model was combined with a stochastic dynamic portfolio model in order to evaluate different risk measures and inflation.

A survey on linear programming models in portfolio optimization was given in [7]. Besides discussing pros and cons of some models (e.g., the ones mentioned above) and the main algorithms (i.e., exact ones and heuristics) proposed in the literature, these authors also commented on the need of including real features (e.g., transaction costs and lots, cardinality, and thresholds on investments) in models to tackle practical financial problems.

Similarly, in [4] classical mean-risk models related to the Markowitz's mean-variance model were reviewed. It was also discussed extensions of such models in order to include transaction costs, investment guidelines, and institutional features; the usage of Bayesian techniques, stochastic optimization, or robust optimization approaches to estimate errors in risk measures; and, multi-period approaches to incorporate different practical constraints.

With this in mind, in this work, we compare mean-risk models (the MAD and GMD models), which are linear programming computable, in the presence of three real features. The mentioned features are: transaction lots [1], cardinality constraint [11], and investment threshold constraints [6]. These models have played an important role in portfolio optimization and only a few studies have investigated real features associated with them. Therefore, in order to evaluate the models, real data from the Dow Jones Industrial Average index is used [2], comprising a period between 1990 and 2016.

The paper is organized as follows. The next section states the problem and the original MAD and GMD models. In Sect. 3 there is a description of the real features and the resulting models after adding such features. The computational experiments are described in Sect. 4, while the last section contains the concluding remarks and directions for future research.

2 Problem and Literature Models

The portfolio problem solved here considers a single period of investment, following [9]. There is a set of n assets, where for each asset i, $j = 1, \ldots, n$: R_i is a random variable representing the return; μ_i is the expected return, that is $\mu_i = E(R_i)$; r_{it} is the return at time t, for a discrete set of times $t = 1, \ldots, T$; σ_{ij} is the covariance between R_i and R_j, while σ_{ii} is the variance of R_i. The investor has an available budget $\bar{C}$ to invest.

Associated to each asset i there is a continuous variable $x_i \geq 0$ indicating the fixed percentage invested in i, where it holds $\sum_{i=1}^{n} x_i = 1$. Then, for the portfolio $\mathcal{X}$, its return (i.e., yield) is $R_{\mathcal{X}} = \sum_{i=1}^{n} R_i x_i$, its mean expected return is $\mu(\mathcal{X}) = \sum_{i=1}^{n} \mu_i x_i$, and its risk measure is given by $\mathcal{V}$. The objective is to achieve a portfolio $\mathcal{X}$ of minimum risk $\mathcal{V}$ attaining at least a given yield $\mathcal{E}$, assuming that $\bar{C}$ is the invested capital.

It is below presented the mean absolute deviation (i.e., MAD) model. Differently from the Markowitz's mean-variance model in which the risk is defined as the variance of return (i.e., $\mathcal{V}_V = \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij} x_i x_j$), the MAD one has the risk defined as the mean absolute deviation from the mean, that is $\mathcal{V}_{MAD} = E\left(|\mu(\mathcal{X}) - R_{\mathcal{X}}|\right)$. According to [5], the latter is linear programming computable and for that, the expected return is defined as $\mu_i = \frac{1}{T} \sum_{t=1}^{T} r_{it}$, for $i = 1, \ldots, n$. Moreover, let $a_{jt} = \mu_j - r_{jt}$, then we have the nonlinear MAD model (1)–(4) for the portfolio problem under study.

$$\text{Minimize} \quad \frac{1}{T} \sum_{t=1}^{T} \left| \sum_{i=1}^{n} a_{jt} x_i \right| \tag{1}$$

$$\text{Subject to:} \quad \sum_{i=1}^{n} x_i = 1, \tag{2}$$

$$\sum_{i=1}^{n} x_i \mu_i \geq \mathcal{E}, \tag{3}$$

$$x_i \geq 0, \quad \forall\, i = 1, \ldots, n. \tag{4}$$

The objective function (1) is related to a solution of minimum risk, in which the risk is formulated as the mean absolute deviation. Constraint (2) impose the total percentage invested on being one, while constraint (2) assure the expected return of being at least the given yield $\mathcal{E}$.

Notice that the objective function (1) is nonlinear, however, it can be linearized by adding continuous variables y_t as representing $|\sum_{i=1}^{n} a_{jt}x_i|$, for each $t = 1, \ldots, T$, resulting in the equivalent linear MAD model (5)–(10). Because of introducing variables y_t, two new set of constraints has emerged, which are (8) and (9), in order to express the risk measure $\mathcal{V}_{MAD}$.

$$\text{Minimize} \quad \frac{1}{T}\sum_{t=1}^{T} y_t \tag{5}$$

$$\text{Subject to:} \quad \sum_{i=1}^{n} x_i = 1, \tag{6}$$

$$\sum_{i=1}^{n} x_i \mu_i \geq \rho, \tag{7}$$

$$y_t + \sum_{i=1}^{n} x_i a_{it} \geq 0, \quad \forall\, t = 1, \ldots, T, \tag{8}$$

$$y_t - \sum_{i=1}^{n} x_i a_{it} \geq 0, \quad \forall\, t = 1, \ldots, T, \tag{9}$$

$$x_i \geq 0, \quad \forall\, i = 1, \ldots, n. \tag{10}$$

The GMD model is the another model considered in this study. In this model, the risk is defined in accordance with the Gini's mean difference. According to [8], this difference can be defined as $\mathcal{V}_{GMD} = E\left(|x_i - x_j|\right) = \frac{1}{n}\sum_{j=1}^{n} |x_i - x_j|$. As this risk is still nonlinear, continuous variables G_{ij} can be used for representing $|x_i - x_j|$, resulting in the linear GMD model (11)–(16).

$$\text{Minimize} \quad \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} G_{ij} \tag{11}$$

$$\text{Subject to:} \quad \sum_{i=1}^{n} x_i = 1, \tag{12}$$

$$\sum_{i=1}^{n} x_i \mu_i \geq \mathcal{E}, \tag{13}$$

$$G_{ij} + x_i - x_j \geq 0, \quad \forall\, i, j = 1, \ldots, n, \tag{14}$$

$$G_{ij} - x_i + x_j \geq 0, \quad \forall\, i, j = 1, \ldots, n, \tag{15}$$

$$x_i \geq 0, \quad \forall\, i = 1, \ldots, n. \tag{16}$$

3 Real Features

The inclusion of real features in portfolio optimization models is motivated by the fact that practical problems have different real-world conditions to be respected. For such, three real features [4, 7] are considered:

- Transaction lots: a transaction lot represents the minimum budget required to invest in an asset. The investment in an asset occurs in multiples of transaction lots. Thus, if this feature is disregarded an infeasible portfolio may be created. To model this feature, let s_i be the value of one transaction lot associated with asset i, while the number of selected lots of i is due to the integer non-negative variable w_i. The following constraint emerges for each asset i: $\bar{C}x_i = s_i w_i$, where $\bar{C}$ is the invested capital.
- Cardinality constraint: to invest all the available budget in only one asset may increase the risk. On the other hand, to invest in many assets may increase monitoring and transaction costs. Thus, it is common to limit the number of assets to invest (i.e., to impose limits on the cardinality of the set of selected assets). To model this constraint, a binary variable z_i is defined for each asset i, indicating i is in the set of selected assets (i.e., $z_i = 1$ stands for $x_i > 0$). Considering lower K_{min} and upper K_{max} bounds on the cardinality of this set, the following constrains are derived: $K_{min} \leq z_i$ and $z_i \leq K_{max}$.
- Investment threshold constraints: these constraints impose lower and upper limits on the fraction of the budget to be invested in each selected asset. These constraints are important to control (i.e., eliminate insignificant or avoid exaggerated) investments in assets. Thus, for each asset i: $l_i z_i \leq x_i$ and $x_i \leq u_i z_i$.

According to [7], there is a lack of works combining real features. Real features allow to model complex situations that emerge in practical financial applications. To allow this modelling, the three previous real features are added into the linear MAD and GMD models described in Sect. 2.

The linear MAD model with the three real features (i.e., transaction lots, cardinality, and investment threshold) is given in (17)–(28). The objective function (17) and constraints (18)–(21) are from the linear MAD model described in Sect. 2. On the other hand, constraints (22) impose z_i on being one if asset i holds an investment. Constraints (23) are related to the transaction lots, while (24) are the cardinality constraints. Investment threshold constraints are given in (25) and variables domains are in (26)–(28).

$$\text{Minimize} \quad \frac{1}{T}\sum_{t=1}^{T} y_t \tag{17}$$

$$\text{Subject to:} \quad \sum_{i=1}^{n} x_i = 1, \tag{18}$$

$$\sum_{i=1}^{n} x_i \mu_i \geq \mathcal{E}, \tag{19}$$

$$y_t + \sum_{i=1}^{n} x_i a_{it} \geq 0, \quad \forall\, t = 1, \ldots, T, \tag{20}$$

$$y_t - \sum_{i=1}^{n} x_i a_{it} \geq 0, \quad \forall\, t = 1, \ldots, T, \tag{21}$$

$$z_i - x_i \geq 0, \quad \forall\, i = 1, \ldots, n, \tag{22}$$

$$\bar{C} x_i - s_i w_i = 0, \quad \forall\, i = 1, \ldots, n, \tag{23}$$

$$K_{min} \leq \sum_{i=1}^{n} z_i \leq K_{max}, \tag{24}$$

$$l_i z_i \leq x_i \leq u_i z_i, \quad \forall\, i = 1, \ldots, n, \tag{25}$$

$$z_i \in \{0, 1\}, \quad \forall\, i = 1, \ldots, n, \tag{26}$$

$$w_i \in \mathbb{Z}_+, \quad \forall\, i = 1, \ldots, n, \tag{27}$$

$$x_i \geq 0, \quad \forall\, i = 1, \ldots, n. \tag{28}$$

Similarly, the linear GMD model with these real features is described in (29)–(40). The objective function (29) and constraints (30)–(33) are the same of the linear GMD model in Sect. 2. Moreover, constraints (34)–(40) are equal to the respective constraints (22)–(28).

$$\text{Minimize} \quad \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} G_{ij} \tag{29}$$

$$\text{Subject to:} \quad \sum_{i=1}^{n} x_i = 1, \tag{30}$$

$$\sum_{i=1}^{n} x_i \mu_i \geq \mathcal{E}, \tag{31}$$

$$G_{ij} + x_i - x_j \geq 0, \quad \forall\, i, j = 1, \ldots, n, \tag{32}$$

$$G_{ij} - x_i + x_j \geq 0, \quad \forall\, i, j = 1, \ldots, n, \tag{33}$$

$$z_i - x_i \geq 0, \quad \forall\, i = 1, \ldots, n, \tag{34}$$

$$\bar{C} x_i - s_i w_i = 0, \quad \forall\, i = 1, \ldots, n, \tag{35}$$

$$K_{min} \leq \sum_{i=1}^{n} z_i \leq K_{max}, \tag{36}$$

$$l_i z_i \leq x_i \leq u_i z_i, \quad \forall\, i = 1, \ldots, n, \tag{37}$$

$$z_i \in \{0, 1\}, \quad \forall\, i = 1, \ldots, n, \tag{38}$$

$$w_i \in \mathbb{Z}_+, \quad \forall\, i = 1, \ldots, n, \tag{39}$$

$$x_i \geq 0, \quad \forall\, i = 1, \ldots, n. \tag{40}$$

4 Computational Experiments

This section has the results of the computational experiments over the MAD and GMD models described in the previous section (i.e., the linear MAD and GMD models with the three real features). The experiments were carried out in a computer with Intel® Core™ i7-2600 3.40 GHz CPU and 32 GB of memory RAM, running Ubuntu 16.04 LTS. The models were coded in C++ programming language and solved by the libraries (with default parameters) of the IBM ILOG CPLEX 12.7 Optimization Studio. A time limit of 3600 s was imposed when solving each model.

The size of the MAD and GMD models is determined by the number of assets (i.e., n) and of time (i.e., T days, weeks, months, etc.). Table 1 has the number of variables of each of these models in the classical versions (in Sect. 2) and with the three real features (in Sect. 3). Each line of this table has the name of the model, the number of real, binary (if any), and integer (if any) variables, and the number of constraints.

Notice that the MAD and GMD models in Sect. 2 are linear, while the respective ones in Sect. 3 (with the three real features) have integer programming variables (i.e., they are integer programming models). Observing Table 1, the MAD model has a linear number of variables and constraints, while in the GMD these numbers grow quadratically.

For the computational experiments, data from the Dow Jones Industrial Average index is used [2]. This index considers $n = 28$ assets and $T = 1363$ weeks with linear returns computed on daily price data, and adjusted for dividends and stock splits. The time interval ranges from February 1990 to April 2016.

Table 1 Number of variables and constraints

Model	#Real	#Binary	#Integer	#Constraints
MAD	2n			2T+n+2
MAD (with the three features)	2n	n	n	2T+9n+3
GMD	n^2+n			$2n^2$+2
GMD (with the three features)	n^2+n	n	n	$2n^2$+9n+4

In the MAD and GMD models with the three real features, the expected return μ_i is averaged over the τ past weeks of T. We tested in the preliminary experiments different values of $\tau = \{10, 20, 30, 40, 50, 100, 1000\}$. We noticed in these experiments that either small ($\tau = \{10, 20, 30, 40\}$) or large ($\tau = \{1000\}$) values of τ may not result in reasonable values of expected returns. The results for the values of $\tau = \{50, 100\}$ are very similar and have no statistical difference. Due to the limitation of space, in this paper we chose to use the $\tau = 100$.

With regard to the invested capital, we assume $\bar{C} = 1$ (i.e., it is normalized since we have relative models). The value of one transaction lot is set to 100, that is $s_i = 100$ for $i = 1, \ldots, n$. The limits on the cardinality of the set of selected assets, that is, values for $\{K_{min}, K_{max}\}$ are $\{1, 10\}$, $\{1, 5\}$, and $\{5, 10\}$, so three cases are taken into consideration. For the investment threshold constraints, we assume that $l_i = 0.1$ and $w_i = 0.9$ for all assets i. Finally, three values for the given expected return $\mathcal{E}$ are considered, which are 50% (R1, indicating a low risk), 75% (R2, indicating a moderate risk), and 95% (R3, indicating a high risk) of the maximum return $\mathcal{E}_{max}$. This return is obtained by solving the respective MAD (or GMD) model without the three real features, considering the $\tau = 100$ past weeks.

Therefore, the strategy, which is used to obtain the results of the next subsections, consists of: (i) for each week t, starting at $t = 101$; (ii) considering the $\tau = 100$ past weeks, obtain the expected return μ_i (for all assets i) and solve the linear MAD (or GMD) model without any of the three real features in order to obtain $\mathcal{E}_{max}$; (iii) with the values of steps (ii) and the other ones mentioned above, solve the linear MAD (or GMD) model with the three real features; (iv) return to step (i) if $T < 1363$.

The results of the MAD and GMD models with the three real features (which are obtained by the strategy above) are compared with a market average indicator of return, which we call benchmark. The benchmark is calculated as $benchmark_t = \sum_{i=1}^{n} \frac{1}{n} \mu_i^*$, where $x_i = \frac{1}{n}$ and μ_i^* represents the real return of the asset i, which is known at the end of each week t.

Table 2 has results of the linear MAD model with the three real features. In this table, there are: the value of the parameters s_i, K_{min}, K_{max}, l_i, u_i; and, type of value for R1, R2, R3, and benchmark: minimum and maximum returns over all T weeks; final value at the end of the last week (i.e., for $T = 1363$) (in the table is Final value); the percentage per week for each model, and variance of returns over all T weeks.

Observing Table 2, the benchmark at the end of the last week corresponds to 1861.40, with a percentage per week of 0.2186 and variance of 493.39, while the MAD for $\{K_{min}, K_{max}\} = \{5, 10\}$ has the percentage per week of 0.3826, although its variance is quite high (of 6983.08) and so its risk (R3). In this case, the MAD is about 9.5 superior than the benchmark. For low and moderate risks, better results are achieved with the MAD when $\{K_{min}, K_{max}\} = \{1, 5\}$ and $\{1, 10\}$, respectively.

Results of the linear GMD model with the three real features are given in Table 3. Once again the benchmark has the worst return, although it has the lowest variance. The GMD has the best final value (of 15,984.84) at the end of the last week for $\{K_{min}, K_{max}\} = \{1, 10\}$, with a percentage per week from 0.3734 but for a high risk (R3). This final value is about 8.5 times greater than the benchmark. For low and

Table 2 Results of the linear MAD with the three features

s_i	K_{min}	K_{max}	l_i	u_i	Type	R1	R2	R3	Benchmark
100	1	5	0.1	0.9	Min value	92.61	89.78	88.40	97.77
					Max value	935.73	4459.21	24,724.81	1902.96
					Final value	922.92	4394.70	18,121.67	1861.40
					Percentage per week	0.1707	0.2796	0.3826	0.2186
					Variance	217.06	1253.50	6983.08	493.39
100	5	10	0.1	0.9	Min	92.02	91.11	91.52	97.77
					Max	809.40	2104.73	9303.23	1902.96
					Final value	799.14	2076.08	8900.84	1861.40
					Percentage per week	0.1613	0.2262	0.3307	0.2186
					Variance	181.73	524.79	2685.31	493.39
100	1	10	0.1	0.9	Min value	91.92	89.26	88.40	97.77
					Max value	915.37	4606.23	24,724.81	1902.96
					Final value	899.43	4541.66	18,121.67	1861.40
					Percentage per week	0.1690	0.2820	0.3826	0.2186
					Variance	202.36	1241.76	6983.08	493.39

Table 3 Results of the linear GMD with the three features

s_i	K_{min}	K_{max}	l_i	u_i	Type	R1	R2	R3	Benchmark
100	1	5	0.1	0.9	Min value	87.87	91.95	87.93	97.77
					Max value	2143.88	3484.81	18,682.96	1902.96
					Final value	2003.27	3281.8	14,181.16	1871.82
					Percentage per week	0.2236	0.2587	0.3647	0.2190
					Variance	516.59	956.09	5251.62	493.42
100	5	10	0.1	0.9	Min value	92.95	92.28	93.09	97.77
					Max value	3667.6	5255.31	9095.89	1902.96
					Final value	3666.45	5161.66	8230.38	1871.82
					Percentage per week	0.2666	0.2912	0.3250	0.2190
					Variance	958.98	1395.94	2604.67	493.42
100	1	10	0.1	0.9	Min value	90.95	91.1	95.71	97.77
					Max value	2646.57	4899.95	19,652.99	1902.96
					Final value	2510.14	4515.96	15,984.84	1861.4
					Percentage per week	0.2396	0.2815	0.3734	0.2190
					Variance	721.04	1417.61	5678.49	493.39

moderate risks, the best final value of the GMD are achieved when $\{K_{min}, K_{max}\} = \{5, 10\}$.

Observing all the experiments, effective results can be achieved by both models (MAD and GMD) when a high final value is desired (although this also means there is a high risk to pay back) and more assets are considered for investment (i.e., a diversified portfolio is generally preferable).

5 Conclusions

A portfolio optimization problem is investigated in this work by means of linear programming computable models. In this problem, we aim the portfolio of minimum risk and which is able to satisfy a given expected return. Two mean-risk models from the literature are considered, the GMD model and the MAD model. Besides, real features, like transaction lots, cardinality, and investment threshold, are taken into account when solving these models.

The experimental results show that both MAD and GMD models with the three real features improve the return, with a lower risk, when compared with the market average indicator of return (i.e., benchmark). Depending on the desired expected return, the differences between these models and the benchmark values are relatively large (respectively, around 9.5 and 8.5 times better than the benchmark).

Future works will focus on heuristics (e.g., non-dominated sorting genetic algorithm and multi-objective variable neighborhood search) to solve the bi-objective portfolio problem defined in [9] when different real features are considered, including features not considered here, like transaction costs and decision dependency requirements.

Acknowledgements The authors would like to thank CNPq (grant 308312/2016-3), CAPES, FAPEG, FAPESP (2013/07375-0) and Intel for their support.

References

1. Baumann, P., Trautmann, N.: Portfolio-optimization models for small investors. Math. Meth. Oper. Res. **77**(3), 345–356 (2013)
2. Bruni, R., Cesarone, F., Scozzari, A., Tardella, F.: Real-world datasets for portfolio selection and solutions of some stochastic dominance portfolio models. Data Brief **8**, 858–862 (2016)
3. Chen, Y., Sun, X., Li, J.: Pension fund asset allocation: a mean-variance model with CVaR constraints. Proc. Comput. Sci. **108**, 1302–1307 (2017). International Conference on Computational Science, ICCS 2017, 12–14 June 2017, Zurich, Switzerland
4. Kolm, P.N., Tütüncü, R., Fabozzi, F.J.: 60 years of portfolio optimization: practical challenges and current trends. Eur. J. Oper. Res. **234**(2), 356–371 (2014)
5. Konno, H., Yamazaki, H.: Mean absolute deviation portfolio optimization model and its applications to Tokyo stock market. Manag. Sci. **37**, 519–529 (1991)
6. Mansini, R., Ogryczak, W., Speranza, M.G.: Conditional value at risk and related linear programming models for portfolio optimization. Ann. Oper. Res. **152**(1), 227–256 (2007)
7. Mansini, R., Ogryczak, W., Speranza, M.G.: Twenty years of linear programming based portfolio optimization. Eur. J. Oper. Res. **234**(2), 518–535 (2014)
8. Mansini, R., Ogryczak, W., Speranza, M.: Linear and Mixed Integer Programming for Portfolio Optimization. EURO Advanced Tutorials on Operational Research. Springer, Switzerland (2015)
9. Markowitz, H.: Portfolio selection. J. Financ. **7**(1), 77–91 (1952)
10. Rockafellar, R.T., Uryasev, S.: Optimization of conditional value-at-risk. J. Risk **2**(3), 21–41 (2000)

11. Woodside-Oriakhi, M., Lucas, C., Beasley, J.: Portfolio rebalancing with an investment horizon and transaction costs. Omega **41**(2), 406–420 (2013)
12. Yitzhaki, S.: Stochastic dominance, mean variance, and Gini's mean difference. Am. Econ. Rev. **72**(1), 178–185 (1982)
13. Young, M.R.: A minimax portfolio selection rule with linear programming solution. Manag. Sci. **44**(5), 673–683 (1998)
14. Yu, Y.: A preliminary exploration on stochastic dynamic asset allocation models under a continuous-time sticky-price general equilibrium. Appl. Econ. **51**(4), 373–386 (2019)

Portfolio Leverage in Asset Allocation Problems

Mario Maggi and Pierpaolo Uberti

Abstract In the classical portfolio optimization framework, the leverage of a portfolio is not taken into account and, by assumption, the risk of a portfolio is totally described by the volatility of its returns. As a consequence, the portfolios on the classical mean-variance efficient frontier are not indifferent in terms of leverage. The introduction of leverage measurement in portfolio theory permits to consider other kinds of risk, like margin calls, forced liquidations at undesired prices and losses beyond the total capital. The literature on this topic is very limited while portfolio leverage is of central importance, in particular to set up operative investment strategies. In this paper we propose a simple definition of leverage and we try to introduce it in the classical portfolio selection scheme. We define the concept of leverage free equivalent portfolios in order to compare different investment alternatives for given levels of leverage. The central result of the paper is that the leverage free equivalent of the classical mean-variance efficient portfolios do not preserve the original mean-variance dominance structure. This permits to discriminate if an increase in the expected return of a portfolio totally depends on the leverage effect or is a consequence of a more efficient allocation.

Keywords Portfolio optimization · Portfolio leverage · Asset allocation

M. Maggi (✉)
Department of Economics and Management, Università di Pavia, Pavia, Italy
e-mail: mario.maggi@unipv.it

P. Uberti
DIEC Department of Economics, Università di Genova, Genova, Italy
e-mail: uberti@economia.unige.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_5

1 Introduction

Classic Portfolio Theory is derived using mean-variance optimization in order to calculate portfolio's optimal weights on the basis of expected returns, variances, and covariances. The resulting portfolios are mean-variance efficient and provide the maximum expected return for a given level of expected risk, or minimum expected risk once fixed the expected return [6]. The popularity of this approach testifies its acceptance over the years. However, the nature of financial markets and investment portfolios has changed considerably since 1952: the use of futures, options and leveraged debt instruments facilitate financial leverage. In particular, for hedge funds and investment banks, leverage has sometimes reached extreme and untenable levels.

The mean-variance model does not consider the leverage when comparing different portfolios. As a consequence, the classic approach is silent on the risks that are peculiar when using leverage such as the costs of margin calls, which can force borrowers to liquidate securities at adverse prices due to illiquidity, losses exceeding the capital invested and the possibility of bankruptcy [3].

Mean-variance optimization implicitly assumes that the investor has no aversion to the particular risks related to leverage positions. This results in arbitrarily large amounts of leverage. For an investor who does not consider leverage, mean-variance analysis provides optimal long-short portfolios. On the other hand, if an investor takes leverage into account, mean-variance analysis is inadequate. Actually, investors are in general averse to leverage: if two portfolios are indifferent in terms of expected return and variance, a rationale investor would prefer the portfolio with the lower level of leverage.

In [1], some restriction on borrowing are introduced to analyze the market equilibrium. The literature on the introduction of leverage in the portfolio selection scheme is recent and very limited. Jacobs and Levy [3] suggest to include a term for leverage aversion in the mean-variance utility function, turning it into a mean-variance-leverage utility function; this approach introduces in addition to the risk-aversion term, the leverage-aversion term. The mean-variance-leverage utility function lets investors trade off expected return with volatility risk and leverage risk. With mean-variance-leverage optimization, the efficient portfolios belong to a three-dimensional mean-variance-leverage surface [4] and the optimal individual portfolio depends on the individual tolerance for volatility and leverage. Each leverage tolerance level corresponds to a two-dimensional mean-variance efficient frontier. The introduction of leverage aversion in portfolio selection results in a general reduction of leverage compared to conventional mean-variance analysis. Less leveraged portfolios may be beneficial not only for leverage-averse investors, but also for the entire economy, considering that enormous levels of leverage exacerbated several financial catastrophes and systemic events [5, 7, 9]. In operative asset manager problems, Clarke et al. [2] consider limited short positions; they point out that fund managers often face borrowing and leverage constraints. Also Sorensen et al. [8] explicitly consider the introduction of constrained short positions.

In this paper we propose a model that reflects the actual operative management of an investment fund. A portfolio can be indicated by a vector $x \in \mathbb{R}^n$, where each component describes the proportions of the individual wealth invested in asset i, $i = i, \ldots, n$. We interpret the portfolio weights as speculative positions such that the signs of the weights reflect the bet on the asset price direction while the absolute weights are the size of the bet on a certain asset class. When investing in practice, the negative positions (i.e. the bets on a future decrease of the price) are usually taken through the use of derivatives, resulting in a positive allocation of capital to margin the position. This interpretation is opposite compared to the classic one, where the negative position on one asset is interpreted as a short selling, resulting in a cash inflow that can be invested in long position in excess to the available wealth. In particular, using futures or options, short positions in portfolios allocation can be assimilated to simply betting on a decrement of the price without cashing money to overweight long positions. Therefore, in our settings it seems natural to consider the absolute values of the weights.[1] Consequently, we remove the budget constraint (i.e. the requirement that the sum of the weights equals 1), and we introduce a definition of portfolio leverage that partially differs from the one proposed in [3]. We assume that the liquidity is used to manage difference between 1 and the sum of the absolute value of the portfolio weights. When two vectors of n components differ for a multiplicative constant, i.e. the relative proportions between the risky assets is equal, the corresponding portfolios may be considered somehow equivalent. Roughly speaking, they differ in terms of leverage while qualitatively their allocation in the risky assets is equivalent.

Following this idea, we propose to define the leverage of a portfolio considering the sum of the absolute weights of the risky positions. In these settings, for each portfolio, the portfolio with the same relative positions and with null leverage is defined leverage-free equivalent. The central finding of the paper is the analysis of the set of the leverage-free equivalent portfolios obtained starting from the mean-variance efficient portfolios. In this case, the leverage-free equivalent portfolios do not necessarily preserve the dominance structure of the original efficient portfolios, giving the investor the possibility to trivially choose to invest in portfolios with higher expected return and lower risk.

The paper is organized as follows: Sect. 2 contains the theoretical formalization of the proposal, Sect. 3 provides some empirical evidence of the results and Sect. 4 concludes the paper.

[1] Another common technique in order to transform a short position in pseudo long position is to take the short position buying an ETF-short, an exchange traded fund that replicates the desired underline when its price is decreasing.

2 The Theoretical Proposal

Let x be an column vector whose components $x_1, \ldots, x_n$ denote the proportion of wealth allocated to the ith risky asset, with $i = 1, 2, \ldots, n$; $\mathbf{1}$ is the column vector of ones, and transposition is denoted by the superscript T. We assume that x is composed by n risky assets and that $x_0 = 1 - \sum_{i=1}^{n} |x_i|$ is the residual proportion of wealth allocated in the risk free asset, or in liquidity. The sign of x_0 has to be interpreted in the standard way: when x_0 is positive, part of the initial wealth is not invested in risky positions, while, on the opposite, when x_0 is negative, some money is borrowed to overweight risky allocations. We assume that the liquidity has a null risk free rate of return. The asset returns are collected in the random vector $\tilde{R}$ with expected return R (column) and covariance matrix V. The matrix V is symmetric, positive definite and so non-singular. For notational convenience, we refer to the portfolio as represented directly by the vector x of the risky components, knowing that the effective investment portfolio is obtained considering the residual allocation in x_0.

Definition 1 (Leverage of a Portfolio) Given the portfolio x, its leverage $L(x)$ is defined as:

$$L(x) = \sum_{i=1}^{n} |x_i|$$

Obviously $L(x) \geq 0$. A portfolio with $L(x) > 1$ is leveraged, because the total amount of the positions overcome the investor's total wealth and, consequently, $x_0 < 0$.

Definition 2 (Leverage Free Portfolios) A portfolio x is leverage free if $L(x) = 1$.

From Definition 2, a portfolio is considered leverage free if the total wealth is invested in risky positions.

Proposition 1 *Let x be a vector such that the budget constraint holds, i.e. $x^T \mathbf{1} = 1$, then $L(x) \geq 1$; $L(x) = 1$ if and only if $x_i \geq 0$ for $i = 1, \ldots, n$.*

Proof The first part of the proposition is proved observing that $|x_i| \geq x_i$ for $i = 1, \ldots, n$, so we obtain that

$$L(x) = \sum_{i=1}^{n} |x_i| \geq \sum_{i=1}^{n} x_i = 1.$$

If $x^T \mathbf{1} = 1$ and $x_i \geq 0$ for $i = 1, \ldots, n$, then $x_i = |x_i|$ for $i = 1, \ldots, n$ and

$$L(x) = \sum_{i=1}^{n} |x_i| = \sum_{i=1}^{n} x_i = 1.$$

On the other hand, if $x^T \mathbf{1} = 1$, $L(x) = 1$ and it would exist an index $1 \leq j \leq n$ such that $x_j < 0$, then $\sum_{i=1, i\neq j}^{n} x_i + x_j = 1$, from which we obtain $\sum_{i=1, i\neq j}^{n} x_i > 1$. The leverage $L(x)$ for this portfolio would be

$$L(x) = \sum_{i=1}^{n} |x_i| \geq \sum_{i=1, i\neq j}^{n} x_i + |x_j| > \sum_{i=1, i\neq j}^{n} x_i > 1,$$

which is a contradiction. □

Proposition 1 relates the budget constraint to the concept of leverage, underlying that, when the budget constraint is assumed to hold, by construction $L(x) \geq 1$.

It is interesting to note that:

- In general, asset allocation models that assume the budget constraint as a restriction on assets weights do not control the leverage with the potential consequence of comparing portfolios with huge differences in the leverage.
- A model that compares portfolios with the same leverage is the one that assume both the budget constraint and the restrictions of non-negativity of the weights. In this case, the portfolios have a constant leverage equal to 1.

Definition 3 (Equivalent Portfolios) Given two portfolios x and y, they are defined to be equivalent, $x \equiv y$ if

$$\frac{r_x}{\sigma_x} = \frac{r_y}{\sigma_y}$$

where r_x and σ_x are respectively the expected return and the standard deviation of the returns of x.

Definition 3 provides a very intuitive condition for portfolios equivalence on the basis of their Sharpe ratio.

Proposition 2 *If* $\exists \lambda \in \mathbb{R} : x = \lambda y$, *the portfolios* x *and* y *are equivalent in the sense of Definition 3.*

Proof If $x = \lambda y$, then $r_x = \lambda r_y$ and $\sigma_x = \lambda \sigma_y$ and we trivially have that

$$\frac{r_x}{\sigma_y} = \frac{\lambda r_y}{\lambda \sigma_y} = \frac{r_y}{\sigma_y}.$$

Proposition 2 formalizes the intuition explained in the introduction. A portfolio describes the relative proportion of wealth invested in each asset; if the relative proportions of the risky constituents of two portfolios are equal, they have the same Sharpe ratio. As a consequence two equivalent portfolios differ only for their leverage.

Following the idea described in the introduction, we compare portfolios with a given level of leverage, in order to consider the correct ranking between risk-return alternatives. A convenient choice is to set $L(x) = 1$.

Proposition 3 *Given a portfolio x, the equivalent leverage free portfolio is $\overline{x} = \left(\frac{1}{L(x)}\right) x$.*

Proof From Proposition 2 we obtain that $x \equiv \overline{x}$. By construction, the leverage of $\overline{x}$ is:

$$L(\overline{x}) = \sum_{i=1}^{n} |\overline{x}_i| = \frac{1}{L(x)} \sum_{i=1}^{n} |x_i| = \frac{L(x)}{L(x)} = 1.$$

So the proposition is proved. □

It is interesting to compare the efficient portfolios belonging to the standard mean-variance frontier to their corresponding equivalent leverage free portfolios. We can define the two sets of portfolios as follows.

Definition 4 The sets of mean variance efficient portfolios and the equivalent leverage free portfolios, respectively X_{mv} and $\overline{X}_{lf}$, are:

$$X_{mv} = \{x : x = x(\sigma_0), \forall \sigma_0 \geq \sigma_v\} \tag{1}$$

where $x(\sigma_0)$ is the solution of the optimization problem

$$\begin{aligned} &\max_{x} R^t x \\ &\text{s.t. } x' V x = \sigma_0 \\ &\quad x' \mathbf{1} = 1 \end{aligned} \tag{2}$$

and $\sigma_v = \arg \min_{x'\mathbf{1}=1} \left\{x' V x\right\}$;

$$\overline{X}_{lf} = \left\{x : x = \frac{1}{L(y)} y \text{ and } y \in X_{mv}\right\}. \tag{3}$$

One first interesting result links the risk measured by the variance of returns to the risk described by the leverage.

Proposition 4 *The set $\overline{X}_{lf}$ is bounded in terms of standard deviation and expected return.*

Proof Let define

$$X^*_{lf} = \left\{x : L(x) = 1, \forall x \in R^n\right\} \tag{4}$$

and note that it is a closed subset of the hypersphere in R^n with center the origin and unitary radius, $H_1(\mathbf{0})$. By construction we have:

$$\overline{X}_{lf} \subset X^*_{lf} \subset H_1(\mathbf{0})$$

so that $\overline{X}_{lf}$ is limited being a subset of the limited set $H_1(\mathbf{0})$. Since the set $\overline{X}_{lf}$ is closed and the functions $\sigma(x)$ and $r(x)$ are continuous, for the Weierstrass theorem, both the functions admit minimum and maximum on the domain $\overline{X}_{lf}$. □

The interpretation of Proposition 4 is intuitive: once we introduce a restriction on the leverage of the portfolio, the risk an investor can take is automatically bounded and also the expected return is limited. The result is well known in some special cases: for example, when in the classic optimization problem, see Eq. (2), only long-only positions are allowed, i.e. $x_i \geq 0$ for $i = 1, \ldots, n$, producing a portfolio that is a convex linear combination of the securities.

The principal consequence of this new approach is on the structure of mean-variance dominance of the portfolios in the set $\overline{X}_{lf}$. The elements of X_{mv} are mean-variance efficient: none of the portfolios in X_{mv} dominates any other portfolio of the set, in mean-variance terms. On the other hand, if we consider the set $\overline{X}_{lf}$, the structure of mean variance dominance changes. In fact, it is possible to find at least two portfolios x and y in $\overline{X}_{lf}$ such that $\sigma(x) \leq \sigma(y)$ and $r(x) \geq r(y)$, with at least one strict inequality holding. In next section we provide some evidence.

The operation that replaces the classic mean-variance efficient portfolios with the equivalent leverage free portfolios causes an unexpected consequence on the set $\overline{X}_{lf}$: if we purify the efficient portfolios from the leverage component, some portfolios in $\overline{X}_{lf}$ are dominated by other portfolios in the same set. The intuition behind the result is very interesting: starting from a certain point on, the portfolios on the efficient frontier permit to reach the desired level of expected return (and the associated risk), simply increasing the leverage. This answers the question if the extra performances of certain portfolios depend only on an increase of the leverage or there is some more useful information beside the leverage. The fact that in $\overline{X}_{lf}$ some portfolios are dominated may change the role of the individual utility function in the decision framework.

3 Empirical Evidence

In this section we show, both on real and simulated data, the results described in the theoretical section. Figure 1 shows the X_{mv} set on the mean-variance plane and the corresponding level of leverage of each single portfolio on the efficient frontier. The efficient frontier is drawn on the base of daily returns of the ten sectors of S&P 500 from January 1991 to August 2011. As described in the previous section, the portfolios on the frontier can have different leverage values. In particular, from Fig. 1 (bottom pane) it is possible to note that the leverage is not smaller than 1,

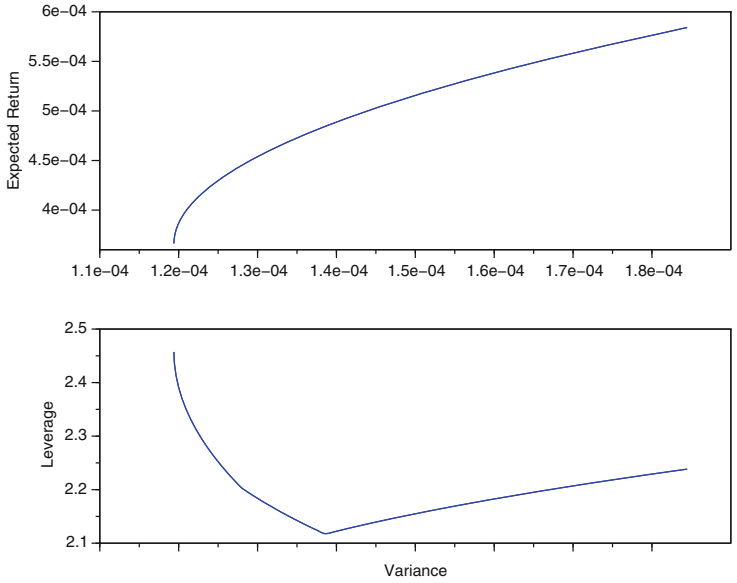


Fig. 1 The mean variance frontier and the corresponding leverage level of the efficient portfolios, based on the 10 sector indexes of the S&P 500 from January 1991 to August 2011

consistently with Proposition 1. Moreover, in practice, the expected return of a portfolio can be indefinitely increased, expanding its leverage. Figure 1 displays the efficient frontier X_{mv} in the variance-mean plane (top pane), together with its leverage value $L(x)$ (bottom pane). The mean-variance combinations of the leverage free equivalent portfolios belonging to the set $\overline{X}_{lf}$ are shown in Fig. 2.

From Fig. 2 it is possible to note that the set $\overline{X}_{lf}$ is limited and the frontier traces back, following a different path. As a consequence, some portfolios dominate other portfolios of the same set. In other words, without considering the leverage, some portfolios show an higher return and a lower variance. We can therefore conclude that the removal of leverage affects the mean-variance dominance of the portfolios.

To present some possible shapes of the set $\overline{X}_{lf}$, we also consider some simulated cases. We generate a sample of 1000 observations from $n = 10$ assets with independent, identically and normal distributed returns. Then, we compute the efficient frontier X_{mv} and the corresponding set $\overline{X}_{lf}$. We note that, as proved in the theoretical section, the frontiers are not concave and some portfolios dominate other portfolios. In particular, these two last aspects need a more rigorous empirical investigation. It is interesting to point out that in Fig. 3 two different qualitative behavior are present: in subfigures (I), (II) and (III) the frontier reaches the maximum risk level and then traces back dominating. In words, portfolios associated to high risk and return reaches higher performances not only due to the leverage effect. On the opposite, in subfigure (IV) the frontier traces back but the portfolios are dominated, underlying that in this case the extra risk and return were totally

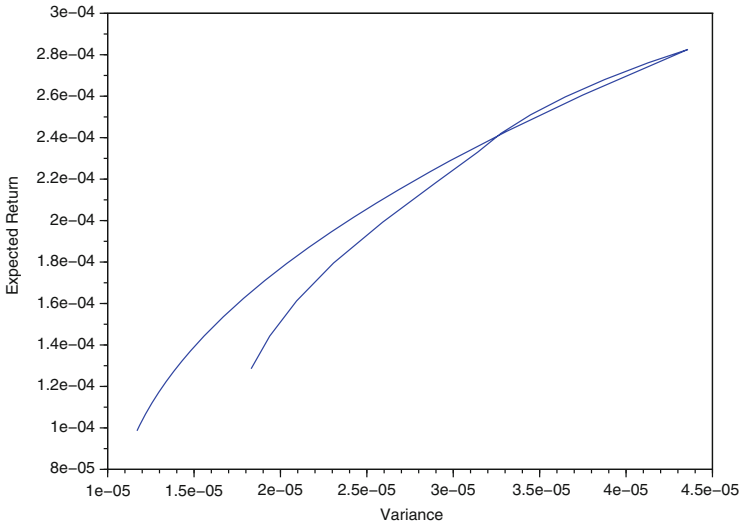


Fig. 2 The shape of the set $\overline{X}_{lf}$ on the mean-variance plane, based on the 10 sector indexes of the S&P 500 from January 1991 to August 2011

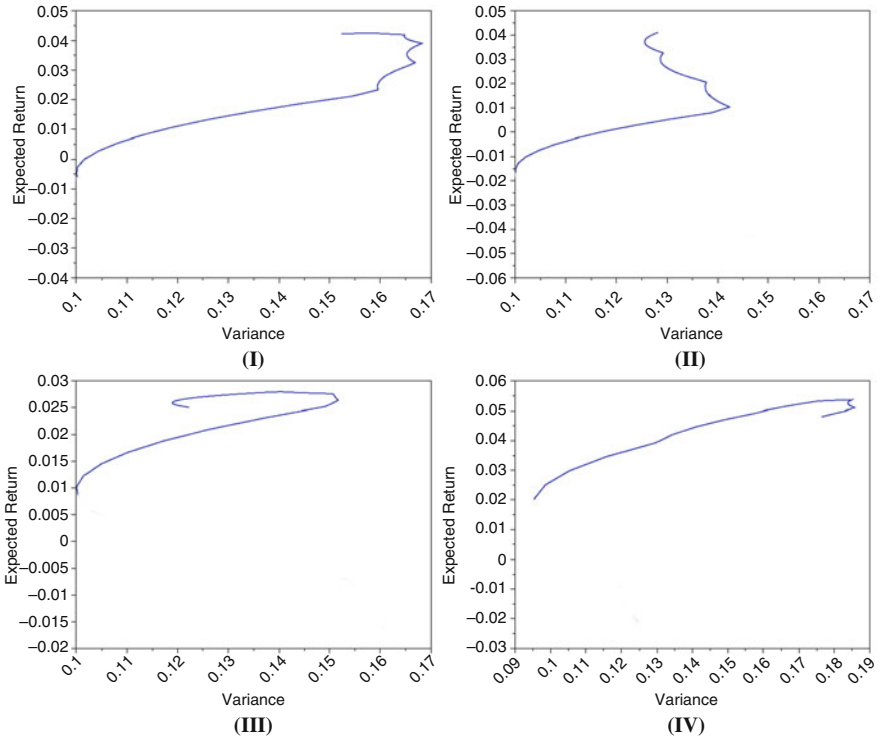


Fig. 3 The shapes of $\overline{X}_{lf}$ for $n = 10$ and four different independent samples

determined by the leverage effect and the portfolios are not interesting once deleveraged.

4 Conclusion

The mean-variance model does not take into account leverage and the associated risks, when comparing different investment portfolios. In this paper we propose a definition of leverage and a technique to incorporate it in the decision scheme. The main result of the paper is the modification on the dominance structure of the portfolio set obtained by depurating the efficient portfolios from leverage. From a practical point of view, the paper provides a realistic and useful approach to manage and control portfolio leverage. Moreover, we show that some classic mean-variance portfolios are of great interest because their performances depend not only by an increase in leverage. Further research is needed in order to rigorously test the results from an empirical point of view and to relate the size n of the portfolio and the correlation between its constituents to the shape of the set $\overline{X}_{lf}$ in the mean-variance plane.

References

1. Black, F.: Capital market equilibrium with restricted borrowing. J. Bus. **45**(3), 444–455 (1972)
2. Clarke, R., De Silva, H., Thorley, S.: Portfolio constraints and the fundamental law of active management. Financ. Anal. J. **58**(5), 48–66 (2002)
3. Jacobs, B., Levy, K.: Leverage aversion and portfolio optimality. Financ. Anal. J. **68**(5), 89–94 (2012)
4. Jacobs, B., Levy, K.: Leverage aversion, efficient frontiers, and the efficient region. J. Portf. Manag. **39**(3), 54–64 (2013)
5. Jacobs, B., Levy, K., Markowitz, H.: Portfolio optimization with factors, scenarios, and realistic short positions. Oper. Res. **53**(4), 586–599 (2005)
6. Markowitz, H.M.: Portfolio selection. J. Financ. **7**(1), 77–91 (1952)
7. Markowitz, H.M.: How to represent mark-to-market possibilities with the general portfolio selection model. J. Portf. Manag. **39**(4), 1–3 (2013)
8. Sorensen, E.H., Hua, R., Qian, E.: Aspects of constrained long-short equity portfolios. J. Portf. Manag. **33**(2), 12–20 (2007)
9. Sullivan, R.N.: Speculative leverage: a false cure for pension woes. Financ. Anal. J. **66**(3), 6–8 (2010)

Energy-Efficient Train Control

A Practical Application

Valentina Cacchiani, Antonio di Carmine, Giacomo Lanza, Michele Monaci, Federico Naldini, Luca Prezioso, Rosalba Suffritti, and Daniele Vigo

Abstract This research presents a practical application of the *Energy Efficient Train Control* (EETC) problem, which involves a collaboration between the Operations Research group of the University of Bologna and ALSTOM Ferroviaria SpA. The work is carried out within the framework of project *Swift*, funded by the Emilia-Romagna regional authority. Given a train running on a certain line, the problem requires to determine a speed profile that minimizes the traction energy consumption. In particular, we consider the setting of a real-time application, in which the speed profile has to be recomputed due to changes in the schedule caused by unpredictable events. We introduce three solution methods: a constructive heuristic, a multi-start randomized constructive heuristic, and a Genetic Algorithm. Numerical experiments are performed on real-life instances. The results show that high quality solutions are produced and that the computing time is suitable for real-time applications.

Keywords Heuristic · Railway optimization · Energy · Train control · Real-time

1 Introduction

One of the major costs for railway companies is given by energy consumption. For this reason, the development into a more mature and competitive market makes an efficient energy management imperative for reducing the operating costs. Ecological

V. Cacchiani · G. Lanza · M. Monaci · F. Naldini (✉) · D. Vigo
DEI, Università di Bologna, Bologna, Italy
e-mail: valentina.cacchiani@unibo.it; giacomo.lanza2@unibo.it; michele.monaci@unibo.it; federico.naldini4@unibo.it; daniele.vigo@unibo.it

A. di Carmine · L. Prezioso · R. Suffritti
Alstom Ferroviaria SpA, Bologna, Italy
e-mail: antonio.di-carmine@alstomgroup.com; luca.prezioso@alstomgroup.com; rosalba.suffritti@alstomgroup.com

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_6

awareness is also a major driver for energy efficiency in railway systems in an effort towards reducing air pollutants, e.g., carbon dioxide, whose emissions are one of the causes of global warming [10].

The construction of energy-efficient driving profiles has attracted large attention from researchers in the recent years, being an effective way of saving energy. The resulting problem is known as *Energy-Efficient Train Control* (EETC), and is sometimes also referred to as *eco-driving* or *train trajectory planning problem*. This problem aims at finding the most energy-efficient driving profile for a given train traveling on a certain line, while satisfying a number of operational constraints to ensure a safe and punctual journey. Recent surveys on EETC have been proposed by Yang et al. [16] and Scheepmaker et al. [13]. A complete review on the Optimal Train Control Theory has been given in [1, 2].

Analyses based on the *Pontryagin Maximum Principle (PMP)* have shown that an optimal driving strategy consists of a sequence of four *driving regimes*, namely, maximum acceleration/traction (MT), cruising/speed-holding (SH), coasting (CO) and maximum braking (MB), see, e.g., Howlett et al. [8] and Albrecht et al. [1, 2]. An example speed profile consisting of MT-SH-CO-MB is shown in Fig. 1.

Based on this result, most of the algorithms proposed in the literature [2, 13, 16] define the driving profile as a sequence of these four driving regimes, identifying suitable switching points between consecutive driving regimes.

In this paper we focus on the EETC in its basic version, arising when the driving profile of a single train has to be determined. We do not consider railway traffic management, as we assume that the schedule of the train has already been optimized by a (possibly online) scheduling algorithm (see, e.g., Bettinelli et al. [3] and Fischetti and Monaci [6]). In this context, safety requirements impose that the schedule of the train is given on input and cannot be changed. In addition, we do not consider possible interactions between trains in terms of power exchange, as it could happen in complex networks equipped with appropriate infrastructures and energy storage systems.

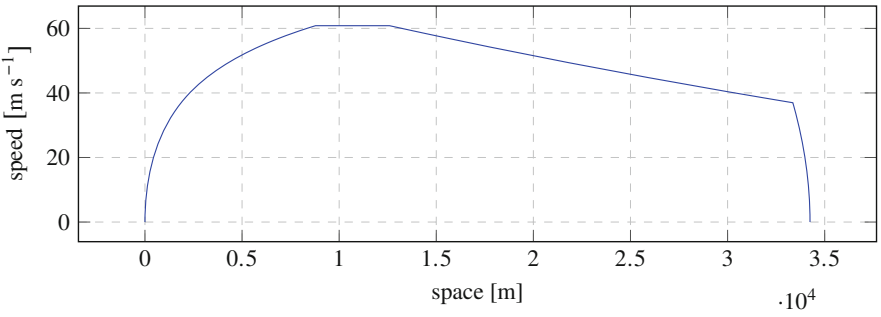


Fig. 1 A profile consisting of MT, SH, CO and MB. The train runs on a track approximately 35 kilometers long, and starts and ends at zero speed

This paper is organized as follows. In Sect. 2 we define the problem, and in Sect. 3 the solution approaches are outlined. The results of the computational testing on real-world instances are given in Sect. 4. Finally, Sect. 5 draws some conclusions.

2 Problem Definition

We are given a rail track having length D, and a train running on the track. The train has a fixed schedule, imposing its travel time be equal to exactly T time units, and is characterized by some known rolling-stock properties (e.g., its weight). The track geometry, e.g., the position-varying slope and radius of curvature, is also known. Finally, there are given speed limits that the train must respect, and that vary along the track. The problem requires to determine a driving profile for the train such that (1) the travel time of the train is exactly T; (2) speed limits are respected; and (3) the total amount of energy required for running the train is minimized.

We represent the track as a segment $[0, D]$ and assume that the train travels on the track from time instant 0 to time instant T.

For the sake of simplicity, we do not consider the case of *steep* uphill/downhill tracks, i.e., we assume that the train, regardless of its speed, is always capable of negotiating uphills/downhills or speed-holding. Moreover, we assume a continuous control and neglect any form of energy recovery, as it happens with the so-called *Regenerative Brake*. For details on steep uphills/downhills and on Regenerative Brake, the reader is referred to [1, 2, 8].

In our setting, the motion of a train is approximated using the following point-mass model

$$\frac{dv}{dt} = u(t) - R(v(t)) + G(x(t)) \tag{1}$$

$$\frac{dx}{dt} = v(t), \tag{2}$$

where time $t \in [0, T]$, is the independent variable, while speed $v(t)$, position $x(t)$ are coordinates of the dynamical system and $u(t)$ is the controlled acceleration.

Term $R(v)$ represents the so-called *Basic Resistance*, taking into account several resistive phenomena depending only on the given rolling-stock. In particular, it includes rolling resistances, mechanical resistances that are proportional to speed (e.g., rotation of axels and shaft, mechanical transmission, etc.), and air resistance (see, Profillidis [11]).

Basic resistance is typically approximated using the well-known *Davis Equation*

$$R(v) = r_0 + r_1\, v + r_2\, v^2, \tag{3}$$

where r_0, r_1 and r_2 are positive empirical constants depending on the rolling-stock (see, Davis Jr [4]).

Term $G(x)$ is the *Line Resistance*, which measures the position-dependent forces acting on the train along its route. This resistance considers the horizontal component of gravity (depending on the position-varying track slope along the track) and the *curve resistance* (depending on the position-varying radius of curvature along the track). The analytic expression of $G(x)$ has been provided by our industrial partner as a piece-wise function through the empirical formula:

$$G(x) := -g \ \sin\left[\arctan\left(\frac{0.8}{\rho(x)} + \tan\theta(x)\right)\right], \tag{4}$$

$\theta(x)$ and $\rho(x)$ being the (position-dependent) track grade and radius of curvature respectively, and g the gravitational acceleration (see, Fayet [5] and Sapronova et al. [12]).

The controlled acceleration $u(t)$ is bounded by two functions, i.e.

$$-u_L(v) \leq u(t) \leq u_U(v). \tag{5}$$

where bounds $u_L(v) \geq 0$ and $u_U(v) \geq 0$ are the *maximum deceleration* and the *maximum acceleration*, respectively, that the train engine can handle. In the following, we will always assume that u_L and u_U are decreasing non-linear functions of the speed.

As mentioned above, speed limits impose an upper bound on the maximum speed

$$0 \leq v(t) \leq \bar{V}(x(t)) \tag{6}$$

where the upper bound value $\bar{V}(x(t))$ depends on the geometry of the track. In addition, the following boundary conditions

$$x(0) = 0, \qquad x(T) = D, \qquad v(0) = v_{init}, \qquad v(T) = v_{final}, \tag{7}$$

are imposed to fix the position and speed of the train at time instants $t = 0$ and $t = T$, with v_{init} and v_{final} given on input.

Finally, the objective is to minimize the total energy spent by the train, given by

$$E = \int_0^T \frac{u(t) + |u(t)|}{2} v(t) dt. \tag{8}$$

Equation (8) is derived from the definition of *work* done on a point mass. Since we do not account for energy recovery, negative values of u must be ignored. We therefore introduce $(u + |u|)$ so that energy contribution is zero whenever $u < 0$. The reader is referred to Albrecht et al. [1] and Scheepmaker et al. [13] for further details.

Observe that Eqs. (1)–(8) constitute a well-known Optimal Control Formulation for the EETC, see Albrecht et al. [1, 2] and Scheepmaker et al. [13].

2.1 Overview of the Solution Approach

In most practical cases, track geometry and speed-limits are described as piecewise constant functions of the space. Therefore, it is possible to partition the track $[0, D]$ into a sequence of n consecutive sections $S_k = [x_o^{(k)}, x_f^{(k)}]$, with $x_o^{(0)} = 0$, $x_f^{(n)} = D$ and $x_f^{(k)} = x_o^{(k+1)}$, $(k = 0, 1, \ldots, n-1)$, that have constant slope, radius of curvature and speed limit (see [7, 14, 15]). We will refer to those sections as *segments*. This results in having a fixed *Line Resistance* along with a fixed speed limit. In particular $\forall\ k = 1 \ldots n$:

$$G(x) = g^{(k)} \qquad \bar{V}(x) = \bar{V}^{(k)}. \tag{9}$$

As mentioned in the introduction, similar to what is done in [9], our approach is based on the Optimal Train Control theory and assumes that a concatenation of at most four given driving regimes is used to define the speed profile in each section. The driving regimes are characterized by the following conditions:

$$\text{Maximum Traction, MT:} \qquad u_1 = u_U(v) \tag{10}$$

$$\text{Speed Holding (or Cruising), SH:} \qquad u_2 = R(v) - g^{(k)} \tag{11}$$

$$\text{Coasting, CO:} \qquad u_3 = 0 \tag{12}$$

$$\text{Maximum Braking, MB:} \qquad u_4 = -u_L(v), \tag{13}$$

where u_1, u_2, u_3 and u_4 are the accelerations in every regime. Within a segment, each regime can be executed at most once, and regimes must appear in fixed order, namely, MT, SH, CO, and MB. Furthermore, some regimes may not be used in a segment.

As a consequence, Eqs. (1) and (2) can be simplified, because the term u corresponds to one of the u_j $(j = 1, \ldots 4)$ shown in Eqs. (10)–(13) and $G(x) = g^{(k)}$ in each segment k.

We observe that, within our settings, energy is only consumed during the MT and SH driving regimes. Let $E_{MT}^{(k)}$ and $E_{SH}^{(k)}$, respectively, be the energy consumed in the MT and SH regimes for segment k $(k = 1 \ldots n)$. Then Eq. (8) reduces, for MT, to

$$E_{MT}^{(k)} = \int_{t_1^{(k)}}^{t_2^{(k)}} u_1(v(t))\ v(t)\ dt, \tag{14}$$

where $t_1^{(k)}$ and $t_2^{(k)}$ are the start and end time for MT in segment k. Similarly, for regime SH, we have that (8) simplifies to

$$E_{SH}^{(k)} = u_2(v_2^{(k)})\ v_2^{(k)}(t_3^{(k)} - t_2^{(k)}), \tag{15}$$

where $v_2^{(k)}$ is the constant speed value, while $t_2^{(k)}$ and $t_3^{(k)}$ are respectively the start and end times for SH regime.

We note that, Eqs. (1), (2) and (14), can be pre-computed, for each segment and driving regime, by using numerical methods or closed-form expressions under some assumptions on u as in [17].

3 Heuristic Solution Approaches

In our application, train control has to be determined in short computing times, in order to take into account possible changes in the schedule of the train due to some unpredictable events. For this reason, we propose heuristic approaches for its solution. In this section, we first introduce a constructive heuristic (CH), then, we present a multi-start randomized constructive heuristic (RCH) and a Genetic Algorithm (GA).

3.1 Constructive Heuristic

The proposed constructive heuristic is an iterative procedure that starts from an infeasible solution and, at each iteration, tries to reduce infeasibility, until a feasible solution is produced.

At the beginning, the algorithm computes the so-called *allout* speed profile, i.e. the solution obtained by running the train at its maximum allowed speed. When computing this speed profile, train motion laws and speed limits are taken into account.

Observe that the running time τ_{AO} associated with this solution provides a lower bound on the running time for the train in any feasible solution. If this running time is not equal to the required time T, the current solution is infeasible. To recover feasibility we consider the actual maximum speed $\tilde{V}^{(k)}$ reached in every segment k, and slow the train down by artificially reducing the speed limits. This speed reduction is applied in one segment at a time.

For each segment k, we reduce its maximum speed by a given amount s (which is a parameter of the algorithm), and re-compute the associated speed profile and running time τ. If $\tau > T$, then the last update of the maximum speed is canceled. If, instead $\tau < T$, the next segment is considered. This procedure continues until a feasible solution is obtained. The pseudo-code of the algorithm is shown in Algorithm 1.

```
1  Compute the allout speed profile for the given route. Store travel time in τ = τ_AO.
   Store in Ṽ^(k) the maximum speeds that are actually reached by the allout speed profile
   in each segment k;
2  Set w_k := Ṽ^(k), ∀k = 1...n;
3  while τ < T do
4  |  for k = 1...n do
5  |  |  w_k := w_k − s;
6  |  |  Re-compute speed profile using the current speed limits in each segment (i.e.
   |  |  w_1, ... w_n). Store travel time in τ;
7  |  |  if τ > T then
8  |  |  |  Undo the change, namely w_k := w_k + s
9  |  |  end
10 |  end
11 end
```

Algorithm 1: CH

3.2 *Multi-Start Randomized Constructive Heuristic*

The constructive heuristic of the previous section is used in a multi-start fashion to determine a pool of feasible solutions. This allows to possibly find better solutions and to determine an initial population of the genetic algorithm (see Sect. 3.3).

The multi-start randomized constructive heuristic performs a fixed number of iterations, changing a set of parameters in a random way. At each iteration, γ_1 segments are selected (where γ_1 is a parameter of the algorithm), and the maximum speed of the selected segments is reduced by a random value. If the travel time of the allout profile associated with the current w'_k values is not larger than T, the constructive heuristic of Sect. 3.1 is applied. Otherwise, the speed limits are re-defined, and the process is repeated.

The algorithm produces one feasible solution per iteration and is executed for NI (say) iterations. The best solution found among all iterations is then returned. The pseudo-code is reported in Algorithm 2.

1 Compute the *allout* speed profile for the given route. Store in $\tilde{V}^{(k)}$ the maximum speeds that are actually reached by the *allout* speed profile in each segment k;
2 Set $w_k := \tilde{V}^{(k)},\ \forall k = 1 \dots n$;
3 **for** $i = 1 \dots, NI$ **do**
4 **repeat**
5 set $w'_k = w_k$ for each segment k;
6 **for** $c = 1 \dots \gamma_1$ **do**
7 Randomly select a segment $\tilde{k}$ according to a uniform random distribution;
8 Define $w'_{\tilde{k}}$ as a random number within $(0, \gamma_2\ w_{\tilde{k}}]$ with $0 \leq \gamma_2 \leq 1$;
9 **end**
10 Re-compute speed profile using the current speed limits w'_k;
11 Store travel time in τ;
12 **until** $\tau \leq T$;
13 Set $w_k := w'_k,\ \forall k = 1 \dots n$, and apply lines 3-11 of **Algorithm 1**;
14 **end**

Algorithm 2: RCH

3.3 Genetic Algorithm (GA)

In this section, we present a Genetic Algorithm, inspired by the work of Li and Lo [9].

In our algorithm, each chromosome is associated with a solution and is represented by a vector $\mathbf{V}$ containing the values of the initial speed for each driving regime. In particular we have

$$\mathbf{V} = \left[v_j^{(k)} \right]_{\substack{k=1\dots n \\ j=1\dots 4}}. \tag{16}$$

The population contains M individuals. During each iteration, by means of crossover and mutation operators, the current population might grow larger than M. The selection operator restores the number of individuals to M by eliminating the lowest ranking ones from the current population. A fixed number h of *Elite* individuals is kept intact over the iterations.

The evaluation of each chromosome c of a current population is carried out by means of the following fitness function

$$f(c) = k_1 E(c) + k_2 H(c), \tag{17}$$

where k_1 and k_2 are tuning parameters, $E(c)$ represents the traction energy that is spent driving the train according to solution c, and $H(c)$ measures the diversity of chromosome c with respect to the rest of the current population (thus making

the fitness function biased towards favoring diversity, instead of considering energy efficiency only). $H(c)$ is computed as a mean difference between c and the other individuals.

The main components of the GA are reported in the following list, according to the order in which chromosomes are processed in each iteration:

1. **Mutation Operator:** this operator is applied, with probability P_m, to each non-elite chromosome, producing new individuals. It consists of applying random variations of speed to a set of randomly selected segments, according to a uniform random distribution. The maximum variation amount is given as a percentage m (parameter of the algorithm) of the maximum speed in a each segment. Preliminary experiments showed that better performances are achieved when these operators are not applied to elite chromosomes;
2. **Crossover Operator:** it implements a single-point crossover to produce two children chromosomes from a couple of parents, by recombining their genome after splitting on a randomly selected segment boundary. The crossover break point is randomly selected among all those genes that correspond to the end of some segment. This operator has a probability P_c of affecting each chromosome, excluding Elites and those chromosomes produced by mutation at the current iteration;
3. **Repair Operator:** it restores feasibility of solutions after mutation and crossover, if needed. It implements a procedure similar to the constructive algorithm introduced in Sect. 3.1
4. **Selection Operator:** it ranks each chromosome c of the current population by means of the fitness function.Then, it deletes the lowest ranking individuals exceeding the maximum population size M, while preserving a group of h Elites intact.

4 Computational Experiments

The algorithms described in the previous section were implemented in C and executed on a 2.9-Ghz Intel Core i7-7500U with 16 GB of RAM. Our benchmark is composed by real-world instances provided by ALSTOM. All instances are associated with two railway lines, denoted as ROUTE 1 and ROUTE 2. These lines have lengths equal to 250 and 270 km, respectively, and are composed by 42 and 67 segments, respectively. For each line, we considered five different train models, denoted with letters A to E in the following, and having different characteristics. This produced a benchmark of ten different instances. To evaluate the quality of the obtained solutions, we used the following measure of efficiency, commonly used by ALSTOM's practitioners:

$$e = 100 \cdot \frac{E_A - E}{E_A}, \tag{18}$$

Table 1 Results on ROUTE 1 (250 km, 42 segments)

Train	T (s)	Efficiency			CPU time (seconds)		
		CH	RCH	GA	CH	RCH	GA
A	4560	3.2	5.4	7.2	0.2	1.5	4.4
B	4560	8.2	10.4	11.6	0.1	1.2	2.5
C	6525	6.6	11.5	25.3	0.3	4.9	10.6
D	10,920	10.9	12.5	12.7	0.1	0.6	2.2
E	7079	18.8	20.2	21.6	0.2	0.7	2.1

Table 2 Results on ROUTE 2 (270 km, 67 segments)

Train	T (s)	Efficiency			CPU time (seconds)		
		CH	RCH	GA	CH	RCH	GA
A	7285	6.6	13.7	23.7	0.5	9.6	15.8
B	7142	8.7	14.8	29.0	0.6	11.4	21.7
C	7933	9.9	13.4	27.2	0.8	15.9	29.4
D	11,729	12.7	13.2	13.9	0.2	0.9	2.4
E	7595	18.2	18.9	22.2	0.3	1.7	5.2

where E_A represents the energy consumption of the aforementioned allout profile, and the term E corresponds to the energy consumption associated with the given solution.

The RCH algorithm (see Algorithm 2), which also initializes the GA, was iterated 32 times in each experiment in a multi-start fashion. Parameter s was set to 1 (m/s), $\gamma_1 = 20$ and $\gamma_2 = 40\%$. The GA was configured according to the following parameter values: $P_c = 0.97$, $P_m = 0.95$, $M = 9$, $h = 5$ and $m = 5\%$. A maximum number of 300 generations was imposed. Moreover the GA was set to terminate after 11 non-improving consecutive generations.

Table 1 shows the results associated with ROUTE 1. For each train model we report the fixed travel time T, the level of efficiency computed according to (18), and the associated CPU time (in seconds).

The results show that, when the first line is considered, the constructive heuristic is able to reduce the energy consumption, with respect to the allout profile, by around 10% on average. The associated computing times are rather small, being always below 2 s. By executing the randomized multistart constructive heuristic and the genetic algorithm we obtain further savings, about 12% and 15%, respectively. Although these improvements could be expected, as the algorithms are built "on-top" of each other, the additional savings that we obtained are quite significant. While the increase in CPU time for the randomized multistart constructive heuristic is rather limited, the genetic algorithm requires significantly larger computing times for some instances: in these cases, a time limit can be imposed in order to use this algorithm in a real-time system.

The results for the second line, reported in Table 2, confirm those obtained for the first line, although the computing times are larger than before. However, we observe that energy reduction in Table 2 is also higher than in Table 1.

5 Conclusions and Future Research

In this paper we studied the Energy-Efficient Train Control (EETC) problem. This problem was solved using three approaches: a constructive heuristic, a multi-start randomized constructive heuristic, and a genetic algorithm.

Computational results on real-life instances show that, in most cases, the computing times of the algorithms are short enough to allow their use in a real-time application.

Future research directions will be to consider real-time traffic management aspects, related to the possibility of slightly changing the schedule of the train, as well as the adoption of a system which allows energy recovery through regenerative brake.

Acknowledgements The research has been partially supported by Regione Emilia-Romagna, under the project POR-FESR 2014-2020 ASSE 1 "Ricerca e Innovazione" called SWIFT.

References

1. Albrecht, A., Howlett, P., Pudney, P., Vu, X., Zhou, P.: The key principles of optimal train control-part 1: formulation of the model, strategies of optimal type, evolutionary lines, location of optimal switching points. Transp. Res. B Methodol. **94**, 482–508 (2016)
2. Albrecht, A., Howlett, P., Pudney, P., Vu, X., Zhou, P.: The key principles of optimal train control-part 2: existence of an optimal strategy, the local energy minimization principle, uniqueness, computational techniques. Transp. Res. B Methodol. **94**, 509–538 (2016)
3. Bettinelli, A., Santini, A., Vigo, D.: A real-time conflict solution algorithm for the train rescheduling problem. Transp. Res. B Methodol. **106**, 237–265 (2017)
4. Davis Jr, J.W.: The tractive resistance of electric locomotives and cars. Gen. Electr. Rev. **29**, 2–24 (1926)
5. Fayet, P.: Modélisation des réseaux électriques ferroviaires alimentés en courant alternatif. PhD thesis, 2008
6. Fischetti, M., Monaci, M.: Using a general-purpose mixed-integer linear programming solver for the practical solution of real-time train rescheduling. Eur. J. Oper. Res. **263**, 258–264 (2017)
7. Haahr, J.T., Pisinger, D., Sabbaghian, M.: A dynamic programming approach for optimizing train speed profiles with speed restrictions and passage points. Transp. Res. B Methodol. **99**, 167–182 (2017)
8. Howlett, P., Milroy, I., Pudney, P.: Energy-efficient train control. Control Eng. Pract. **2**, 193–200 (1994)
9. Li, X., Lo, H.K.: An energy-efficient scheduling and speed control approach for metro rail operations. Transp. Res. B Methodol. **64**, 73–89 (2014)
10. Luijt, R.S., van den Berge, M.P., Willeboordse, H.Y., Hoogenraad, J.H.: 5 years of Dutch eco-driving: managing behavioural change. Transp. Res. A Policy Pract. **98**, 46–63 (2017)
11. Profillidis, V.: Railway Management and Engineering. Routledge (2016)
12. Sapronova, S.Yu., Tkachenko, V.P., Fomin, O.V., Kulbovskiy, I.I., Zub, E.P.: Rail Vehicles: The Resistance to the Movement and the Controllability: Monograph, 160 pp. Ukrmetalurginform STA, Dnipro (2017)
13. Scheepmaker, G.M., Goverde, R.M., Kroon, L.G.: Review of energy-efficient train control and timetabling. Eur. J. Oper. Res. **257**, 355–376 (2017)

14. Wang, Y., Schutter, B.D., van den Boom, T.J., Ning, B.: Optimal trajectory planning for trains under operational constraints using mixed integer linear programming. IFAC Proc. Vol. **45**, 13–18 (2012). 13th IFAC Symposium on Control in Transportation Systems
15. Wang, Y., Schutter, B.D., van den Boom, T.J., Ning, B.: Optimal trajectory planning for trains – a pseudospectral method and a mixed integer linear programming approach. Transp. Res. C Emerg. Technol. **29**, 97–114 (2013)
16. Yang, X., Li, X., Ning, B., Tang, T.: A survey on energy-efficient train operation for urban rail transit. IEEE Trans. Intell. Transp. Syst. **17**, 2–13 (2016)
17. Ye, H., Liu, R.: Nonlinear programming methods based on closed-form expressions for optimal train control. Transp. Res. C Emerg. Technol. **82**, 102–123 (2017)

Gradient Boosting with Extreme Learning Machines for the Optimization of Nonlinear Functionals

Cristiano Cervellera and Danilo Macciò

Abstract In this paper we investigate the use of the Extreme Learning Machine (ELM) paradigm for the approximate minimization of a general class of functionals which arise routinely in operations research, optimal control and statistics problems. The ELM and, in general, neural networks with random hidden weights, have proved to be very efficient tools for the optimization of costs typical of machine learning problems, due to the possibility of computing the optimal outer weights in closed form. Yet, this feature is possible only when the cost is a sum of squared terms, as in regression, while more general cost functionals must be addressed with other methods. Here we focus on the gradient boosting technique combined with the ELM to address important instances of optimization problems such as optimal control of a complex system, multistage optimization and maximum likelihood estimation. Through the application of a simple gradient boosting descent algorithm, we show how it is possible to take advantage of the accuracy and efficiency of the ELM for the approximate solution of this wide family of optimization problems.

Keywords Nonlinear optimization · Gradient boosting · Extreme learning machines

1 Introduction

In recent years, there has been a growing interest in neural networks where hidden layer weights are extracted *a priori* and kept fixed during the training. In particular, the term "extreme learning machine" (ELM) has gained popularity in the past few years to denote a family of methods centered on this concept (see, e.g., [1–4] and the references therein). Despite its simplicity, this paradigm has proved to be effective in machine learning problems since it enables a very fast training procedure. In fact,

C. Cervellera · D. Macciò (✉)
Institute of Marine Engineering, National Research Council of Italy, Genova, Italy
e-mail: cristiano.cervellera@cnr.it; danilo.maccio@cnr.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_7

once the inner weights are set, the outer weights can be determined analytically through pseudoinversion.

However, this fast one-step solution is true only for cost functions that can be expressed as a sum of squares, which limits the scope mainly to regression based on a mean squared error criterion. Here we investigate the use of the ELM paradigm for the solution of more general costs with respect to the mean squared error. More specifically, we consider a differentiable nonlinear functional having a general form $\mathcal{L}(x, f(x))$, where $x \in \mathbb{R}^n$ is the input and $f : \mathbb{R}^n \to \mathbb{R}^m$ is the output of the ELM model through which we want to optimize $\mathcal{L}$. We address the problem of minimizing the expected value of the functional

$$f^* = \arg\min_{f \in \mathcal{F}} \mathrm{E}_{x \sim p} [\mathcal{L}(x, f(x))]$$

where $\mathcal{F}$ is a suitable class of functions and the expectation over x is defined through a density $p(x)$.[1] This class of optimization problems is a very general paradigm for many important problems in operations research, optimal control and statistics, such as dynamic optimization, trajectory planning, maximum likelihood estimation, dimensionality reduction, etc.

In order to apply the ELM to this kind of problem, we focus on the gradient boosting procedure [5, 6]. According to this paradigm, especially popular for decision trees, the solution minimizing $\mathcal{L}$ is built as an ensemble of "weak" models, obtained iteratively through an optimized steepest descent procedure. In the ELM implementation, a "small" ELM is optimized and added at each step to produce an ensemble that yields the optimal solution.

The boosting paradigm has been investigated in the ELM context through the Adaboost framework [7], but this method is specifically tailored for classification in machine learning problems. A proper gradient boosting approach has been considered for ELM training in [8], again with a focus on regression and classification. In that work the basic algorithm is modified adding a regularization term, and the training procedure is performed in two phases, involving the computation of second-order derivatives.

In this paper, for the investigation of the considered class of optimization problems through the ELM, we adopt a more basic algorithm to combine the latter with gradient boosting, relying on a one-phase first-order procedure with optimized descent. Using this approach, that we call ELM Gradient Boosting for Optimization (EGBO), we test the combination of gradient boosting and ELM in three different important applications where the need to optimize a nonlinear functional arises: optimal control of a complex system, multistage optimization and maximum likelihood estimation. In all cases, we show how the approximate solution obtained through this methodology compares very well with the reference/optimal

[1] We assume that the minimum exists. Otherwise, the problem can be redefined in terms of ε-optimal solutions.

solution for the considered problem, yet retaining the light computational burden typical of the ELM.

In light of this small computational burden, the EGBO can also be seen as a particularly efficient instance of the family of methods that address variational and optimization problems through linear combinations of variable basis functions (see, e.g., [9–13]).

2 The Basic ELM Paradigm

Here we recall the typical elements of the standard ELM framework, as the basis for the EGBO approach that will be described in the next section.

Consider a training set of N input points $X_N = \{x_1, \ldots, x_N\}$ where $x_i \in \mathcal{X} \subset \mathbb{R}^n$, and a set of associated outputs $Y_N = \{y_1, \ldots, y_N\}$ where $y_i \in \mathcal{Y} \subset \mathbb{R}^m$. The basic ELM is a feedforward neural network with single hidden layer in which the weights of the neural units are kept fixed during the training phase. Specifically, the output of the l-th activation function h_l for an input x_i is obtained as $h_l(x_i) = h(a_l^\mathsf{T} x_i + b_l)$, where $a_l \in \mathbb{R}^n$ and $b_l \in \mathbb{R}$ are extracted randomly according to some probability distribution (typically, the uniform one).

Then, the m-dimensional output of the ELM can be written as $f(x) = \beta\, h(x)$, where β is the $(m \times L)$ matrix of outer weights and $h(x) = [h_1(x), \ldots, h_L(x)]^\mathsf{T}$ is the vector of the L activation functions.

As said, the parameters a_l and b_l of the hidden layer are not trained, thus the only parameters that are optimized are the output weights of the linear combination β.

To this purpose, collect the outputs in the training set in the matrix $Y = [y_1, \ldots, y_N]^\mathsf{T}$. Also define the $(N \times L)$ matrix H collecting all the outputs of the neural units computed in the input points, i.e., $H = [h(x_1), \ldots, h(x_N)]^\mathsf{T}$.

The training process consists in finding the matrix β^* minimizing $J = \alpha \sum_{i=1}^{N} |y_i - \beta h(x_i)|^2 + |\beta|^2$, where $|\cdot|$ denotes the Euclidean norm and α is a regularization parameter. This problem can be solved in a single step analytically, yielding

$$\beta^* = (I/\alpha + H^\mathsf{T} H)^{-1} H^\mathsf{T} Y \tag{1}$$

This single-step computation of the optimal outer weights makes the training procedure very fast when the number L of neural units and the training set size N are small.

3 Gradient Boosting for Optimization with the ELM

The boosting approach [5] consists first in approximating the expected value with an average computed over a set $X_N = \{x_1, \ldots, x_N\}$ of N realizations of the input points. This leads to the substitution of the original problem stated in Sect. 1 with

$$f^* = \arg\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(x_i f(x_i))$$

where $x_i \in X_N$.

Then, the functional is minimized over the components of f through a steepest descent algorithm with step optimization, building f iteratively at each step through the addition of so-called "weak learners".

Here the class $\mathcal{F}$ is the class of functions that can be represented as an ensemble of ELMs, and the weak models are the single small ELMs of the ensemble. By "small" we mean having a number L of hidden units much smaller than the number that would be required if a single ELM was employed to represent the solution of the problem. In the extreme case, $L = 1$.

The application of the gradient boosting approach to the aforementioned class $\mathcal{F}$ leads to Algorithm 1.

At the k-th iteration of the algorithm, we denote by g_{ijk} the negative gradient of $\mathcal{L}$ with respect to $f_j(x_i)$ computed at $f^{k-1}(x_i)$, i.e,

$$g_{ijk} = -\left.\frac{\partial \mathcal{L}(x_i, f(x_i))}{\partial f_j(x_i)}\right|_{f(x_i)=f^{k-1}(x_i)} \tag{2}$$

Finally, we collect all the components g_{ijk} in the $(N \times m)$ matrix $G^{(k)}$, whose columns are the target vectors that have to be learned by the weak model.

The EGBO algorithm returns a function represented by a weighted sum of small ELMs, that provides an approximate minimizer for the functional Q.

At step 1 of the algorithm we initialize the solution with a constant value for all $x \in \mathcal{X}$.

Step 3 consists in the selection of a small batch from X_N and corresponding outputs from Z_N.

At step 4 we compute the negative gradient matrix $G^{(k)}$ using (2), evaluated in the outputs of the current solution $f^{(k-1)}(x_i)$ for all the points in S_B.

Steps 5 and 6 consist in the creation of the new weak ELM that approximates the gradient, to be added to the ensemble. Specifically, first we extract a set of L hidden units $h_l^{(k)}$, $l = 1, \ldots, L$, with random weights as described in Sect. 2. Then, we compute the optimal ELM output weights $\beta^{(k)}$ as in (1), using S_B as the training patterns and the elements of the negative gradient matrix $G^{(k)}$ as the targets.

Step 7 implements the optimized gradient descent step. In this step, we minimize Q along the direction given by the approximate gradient yielded by the ELM

Algorithm 1 EGBO

Require: X_N: the sets of N input points; B: the number of batch training points; L: the number of hidden units in the weak ELM; α: the regularization parameter; K: the number of boosting iterations.

1: *Initialization*: Extract a random subset $S_B \subset X_N$ of B input points and compute

$$f^{(0)}(x) = \varrho_0 = \arg\min_{\varrho} \frac{1}{B} \sum_{i=1}^{B} \mathcal{L}(x_i, \varrho)$$

2: **for** $k = 1, \ldots, K$ **do**
3: Extract a random subset $S_B \subset X_N$ of B input points.
4: Compute $G^{(k)}$ in $f^{(k-1)}(x_i)$ for $x_i \in S_B$ as in (2).
5: Extract a random set of L hidden units $h^{(k)}$.
6: Find the output weights $\beta^{(k)}$ using $G^{(k)}$ as targets.
7: Compute

$$\varrho_k = \arg\min_{\varrho} \frac{1}{B} \sum_{i=1}^{B} \mathcal{L}(x_i, f^{(k-1)}(x_i) + \varrho\, \beta^{(k)} h^{(k)}(x_i)).$$

8: Update $f^{(k)} = f^{(k-1)} + \varrho_k\, \beta^{(k)} h^{(k)}$.
9: **end for**
10: *Output*: $\hat{f} = f_K$.

$\beta^{(k)} h^{(k)}$ created and trained at steps 5 and 6. In this way, we find the optimal weight for this new ELM that is eventually added to the ensemble.

Finally, at step 8 we build the new current solution $f^{(k)} = f^{(k-1)} + \varrho_k\, \beta^{(k)} h^{(k)}$.

Also notice that the number L of neurons at step 5 is expected to be small. Thus, the combination of a small batch of B training points and a small number L of neurons makes the training of the new ELM through pseudoinversion at point 6 very fast. At the same time, the optimized gradient descent step at point 7 is a one-dimensional minimization that can be performed efficiently using any solver from various available commercial and open packages.

4 Application to Optimization Problems

In this section we showcase the application of the EGBO approach in three different instances of the considered class of optimization problems, namely (1) optimal control, (2) multistage optimization, and (3) maximum likelihood estimation. For all the examples we provide simulation experiments. All the tests have been implemented in Python, using functions from the Numpy and Scipy libraries when needed, and the simulations were run on a PC equipped with an Intel Core i7-8700K 3.7 GHz processor with 16 GB of RAM.

4.1 Optimal Control

The first problem we consider is the closed-loop myopic optimal control of a nonlinear system. In this case, we have a dynamic system characterized by a n-dimensional state x_t whose evolution is ruled by a state equation $x_{t+1} = s(x_t, u_t)$.

The goal of the myopic optimization problem is to find a closed-loop control u_t as a function of the current state, i.e., $u_t = f(x_t)$ such that u_t minimizes at each t a cost function $c(x_{t+1}) = c(s(x_t, u_t))$.

The test system we consider is composed by a sliding trolley carrying a hanging payload. Mathematically, this system is modeled as pendulum, with a mass above that is free to move horizontally, and a smaller mass (the payload) that is attached to the sliding one through a rigid wire. The state x_t of the system is the 5-dimensional vector $[p_t, v_t, \theta_t, \omega_t, \mu_t]^\mathsf{T}$, where p_t and v_t are the trolley position and speed, θ_t and ω_t are the wire angle and angular velocity, and μ_t is the mass of the payload. A more detailed description of the system can be found in [14].

The goal of the control is to bring the payload to a desired position (here taken as 0 for simplicity) and stop. The one-dimensional control is the force we apply to the trolley, while the myopic cost function has the form $c(x_{t+1}) = v \cdot x_{t+1}^2$. For the tests we considered initial points x_0 in the range $[-0.55, 0, 0, 0, 1]$, $[0.55, 0, 0, 0, 5]$ and $v = [5, 0.1, 0, 0, 0]$. The control u_t that minimizes this cost function at each stage t is able to stabilize the trolley at 0 smoothly, starting from any initial position and with any payload in the above-defined range.

Here the functional to minimize is the expectation of $\mathcal{L}(x_t, f(x_t)) = c(s(x_t, f(x_t)))$. The training set was composed of $X_N = 1000$ points randomly extracted in the range $[-0.55, -0.2, -0.1, -0.25, 1]$, $[0.55, 0.2, 0.1, 0.25, 5]$. The EGBO algorithm was applied using $L = 10$ hidden units for the weak ELM learner (with logistic activation functions and all the weights randomly extracted in the range $[-1, 1]$ with uniform distribution), $K = 50$ iterations, regularization parameter $\alpha = 10^3$, size of the training batch $B = 200$.

To test the performance of the obtained controller $\hat{f}$, 100 initial test points x_0 were extracted uniformly in the range $[-0.55, 0, 0, 0, 1]$, $[0.55, 0, 0, 0, 5]$, i.e., with different initial positions and payloads to move. For each x_0, a reference optimal trajectory was derived for $T = 100$ stages by computing at each stage the reference myopic optimal control $u_t^* = f^*(x_t)$ through the direct numerical minimization of $c(x_{t+1})$, using the *minimize_scalar* function from the Scipy package. Then, for each of the 100 initial test points, the control action computed by the $\hat{f}$ controller yielded by the EGBO procedure was applied to drive the trajectory.

Figure 1 reports the boxplots of the sum of the cost over the 100 stages for all the 100 initial test points, for the reference and the EGBO solution, together with an example of resulting trajectory (only the first component, i.e., position). It can be noticed that the EGBO solution yields the same costs as the reference optimal control, corresponding to very similar resulting trajectories, without the need to resort at each stage to a numerical minimization procedure. For this problem, the time required to obtain the solution was 26.7 s.

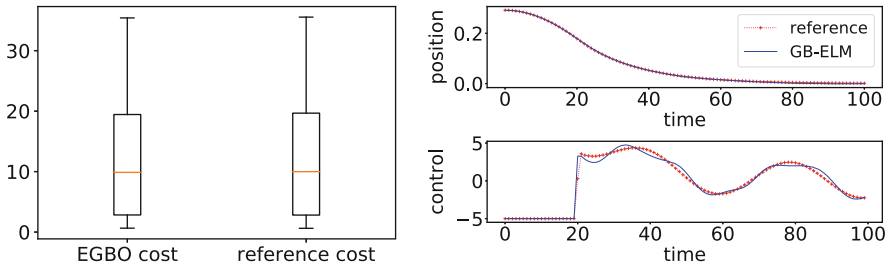


Fig. 1 Left: boxplots of the costs for the 100 initial test points in the optimal control example. Right: comparison between an example trajectory (first component, i.e., position) yielded by the reference and EGBO solutions

4.2 Multistage Stochastic Optimization

Another example of problem where EGBO can be applied is multistage stochastic optimization. This kind of problem is characterized by a Markovian state equation of the kind $x_{t+1} = s(x_t, u_t, \xi_t)$, where ξ_t is a random vector with a given probability distribution, and the goal is to minimize the sum of the expected value over ξ of a cost $c_t(x_t, u_t, \xi_t)$ over a finite number T of stages, i.e., find the sequence of vectors $\{u_0, \ldots, u_{T-1}\}$ that minimizes $\sum_{t=1}^{T-1} \mathrm{E}_{\xi_t}[c_t(x_t, u_t, \xi_t) + c_T(x_T)]$, where c_T is the cost of the final stage.

As a test case we consider an inventory forecasting problem, that is a classic testbed for this kind of problems (see, e.g., [15, 16]). Here we address a six-dimensional instance characterized by a state vector x_t in which the components $x_{i,t}$, for $i = 1, 2, 3$ are the current levels of item i in the inventory, while components $x_{i+3,t}$ are the forecast demands for item i in the next stage. The control vector u_t has dimension $m = 3$, representing the quantity of the 3 items to order at stage t. The evolution of the state vector is ruled by the state equation $x_{i,t+1} = x_{i,t} + u_{i,t} - x_{i+3,t}\,\xi_{i,t}$ for $i = 1, 2, 3$, and $x_{i,t+1} = \mu_i\,\xi_{i,t}$ for $i = 4, 5, 6$, where ξ_t is vector of random corrections on the forecasts having a lognormal distribution, and $\mu_i = [18, 15, 20]^\mathsf{T}$.

The cost function for stage t corresponds to the goal of keeping the inventory level close to zero, plus penalty functions to take into account constraints on the positivity of the orders and the maximum allowed value for their sum, equal to 20. Specifically, the cost function has the form $c_t = \sum_{i=1}^{3}[v_i^+ \max(0, x_{i,t}) + v_i^- \max(0, -x_{i,t})] + p(u_t)$, where $v^+ = [4, 7, 2]$ and $v^- = [2, 5, 1]$ are surplus and deficit coefficients, respectively, and $p(u)$ is defined as $p(u) = \sum_{i=1}^{3} \max(0, -u_i)^2 + \max(0, \sum_{i=1}^{3} u_i - 20)^2$. The cost function c_T for the last stage is equal to c_t without the term $p(u_t)$.

Here we consider the solution for the last stage, which is the main building block to find the solution for horizons with an arbitrary number of stages, through approximate dynamic programming [17, 18]. Specifically, we look for the optimal policy $u^*_{T-1} = f(x_{T-1})$ minimizing

$$\mathcal{L}(x_{T-1}, f(x_{T-1})) = \frac{1}{P}\sum_{p=1}^{P}[c_{T-1}(x_{T-1}, u_{T-1}, \xi^{(p)}_{T-1}) + c_T(s(x_{T-1}, u_{T-1}, \xi^{(p)}_{T-1}))]$$

where $\xi^{(p)}_{T-1}$, for $p = 1, \ldots, P$, is a realization of the random variable ξ drawn with lognormal distribution.

The training set was composed of $X_N = 1000$ points randomly extracted in the range $[-10, -10, -10, 15, 12, 17]$, $[10, 10, 10, 21, 18, 23]$.

The EGBO algorithm was applied again using $L = 10$ hidden units for the weak ELM learner (with all the weights randomly extracted in the range $[-1, 1]$ with uniform distribution), $K = 50$ iterations, regularization parameter $\alpha = 10^3$, size of the training batch $B = 200$.

To test the performance of the obtained policy $\hat{f}$, 100 test points x_{T-1} were extracted uniformly in the same range as above and, for each point, a reference optimal control vector $u^*_{T-1} = f^*(x_{T-1})$ was computed using the *minimize* function from the SciPy package to minimize $\mathcal{L}(x_{T-1}, u_{T-1})$ directly. For a further comparison, a heuristic myopic policy f^h was also considered, consisting in ordering, for each item i, the forecast demand minus the available stock, i.e., $u^h = f^h(x_t) = x_{i+3,t} - x_{i,t}$. The components of the resulting control vector are then rescaled proportionally to satisfy the constraint of having the sum bounded by 20.

Figure 2 reports the boxplots of the cost for all the 100 test points, for the reference, the EGBO and the heuristic solution.

It can be noticed that the EGBO solution yields practically the same costs as the reference optimal control, and much better performance with respect to the heuristic control policy. In this case, the computational effort to get the approximate solution was 22.3 s.

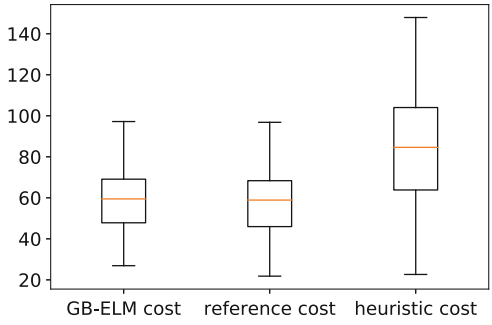


Fig. 2 Boxplots of the costs for the 100 test points in the multistage stochastic optimization example

4.3 Maximum Likelihood Estimation

The last example of optimization problem we present where the EGBO can be applied is maximum likelihood (ML) estimation.

In this kind of problem we have a random variable $r \in \mathbb{R}$ characterized by a probability density function that we know except for a finite set of parameters, i.e., $r \sim p(r, \gamma)$, where γ is the vector of unknown parameters that must be estimated from available observations. The ML approach consists in finding the parameter γ^* that maximizes the likelihood, i.e., the joint density of the sample of n observations $x = [r_1, \dots r_n]^\mathsf{T}$. Since we assume the sample to be i.i.d., the joint density $p(r_1, \dots, r_n, \gamma)$ can be expressed as the product of the single densities $p(r_i, \gamma)$. Then, the typical approach to estimate the unknown parameters γ consists in maximizing the log-likelihood, which is more practical than the likelihood in product form. Eventually, γ^* for an observed sample x is the output of the ML estimator f that minimizes the following functional $\mathcal{L}(x, f(x)) = -\sum_{i=1}^{n} \log p(r_i, f(x))$.

For many commonly employed density functions we can compute the optimal estimator f^* in closed form, by which we can obtain the estimated parameter $\gamma^* = f^*(x)$ directly. However, for a general density there is no analytic solution, and some numerical approximation must be employed. Here we show how the EGBO method can be applied to derive an approximate ML estimator that yields an accurate estimate for γ without the need to resort to a numerical solver for each new observed sample x. For the tests we consider a random variable z distributed according to the Rayleigh distribution, having density $p(z, \gamma) = z \exp(-z^2/(2\gamma^2))/\gamma^2$. The parameter that needs to be estimated is the scale γ. To this purpose, we applied the EGBO algorithm using a training set composed of $X_N = 1000$ samples of $n = 20$ points z having a Rayleigh distribution, with scale parameter γ randomly extracted in the range $[1, 5]$.

The EGBO algorithm was applied using $L = 30$ hidden units for the weak ELM learner (with all the weights randomly extracted in the range $[-1, 1]$ with uniform distribution), $K = 50$ iterations, regularization parameter $\alpha = 10^3$, and the full set X_N as the training batch S_B.

The performance of the approximate ML estimator can be evaluated using the true ML estimator that in this case can be computed in closed form, yielding $\gamma^* = \sqrt{\sum_{i=1}^{n} z_i^2/2n}$. In particular, 100 20-dimensional test samples were generated with a random scale parameter in the $[1, 5]$ range, and for each sample the output of the EGBO estimator was compared with the true ML estimate by means of a relative absolute error defined as $e = (\gamma^* - \hat{\gamma})/\gamma^*$, where $\hat{\gamma}$ is the scale parameter estimated by the ELM estimator.

The mean of the relative error turns out to be equal to 0.029, which indicates how the EGBO approach can yield an accurate estimation for a wide range of values of the unknown parameter of the distribution. Figure 3 reports the boxplot of the errors for all the 100 test samples, together with an illustration of the variability of the Rayleigh distribution in the considered range. The time to run the algorithm to compute $\hat{\gamma}$ was 29 s.

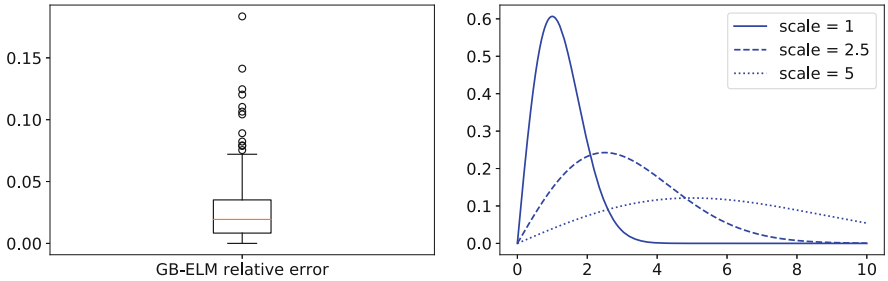


Fig. 3 Left: boxplot of the errors for the 100 test samples in the maximum likelihood estimation case. Right: illustration of the high variability of the Rayleigh distribution in the range considered for the tests

5 Conclusions and Future Work

The gradient boosting technique has been investigated using the ELM model as the base weak learner. This allows to extend the ELM paradigm to all the optimization problems in which the cost/loss function is not based on a mean squared error criterion, with a unified and very general approach. The resulting algorithm, that we called EGBO, yields approximate solutions that retain the accuracy of the ELM, since the output is an ensemble of ELMs that still has the structure of a single ELM. Furthermore, since each step of the algorithm is based on the pseudoinversion of small matrices, the method is very fast, and applicable also to large datasets through batch optimization.

To showcase the versatility and accuracy of the proposed methodology, the EGBO algorithm has been applied to three important instances of optimization problems in which the cost/loss function is not a sum of squares: optimal control, multistage optimization, and maximum likelihood estimation. In all these examples the EGBO has yielded approximate solutions that compare very well with the optimal/reference ones, confirming its accuracy despite the generality of the approach.

Future work will be aimed at optimizing the performance of the method, and extensively investigating the convergence of Algorithm 1.

References

1. Huang, G., Huang, G.-B., Song, S., You, K.: Trends in extreme learning machines: a review. Neural Netw. **61**, 32–48 (2015)
2. Huang, G.-B., Cambria, E., Toh, K.-A., Widrow, B., Xu, Z.: New trends of learning in computational intelligence. IEEE Comput. Intell. Mag. **10**, 16–17 (2015)
3. Huang, G.-B., Chen, L., Siew, C.-K.: Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans. Neural Netw. **17**(4), 879–892 (2006)

4. Cervellera, C., Macciò, D.: Low-discrepancy points for deterministic assignment of hidden weights in extreme learning machines. IEEE Trans. Neural Netw. Learn. Syst. **27**(4), 891–896 (2016)
5. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference and Prediction. Springer, New York (2009)
6. Mason, L., Baxter, J., Bartlett, P., Frean, M.: Boosting algorithms as gradient descent. In: Solla, S.A., Leen, T.K., Müller, K. (eds.) Advances in Neural Information Processing Systems, vol. 12, pp. 512–518. MIT Press, Cambridge (1999)
7. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55**(1), 119–139 (1997)
8. Guo, H., Wang, J., Ao, W., He, Y.: SGB-ELM: an advanced stochastic gradient boosting-based ensemble scheme for extreme learning machine. Comput. Intell. Neurosci. **2018**, 1–14, (2018)
9. Gelfand, I.M., Fomin, S.V.: Calculus of Variations. Prentice Hall, New Jersey (1963)
10. Zoppoli, R., Sanguineti, M., Parisini, T.: Approximating networks and extended Ritz method for the solution of functional optimization problems. J. Optim. Theory Appl. **112**, 403–439 (2002)
11. Cervellera, C., Macciò, D., Muselli, M.: Functional optimization through semi-local approximate minimization. Oper. Res. **58**(5), 1491–1504 (2010)
12. Alessandri, A., Cervellera, C., Macciò, D., Sanguineti, M.: Optimization based on quasi-Monte Carlo sampling to design state estimators for non-linear systems. Optimization **69**(7), 963–984 (2010)
13. Gnecco, G., Sanguineti, M.: Neural approximations in discounted infinite-horizon stochastic optimal control problems. Eng. Appl. Artif. Intell. **74**, 294–302 (2018)
14. Macciò, D., Cervellera, C.: Local kernel learning for data-driven control of complex systems. Expert Syst. Appl. **39**(18), 13399–13408 (2012)
15. Fan, H., Tarun, P.K., Chen, V.: Adaptive value function approximation for continuous-state stochastic dynamic programming. Comput. Oper. Res. **40**(4), 1076–1084 (2013)
16. Cervellera, C., Macciò, D.: F-discrepancy for efficient sampling in approximate dynamic programming. IEEE Trans. Cybern. **46**, 1628–1639 (2016)
17. Bertsekas, D.: Dynamic Programming and Optimal Control, vol. I, 2nd edn. Athena Scientific, Belmont (2000)
18. Cervellera, C., Macciò, D.: A novel approach for sampling in approximate dynamic programming based on F-discrepancy. IEEE Trans. Cybern. **47**(10), 3355–3366 (2017)

A MILP Model for Biological Sample Transportation in Healthcare

Mario Benini, Paolo Detti, and Garazi Zabalo Manrique de Lara

Abstract In this paper, a real-world transportation problem is addressed, concerning the collection and the transportation of blood and biological sample tubes from draw centers to a main hospital. Blood and other biological samples are collected in different centers during morning hours and have to be transported to the main hospital by a fleet of vehicles. Each sample has a given lifetime, i.e., a deadline. If a sample can not arrive to the hospital before the deadline either is discarded or a *stabilization* process must be carried out in dedicated centers. After stabilization, a sample can be delivered to the main hospital by a new deadline. A time-indexed Mixed Integer Linear Programming formulation of the problem is provided and tested on different instances generated from real-life data.

Keywords Transportation in healthcare · Vehicle routing · Mathematical programming

1 Introduction

The transportation problem addressed in this paper arises from a real-world healthcare application, concerning the reorganization of the collection, transportation and analysis of biological sample tubes in Bologna, Italy. The reorganization consists on the activation of a single laboratory, hereafter called HUB, that has in charge the analysis of all the samples collected at the draw centers located in the metropolitan area, and of a number of laboratories devoted to the preprocessing of the samples, called *Spoke* Centers. The main objective of the reorganization project is of optimizing laboratory processes, standardizing procedures and introducing a single control system. In this new system (operating from 2016), blood and other biological samples are drawn out in different centers, usually during morning hours

M. Benini · P. Detti (✉) · G. Z. M. de Lara
University of Siena, Siena, Italy
e-mail: mbenini@diism.unisi.it; detti@diism.unisi.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_8

and then transported to the HUB. Each sample has a limited lifetime, i.e., a deadline, and has to arrive to the HUB by the lifetime. When a sample can not arrive to the HUB by the lifetime either is discarded or can be preprocessed, i.e., *stabilized*, at a Spoke Center, hereafter also called spoke, to give it an extra lifetime. The *stabilization* of a sample takes some time and, after stabilization, a sample can be delivered to the HUB by a new deadline. Vehicles devoted to the transportation of the samples are located in geographically distributed depots. When samples are transported to a Spoke Center by a vehicle, the vehicle does not have to wait until the end of the stabilization process, but can depart after dropping the samples off. Then, another vehicle will pass to pick up the samples after the stabilization process is finished.

The problem consists in collecting all samples and in delivering them to the HUB on time, eventually by using the spokes to gain samples' extra lifetime.

In this work, we propose a mathematical programming model including all the characteristics of the problem. It allows to minimize the total travel time while delivering all samples on time to the HUB, i.e. within their deadlines.

In the literature there are different works that address blood sample collection problems, [2–4, 7, 8]. In [2], a transportation problem originating from the blood collection process of the Austrian Red Cross blood program is addressed. The problem is modeled as a vehicle routing problem with multiple interdependent time windows, and solved by a mixed-integer programming formulation and different heuristics. Grasas et al. [3] consider the problem of sample collection and transportation from different collection points to a core laboratory for testing in Spain. The problem is modeled as a variant of the capacitated vehicle routing problem with open routes and route length constraints, and a heuristic based on a genetic algorithm is proposed. A similar problem is addressed in [7] where a mobile blood collection system is designed and a routing problem is proposed with the aim of transporting blood samples from blood mobile draw centers to the depot. A mathematical model and a 2-stage IP based heuristic algorithm are proposed to solve the problem. In [4], a vehicle routing problem is addressed for blood transportation between hospitals or donor/client sites. A hybrid meta-heuristic algorithm including genetic algorithms and local search is developed able to reduce the cost and the response time for emergency. In [8], the problem of allocating units of blood from a regional blood transfusion centre to the hospitals of its area is considered. The problem is formulated as a multiobjective transportation problem.

A new feature of the problem addressed in this paper is the possibility of stabilizing samples in Spoke Centers to gain extra lifetime, that can be used to deliver a sample to the main hospital on time. From a modeling point of view, Spoke Centers can be modeled as transfer points, where the sample may be "transferred" from one vehicle to another, after the end of the stabilization process. Indeed, the vehicle delivering a sample to a Spoke Center does not have to wait until the end of the stabilization process, but may depart after dropping the samples off. Transportation problems with transfers have been addressed in the literature [1, 5, 6]. In [6], the pickup-and-delivery problem in which transfers are allowed is addressed, and mixed integer-programming formulations are proposed and evaluated. The

pickup-and-delivery problem with transfers is also addressed in [5]. The authors propose heuristics capable of efficiently inserting requests through transfer points and embed them into an Adaptive Large Neighborhood Search (ALNS) scheme. The approach is evaluated on real-life instances. Cortes et al. [1] address Dial-a-Ride problems where passengers may be transferred from one vehicle to another at specific locations. A mathematical programming formulation is presented and a solution method based on Benders decomposition is proposed.

In this paper, a time-indexed Mixed Integer Linear Programming (MILP) model is provided able to represent all the characteristics of the problem. Time indexing is used to model the possibility of multiple visits of the Spoke Centers by the same vehicle and/or the same sample (even if a sample can be stabilized at most once). A preliminary computational campaign on different sets of instances based on real-life data is presented. The paper is organized as follows. In Sect. 2 a detailed description of the problem is presented. Section 3 describes the MILP model. The preliminary computational results are reported in Sect. 4.

2 Problem Description

In the addressed problem, a set of transportation requests, i.e., biological samples, must be carried from draw centers to the HUB. A fleet of vehicles, located in geographically distributed depots is available to perform the transportation requests. Given the small dimension of the samples, vehicles can be considered with unlimited capacity. The problem consists in assigning the transportation requests to the vehicles and in finding the routing of each vehicle in such a way that: (1) all the samples are delivered on time to the HUB; (2) the total travel time is minimized. Some constraints must be taken into account, regarding the arrival to the main hospital of the samples and the fulfillment of the time windows on the pickup locations. In fact, a sample must be delivered to the hospital before a pre-specified time span from its withdrawal. In other words, each sample has a lifetime, defining a deadline in which the sample has to be delivered to the main hospital.

The pickups of the samples can be performed during the opening hours of the draw centers, and pickup and delivery operations require given times to load or unload a request. If a request can not be delivered to the HUB by the deadline, a stabilization process can be performed that will give an extra lifetime to the biological sample. The stabilization process is made in geographically distributed Spoke Centers. Observe that, in a Spoke Center, a vehicle can depart after dropping the samples off, and another vehicle can pass to pick up the sample after the end of the stabilization process. Figure 1 shows a scheme of the two transportation modes of a request: either directly from the draw center to the HUB or first from the draw center to a Spoke Center and then to the HUB.

In the remainder of this section, notation is introduced and a formal definition of the problem is given. Let $G = (V, A)$ be a complete directed graph, where $V = \{P \cup S \cup D \cup H\}$ is the node set and $A = \{(i, j) : i, j \in V\}$ is the arc set. $P = \{p(r) : r \in R\}$ is the set of pickup nodes of the transportation requests,

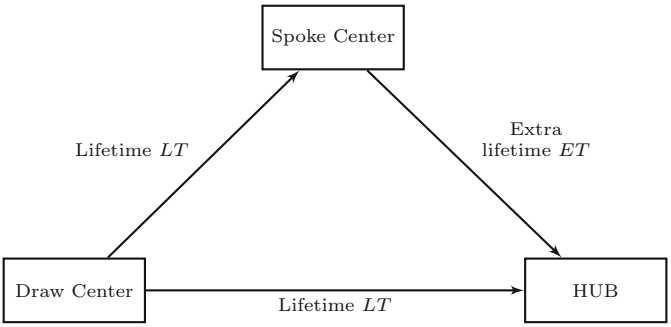


Fig. 1 Transportation modes of a sample

$S = \{1, \ldots, |S|\}$ is the set of spoke nodes, $D = \{d(k) : k \in K\}$ is the set of depot nodes and H is the HUB node. Let $K = \{1, \ldots, |K|\}$ be the set of vehicles and $R = \{1, \ldots, |R|\}$ be the set of requests. For modeling reasons, we associate one pickup node with each request in the set of requests R. Then, the requests that belong to the same draw center have different pickup nodes with the same physical location. For each request r in R, LT_r is the lifetime of request r, ET is the extra lifetime gained if r is processed in a Spoke Center, and $[e_r, l_r]$ is the related pickup time window. The lifetime LT_r is the deadline of request r in R when no stabilization process is performed. On the other hand, if request r is processed in a Spoke Center (i.e., if it is stabilized), the new deadline within which request r must be delivered to the HUB is ET. The time window $[e_r, l_r]$ of each request $r \in R$ imposes that request r can only be picked up from its pickup location between time e_r and l_r. A vehicle is allowed to arrive at the location of r before the start of the time window, but it has to wait until e_r to begin the loading operation. The service time related to the loading/unloading operation is st at each node. The stabilization time is $stbt$ at each spoke for each request. Finally, let $TT = \{0, 1, \ldots, T\}$ be the time interval considered for the time indexing of the variables. The problem is that of finding routes on G such that: each pickup node $i \in P$ is visited exactly once in the interval $[e_i, l_i]$; each request is stabilized in a Spoke Center at most once; each request must be delivered to the HUB before its lifetime or its extra lifetime after the stabilization; the total travel time of the routes is minimized.

3 A Time-Indexed MILP Formulation for the Problem

In this section, a MILP formulation for the addressed problem is presented, in which both integer and real variables are used. The variables are listed below.

- A_i arrival time of a vehicle at a node i to pick up the relative request;
- B_i beginning of the loading service of the relative request at node i;

- A_r^j arrival time of a request r at the spoke j for stabilization;
- P_r^j departure time of a request r from the spoke j where r is picked up after stabilization;
- PT_r^j departure time of a request r from the spoke j where r is just transited;
- A_{r+n} arrival time of request r to the HUB;
- $x_{ij}^{kt} \in \{0, 1\}$ be equal to 1 if vehicle k traverses arc (i, j) arriving in j at t and 0 otherwise;
- $y_{ij}^{krt} \in \{0, 1\}$ is 1 if vehicle k traverses arc (i, j) carrying request r arriving in j at t and 0 otherwise;
- $z_j^{krt} \in \{0, 1\}$ be equal to 1 if vehicle k arrives in spoke j at t to leave request r and 0 otherwise;
- $w_j^{krt} \in \{0, 1\}$ be equal to 1 if vehicle k arrives in spoke j at t to pick up request r and 0 otherwise.

The objective function reads as:

$$\min \sum_{t \in TT} \sum_{k \in K} \sum_{(i,j) \in A} t_{ij} x_{ij}^{kt} \tag{1}$$

The constraints of the model can be divided into three classes: (i) *Flow Constraints*, describing the vehicle and request flows; (ii) *Spoke Constraints*, describing how the spoke nodes work; (iii) *Time Constraints*, imposing the timing restrictions of the problem and the compliance of visit instants of the nodes. In what follows, the Flow Constraints of the formulation are reported.

$$\sum_{t \in TT} \sum_{j \in N, j \neq d(k)} x_{d(k)j}^{kt} \leq 1, \forall k \in K \tag{2}$$

$$\sum_{t \in TT} x_{Hd(k)}^{kt} = \sum_{t \in TT} \sum_{i \in N, i \neq H} x_{iH}^{kt}, \forall k \in K \tag{3}$$

$$\sum_{t \in TT} x_{ij}^{kt} + \sum_{t \in TT} x_{ji}^{kt} \leq 1, \forall i \in D, \forall j \in S, \forall k \in K \tag{4}$$

$$\sum_{t \in TT} \sum_{j \in N, j \neq d(k)} x_{d(k)j}^{kt} = \sum_{t \in TT} \sum_{j \in N, j \neq d(k)} x_{jd(k)}^{kt}, \forall k \in K \tag{5}$$

$$\sum_{t \in TT} \sum_{j \in N, j \neq i} x_{ij}^{kt} - \sum_{t \in TT} \sum_{j \in N, j \neq i} x_{ji}^{kt} = 0, \forall k \in K, \forall i \in N \tag{6}$$

$$x_{ji}^{kt} = 0, \forall k \in K, \forall t \in TT, \forall ji \in A : loc(i) = loc(j), e(i) < e(j) \tag{7}$$

$$\sum_{t \in TT} \sum_{k \in K} \sum_{j \in N, j \neq i} x_{ij}^{kt} = 1, \forall i \in P \tag{8}$$

$$\sum_{t \in TT} \sum_{k \in K} \sum_{j \in N, j \neq p(r)} y_{p(r)j}^{krt} = 1, \forall r \in R \tag{9}$$

$$\sum_{t\in TT}\sum_{k\in K}\sum_{i\in N, i\neq H} y_{iH}^{krt} = 1, \forall r \in R \quad (10)$$

$$\sum_{t\in TT}\sum_{k\in K}\sum_{j\in N, j\neq i} y_{ij}^{krt} - \sum_{t\in TT}\sum_{k\in K}\sum_{j\in N, j\neq i} y_{ji}^{krt} = 0, \forall r \in R, \forall i \in S \quad (11)$$

$$\sum_{t\in TT}\sum_{j\in N, j\neq i} y_{ij}^{krt} - \sum_{t\in TT}\sum_{j\in N, j\neq i} y_{ji}^{krt} = 0, \forall r \in R, \forall k \in K, \forall i \in P \setminus p(r) \quad (12)$$

$$\sum_{t\in TT}\sum_{k\in K}\sum_{j\in N, j\neq p(r)} y_{jp(r)}^{krt} = 0, \forall r \in R \quad (13)$$

$$\sum_{t\in TT}\sum_{k\in K}\sum_{j\in N, j\neq p(r)} y_{p(r)j}^{krt} - \sum_{t\in TT}\sum_{k\in K}\sum_{j\in N, j\neq p(r)} y_{jp(r)}^{krt} = 1, \forall r \in R \quad (14)$$

$$y_{ij}^{krt} \leq x_{ij}^{kt}, \forall r \in R, \forall k \in K, \forall (i, j) \in A, \forall t \in TT \quad (15)$$

Constraints (2) ensure that each vehicle leaves at most once from its depot, while Constraints (3) state that a vehicle must come back to its depot after visiting the HUB. Constraints (4) state that each vehicle can not go directly from its depot to a spoke and come back to its depot immediately after. Constraints (5) ensure that each vehicle that starts from its depot must return in the same depot. Vehicle flow conservation is imposed by equalities (6), while Constraints (7) are used to forbid sub-cycles between pickup nodes. Constraints (8) ensure, together with (6), that each pickup node must be visited by exactly one vehicle. Constraints (9) and (10) ensure that each request is served exactly once. Constraints (11) maintain the request flow conservation at the spoke nodes allowing requests to switch from one vehicle to another. Constraints (12) and (13) ensure the request flow conservation at the pickup nodes requiring that every vehicle bringing a request must also leave carrying the same request. Constraints (15) link variables x and y. The Spoke Constraints are:

$$\sum_{t\in TT}\sum_{k\in K}\sum_{j\in S} z_j^{krt} \leq 1, \forall r \in R \quad (16)$$

$$z_j^{krt} \leq \sum_{i\in N, i\neq j} y_{ij}^{krt}, \forall j \in S, \forall k \in K, \forall r \in R, \forall t \in TT \quad (17)$$

$$\sum_{t\in TT}\sum_{k\in K} z_j^{krt} = \sum_{t\in TT}\sum_{k\in K} w_j^{krt}, \forall j \in S, \forall r \in R \quad (18)$$

$$x_{ij}^{kt} \leq \sum_{r\in R} z_j^{krt} + \sum_{r\in R} w_j^{krt}, \forall j \in S, \forall k \in K, \forall i \in N, i \neq j, \forall t \in TT \quad (19)$$

$$w_j^{krt} \leq \sum_{t_1=t+st+t_{ij}}^{t+st+stbt+st+t_{ij}} \sum_{i\in N, i\neq j} y_{ij}^{krt_1}, \forall j \in S, \forall k \in K, \forall r \in R, \forall t \in TT \quad (20)$$

$$w_j^{krt} \leq \sum_{i \in N, i \neq j} x_{ij}^{kt}, \forall j \in S, \forall k \in K, \forall r \in R, \forall t \in TT \quad (21)$$

$$\sum_{h \in N} x_{ih}^{kt+st+t_{ih}} \geq 1 - M[(1 - z_i^{krt}) + \sum_{h \in N, h \neq j} x_{ih}^{kt+st+t_{ih}} + \sum_{r_1 \in R} w_i^{kr_1 t} \quad (22)$$

$$\forall i \in S, \forall j \in N, \forall k \in K, \forall r \in R, \forall t \in TT$$

$$x_{ij}^{kt_1} \leq M(1 - z_i^{krt}) + M(1 - w_i^{krt}), \forall i \in S, \forall j \in N, \forall k \in K, \forall r \in R, \quad (23)$$

$$\forall t \in TT, \forall t_1 \in TT : t \leq t_1 \leq t + st + stbt + st + t_{ij}$$

Constraints (16) impose each request to be stabilized in a spoke at most once. Constraints (17) link variables y and z, while Constraints (18) ensure that, if a request is left in a spoke node for the stabilization, then it must be picked up from that spoke node. Constraints (19)–(21) ensure the matching of time indexed variables when a request is downloaded or picked up at a certain spoke node. Constraints (22) impose that a vehicle can wait at a spoke node only if it has to pick up a request. Constraints (23) state that if a vehicle arrives at a spoke node at the same time instant both to download and to pick up the same request, then it can not leave that spoke node within the time interval between that time instant and the end of the stabilization of that request. Time Constraints can be modeled as:

$$e_r \leq B_{p(r)} \leq l_r, \forall r \in R \quad (24)$$

$$B_i \geq A_i, \forall i \in P \quad (25)$$

$$A_r^j \leq M \sum_{t \in TT} \sum_{k \in K} z_j^{krt}, \forall j \in S, \forall r \in R \quad (26)$$

$$P_r^j \leq M \sum_{t \in TT} \sum_{k \in K} z_j^{krt}, \forall j \in S, \forall r \in R \quad (27)$$

$$P_r^j \geq B_r^j + 2st + stbt - M(1 - \sum_{t \in TT} \sum_{k \in K} z_j^{krt}), \forall j \in S, \forall r \in R \quad (28)$$

$$A_r^j - e_r + st \leq LT_r \sum_{t \in TT} \sum_{k \in K} z_j^{krt}, \forall j \in S, \forall r \in R \quad (29)$$

$$A_{r+n} + st - [A_r^j + (st + stbt) \sum_{t \in TT} \sum_{k \in K} z_j^{krt}] \leq$$

$$ET \sum_{t \in TT} \sum_{k \in K} z_j^{krt} + M(1 - \sum_{t \in TT} \sum_{k \in K} z_j^{krt}), \forall j \in S, \forall r \in R \quad (30)$$

$$A_{r+n} + st - e_r \leq LT_r(1 - \sum_{t \in TT} \sum_{k \in K} \sum_{j \in S} z_j^{krt}) + M \sum_{t \in TT} \sum_{k \in K} \sum_{j \in S} z_j^{krt}, \forall r \in R \quad (31)$$

Time windows compliance is ensured by Constraints (24), while Constraints (25) state that the service at a pickup node can start after the arrival of a vehicle.

Constraints (26) and (27) set to zero the arrival and departure time variables of a request at each spoke node if that request is never left in that spoke for the stabilization (M is a big enough constant). Constraints (28) state that a stabilized request can depart from the spoke after the end of the stabilization. Constraints (29)–(31) ensure the maximum ride time compliance whether a request is stabilized or not. The following constraints (from (32) to (60)) ensure the time compliance between consecutively visited nodes (all possible scenarios are considered).

1. Both the starting node and the arrival node are pickup nodes. Then these constraints are $\forall i, j \in P, \forall r \in R, \forall k \in K, \forall t \in TT$:

$$A_j \geq ty_{ij}^{krt} - l_i(1 - y_{ij}^{krt}) \tag{32}$$

$$A_j \leq ty_{ij}^{krt} + M(1 - y_{ij}^{krt}) \tag{33}$$

   – If $t_{ij} = 0$:

$$A_j \geq B_i - l_i(1 - y_{ij}^{krt}) \tag{34}$$

$$A_j \leq B_i + M(1 - y_{ij}^{krt}) \tag{35}$$

   – If $t_{ij} \neq 0$:

$$A_j \geq B_i + st + t_{ij} - (l_i + st + t_{ij})(1 - y_{ij}^{krt}) \tag{36}$$

$$A_j \leq B_i + st + t_{ij} + M(1 - y_{ij}^{krt}) \tag{37}$$

2. The starting node is a pickup node while the arrival node is a spoke node. Two possible sub-scenarios exist. The following constraints hold $\forall i \in P, \forall j \in S, \forall r \in R, \forall k \in K, \forall t \in TT$:

   – If request r is left in j:

$$A_r^j \geq B_i + st + t_{ij} - (l_i + st + t_{ij})[(1 - y_{ij}^{krt}) + (1 - z_j^{krt})] \tag{38}$$

$$A_r^j \geq tz_j^{krt} - l_i[(1 - y_{ij}^{krt}) + (1 - z_j^{krt})] \tag{39}$$

$$A_r^j \leq B_i + st + t_{ij} + M[(1 - y_{ij}^{krt}) + (1 - z_j^{krt})] \tag{40}$$

$$A_r^j \leq tz_j^{krt} + M[(1 - y_{ij}^{krt}) + (1 - z_j^{krt})] \tag{41}$$

   – If request r is picked up in j:

$$P_r^j \geq B_i + st + t_{ij} - (l_i + st + t_{ij})[(1 - x_{ij}^{kt}) + (1 - w_j^{krt})] \tag{42}$$

$$tx_{ij}^{kt} \geq B_i + st + t_{ij} - M(1 - x_{ij}^{kt}) \tag{43}$$

$$tx_{ij}^{kt} \leq B_i + st + t_{ij} + M(1 - x_{ij}^{kt}) \quad (44)$$

3. The starting node is a spoke node while the arrival node is a pickup node. Also in this case there are two possible sub-scenarios. Then, $\forall j \in P, \forall i \in S, \forall r \in R, \forall k \in K, \forall t \in TT$, we have:

   – If request r is picked up in i:

$$A_j \geq P_r^i + t_{ij} - M[(1 - y_{ij}^{krt}) + (1 - \sum_{t_1=t-t_{ij}-st-stbt-st}^{t-t_i\,j-st} w_i^{krt_1})] \quad (45)$$

$$A_j \geq ty_{ij}^{krt} - M(1 - y_{ij}^{krt}) \quad (46)$$

$$A_j \leq P_r^i + t_{ij} + M[(1 - y_{ij}^{krt}) + (1 - \sum_{t_1=t-t_{ij}-st-stbt-st}^{t-t_i\,j-st} w_i^{krt_1})] \quad (47)$$

$$A_j \leq ty_{ij}^{krt} + M(1 - y_{ij}^{krt}) \quad (48)$$

   – If request r is left in i:

$$A_j \geq A_r^i + st + t_{ij} - M[(1 - x_{ij}^{kt}) + y_{ij}^{krt} + (1 - z_i^{krt+st+t_{ij}})] \quad (49)$$

$$A_j \geq tx_{ij}^{kt} - M(1 - x_{ij}^{kt}) \quad (50)$$

$$A_j \leq A_r^i + st + t_{ij} + M[(1 - x_{ij}^{kt}) + y_{ij}^{krt} + (1 - z_i^{krt+st+t_{ij}})] \quad (51)$$

$$A_j \leq tx_{ij}^{kt} + M(1 - x_{ij}^{kt}) \quad (52)$$

4. Both the starting node and the arrival node are spoke nodes. Then, we have to consider four possible sub-scenarios. The following constraints hold $\forall i, j \in S, \forall r \in R, \forall k \in K, \forall t \in TT$:

   – If request r is picked up in i and request h is left in j:

$$A_h^j \geq P_r^i + t_{ij} - M[(1 - y_{ij}^{krt}) + (1 - y_{ij}^{kht}) + (1 - z_j^{kht}) + (1 - \sum_{t_1=t-t_{ij}-st-stbt-st}^{t-t_i\,j-st} w_i^{krt_1})] \quad (53)$$

$$A_h^j \geq tz_j^{kht} - M[(1 - y_{ij}^{krt}) + (1 - y_{ij}^{kht}) + (1 - z_j^{kht})] \quad (54)$$

   – If request r is left in i and request h is left in j:

$$A_h^j \geq B_r^i + st + t_{ij} - M[(1 - y_{ij}^{kht}) + y_{ij}^{krt} + (1 - z_j^{kht}) + (1 - z_i^{krt-st-t_{ij}})] \quad (55)$$

$$A_h^j \geq tz_j^{kht} - M[(1 - y_{ij}^{kht}) + (1 - z_j^{kht})] \quad (56)$$

– If request r is picked up in i and request h is picked up in j:

$$P_h^j \geq P_r^i + t_{ij} + st - M[(1 - y_{ij}^{krt}) + y_{ij}^{kht} + (1 - w_j^{kht}) + (1 - \sum_{t_1=t-t_{ij}-st-stbt-st}^{t-t_i j-st} w_i^{krt_1})] \quad (57)$$

$$P_h^j \geq (t + st) w_j^{kht} - M[(1 - y_{ij}^{kht}) + (1 - w_j^{kht})] \quad (58)$$

– If request r is left in i and request h is picked up in j:

$$P_h^j \geq B_r^i + st + t_{ij} + st - M[(1 - x_{ij}^{kt}) + y_{ij}^{krt} + y_{ij}^{kht} + (1 - w_j^{kht}) + (1 - z_i^{krt-st-t_{ij}})] \quad (59)$$

$$P_h^j \geq (t + st) w_j^{kht} - M[(1 - x_{ij}^{kt}) + (1 - w_j^{kht})] \quad (60)$$

Constraints (32)–(37) ensure time continuity whether both the departure node and the arrival node are pickup nodes using the Big-M technique. In particular, Constraints (32) and (33) enable the continuous variables to match with the (time indexed) binary variables. When two pickup nodes are related to the same physical location, Constraints (34) and (35) ensure that the arrival time in j is equal to the beginning of the service in i. On the other hand, if the two nodes are related to different locations, Constraints (36) and (37) state that the arrival time in j must be equal to the beginning of the service in i plus the service time plus the travel time from i to j. Similarly, Constraints (38)–(44) ensure time continuity compliance when the departure node is a pickup node and the arrival node is a spoke node. More precisely, Constraints (38)–(41) only are active when a request is left in the spoke node for the stabilization, while Constraints (42)–(44) only work when a request is picked up from the spoke node. In the same way, Constraints (45)–(52) ensure the time continuity compliance when the departure node and the arrival node are, respectively, a spoke node and a pickup node. Constraints (53)–(60) ensure the time continuity compliance when both the departure node and the arrival node are spoke nodes, by covering every possible scenario.

$$PT_{r_1}^j \geq PT_{r_2}^j - M[(1 - y_{ij}^{kr_1t}) + z_j^{kr_1t} + (1 - w_j^{kr_2t})]$$
$$\forall k \in K, \forall r_1, r_2 \in R : r_1 \neq r_2, \forall i \in N, \forall j \in S, \forall t \in TT \quad (61)$$

$$PT_{r_1}^j \geq (t + st) y_{ij}^{kr_1t} - M[(1 - y_{ij}^{kr_1t}) + z_j^{kr_1t} + (1 - w_j^{kr_2t})]$$
$$\forall k \in K, \forall r_1, r_2 \in R : r_1 \neq r_2, \forall i \in N, \forall j \in S, \forall t \in TT \quad (62)$$

$$PT_{r_1}^j \geq B_{r_2}^j + st - M[(1 - y_{ij}^{kr_1t}) + (1 - y_{ij}^{kr_2t}) + z_j^{kr_1t} + (1 - z_j^{kr_2t})]$$
$$\forall k \in K, \forall r_1, r_2 \in R : r_1 \neq r_2, \forall i \in N, \forall j \in S, \forall t \in TT \quad (63)$$

$$PT_{r_1}^j \geq (t + st) y_{ij}^{kr_1t} - M[(1 - y_{ij}^{kr_1t}) + (1 - y_{ij}^{kr_2t}) + z_j^{kr_1t} + (1 - z_j^{kr_2t})]$$

$$\forall k \in K, \forall r_1, r_2 \in R : r_1 \neq r_2, \forall i \in N, \forall j \in S, \forall t \in TT \quad (64)$$

$$A_{r+n} \geq PLT_r^j + t_{jH} - M(1 - y_{jH}^{krt}), \forall k \in K, \forall r \in R, \forall j \in S, \forall t \in TT \quad (65)$$

$$A_{r+n} \geq P_r^j + t_{jH} - M(1 - y_{jH}^{krt}), \forall k \in K, \forall r \in R, \forall j \in S, \forall t \in TT \quad (66)$$

$$A_{r+n} \geq ty_{jH}^{krt} - M(1 - y_{jH}^{krt}), \forall k \in K, \forall r \in R, \forall j \in S, \forall t \in TT \quad (67)$$

$$A_{r+n} \leq ty_{jH}^{krt} + M(1 - y_{jH}^{krt}), \forall k \in K, \forall r \in R, \forall j \in S, \forall t \in TT \quad (68)$$

$$A_{r+n} \geq B_i + st + t_{iH} - M(1 - y_{iH}^{krt}), \forall k \in K, \forall r \in R, \forall i \in P, \forall t \in TT \quad (69)$$

$$A_{r+n} \geq ty_{iH}^{krt} - M(1 - y_{iH}^{krt}), \forall k \in K, \forall r \in R, \forall i \in P, \forall t \in TT \quad (70)$$

$$A_{r+n} \leq B_i + st + t_{iH} - M(1 - y_{iH}^{krt}), \forall k \in K, \forall r \in R, \forall i \in P, \forall t \in TT \quad (71)$$

$$A_{r+n} \leq ty_{iH}^{krt} - M(1 - y_{iH}^{krt}), \forall k \in K, \forall r \in R, \forall i \in P, \forall t \in TT \quad (72)$$

$$A_{r_1}^j \geq A_{r_2}^j - M[(1 - z_j^{kr_1t}) + (1 - z_j^{kr_2t}) + (1 - y_{ij}^{kr_1t}) + (1 - y_{ij}^{kr_2t})]$$
$$\forall k \in K, \forall r_1, r_2 \in R, \forall i \in N, \forall j \in S, \forall t \in TT \quad (73)$$

$$A_{r_1}^j \leq A_{r_2}^j + M[(1 - z_j^{kr_1t}) + (1 - z_j^{kr_2t}) + (1 - y_{ij}^{kr_1t}) + (1 - y_{ij}^{kr_2t})]$$
$$\forall k \in K, \forall r_1, r_2 \in R, \forall i \in N, \forall j \in S, \forall t \in TT \quad (74)$$

$$P_{r_1}^j \geq P_{r_2}^j - M[(1 - w_j^{kr_1t}) + (1 - w_j^{kr_2t})], \forall k \in K, \forall r_1, r_2 \in R, \forall j \in S, \forall t \in TT \quad (75)$$

$$P_{r_1}^j \leq P_{r_2}^j + M[(1 - w_j^{kr_1t}) + (1 - w_j^{kr_2t})], \forall k \in K, \forall r_1, r_2 \in R, \forall j \in S, \forall t \in TT \quad (76)$$

$$A_{r_1+n} \geq A_{r_2+n} - M[(1 - y_{iH}^{kr_1t}) + (1 - y_{iH}^{kr_2t})], \forall k \in K, \forall r_1, r_2 \in R, \forall i \in N, \forall t \in TT \quad (77)$$

$$A_{r_1+n} \leq A_{r_2+n} + M[(1 - y_{iH}^{kr_1t}) + (1 - y_{iH}^{kr_2t})], \forall k \in K, \forall r_1, r_2 \in R, \forall i \in N, \forall t \in TT \quad (78)$$

$$\sum_{t \in TT} \sum_{j \in N, j \neq i} y_{ij}^{krt} - \sum_{t \in TT} \sum_{j \in N, j \neq i} y_{ji}^{krt} \leq M \sum_{t \in TT} z_i^{krt}, \forall r \in R, \forall i \in S, \forall k \in K \quad (79)$$

$$\sum_{t \in TT} \sum_{j \in N, j \neq i} y_{ij}^{krt} - \sum_{t \in TT} \sum_{j \in N, j \neq i} y_{ji}^{krt} \geq -M \sum_{t \in TT} z_i^{krt}, \forall r \in R, \forall i \in S, \forall k \in K \quad (80)$$

$$w_i^{krt} \leq \sum_{q=t+1}^{TT} \sum_{j \in N, j \neq i} y_{jH}^{krt}, \forall r \in R, \forall i \in S, \forall k \in K \quad (81)$$

$$\sum_{q=1}^{t-1} \sum_{j \in N, j \neq i} y_{p(r)j}^{krt} \geq z_i^{krt}, \forall r \in R, \forall i \in S, \forall k \in K \quad (82)$$

$$\sum_{q=t+stab}^{TT} \sum_{k \in K} w_i^{krq} \geq \sum_{k \in K} z_i^{krt}, \forall r \in R, \forall i \in S, \forall t = 1, \ldots, TT - stbt \quad (83)$$

$$A_j \geq tx_{ij}^{kt} - M(1 - x_{ij}^{kt}), \forall k \in K, \forall i \in D, \forall j \in P, \forall t \in TT \quad (84)$$

$$A_j \leq tx_{ij}^{kt} + M(1 - x_{ij}^{kt}), \forall k \in K, \forall i \in D, \forall j \in P, \forall t \in TT \quad (85)$$

$$A_j \geq t_{ij} - M(1 - x_{ij}^{kt}), \forall k \in K, \forall i \in D, \forall j \in P, \forall t \in TT \quad (86)$$

Constraints (61)–(64) define the departure time of a request from a spoke when it only transits without being downloaded for stabilization. In fact, it is possible that a vehicle visits a spoke node to download or to pick up a request without downloading the other requests that it is carrying on. Constraints (65)–(68) (Constraints (69)–(72)) define the arrival time of a request at the HUB when the previous visited node is a spoke (the previous visited node is a pickup node). Constraints (73)–(78) link the variables of departure and arrival time at spoke nodes and the arrival time at the HUB of different requests carried by the same vehicle. Constraints (79)–(80) state that a request can change vehicle at a spoke only if it is stabilized. Constraints (81) impose that a request is picked up and eventually delivered to a spoke by the same vehicle, and Constraints (82) impose that a request is eventually picked up from a spoke and delivered to the HUB by the same vehicle. Constraints (83) state that a request can be picked up from a spoke after the stabilization process. Finally, Constraints (84)–(86) define the arrival time at a pickup node if the previous node is a depot.

4 Preliminary Computational Results

A preliminary experimental campaign has been performed on different instances generated by using real data. According to the real data, in a center, a sample is drawn every 3 min, approximately. The lifetime of a sample is 90 min. The stabilization process takes 20 min and gives to the stabilized sample 90 min of (extra) lifetime (see Fig. 1). The service time required by a vehicle to load or unload samples at a draw center or at a Spoke Center is $st_i = 10$ min. According to the data introduced above, 13 instances have been generated and solved. The MILP formulations have been solved by the solver Gurobi on a PC equipped with Intel i5 processor and 8 Gb of RAM. In our experiments, we consider different lengths of the time steps contained in time interval TT, i.e., 1 or 5 min. Obviously, the bigger the time step is the smaller the number of variables of the MILP is. Table 1 shows the results. In the first 5 columns, for each instance, the instance id, the time step used, and the number of requests, vehicles and spokes are reported. Columns 6–9 respectively report the number of B&B nodes, the first and best lower bounds at the root node, the optimal solution value and the computational time attained by the solver Gurobi. A "*" in Column 9 indicates that a time limit has been reached, i.e., the optimality of the solution has not been certified. Instances 2 and 7 are the same instance solved with different time steps: by increasing the time step from 1 to 5 min the computational time drops from about 23 to 1.29 s. Similarly, the computational time of instances 6 and 8 (the same instance in which only the time step varies) drops from about 1536 to 56 s, about a 95% decrease in the computational time with an acceptable approximation. Instance 9*a* (Instance 9*b*) has been obtained by Instance 9 by adding a new spoke (a new spoke and a new vehicle). Instance 10*a* has been obtained by Instance 10 by adding a new spoke and a new vehicle, too. Observe that, an optimal solution can be found on Instances 9 and 10 in at most 4 h. However, by

Table 1 Computational results

Id	Time step	$\|R\|$	$\|K\|$	$\|S\|$	B&B nodes	First/Best LB	Obj.	Time (s)
1	1	2	2	1	0	120/120	120	9.98
2	1	3	1	1	696	160/160	241	22.97
3	1	4	2	1	5060	314/314	366	222.26
4	1	5	3	1	0	235/276	276	230.87
5	1	8	3	1	5132	423/423	475	1500.45
6	1	9	4	1	0	502/502	502	1535.72
7	5	3	1	1	0	55/145	145	1.29
8	5	9	4	1	0	295/330	330	56.78
9	5	12	6	1	2081	390/455	710	2945.97
9a	5	12	6	2	2928	390/623	710	28,880*
9b	5	12	7	2	2955	390/517.7	710	28,880*
10	5	15	8	1	22,002	390/485	715	14,021.33
10a	5	15	9	2	46	390/403.7	710	43,200*

increasing the number of spokes and vehicles, the optimality of the solution can not be certified even within 8 h (Instances 9*a* and 9*b*) and 12 h (Instance 10*a*) of computation. Furthermore, during the Branch&Cut procedure, no feasible solution is found for Instances 9*a* and 9*b* (Instance 10*a*) within 4 h (8 h) of computation. Finally, we note that for Instance 10*a* we get a feasible solution with value strictly better than the optimal solution of Instance 10.

5 Conclusion

In this paper, a transportation problem arising from a real-world healthcare application has been presented, in which biological samples must be transported from draw centers to a main laboratory by their lifetimes. Dedicated centers, called Spoke Centers, can be used to gain extra lifetimes for the samples. In the problem, Spoke Centers can be modeled as transfer points where a request can be assigned to a different vehicle, after the stabilization process is concluded. A MILP model has been proposed for the problem and tested on instances generated from real data. As shown by the results and as expected, the model can be directly used to optimally solve small instances of the problem. However, different heuristics could be developed to solve bigger real-world problems: the whole area of interest can be partitioned into a certain number of sub-areas and one sample transportation problem can be solved for each sub-area; the MILP model can be employed in a matheuristic scheme to find feasible solutions in a reduced amount of time. Finally we point out that, thanks to the time indexing, our model can be used to account the time-dependency of the data, e.g., the variation of travel times depending on the hour of the day.

References

1. Cortés, C.E., Matamala, M., Contardo, C.: The pickup and delivery problem with transfers: formulation and a branch-and-cut solution method. Eur. J. Oper. Res. **200**(3), 711–724 (2010)
2. Doerner, K.F., Gronalt, M., Hartl, R.F., Kiechle, G., Reimann, M.: Exact and heuristic algorithms for the vehicle routing problem with multiple interdependent time windows. Comput. Oper. Res. **35**(9), 3034–3048 (2008)
3. Grasas, A., Ramalhinho, H., Pessoa, L.S., Resende, M.G., Caballé, I., Barba, N.: On the improvement of blood sample collection at clinical laboratories. BMC Health Serv. Res. **14**(1), 12 (2014)
4. Karakoc, M., Gunay, M.: Priority based vehicle routing for agile blood transportation between donor/client sites. In: 2017 International Conference on Computer Science and Engineering (UBMK), pp. 795–799. IEEE (2017)
5. Masson, R., Lehuédé, F., Péton, O.: An adaptive large neighborhood search for the pickup and delivery problem with transfers. Transp. Sci. **47**(3), 344–355 (2013)
6. Rais, A., Alvelos, F., Carvalho, M.: New mixed integer-programming model for the pickup-and-delivery problem with transshipment. Eur. J. Oper. Res. **235**(3), 530–539 (2014)
7. Şahinyazan, F.G., Kara, B.Y., Taner, M.R.: Selective vehicle routing for a mobile blood donation system. Eur. J. Oper. Res. **245**(1), 22–34 (2015)
8. Sapountzis, C.: Allocating Blood to Hospitals as a Multiobjective Transportation Problem. In: Medical Informatics Europe90, pp. 733–739. Springer, Berlin (1990)

Infinite Kernel Extreme Learning Machine

Elisa Marcelli and Renato De Leone

Abstract This paper addresses the analysis of the problem of combining Infinite Kernel Learning (IKL) approach and Extreme Learning Machine (ELM) structure. ELM represents a novel and promising alternative to Neural Networks, for its simplicity in implementation and high efficiency, especially concerning convergence and generalization performance. A currently underdeveloped topic concerning ELM implementation is given by the optimization process of base kernels: choosing different kernel combinations may lead to very dissimilar performance results. An innovative ELM approach using a combination of multiple kernels has been proposed in Liu et al. As a change of paradigm, we are interested in using an infinite set of base kernels, defining in this way an original ELM based algorithm called Infinite Kernel Extreme Learning Machine (IK-ELM). About that, a novel 3-step algorithm combining IKL and ELM is proposed. Finally, a brief analysis about further possible directions is discussed.

Keywords Extreme learning machine · Infinite kernel learning · Feedforward neural network

1 Introduction

Extreme Learning Machine (ELM) is a supervised Machine Learning (ML) algorithm first introduced in 2004 [7] which may be considered a cross between feedforward neural network and Support Vector Machine (SVM). Specifically, ELM has the structure of classical feedforward neural networks with one or multiple hidden layers but is characterized by a fundamental feature: hidden nodes variables (i.e., parameters and weights) need not to be tuned but are randomly assigned at

E. Marcelli (✉) · R. De Leone
Department of Mathematics, School of Sciences and Technology, University of Camerino, Camerino MC, Italy
e-mail: elisa.marcelli@unicam.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_9

the beginning of the algorithm and remain fixed throughout all its duration. In this way, ELM is marked by less computational complexity, moving from an iterative based approach to a one-step type of algorithm, maintaining however a remarkable performance compared to classical ML methods. ELM proposes an algorithm with two important features that characterize it from classical ML approaches: better generalization performance and faster convergence. Specifically, it achieves the first since it not only aims to minimize the approximation error, given by the difference between the expected output and the computed result, but it also looks for the smallest norm weights. The basic idea behind ELM is to implement a method that can improve the learning speed of classical single-hidden layer feedforward neural networks (SLFNs). Specifically, when a feedforward network is used, in order to obtain the optimal parameters all weights and biases characterizing each layer have to be tuned, making the learning algorithm quite slow. Moreover, among the most used algorithms to solve the problem we find gradient method, which is usually characterized by a slow convergence and may converge to local minima. In the first place, ELM was introduced in the field of SLFNs with the aim of solving these issues, developing a model which could reduce the learning speed and at the same time reach a global optimum [7–9]. Later on, it was broaden to the general field of SLFNs. The key idea behind ELM affects the hidden layer: the parameters of hidden nodes need not to be learned but are randomly chosen as an initial step of the algorithm. In particular, let us consider a SLFN with L hidden nodes and m output nodes and let $\{x^i, y^i\}_{i=1}^N$ be the training set with $x^i \in \mathbb{R}^n$ and $y^i = [y_1^i, \ldots, y_m^i]^{\mathrm{T}} \in \mathbb{R}^m$, with $y_j^i \in \{0, 1\}$ whether or not x^i belongs to the jth class, $1 \leq j \leq m$. Then, given a general input x^i, a classical ELM architecture is mathematically modelled as $f(x^i) = \sum_{k=1}^K \beta_k h_k(x^i) = h(x^i)^{\mathrm{T}}\beta$, where $\beta = [\beta_1, \ldots, \beta_K]^{\mathrm{T}}$ is the output weight vector, connecting the hidden layer with the output layer, and $h : \mathbb{R}^n \longrightarrow \mathbb{R}^K$ maps data from the n-dimensional input space to the K-dimensional hidden space. Specifically, $h_k(x^i) = g_k(a_k, b_k, x^i)$ is the activation function (or output function) of the kth hidden node with respect to input x^i and parameters (a_k, b_k). Note that, the activation function is decided at the beginning of the algorithm and fixed throughout all its length and may not be unique, meaning that each hidden node may have assigned a different output function.

In this paper we wish to define a new approach called Infinite Kernel Extreme Learning Machine combining ELM together with the approach of Infinite Kernel Learning (IKL), i.e., using a possibly infinite set of base kernels. Given a generic data set $\mathcal{X}$, kernel methods are based on the idea of projecting points into a higher dimensional space $\mathcal{F}$ in order to apply simple linear processes in such a space. This projection is not done in a direct way, i.e., computing the coordinates of data in the new space, but with the use of the so called kernel functions: specific functions that allow to operate in high dimensions just computing the inner products of the images of the data. Specifically, given a function $\phi : \mathcal{X} \longrightarrow \mathcal{F}$, a kernel function is a function k such that $k(x^i, x^j) = \langle \phi(x^i), \phi(x^j) \rangle$ for all $x^i, x^j \in \mathcal{X}$, where $\langle \cdot, \cdot \rangle$ is the scalar product in space $\mathcal{F}$. Nowadays kernel methods are mainly used in shallow structures such as SVM [16, 17] but may also be used in more complex architectures

[4]. The most immediate and important advantage that comes from using kernel-based algorithms is the fact that computing the inner products may actually be computationally less expensive than calculating the new coordinates of the points. The choice of the kernel is a crucial issue in these algorithms: different kernels will describe different nonlinear functions and the performance of the method will often depend on the appropriate choice of the kernel. The structure of this paper is organized as follows. In Sect. 2 we provide an overview about ELM. In Sect. 3 we take stock on the theory on the base of IKL. In Sect. 4 we present our approach. Finally in Sect. 5 we draw conclusions.

2 Extreme Learning Machine

Similarly to feedforward neural networks, ELM has the goal of minimizing the training error but at the same time it aims to reach the smallest norm of the output weights β_k as well: as underlined in [3], the number of parameters does not effect the generalization performance of the network, on the contrary, the size of the weights tends to have affects on the accuracy of the algorithm. In order to obtain such expected results, the concept of minimum norm least-square solution of a general linear system is used. Given a training set $\{x^i, y^i\}_{i=1}^N$, let $\mathrm{H} \in \mathbb{R}^{N \times K}$ and $\mathrm{Y} \in \mathbb{R}^{N \times m}$ respectively be the hidden layer output matrix and the matrix containing all the labels y^i for $i = 1 \dots N$, namely

$$\mathrm{H} = \begin{bmatrix} h(x^1) \\ \vdots \\ h(x^N) \end{bmatrix} = \begin{bmatrix} h_1(x^1) & \dots & h_K(x^1) \\ \vdots & \vdots & \vdots \\ h_1(x^N) & \dots & h_K(x^N) \end{bmatrix}, \qquad \mathrm{Y} = \begin{bmatrix} (y^1)^{\mathrm{T}} \\ \vdots \\ (y^m)^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} y_1^1 & \dots & y_m^1 \\ \vdots & \vdots & \vdots \\ y_1^N & \dots & y_m^N \end{bmatrix}.$$

ELM key idea is to solve the following problem

$$\text{minimize:} \quad ||\mathrm{H}\beta - \mathrm{Y}||_q^{p_1} \quad \text{and} \quad ||\beta||_q^{p_2}$$

In this way, the method tends to have a better generalization performance with respect to classical feedforward neural networks. In particular, its goal is not only reducing the approximation error, meaning the gap between the given and the computed output, but it also looks for the smallest norm weights: β is sought to be the minimum norm least-square solution of the system $||\mathrm{H}\beta - \mathrm{Y}||$, i.e., the smallest norm solution among all the least-square solutions. Hence, ELM aims to solve the following minimization problem

$$\min_{\beta \in \mathbb{R}^{K \times m}} \quad ||\mathrm{H}\beta - \mathrm{Y}|| \tag{1}$$

The optimal solution to Problem (1) is given by $\hat{\beta} = \mathrm{H}^{\dagger}\mathrm{Y}$, with $\mathrm{H}^{\dagger}$ the Moore-Penrose generalized pseudoinverse of matrix H. Note that, it was shown that this special matrix $\mathrm{H}^{\dagger}$ is indeed the minimum norm least-square solution of Problem (1) [7]. Moreover, the convergence speed is characterized by the fact that ELM randomly assigns values to the hidden layer, only adjusting the output weights. In this way, the classical minimization problem to be solved with iterative adjustments becomes a one step algorithm computing the pseudoinverse of matrix H, reducing in a deep way the speed of the algorithm. The Problem described above may be written as

$$\begin{aligned} \min_{\beta \in \mathbb{R}^{K \times m}} \quad & \frac{1}{2}||\beta||^2 + \frac{C}{2}\sum_{i=1}^{N}||\xi^i||^2 \\ \text{s.t} \quad & h(x^i)\beta = y^{i\mathrm{T}} - \xi^{i\mathrm{T}} \quad i = 1, \ldots, N \end{aligned} \tag{2}$$

where $\xi^i = [\xi_1^i, \ldots, \xi_N^i]^{\mathrm{T}}$ is the training error vector with respect to input x^i and C is the regularization parameter. Then, the Lagrangian function associated to Problem (2) is

$$\mathcal{L}_{\mathrm{ELM}}(\beta, \xi, \alpha) = \frac{1}{2}||\beta||^2 + \frac{C}{2}\sum_{i=1}^{N}||\xi^i||^2 - \sum_{i=1}^{N}\sum_{j=1}^{m}\alpha_j^i(h(x^i)\beta_j - y_j^i + \xi_j^i) \tag{3}$$

where $\alpha^i = [\alpha_1^i, \ldots, \alpha_m^i]^{\mathrm{T}}$, $i = 1, \ldots, N$, are the Lagrangian parameters and $\alpha = [\alpha^1, \ldots, \alpha^N]^{\mathrm{T}}$. The Karush-Kuhn-Tucker (KKT) optimality conditions corresponding to (3) are

$$\frac{\partial \mathcal{L}_{\mathrm{ELM}}}{\partial \beta_j} = \beta_j - \sum_{i=1}^{N}\alpha_j^i h(x^i)^{\mathrm{T}} = 0 \Rightarrow \beta = \mathrm{H}^{\mathrm{T}}\alpha \tag{4a}$$

$$\frac{\partial \mathcal{L}_{\mathrm{ELM}}}{\partial \xi^i} = C\xi^i - \alpha^i = 0 \Rightarrow \alpha^i = C\xi^i, \quad i = 1, \ldots, N \tag{4b}$$

$$\frac{\partial \mathcal{L}_{\mathrm{ELM}}}{\partial \alpha^i} = h(x^i)\beta - (y^{i\mathrm{T}} - \xi^{i\mathrm{T}}) = 0, \quad i = 1, \ldots, N \tag{4c}$$

Based on the size of the data set and on the number of chosen hidden neurons, different solutions of Problem (2) exist.

If $N < L$, i.e., the number of training data is not bigger than the number of hidden neurons, matrix H has more columns than rows. In this specific scenario, combining the equations above, we get $\beta = \mathrm{H}^{\mathrm{T}}\left(\frac{\mathrm{I}}{C} + \mathrm{H}\mathrm{H}^{\mathrm{T}}\right)^{-1}\mathrm{Y}$ and the output

function corresponding to the generic input value x^i is

$$f(x^i) = \sum_{k=1}^{K} \beta_k h_k(x^i) = h(x^i)\mathrm{H}^{\mathrm{T}} \left(\frac{\mathrm{I}}{C} + \mathrm{H}\mathrm{H}^{\mathrm{T}} \right)^{-1} \mathrm{Y} \tag{5}$$

Otherwise, if $N > L$, namely when a very big data set is considered, a different solution is used. Specifically, $\beta = \left(\frac{\mathrm{I}}{C} + \mathrm{H}^{\mathrm{T}}\mathrm{H} \right) \mathrm{H}^{\mathrm{T}}\mathrm{Y}$ and ELM classifier for a generic input x^i becomes

$$f(x^i) = \sum_{k=1}^{K} \beta_k h_k(x^i) = h(x^i) \left(\frac{\mathrm{I}}{C} + \mathrm{H}^{\mathrm{T}}\mathrm{H} \right) \mathrm{H}^{\mathrm{T}}\mathrm{Y} \tag{6}$$

Note that, in Eqs. (5) and (6) respectively appear the products $\mathrm{H}\mathrm{H}^{\mathrm{T}}$ and $\mathrm{H}^{\mathrm{T}}\mathrm{H}$: these expressions are also known as *ELM kernel matrix* with elements of the form $h(x^i) \cdot h(x^j)$. In this context, it is important to underline that similar versions of ELM exist for both regression and classification problems. Specifically, Saunders et al. [15] proposes a ridge regression procedure with equality constraints, studying its dual form using SVM as a reference. Regarding classification problems, least square support vector machine (LS-SVM) [19, 20] modifies classical SVM structure replacing the inequality constraints with equality constraints, getting a very similar formulation to Problem (2).

In Sect. 3 we propose a brief analysis summary concerning the idea of Infinite Kernel Learning.

3 Infinite Kernel

The choice of kernel may cause relevant differences in terms of performance. Kernel-based algorithms effectively deal with the problem of choosing the "best" kernel. The use of the so called Multiple Kernel Learning (MKL) [2, 12] was proposed and implemented in the case of classical SVM [18] and LS-SVM [10]. Given a predefined finite set of kernels, MKL combines these kernels learning the combination parameters as part of the algorithm. Specifically, this combination is given by $K(\cdot,\cdot) = \sum_{i=1}^{p} \beta_i k_i(\cdot,\cdot)$, where $\{k_i(\cdot,\cdot)\}_{i=1}^{p}$ and $\{\beta_i\}_{i=1}^{p}$ are respectively the predefined kernels and the associated combination coefficients such that $\sum_{i=1}^{p} \beta_i = 1$, to be found simultaneously with the other architecture parameters.

The next step in terms of performance is not to consider the number of base kernels fixed. Specifically, as a direct application of Theorem 4.2 in [6], it can be shown that fixing the number of base kernels finite is an unneeded limitation and, consequently, that the search for combination parameters may be done over a possibly infinite set of kernels. From a mathematical point of view, combining

an infinite number of kernels may be represented as $K(\cdot,\cdot) = \int_\Theta k(\cdot,\cdot,\theta)\,\mathrm{d}\beta(\theta)$, where $\theta \in \Theta$ is a kernel parameter, i.e., specifies the associated type of kernel $k(\cdot,\cdot,\theta)$, and β is a monotonically increasing function such that $\int_\Theta \mathrm{d}\beta(\theta) = 1$. These methods have been implemented in different contexts, for example [1, 5, 14], all of them having in common the fact of considering the set of kernels as the convex hull of a predefined set of continuously parametrized basic kernels. In particular, Gehler and Nowozin [5] presents an interesting approach called Infinite Kernel Learning (IKL) where the previously analysed approach of using a combination of selected kernels is extended to a possible uncountable infinite set through a practical approach. Such a method broadens the classical MKL formulation in the field of SVM, optimizing the problem over a set of kernel parameters Θ of arbitrary cardinality. Again, given a training set $\{x^i, y^i\}_{i=1}^N$, the problem takes the following form

$$\begin{aligned} \inf_{\Theta_f \subset \Theta} \min_{\beta, v, \xi, b} \quad & \frac{1}{2} \sum_{\theta \in \Theta_f} \frac{1}{\beta_\theta} ||v_\theta||^2 + C \sum_{i=1}^N \xi^i \\ \text{subject to} \quad & y^i \left(\sum_{\theta \in \Theta_f} \langle v_\theta, \phi_\theta(x^i) \rangle + b \right) \geq 1 - \xi^i,\ i = 1, \ldots, N \\ & \xi^i \geq 0, \quad i = 1, \ldots, N \\ & \sum_{\theta \in \Theta_f} \beta_\theta = 1, \quad \beta_\theta \geq 0 \end{aligned} \tag{7}$$

where $\theta \in \Theta_f$ is a parameter specifying the associated type of kernel, Θ is the set of kernel parameters and v_θ is defined as $v_\theta := \beta_\theta w_\theta$ in order to study a convex problem w.r.t. v_θ and β_θ. Note that w_θ and b are the classical SVM parameters.

As next step, the paper takes the Lagrangian function of Problem (7) and computes its derivatives with respect to the primal variables $\beta_\theta, v_\theta, b, \xi^i$. Defined λ as the Lagrange multiplier associated to the equality constraint and α as the Lagrange multiplier associated to the inequality $y^i \left(\sum_{\theta \in \Theta_f} \langle v_\theta, \phi_\theta(x^i) \rangle + b \right) \geq 1 - \xi^i$, substituting the derivatives into the Lagrangian function we get the following dual problem

$$\begin{aligned} \sup_{\Theta_f \subset \Theta} \max_{\lambda, \alpha} \quad & \sum_{i=1}^N \alpha_i - \lambda \\ \text{subject to} \quad & \alpha \in \mathbb{R}^N,\ \lambda \in \mathbb{R} \\ & 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, N \\ & T(\theta, \alpha) \leq \lambda, \quad \forall \theta \in \Theta \end{aligned} \tag{8}$$

where $T(\theta, \alpha) = \frac{1}{2}\sum_{i,j=1}^{N} \alpha_i \alpha_j k(x^i, x^j, \theta)$. Moreover, the paper proposes and implements an algorithm based on the approach just described, showing results comparing values obtained using SVM, MKL and IKL.

In the next section we wish to describe a new approach combining IKL and ELM and propose a specific algorithm.

4 Infinite Kernel Extreme Learning Machine

In this work we wish to combine ELM together with the idea behind IKL, i.e., theoretically using an infinite combination of base kernels, introducing an algorithm called Infinite Kernel Extreme Learning Machine (IK-ELM). In particular, our starting point is the work by Liu et al. [13], where an approach called Multiple Kernel Extreme Learning Machine (MK-ELM) is described. The paper base concept is merging ELM structure with MKL: optimal kernel is a combination of predefined based kernels and the coefficients of such combination, together with ELM parameters, are learnt during the process. Specifically, given a generic training set $\{x^i, y^i\}_{i=1}^N$, MK-ELM problem is defined as follows:

$$\begin{aligned} &\min_{\beta} \ \min_{\tilde{w},\xi} \ \frac{1}{2}\sum_{p=1}^{l} \frac{||\tilde{w}_p||^2}{\beta_p} + \frac{C}{2}\sum_{i=1}^{N} ||\xi^i||^2 \\ &\text{subject to} \ \sum_{p=1}^{l} \langle \tilde{w}_p, \phi_p(x^i)\rangle = y^i - \xi^i, \quad \forall i = 1, \ldots, N \\ &\qquad\qquad \sum_{i=1}^{l} \beta_p = 1, \quad \beta_p \geq 0, \quad \forall p = 1, \ldots, l \end{aligned} \tag{9}$$

where $\{\phi_p(\cdot)\}_{p=1}^{l}$ are the feature mapping corresponding to l predefined base kernels $\{k_p(\cdot,\cdot)\}_{p=1}^{l}$, $\{\beta_p\}_{p=1}^{l}$ are the base kernel combination parameters and $\tilde{w}_p := \sqrt{\beta_p} w_p$ for $p = 1, \ldots, l$. Using the Lagrangian function corresponding to Problem (9), the paper outlines an iterative scheme of MK-ELM algorithm for both the sparse and non-sparse case.

Here, we extend the formulation above using a combination of possibly infinitely many base kernels. Note that, as in [5], Θ_f and Θ are respectively a finite set and a set of undefined cardinality of kernel parameters. Then, the problem we wish to solve is the following

$$\begin{aligned} \inf_{\Theta_f \subset \Theta} \min_{\beta} \min_{\tilde{w}_\gamma, \xi} \; & \frac{1}{2} \sum_{\gamma \in \Theta_f} \frac{||\tilde{w}_\gamma||^2}{\beta_\gamma} + \frac{C}{2} \sum_{i=1}^{N} ||\xi^i||^2 \\ \text{subject to} \; & \sum_{\gamma \in \Theta_f} \langle \tilde{w}_\gamma, \phi_\gamma(x^i) \rangle = y^i - \xi^i, \quad \forall i = 1, \ldots, N \\ & \sum_{\gamma \in \Theta_f} \beta_\gamma = 1, \quad \beta_\gamma \geq 0, \quad \gamma \in \Theta_f \end{aligned} \tag{10}$$

with the set of all possible kernels, given by the convex hull of $\{k(\cdot, \cdot, \gamma), \gamma \in \Theta\}$, which may theoretically contain an uncountable number of elements. In Problem (10) we can define an inner problem and an outer problem searching for the best finite subset of $\Theta_f \subset \Theta$. To build the dual problem, we consider the Lagrangian function corresponding to the inner problem

$$\begin{aligned} \mathcal{L}_{\text{IK-ELM}}(\beta, \tilde{w}_\gamma, \xi, \alpha, \lambda, \delta_\gamma) = & \frac{1}{2} \sum_{\gamma \in \Theta_f} \frac{||\tilde{w}_\gamma||^2}{\beta_\gamma} + \frac{C}{2} \sum_{i=1}^{N} ||\xi^i||^2 \\ & - \sum_{i=1}^{N} \alpha_i \left(\sum_{\gamma \in \Theta_f} \langle \tilde{w}_\gamma, \phi_\gamma(x^i) \rangle + y^i - \xi^i \right) \\ & - \sum_{\gamma \in \Theta_f} \delta_\gamma \beta_\gamma + \lambda \left(\sum_{\gamma \in \Theta_f} \beta_\gamma - 1 \right) \end{aligned} \tag{11}$$

and we compute the corresponding KKT conditions

$$\frac{\partial \mathcal{L}_{\text{IK-ELM}}}{\partial \tilde{w}_\gamma} = \frac{1}{\beta_\gamma} \tilde{w}_\gamma - \sum_{i=1}^{N} \alpha_i \phi_\gamma(x^i)^{\mathrm{T}} = 0 \tag{12a}$$

$$\frac{\partial \mathcal{L}_{\text{IK-ELM}}}{\partial \beta_\gamma} = -\frac{1}{2} \frac{1}{\beta_\gamma^2} ||\tilde{w}_\gamma||^2 + \lambda - \delta_\gamma = 0 \tag{12b}$$

$$\frac{\partial \mathcal{L}_{\text{IK-ELM}}}{\partial \xi^i} = C\xi^i - \alpha_i = 0 \tag{12c}$$

$$\frac{\partial \mathcal{L}_{\text{IK-ELM}}}{\partial \alpha_i} = \sum_{\gamma \in \Theta_f} \tilde{w}_\gamma^{\mathrm{T}} h_\gamma(x^i) + \xi^i - y^i = 0 \tag{12d}$$

Substituting Eqs. (12a)–(12d) in Eq. (11) and taking into account the fact that the Lagrange multiplier δ_γ is non-negative, we obtain the following dual form of Problem (10)

$$\begin{aligned} \max_{\alpha,\lambda} \quad & \sum_{i=1}^{N} \alpha_i \left(y^i - \frac{\alpha_i}{2C} \right) - \lambda \\ \text{subject to} \quad & T(\gamma, \alpha) \leq \lambda, \quad \forall \gamma \in \Theta \end{aligned} \tag{13}$$

where $T(\gamma, \alpha) := \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j \langle \phi_\gamma(x^i), \phi_\gamma(x^j) \rangle$ and the constraint $T(\gamma, \alpha) \leq \lambda$ is obtained using Eq. (12b). Note that, if for fixed (a^*, λ^*) the condition $T(\gamma, \alpha^*) \leq \lambda^*$ is satisfied for all γ, then (a^*, λ^*) solves Problem (10). It is important to point out that, if there exists a solution (α^*, λ^*), with corresponding values $\xi^*, \tilde{w}_p^*, \beta^*$, satisfying the condition $T(\gamma, \alpha^*) \leq \lambda^*, \quad \forall \gamma \in \Theta$, then it also satisfies the condition for all the finite sets $\Theta_f \subset \Theta$. Moreover, as briefly mentioned in Section 2, Theorem 4.2 of [6] proves that if Problem described by Eq. (13) admits solutions, then its optimal solution is defined by only a finite number of β_γ different from zero. Therefore, the algorithm that follows search for a finite set Θ_f as a solution of our problem. Based on the above calculations, we propose an iterative 3-step algorithm for IK-ELM, schematized in Algorithm 1. With reference to the inner problem, we again define an outer problem $\min_\beta \mathrm{S}(\beta)$, with $\mathrm{S}(\beta) := \min_{\tilde{w},\xi} \quad \frac{1}{2} \sum_{p=1}^{l} \frac{||\tilde{w}_p||^2}{\beta_p} + \frac{C}{2} \sum_{i=1}^{N} ||\xi^i||^2$. First, we solve the most internal part of Problem (10) with values of vector β fixed, secondly we take into account the minimization with respect to β over the unit simplex. Once these steps are completed, we look for the value of $\gamma \in \Theta$ maximizing the function $T(\gamma, \alpha)$.

Note that, step 1 of Algorithm 1 consists of solving a classical minimization problem with respect to $\tilde{w}$ and ξ, with value of β fixed. Therefore, any optimization algorithm, e.g., Gradient Descent, Iterative methods, may be used to do it, finding the optimal solution ξ^* and, consequently, α^* using Eq. (12c). We now focus on step 2 of the presented algorithm. It involves the search of a vector of fixed size β, minimizing the function $\mathrm{S}(\beta)$ over the unit simplex $\Delta := \{\beta : \mathrm{e}^{\mathrm{T}}\beta = 1, \ \beta \geq 0\}$. Although optimizing over a simplex may be a difficult task [11], many simple to implement approaches may be used to solve it, e.g., Genetic Algorithm, Accelerated Projected Gradient Descent, Exponentiated Gradient Descent. Note that, once we find the value β^*, we can compute λ^* using the corresponding KKT conditions. Finally, with respect to step 3, the value of function $T(\gamma, \alpha^*)$ plays the role of some kind of evaluation threshold, specifying whether or not a new kernel may be selected. On the subject of convergence of the proposed IK-ELM algorithm, since it may be considered as an "exchange method", Theorem 7.2 in [6] guarantees that, if a faced problem admits solution, either Algorithm 1 ends after a finite number of steps, leading to a solution of the problem, or it has at least one accumulation point each one solving the algorithm. Namely, similarly to [5], there is no assurance that, given a feasible problem, Algorithm 1 would find a finite set Θ_f, but if IK-ELM algorithm converges, it obtains global optimality.

Algorithm 1: Infinite Kernel Extreme Learning Machine

Begin

Given Θ, select $\Theta_f \subset \Theta$;

Step 1 Solve

$$\mathrm{S}(\beta) := \min_{\tilde{w},\xi} \quad \frac{1}{2}\sum_{p=1}^{l}\frac{||\tilde{w}_p||^2}{\beta_p} + \frac{C}{2}\sum_{i=1}^{N}||\xi^i||^2$$

$$\text{subject to} \quad \sum_{p=1}^{l}\langle \tilde{w}_p, \phi_p(x^i)\rangle = y^i - \xi^i, \quad \forall i = 1,\ldots,N$$

To obtain ξ^*, α^*;

Step 2 Solve

$$\min_{\beta} \quad \mathrm{S}(\beta)$$

$$\text{subject to} \quad \mathrm{e}^{\mathrm{T}}\beta = 1, \quad \beta \geq 0$$

To obtain β^*, λ^*;

Step 3 Compute

$$\max_{\gamma\in\Theta} \quad T(\gamma, \alpha^*)$$

if $T(\gamma, \alpha^*) \leq \lambda^*$ **then**

  STOP ;

else

  add γ to Θ_f and return to Step 1;

end

end

5 Conclusions

In this paper we proposed a new algorithm called Infinite Kernel Extreme Learning Machine based on the key idea of combining ELM structure together with IKL, i.e., using in theory an uncountable infinite set of base kernels. The automatic selection of the best kernel has already proved beneficial for SVM and is here extended to ELM. After delineating the marking lines of ELM algorithm, we focused on discussing the IKL approach and possible combination of ELM and IKL, describing an existing research work done combining IKL in SVM structure. In this regard, we proposed our 3-step algorithm merging ELM and IKL, considering as a starting point the research work done by Liu et al. [13].

References

1. Argyriou, A., Hauser, R., Micchelli, C.A., Pontil, M.: A DC-programming algorithm for kernel selection. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 41–48. ACM, New York (2006)
2. Bach, F.R., Lanckriet, G.R.G., Jordan, M.I.: Multiple kernel learning, conic duality, and the SMO algorithm. In: Proceedings of the Twenty-First International Conference on Machine Learning, p. 6. ACM, New York (2004)
3. Bartlett, P.L.: The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. IEEE Trans. Inf. Theory **44**(2), 525–536 (1998)
4. Cho, Y., Saul, L.K.: Kernel methods for deep learning. Adv. Neural Inf. Proces. Syst. 342–350 (2009)
5. Gehler, P.V., Nowozin, S.: Infinite kernel learning. Technical Report 178. Max-Planck Institute for Biological Cybernetics, Tübingen (2008)
6. Hettich, R., Kortanek, K.O.: Semi-infinite programming: theory, methods, and applications. SIAM Rev. **35**(3), 380–429 (1993)
7. Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of 2004 IEEE International Joint Conference on Neural Networks, vol. 2, pp. 985–990. IEEE (2004)
8. Huang, G.-B., Chen, L., Siew, C.K., et al.: Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans. Neural Netw. **17**(4), 879–892 (2006)
9. Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: Extreme learning machine: theory and applications. Neurocomputing **70**(1-3), 489–501 (2006)
10. Jian, L., Xia, Z., Liang, X., Gao, C.: Design of a multiple kernel learning algorithm for LS-SVM by convex programming. Neural Netw. **24**(5), 476–483 (2011)
11. De Klerk, E., Hertog, D.D., Elabwabi, G.: On the complexity of optimization over the standard simplex. Eur. J. Oper. Res. **191**(3), 773–785 (2008)
12. Lanckriet, G.R.G., Cristianini, N., Bartlett, P., Ghaoui, L.E., Jordan, M.I.: Learning the kernel matrix with semidefinite programming. J. Mach. Learning Res. **5**(Jan), 27–72 (2004)
13. Liu, X., Wang, L., Huang, G.-B., Zhang, J., Yin, J.: Multiple kernel extreme learning machine. Neurocomputing **149**, 253–264 (2015)
14. Liu, Y., Liao, S., Lin, H., Yue, Y., Wang, W.: Infinite kernel learning: generalization bounds and algorithms. In: Thirty-First AAAI Conference on Artificial Intelligence (2017)
15. Saunders, C., Gammerman, A., Vovk, V.: Ridge regression learning algorithm in dual variables. In: Proceedings of the 15th International Conference on Machine Learning, pp. 515–521. Morgan Kaufmann (1998)
16. Schölkopf, B., Smola, A.J., Bach, F., et al.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2002)
17. Shawe-Taylor, J., Cristianini, N.: An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods, vol. 204. Cambridge University Press, Cambridge (2000)
18. Sonnenburg, S., Rätsch, G., Schäfer, C., Schölkopf, B.: Large scale multiple kernel learning. J. Mach. Learn. Res. **7**(Jul), 1531–1565 (2006)
19. Suykens, J.A.K., Vandewalle, J.: Least squares support vector machine classifiers. Neural. Process. Lett. **9**(3), 293–300 (1999)
20. Suykens, J.A.K., Van Gestel, T., De Brabanter, J.: Least Squares Support Vector Machines. World Scientific, Singapore (2002)

Least Action Principles and Well-Posed Learning Problems

Alessandro Betti and Marco Gori

Abstract Machine Learning algorithms are typically regarded as appropriate optimization schemes for minimizing risk functions that are constructed on the training set, which conveys statistical flavor to the corresponding learning problem. When the focus is shifted on perception, which is inherently interwound with time, recent alternative formulations of learning have been proposed that rely on the principle of Least Cognitive Action, which very much reminds us of the Least Action Principle in mechanics. In this paper, we discuss different forms of the cognitive action and show the well-posedness of learning. In particular, unlike the special case of the action in mechanics, where the stationarity is typically gained on saddle points, we prove the existence of the minimum of a special form of cognitive action, which yields forth-order differential equations of learning. We also briefly discuss the dissipative behavior of these equations that turns out to characterize the process of learning.

Keywords Least cognitive action · Online learning · Variational methods for learning

1 Introduction

Whenever a learning process is embedded in a temporal environment; i.e. the data presented to the agent has a temporal structure (video and audio signals for example) it seems natural to define the learning process directly through the definition of a

A. Betti (✉)
DINFO, Firenze, Italy

DIISM, Siena, Italy
e-mail: alessandro.betti@unifi.it

M. Gori
DIISM, Siena, Italy
e-mail: marco@dii.unisi.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_10

suitable temporal dynamics. In other words one might start to think that the updating of the model's parameters, which is what we usually call "learning", must be synced with the temporal structure of data. This suggests investigating the continuous map $t \mapsto w(t)$ as a response to the input $u(t)$, thus regarding t as time and not simply an iteration index of popular machine learning algorithms.

In order to be able to select the correct dynamics of the weights of an agent we believe that a functional formulation of the problem is particularly useful. For example, the Lagrangian formulation of physical theories offers the possibility of imposing all the symmetries of a theory simply adding to the Lagrangian terms that satisfy such symmetry (see for example [1]). In the same way [2], this approach makes it easier to incorporate constraints on the dynamic of the learned weights. A variational approach based on an integral functional like the action of classical mechanics can be conceived which specifies in one single scalar function (what in mechanics is called the Lagrangian) both the "static" goodness criterion, the potential, and the dynamical part of learning by a kinetic term [3].

For example, consider a classical batch problem in machine learning where the functional risk has been approximated with a function $V(w)$. As we will discuss in Sect. 2 we can find appropriate functional indexes that have as stationarity condition the following differential equation

$$m\ddot{w} + \eta\dot{w} + \nabla V(w) = 0 \qquad m, \eta > 0. \tag{1}$$

This equation can be considered as the continuous form of a classic multistep first order method (see [4]) known as the *heavy ball* method. The name of this method derive from the fact that Eq. (1) can be interpreted as the equation of motion of an heavy ball with friction subject to the potential $V(w)$. Equation (1) is also closely related to the continuous approximation of other first order methods (see [5]). More directly in the case $m \to 0$ and η fixed we get the continuous version of a plain gradient descent method with learning rate $1/\eta$:

$$\dot{w} = -\frac{1}{\eta}\nabla V(w).$$

Notice the importance of the first order term in Eq. (1); without dissipation we wouldn't be able to recover the classical gradient descent method. Even worse, in general without the presence of the η term there is no hope for the dynamic to reach a stationary point of V. Indeed, broadly speaking, since in that case the mechanical energy would be conserved lower values of V correspond to higher values of the velocity so that the system do not have any chance to settle in a minimum of the potential.

More generally, as we already stated we believe that this "dynamical" approach to ML can be particularly fruitful when we want to consider online learning problems, that is to say problems where the temporal evolution of the parameters of the model at a certain stage of development depends explicitly on the data presented to the agent at the same time. This means that it is particularly important to handle the

case in which the potential depends on time also trough a signal $u(t)$. Under this assumption Eq. (1) assumes the form

$$m\ddot{w}(t) + \eta\dot{w}(t) + \nabla U(w(t), u(t)) = 0.$$

This equation, in the limit $m \to 0$ yields

$$\dot{w}(t) = -\frac{1}{\eta}\nabla U(w(t), u(t)),$$

that can be interpreted as the continuous counterpart of a stochastic gradient descent method, when $u(t)$ is interpreted as the realization of the random variable associated with the data at the step t. It is important to realize that whereas SGD is typically used in ML assuming that the values of $u(t)$ are drawn from a training set according to some probability distribution it is only when formulating the problem using a signal $u(t)$ which has a temporal regularity (coherence) that we can properly speak of online learning.

The paper is organized as follows: In Sect. 2 we will show how to reformulate least action principles in a more precise manner following what has been done in [6], Sect. 3 then shows how to extend some of the results of [6] (namely the existence of the minimum for approximating problems) also in the particularly interesting case where the potential explicitly depends on time. Eventually Sect. 4 closes the paper with some final considerations.

2 Lagrangian Mechanics

Following the approach proposed in [6], we will now discuss how it is possible to reformulate, in a more precise manner, the least action principle in classical mechanics. The following approach can be directly applied, in the case of dissipative dynamics, to learning processes simply through the identification of the generalized coordinates of mechanics with the parameters of the learning model (Table 1). In the remainder of the paper we will replace the variable w which we used in the introduction to stress the connection with the typical parameters (weights) used in ML with the generic coordinates $\boldsymbol{q}$.

Table 1 Links between learning theory and classical mechanics

Learning	Mechanics	Remarks
w	$\boldsymbol{q}$	Weights and neuronal outputs are interpreted as generalized coordinates
$\dot{w}$	$\dot{\boldsymbol{q}}$	Weight variations and neuronal variations are interpreted as generalized velocities

Usually (see [7] and [8]) Hamilton's principle is formulated as follows: Newton's laws of motion

$$\frac{d}{dt}(m\dot{\boldsymbol{q}}_i(t)) + \nabla_i V(\boldsymbol{q}(t)) = 0, \tag{2}$$

coincide with extremals of the functional

$$\mathsf{S}(\boldsymbol{q}) := \int_0^T L\,dt, \quad \text{where} \quad L = \frac{1}{2}m|\dot{\boldsymbol{q}}|^2 - V(\boldsymbol{q}), \tag{3}$$

where $|\cdot|$ is the n-dimensional Euclidean norm. This statement is usually also called *least action principle* even though it is well known that the trajectory $\boldsymbol{q}(t)$ is not always a minimum for the action. Another unsatisfactory aspect of this principle is the way in which the initial conditions are handled; in Newtonian mechanics Eq. (2) is typically coupled with Cauchy initial conditions

$$\boldsymbol{q}(0) = \boldsymbol{q}^0, \qquad \dot{\boldsymbol{q}}(0) = \boldsymbol{q}^1, \tag{4}$$

that uniquely determine the motion of the system. On the other hand Eq. (2) cannot be obtained from Hamilton's principle with conditions (4); usually the derivations make use of Dirichlet boundary conditions (see [7]).

It has been shown (in [6]) that Hamilton's principle can be replaced by a *minimization* problem together with a limiting procedure. In particular, let us consider the functionals

$$\mathsf{W}_\varepsilon(\boldsymbol{q}) := \int_0^T e^{-t/\varepsilon}\left(\frac{\varepsilon^2 m}{2}|\ddot{\boldsymbol{q}}(t)|^2 + V(\boldsymbol{q}(t))\right)dt, \tag{5}$$

defined on the set $\mathrm{dom}(\mathsf{W}_\varepsilon) := \{\boldsymbol{q} \in H^2((0,T);\mathbb{R}^n) \mid \boldsymbol{q}(0) = \boldsymbol{q}^0, \dot{\boldsymbol{q}}(0) = \boldsymbol{q}^1\}$, where $V \in C^1(\mathbb{R}^n)$ and bounded from below and $m > 0$.

The first property of this functional is that it admits a minimizer on its domain; actually adding little bit of regularity on V and choosing ε sufficiently small the minimizer turns out to be unique (for a precise statement of this result see Lemma 4.1 of [6]). Moreover the Euler-Lagrange equations for the minimizers of W_ε are (see Section 4 of [6])

$$\varepsilon^2 m\boldsymbol{q}^{(4)}(t) - 2\varepsilon m\boldsymbol{q}^{(3)}(t) + m\ddot{\boldsymbol{q}}(t) + \nabla V(\boldsymbol{q}(t)) = 0 \quad t \in (0,T), \tag{6}$$

$$\boldsymbol{q}(0) = \boldsymbol{q}^0, \quad \dot{\boldsymbol{q}}(0) = \boldsymbol{q}^1, \tag{7}$$

$$\ddot{\boldsymbol{q}}(T) = \boldsymbol{q}^{(3)}(T) = 0. \tag{8}$$

Notice that from the stationarity condition of (5) we get two extra boundary conditions at time $t = T$ that seems to destroy causality of the solution; one of the strengths of this approach however is that, unlike Hamilton Principle, the boundary

conditions (8) will disappear in the limit $\varepsilon \to 0$ leaving the solution dependent only on the initial state.

In the same limit ($\varepsilon \to 0$), we have that if $\boldsymbol{q}_\varepsilon$ solves (6)–(8), then (Theorem 4.2 of [6]) $\boldsymbol{q}_\varepsilon \to \boldsymbol{q}$ weakly in $H^1((0,T);\mathbb{R}^n)$, where $\boldsymbol{q}$ solves (2) with (4). This last assertion makes clear that Hamilton principle can be reformulated in terms of (5) in the following way:

1. For each fixed ε minimize W_ε,
2. take the limit $\varepsilon \to 0$.

Like Hamilton's principle this procedure is a variational approach to classical mechanics, with respect to the principle of least cognitive action however, as anticipated, it involves a true minimization of the functional (5) and it automatically reaches causality.

It is interesting to notice that if we omit step 2. in the procedure described above, stationarity conditions of (5) would imply a dynamic based on differential equations of order higher than two (which has been actually considered in physics [9] and [10]). However the presence of the right boundary conditions (8) for each $\varepsilon > 0$ would render the resulting laws non-causal.

To conclude this section we will discuss what can be considered yet another advantage of this approach by showing how naturally it can handle dissipative dynamics.

Dissipative Dynamics In the introduction we have briefly discussed how dissipation is a fundamental feature for the formulation of learning ad as a dynamical process; for this reason this point deserves a careful discussion.

First of all notice that it is not possible to modify L in Eq. (3) by choosing an appropriate V or by adding additional derivative terms in order to reproduce the following dissipative dynamics:

$$m\ddot{\boldsymbol{q}} + \eta\dot{\boldsymbol{q}} + \nabla V(\boldsymbol{q}) = 0, \tag{9}$$

with $\eta > 0$. Nevertheless it has been shown (see [11] and [3]) that it is possible to include this kind of dynamic by the following modification of the action:

$$\mathsf{S}(\boldsymbol{q}) \to \overline{\mathsf{S}}(\boldsymbol{q}) := \int_0^T e^{\eta t/m}\left(\frac{1}{2}m|\dot{\boldsymbol{q}}|^2 - V(\boldsymbol{q})\right)dt.$$

This formulation changes the structure of the action functional making it more similar to the W_ε functional. Still this variational approach suffers of the same problems that has been discussed previously in this section.

On the other hand in order to include dissipation in (5) it is sufficient to modify the W_ε functional in the following way:

$$\mathsf{W}_\varepsilon(\boldsymbol{q}) \to \overline{\mathsf{W}}_\varepsilon(\boldsymbol{q}) := \int_0^T e^{-t/\varepsilon}\left(\frac{\varepsilon^2 m}{2}|\ddot{\boldsymbol{q}}(t)|^2 + \frac{\varepsilon\eta}{2}|\dot{\boldsymbol{q}}(t)|^2 + V(\boldsymbol{q}(t))\right)dt.$$

Then through the same minimization and limiting procedure described above we recover Eq. (9) together with the correct initial conditions (4).

The modification $\mathsf{W}_\varepsilon(\boldsymbol{q}) \rightarrow \overline{\mathsf{W}}_\varepsilon(\boldsymbol{q})$ feels less artificial than $\mathsf{S}(\boldsymbol{q}) \rightarrow \overline{\mathsf{S}}(\boldsymbol{q})$ and the term added to W_ε seems a natural term to add. The reason why the dissipative behaviour is recovered so easily by the variational approach based on W_ε is that this principle is not invariant by time reversal to begin with.

3 Generalization to Time-Dependent Potential

The analysis presented in this section extends the result on the existence of a minimizer to a family of functionals that include (5) where, in particular, we allow an explicit dependence on time through the potential.

The following theory is relevant at least for two distinct reason; first of all it is a first result that goes in the direction of extending the theory presented in [6]. In second place it is interesting in its own (i.e. also if it is not coupled with a limiting procedure) to ensure well-posedness of theories that relies on the minimization of a functional of the form that we will consider. Recently learning theories based on variational indexes considered in this section has been used in Vision; in particular the proposed theory has been directly applied to the problem of feature extraction from a video signal $u(t)$ in an unsupervised manner with the potential U chosen to be the mutual information between the visual data and a set of symbols (see [2]).

Let $T \in (0, \infty)$, $U \in C^0(\mathbb{R}^n \times \mathbb{R}^m)$ be bounded from below such that $U(\cdot, 0) \equiv 0$ and $\varpi \in L^\infty(0, T)$ with $0 < C_1 \leq \varpi(t) \leq C_2 < +\infty$ for a.e. $t \in (0, T)$. Let $u\colon [0, +\infty) \rightarrow \mathbb{R}^m$ be an external input function that for the moment can be considered a continuous function of time. Consider the functional

$$\Gamma(\boldsymbol{q}) = \int_0^T \varpi(t)\Big(\frac{\mu}{2}|\ddot{\boldsymbol{q}}(t)|^2 + \frac{\nu}{2}|\dot{\boldsymbol{q}}(t)|^2 + \gamma\dot{\boldsymbol{q}}(t)\cdot\ddot{\boldsymbol{q}}(t) + \frac{\kappa}{2}|\boldsymbol{q}(t)|^2 + U\big(\boldsymbol{q}(t), u(t)\big)\Big)\,dt, \tag{10}$$

where $\mu = \alpha + \gamma_2^2$, $\nu = \beta + \gamma_1^2$, $\gamma = \gamma_1\gamma_2$, $\kappa > 0$ are real numbers so that (10) can always be rewritten as

$$\Gamma(\boldsymbol{q}) = \int_0^T \varpi(t)\Big(\frac{\alpha}{2}|\ddot{\boldsymbol{q}}(t)|^2 + \frac{\beta}{2}|\dot{\boldsymbol{q}}(t)|^2 + \frac{1}{2}|\gamma_1\dot{\boldsymbol{q}}(t) + \gamma_2\ddot{\boldsymbol{q}}(t)|^2 + \frac{\kappa}{2}|\boldsymbol{q}(t)|^2 + U\big(\boldsymbol{q}(t), u(t)\big)\Big)\,dt,$$

with α, β, real and positive and $\boldsymbol{q} \in \mathrm{dom}(\Gamma) := \{\, \boldsymbol{q} \in H^2((0, T); \mathbb{R}^n) \mid \boldsymbol{q}(0) = \boldsymbol{q}^0, \quad \dot{\boldsymbol{q}}(0) = \boldsymbol{q}^1 \,\}$, where $\boldsymbol{q}_0, \boldsymbol{q}_1 \in \mathbb{R}^n$ are given.

Suppose furthermore that we equip $\mathrm{dom}(\Gamma)$ with the following notion of convergence:

$$\begin{aligned} \boldsymbol{q}_k &\to \boldsymbol{q} \qquad \text{strongly in } H^1((0,T);\mathbb{R}^n); \\ \ddot{\boldsymbol{q}}_k &\rightharpoonup \ddot{\boldsymbol{q}} \qquad \text{weakly in } L^2((0,T);\mathbb{R}^n). \end{aligned} \tag{11}$$

Then the following remark holds:

Remark 1 The set $\mathrm{dom}(\Gamma)$ is closed under the convergence in (11), i.e., if $\boldsymbol{q}_k \in \mathrm{dom}(\Gamma)$, $\boldsymbol{q}_k \to \boldsymbol{q}$ in $\mathrm{dom}(\Gamma)$, then $\boldsymbol{q} \in \mathrm{dom}(\Gamma)$.

Indeed, since $H^1(0,T)$ compactly embeds in $C([0,T])$ (see [12] p. 213 Eq. (6)) and a weakly convergence sequence is strongly bounded ([12] Prop. 3.5 (iii)), $\langle \boldsymbol{q}_k \rangle$ has a (not relabelled) subsequence such that $\boldsymbol{q}_k \to \boldsymbol{q}$ and $\dot{\boldsymbol{q}}_k \to \dot{\boldsymbol{q}}$ uniformly in $[0,T]$, therefore $\boldsymbol{q}(0) = \boldsymbol{q}^0$ and $\dot{\boldsymbol{q}}(0) = \boldsymbol{q}^1$.

We are now in the position to state the main result on the existence of a minimum of the functional in (10).

Theorem 1 *The problem* $\min\{\, \Gamma(\boldsymbol{q}) \mid \boldsymbol{q} \in \mathrm{dom}(\Gamma) \,\}$*, has a solution.*

Proof We simply apply the direct method in the calculus of variations, namely we have to show that Γ is lower semicontinuous and coercive with respect to the convergence in (11) and then we conclude in view of Remark 1.

Lower Semicontinuity The maps $\boldsymbol{q} \in \mathrm{dom}(\Gamma) \mapsto \int \varpi(t)|\boldsymbol{q}(t)|^2\,dt$ and $\boldsymbol{q} \in \mathrm{dom}(\Gamma) \mapsto \int \varpi(t)|\dot{\boldsymbol{q}}(t)|^2\,dt$ are continuous, while $\boldsymbol{q} \in \mathrm{dom}(\Gamma) \mapsto \int \varpi(t)|\ddot{\boldsymbol{q}}(t)|^2\,dt$ is lower semicontinuous (see [12] Prop. 3.5 (iii)); moreover $\boldsymbol{q} \in \mathrm{dom}(\Gamma) \mapsto \int \varpi(t)\dot{\boldsymbol{q}}(t) \cdot \ddot{\boldsymbol{q}}(t)\,dt$ is continuous because of the strong-weak convergence of the scalar product in a Hilbert space (see [12] Prop. 3.5 (iv)). Finally the map $\boldsymbol{q} \in \mathrm{dom}(\Gamma) \mapsto \int \varpi(t)U(\boldsymbol{q}(t), u(t))$ is lower semicontinuous because of our assumptions on U and as a direct consequence of Fatou's Lemma.

Coercivity Since U is bounded from below and $T < +\infty$ and in view of our assumptions on w, α, β, κ it immediately follows that if $\sup_{k\in\mathbb{N}} \Gamma(\boldsymbol{q}_k) < +\infty$, then there exists a constant $C > 0$ such that $\|\boldsymbol{q}_k\|_{H^2} \le C$ for every $k \in \mathbb{N}$. Then from Theorem 3.16 in [12] it follows that $\langle \boldsymbol{q}_k \rangle$ has a subsequence weakly converging in $H^2(0,T)$. Moreover since $H^2(0,T)$ compactly embeds in $H^1(0,T)$ then there is a subsequence that converges strongly in $H^1(0,T)$. This means that indeed the sublevels of Γ are compact with respect to the convergence in Eq. (11).

□

4 Conclusions

In this paper we presented an extension of the minimality result discovered in [6] that entails the well-posedness of a class of learning problems based on a Least Action Principle defined over the class of functionals (10). We prove that the

existence of the minimum of Γ (Theorem 1) holds for a general weight function ϖ. Moreover, we argue that since learning requires dissipation, the correspondent dynamics can be reproduced from (10) by choosing ϖ as an exponential function of time, as discussed in Sect. 2. This paper provides motivations to use the variational framework initially proposed in [3], since it shows that, unlike the action of mechanics, the opportune selections of the cognitive action leads to well-posed learning problems where a global minimum can be discovered.

Acknowledgements We thank Giovanni Bellettini for having brought to our attention the extended formulation of Newtonian mechanics and for insightful discussions.

References

1. Weinberg, S.: The quantum theory of fields. In: Foundations, vol. 1. Cambridge University Press, Cambridge (1995)
2. Betti, A., Gori, M., Melacci, S.: Cognitive action laws: the case of visual features. In: IEEE Transaction on Neural Networks and Learning Systems (2019). arXiv:cs.CV/1808.09162v1
3. Betti, A., Gori, M.: The principle of least cognitive action. Theor. Comput. Sci. **633**, 83–99 (2016)
4. Polyak, B.T.: Some methods of speeding up the convergence of iteration methods. USSR Comput. Math. Math. Phys. **4**, 1–17 (1964)
5. Su, W., Boyd, S., Candes, E.: A differential equation for modeling Nesterov's accelerated gradient method: theory and insights. In: Advances in Neural Information Processing Systems, pp. 2510–2518 (2014)
6. Liero, M., Stefanelli U.: A new minimum principle for Lagrangian mechanics. J. Nonlinear Sci. **23**, 179–204 (2013)
7. Arnold, V.I.: Mathematical methods of classical mechanics. Grad. Texts Math. **60** (1989)
8. Goldstein, H., Poole, C., Safko, J.: Classical Mechanics. Addison Wesley, Boston (2002)
9. Suykens, J.A.K: Extending Newton's law from nonlocal-in-time kinetic energy. Phys. Lett. A **373**(14), 1201–1211 (2009)
10. El-Nabulsi, R.A.: On maximal acceleration and quantum acceleratum operator in quantum mechanics. Quantum Stud. Math. Found. **5**(4), 543–550 (2018)
11. Herrera, L., Nunez, L., Patino, A., Rago, H.: A variational principle and the classical and quantum mechanics of the damped harmonic oscillator. Am. J. Phys. **54**(3), 273–277 (1986)
12. Brezis, H.: Functional Analysis, Sobolev Spaces and Partial Differential Equation. Springer, Berlin (2010)

Heuristic Data-Driven Feasibility on Integrated Planning and Scheduling

Marco Casazza and Alberto Ceselli

Abstract We study the merging of data-driven approaches and mathematical programming formulations to solve an integrated planning and scheduling problem where jobs can be split in two separate tasks, one of them allowed to exceed its deadline at a price. Our study is driven by the increasing structural complexity of industrial scheduling problems that in some cases become too hard to be modeled as mathematical programs even by domain experts. We experiment on how to ensure the feasibility at a scheduling level by training a data-driven model, subsequently encoding it with a mathematical programming formulation, to be finally embedded in a planning model. Our experiments prove that our framework provides an effective heuristic approach, competing to exact formulations in terms of both accuracy and quality of the solutions, and it could be extended to those kind of problems where it is too hard to model the schedule feasibility.

Keywords Integrated planning scheduling · Data-driven · Decision tree · On time in full

1 Introduction

Decision support systems play a key role in logistics. Many production companies, indeed, enrich their Enterprise Resource Planning (ERP) software with dedicated components as Material Resource Planning (MRP) systems for inventory management and Advanced Planning and Scheduling (APS) tools for day-by-day operational production schedules. While technologically designed to meet industrial integration standards, these tools often lack optimization potential. For instance, MRPs often assume *infinite* production capacity. Such an approximation yields task allocations which are found to be infeasible by the operators while running

M. Casazza (✉) · A. Ceselli
Dipartimento di Informatica, Università degli Studi di Milano, Crema, Italy
e-mail: marco.casazza@unimi.it; alberto.ceselli@unimi.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_11

APS to create schedules, or even during the *execution* of them, thereby leading to disruptions of planning. This phenomenon is critical in settings like cosmetics manufacturing [1, 6]. Orders are highly custom; products are very seasonal and need to hit the market in a timely way. Deadlines are tight, lead times uncertain, and production loads very dynamic. As a consequence, delivery dates are often violated and the perceived quality of service is substantially lowered.

One common reason to resort to infinite capacity approximation is computational: integrated planning and scheduling models are difficult to manage. Several times an even more involved issue arises: it is too difficult to obtain *reliable models* of production capacity. That is, often not even a domain expert is able to formalize the complexity of a scheduling process in a mathematical model. It is certainly the case of cosmetics manufacturing, but in general of any scenario where production times are influenced by workers skills [5], which are too difficult to track in input data.

We therefore propose an alternative approach to the problem, consisting of the following methodology. First, a framework based on a mathematical programming formulation for integrated planning and scheduling is built. Second, the feasibility of scheduling in a particular time slot is encoded by a data-driven model, which is created by training on historical data. Third, such a model is used as *white box*, drawing a mathematical programming formulation of it. Fourth, the scheduling feasibility constraints in the starting framework are replaced by such a formulation, and the overall mathematical program can finally be solved by general purpose tools.

Indeed, there is a hype for integrating mathematical programming and machine learning in this kind of context. A few attempts have showed to be successful; as key examples we mention the use of machine learning to obtain aggregate solutions of operational problems [7], and the formulation of machine learning models through mathematical programming [4].

In this paper we present a proof of concept, indicating our methodology to be promising. The design of such a proof of concept is not trivial since it needs to be realistic from an application point of view, but still computationally manageable, to be able to compare the outcome of our experiments with that of straight optimization.

Therefore, we first introduce an overall mathematical programming framework for integrated planning and scheduling (Sect. 2). We restrict the planning part to a simple assignment model, and we choose a realistic scheduling model. The latter yields NP-Complete feasibility subproblems, which are however tractable in practice by general purpose solvers. Then, we describe how to replace the scheduling part by a suitable encoding of a Decision Tree classification model (Sect. 3) and selected features of scheduling instances. Finally (Sect. 4) we provide experimental evidence that, if a careful training of the Decision Tree is carried out, our data-driven method is highly effective in approximating the set of feasible solutions of the starting mathematical model, offering at the same time more modeling flexibility.

2 Problem Description

We consider a set of jobs J and a set of machines M to be given. We suppose that each job $j \in J$ has a release date r_j, a deadline d_j and a processing time p_j. We also suppose to have a set W of production time slots each of $\mathcal{H}$ time units. Each job in J must be assigned to one of the slots in W (planning), and a proper sequencing of the jobs in the same slot on the machines in M must be found (scheduling). Assigning job $j \in J$ to slot $w \in W$ has a cost c_{jw}.

An overall framework formulation for our Integrated Planning and Scheduling problem (IPSP) is the following:

$$\text{minimize} \sum_{w \in W} \sum_{j \in J} c_{jw} \cdot u_{jw} \tag{1}$$

$$\text{s.t. } (u_{jw}) \in P^j \qquad \forall j \in J \tag{2}$$

$$(u_{jw}) \in S^w \qquad \forall w \in W \tag{3}$$

where each variable $u_{jw} \in \mathbb{B}$ takes value 1 if job j is planned for time slot w, 0 otherwise. The objective function (1) minimizes the assignment cost. Planning constraints are implicitly modeled by (2). Similarly, scheduling constraints are implicitly modeled by (3).

Different choices of P^j and S^w give rise to different integrated problems. As a case study, in our proof of concept we consider the following. We allow for split assignment options, which are designed to model those scenarios where manufacturers cannot fully complete the production before the deadline, but after negotiations the customer agrees to receive only part of the production before the deadline and the rest afterwards, as soon as possible [3]. Formally, two types of assignments are made possible: **full**—a job is assigned to a single machine and scheduled in such a way that it starts after the release date and it ends before the deadline, and **split**—a job is split in two tasks, potentially assigned to different machines and different time slots. The first task is scheduled between the release date and the deadline, while the second task is scheduled after the first, potentially exceeding the deadline. We model such a split option by considering a set $\bar{J}$ containing the original (unsplit) jobs, a set J' (resp. J'') containing the first (resp. second) task after split. Therefore $J = \bar{J} \cup J' \cup J''$. For each $j \in \bar{J}$, we denote as j' (resp. j'') its first (resp. second) task after split. The tasks in J' keep release and deadlines of the jobs in $\bar{J}$; the tasks in J'' keep the release, but have infinite deadline. We assume the processing times of tasks in J' and J'' to be data, obtained in the negotiation phase, such that the sum of processing times of the two tasks after split equals the processing time of the initial job. In this case, a pertinent objective function is to minimize the number of splits (i.e. $c_{jw} = 1$ for each $j \in J$ and

$w \in W$). The sets P^j can be modeled as follows, for each $j \in \bar{J}$

$$P^j = \text{set of } (u_{jw}, u_{j'w}, u_{j''w})$$

$$\text{s.t.} \sum_{w \in W} u_{jw} + u_{j'w} = 1 \tag{4}$$

$$\sum_{w \in W} u_{jw} + u_{j''w} = 1 \tag{5}$$

$$\sum_{w'' \leq w} u_{j''w''} \leq 1 - u_{j'w} \qquad \forall w \in W \tag{6}$$

Conditions (4) and (5) guarantee that either j or both j' and j'' are selected; conditions (6) impose j' to always be planned before j''. Such a choice meets the first of our requirements, which is to have a simple planning structure.

For what concerns scheduling feasibility, each set S^w can be modeled as follows:

$$S^w = \text{set of } (u_{jw}, u_{j'w}, u_{j''w})$$

$$\text{s.t. } (w-1) \cdot \mathcal{H} \cdot u_{jw} \leq s_j \qquad \forall j \in J \tag{7}$$

$$e_t \leq w \cdot \mathcal{H} \cdot u_{jw} + \mathcal{M} \cdot (1 - u_{jw}) \qquad \forall j \in J \tag{8}$$

$$\sum_{m \in M} x_{jm} = u_{jw} \qquad \forall j \in J \tag{9}$$

$$x_{im} + x_{jm} \leq y_{ij} + y_{ji} + 1 \qquad \forall i, j \in J, m \in M \tag{10}$$

$$r_j \leq s_j \qquad \forall j \in J \tag{11}$$

$$e_j \leq d_j \qquad \forall j \in \bar{J} \cup J' \tag{12}$$

$$s_j + p_j = e_j \qquad \forall j \in J \tag{13}$$

$$e_i \leq s_j + \mathcal{M} \cdot (1 - y_{ij}) \qquad \forall i, j \in J \tag{14}$$

where variable $x_{jm} \in \mathbb{B}$ is 1 if a job j is assigned to machine m, and 0 otherwise, while variable $y_{ij} \in \mathbb{B}$ is 1 if job i precedes job j on the same machine, 0 otherwise. Finally, continuous variables $s_j \geq 0$ and $e_j \geq 0$ are the starting and ending time of job j, respectively.

Constraints (7) and (8) impose that a job starts and ends in the selected time slot w. Constraints (9) impose the assignment of each job to a machine. Constraints (10) set variables y_{ij} when two jobs are scheduled on the same machine. Constraints (11) ensure that each job starts after the release date, while constraints (12) impose that both full jobs and first tasks end before the deadline. Constraints (13) impose full processing without preemption. Constraints (14) avoid overlapping between jobs assigned to the same machine. We remark that even by fixing the value of u_{jw} variables, the problem of deciding if a feasible schedule exists in S^w remains NP-

complete. In fact, even if no release nor deadline falls inside a time slot, a Bin Packing Problem remains to be solved to detect feasibility. Nevertheless we expect general purpose solvers to be effective on optimizing the overall model.

3 Data-Driven Scheduling

In the following, we propose an alternative formulation where the scheduling feasibility of the weekly plan is ensured by a Decision Tree (DT) trained through a large dataset of (feasible and infeasible) scheduling instances, and encoded by mathematical programming constraints. The experimental tractability of both models allow us to perform a comparison of them as discussed in Sect. 4.

In this Section we introduce in turn a set of parameters that can be computed in preprocessing, the set of features that involve values of u_{jw} variables, and finally the reformulation of set S^w.

Additional Parameters We define new sets of parameters that we compute in a preprocessing phase. Given a single job $j \in J$ and a time slot $w \in W$, we define a release date r_j^w as the maximum between the release date of the job j and starting time of the time slot w, that is

$$r_j^w = \max\{r_j, (w-1) \cdot \mathcal{H}\} \tag{15}$$

and we also define a deadline d_j^w as the minimum between the deadline of the job j, and the ending time of time slot w, that is

$$d_j^w = \min\{d_j, w \cdot \mathcal{H}\} \tag{16}$$

We assume that $u_{jw} = 0$ whenever $p_j > d_j^w - r_j^w$, since any scheduling instance having a job with a processing time greater than the available time would be infeasible.

In addition, we define the *processing ratio* α_j^w as the ratio between the processing time of a job and the time available for its processing, that is

$$\alpha_j^w = \frac{p_j}{d_j^w - r_j^w}, \tag{17}$$

and also the *overlapping ratio* β_{ij}^w, that is the fraction of time that two jobs may share:

$$\beta_{ij}^w = \frac{\min\{d_i^w, d_j^w\} - \max\{r_i^w, r_j^w\}}{\max\{d_i^w, d_j^w\} - \min\{r_i^w, r_j^w\}}. \tag{18}$$

We remark that α_j^w values range in $[0, 1]$, where higher values indicate more stringent deadlines, while β_{ij}^w values range in $[-1, 1]$, where negative values mean jobs that never overlap, while high positive values mean that two jobs largely overlap in time.

Selected Features During preliminary experimental analysis we highlighted that the following features have a predictive power in detecting feasibility:

- the number n^w of jobs planned in a time slot w:

$$n^w = \sum_{j \in J} u_{jw}, \text{ with } n^w \geq 0 \tag{19}$$

- the number ρ^w of planned jobs per machine:

$$\rho^w = \frac{n^w}{|M|}, \text{ with } \rho^w \geq 0 \tag{20}$$

- the mean μ_α^w of α_j^w values:

$$\mu_\alpha^w = \frac{\sum_{j \in J} \alpha_j^w \cdot u_{jw}}{n^w}, \text{ with } 0 \leq \mu_\alpha^w \leq 1 \tag{21}$$

- the mean μ_β^w of β_{ij}^w values:

$$\mu_\beta^w = \frac{\sum_{i,j \in J} \beta_{ij}^w \cdot u_{iw} \cdot u_{jw}}{(n^w)^2}, \text{ with } -1 \leq \mu_\beta^w \leq 1 \tag{22}$$

- the number ν^w of jobs whose processing must overlap per machine:

$$\nu^w = \frac{\sum_{i,j \in J} \left\lfloor \frac{p_i + p_j}{\max\{d_i^w, d_j^w\} - \min\{r_i^w, r_j^w\}} \right\rfloor \cdot u_{iw} \cdot u_{jw}}{|M|}, \text{ with } \nu^w \geq 0 \tag{23}$$

When $n^w = 0$ we fix all features to value 0. Furthermore, we let the DT to include as a feature the number m^w of machines available in time slot w:

$$m^w = |M|, \text{ with } m^w > 0. \tag{24}$$

In order to express the DT in mathematical programming language, each feature must become a variable in our model. We can observe that nonlinearities arise due to n^w at the denominator, and the bilinear terms $u_{iw}u_{jw}$. We therefore linearise (21):

$$\sum_{j\in J} \phi^{\alpha}_{jw} = \sum_{j\in J} \alpha^{w}_{j} \cdot u_{jw} \tag{25}$$

$$0 \le \phi^{\alpha}_{jw} \le u_{jw} \qquad \forall j \in J \tag{26}$$

$$\mu^{w}_{\alpha} - (1 - u_{jw}) \le \phi^{\alpha}_{jw} \le \mu^{w}_{\alpha} + (1 - u_{jw}) \qquad \forall j \in J \tag{27}$$

$$0 \le \mu^{w}_{\alpha} \le 1 \tag{28}$$

We introduce variable $y_{ijw} \in \mathbb{B}$ imposing that $u_{iw} + u_{jw} \le 1 + y_{ijw}$ for all $i, j \in J$, and linearise equation (23) accordingly. Then we linearise equations (22) into

$$\sum_{i,j\in J} \phi^{\beta}_{ijw} = \sum_{i,j\in J} \beta^{w}_{ij} \cdot y_{ijw} \tag{29}$$

$$-y_{ijw} \le \phi^{\beta}_{ijw} \le y_{ijw} \qquad \forall i, j \in J \tag{30}$$

$$\mu^{w}_{\beta} - (1 - y_{ijw}) \le \phi^{\beta}_{ijw} \le \mu^{w}_{\beta} + (1 - y_{ijw}) \qquad \forall i, j \in J \tag{31}$$

$$-1 \le \mu^{w}_{\beta} \le 1 \tag{32}$$

We remark that μ^{w}_{α} and μ^{w}_{β}, as well as the support terms ϕ^{α}_{j} and ϕ^{β}_{ij}, become variables in our model.

Decision Tree We consider a binary DT, $\mathcal{T} = (N, A)$, which is formally a binary tree where N is the set of nodes and A the set of the arcs. We define node $0 \in N$ as the root of $\mathcal{T}$ and the set $L \subseteq N$ as the set of leaves. Each node $p \in N \setminus L$ has a right child $r_p \in N$ and left child $l_p \in N$ such that $(p, r_p), (p, l_p) \in A$. Furthermore, each node $p \in N \setminus L$ is labeled by a pair (f_p, v_p): f_p is one of the features presented above and v_p is a threshold value. The same feature can appear in different labels, possibly associated to different thresholds. Instead, each leaf $p \in L$ is labelled either as *feasible* of *infeasible*.

A DT serves as a classification tool: when a new instance appears, its feature values $\bar{f}_p$ are measured. Then, the DT is traversed from the root to a leaf as follows: at each node $p \in N \setminus L$, if $\bar{f}_p \le v_p$ then the branch corresponding to the left child l_p is chosen and explored recursively, otherwise the branch corresponding to the right child r_p is chosen. The recursive exploration is repeated until a leaf is reached: its label is then used to classify the new instance as either feasible or infeasible.

The induction of an effective DT from historical data is a combinatorial optimization problem, for which heuristics are commonly used. In our case the DT induction phase is carried out in preprocessing, as described in Sect. 4. Therefore, we assume the DT to be given as input to the overall optimization model.

The feature values $\bar{f}_p$, instead, are unknown, as they depend on the subset of jobs planned in each time slot during the optimization phase. Therefore, we encode a DT for time slot $w \in W$, using mathematical programming, as follows:

$$\sum_{j \in J} u_{jw} \leq |J| \cdot z_0^w \tag{33}$$

$$z_p^w = z_{l_p}^w + z_{r_p}^w \qquad \forall p \in N \setminus L \tag{34}$$

$$f_p^w \leq v_p \cdot z_{l_p}^w + \overline{\mathcal{M}} \cdot (1 + z_{r_p}^w - z_p^w) \qquad \forall p \in N \setminus L \tag{35}$$

$$f_p^w \geq (v_p + \epsilon) \cdot z_{r_p}^w + \underline{\mathcal{M}} \cdot (1 + z_{l_p}^w - z_p^w) \qquad \forall p \in N \setminus L \tag{36}$$

$$z_p^w = 0 \qquad \forall p \in L, \, p \text{ infeasible} \tag{37}$$

where each variable $z_p^w \in \mathbb{B}$ is 1 if a node is traversed, 0 otherwise, and each variable f_p^w is the feature variable corresponding to the feature of node p in time slot w.

Assuming that $\underline{\mathcal{M}}$ and $\overline{\mathcal{M}}$ are lower and upper bounds of both feature f_p^w and threshold v_p, we have that constraint (33) activates the DT of a time slot w when at least one job is planned in w. Constraints (34) impose that each traversed node has a traversed child. Constraints (35) and (36) select one between the left and right children to be traversed, depending on the values of the feature variables and the thresholds. Finally, conditions (37) set all the variables corresponding to infeasible leaves to 0, in such a way that our DT is forced to select values for feature variables that correspond to a feasible classification.

Then, for each time slot w, we can replace set S^w in formulation (1)–(3) with:

$$\begin{aligned} \bar{S}^w = \; & \text{set of } (u_{jw}, u_{j'w}, u_{j''w}) \\ & \text{s.t. (19)–(37)} \end{aligned} \tag{38}$$

We remark that, besides u_{jw}, the following are variables in our model: n^w, ρ^w, μ_α^w, μ_β^w, ν^w, ϕ_{jw}^α, ϕ_{ijw}^β, y_{ijw}, z_p^w.

4 Experimental Analysis

Summarizing, we have two versions of formulation (1)–(3): the first defined by the pair (P^j, S^w), that is the original one, and the second defined by $(P^j, \bar{S}^w)$, that is the one where the scheduling feasibility is led by a data-driven approach. We remark that formulation $(P^j, \bar{S}^w)$ is a heuristic approach and that its solution may not be feasible to formulation (P^j, S^w). Therefore, in the following we provide empirical evidence that the predicting accuracy of a DT holds even when such DT is integrated as a component of a bigger mathematical programming formulation.

Our experimental analysis is organised as follows: first we face the issue of inducing an effective DT from data, then we compare the accuracy of formulation $(P^j, \bar{S}^w)$ against (P^j, S^w).

Decision Tree Induction and Testing Finding an effective DT is not trivial: it needs to be produced in a preprocessing phase, before the optimization starts, and it needs to be accurate on instances produced by settings of u_{jw} variables during an optimization process. Neither the single feasibility instances nor their structure are known before the optimization process starts. Therefore, we produced a large dataset of artificial single time slot scheduling instances. We used formulation S^w as a feasibility model, we computed feasibility using exact optimization of a general purpose tool, and we classified those instances as either feasible or infeasible according to the outcome of these runs. This was computationally possible because the subproblems refer to a single time slot, and are therefore much smaller than the integrated model. Then we used such a dataset both to train a DT and to test its accuracy.

In details, our first dataset consists of 4600 randomly generated scheduling instances. Each instance has a random number of jobs $|J| \in [25, 100]$, release dates $r_j \in [0, 1000]$, processing times $p_j \in [10, 1000]$, and deadlines $d_j = r_j + p_j \cdot [1, k]$, with k from 1 to 3 by step 0.5. The number of machines was computed as $|M| = \lceil k \cdot (\sum p_j)/(\max d_j) \rceil$, with k from 0.5 to 5 by step 0.1. We then solved each scheduling instance with CPLEX 12.6.3 to classify the dataset in order to train and test the DT. In our experiment we randomly sampled the 66% of this dataset for training, while the remaining was used for testing. Both training and testing are implemented entirely in Python using Scikit-learn [2, 8].

In Table 1 we report the classification results for DTs of different depth: first we trained the DT with unbounded depth, and then we repeated the training setting a maximum depth of the DT equal to 25, 50, and 75% of the depth reached in the unbounded case. We can observe that even when the depth is reduced, high level of accuracy can be reached. For example at 50% maximum depth, both precision and recall are above 97%. The depth of the tree is of particular importance in our model, since a deeper tree corresponds to a higher number of variables and constraints.

Solving the IPSP We then created a second dataset, made of 400 randomly generated instances divided in two sets: a first set called D1 having a time horizon $W = 1 \ldots 8$, and a second, called D2, with $W = 1 \ldots 4$ time slots. Both datasets have $\mathcal{H} = 120$ time units per time slot. The two datasets have the following features:

Table 1 DT classification results for different tree depths

DT depth (%)	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
25	0.902174	0.916529	0.955211	0.935470
50	0.964834	0.973459	0.979328	0.976385
75	0.974425	0.986968	0.978467	0.982699
100	0.982737	0.987952	0.988803	0.988377

Table 2 Results obtained on dataset D1

DT depth (%)	Accuracy (%)	Feasible schedules (%)	Objective function (%)	Computing time (s)
25	100.00	95.79	−0.01	13.26
50	100.00	95.21	0.00	14.18
75	100.00	94.86	0.00	13.89
100	100.00	96.03	−0.50	19.99

Table 3 Results obtained on dataset D2

DT depth (%)	Accuracy (%)	Feasible schedules (%)	Objective function (%)	Computing time (s)
25	70.35	97.75	0.00	6.22
50	87.44	96.85	−0.02	6.29
75	96.48	96.31	−0.64	72.26
100	96.48	97.62	−1.10	33.81

a set of jobs $|J| \in [50, 100]$, a set of machines $|M| \in [5, 10]$, release dates $r_j \in [0, |W| \cdot \mathcal{H}]$, processing times $p_{j'} \in [1, 24]$ (resp. $p_j \in [1, 12]$ for D2), $p_{j''} = p_{j'} \cdot [0, 1]$, and $p_j = p_{j'} + p_{j''}$, and deadlines $d_j = r_j + p_j \cdot [1, 3]$.

We used CPLEX 12.6.3 to solve each instance of the two datasets using both model (P^j, S^w) and $(P^j, \bar{S}^w)$. First, each instance was solved to optimality using model (P^j, S^w), finding 255 feasible instances out of 400. Each instance was solved in an average computing time of 2.4 s for dataset D1 and 4.0 s for dataset D2.

Then we solved again the two datasets but using model $(P^j, \bar{S}^w)$. Results on dataset D1 are reported in Table 2, where we observe that model $(P^j, \bar{S}^w)$ detected the feasibility with an accuracy of 100%, with a reduction of the value of the objective function that is on average always less than 1%. Furthermore, we checked the feasibility of each time slot plan against S^w, observing that the number of false positives is on average less than 5%.

In Table 3 we report results on dataset D2: in this case the accuracy decreases especially for smaller depths of the DT, while DTs deeper than 50% obtain similar results. We finally mention that we found $(P^j, \bar{S}^w)$ to be consistently slower to optimize with respect to (P^j, S^w). This was indeed expected, as the details of the planning and scheduling parts were tuned not to become a computational challenge.

Conclusions and Future Works Overall, we consider our framework to be a promising starting point to solve those problems where additional flexibility is required that can hardly be achieved by classic mathematical programming formulations.

In fact the experimental results indicate that a decision tree is highly effective in approximating the scheduling feasibility stage. Computational questions remain open, and at this regard heuristic approaches are promising. Indeed heuristics can be incorporated in our framework at three stages. First, they may be effective in optimizing the full integrated problem; since feasibility is already handled heuristically, such an approach could even increase the robustness of the overall method. Second, planning heuristics can be used to drive a decomposition approach,

where subproblems are simple evaluations of data-driven models. Third, scheduling heuristics can be used for creating large datasets to enrich the training phase.

Acknowledgement Partially funded by Regione Lombardia, grant agreement n. E97F170000 00009, Project AD-COM.

References

1. AD-COM: Advanced Cosmetics Manufacturing. https://ad-com.net/
2. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth, Belmont (1984)
3. Casazza, M., Ceselli, A., Righini, G.: A single machine on-time-in-full scheduling problem. In: Accepted at Cologne-Twente Workshop on Graphs and Combinatorial Optimization (2019)
4. Fischetti, M., Jo, J.: Deep neural networks and mixed integer linear optimization. Constraints **23**(3), 296–309 (2018)
5. Fletcher, S.R., Baines, T.S., Harrison, D.K.: An investigation of production workers' performance variations and the potential impact of attitudes. Int. J. Adv. Manuf. Technol. **35**(11), 1113 (2006)
6. Hmida, J.B., Lee, J., Wang, X., Boukadi, F.: Production scheduling for continuous manufacturing systems with quality constraints. Product. Manuf. Res. **2**(1), 95–111 (2014)
7. Larsen, E., Lachapelle, S., Bengio, Y., Frejinger, E., Lacoste-Julien, S., Lodi, A.: Predicting tactical solutions to operational planning problems under imperfect information (2019). arXiv:1807.11876 [cs.LG], Technical report
8. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)

Evaluating Automated Storage and Retrieval System Policies with Simulation and Optimization

Michele Barbato, Alberto Ceselli, and Marco Premoli

Abstract In this paper we present a methodology to evaluate policies for automated storage and retrieval system (AS/RS) in warehouses. It is composed by four steps: (1) formal definition of the physical AS/RS and descriptive modeling on a simulation framework; (2) model validation and finding of potential bottlenecks by the statistical analysis of data logs; (3) definition of operational optimization policies to mitigate such bottlenecks; (4) evaluation of the policies using the simulation tool through key performance indicators (KPI). In particular, we take into consideration a unit-load AS/RS, we present a new simulation model combining discrete events and agent based paradigms. We consider an industrial test case, focusing on scheduling policies that exploit mathematical optimization, and we evaluate the effects of our approach on real world data. Experiments prove the effectiveness of our methodology.

Keywords AS/RS policies · Simulation · Matching · Mathematical optimization

1 Introduction

Automated storage and retrieval systems (AS/RSs) are widely used in warehouses and distributions centers all around the world since their introduction in the 1960s. Their main advantages are high space utilization, reduced labor costs, short retrieval times and improved inventory control. In order to get the best performance from an AS/RS, its design and operational planning should be wisely chosen.

A vast literature has accumulated over the years, both on general warehousing systems [3, 7, 8] and on diverse optimization problems arising in the design of AS/RS policies [2, 10]. While all these works carry out numerical studies, very few of them explicitly tackle the task of providing a full descriptive model of

M. Barbato · A. Ceselli · M. Premoli (✉)
Università Degli Studi di Milano, Department of Computer Science, Milano, Italy
e-mail: michele.barbato@unimi.it; alberto.ceselli@unimi.it; marco.premoli@unimi.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_12

such a complex system: in [4] several rules of thumb to design a simulation tool for warehouse are presented, such as key performance indicators, simulation initialization, object-oriented design patterns to implement the simulation tool; authors also present a case study using discrete-event paradigm; authors in [6] aim to present a discrete event simulation model that can be easily extended to most of the features found in AS/RS industrial settings; [9] is one of the few examples in which the agent-based paradigm is used; authors propose a multi-agent model for a mini-load system, claiming that autonomous vehicles control is easier in such paradigm.

Indeed, there is a large gap in the literature. On one side, optimization algorithms for AS/RS assume simplified models and hypothetical critical sub-systems, thereby providing solutions whose practical implementation potential still needs to be validated. On the other side, simulation approaches propose more realistic and reliable descriptive models, disregarding however the algorithmic resolution of key AS/RS optimization problems. In this paper we aim at reducing this gap.

We present a four-steps methodology to identify and evaluate optimization policies for AS/RS: (1) formal definition of a physical AS/RS and modeling on a simulation framework (Sect. 2.1), (2) model validation and finding of potential bottlenecks by statistical analysis of data logs (Sect. 2.2), (3) definition of operational optimization policies to mitigate such bottlenecks (Sect. 2.3), (4) policies evaluation using the simulation tool through *key performance indicators* (KPI) (Sect. 2.4).

In particular, we present a new mixed multi-agent and discrete-event simulation model for a unit-load AS/RS that, besides being very common in practice, matches the technology used by our industrial partners [1], and we propose a scheduling policy to tackle the found bottleneck exploiting mathematical optimization, which in turn extends a shift-based scheduling policy found in literature.

Finally, in Sect. 3 we draw our conclusions.

2 Design and Evaluation of AS/RS Optimization Policies

In the following we detail the four steps of our methodology.

2.1 Step (1): AS/RS Descriptive Model

The AS/RS under study has one aisle with racks on both sides. Each rack has double depth homogeneous cells with fifty-three lanes and nine levels. Stock-keeping-units (SKU) are represented by uniform size bins, with the same size of a rack slot. SKUs are moved with a double-shuttle aisle-captive crane, with separate drives for horizontal and vertical movements. The AS/RS has two I/O points, one in the front-end and one central, both with FIFO conveyor buffers to access the crane. The central I/O is connected to working stations where operators acts on SKUs. To move

the SKU from (resp. to) the central-bottom I/O to (resp. from) the working stations an automated shuttle of capacity two is used, which moves on an horizontal rail.

Working stations are used to compose customers orders. An order contains multiple items and is collected in a dedicated SKU (*order bin*). After the arrival of a SKU containing an item (*item bin*) in a working station, the item has to be processed before being assigned to an order bin; hence, the item bin has to wait on a working station before it to be stored again in the warehouse.

Daily orders have no release date and have to be completed within the working day without priorities. Items can be inserted in an order bin in any sequence. Multiple orders can be processed simultaneously (i.e. multiple order bins can be present on working stations) by multiple operators. Operators do not have to coordinate with each other. The complete list of orders to build is known at the beginning of the working day. The quantity of items needed to build all orders is available at the beginning of the working day.

AS/RS Descriptive Model To build a model of the AS/RS process described previously we characterize it by means of entities, events and activities. We can identify six types of entities:

- operators: resident entities, characterized by a cycle of activities that act on the SKUs; an operator O requests an empty order bin and a sequence of item bins, one at a time, to be retrieved on working stations; each time the production process on an item bin is over, O requests its storage back to the rack; when the order is complete, O requests the storage of the order bin and the cycle starts over;
- the system controller, or *orchestrator*: it is the second resident entity, characterized by a cycle of activities that act on the SKUs, on the crane and on the shuttle; the orchestrator listens to operators requests and translates them into actions to be executed by the last two entities, which in turns moves SKUs in the AS/RS;
- three resource entities, in charge of the execution of the orders given by the orchestrator to move SKUs: crane, shuttle and the conveyors. These resources need to synchronize in order to pass SKUs to each other: for example if one of the conveyor is full, the crane can not deliver an SKU on the conveyor until another SKU exits on the other side of the conveyor, and it has to stop and wait.
- SKUs: can be identified as the transient entities, characterized by the type, either item bin or order bin, and by activities like (1) storage and retrieval from the front-end I/O, (2) storage and retrieval from the working stations, (3) handling inside the warehouse and inside the working stations; each activities begin with an event generated either by an operator or by the system controller.

In Fig. 1 we present the schema of the relationships among model entities: an operator interacts with the orchestrator sending requests and receives notifications by the orchestrator and by the requested SKU; the orchestrator receives operators requests and translate them into orders for the crane and the shuttle; these latter move SKUs with pick and drop synchronizations; the synchronization among resources (crane, shuttle and conveyor) is done through the moved SKU, which acts

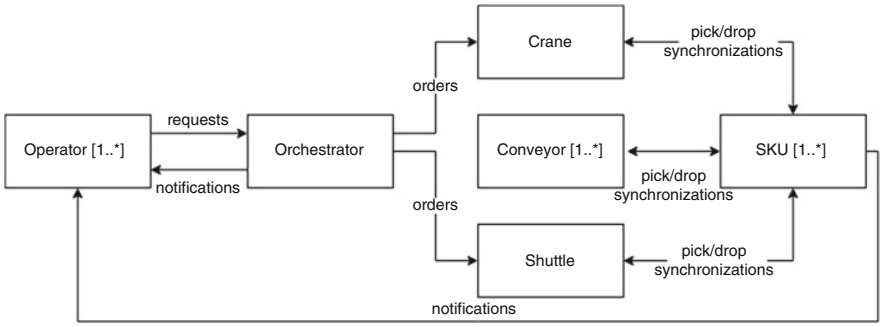


Fig. 1 Simulation model entities relationship

as a mediator; at last, an SKU notifies the operator when it arrives at the working station.

We can conveniently model each entities as an agent in our model and every relationship with a message passing interface.

At the same time, a discrete events approach can be used to model order composition of single operators, which generate requests of SKUs to the orchestrator: (1) the operator starts an order composition requesting to the orchestrator an empty order bin and the first item bin to process; (2) it awaits until both SKUs arrive; (3) when both SKUs arrive, it starts the working process on the item bin, which can require a non-deterministic amount of time; (4) at the end of the working process, it requires the storage of the item bin to the orchestrator; (5) if the order is not finished, it requires the retrieval of the next item bin to the orchestrator, otherwise (6) if a new order is available, the process goes back to step (1), otherwise it ends.

A critical analysis of our modeling framework allows to identify both strengths and weaknesses of our approach.

Strengths The synchronization among resource entities is simplified; resource entities act autonomously and their coordination is modeled via a message passing interface that might be asynchronous.

Resource agents statechart can be generalized and it abstracts from the features of the system; it includes few general states: (1) idle state, where the resource waits new orders from orchestrator; when new orders arrive, a loop of order execution starts and (2) the resource moves to the order execution location; when it arrives (3) it sends the order (either pick or drop) to the SKU and awaits for its answer, which notifies the end of the order; if any order is left, the loop continues from (2), otherwise the resource goes back to idle state (1).

A resource has no knowledge on the status and features of the external system; on the contrary, the orchestrator has a centralized view of the complete system, and it considers the current status of this latter to generate the sequence of orders for the resources. A resource stores parameters characterizing its current internal state, changed by the single order. For example the crane stores its current location (aisle, lane, level) and the SKUs currently on board.

Weaknesses The modeling complexity is shifted towards: (1) the statechart of the SKU entity, which acts as a mediator among resource entities and has to consider all combinations of orders from the resources and its current position, in order to enable customization of execution time of pick and drop actions for each combination; (2) the orchestrator agent, which listens for operators request and has to implement an on-line scheduling algorithm to generate the sequence of SKU movements. This algorithm has to consider the complete features and statuses of the system, which is not visible to the single components.

However, we stress that the complexity of the orchestrator agent, that looks like a weakness at a descriptive modeling level, turns out to offer a large degree of flexibility as soon as optimization models and policies are implemented to define its logic. It therefore fits perfectly in our setting.

2.2 Step (2): Model Validation and Bottleneck Analysis

Thanks to an industrial partner we had access to a dataset with the mechanical movements of SKUs in an AS/RS with the characteristics described in Sect. 2.1.

In particular we had access to all timestamps of arrival of an entity in a position in the warehouse, such as the arrival of the crane at a cell, and the end of pick and drop of an SKU from a resource (crane, shuttle and conveyor), covering 188 days.

Unfortunately, we had no access neither to the time, and purpose, for which operators sent their request to the system, nor to the orders composition.

Travel Time Model and Validation First, we make use of the available data logs to validate the implementation of the travel time models within our simulation tool.

Let (c, l) represent the position of a rack cell, where c is the horizontal lane and l is the vertical level. We model crane movements from location (c_1, l_1) to (c_2, l_2), with the following deterministic function: $\max(|c_1 - c_2| \cdot v_x^c, |l_1 - l_2| \cdot v_y^c) + f^c$ where we sum a constant value f^c, with which we model the picking/dropping time and the time to reach a constant speed, to the traveling time modelled with Chebyshev distance metric where v_x^c is the constant horizontal speed (seconds per lane) of the rack and v_y^c is the constant vertical speed (seconds per level). It would be easy to model a travel time that depends also on the load of the crane, which can be computed by the crane agent starting from the currently loaded SKUs; however the data log analysis showed that the speed of the considered crane is not affected by the load.

We model shuttle movements from working station w_1 to working station w_2 with the following linear function: $|w_1 - w_2| \cdot v_x^s + f^s$, where we sum a constant value f^s, which models the picking/dropping time and the time to reach a constant speed, to the travelling time modeled as a uniform linear motion with a constant horizontal travel speed v_x^s (seconds per working station). Finally, we model conveyors travel time with the a constant value $f^r + t^r$ where f^r is the constant

time of picking/dropping on the conveyor and t^r is the constant time to travel all the conveyor length. Values of all parameters were estimated analyzing the real world dataset.

We implemented our simulation tool with Anylogic, a Java based simulation software supporting discrete-event, system dynamic and agent-based modelling, and we validated our simulation model using five complete days of our dataset, comparing the time τ_r measured by our industrial partner on the real system with the time τ_s arising from simulations of single actions for both the crane and the shuttle, for a total of around 2500 actions for both.

Differences of crane movements $\Delta^\tau = \tau_r - \tau_s$ could be approximated by a normal distribution with mean $\mu \simeq 0.339$ s and standard deviation $\sigma \simeq 3.688$, where the mean time for a single action is around 45 s. For shuttle movements, differences Δ^τ have a median value of -0.73 s, a first quartile of -1.135 s and a third quartile of -0.36 s, having a mean time for the single action around 30 s. We also measured the overall time spent by crane and shuttle in mechanical operations, finding the average gap between real data and simulated one to be about 1.79%. That is, simulated data match well data of the real world system.

Identification of Bottlenecks Intuitively, we consider KPIs and event categories, and we evaluate if the disaggregation of events in categories makes KPI patterns to arise. Following this approach, we found a decomposition of storage and retrieval operations arising from the two different I/O points to be suitable for our approach. In Fig. 2 we present the average hourly usage of each I/O points in the dataset. It is clear that central I/O points, towards working stations, have higher frequency of usage with respect to front-end I/O points, which in turn occur throughout the day.

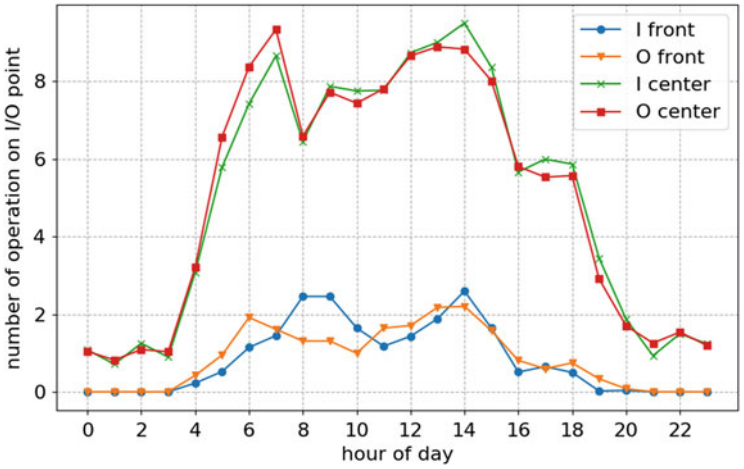


Fig. 2 Average hourly usage of AS/RS I/O points

Moreover, we know from our industrial partner that: (1) the front-end I/O operations can be executed without overlapping with central I/O operations, at the beginning and at the end of the working day; (2) all front-end I/O operations are known at the beginning of the day and, within them, storage operations can be executed independently from retrieval operations.

From the union of the insights coming from the analysis of the data and the industrial knowledge a new scheduling policy emerges to improve SKUs movements, that is the disaggregated optimization of front-end I/O operations and central I/O operations. Such problem has been identified in [5] as the *shift-based sequencing* problem for the AS/RS with twin-shuttle crane and single depth rack. Given a set of SKUs to retrieve (or to store), authors: (1) model the set as a weighted undirected graph, whose edges weights are given by a distance metric obeying the triangle inequality and represent the time to perform an action on pairs of SKUs; (2) the sequence of actions to perform is given by the solution of a minimum-cost perfect matching problem over the built graph, which can be solved at optimality with a polynomial time algorithm.

2.3 Step (3): Optimizing Bottleneck

In cases where the distance metric does not obey the triangle inequality, the modeling proposed in [5] does not guarantee optimality; in particular it always forms pairs of SKUs on which to perform an action, while with generic distance metric moving a single SKU might be less expensive. We propose the following variant: let C^* be the set of all SKUs that have to be retrieved; let $c(C)$ be the time required to retrieve SKUs C, i.e. the sum of the time for the crane to go from the I/O point to the cell containing C, to pick C on the crane and to return to the I/O point; let $c(\{C_1, C_2\})$ be the time required to retrieve the pair of SKUs C_1 and C_2, i.e. the sum of time for the crane to go from the I/O point to the cell containing C_1, to go from C_1 to C_2, to return to the I/O point and to pick both C_1 and C_2 on the crane.

Let us consider G and H two complete graphs on $|C^*|$ vertices. Each vertex of G corresponds to one SKU in C^*. We assign weight $c(\{C_1, C_2\})$ to each edge $\{C_1, C_2\}$ of G. We assign weight 0 to each edge of H. We then link each vertex of G with its counterpart vertex in H. If $C \in V(G)$ and $C' \in V(H)$ are linked we assign weight $c(C)$ to edge $\{C, C'\}$. Let T denote the weighted graph obtained so far. By using the edges $\{C, C'\}$ with $C \in V(G)$ and $C' \in V(H)$, we see that T always has a perfect matching and that a minimum weight perfect matching of T gives an optimal retrieval plan for C^*. An example of a weighted graph T is presented in Fig. 3.

Retrieval and Storage In the case of SKU retrieval in a double depth rack, we need to consider that several SKUs might be not directly accessible by the double-shuttle crane, i.e. they lie in the back slot while the front slot is storing another SKU. The retrieval for such SKU is possible only when both shuttles of the crane are free. To

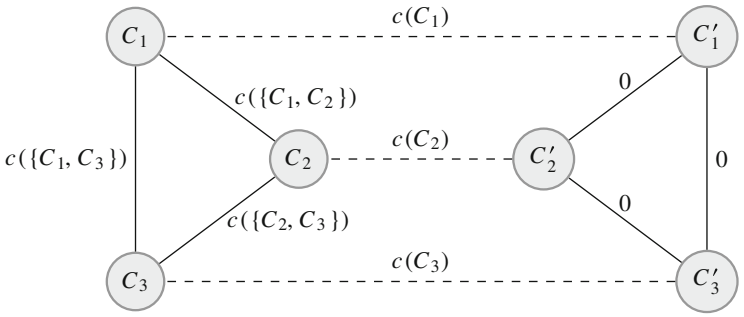


Fig. 3 Example of graph T with $C^* = \{C_1, C_2, C_3\}$

consider this scenario, let $B \subseteq C^*$ be the set of such not directly accessible SKUs, and $A = C^* \setminus B$ be the set of SKUs directly accessible by a single shuttle crane. We set to $+\infty$ the cost $c(\{C_1, C_2\})$ of all edges $\{C_1, C_2\} \in T$ connecting a pair of vertices $C_1, C_2 \in B$, while edges connecting all other pairs of vertices remain unchanged. Operatively, to retrieve a pair of SKU $C_1 \in B$ and $C_2 \in A$, C_1 will be retrieved first, when both shuttles of the crane are free.

As for the storage,we define this simple storage policy for the operations under optimization: incoming SKUs are stored only in the front slot of each cell in the rack.

Optimization Policy Our simple shift-based scheduling policy for a double-depth rack AS/RS with twin-shuttle crane is defined as follows:

- if an SKU has to be retrieved and it is not used in a working station throughout the day, it is retrieved at the beginning of the day, otherwise at the end of the day;
- if an SKU has to be stored and it is not used in a working station throughout the day, it is stored at the end of the day, otherwise at the beginning of the day;
- the position where to store incoming SKUs at the beginning of the day is chosen among front slot cells that are not used throughout all day. Among those cells, the nearest to the central I/O point are chosen;
- the position where to store incoming SKUs at the end of the day is chosen among all free front slot cells;
- the decomposed retrieval and storage operations, both for the morning and the evening, are optimized with the min-cost perfect matching algorithm.

The original order of center I/O operations does not have to be changed, and we can compare original and optimized operations.

2.4 Step (4): Experimental Evaluation

We propose the following experimental setup to evaluate our scheduling policy using our simulation tool: given a complete day of operations from our dataset, we apply our shift-based scheduling policy. We compare the original and the modified sequence of operations, feeding them to our simulation tool in a trace-driven style. In this scenario, we do not use the orchestrator and the operators agents, but we directly order crane and shuttle to move SKUs in a given sequence. These resources will execute one order at a time, synchronizing independently with each other; once they finish an order, they will ask for the next one until no more orders are available.

Key Performance Indicator As KPI, an interesting measure would be the total waiting time of an SKU by an operator, and the total amount of time to complete all orders. However, given that we can not recover operators information with the available data, we consider as KPI the total amount of time spent by the crane to move all SKUs, comparing the original actions with the modified ones.

Experimental Results In Table 1 we present the results of the execution of the shift based scheduling on ten different days of our dataset. We compare the time t (in seconds) spent by the crane in the original (subscript o) and modified (subscript m) sequence of actions, for the front-end I/O actions (superscript f), the central I/O actions (superscript c) and all set of actions (no superscript). In particular we present the percentage gain or loss of time given by the modified sequence ($(t_o - t_m)/t_o$): a positive value is a gain, a negative value a loss.

For every instance, I_o is the percentage of central I/O actions that in the original sequence are nested with front-end I/O actions. In average, I_o is around 1% on all instances, except instance 9 where its value is 3.81%. This latter also corresponds to a gain of 7% given by our scheduling policy, which in fact always grants $I_m = 0\%$.

In all instances the modified sequence provides a substantial improvement in execution time for the front-end I/O actions, in the range 10–30%. However, the central I/O actions have higher execution times with a worsening in the range 1–3%, due to the new positions of SKUs entering from the front-end I/O point, defined by our simple storage policy. Considering the complete sequence of actions, the modified sequence grants only a slight improvement in the range 0.5–3%, with a single substantial gain around 7% but also one worsening of 0.22%.

Although our policy leads to a substantial time gain on the optimized operations, the advantage on the total operations time is limited. The time loss on central I/O operations indicates that a better storage policy is needed for the shift-based scheduling, to make a cleverer use of the rack cells.

Finally, in last column of Table 1, we report relative gaps between the overall time spent by crane and shuttle in mechanical operations measured on the real data (ϕ_r), and that obtained by the modified sequence in our simulation (ϕ_m). The overall picture is similar to that concerning SKUs operation times.

Table 1 Comparison of total time spent by the crane (in seconds)

		Front-end I/O actions			Central I/O actions			All actions			Mechanical times
d.	I_o	t_o^f	t_m^f	$\frac{(t_o^f - t_m^f)}{t_o^f}$	t_o^c	t_m^c	$\frac{(t_o^c - t_m^c)}{t_o^c}$	t_o	t_m	$\frac{(t_o - t_m)}{t_o}$	$\frac{(\phi_r - \phi_m)}{\phi_r}$
1	1.51%	1671	1423	14.87%	16,657	16,590	0.40%	18,328	18,013	1.72%	7.88%
2	0.80%	2313	1942	16.04%	14,787	14,936	−1.01%	17,100	16,878	1.30%	3.54%
3	1.28%	2887	2200	23.78%	13,549	14,024	−3.50%	16,436	16,224	1.29%	1.58%
4	1.17%	1923	1555	19.16%	14,463	14,754	−2.01%	16,386	16,309	0.47%	7.57%
5	1.32%	2113	1715	18.80%	13,601	14,033	−3.18%	15,713	15,748	−0.22%	1.00%
6	0.40%	2867	2021	29.51%	14,134	14,516	−2.70%	17,001	16,537	2.73%	1.63%
7	0.99%	3119	2309	25.96%	11,674	11,909	−2.01%	14,793	14,218	3.89%	2.80%
8	1.44%	4326	3709	14.26%	12,009	12,307	−2.48%	16,335	16,016	1.95%	0.24%
9	3.81%	7363	5696	22.64%	12,689	12,921	−1.82%	20,052	18,616	7.16%	9.57%
10	0.33%	2019	1788	11.47%	18,014	17,927	0.49%	20,033	19,714	1.59%	0.30%

3 Conclusions

We proposed a generic approach to the identification of critical AS/RS sub-components, and the design of corresponding optimization policies. It makes use of both descriptive and optimization models.

Our framework turned out to be more flexible than those proposed in the literature; one of its key features is to shift complexity in the logic component of an orchestrator, that in turn can be effectively encoded by optimization models.

The building of a descriptive model turns out to be an asset of our methodology, allowing to (indirectly, but reliably) evaluate the effect of optimization on the real world system. In our case, the design and application of a custom optimization policy allows to reduce operational times of certain system functions at the expense of others. We indeed believe this to be a common issue when optimization policies are applied in practice. The potential of combining optimization and simulation actually unfolds at this stage: the decision maker can evaluate the impact of different optimization strategies, choosing the one yielding a suitable balancing for his/her system.

Acknowledgement Partially funded by Regione Lombardia, grant agreement n. E97F17000000 009, Project AD-COM.

References

1. AD-COM: (2019). https://www.ad-com.net. Accessed April 2019
2. Boysen, N., Stephan, K.: A survey on single crane scheduling in automated storage/retrieval systems. Eur. J. Oper. Res. **254**(3), 691–704 (2016)
3. Boysen, N., de Koster, R., Weidinger, F.: Warehousing in the e-commerce era: a survey. Eur. J. Oper. Res. **277**(2), 396–411 (2018)
4. Colla, V., Nastasi, G.: Modelling and simulation of an automated warehouse for the comparison of storage strategies. In: Modelling, Simulation and Optimization, pp. 471–486 (2010). IntechOpen
5. Dooly, D.R., Lee, H.F.: A shift-based sequencing method for twin-shuttle automated storage and retrieval systems. IIE Trans. **40**(6), 586–594 (2008)
6. Gagliardi, J.-P., Renaud, J., Ruiz, A.: A simulation modeling framework for multiple-aisle automated storage and retrieval systems. J. Intell. Manuf. **25**(1), 193–207 (2014)
7. Gu, J., Goetschalckx, M., McGinnis, L.F.: Research on warehouse operation: a comprehensive review. Eur. J. Oper. Res. **177**(1), 1–21 (2007)
8. Gu, J., Goetschalckx, M., McGinnis, L.F.: Research on warehouse design and performance evaluation: a comprehensive review. Eur. J. Oper. Res. **203**(3), 539–549 (2010)
9. Güller, M., Hegmanns, T.: Simulation-based performance analysis of a miniload multishuttle order picking system. Proc. CIRP **17**, 475–480 (2014)
10. Wauters, T., Villa, F., Christiaens, J., Alvarez-Valdes, R., Berghe, G.V.: A decomposition approach to dual shuttle automated storage and retrieval systems. Comput. Ind. Eng. **101**, 325–337 (2016)

Rolling-Horizon Heuristics for Capacitated Stochastic Inventory Problems with Forecast Updates

Emanuele Tresoldi and Alberto Ceselli

Abstract In this paper we propose a practical optimization approach based on the rolling-horizon paradigm to address general single-product periodic-review inventory control problems. Our framework supports many constraints and requirements that are found in real inventory problems and does not rely on any assumption on the statistical distribution of random variables. Ambiguous demand and costs, forecast updates, constant lead time, lost sales, flexible inventory capacity and product availability can all be taken into account. Three, increasingly sophisticated, solution methods are proposed and implemented within our optimization framework: a myopic policy, a linear programming model with risk penalization and a scenario-based stochastic programming model. The effectiveness of our approach is proved using a dataset of realistic instances.

Keywords Inventory; Stochastic programming; Finite capacity

1 Introduction

Inventory management is one of the main pillars of production planning. Indeed, there is currently a significant push for the introduction of innovative systems and advanced decision support tools in such a context. As an example, several European countries are sponsoring programs in the Industry 4.0 stream [1].

Inventory control systems are studied since decades in the Operations Research literature [17]. Several features arise in applications, the most common being the need to handle uncertainty in data.

At their core, these methods assume the following structure of the system: a single product need to be ordered by a company in a certain time period. The unit cost of that product might change over time. Then, such an order is either employed

E. Tresoldi (✉) · A. Ceselli
Dipartimento di Informatica, Università degli Studi di Milano, Milan, Italy
e-mail: emanuele.tresoldi@unimi.it; alberto.ceselli@unimi.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_13

to satisfy demand, as soon as the product is dispatched or kept in stock at a cost to satisfy demand in subsequent time periods. Demand is changing over time as well: neither costs nor demands can be assumed to be deterministically known in advance.

Therefore, classical approaches consist in optimization policies providing probabilistic guarantees. For instance, the (s, S) policy [6] and the (r, Q) policy [5] apply a fundamental idea of computing suitable reorder points and reorder quantities, and then statically perform reorders whenever the stock falls below the reorder threshold. Despite their simplicity, under assumptions about the distribution of demands and costs they can proved to be optimal. The complexity relies in the evaluation of these parameters, requiring complex computations, and eventually restricting their practical implementation in industrial inventory systems.

Alternative policies have also been devised, in order to obtain tractable methods preserving quality guarantees. For instance, balancing policies compute trade-offs between order and stock costs, dynamically deciding reorder quantities [13]. Balancing policies are often able to yield 2-approximation guarantees. It is the case, for instance, of additional constraints stock capacity [15], lost sales [14] and lead time [12]. A drawback of such an approach, however, is that strong assumptions on the structure of costs and demands need to be performed to keep both quality guarantees and computational efficiency. For instance, the fundamental setting of [15] assumes no speculative motivation for holding inventory or having backorders in the system, which is instead the main motivation for inventory management in many companies. In the same way, not all the features of real systems can fit the framework required by the approximation analysis. The difficult practical applicability of these techniques has motivated more recent heuristic approaches [20], either as generic optimization algorithms [16, 18], or for computing good thresholds in policies like (r, Q) [2] even at the cost of losing quality guarantees.

Summarizing, even if the literature gives a deep theoretical understanding about stochastic inventory problems, methods proving their practical applicability are still very limited. Therefore, in this paper we propose a practical optimization framework based on the rolling-horizon paradigm to address general single-product periodic-review inventory control problems. In our framework several real-world features like ambiguous demand and costs, constant lead time, lost sales, hard and flexible inventory capacity and product availability are taken into account. Our framework gracefully exploits forecast updates [19], and a-priory knowledge about data distribution, but do not rely on them.

In Sect. 2 we formalize our inventory model, and in Sect. 3 we introduce the algorithmic framework. In Sect. 4 we explain our test methodology, report experimental results on realistic instances and draw a few conclusions.

2 Modeling

In this paper, we consider a single-product periodic-review inventory system with stochastic demand and ordering cost, forecast updates and constant lead time. We take into account a finite planning horizon $T : \{0, \ldots, n\}$, divided in $n + 1$ time

periods where 0 represents the period before the beginning of the planning horizon. In any period t decisions must be taken. Future demands and costs are stochastic quantities. Instead, the maximum availability of product (a_t), a forecast of the demand ($d^t_{t'}$) and ordering cost ($c^t_{t'}$) for all subsequent time periods $t' \in \{t, \ldots, n\}$ are given. The system is characterized by a fixed lead time L and flexible storage capacity defined by soft upper and lower limitation ($\bar{H}$ and $\underline{H}$) that can be exceeded paying penalties. At the beginning of each time period one ordering decision is made base on the status of the inventory and demand and costs forecast. Considering the lead time, products ordered in time period t are delivered in time period $t + L$. At the end of each time period the demand is realized and a new inventory status is obtained. Demand that is not satisfied due to inventory limitation is lost. Several costs have to be taken into account in defining an ordering plan: forecast of ordering cost $c^t_{t'}$ per unit of product in time period t', holding cost h to store one unit of product for a single time period, penalties $\bar{h}$ and $\underline{h}$ per unit of product exceeding the upper and lower storage limitations for a single time period and forecast cost for lost sales $p^t_{t'}$ per unit of demand not satisfied in time period t', i.e. loss of potential earning due to the lack of product to sell in the inventory. The goal of the problem is to define an ordering plan that minimizes the total operational cost over the complete time period.

Deterministic Linear Programming (LP) Model This formulation will serve as both building-block for our heuristic algorithms and as benchmark for the performance evaluation. Given D_t, C_t and P_t as the actual realization of the stochastic parameters associated with the demand, ordering and lost sales costs in time period t, the problem is defined using several, non-negative, continuous variables. In details, for each time period $t \in T$, variable x_t represents the amount of product ordered, z_t accounts for the demand not satisfied, variable y_t identifies the amount of product stored at the end of period t and variables $\bar{y}_t$ and $\underline{y}_t$ correspond to the amount of stored product above $\bar{H}$ and below $\underline{H}$ at the end of time period t.

$$\min \sum_{i \in T} C_t x_t + h y_t + \bar{m}\bar{y}_t + \underline{m}\underline{y}_t + P_t z_t \tag{1}$$

$$s.t. \quad x_{t-L} + y_{t-1} = D_t - z_t + y_t \qquad \forall t \in T : t - L > 0 \tag{2}$$

$$\underline{H} - \underline{y}_t \leq y_t \leq \bar{H} + \bar{y}_t \qquad \forall t \in T : t > 0 \tag{3}$$

$$x_t \leq a_t \qquad \forall t \in T : t > 0 \tag{4}$$

The objective function (1) aims at the minimization of the total costs given by the sum of purchasing costs ($C_t x_t$), holding costs ($h y_t$), penalty costs to exceed storage capacity ($\bar{h}_t \bar{y}_t$ and $\underline{h}_t \underline{y}_t$) and lost sales ($P_t z_t$). Equations (2) represent inventory constraints while inequalities (3) allow for a flexible storage capacity and define the value of variables $\bar{y}_t$ and $\underline{y}_t$. Finally, constraints (4) bound the maximum amount of product that can be bought in each time period to a_t.

3 Algorithms

We present a general framework for the solution of inventory problems based on the rolling-horizon paradigm [4]. The main idea is to solve the problem decomposing it in a sequence of optimize decisions one for each time period. Each decision is taken solving a small sub-problem considering a limited look-ahead windows composed by the $W > L$ following time periods. In details, at the beginning of each period t, taking into account forecast information available for time periods in $\{t, \ldots, \max(t + W, n)\}$ we only decide the amount of product to order in the current period: x_t. At the end of the time period the demand is realized, the costs are computed and the inventory status is updated considering the actual values for the demand and the ordering costs. Then we move to time period $t + 1$ and we repeat the process.

In Algorithm 1 the pseudo-code of our rolling-horizon procedure is reported. Lines from 1 to 4 provide the initialization for the solution vector x^*, the look-ahead window T', the vector q of the amount of product ordered in previous time periods and delivered within T' and the status of the inventory y_0 at the beginning of T'. In line 5 forecast information for demand (vector d), ordering cost (vector c) and cost for lost sales (vector p) are generated using any forecasting model (see Sect. 4 for the description of the method used in our tests). For each time period $t \in T$ we solve, using the $SolveInventory$ function, a small inventory sub-problem defined over time horizon T' taking into consideration the initial status of the inventory y_0, previous orders q, availability of product a in T' and all forecast information available at time t: d^t, c^t and p^t (line 7). Only the decision taken in the first time period in T' is returned and it is used to update the past orders vector q, the inventory status y_0 and the solution vector x^* then the look-ahead window T' is moved forward in order to solve the problem defined on the next time period t+1 (lines from 1 to 4). The procedure ends once the last sub-problem has been solved and the complete ordering plan x^* is defined. The function $SolveInventory$ is kept generic: details of three different solution policies are provided in the next paragraphs.

Myopic Policy The first solution policy is the myopic one. In each time period t, considered the availability of product a_t, we order an amount of product (x_t) that is as close as possible to the forecast demand for period $t + L$. The status of the inventory at the end of the time period t, the exact value of x_t and all other associated quantities can be directly computed as $x_t = \min(d^t_{t+L}, a_t)$, $y_t = \max(D_t - q_t + y_{t-1}, 0)$ $\bar{y}_t = \max(y_t - \bar{H}, 0)$, $\underline{y}_t = \max(\underline{H} - y_t, 0)$, $z_t = |\min(D_t - q_t + y_{t-1}, 0)|$. Where q_t is the amount of product delivered at time period t, i.e. $q_t = x_{t-L}$; if $L = 0$ then $q_t = x_t$. The total actual cost for period t is then equal to $C_t x_t + h y_t + \bar{m}\bar{y}_t + \underline{m}\underline{y}_t + P_t z_t$. It is worth noting that the size of the look-ahead window W does not have any impact on the myopic decisions since they do not consider the effect of the order placed in t on subsequent time periods. For this reason, in the Myopic policy $t' = t$.

Algorithm 1: Rolling-horizon heuristic procedure

Input: W look-ahead window size, T set of time periods, L lead time, D demand vectors, a availability vector, $C, h, \bar{h}, \underline{h}, p$ cost vectors.
Output: x^* an optimized ordering plan.

$x_t^* \leftarrow 0 \quad \forall t \in \{1 \dots |T|\}$;
$T' \leftarrow \{0 \dots W\}$;
$q_t \leftarrow 0 \quad \forall t \in \{1 \dots L\}$;
$y_0 \leftarrow \underline{H}$;
$d, c, p \leftarrow Forecast(D, C)$;
forall the $t \in T$ **do**
  $x_1 \leftarrow SolveInventory(T', q, y_0, a, d^t, c^t, p^t)$;
  for $i \in \{1 \dots L-1\}$ **do**
    $q_i \leftarrow q_{i+1}$;
  $q_L \leftarrow x_1$;
  $y_0 \leftarrow \max(D_1 - q_1 + y_0, 0)$;
  $x_t^* \leftarrow x_1$;
  $T' \leftarrow \{0 \dots \min(W, |T| - t)\}$;
return x^*

Risk Penalization Policy The second solution policy is based on a LP formulation similar to the deterministic model presented in Sect. 2. This problem is defined on a limited planning horizon T', whose size depends on W. Instead of deterministic demand and costs (D_t, C_t and P_t) forecasts $d_{t'}^t$, $c_{t'}^t$ and $p_{t'}^t$ are used. In order to take into account the progressive degradation of the forecast quality we introduced an additional term in the objective function: $\alpha^{t'}\beta x_{t'}$ (5). This term includes two user-defined penalization parameters. The first $\alpha^{t'}$ increases exponentially with the distance between the first time period of T' and the period t': orders are less likely to be placed in the last part of the look-ahead window when demand and cost forecasts are less reliable. In Sect. 4 we evaluate the effect of this setting. The second, β, is simply a scaling factor: after preliminary tests, we found $\frac{\sigma(\hat{d}^t)}{\max_t d^t} + \frac{\sigma(\hat{c}^t)}{\max_t c^t}$ to be a suitable value, where σ is the sample standard deviation computed on a large set of demand and cost forecast vectors ($\hat{d}^t$ and $\hat{c}^t$). Moreover, inventory constraints (6) have to be modified to include parameters q_t' representing orders placed before T'.

$$\min \sum_{t' \in T'} c_{t'}^t x_{t'} + h y_{t'} + \bar{m}\bar{y}_{t'} + \underline{m}\underline{y}_{t'} + p_{t'}^t z_{t'} + \alpha^{t'}\beta x_{t'} \tag{5}$$

$$s.t. y_{t'-1} - d_{t'}^t + z_{t'} - y_{t'} = \begin{cases} q_{t'} & \text{if } t' \leq L \\ x_{t'-L} & \text{if } t' > L \end{cases} \quad \forall t' \in T', \forall s \in S : t' > 0 \tag{6}$$

and (3)–(4). When the optimal solution has been found we return the value of x_1.

Scenario-Based Policy The last policy consists of a scenario-based stochastic programming model, see [3] and [11] for a general introduction on stochastic programming models. Our formulation is based on the deterministic one presented in Sect. 2 with the addition of the set of scenarios S. Each scenario $s \in S$ defines parameters $d_{t'}^{ts}$, $c_{t'}^{ts}$ and $p_{t'}^{ts}$ representing a possible realization of demand and costs generated with the procedure described in Sect. 4. In our stochastic model all variables are scenario-dependent and in the objective function (7) we minimize the average cost over all scenarios. As in the previous policy we modified the inventory constraints (8) to account for past orders $q_{t'}$. Finally, in order to force the same amount of product to be ordered in the first period we set x_1^{ts} to be equal in all scenarios (9).

$$\min \sum_{t' \in T'} \sum_{s \in S} c_{t'}^{ts} x_{t'}^{s} + h y_{t'}^{s} + \bar{m} \bar{y}_{t'}^{s} + \underline{m} \underline{y}_{t'}^{s} + p_{t'}^{ts} z_{t'}^{s} \tag{7}$$

$$s.t. \quad y_{t'-1}^{s} - d_{t'}^{ts} + z_{t'}^{s} - y_{t'}^{s} = \begin{cases} q_{t'} & \text{if } t' \leq L \\ x_{t'-L}^{s} & \text{if } t' > L \end{cases} \qquad \forall t' \in T', \forall s \in S : t' > 0 \tag{8}$$

$$x_1^{s-1} = x_1^{s} \qquad \forall s \in S : s > 0 \tag{9}$$

$$\underline{H} - \underline{y}_{t'}^{s} \leq y_{t'}^{s} \leq \bar{H} - \bar{y}_{t'}^{s} \qquad \forall t' \in T', \forall s \in S : t' > 0 \tag{10}$$

$$x_{t'}^{s} \leq a_{t'} \qquad \forall t' \in T' : t' > 0 \tag{11}$$

Once the model is solved to optimality, we return the amount of product ordered in the first time period, i.e. the value of any variable x_1^{ts}.

4 Experimental Evaluation

In order to analyze the effectiveness and the efficiency of our approach we set-up a testing campaign based on realistic data coming from the Italian day-ahead gas market [7]. In particular, we simulate the behavior of a single virtual company that has to define ordering plans to deal with the entirety of the Italian market. The rolling-horizon algorithm has been implemented in Python using Pyomo as optimization modeling language [10] and Gurobi 8.1.0 as LP solver [9]. All tests have been conducted on a Windows 10 64bit machine equipped with processor Intel i7-4702hq and 16 GB of RAM.

Structure of Demand and Cost Predictions We do not rely on specific statistical properties of demand and cost forecasting; that is, the forecasting model is irrelevant for our models. However, in practice we expect their accuracy to decrease over time. In order to simulate this behavior, in our tests, we proceeded as follows. For each time period $t \in T$ let $E_t^d = \{e_t^d \dots e_n^d\}$ and $E_t^c = \{e_t^c \dots e_n^c\}$ be two sequences of random coefficients representing the forecast errors for each subsequent time period for demand and ordering costs. We start computing perturbed demands and costs D_t' and C_t' as:

$$D_t' = D_t + e_t^d D_t, \text{ and } C_t' = C_t + e_t^c C_t \quad \forall t \in T. \tag{12}$$

Then, taking into account other sequences of random coefficients $E'^d_t = \{e'^d_t \dots e'^d_n\}$ and $E'^c_t = \{e'^c_t \dots e'^c_n\}$, in each time period we consider as forecast error the sum of all errors in previous periods. More formally, we generate forecast for all periods $t' \in \{t, \dots, n\}$ according to the following rules:

$$d_{t'}^t = D_t' + \sum_{i \in \{t, \dots t'\}} e_i'^d D_t', \text{ and } c_{t'}^t = C_t' + \sum_{i \in \{t, \dots t'\}} e_i'^c C_t'. \tag{13}$$

Where $d_{t'}^t$ and $c_{t'}^t$ are the demand and cost forecasts for time period t' based on information available at time t. Forecast for lost sales cost are directly associated with ordering cost and are computed as $p_{t'}^t = \delta c_{t'}^t$ with $\delta > 1$.

Instance Data Our test set is made up of 96 instances. They are generated from real data for years 2017 and 2018 of the Italian day-ahead gas market (raw data can be retrieved at [8]) considering different setting for the lead time, parameter L, and for the penalization coefficients $\bar{h}$ and $\underline{h}$. In details, we create one instance for each period of 30 days ($|T| = 30$) considering two different lead time values 0 and 3 and two penalization settings $high$ and low. In the $high$ setting the penalization cost is higher than the maximum ordering cost ($\bar{h} = \underline{h} = 60$) making the capacity limitation very rigid. On the other hand, in the low setting, exceeding the storage capacity can be profitable since the penalization is lower than the minimum ordering cost ($\bar{h} = \underline{h} = 1$). In our instances forecast for demand and costs are obtained starting from real daily market demand (in MWh) and average daily cost (in millions of EUR). Forecast error coefficients E_t^d and E_t^c are randomly generated using a normal distribution with mean 0 and standard deviation 0.0625 to obtain an average forecast error of about 10%. The sets of demand and cost forecast vectors d^t and c^t used for the computation of $\sigma(\hat{d^t}) + \sigma(\hat{c^t})$ and for the scenario generation are composed by 1000 elements each. Parameter δ used to define lost sales costs forecast $p_{t'}^t$ is always equal to 5 so that $\bar{h} = \underline{h} \leq p_{t'}^t$. Product availability is virtually unlimited so we set $a_t = \infty \; \forall t \in T$. Since, in our application, the values of the demand and the daily average cost are known at the beginning of each day then we set forecasts for the current day equal to the actual values that is $d_1^t = D_t \forall t \in T$ and $c_1^t = C_t \forall t \in T$. Holding cost is fixed and always equal to 1. Values $\underline{H}$ and $\bar{H}$ are set to 25,000 and 80,000 MW roughly corresponding to the average daily demand and

75% of the maximum daily demand. We assume that the inventory level is equal to $\underline{H}$ at the beginning of the first time period. Finally, we set the value of W, the size of the look-ahead window, to 7. This allows us to properly show differences in the solutions computed with the three policies. Moreover, it represents a realistic look-ahead window size since, in the gas market, many companies rely on weekly forecast information.

Risk Penalization Fine Tuning The risk penalization policy presented in Sect. 3 makes use of a user defined coefficient α to penalize orders placed on future days. In order to find the best values for α we performed extensive testing solving all instances in our data-set with α ranging from 1 to 5 with 0.1 step. We omit detailed results with $L = 0$: when penalties are high the value of α has a very mild impact; when penalties are low it always pays off to keep α low. Instead, an overview of the results obtained with $L = 3$ on four significant cases, is reported in Fig. 1. The results show that the best setting for α is largely influenced by the value of the lead time. In fact, on the one hand when $L = 0$ the best value for α is 1 on the other hand 2.9 is the value for α that generates the best results with $L = 3$. This behavior can be explained taking into consideration the rolling-horizon context. There is not much harm in trusting the forecasts when the order and the delivery take place in the same day. Indeed, there is always time the next day to make up for the forecast

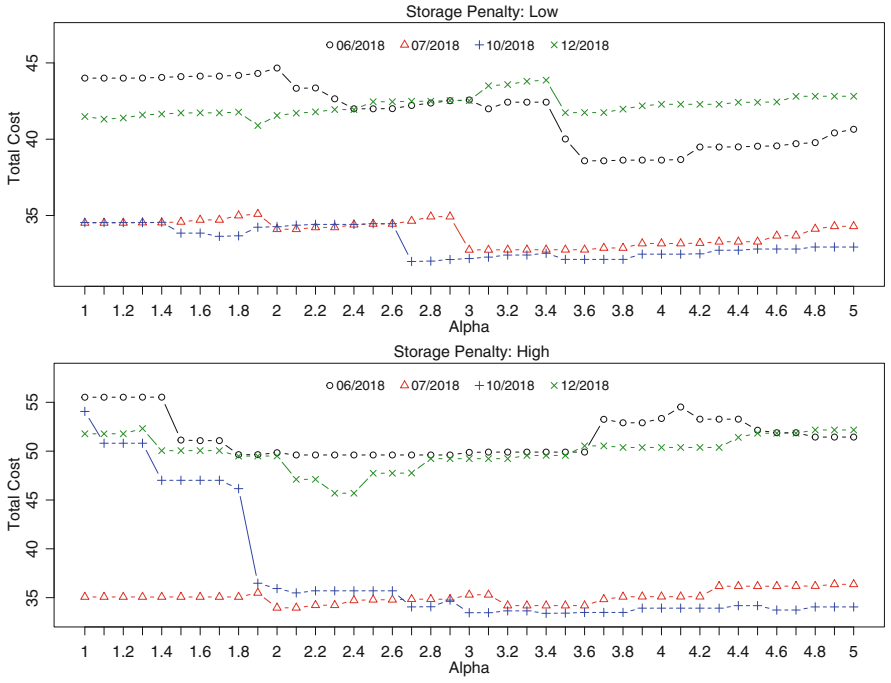


Fig. 1 Effect of alpha: year 2018, lead time 3

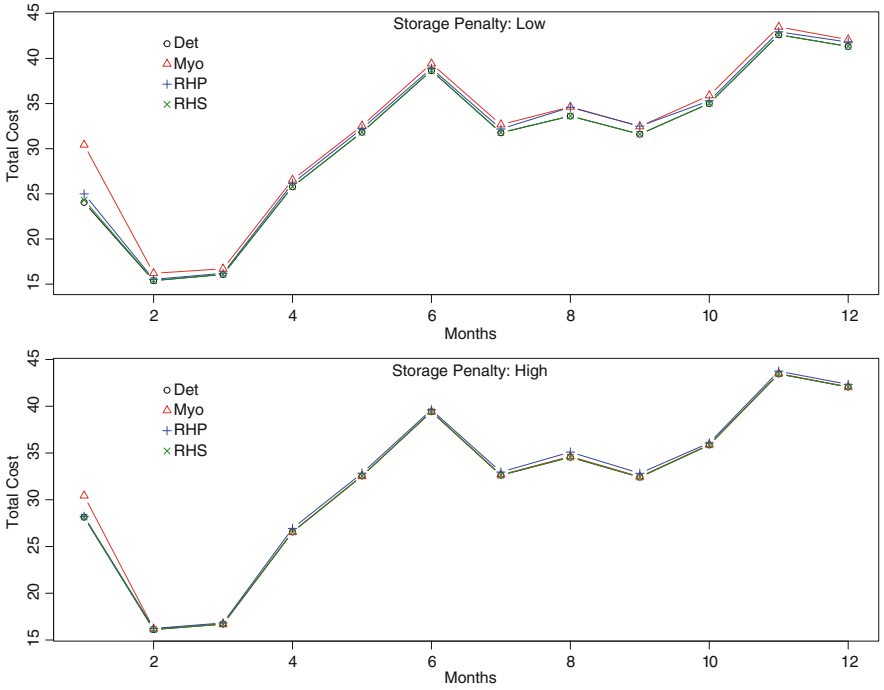


Fig. 2 Results: year 2018, lead time 0

error in the previous day, particularly in our application where demand and costs for the first day are always deterministic. This is not true when $L > 0$ since forecast are less and less reliable in future days and so it is better to place orders in the first part of the look ahead window.

Performance Analysis In evaluating the performance of all three solution policies for our rolling-horizon algorithm we use the deterministic model (Det, Sect. 2) as benchmark. A graphical representation of the results obtained on instances for year 2018, with different lead time and penalization settings, is depicted in Figs. 2 and 3. The three solution policies exhibit very different behaviors. Considering all instances, the scenario-based policy (RHS) obtains always the best results with a total average error, with respect to Det, equal to 1.01%, the risk penalization policy (RHP) achieves an average error of 8.02% while the myopic policy (Myo) averages at 17.97%. In details, when settings are very restrictive, i.e. $L = 0$ and the penalty is $high$, Myo behaves well (average error 0.82%) outperforming RHP that obtains an error equal to 1.10%. With all other settings, RHP is on average better than Myo, especially when $L = 3$. With this setting, RHP reduces on average by half the error of Myo (14.29% vs 30.90%). There are, however, a few instances where the total cost achieved with Myo is lower. In these cases, the optimal deterministic

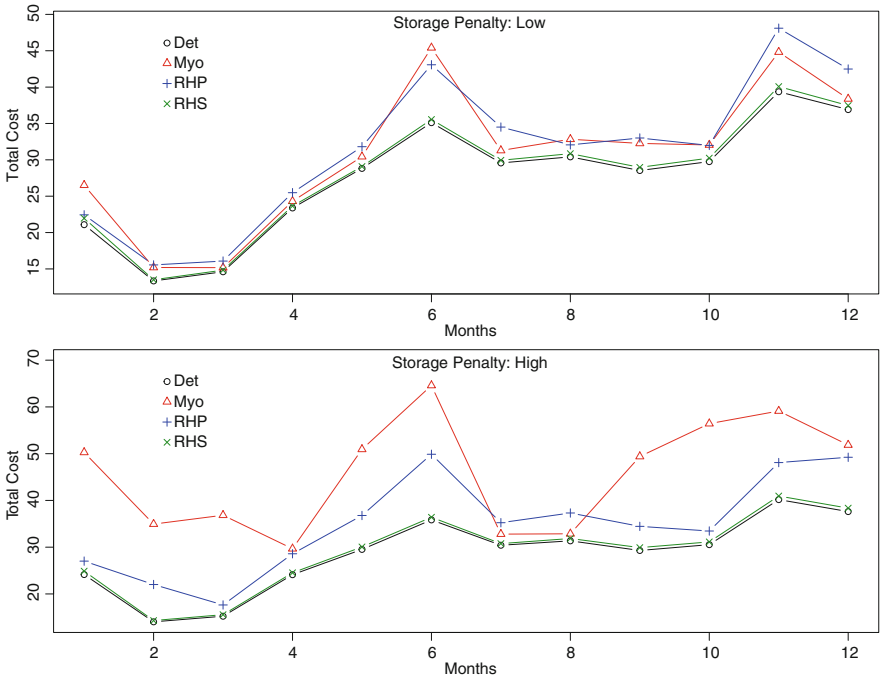


Fig. 3 Results: year 2018, lead time 3

solution shows a structure similar to the Myo solution and indeed the daily optimal order amount is very close to the demand for day $t + L$.

Moreover, considering the consistency of the results, the RHS policy shows a clear advantage. Indeed, the maximum error for RHS is equal to 4.18% while RHP and Myo reach 56.88% and 149.01% respectively.

As regard as the computational time, Myo runs in a fraction of a second, RHP solves an instance in less than 1.49 s while RHS takes about 105.33 s. Execution time is roughly the same for all instances. Therefore, there is a clear trade-off between quality of the solution and computational time. However, considering that a good quality solution can save millions of euro every day and that requires only about 100 s the RHS policy is the undeniable winner of this comparison.

Acknowledgement This work has been partially funded by Regione Lombardia, grant agreement n. E97F17000000009, Project AD-COM.

References

1. AD-COM: Advanced Cosmetic Manufacturing (2019). https://ad-com.net/?lang=en. Last visited: 01 April 2019
2. Al-Rifai, M.H., Rossetti, M.D.: An efficient heuristic optimization algorithm for a two-echelon (r, q) inventory system. Int. J. Prod. Econ. **109**(1–2), 195–213 (2007)
3. Birge, J.R., Louveaux, F.: Introduction to Stochastic Programming, 2nd edn. Springer, Berlin (2011)
4. Dalalah, D., Bataineh, O., Alkhaledi, K.A.: Platelets inventory management: a rolling horizon Sim–Opt approach for an age-differentiated demand. J. Simul. **13**(3), 209–225 (2019)
5. Federgruen, A., Zheng, Y.: Efficient algorithm for computing an optimal (r, q) policy in continuous review stochastic inventory systems. Oper. Res. **40**(4), 808–813 (1992)
6. Feng, Y., Xiao, B.: A new algorithm for computing optimal (s, s) policies in a stochastic single item/location inventory system. IIE Trans. **32**(11), 1081–1090 (2000)
7. G.M.E.: About the Gas Market. http://www.mercatoelettrico.org/En/Mercati/MGas/MGas.aspx. Last visited: 01 April 2019
8. G.M.E.: MGP-GAS Results and Statistics. http://www.mercatoelettrico.org/En/Esiti/MGP-GAS/EsitiGasMGP.aspx. Last visited: 01 April 2019
9. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2018). http://www.gurobi.com. Last visited: 01 April 2019
10. Hart, W.E., Laird, C.D., Watson, J.P., Woodruff, D.L., Hackebeil, G.A., Nicholson, B.L., Siirola, J.D.: Pyomo–Optimization Modeling in Python, vol. 67, 2nd edn. Springer, Berlin (2017)
11. King, A.J., Wallace, S.W.: Modeling with Stochastic Programming, 1st edn. Springer, New York (2012)
12. Levi, R., Shi, C.: Approximation algorithms for the stochastic lot-sizing problem with order lead times. Oper. Res. **61**(3), 593–602 (2013)
13. Levi, R., Pal, M., Roundy, R.O., Shmoys, D.B.: Approximation algorithms for stochastic inventory control models. Math. Oper. Res. **32**(2), 284–302 (2007)
14. Levi, R., Janakiraman, G., Nagarajan, M.: A 2-approximation algorithm for stochastic inventory control models with lost sales. Math. Oper. Res. **33**(2), 351–374 (2008)
15. Levi, R., Roundy, R.O., Shmoys, D.B., Van Truong, A.: Approximation algorithms for capacitated stochastic inventory control models. Oper. Res. **56**(5), 1184–1199 (2008)
16. Mishra, V.K.: Application of Genetic algorithms in inventory control. In: Optimal Inventory Control and Management Techniques, pp. 32–46. IGI Global, Pennsylvania (2016)
17. Porteus, E.L.: Chapter 12 stochastic inventory theory. In: Stochastic Models. Handbooks in Operations Research and Management Science, vol. 2, pp. 605–652. Elsevier, Amsterdam (1990)
18. Rahdar, M., Wang, L., Hu, G.: A tri-level optimization model for inventory control with uncertain demand and lead time. Int. J. Prod. Econ. **195**, 96–105 (2018)
19. Sethi, S.P., Yan, H., Zhang, H.: Inventory and Supply Chain Models with Forecast Updates, pp. 1–21. Springer, Boston (2005)
20. Taleizadeh, A.A., Cárdenas-Barrón, L.E.: Metaheuristic algorithms for supply chain management problems. In: Meta-Heuristics Optimization Algorithms in Engineering, Business, Economics, and Finance, pp. 110–135. IGI Global Pennsylvania (2012)

Paths and Matchings in an Automated Warehouse

Michele Barbato, Alberto Ceselli, and Giovanni Righini

Abstract We analyze a number of variations of a combinatorial optimization problem arising from the optimization of an automated warehouse. We classify these variations according to four relevant parameters and we analyze which combinations are polynomially solvable, owing to dynamic programming recursions or to reductions to known graph optimization problems such as the shortest path problem and the minimum cost perfect matching problem.

Keywords Scheduling · Combinatorial optimization · Dynamic programming

1 The Automated Warehouse

This study arises from an applied research project on smart manufacturing, involving some companies in the cosmetic industry. In particular, we consider the optimization of automated warehouses where components such as colors, pigments and bulk products are stored in identical boxes. When a production order is processed its bill of materials is sent to the weighing unit, close to the warehouse. The boxes containing the needed components are then brought to the weighing unit through a conveyor and an AGV system; with the same means the boxes are returned to the warehouse afterwards.

The structure of the automated warehouse under study is illustrated in Fig. 1. The boxes are stored on the two sides of a rail, carrying a crane. In each of the H available positions along the rail, V vertically aligned locations are available and the crane can reach them by moving up and down. In addition, each location can contain two boxes: one directly facing the corridor with the crane (front layer) and the second one just behind it (rear layer). The crane has capacity two, i.e. it can carry up to two boxes simultaneously. To access a box in the rear layer the crane first extracts

M. Barbato · A. Ceselli · G. Righini (✉)
Department of Computer Science, University of Milan, Milan, Italy
e-mail: michele.barbato@unimi.it; alberto.ceselli@unimi.it; giovanni.righini@unimi.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_14

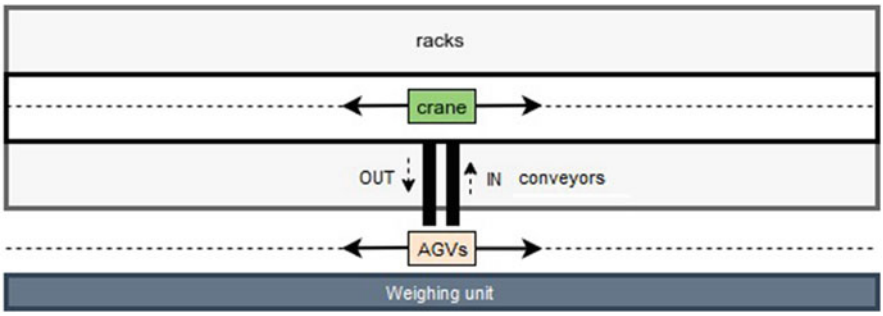


Fig. 1 The automated warehouse

the box in the front layer, then it extracts the box in the rear layer and finally it reinserts the box in the front layer. Hence, to perform this sequence of operations the crane must be initially empty. Two adjacent locations of the automated warehouse are taken by the input and the output conveyors, running in opposite directions. In the remainder the locations of the conveyors are identified as the *origin*, since they are the position from which the crane starts and to which the crane returns. A *trip* is the sequence of movements of the crane between two consecutive visits to the origin. In the remainder we call *site* the position where a box can be stored in the warehouse. So, the warehouse is assumed to have $V \times H$ locations on one side, $V \times H - 2$ locations on the other side (where the conveyors are) and 2 sites (front and rear) for each location.

The crane must execute a set of pickup operations to send the required boxes to the weighing unit and a set of delivery operations needed to put the used boxes back into their locations. The same component may be stored in several boxes: therefore the same pickup order may be satisfied by visiting any site in a given subset. On the contrary, delivery operations are assumed to be constrained to store the boxes into the same sites from which they had been extracted: therefore each delivery operation is satisfied by visiting a given site. Another asymmetry between pickup and delivery operations is their sequence. Pickup operations can be satisfied in any order, because there is no precedence between weighing different components in a bill of materials. On the contrary, when used boxes are returned to the warehouse, the order in which they arrive depends on the weighing unit and it cannot be chosen in the warehouse: the crane must process the incoming boxes in the order they arrive on the input conveyor. Therefore the input of the optimization algorithm may include a set of pickup orders (each corresponding to a subset of sites), a sequence of deliveries (each corresponding to a single site) or both.

The optimization of the crane movements is a hard combinatorial problem that, in addition, may need to be solved both off-line and on-line (i.e. re-optimized in real time). The study originates from the collaboration with a manufacturing company whose warehouse has $H = 53$, $V = 8$ and 4 weighing units. In average, a typical instance to be solved may have 13 orders per day with 6 ingredients per order, while

the speed of the crane corresponds to about 2 horizontal sites per second and 0.4 vertical sites per second.

In [2] we evaluate the applicability of state-of-the-art on-line algorithms in a similar context. In this short paper we concentrate on the off-line version, in which we have considered the simplest variations of the problem for three purposes: (a) to possibly establish a starting point for a decomposition algorithm allowing for the exact off-line optimization of the most difficult variations of the problem; (b) to pave the way for the design of on-line optimization strategies; (c) to have a term of comparison against which an on-line strategy can be compared.

In particular, we classified the problem variations according to the following features:

1. Capacity q of the crane: [1], [2], [$q > 2$];
2. Dimensions l: [1] (a line or two lines originating from the origin, i.e. the I/O position), [2] (a $H \times V$ matrix of locations), [3] (double layer);
3. Operations o: pickup only [P], delivery only [D], mixed pickups and deliveries [PD];
4. Sites s: fixed [F] or variable [V] sites;

In the remainder we use a four fields notation $q/l/o/s$, where the letters indicate "any case" while the values listed above in square brackets indicate specific cases.

Here we assume that the objective to be minimized is the overall traveling time taken by the crane to perform all its duties. Other objectives may be also relevant in an industrial context, such as the minimization of energy consumption.

2 Problem Variations

2.1 Problem Variations with Capacity 1 ($1/l/o/s$)

The case with capacity 1 (which implies no double layer) is trivial in almost all variations, since the crane can visit only one of the sites for each trip. The only non-trivial case is with mixed pickups and deliveries ($1/l/PD/s$). In this case the problem can be transformed into a minimum cost bipartite matching problem. First, dummy deliveries or dummy pickups at the origin site are generated, in order to have the same number of both operations. Then a minimum cost matching between pickups and deliveries is computed, where the cost of matching a pickup i with a delivery j is $\min_{k \in P_i}\{d_{jk}\}$, where P_i is the set of sites corresponding to pickup i and d_{jk} is the distance (or travel time) between the site of j and that of k.

Complexity: $O(n^3)$, being n the number of required operations.

In the remainder we only consider variations with crane capacity equal to 2 or larger.

2.2 Basic Problem Variation: (2/1/P/F)

This variation corresponds to the easiest combination of the four fields. The problem can be solved by initially sorting the sites to be visited by non-increasing distance from the origin (separately for each line, in the case of two lines). Then they are paired so that each trip of the crane visits the two farthest sites (on the same side of the origin) not yet visited. In the case of two lines if the number of sites is odd on one of the two lines, the last trip visits a single trip, that is the closest to the origin on the line with an odd number of sites. If the number of orders is odd on both lines, the last trip visits the two sites closest to the origin, one on each line.

Complexity: $O(n \log n)$.

2.3 One Complicating Feature

Here we examine the problem variations in which only one of the four fields takes a different value with respect to the basic variation.

2.3.1 Capacity $q > 2$ (q/1/P/F)

Sites are sorted as in the basic variation. Each trip visits the q farthest sites (on the same line) not yet visited. If there are two lines with a number of sites that is not a multiple of q, let $r_1 < q$ and $r_2 < q$ the remaining sites to be visited, Then, if $r_1 + r_2 > q$ two final trips, one on each line, visit the remaining sites; else, $r_1 + r_2 \leq q$ a single final trip is enough to visit all the remaining sites.

Complexity: $O(n \log n)$.

2.3.2 Two Dimensions (2/2/P/F)

For each (unordered) pair of sites $[i, j]$ we define a matching cost

$$c_{[i,j]} = \min\{d_{0i} + d_{ij} + d_{j0}, d_{0j} + d_{ji} + d_{i0}\} \tag{1}$$

and for each site i we define a non-matching cost

$$c_{ii} = d_{0i} + d_{i0}, \tag{2}$$

where d are the (possibly asymmetric) distances or traveling times to be optimized. If the traveling times $d(w)$ also depend on the transported weight, w, then we have

$$c_{[i,j]} = \min\{d_{0i}(w_i + w_j) + d_{ij}(w_j) + d_{j0}(0), d_{0j}(w_i + w_j) + d_{ji}(w_i) + d_{i0}(0)\} \tag{3}$$

and

$$c_{ii} = d_{0i}(w_i) + d_{i0}(0), \tag{4}$$

Consider a graph $G(N \cup N', E \cup E' \cup E'')$ where

- N is the set of orders (pickups);
- N' is a copy of the same set;
- E includes edges $[i, j]$ with weight equal to half of $c_{[i,j]}$ defined as above for each pairs of elements of N;
- E' includes edges $[i', j']$ with weight equal to half of $c_{[i,j]}$ defined as above for all pairs of elements of N';
- E'' includes edges (i, i') with weight c_{ii} defined as above, linking the two copies of each vertex in N and in N'.

A perfect matching in G is made by some edges of E'', corresponding to trips visiting a single site, and by edges $[i, j]$ and $[i', j']$ corresponding to trips visiting two sites. In particular, it is trivial to prove that there exists a perfect matching of minimum cost where edges in E and edges in E' form twice the same matching.

Complexity: $O(n^3)$.

2.3.3 Three Dimensions (2/3/*P*/*F*)

The same definitions given above still apply, but now the traveling times d also take into account the time needed to access the rear layer. Furthermore, if two sites i and j are both in the rear layer, the pair $[i, j]$ is infeasible and $c_{[i,j]}$ must be set to ∞ in (1) or (3)).

For each pair of sites it is necessary to select the optimal sequence of visits, with the constraint that sites in the rear layer must be visited first. Then $d_{ij}(w)$ is set to ∞ for all i and w in (3) if j is in the rear layer.

A polynomially solvable matching problem is generated as in the previous case.

Complexity: $O(n^3)$.

2.3.4 Deliveries (2/1/*D*/*F*)

Deliveries imply an additional constraint with respect to pickups: if box i is matched with box $j > i + 1$ in a same trip, then all boxes between i and j in the input sequence must remain unmatched: the crane takes box i from the input conveyor without delivering it, then it delivers all boxes between i and j one by one and finally it takes box j and delivers both i and j in a single trip.

This problem can be reformulated as a shortest path problem on an acyclic digraph whose nodes are numbered according to the input sequence and the weight of each arc (i, j) with $j > i$ is defined as follows:

$$c_{i,j+1} = \begin{cases} \min\{d_{0,i}(w_i + w_j) + d_{i,j}(w_j) + d_{j,0}(0), d_{0,j}(w_i + w_j) + \\ +d_{j,i}(w_i) + d_{i,0}(0)\} & \text{if } j = i + 1 \\ d_{0,i}(w_i) + d_{i,0}(0) & \text{if } j = i \\ \min\{d_{0,i}(w_i + w_j) + d_{i,j}(w_j) + d_{j,0}(0), d_{0,j}(w_i + w_j) + \\ +d_{j,i}(w_i) + d_{i,0}(0)\} + \\ +\sum_{k=i+1}^{j-1}(d_{0,k}(w_i + w_k) + d_{k,0}(w_i)) & \text{if } j > i + 1 \end{cases}$$

where $d(w)$ are the traveling costs on the line, possibly dependent on the transported weight w.

Complexity: $O(n^2)$.

2.3.5 Mixed Pickups and Deliveries (2/1/*PD*/*F*)

In this case the problem cannot be transformed into a matching problem, because the cost of a trip visiting two pickup and delivery pairs is not given by the sum of two terms each one depending on a single pair. We could devise no straightforward reformulation of this problem into a polynomially solvable graph optimization problem. Therefore this remains an open question.

2.3.6 Variable Sites (2/1/*P*/*V*)

In this case we indicate with n the set of requested pickup operations and by P_i the subset of sites from which the pickup order i can be satisfied. The problem can be reformulated as a minimum cost perfect matching problem on a suitable graph. If n is odd, add a dummy pickup at the origin. The graph has a vertex for each pickup operation, it is complete and the weight of each edge $[i, j]$ is the cost (traveling time) of the most convenient trip among those that visit a site in P_i and a site in P_j.

Complexity. If each pickup order can be satisfied in p sites, the weight of each edge is the minimum among p^2 trip costs. Hence, to define the weighted graph costs $O(n^2 p^2)$ time. The computation of the minimum cost perfect matching costs $O(n^3)$ time. Therefore the worst-case time complexity is $O(n^2 p^2 + n^3)$.

Remark The same construction holds also in two and three dimensions, i.e. for variations $(2/2/P/V)$ and $(2/3/P/V)$. In one dimension it is optimal to keep only the site closest to the origin on each line for each pickup order. This reduces p to 2 and the time complexity to $O(n^3)$.

2.4 Two Complicating Features

Now we consider the six variations arising from considering two complicating features at a time.

2.4.1 Capacity and Dimensions ($q/2/P/F$) and ($q/3/P/F$)

The problem is now a Capacitated Vehicle Routing Problem with unit demands, i.e. the capacity limits the number of vertices that can be visited in each route. The problem is known to be NP-hard [1].

2.4.2 Capacity and Deliveries ($q/1/D/F$)

This variation can be efficiently solved with dynamic programming. Consider a delivery order at a time, according to the given input sequence. For each order u one has to decide whether to keep it on the crane or to deliver it. In the former case the crane remains at the origin; Moving the crane without delivering the last loaded order u is always dominated by another policy, i.e. moving the crane to do perform the same operations before loading u; but this is already considered in the dynamic programming states generated before considering u. In the second case the crane goes at least up to the site of u and along its way it serves all delivery orders kept in it up to that moment, if they are closer to the origin than u. Moreover the crane can also go further to possibly serve more delivery orders previously accumulated. Each of these possible decisions generates a new dynamic programming state.

Complexity: the number of iterations is n. After each iteration u the number of possible states is bounded by $u + u^2 + u^3 + \ldots + u^{q-1}$, which grows as $O(n^{q-1})$. Therefore the number of states grows as $O(n^q)$. From each state at most $q + 1$ possible extensions must be considered. This yields a time complexity $O(qn^q)$, that is polynomial for each q fixed.

2.4.3 Capacity and Variable Sites ($q/1/P/V$)

On a single line, the problem is trivial: it is always optimal to select the site closest to the origin for each pickup order, which makes the problem equivalent to the variation with fixed sites ($q/1/P/F$). On two lines, it is easy to prove that for each line we can keep only the site closest to the origin for each pickup order. Hence each pickup order can be satisfied in either of two sites, on opposite sides with respect to the origin. However, in spite of this simplification we could not find any polynomial time algorithm to solve the resulting model.

Therefore this problem remains open.

2.4.4 Dimensions and Deliveries (2/2/*D*/*F*) and (2/3/*D*/*F*)

The same construction of variation $(2/1/D/F)$ holds. The only change is in how arc costs are computed from traveling times. Hence, the problem can be transformed into a shortest path problem on an acyclic weighted digraph.

Complexity: $O(n^2)$.

2.4.5 Dimensions and Variable Sites (2/2/*P*/*V*) and (2/3/*P*/*V*)

In two dimensions we can re-use the same transformation of variation $(2/2/P/F)$, leading to a minimum cost perfect matching problem, after selecting:

- the most convenient site, for each single pickup order;
- the most convenient pair of sites, for each pair of pickup orders.

In three dimensions the same construction holds again, but there is no guarantee that all trips visiting two sites are feasible: any two sites in the rear layer are incompatible, because both require the crane be empty to execute the pickup operation. However the pre-processing step needed to check for incompatibilities has polynomial complexity: with n orders and p sites for each order the computation of the arc costs takes $O(n^2p^2)$. Solving the resulting minimum cost perfect matching problem requires $O(n^3)$.

Complexity: $O(n^2p^2 + n^3)$.

2.4.6 Deliveries and Variable Sites (2/1/*D*/*V*)

On a single line it is always optimal to select the site closest to the origin for each delivery order. Hence, the problem is equivalent to $(2/1/D/F)$ and can be solved in polynomial time as a shortest path problem.

On two lines it is always optimal to select on each line the site closest to the origin for each delivery order. Hence, the same transformation of variation $(2/1/D/F)$ holds again with the only difference that the quantity $\min\{d_{0i}(w_i + w_j) + d_{ij}(w_j) + d_{j0}(0), d_{0j}(w_i + w_j) + d_{ji}(w_i) + d_{i0}(0)\}$ must be chosen in an optimal way among four possibilities, corresponding to the four combinations of the two sites for each of the two orders i and j.

Complexity: $O(n^2)$.

2.5 Three Complicating Features

There is only one triplet of features that generate polynomial problems when they are taken two at a time.

2.5.1 Dimensions, Deliveries and Variable Sites $(2/2/D/V)$ and $(2/3/D/V)$

By enumeration it is possible to select the most convenient sites for each order served alone and the most convenient pairs of sites for each pair of orders served in the same trip. After that, one can still use the transformation of variation $(2/1/D/F)$ leading to a shortest path problem. The pre-processing step takes polynomial time as in the previous cases (for instance $(2/2/P/V)$ and $(2/3/P/V)$).

Complexity: $O(n^2p^2 + n^3)$.

3 Conclusions

We have analyzed several variations of a problem of optimizing the traveling time of a crane in an automated warehouse. For many variations we have shown polynomial-time transformations that allow the problem to be efficiently solved with existing algorithms for computing shortest paths or perfect matchings or with dynamic programming.

Establishing the complexity of two variations, namely $(2/1/PD/F)$ and $(q/1/P/V)$, remains open.

Other variations are already NP-hard even in this simplified version of the original problem.

The knowledge about what features are complicating and how the others can be efficiently dealt with paves the way for solving the original problem with exact optimization algorithms based on suitable decompositions or relaxations. The model can be further enriched by considering additional features such as deadlines and energy consumption minimization as well as uncertainty and real-time decision policies. This is the subject of an ongoing research program.

Acknowledgement This research was partially funded by Regione Lombardia, grant agreement n. E97F17000000009, project AD-COM.

References

1. Toth, P., Vigo, D. (eds.) The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia (2002)
2. Barbato, M., Ceselli, A., Mosconi, F.: A Computational Evaluation of Online ATSP Algorithms, Technical Report of University of Milan, submitted to ODS 2019

Coordinating the Emergency Response of Ambulances to Multiple Mass Casualty Incidents using an Optimization-based Approach

Haya Aldossary and Graham Coates

Abstract During the emergency response to multiple mass casualty incidents (MCIs), a number of coordination (allocation) decisions need to be made in a timely manner. This paper reports on an optimization-based approach that has been developed to solve the ambulance-to-casualty and casualty-to-hospital allocation problems. A number of constraints are taken into consideration such as the number of ambulances and hospitals, along with the capacity of hospitals. Within the approach, the road network of the geographical area under consideration is modelled realistically. Further, the day of the week and the time of day at which multiple MCIs occur are considered as factors influencing the speed of the ambulances. The approach includes a Neighborhood Search Algorithm that has been developed and used to obtain solutions to a multiple MCI case study involving a number of scenarios.

Keywords Coordination · Multiple mass casualty incidents · Emergency response · Ambulance-to-casualty allocation · Casualty-to-hospital allocation

1 Introduction and Problem Statement

A mass casualty incident (MCI) can be defined as that which results in a number of casualties with varying degrees of injury and which may overwhelm the emergency services and hospitals allocated to receive casualties. An example of a multiple MCI

H. Aldossary (✉)
Newcastle University, Newcastle upon Tyne, UK

Imam Abdulrahman Bin Faisal University, Jubail, Saudi Arabia
e-mail: h.aldossary2@newcastle.ac.uk; healdossary@iau.edu.sa

G. Coates
Newcastle University, Newcastle upon Tyne, UK
e-mail: Graham.Coates1@newcastle.ac.uk

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_15

is the terrorist bombings in 2005 in London, UK [1]. In relation to the London bombings, the UK Cabinet reported a potential lack in the coordination between the emergency services [1]. During a MCI, coordination challenges may arise due to the limited number of emergency service resources, demands on local hospitals, and potential impact on the road network. Thus, the emergency service coordination challenges that could be involved in a multiple MCI are task flow including task allocation and task scheduling, resource allocation and re-allocation, information flow between responders, the speed of making decisions, and the relationship between responders [2]. In this paper, resource allocation is considered as the efficient sharing of resources, such as ambulances and hospitals, between casualties. Further, task allocation is considered as assigning a number of tasks to ambulances involved in the emergency response to a multiple MCI and ensuring that there is no conflict, overlap, or duplication between ambulance assignments.

In the context of a multiple MCI, Wilson et al. [3] designed a Neighbourhood Search Algorithm to solve the task and resource allocation problems. The tasks considered include pre-extricating treatment, rescuing trapped casualties, pre-transportation treatment, and transporting casualties to hospital, all of which are assigned to a number of responders. Similarly, Repoussis et al. [4] proposed a mixed integer programming model for optimizing emergency resource utilization. In that work, two tasks have been taken into consideration: casualty transportation to a hospital and casualty treatment at hospital. Amram et al. [5] have focused on solving the challenge of patient-to-hospital allocation by proposing a web-based simulation model. The model is designed to provide the responders at an incident site with real-time information regarding driving time, the location of hospitals, and the level of the trauma service available at these hospitals. Importantly, this model requires real-time data, which is not always available. In the field of disaster logistics, Salman and Gul [6] proposed a multi-period mixed integer programming model to optimize the decisions of allocating resources and transporting casualties with the aim of minimizing the response time. In addition to the allocation challenges mentioned, a number of researchers in the emergency field have focused on the triage process and patient prioritization [7–9]. For example, Mills et al. [9] presented a model of casualty triage in a MCI that incorporates resource limitations and changes in survival probabilities with respect to time. In this model, patients are catgeorised as immediate (requiring treatment within 1 h), urgent (needing treatment between 1 and 4 h), delayed (can wait for treatment beyond 4 h) and expectant (expected to become deceased). The performance of the model presented is examined using a set of scenarios where the number of ambulances has been varied to compare the impact of the resource-scarce and resource-abundant situations on the deterioration of casualties.

Some authors have assumed that the injury level of casualties is static during the emergency response process [4, 10] or has no effect [11]. In this paper, the injury state of casualties is considered to be static but it is used in prioritizing casualties in relation to their order of transportation to hospitals. Further, in simulating the response to emergency situations, a non-realistic representation of the road network is commonly modelled using a simple representation such as a grid [4, 12, 13]. However, a realistic and detailed representation of the road network is required, as

in [3], to accurately determine the distance to and from incident sites, and to and from hospitals. Relatedly, to obtain accurate travel times of ambulances needing to travel between two destinations, factors such as distance and speed are required. Based on the distance, Wilson et al. [3] applied two functions proposed by [14] to determine the travel time, although these functions are not time-dependent and the time of the occurrence of the incident is assumed to be constant. Models such as [9] also consider speed and distance as static values. In contrast, in [12] speed is proposed to be a variable based on time of day and day of the week.

The aim of the research presented in this paper is to solve the coordination problem of the emergency response of a limited number of ambulances to multiple, simultaneously occurring MCIs, each of which has a number of casualties with varying levels of injury. Ambulances are available to deliver casualties to one of a fixed number of hospitals, each of which has a limited capacity.

Specifically, the coordination problem to be solved consists of a number of MCIs located at a number of incident sites, n_{IS}, that involve multiple casualties, n_C, each of which must initially be triaged (i. e. classified as immediate, urgent, or delayed) by ambulance crew on arrival at the incident site. In this research, it is assumed that all casualties have been triaged at the beginning of the emergency response process. A limited number of ambulances, n_A, initially located at a number of ambulance stations, n_{AS}, are used to transport each casualty from an incident site to one of a number of hospitals, n_H, each of which has a limited capacity. For each ambulance to which a casualty is allocated, a number of tasks, n_T,need to be completed; see Fig. 1.

Each ambulance located at an ambulance station begins its first task (i.e. preparing to respond) once the MCI emergency call is received. The first journey of each ambulance starts when it departs from an ambulance station to travel to an incident site to collect a casualty and ends when the triaged casualty is dropped-off at a hospital. Normally, each casualty is transferred individually by an ambulance to the allocated hospital. However, in some instances, pairs of casualties with the same triage classification may be transported to hospital in the same ambulance. A number of allocation decisions are made as part of the response including: (a) which ambulance from which ambulance station should be dispatched to which casualty at which incident site; (b) which casualty located at an incident site should be allocated to a which hospital.

The contribution of the research lies in the development of an optimization-based approach, which includes a GIS-based representation of a multiple MCI geographic environment in any area of the UK under consideration coupled with a Neighborhood Search Algorithm (NSA), to solve the coordination problem

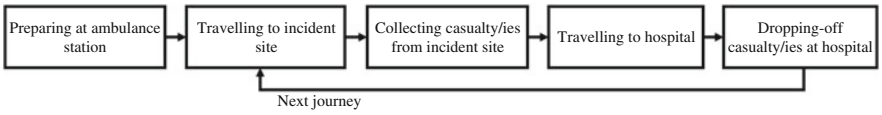


Fig. 1 The sequence of tasks allocated to an ambulance

mentioned. In relation to the NSA, a number of objectives are considered, i.e. the minimization of the: (a) maximum response time of all ambulances; (b) time of delivering the last 'immediate' casualty to hospital; (c) time of delivering the last 'urgent' casualty to hospital; (d) total number of journeys of all ambulances; (e) total idle time of all ambulances. The NSA developed consists of seven structures that can be applied to solve two allocation problems: ambulance-to-casualty and casualty-to-hospital. Further, road traffic is simulated by varying the speed of ambulances based on the day and time of the occurrence of the multiple MCIs.

The remainder of this paper is organized as follows, Sect. 2 presents the optimization-based approach to solve the coordination problem defined in Sect. 1 in terms of the initialization of multiple MCIs in a given geographical area, objective functions and a Neighborhood Search Algorithm that has been developed. A hypothetical case study is described in Sect. 3, followed by the associated results and discussion in Sect. 4. Finally, the paper is concluded in Sect. 5.

2 Optimization-Based Approach

2.1 MCI Initialization

To simulate the affected geographical area of the multiple MCIs, the approach's solution method requires initialization of the associated road network. In this research, the road network is constructed using Ordnance Survey MasterMap GIS data [15] and modelled as an undirected graph, which has sufficient details to determine the actual distance between any two locations and thus the travelling time between them. Further, the MCIs must be initialized to include the number and location of: (a) incident sites, *IS*; (b) casualties, *C*, at each incident site (including their injury level); (c) hospitals, *H*; (d) ambulance stations, *AS*; (e) ambulances, *A*, at each ambulance station. Each location in the GIS-based representation of the area is assigned to the vertex on the graph nearest its *real* location (Table 1).

2.2 Objective Functions

The response time, which is the time of delivering the last casualty among all incident sites to hospital, is minimized

$$f_1 = \sum_{k=1}^{n_t} d_{k,i,j} \tag{1}$$

where $d_{k,\,i,\,j}$ refers to the duration of the k-th task allocated to the i-th ambulance, which was originally located at the j-th ambulance station.

Table 1 Sets and parameters

Sets	Definition
A	Set of ambulances
AS	Set of ambulance stations
C	Set of casualties
H	Set of hospitals
IS	Set of incident sites
T	Set of tasks
Parameters	Definition
a_{cap}	Ambulance capacity
$h_{i,\,cap}$	The capacity of hospital i
n_{IS}	Number of incident sites
n_C	Number of casualties
n_H	Number of hospitals
n_{AS}	Number of ambulance stations
n_A	Number of ambulances
n_T	Number of tasks assigned to each ambulance
n_J	Number of journeys taken by each ambulance
$s_d^{(t)}$	Ambulance speed based on the day of week d and time of day t

The time of delivering the (a) last immediate casualty then (b) the last urgent casualty among all incident sites to the hospital are minimized

$$f_2 = \max_{i,j} \; dt_{I,i,j} \tag{2}$$

$$f_3 = \max_{i,j} \; dt_{U,i,j} \tag{3}$$

where $dt_{I,\,i,\,j}$ and $dt_{U,\,i,\,j}$ indicate the delivery time of the last immediate and last urgent casualty who have been transferred to a hospital by the i-th ambulance, which was originally located at the j-th ambulance station.

The number of journeys for all ambulances is minimized

$$f_4 = \sum\nolimits_{t=1}^{n_J} aj_{t,i,j} \tag{4}$$

where $aj_{t,\,i,\,j}$ indicates the $\mathrm{t}-\mathrm{th}$ journey taken by the $\mathrm{i}-\mathrm{th}$ ambulance, which was originally located at the $\mathrm{j}-\mathrm{th}$ ambulance station. Further, n_J indicates the number of journeys taken by each ambulance.

The total idle time of all ambulances is minimized

$$f_5 = \sum\nolimits_{t=1}^{n_J} ait_{t,i,j} \tag{5}$$

where $ait_{t,\,i,\,j}$ denotes the total idle time of the i-th ambulance, which was originally located at the j-th ambulance station.

2.3 Neighborhood Search Algorithm

A NSA has been developed to solve the allocation of ambulances-to-casualties and casualties-to-hospitals. At each iteration, the NSA generates a new response plan by selecting and applying one of seven structures: (a) change the order of processing the selected casualty within the same ambulance (COCSA); (b) allocate the selected casualty to a different ambulance (MCDA); (c) swap two casualties within the same ambulance (SCSA); (d) swap two casualties within different ambulances (SCDA); (e) balance the workload between the ambulance that has the highest workload and the one has the lowest (BW); (f) allocate a casualty to a different hospital (CH); (g) swap the assigned hospital of two selected casualties (SH). The new response plan generated each iteration is only accepted when there is an improvement when compared with the current plan; otherwise, the new plan is discarded. In terms of improvement, objective function f_1 is considered first; however, if the value of f_1 obtained for the new plan is equal to that of the current plan then f_2 will be considered and so on; refer to Fig. 2 for details.

Dijkstra's algorithm is first used to find the shortest distance between two locations in the road network of the area of interest, which is then used to obtain the travel time between those locations. Subsequently, the average speed of ambulances used to travel the distance between those locations is taken from [12] based on the time and day of the occurrence of the multiple MCIs modelled.

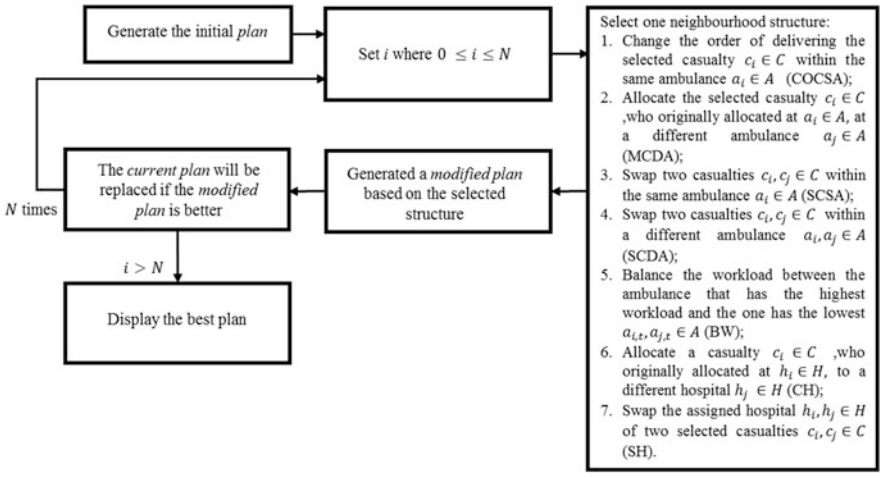


Fig. 2 Flowchart of the NSA

3 Case Study: Hypothetical Multiple MCI

A hypothetical multiple MCI involving three incidents at separate sites is assumed to occur simultaneously in Leeds on a Monday at 10:00 a.m. The incident sites $\{is_1, is_2, is_3\} \in IS$ are located at Gotts Park (GP), Royal Armouries Museum (RAM), and Bedford Fields Green Link (BFGL) as indicated in Fig. 3.

In Fig. 3, the top right coordinates (432593.2, 436520) and bottom left coordinates (427084.1, 730701.8) represent the easting and northing respectively. Also in Fig. 3, three ambulance stations $\{as_1, as_2, as_3\} \in AS$ are shown : St John Ambulance Leeds – Harehills (SJH) which has 8 ambulances; St John Ambulance Leeds – City Centre (SJC) which has 10 ambulances; Leeds Central Ambulance Station (LC) which has 20 ambulances. All ambulances are assumed to be the same and are available to respond simultaneously to the MCI's three incident sites $\{is_1, is_2, is_3\} \in IS$. Two hospitals $\{h_1, h_2\} \in H$ are able to receive a limited number of casualties based on the capacity of each hospital $h_{i,\,cap}$; Leeds General Infirmary (LGI) can receive up to 50 casualties and St James' University Hospital (SJUH) can receive up to 40 casualties.

As summarized in Table 2, a total of 80 casualties are distributed over the three incident sites: 20 casualties at GP (7 immediate, 10 urgent and 3 delayed); 40 casualties at RAM (11 immediate, 16 urgent and 13 delayed); 20 casualties at BFGL (5 immediate, 7 urgent and 8 delayed).

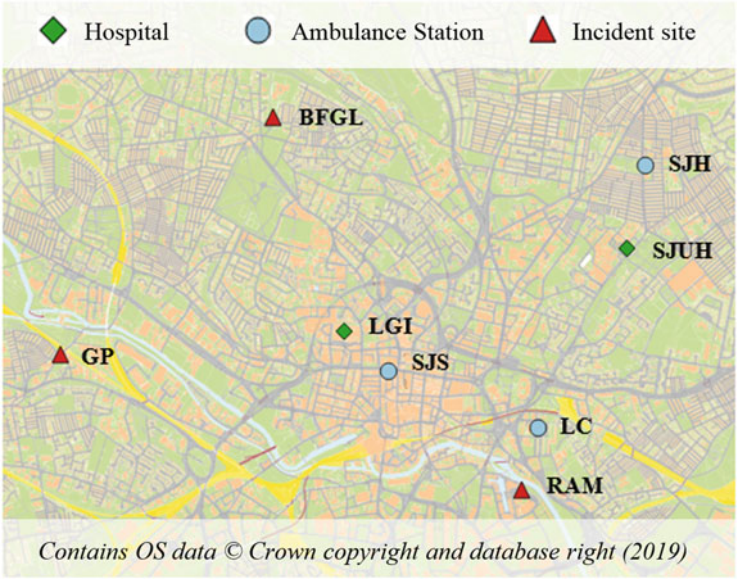


Fig. 3 Map of Leeds with the road network denoted by grey lines

Table 2 Distribution of casualties over the three incident sites

Incident site	Number of casualties			Total
	Immediate (I)	Urgent (U)	Delayed (D)	
GP	7	10	3	20
RAM	11	16	13	40
BFGL	5	7	8	20

At each incident site, casualties are assumed to be triaged and waiting to be transported to one of the hospitals. The priority of casualties being transferred to hospitals is based on their triage category.

4 Results and Discussion

To enable the performance of the optimization-based approach to be evaluated, three scenarios of the hypothetical multiple MCI case study described in Sect. 3 have been considered. In scenario 1, immediate casualties must be transferred individually to a hospital by an ambulance whereas delayed casualties are assumed to not require transfer to a hospital. Also, up to two urgent casualties can be transferred in a single ambulance from an incident site to a hospital. In scenario 2, the same conditions are applied as in scenario 1 except that pairs of urgent casualties cannot be transferred together in the same ambulance. In scenario 3, the same conditions are applied as in scenario 1 except delayed casualties need to be transferred to hospital, which may be done individually or in pairs when the opportunity arises. In all three scenarios, the NSA has been applied 50 times, each time involving 4000 iterations taking approximately 3–4 min. The algorithm is implemented in the Java language and executed on a PC with 2.0 GHz and 1.0 GB of RAM. The three scenarios are summarized in Table 3.

In relation to scenario 1, the horizontal axis in Fig. 4 indicates all instances of an improvement in response time (f_1) for the best run of the NSA. That is, each improvement indicated corresponds with a particular iteration of the NSA; 38 improvements were made over 4000 iterations. In addition, for each improvement in f_1, indications are given for the time of delivering the last immediate casualty to hospital (f_2), the time of delivering the last urgent casualty to hospital (f_3), and the total number of journeys for all ambulances (f_4).

Figure 5 shows the total idle time of all ambulances (f_5) corresponding with each improvement in response time seen in Fig. 4.

Table 4 summarizes the best results, i.e. f_1–f_5, for all three scenarios.

In Table 4, a comparison of the results of scenarios 1 and 2 indicates that transferring urgent casualties in pairs (permissible in scenario 1) could save 2 min and 53 s in response time (f_1) and the time taken to deliver the last urgent casualty (f_3). It is noted that these two objectives are equivalent in these two scenarios since

Table 3 Definitions of scenarios

	Transfer of casualties to hospitals					
	Immediate		Urgent		Delayed	
Scenario	Individually	Pairs	Individually	Pairs	Individually	Pairs
1	Yes	No	Yes	Yes	No	No
2	Yes	No	Yes	No	No	No
3	Yes	No	Yes	Yes	Yes	Yes

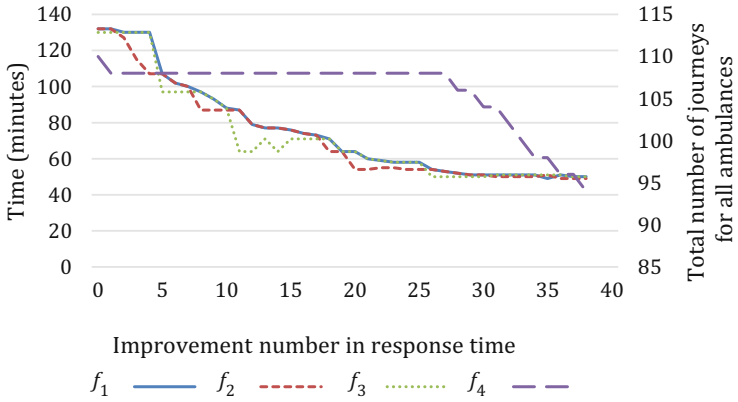


Fig. 4 Values of objectives f_1–f_4 for scenario 1

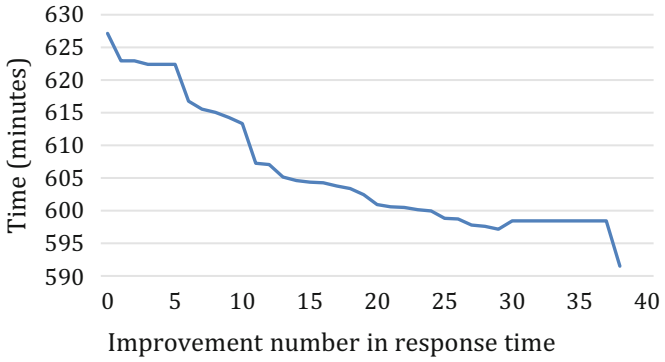


Fig. 5 Value of objective f_5 for scenario 1

Table 4 'Best' values of objectives for all three scenarios

Scenario	f_1 (min:s)	f_2 (min:s)	f_3 (min:s)	f_4	f_5 (h:min:s)
1	49:32	49:00	49:32	94	09:51:30
2	52:25	48:31	52:25	112	09:10:12
3	58:14	50:01	58:14	130	06:43:42

Table 5 Details of the journeys of all ambulances for all scenarios

Scenario	Number of journeys							
	AS to *IS*	*IS* to *H*					*H* to *IS*
		I	U	U_P	D	D_P	
1	36	23	15	9	–	–	11
2	38	23	33	–	–	–	18
3	38	23	23	5	4	10	27

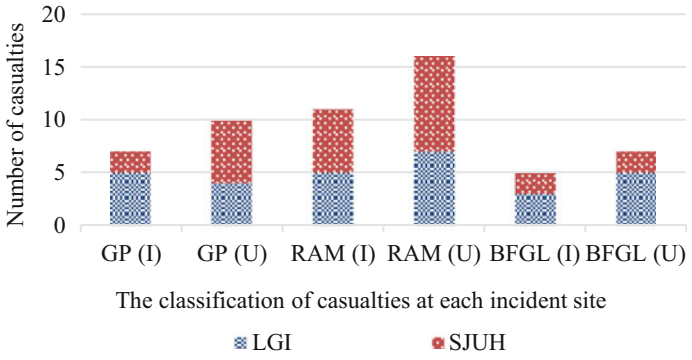


Fig. 6 Scenario 1: allocation of immediate (I) and urgent (U) casualties from incident sites to hospitals

delayed casualties are not considered. Also, the total number of journeys of all ambulances (f_4) could be reduced by 18. In scenario 3, in which delayed casualties are transferred to hospitals, the response time (f_1) and the time of delivering the last urgent casualty (f_3) both increase by 8 min and 42 s when compared with scenario 1. In addition, the total number of journeys of all ambulances (f_4) increases by 36, which is in part due to the additional 24 delayed casualties needing to be transferred to hospital. For scenarios 1–3, Table 5 presents a breakdown of the number of ambulance journeys between ambulance stations *AS* and incident sites *IS*, incident sites and hospitals *H*, and hospitals and incident sites. Further, in relation to the number of journeys between incident sites and hospitals, a further breakdown is given in terms of journeys involving an ambulance carrying a single immediate casualty (I), a single urgent casualty (U), a pair of urgent casualties (U_P), a single delayed casualty (D), and a pair of delayed casualties (D_P).

For scenario 1, Fig. 6 shows the number of casualties allocated to each hospital versus their classification at each incident site. In summary, 29 casualties are taken to hospital LGI whereas 27 casualties are taken to hospital SJUH. In Fig. 6, the results show that most of the immediate casualties located at the GP incident site are transferred to the nearest hospital, which is LGI. However, the immediate casualties located at the RAM and BFGL incident sites are allocated almost equally between both hospitals LG1 and SJUH. For the immediate casualties located at the RAM, this near-equal casualty-to-hospital allocation could be attributed to the fact that the distance between this incident site and both hospitals is almost equidistant (i.e. 1.8 km).

In relation to the GP incident site, eight casualties (i.e. 2 immediate and 6 urgent) are transferred to hospital SJUH despite the distance between these locations being 2.2 km greater than the distance between the incident site and hospital LGI. The reason for this is that the ambulance journeys to deliver these casualties to hospital SJUH are not critical in terms of affecting the objective values f_1, f_2 and f_3. That is, in scenario 1, the critical path involves the journeys from ambulance station SJH to incident site GP to collect casualties, who are transported to hospital LGI.

5 Conclusion

MCIs require a sequence of actions to be undertaken during the emergency response to enable casualties to be delivered to the appropriate hospital in a timely manner. Indeed, emergency response time is critical in multiple MCIs in terms of saving lives and reducing suffering of casualties with varying injury levels. A number of coordination problems arise during the emergency response to multiple MCIs including ambulance-to-casualty allocation and casualty-to-hospital allocation. This paper has presented an optimization-based approach, incorporating a Neighborhood Search Algorithm, to solve these coordination problems. Further, a hypothetical, multiple MCI has been modelled in which three scenarios are considered. These scenarios differ in terms of how casualties can be transported to hospital via ambulances. The results show the expected improvement in response time in allowing pairs of urgent casualties and pairs of delayed casualties to be transported to hospital in a single ambulance. However, the degree of improvement also involves consideration of the relative position of ambulance stations at which ambulances are initially located, incident sites from which casualties must be collected and hospitals to which they should be delivered.

For future work, the dynamic nature of the injury level or health state of casualties could be taken into consideration when allocating ambulances to casualties, and casualties to hospitals. In addition, multiple MCIs occurring at different times during the response could be modelled, which requires re-planning the response and re-allocating the emergency services performing the response.

References

1. The UK Cabinet Office: National Risk Register of Civil Emergencies (2017)
2. Chen, R., Sharman, R., Rao, H.R., Upadhyaya, S.J.: Coordination in emergency response management. Commun. ACM. **51**(5), 66–73 (2008)
3. Wilson, D.T., Hawe, G.I., Coates, G., Crouch, R.S.: Online optimization of casualty processing in major incident response: an experimental analysis. Eur. J. Oper. Res. **252**(1), 334–348 (2016)
4. Repoussis, P.P., Paraskevopoulos, D.C., Vazacopoulos, A., Hupert, N.: Optimizing emergency preparedness and resource utilization in mass-casualty incidents. Eur. J. Oper. Res. **255**, 531–544 (2016)

5. Amram, O., Schuurman, N., Hedley, N.: A web-based model to support patient-to-hospital allocation in mass casualty incidents. J. Trauma. **72**(5), 1323–1328 (2012)
6. Salman, F.S., Gul, S.: Deployment of field hospitals in mass¨casualty incidents. Comput. Ind. Eng. **74**(1), 37–51 (2014)
7. Hupert, N., Hollingsworth, E., Xiong, W.: Overtriage and outcomes: insights from a computer model of trauma system mass casualty response. Disaster Med. Public Health Prep. **1**(Suppl. 1), 14–24 (2007)
8. Sung, I., Lee, T.: Optimal allocation of emergency medical resources in a mass casualty incident: patient prioritization by column generation. Eur. J. Oper. Res. **252**(2), 623–634 (2016)
9. Mills, A.F., Argon, N.T., Ziya, S.: Resource-based patient prioritization in mass-casualty incidents. Manuf. Serv. Oper. Manag. **15**(3), 361–377 (2013)
10. Gong, Q., Batta, R.: Allocation and reallocation of ambulances to casualty clusters in a disaster relief operation. IIE Trans. **39**(1), 27–39 (2007)
11. Mete, H.O., Zabinsky, Z.B.: Stochastic optimization of medical supply location and distribution in disaster management. Int. J. Prod. Econ. **126**(1), 76–84 (2010)
12. McCormack, R., Coates, G.: A simulation model to enable the optimization of ambulance fleet allocation and base station location for increased patient survival. Eur. J. Oper. Res. **247**(1), 294–309 (2015)
13. Niessner, H., Rauner, M.S., Gutjahr, W.J.: A dynamic simulation–optimization approach for managing mass casualty incidents. Oper. Res. Health Care. **17**, 82–100 (2017)
14. Kolesar, P., Walker, W., Hausner, J.: Determining the relation between fire engine travel times and travel distances in New York City. Oper. Res. **23**(4), 614–627 (1975)
15. Ordnance Survey. Ordnance survey: Britain's mapping agency. (Online). https://www.ordnancesurvey.co.uk/. Accessed 07 Feb 2019

A Game Theory Model of Online Content Competition

Georgia Fargetta and Laura Scrimali

Abstract This paper develops a game theory model consisting of online content providers and viewers, where providers compete for the diffusion of their contents on a user-generated content platform. Each provider seeks to maximize the profit by determining the optimal views and quality levels of their digital products. The viewers reflect their preferences through the feedback functions, which depend on the amount of views and on the average quality level. The governing equilibrium conditions of this model are formulated as a variational inequality problem. Moreover, we analyze the Lagrange multipliers and discuss their role in the behavior of providers. Finally, our results are applied to an example of content competition on YouTube.

Keywords User-generated contents · Nash equilibrium · Variational inequalities

1 Introduction

On the platform of the World Wide Web, online contents have registered a tremendous growth. Most of such contents are digital and posted by the contents' owners on a user-generated content (UGC) platform, like Instagram, Facebook, YouTube, Twitter and more.

In an overcrowded digital marketplace, with millions of blogs, guides, etc., ensuring a large audience to a content is not an easy task. YouTube, for instance, provides tools to accelerate the dissemination of contents, using recommendation lists and other re-ranking mechanisms. Therefore, the diffusion of a content can be increased by paying an additional cost for advertisement. As a consequence, the content will gain some priority in the recommendation lists and will be accessed more frequently by users. Finally, the acceleration mechanism generates

G. Fargetta · L. Scrimali (✉)
DMI, University of Catania, Catania, Italy
e-mail: scrimali@dmi.unict.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_16

competition among online content providers to gain popularity, visibility, influence and reputation.

The literature on the competition of online contents is vast and mainly focuses on the evolution of popularity of online contents; see [3, 4]. The aim is to develop models for early-stage prediction of contents' popularity. In [1, 2, 7], the authors model the behavior of contents' owners as a dynamic game. In addition, some acceleration mechanisms of views are incorporated in the formulation.

In this paper, we develop a game theory model consisting of online content providers and viewers, where providers behave in a non cooperative manner for the diffusion of their contents. Each provider seeks to maximize the profit by determining the optimal views and quality levels of his digital product. Viewers express their preferences through their feedback functions, that may depend upon the entire volume of views and on the average quality level. The governing concept is that of Nash equilibrium (see, [11, 12]), which is then formulated as a variational inequality (see, [8, 10]). We also present an alternative formulation of Nash equilibria using the Lagrange multipliers, that allows us to analyze better the strategic decisions of providers and the marginal profits. Several papers are devoted to the study of equilibrium models by means of the Lagrange multipliers; see, for instance, [6] for the financial equilibrium problem, [9] for the pollution control problem, [13] for the electricity market, and [5] for cybersecurity investments.

The paper is organized as follows. In Sect. 2, we present the model, and give the variational inequality formulation. In Sect. 3, we discuss the role of Lagrange multipliers. In Sect. 4, we illustrate a numerical example, and, finally, we draw our conclusion in Sect. 5.

2 The Game Theory Model

In this section, we present an online content diffusion network that consists of m content providers and n viewers, see Fig. 1.

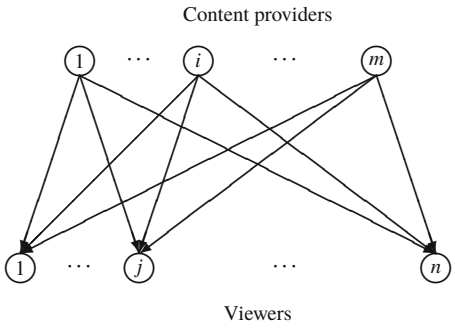


Fig. 1 The two-layer online content diffusion network

Each content provider i, $i = 1, \ldots, m$, posts a content that can be accessed by each viewer j, $j = 1, \ldots, n$. The contents are assumed to be homogeneous, namely, of the same type (for instance, blogs, videos, podcasts, social media contents, ebooks, photos, etc.), and of a similar topic (for instance, music, travels, film reviews, recipes, etc.). The viewers can access each of the m contents at the first opportunity.

Let Q_{ij} denote the access selected by viewer j of content i. We group the $\{Q_{ij}\}$ elements for all j into the vector $Q_i \in \mathbb{R}^n_+$ and then we group all the vectors Q_i for all i into the vector $Q \in \mathbb{R}^{mn}_+$.

In addition, q_i denotes the quality level of content i and takes a value in the interval $I = [1, 5]$, where 1 = sufficient, 2 = satisfactory, 3 = good, 4 = very good, 5 = excellent. We group the quality levels of all providers into the vector $q \in \mathbb{R}^m_+$.

All vectors here are assumed to be column vectors, except where noted. The interest towards contents of each viewer j, denoted by d_j, reflects the taste for the digital product that is posted and must satisfy the following conservation law:

$$d_j = \sum_{i=1}^{m} Q_{ij}, \quad j = 1, \ldots, n. \tag{1}$$

Let s_i denote the number of views of the content posted by provider i, which is given by $s_i = \left\lfloor \sum_{j=1}^{n} Q_{ij} \right\rfloor$, $i = 1, \ldots, m$. Hence, the number of views of the content posted by each provider is equal to the sum of the accesses of all the viewers.

Usually, a content must reach a minimum threshold of accesses to gain the interest of viewers and be in competition with the other homogeneous contents. In addition, each content has a lifetime, namely, the amount of views is limited. Therefore, for each Q_{ij}, we introduce the lower bound $\underline{Q}_{ij} \geq 0$ and the upper bound $\overline{Q}_{ij} \geq 0$, so that $\underline{Q}_{ij} \leq Q_{ij} \leq \overline{Q}_{ij}$ for all i, j. We group the $\{\underline{Q}_{ij}\}, \{\overline{Q}_{ij}\}$ elements for all j into the vectors $\underline{Q}_i, \overline{Q}_i \in \mathbb{R}^n_+$, respectively, and then we group all the vectors $\underline{Q}_i, \overline{Q}_i$ for all i into the vectors $\underline{Q}, \overline{Q} \in \mathbb{R}^{mn}_+$, respectively.

We associate with each content provider i a production cost

$$f_i(Q, q_i), \quad i = 1, \ldots, m, \tag{2}$$

and consider the general situation where the production cost of i may depend upon the entire amount of views and on its own quality level. We assume that the production cost is convex and continuously differentiable.

We also assume that providers can pay a fee for the advertisement service in the UGC platform. Such a fee allows a provider to accelerate the views. Hence, for each provider i, we introduce the advertisement cost function

$$c_i \sum_{j=1}^{n} Q_{ij}, \quad i = 1, \ldots, m, \tag{3}$$

with $c_i \geq 0$, $i = 1, \ldots, m$. Similarly, the revenue of provider i (revenue for hosting advertisements, benefits from firms, etc.) is given by

$$p_i \sum_{j=1}^{n} Q_{ij}, \quad i = 1, \ldots, m, \tag{4}$$

with $p_i \geq 0$, $i = 1, \ldots, m$. Each viewer j reflects the preferences through the feedback function, that represents the evaluation of the contents:

$$F_j(d, \bar{q}), \quad j = 1, \ldots, n, \tag{5}$$

where $\bar{q} = \dfrac{1}{m} \sum_{i=1}^{m} q_i$ is the average quality level. Due to (1), with abuse of notation, we can write $F_j(d, \bar{q}) = F_j(Q, q)$ for all j. Thus, we consider a general case where the feedback function may depend upon the entire amount of views Q and the total quality level.

Now, we can define the reputation or popularity function of provider i as the function

$$\sum_{j=1}^{n} F_j(Q, q) Q_{ij}, \quad i = 1, \ldots, m. \tag{6}$$

We assume that the reputation function is concave and continuously differentiable.

The content diffusion competition can be represented as a game where we define players, strategies and utilities. Players are content providers, who compete for the diffusion of their contents. Strategic variables are content views Q and quality level q. Profit for player i is the difference between total revenues and total costs, namely,

$$U_i(Q, q) = \sum_{j=1}^{n} F_j(Q, q) Q_{ij} + p_i \sum_{j=1}^{n} Q_{ij} - f_i(Q, q_i) - c_i \sum_{j=1}^{n} Q_{ij}, \quad i = 1, \ldots, m. \tag{7}$$

We observe that due the concavity of the reputation function and the convexity of the production cost, the profit function U_i is concave. This will allow us to present a variational inequality formulation of the game, see subsequent Theorem 1.

Let $\mathbb{K}_i$ denote the feasible set of content provider i, where

$$\mathbb{K}_i = \left\{ (Q_i, q_i) \in \mathbb{R}^{n+1} : \underline{Q}_{ij} \leq Q_{ij} \leq \overline{Q}_{ij}, \forall j; 1 \leq q_i \leq 5 \right\}. \tag{8}$$

We also define

$$\mathbb{K} = \prod_{i=1}^{m} \mathbb{K}_i = \left\{ (Q, q) \in \mathbb{R}^{mn+m} : \underline{Q}_{ij} \leq Q_{ij} \leq \overline{Q}_{ij}, \forall i, j; 1 \leq q_i \leq 5, \forall i \right\}. \tag{9}$$

In our model, the m providers post their contents and behave in a non-cooperative fashion, each one trying to maximize his own profit (see also, [14]). We note that the production cost functions capture competition for contents since the production cost of a particular provider depends not only on his views, but also on those of the other providers. Moreover, the feedback functions show that viewers care about the quality level associated with their favorite contents, but also on that of the other viewers, as well as the content views. Therefore, we seek to determine the amount of views and the quality level pattern (Q^*, q^*) for which the m providers will be in a state of equilibrium as given in the following definition.

Definition 1 (Nash Equilibrium) A view amount and quality level pattern $(Q^*, q^*) \in \mathbb{K}$ is said to be a Nash equilibrium if for each content provider $i; i = 1, \ldots, m$,

$$U_i(Q_i^*, q_i^*, Q_{-i}^*, q_{-i}^*) \geq U_i(Q_i, q_i, Q_{-i}^*, q_{-i}^*), \quad \forall (Q_i, q_i) \in \mathbb{K}_i, \tag{10}$$

where Q_{-i} denotes the content posted by all the providers except for i. Analogously, q_{-i} expresses the quality levels of all the providers' contents except for i.

Hence, according to the above definition, a Nash equilibrium is established if no provider can unilaterally improve upon his profit by choosing an alternative vector of views and quality level, given the contents posted and quality level decisions of the other providers.

Theorem 1 (Variational Inequality Formulation) *Let us assume that for each content provider i the profit function $U_i(Q, q)$ is concave with respect to the variables $(Q_{i1}, \ldots, Q_{in})$, and q_i, and is continuous and continuously differentiable. A view amount and quality level pattern (Q^*, q^*) is a Nash equilibrium if and only if $(Q^*, q^*) \in \mathbb{K}$ is a solution of the variational inequality*

$$-\sum_{i=1}^{m}\sum_{j=1}^{n} \frac{\partial U_i(Q^*, q^*)}{\partial Q_{ij}} \times \left(Q_{ij} - Q_{ij}^* \right) - \sum_{i=1}^{m} \frac{\partial U_i(Q^*, q^*)}{\partial q_i} \times \left(q_i - q_i^* \right) \geq 0,$$

$$\forall (Q, q) \in \mathbb{K}. \tag{11}$$

namely, if it satisfies the variational inequality

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\left[c_i + \frac{\partial f_i(Q^*, q_i^*)}{\partial Q_{ij}} - p_i - F_j(Q^*, q^*) - \sum_{k=1}^{n}\frac{\partial F_k(Q^*, q^*)}{\partial Q_{ij}}\cdot Q_{ik}^*\right]\times(Q_{ij} - Q_{ij}^*)$$
$$+\sum_{i=1}^{m}\left[\frac{\partial f_i(Q^*, q_i^*)}{\partial q_i} - \sum_{k=1}^{n}\frac{\partial F_k(Q^*, q^*)}{\partial q_i}\cdot Q_{ik}^*\right]\times(q_i - q_i^*) \geq 0,$$
$$\forall (Q, q) \in \mathbb{K}. \tag{12}$$

Problem (11) admits a solution since the classical existence theorem, which requires that the set $\mathbb{K}$ is closed, convex, and bounded and the operator is continuous, is satisfied (see [8, 10]).

We note that the quantity $\frac{\partial U_i(Q^*, q^*)}{\partial Q_{ij}}$ represents the marginal profit with respect to the amount of views, and $\frac{\partial U_i(Q^*, q^*)}{\partial q_i}$ is the marginal profit with respect to quality levels.

3 Lagrange Multipliers and Nash Equilibria

In this section, we apply the notion of Lagrange function to present an alternative formulation of Nash equilibria which allows us to interpret the strategic decisions in terms of Lagrange multipliers. The strategy set $\mathbb{K}_i$ of each provider i; $i = 1, \ldots, m$, can be written as

$$\mathbb{K}_i = \left\{(Q_i, q_i) \in \mathbb{R}^{n+1} : -Q_{ij} \leq -\underline{Q}_{ij};\, Q_{ij} \leq \overline{Q}_{ij}, \forall j; -q_i \leq -1; q_i \leq 5\right\}. \tag{13}$$

We assume that each provider i minimizes the value of the loss function $-U_i$. Thus, we can introduce the Lagrange function for $i = 1, \ldots, m$

$$L_i(Q, q, \lambda_{ij}^1, \lambda_{ij}^2, \mu_i^1, \mu_i^2) = -U_i(Q, q) + \sum_{j=1}^{n}\lambda_{ij}^1(-Q_{ij} + \underline{Q}_{ij})$$
$$+\sum_{j=1}^{n}\lambda_{ij}^2(Q_{ij} - \overline{Q}_{ij}) + \mu_i^1(-q_i + 1) + \mu_i^2(q_i - 5),$$

where $(Q, q) \in \mathbb{R}^{mn+n}$, $\lambda^1, \lambda^2 \in \mathbb{R}_+^{mn+n}$, $\mu^1, \mu^2 \in \mathbb{R}_+^m$. It results (see, for instance, [15]):

Theorem 2 *Let us assume that for each content provider i the profit function $U_i(Q, q)$ is differentiable in $(Q^*, q^*) \in \mathbb{K}$. The strategy profile $(Q^*, q^*) \in \mathbb{K}$ is*

a Nash equilibrium if and only if there are Lagrange multipliers $\lambda_{ij}^1, \lambda_{ij}^2 \geq 0$*, for all* i, j*, and* $\mu_i^1, \mu_i^2 \geq 0$*, for all* i*, such that the following conditions are verified*

$$\frac{\partial L_i(Q^*, q^*, \lambda_{ij}^1, \lambda_{ij}^2, \mu_i^1, \mu_i^2)}{\partial Q_{ij}} = 0, \quad \forall i, j, \tag{14}$$

$$\frac{\partial L_i(Q^*, q^*, \lambda_{ij}^1, \lambda_{ij}^2, \mu_i^1, \mu_i^2)}{\partial q_i} = 0, \quad \forall i, \tag{15}$$

$$\lambda_{ij}^1(-Q_{ij}^* + \underline{Q}_{ij}) = 0, \quad \lambda_{ij}^2(Q_{ij}^* - \overline{Q}_{ij}), \quad \forall i, j, \tag{16}$$

$$\mu_i^1(-q_i^* + 1) = 0, \quad \mu_i^2(q_i^* - 5) = 0, \quad \forall i. \tag{17}$$

We now discuss the interpretation of conditions (14)–(17). Lagrange multipliers λ_{ij}^1, λ_{ij}^2, μ_i^1 and μ_i^2 regulate the whole content diffusion system. In particular, λ_{ij}^1, and λ_{ij}^2 represent control variables on the amount of views; whereas μ_i^1 and μ_i^2 are control variables on quality levels.

From (14), we obtain

$$-\frac{\partial U_i(Q^*, q^*)}{\partial Q_{ij}} - \lambda_{ij}^1 + \lambda_{ij}^2 = 0, \quad i = 1, \ldots, m; j = 1, \ldots, n.$$

Thus, if $\underline{Q}_{ij} < Q_{ij}^* < \overline{Q}_{ij}$, if follows that

$$-\frac{\partial U_i(Q^*, q^*)}{\partial Q_{ij}} = c_i + \frac{\partial f_i(Q^*, q_i^*)}{\partial Q_{ij}} - p_i - F_j(Q^*, q^*) - \sum_{k=1}^{n} \frac{\partial F_k(Q^*, q^*)}{\partial Q_{ij}} \cdot Q_{ik}^* = 0,$$

and marginal costs equal marginal revenues.

If $Q_{ij}^* = \underline{Q}_{ij}$, then $\lambda_{ij}^2 = 0$. Thus, we have

$$-\frac{\partial U_i(Q^*, q^*)}{\partial Q_{ij}} = \lambda_{ij}^1, \quad i = 1, \ldots, m; j = 1, \ldots, n,$$

namely, λ_{ij}^1 is equal to the opposite of the marginal profit with respect to views. If $\lambda_{ij}^1 > 0$, we conclude that the marginal utility with respect to views decreases.

If $Q_{ij}^* = \overline{Q}_{ij}$, then $\lambda_{ij}^1 = 0$. We find

$$-\frac{\partial U_i(Q^*, q^*)}{\partial Q_{ij}} = -\lambda_{ij}^2, \quad i = 1, \ldots, m; j = 1, \ldots, n,$$

and λ_{ij}^2 is equal to the marginal profit with respect to views. If $\lambda_{ij}^2 > 0$, then the marginal profit with respect to views increases.

Analogously, from (15), we obtain

$$-\frac{\partial U_i(Q^*, q^*)}{\partial q_i} - \mu_i^1 + \mu_i^2 = 0, \quad i = 1, \ldots, m.$$

Thus, if $1 < q_i^* < 5$, if follows that

$$-\frac{\partial U_i(Q^*, q^*)}{\partial q_i} = \frac{\partial f_i(Q^*, q_i^*)}{\partial q_i} - \sum_{k=1}^{n} \frac{\partial F_k(Q^*, q^*)}{\partial q_i} \cdot Q_{ik}^* = 0,$$

and the marginal cost and marginal revenue with respect to quality levels are balanced.

If $q_i^* = 1$, then $\mu_i^2 = 0$. We have

$$-\frac{\partial U_i(Q^*, q^*)}{\partial q_i} = \mu_i^1, \quad i = 1, \ldots, m,$$

and μ_i^1 is equal to the opposite of the marginal profit with respect to quality levels. If $\mu_i^1 > 0$, then the marginal profit with respect to quality levels decreases.

If $q_i^* = 5$, then $\mu_i^1 = 0$. We find

$$-\frac{\partial U_i(Q^*, q^*)}{\partial q_i} = -\mu_i^2, \quad i = 1, \ldots, m,$$

namely, μ_i^2 is equal to the marginal profit with respect to quality levels. If $\mu_i^2 > 0$, then the marginal profit with respect to quality levels increases.

Lagrange multipliers associated with model constraints are then valuable tools to analyze the online content diffusion.

4 A Numerical Example

The video content sharing platform YouTube is the world's second biggest search engine for more than 1, 8 billion people registered on the site to watch more than 1 billion hours of videos daily. Launched back in 2005, YouTube offers a massive collection of 1,300,000,000 videos, with more than 300 h of HD quality video being uploaded every 60 s. According to third party estimates, in 2015 YouTube was generating 8 billion dollars; in 2010 the company's annual advertising revenue estimate was only 1 billion dollars.

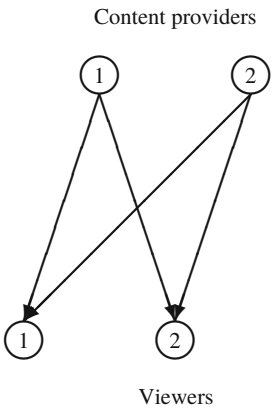


Fig. 2 A Youtube platform model

The major structure unit that YouTube is built on is a channel. There are hundreds of thousands channels; some have very few subscribers and some are very popular. A large number of subscribers allows content providers to monetize the volume of traffic that their videos generates.

The revenue for content's owner is based on the cost per mille (CPM) system, that assigns an advertisement cost per one thousand views. Therefore, the advertiser has to pay one dollar each time the advertisement reaches a thousand views. We note that the revenue goes to YouTube, not directly to the content creator. In fact, YouTube takes the 45% of the CPM.

We now apply our theoretical achievements to analyze the YouTube platform. We consider a population of users divided into social groups, each having a different characteristic according to a certain criterion (for instance, hobbies, age, education, etc...). Therefore, viewers of the same group are aggregated together and represented as a single viewer.

We consider two content providers and two groups of aggregated viewers; see Fig. 2.

The production cost functions are:

$$f_1(Q, q_1) = (Q_{11} + Q_{12})^2 + Q_{21} + Q_{22} + 2q_1^2,$$

$$f_2(Q, q_2) = 0.5(Q_{21} + Q_{22})^2 + 3(Q_{11} + Q_{12}) + q_2^2.$$

For each provider, the coefficients of the cost functions and the revenue functions are:

$$p_1 = 3, \qquad p_2 = 5,$$

$$c_1 = 1.35, \qquad c_2 = 2.25$$

The reputation functions for each provider are:

$$F_1(Q,q) = -(Q_{11}+Q_{21}) + q_1 + 0.2q_2 + \frac{q_1+q_2}{2} + 4,$$

$$F_2(Q,q) = -(Q_{12}+Q_{22}) + q_1 + q_2 + \frac{q_1+q_2}{2} + 8.$$

The profit function of content provider 1 is:

$$U_1(Q,q) = F_1(Q,q)\cdot Q_{11} + F_2(Q,q)\cdot Q_{12} + (-c_1+p_1)(Q_{11}+Q_{12}) - f_1(Q,q_1),$$

whereas the profit function of content provider 2 is:

$$U_2(Q,q) = F_1(Q,q)\cdot Q_{21} + F_2(Q,q)\cdot Q_{22} + (-c_2+p_2)(Q_{21}+Q_{22}) - f_2(Q,q_2).$$

We consider Q_{ij} in the order of tens of thousand. Moreover, following the policy of YouTube, the cost parameter c_i is the 45% of the revenue parameter p_i, for $i = 1, 2$.

We note that the profit functions are concave and continuous on a compact set; hence the existence of solutions to the associated variational inequality is ensured.

Following Theorem 2, we should study all possible combinations of active and non-active constraints; however, we focus only on the case in which all the Lagrange multipliers are null. Thus, we find the exact solutions in tens of thousand:

$$Q_{11}^* = 0.594586,\ Q_{12}^* = 3.06044,\ Q_{21}^* = 2.17959,\ Q_{22}^* = 4.64544,$$

$$q_1^* = 1.37063,\ q_2^* = 4.24694.$$

The total profit amounts to 124,967$ for the first provider, and 206,198$ for the second one.

We note that only the 15% of the views counts as a profit from advertisement strategies, because the only views that make content provider to earn money are those in which viewers watch an advertisement for at least 30 s (or half the ad for a very short video). Hence, the advertisement profit every thousand views, namely, the difference between the advertisement cost and the revenue for hosting advertisements, is approximately 9.04618$ for the first provider, and 28.1533$ for the second one. We notice that the second content is much more appreciated than the first one ($s_1 = 36{,}550$, $s_2 = 68{,}250$), and this depends on the higher quality of the content. In fact, the quality of the first video is almost satisfactory ($q_1^* = 1.37063$), but the second one is a very good content ($q_2^* = 4.24694$). The quality is the key to increase the number of monetized views. This can be realized using proper keywords in titles and descriptions of the videos, making the content as interesting to the viewers as possible, and eliminating every factor that could make the viewers bored with videos.

5 Conclusions

In this paper, we present a network game theory model consisting of online content providers and viewers. Providers compete non-cooperatively for the diffusion of their online contents, each one maximizing the profit until a Nash equilibrium is achieved. Viewers express their preferences through their feedback functions, that may depend upon the entire volume of views and on the average quality level. We derive the variational inequality formulation of the governing equilibrium conditions. Moreover, we present an alternative formulation of Nash equilibria which allows us to interpret the strategic decisions in terms of Lagrange multipliers. The results in this paper add to the growing literature on modeling and analysis of UGC platforms using a game theory approach.

Future research may include the study of the continuous-time evolution of the model, the presence of shared constraints, and the formulation as a Generalized Nash equilibrium problem.

Acknowledgements The research of the first author was partially supported by the research project "Modelli Matematici nell'Insegnamento-Apprendimento della Matematica" DMI, University of Catania. This support is gratefully acknowledged.

References

1. Altman, E., De Pellegrini, F., Al-Azouzi, R., Miorandi. D., Jimenez, T.: Emergence of equilibria from individual strategies in online content diffusion. In: Proceedings of IEEE INFORCOM NetSciCOmm, Turin, Italy (2013)
2. Altman, E., Jain, A., Shimkin, N., Touati, C.: Dynamic games for analyzing competition in the Internet and in on-line social networks. In: Lasaulce, S., et al. (eds.) Network Games, Control, and Optimization, pp. 11–22. Springer, Berlin (2017)
3. Cha, M., Kwak, H., Rodriguez, P., Ahn, Y.-Y., Moon, S.: I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In: Proceeding of ACM IMC, San Diego, California, USA, October 24–26 2007, pp. 1–14 (2007)
4. Cha, M., Kwak, H., Rodriguez, P., Ahn, Y.-Y., Moon, S.: Analyzing the video popularity characteristics of large-scale user generated content systems. IEEE/ACM Trans. Networking **17**(5), 1357–1370 (2009)
5. Colajanni, G., Daniele, P., Giuffré, S., Nagurney, A.: Cybersecurity investments with nonlinear budget constraints and conservation laws: variational equilibrium, marginal expected utilities, and Lagrange multipliers. Int. Trans. Oper. Res. **25**, 1443–1464 (2018)
6. Daniele, P., Giuffrè, S., Lorino, M.: Functional inequalities, regularity and computation of the deficit and surplus variables in the financial equilibrium problem. J. Glob. Optim. **65**(3), 575–596 (2016)
7. De Pellegrini, F., Reigers, A., Altman, E.: Differential games of competition in online content diffusion. In: 2014 IFIP Networking Conference, pp. 1–9 (2014)
8. Facchinei, F., Pang, J.S.: Finite-Dimensional Variational Inequalities and Complementarity Problems, vol. I. Springer, New York (2003)
9. Mirabella, C., Scrimali, L.: Cooperation in pollution control problems via evolutionary variational inequalities. J. Glob. Optim. **70**, 455–476 (2018)
10. Nagurney, A.: Network Economics: A Variational Inequality Approach. Kluwer, Boston (1993)

11. Nash, J.F.: Equilibrium points in n-person games. Proc. Natl. Acad. Sci. **36**, 48–49 (1950)
12. Nash, J.F.: Non-cooperative games. Ann. Math. **54**, 286–295 (1951)
13. Oggioni, G., Smeers, Y., Allevi Em Schaible, S.: A generalized Nash equilibrium model of market coupling in the European power system. Netw. Spat. Econ. **12**(4), 503–560 (2012)
14. Saberi, S., Nagurney, A., Wolf, T.: A network economic game theory model of a service-oriented Internet with price and quality competition in both content and network provision. Serv. Sci. **6**(4), 229–250 (2014)
15. Ungureanu, V.: Pareto-Nash-Stackelberg games and control theory. In: Smart Innovation, Systems and Technologies, vol. 89. Springer, Berlin (2018)

A Variational Formulation for a Human Migration Problem

Giorgia Cappello and Patrizia Daniele

Abstract Many social and economical factors affect the dynamics of human populations, such as poverty, violence, war, dictatorships, persecutions, tsunamis, floods, earthquakes, family reunification as well as economic and educational possibilities or a job. In this paper, we consider a network based model where the aim of each migration class is to maximize the attractiveness of the origin country and we prove that the optimization model can be formulated in terms of a Nash equilibrium problem and a variational inequality. Finally, some numerical results applied to the human migration from Africa to Europe are presented and analyzed.

Keywords Human migration · Nonlinear optimization · Network models

1 Introduction

Human migration is the movement that people do from one place to another with the intention of settling temporarily or permanently in the new location. It typically involves movements over long distances and from one country or region to another. Many social and economical factors affect the dynamics of human populations, such as poverty, violence, war, dictatorships, persecutions, oppression, genocide, ethnic cleansing, climate change, tsunamis, floods, earthquakes, famines, family reunification as well as economic and educational possibilities or a job.

According to the International Migration Report (2017), a biennial publication of the Department of Economic and Social Affairs, there are now an estimated 258 million people living in a country different from that of birth, with an increase of 49% since 2000, which means that 3.4% of the world's inhabitants today are international migrants.

G. Cappello · P. Daniele (✉)
Department of Mathematics and Computer Science, University of Catania, Catania, Italy
e-mail: giorgia.cappello@unict.it; daniele@dmi.unict.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_17

During the 2018 Mediterranean arrivals were 141,475, with more than 2000 dead and missing people; these data include sea arrivals at Italy, Cyprus and Malta and both sea and land arrivals at Greece and Spain, mostly from the north of Africa. From 2018 until January 2019, 17% of arrivals by sea were registered in Italy, compared to 69% in 2017 (UNHCR).

The increasing role of migration in the social, economic and demographic development of countries, regions and the whole of the world, especially in the Mediterranean Basin which has become the theatre of a humanitarian crisis that has challenged the collective leadership around the sea, is becoming more and more evident, thus stimulating interest in mathematical modeling of migration. In the last decades many papers have been devoted to this topic.

In [2] Cojocaru presents an application of the double-layer dynamics theory for modelling dynamics of human migration problems reformulated as transportation networks problems.

In [1] the authors provide a general framework for analyzing migration at macro and micro levels. They also present an overview of the practical application for the models and establish a basis for policy-analysis.

In [3] Cui and Bai describe the evolution of population density and the spread of epidemics in population systems where the spatial movement of individuals depends only on the departure and arrival locations through some mathematical models.

In [9] the authors analyze the interaction of human migration and wealth distribution.

In [7] Nagurney introduces a network equilibrium model of human migration in the presence of movement costs and proves that, in the case of linear utility functions and fixed movement costs, the equilibrium conditions can be reformulated as the solution to an equivalent quadratic programming problem.

In [5] Kalashnykov and Kalashnykova examine the case where the conjectural variations coefficients may be not only constants, but also functions of the total population at the destination and of the group's fraction in it and establish the equivalence of the equilibrium to a solution of an appropriate variational inequality problem.

In this paper we consider a network based model where the aim of each migration class is to maximize the attractivenss of the origin country, which is given by the sum of its utility and its expected increment of utility value, with respect to the destination one. We prove that the optimization model can be formulated in terms of a Nash equilibrium and a variational inequality. Finally, some numerical results applied to the human migration from Africa to Europe are presented and analyzed.

2 The Mathematical Model

We present a model that consists of n locations and J classes of population. The n locations are the nodes from and in which the different classes of population choose to go (Fig. 1).

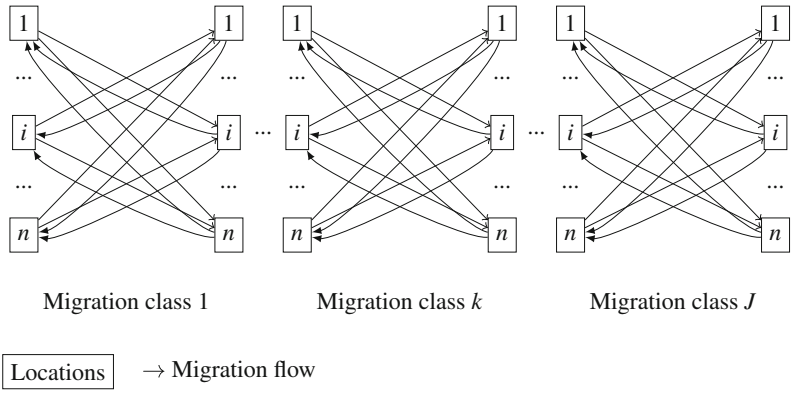


Fig. 1 Multiclass migration network

We suppose that at each location i, $i = 1, \ldots, n$, there is an initial fixed population of the general class k, denoted by $\bar{p}_i^k$. The population of class k at location i is determined by the initial population of class k at location i plus the migration flow, f_{ij}^k, into i of that population, minus the migration flow, f_{ji}^k, out of i. Indeed, the conservation of flow equations, given for each class k and each location i, are given as follows:

$$p_i^k = \bar{p}_i^k - \sum_{\substack{j=1 \\ j \neq i}}^{n} f_{ij}^k + \sum_{\substack{j=1 \\ j \neq i}}^{n} f_{ji}^k \quad \forall i = 1, \ldots, n, \quad \forall k = 1, \ldots, J. \tag{1}$$

We group the populations k in each location i into the vector p^k and the migration flows of population k from each origin node i to each destination node j into the vector f^k.

Each location i is characterized by

- a destination utility function, v_i^k, that is indicative of the attractiveness of that location intended as an idealization of the opportunities that this node can offer as perceived by the migration class k;
- an origin utility function, u_i^k, that is indicative of the attractiveness of that location intended as the awareness of the opportunities that this node can offer as perceived by the migration class k.

Both groups of functions, u_i^k and v_i^k, depend on the population p^k.

Let us introduce $w_{ij}^{k+}(p^k, f^k) \geq 0$, which denotes the influence coefficient taken into account by an individual of the migration class k moving from node i to j. It is the expected variation of the population at node j after a migratory flow, as perceived by the migration class k. Similarly, we introduce $w_{ij}^{k-}(p^k, f^k) \geq 0$, that is the expected rate of change of the total population at node i, as perceived by the migration class k.

After the potential movement of migrants from location i to location j, each person of class k expects a variation of the utility function value at j:

$$f_{ij}^k w_{ij}^{k+}(p^k, f^k) \frac{\partial v_j^k(p^k)}{\partial p_j^k},$$

and a variation of the utility function value at i:

$$-f_{ij}^k w_{ij}^{k-}(p^k, f^k) \frac{\partial u_i^k(p^k)}{\partial p_i^k},$$

where the negative sign denotes the loss for the migration class k, when choosing to abandon the origin node.

Both groups of such variations are assumed concave.

In addition, we denote by c_{ij}^k the movement cost from i to j for the population class k, where

$$c_{ij}^k = c_{ij}^k(f^k), \quad \forall i, j = 1, \ldots, n, \ j \neq i, \ \forall k = 1, \ldots, J.$$

Such costs are assumed to be convex and continuously differentiable. In Table 1 we introduce all the functions, parameters and variables used in the model.

In order to reduce the migration phenomenon and to encourage people to remain in their own country the attractiveness in i (which is given by the sum of the utility in i and its expected increment of utility value) must exceed the sum between the attractiveness in j (which is given by the sum of the utility in j and its expected increment of utility value) and the transportation costs from location i to location j. Hence, in our model, the aim of each migration class k, $k = 1, \ldots, J$, in each

Table 1 Functions, parameters and variables of the model

Symbol	Definition
$\bar{p}_i^k$	Initial population of class k in location i
p_i^k	Population at location i of class k
f_{ij}^k	Migration flow from i to j of class k, with $i \neq j$
$v_i^k(p^k)$	Destination utility function of any location i as perceived by the class k
$u_i^k(p^k)$	Origin utility function of any location i as perceived by the class k
$c_{ij}^k(f^k)$	Movement cost from i to j for the class k
$w_{ij}^{k+}(p^k, f^k)$, $w_{ij}^{k-}(p^k, f^k)$	Influence coefficient

departing node i, $i = 1, \ldots, n$ is to maximize its net utility, namely the following difference:

$$\max_{(p^k, f^k) \in \mathbb{K}^k} U^k(p^k, f^k) = \max_{(p^k, f^k) \in \mathbb{K}^k} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \left(u_i^k(p^k) - f_{ij}^k w_{ij}^{k-}(p^k, f^k) \frac{\partial u_i^k(p^k)}{\partial p_i^k} \right.$$

$$\left. - c_{ij}^k(f^k) - v_j^k(p^k) - f_{ij}^k w_{ij}^{k+}(p^k, f^k) \frac{\partial v_j^k(p^k)}{\partial p_j^k} \right) \quad (2)$$

where

$$\mathbb{K}^k = \left\{ (p^k, f^k) \in \mathbb{R}^{n+n(n-1)} : p_i^k = \bar{p}_i^k - \sum_{\substack{j=1 \\ j \neq i}}^{n} f_{ij}^k - \sum_{\substack{j=1 \\ j \neq i}}^{n} f_{ji}^k, \ \forall i = 1, \ldots, n; \right.$$

$$\left. p_i^k \geq 0, \ f_{ij}^k \geq 0, \ \forall i, \ j = 1, \ldots, n, \ j \neq i; \sum_{\substack{j=1 \\ j \neq i}}^{n} f_{ij}^k \leq \bar{p}_i^k, \ \forall i = 1, \ldots, n \right\}. \quad (3)$$

We also assume that the migration classes compete in a noncooperative manner, so that each maximizes its utility, given the actions of the other classes.

Under the imposed assumptions, the objective function in (2) in concave and continuously differentiable. We also define

$$\mathbb{K} = \left\{ (p, f) \in \mathbb{R}^{Jn+Jn(n-1)} : p_i^k = \bar{p}_i^k - \sum_{\substack{j=1 \\ j \neq i}}^{n} f_{ij}^k - \sum_{\substack{j=1 \\ j \neq i}}^{n} f_{ji}^k, \ \forall i = 1, \ldots, n, \ \forall k = 1, \ldots, J \right.$$

$$p_i^k \geq 0, \ f_{ij}^k \geq 0, \ \forall i, \ j = 1, \ldots, n, \ j \neq i, \ \forall k = 1, \ldots, J;$$

$$\left. \sum_{\substack{j=1 \\ j \neq i}}^{n} f_{ij}^k \leq \bar{p}_i^k, \ \forall i = 1, \ldots, n, \ \forall k = 1, \ldots, J \right\} = \prod_{k=1}^{J} \mathbb{K}^k \quad (4)$$

and the total utility as:

$$U(p, f) = \sum_{k=1}^{J} U^k(p^k, f^k) \quad \forall (p, f) \in \mathbb{K}.$$

Hence, the above game theory model, in which the migration classes compete noncooperatively, is a Nash equilibrium problem. Therefore, we can state the following definition.

Definition 1 A population and migration flow pattern $(p^*, f^*) \in \mathbb{K}$ is said to be a Nash equilibrium if for each migration class k

$$U(p^{k*}, f^{k*}, \hat{p}^{k*}, \hat{f}^{k*}) \geq U(p^k, f^k, \hat{p}^{k*}, \hat{f}^{k*}) \quad \forall (p^k, f^k) \in \mathbb{K}^k,$$

where

$$\hat{p}^{k*} = (p^{1*}, \ldots, p^{k-1*}, p^{k+1*}, \ldots, p^{J*}) \quad and \quad \hat{f}^{k*} = (f^{1*}, \ldots, f^{k-1*}, f^{k+1*}, \ldots, f^{J*}).$$

Hence, according to the above definition, a Nash equilibrium in established if no migration class can unilaterally improve its expected utility by choosing an alternative vector of population and migration flow.

The optimality conditions (2) for all migration classes $k, k = 1, \ldots, J$ simultaneously can be expressed by means of a variational inequality as follows (see [8]).

Theorem 1 (Variational Formulation) *Under the above assumptions,* $(p^*, f^*) \in \mathbb{K}$ *is an equilibrium according to Definition 1 if and only if it satisfies the following variational inequality:*

Find $(p^*, f^*) \in \mathbb{K}$ *such that:*

$$\sum_{l=1}^{n}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\sum_{k=1}^{J}\left(\frac{\partial v_j^k(p^{k*})}{\partial p_l^k} + f_{ij}^k\frac{\partial w_{ij}^{k+}(p^{k*}, f^{k*})}{\partial p_l^k}\frac{\partial v_j^k(p^{k*})}{\partial p_j^k} + f_{ij}^k w_{ij}^{k+}(p^{k*}, f^{k*})\frac{\partial^2 v_j^k(p^{k*})}{\partial p_j^k \partial p_l^k}\right.$$

$$\left. - \frac{\partial u_i^k(p^{k*})}{\partial p_l^k} + f_{ij}^k\frac{\partial w_{ij}^{k-}(p^{k*}, f^{k*})}{\partial p_l^k}\frac{\partial u_i^k(p^{k*})}{\partial p_i^k} + f_{ij}^k w_{ij}^{k-}(p^{k*}, f^{k*})\frac{\partial^2 u_i^k}{\partial p_i^k \partial p_l^k}\right)(p_l^k - p_l^{k*})$$

$$+\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\sum_{k=1}^{J}\left(w_{ij}^{k+}(p^{k*}, f^{k*})\frac{\partial v_j^k(p^*)}{\partial p_j^k} + f_{ij}^{k*}\frac{\partial w_{ij}^{k+}(p^{k*}, f^{k*})}{\partial f_{ij}^k}\frac{\partial v_j^k(p^{k*})}{\partial p_j^k} + \frac{\partial c_{ij}^k(f^{k*})}{\partial f_{ij}^k}\right.$$

$$\left. + w_{ij}^{k-}(p^{k*}, f^{k*})\frac{\partial u_i^k(p^*)}{\partial p_i^k} + f_{ij}^k\frac{\partial w_{ij}^{k-}(p^{k*}, f^{k*})}{\partial f_{ij}^k}\frac{\partial u_i^k(p^{k*})}{\partial p_i^k}\right)(f_{ij}^k - f_{ij}^{k*}) \geq 0,$$

$$\forall (p, f) \in \mathbb{K}. \tag{5}$$

3 Numerical Examples

In order to perform numerical experiments, we consider the flow of migrants from Africa through the Mediterranean sea to Italy during 2018. The network, as we can see in Fig. 2, is composed by six nodes: the first three ones are the origin nodes (Tunisia, Eritrea and Sudan, with a percentage of 23, 8, 15.0, and 7.3% of departures, respectively), the fourth node is Italy, which is the destination chosen by most of the migrants (see [6]), the last two nodes are Germany and France, with a percentage of 31 and 15%, respectively, of arrivals, and represent the countries where most migrants have been relocated from Italy.[4].

In order to better understand the real migration flows, we collected the necessary data, referred to 2018, about population, average movement, measures of the quality of life, transportation costs for each node of the network. We assume that the

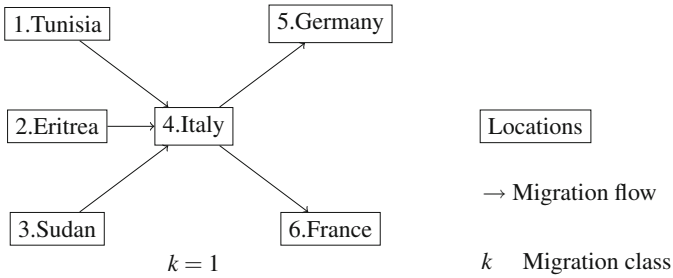


Fig. 2 Multiclass migration network for the first numerical example

migration class is only one ($k = 1$), and it represents the African population that moves from its continent to Europe.

We assume also that the examined migration class is interested in evaluating the displacement between the nodes 1–4; 2–4; 3–4; 4–5; 4–6; and this is because, starting from real data, the other displacements never occur due to political or economic issues, as we can see in Fig. 2.

Now we formulate an illustrative numerical example considering the following data:

Origin utility function	$u_1 = -0.8(p_1^1)^2 + 2p_1^1$
	$u_2 = -0.7(p_2^1)^2 + 3p_2^1$
	$u_3 = -0.85(p_3^1)^2 + 2.5p_3^1$
	$u_4 = -0.5p_4^1$
Destination utility function	$v_4 = -0.5p_4^1$
	$v_5 = 0.3(p_5^1)^2 - 6p_5^1$
	$v_6 = 0.35(p_6^1)^2 - 7p_6^1$
Movement cost	$c_{14} = 3f_{14}^1$
	$c_{24} = 4f_{24}^1$
	$c_{34} = 5f_{34}^1$
	$c_{45} = 0.25f_{45}^1$
	$c_{46} = 0.25f_{46}^1$.
Initial population	$\bar{p}_1 = 55$
	$\bar{p}_2^1 = 45$
	$\bar{p}_3^1 = 40$
	$\bar{p}_4^1 = 35$
	$\bar{p}_5^1 = 50$
	$\bar{p}_6^1 = 52$

Then, optimality problem (2) becomes:

$$\max_{(p^1,f^1)\in\mathbb{K}^1} U^1(p^1,f^1)$$

$$= \max_{(p^1,f^1)\in\mathbb{K}^1} \left\{ u_1(p^1) - f_{14}^1 w_{14}^-(p^1,f^1)\frac{\partial u_1(p^1)}{\partial p_1^1} - c_{14}(f^1) - v_4(p^1) - f_{14}^1 w_{14}^+(p^1,f^1)\frac{\partial v_4(p^1)}{\partial p_4^1} + \right.$$

$$+u_2(p^1) - f_{24}^1 w_{24}^-(p^1,f^1)\frac{\partial u_2(p^1)}{\partial p_2^1} - c_{24}(f^1) - v_4(p^1) - f_{24}^1 w_{24}^+(p^1,f^1)\frac{\partial v_4(p^1)}{\partial p_4^1} +$$

$$+u_3(p^1) - f_{34}^1 w_{34}^-(p^1,f^1)\frac{\partial u_3(p^1)}{\partial p_3^1} - c_{34}(f) - v_4(p^1) - f_{34}^1 w_{34}^+(p^1,f^1)\frac{\partial v_4(p^1)}{\partial p_4^1} +$$

$$+u_4(p^1) - f_{45}^1 w_{45}^-(p^1,f^1)\frac{\partial u_4(p^1)}{\partial p_4^1} - c_{45}(f^1) - v_5(p^1) - f_{45}^1 w_{45}^+(p^1,f^1)\frac{\partial v_5(p^1)}{\partial p_5^1} +$$

$$\left. +u_4(p^1) - f_{46}^1 w_{46}^-(p^1,f^1)\frac{\partial v_4(p^1)}{\partial p_4^1} - c_{46}(f^1) - u_6(p^1) - f_{46}^1 w_{46}^+(p^1,f^1)\frac{\partial u_6(p^1)}{\partial p_6^1} \right\}$$

where

$$\mathbb{K}^1 = \mathbb{K} = \left\{ (p_1^1, p_2^1, p_3^1, p_4^1, p_5^1, p_6^1, f_{14}^1, f_{24}^1, f_{34}^1, f_{45}^1, f_{46}^1) \in \mathbb{R}^{11}; \right.$$

$$p_1^1, p_2^1, p_3^1, p_4^1, p_5^1, p_6^1, f_{14}^1, f_{24}^1, f_{34}^1, f_{45}^1, f_{46}^1 \geq 0$$

$$p_1^1 = 55 - f_{14}^1;\ \ p_2^1 = 45 - f_{24}^1;\ \ p_3^1 = 40 - f_{34}^1;$$

$$p_4^1 = 35 + f_{14}^1 + f_{24}^1 + f_{34}^1 - f_{45}^1 - f_{46}^1;$$

$$\left. p_5^1 = 50 + f_{45}^1;\ \ p_6^1 = 52 + f_{46}^1\ \ f_{14}^1 \leq 55;\ \ f_{34}^1 \leq 40;\ \ f_{45}^1 + f_{46}^1 \leq 35 \right\}, \quad (6)$$

and the associated variational inequality (5) becomes:

$$\begin{aligned}
&\textit{Find}(p^*, f^*) \in \mathbb{K} \quad \textit{such that}: \\
&(1.6p_1^1 - 2 - 1.6f_{14}^1 w_{14}^-) \times (p_1^1 - p_1^{*1}) + (1.4p_2^1 - 3 - 1.4f_{24}^1 w_{24}^-) \times (p_2^1 - p_2^{*1}) \\
&+(1.7p_3^1 - 2.5 - 1.7f_{34}^1 w_{34}^-) \times (p_3^1 - p_3^{*1}) - 0.5 \times (p_4 - p_4^*) \\
&+(0.6p_5^1 - 6 + 0.6f_{45}^1 w_{45}^+) \times (p_5^1 - p_5^{*1}) + (0.7p_6^1 - 7 + 0.7f_{46}^1 w_{46}^+) \times (p_6^1 - p_6^{*1}) \\
&+(w_{14}^-(-1.6p_1^1 + 2) + 3 - 0.5w_{14}^+) \times (f_{14}^1 - f_{14}^{*1}) \\
&+(w_{24}^-(-1.4p_2^1 + 3) + 4 - 0.5w_{24}^+) \times (f_{24}^1 - f_{24}^{*1}) \\
&+(w_{34}^-(-1.7p_3^1 + 2.5) + 5 - 0.5w_{34}^+) \times (f_{34}^1 - f_{34}^{*1}) \\
&+(-0.5w_{45}^- + 0.25 + w_{45}^+(0.6p_5^1 - 6)) \times (f_{45}^1 - f_{45}^{*1}) \\
&+(-0.5w_{46}^+ + 0.25 + w_{46}^+(0.7p_6^1 - 7)) \times (f_{46}^1 - f_{46}^{*1}) \geq 0 \\
&\forall (p_1, p_2, p_3, p_4, p_5, p_6, f_{14}, f_{24}, f_{34}, f_{45}, f_{46}) \in \mathbb{K}.
\end{aligned} \quad (7)$$

It was solved in Matlab using the projection method. The computational time to obtain the optimal flows according to the terms $w_{ij}^{\pm}$ was 9.01 s. The machine used for the simulation is a 4 GB RAM Asus Intel (R) Core (TM) i5-3317U CPU@1.10 GHz.

The optimal flows as solution of (7) and for different values of $w_{ij}^{\pm}$ $i = 1, 2, 3, 4$; $j = 4, 5, 6$, are showed in Table 2. We observed that the optimal flows, from the poorest countries to the richest are high comparing with the initial population, even though the indices $w_{ij}^{\pm}$ let the migration class to conjecture an improvement of utility in the countries of departure and a worsening of the utility of the richer countries due to the migration.

Table 2 Optimal flows obtained from (7) for different values of $w_{ij}^{\pm}$

w_{ij}^{+}	w_{ij}^{-}	Optimal flows
$w_{14}^{+} = 0$	$w_{14}^{-} = 0$	$f_{14}^{1} = 5{,}218{,}706{,}322$
$w_{14}^{+} = 0$	$w_{14}^{-} = 0$	$f_{24}^{1} = 1{,}464{,}240{,}764$
$w_{34}^{+} = 0$	$w_{34}^{-} = 0$	$f_{34}^{1} = 358{,}821{,}929$
$w_{45}^{+} = 0$	$w_{45}^{-} = 0$	$f_{45}^{1} = 0$
$w_{46}^{+} = 0$	$w_{46}^{-} = 0$	$f_{46}^{1} = 0$
$w_{14}^{+} = 1$	$w_{14}^{-} = 1$	$f_{14}^{1} = 3{,}541{,}646{,}178$
$w_{14}^{+} = 1$	$w_{14}^{-} = 1$	$f_{24}^{1} = 192{,}855{,}584$
$w_{34}^{+} = 1$	$w_{34}^{-} = 1$	$f_{34}^{1} = 2{,}490{,}181{,}913$
$w_{45}^{+} = 1$	$w_{45}^{-} = 1$	$f_{45}^{1} = 0$
$w_{46}^{+} = 1$	$w_{46}^{-} = 1$	$f_{46}^{1} = 0$
$w_{14}^{+} = 0.5$	$w_{14}^{-} = 1$	$f_{14}^{1} = 3{,}536{,}437{,}875$
$w_{14}^{+} = 0.5$	$w_{14}^{-} = 1$	$f_{24}^{1} = 1{,}922{,}603{,}507$
$w_{34}^{+} = 0.5$	$w_{34}^{-} = 1$	$f_{34}^{1} = 248{,}527{,}998$
$w_{45}^{+} = 0.2$	$w_{45}^{-} = 0.2$	$f_{45}^{1} = 0$
$w_{46}^{+} = 0.2$	$w_{46}^{-} = 0.2$	$f_{46}^{1} = 0$
$w_{14}^{+} = 0.1$	$w_{14}^{-} = 0.8$	$f_{14}^{1} = 5{,}304{,}662{,}606$
$w_{14}^{+} = 0.1$	$w_{14}^{-} = 0.8$	$f_{24}^{1} = 1{,}812{,}476{,}595$
$w_{34}^{+} = 0.1$	$w_{34}^{-} = 0.8$	$f_{34}^{1} = 2{,}764{,}697{,}497$
$w_{45}^{+} = 0.2$	$w_{45}^{-} = 0$	$f_{45}^{1} = 0$
$w_{46}^{+} = 0.2$	$w_{46}^{-} = 0$	$f_{46}^{1} = 0$
$w_{14}^{+} = 0.2$	$w_{14}^{-} = 1$	$f_{14}^{1} = 3{,}533{,}312{,}893$
$w_{14}^{+} = 0.2$	$w_{14}^{-} = 1$	$f_{24}^{1} = 1{,}919{,}032{,}108$
$w_{34}^{+} = 0.2$	$w_{34}^{-} = 1$	$f_{34}^{1} = 248{,}233{,}882$
$w_{45}^{+} = 0.1$	$w_{45}^{-} = 0.3$	$f_{45}^{1} = 0$
$w_{46}^{+} = 0.1$	$w_{46}^{-} = 0.3$	$f_{46}^{1} = 0$
$w_{14}^{+} = 0$	$w_{14}^{-} = 1$	$f_{14}^{1} = 3{,}531{,}234{,}671$

(continued)

Table 2 (continued)

w_{ij}^{+}	w_{ij}^{-}	Optimal flows
$w_{14}^{+} = 0$	$w_{14}^{-} = 1$	$f_{24}^{1} = 2{,}053{,}554{,}511$
$w_{34}^{+} = 0$	$w_{34}^{-} = 1$	$f_{34}^{1} = 2{,}480{,}381{,}573$
$w_{45}^{+} = 0$	$w_{45}^{-} = 1$	$f_{45}^{1} = 0$
$w_{46}^{+} = 0$	$w_{46}^{-} = 1$	$f_{46}^{1} = 0$
$w_{14}^{+} = 0.25$	$w_{14}^{-} = 1$	$f_{14}^{1} = 3{,}533{,}833{,}724$
$w_{14}^{+} = 0.25$	$w_{14}^{-} = 1$	$f_{24}^{1} = 1{,}919{,}627{,}341$
$w_{34}^{+} = 0.25$	$w_{34}^{-} = 1$	$f_{34}^{1} = 2{,}482{,}829{,}013$
$w_{45}^{+} = 0.5$	$w_{45}^{-} = 1$	$f_{45}^{1} = 0$
$w_{46}^{+} = 0.5$	$w_{46}^{-} = 1$	$f_{46}^{1} = 0$
$w_{14}^{+} = 0.5$	$w_{14}^{-} = 1$	$f_{14}^{1} = 3{,}536{,}437{,}875$
$w_{14}^{+} = 0.5$	$w_{14}^{-} = 1$	$f_{24}^{1} = 1{,}922{,}603{,}507$
$w_{34}^{+} = 0.5$	$w_{34}^{-} = 1$	$f_{34}^{1} = 248{,}527{,}998$
$w_{45}^{+} = 0.25$	$w_{45}^{-} = 1$	$f_{45}^{1} = 0$
$w_{46}^{+} = 0.25$	$w_{46}^{-} = 1$	$f_{46}^{1} = 0$
$w_{14}^{+} = 0$	$w_{14}^{-} = 0.5$	$f_{14}^{1} = 3{,}953{,}106{,}448$
$w_{14}^{+} = 0$	$w_{14}^{-} = 0.5$	$f_{24}^{1} = 1{,}803{,}548{,}139$
$w_{34}^{+} = 0$	$w_{34}^{-} = 0.5$	$f_{34}^{1} = 2{,}757{,}344{,}578$
$w_{45}^{+} = 0$	$w_{45}^{-} = 0.2$	$f_{45}^{1} = 0$
$w_{46}^{+} = 0$	$w_{46}^{-} = 0.2$	$f_{46}^{1} = 0$

Acknowledgements The research of the authors was partially supported by the research project "Modelli Matematici nell'Insegnamento-Apprendimento della Matematica" DMI, University of Catania and by the research project PON SCN 00451 CLARA - CLoud plAtform and smart underground imaging for natural Risk Assessment, Smart Cities and Communities and Social Innovation. These supports are gratefully acknowledged.

References

1. Aleshkovski, I., Iontsev, V.: Mathematical models of migration. Comput. Sci. (2011). https://pdfs.semanticscholar.org/ad24/eb63f6358189aea76d4ec71c18296fe5b922.pdf
2. Cojocaru, M.G.: Double-layer dynamics theory and human migration after catastrophic events. In: Nonlinear Analysis with applications in Economics. Energy and Transportation, pp. 65–86. Bergamo University Press, Bergamo (2007)
3. Cui, S., Bai, M.: Mathematical Analysis of Population Migration and Its Effects to Spread of Epidemics (2014), arXiv:1403.5351 [math.FA]
4. International Organization for Migration's Global Migration Data Analysis Centre. https://gmdac.iom.int/
5. Kalashnykov, V., Kalashnykova, N.: Simulation of a conjectural variations equilibrium in a human migration model. Int. J. Simul. Syst. Sci. Technol. **7**(9), 26–39 (2006)
6. Mediterranean Situation. https://data2.unhcr.org/en/situations/mediterranean

7. Nagurney, A.: A network model of migration equilibrium with movement costs. Math. Comput. Model. **13**(5), 79–88 (1990)
8. Nagurney, A.: Network Economics: A Variational Inequality Approach, 2nd and revised edn. Kluwer, Boston (1999)
9. Volpert, V., Petrovskiic, S., Zincenkoc, A.: Interaction of human migration and wealth distribution. Nonlinear Anal. **159**, 408–423 (2017). Hal archives-ouvertes.fr

Flying Safely by Bilevel Programming

Martina Cerulli, Claudia D'Ambrosio, and Leo Liberti

Abstract Preventing aircraft from getting too close to each other is an essential element of safety of the air transportation industry, which becomes ever more important as the air traffic increases. The problem consists in enforcing a minimum distance threshold between flying aircraft, which naturally results in a bilevel formulation with a lower-level subproblem for each pair of aircraft. We propose two single-level reformulations, present a cut generation algorithm which directly solves the bilevel formulation and discuss comparative computational results.

Keywords Bilevel programming · Aircraft · Deconfliction

1 Introduction

In Air Traffic Management, the act of avoiding that two aircraft might collide is called *aircraft deconfliction*. More in general, it describes the set of methodologies for detecting and resolving aircraft conflicts. Aircraft are said to be potentially *in conflict* if their relative distance is smaller than a given safety threshold. Despite the importance of this kind of control, it is still widely performed manually on the ground by air traffic controllers, who essentially monitor the air traffic in a given, limited space on a radar screen, giving instructions to the pilots. With the increase of aircraft automation, there comes a need for integrating human ground control by algorithmic means.

The two crucial parameters of an aircraft flight that come into play in aircraft deconfliction are the trajectory and the speed. Typically, air traffic controllers change the trajectory, or use heading angle changes (HAC) in order to solve potential conflicts. In this paper, we focus instead on changing the aircraft speeds (which can

M. Cerulli (✉) · C. D'Ambrosio · L. Liberti
CNRS LIX Ecole Polytechnique, Institut Polytechnique de Paris, Palaiseau, France
e-mail: mcerulli@lix.polytechnique.fr; dambrosio@lix.polytechnique.fr; liberti@lix.polytechnique.fr

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_18

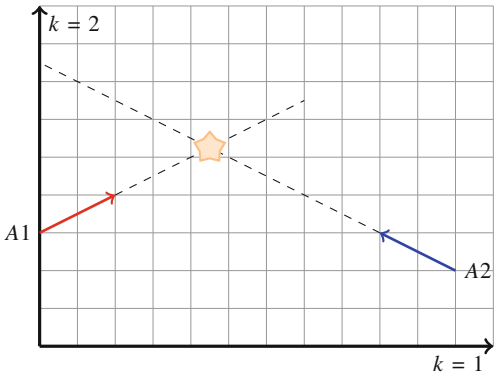


Fig. 1 Two conflicting aircraft

be performed subliminally), while keeping the trajectories unchanged: we present a Mathematical Programming (MP) formulation for aircraft separation based on speed regulation. For a wider introduction to this problem, see [1].

We remark that altitude changes are not usually considered an acceptable way to solve conflicts since they consume more fuel and feel uncomfortable to passengers. We will therefore assume that all aircraft fly within a fixed altitude layer. This will allow us to model travelling aircraft as moving points in $\mathbb{R}^2$ (see Fig. 1 as an example).

There is a large number and variety of approaches to the conflict detection and resolution problem. In this paper we compare our results to those obtained in [1] and [2]. These works propose Mixed-Integer Nonlinear Programming formulations for the deconfliction problem. Specifically, [1] also proposes a heuristic algorithm based on decomposing the problem into subproblems each of which only involves a small number of aircraft. The partial solutions are then combined to form a globally feasible but possibly sub-optimal solution of the original problem. A Feasibility Pump heuristic is proposed in [2]: this algorithm alternately solves two relaxed subproblems at each iteration, while minimizing the distance between the relaxed solutions.

Another approach based on aircraft HAC is proposed in [3]. First a MINLP formulation of the problem of minimizing heading angle changes satisfying the separation condition is presented. Then another mixed 0-1 nonlinear program is proposed: the number of aircraft conflicts that can be solved by speed regulation is maximized. These two MINLPs are solved using existing global solvers and then using a two-step methodology where the solution of the second MINLP is used as a pre-processing step for the first one.

Several papers consider conflicts involving more than two aircraft. In [4], for example, the planar multiple conflicts resolution problem is formulated as a nonconvex quadratically constrained quadratic program, where the objective function is chosen so as to minimize the speed deviations from the desired speed. The problem is then approximated by a convex semidefinite program, the optimal

solution of which is used to randomly generate feasible and locally optimal conflict resolution maneuvers.

An equity-oriented conflict-resolution (ECR) model is introduced in [5]. The ECR model combines three optimization stages, which attempt to: resolve a maximum number of potential conflicts; promote fair conflict-resolution maneuvers (airlines are equally affected by the trajectory adjustments); reduce the delay induced by the trajectory changes. The goal is to identify a set of conflicts that can be resolved altogether, reduce the deviation from total equity and eventually minimize the total delay in the system.

Another approach is presented in [6]. This approach uses the geometric characteristics of aircraft trajectories to obtain closed-form analytical solutions for optimal combinations of heading and speed changes for horizontal-plan conflict resolution, minimizing the magnitude of the velocity vector change. This closed forms can be used also to compute the solution for speed change alone and for heading change alone.

The rest of this paper is organized as follows. In Sect. 2 we introduce the parameters and decision variables of our formulations, the bilevel formulation, and two single-level reformulations. In Sect. 3 we present our cut generation algorithm for solving the bilevel problem. In Sect. 4 we discuss some computational results.

2 Mathematical Formulations

An "optimal deconfliction" must involve a minimal deviation from the original aircraft flight plan, subject to the distance between aircraft to exceed a given safety threshold. The objective function of our formulations will therefore aim at minimizing the sum of the speed change of each aircraft. Requiring that each aircraft pair respects the safety distance at each time instant t of a given interval $[0, T]$ involves the satisfaction of an uncountably infinite set of constraints.

We propose a MP formulation of the speed-change problem variant.

1. **Sets:**
   - $A = \{1, .., n\}$ is the set of aircraft (n aircraft move in the shared airspace)
   - $K = \{1, 2\}$ is the set of directions (the aircraft move in a Euclidean plane)
2. **Parameters:**
   - T is the length of the time horizon taken into account [hours]
   - d is the minimum required safety distance between a pair of aircraft [Nautical Miles NM]
   - x_{ik}^0 is the k-th component of the initial position of aircraft i
   - v_i is the initial speed of aircraft i [NM/h]
   - u_{ik} is the k-th component of the direction of aircraft i
   - $q_i^{\min}$ and $q_i^{\max}$ are the bounds on the potential speed modification for each aircraft

3. **Variables:**

- q_i is the possible increase or decrease of the original speed of aircraft i: $q_i = 1$ if the speed is unchanged, $q_i > 1$ if it is increased, $q_i < 1$ if it is decreased
- t_{ij} is the instant of time defined for the aircraft pair i and j for which the distance between the two aircraft is minimized

The terminology and symbols are taken from [1], where the problem is formulated by a MINLP since there are variables both continuous and integer and nonlinear constraints arise from the separation condition modeling.

2.1 Bilevel Formulation of the Problem

In order to address the issue of uncountably many constraints for each value in $[0, T]$, we propose to formulate the problem as a bilevel MP (for more details on bilevel programming, see [7]) with multiple lower-level problems. Each of these subproblems ensures that the minimum distance between each aircraft pair exceeds the safety distance threshold. Thus, each lower-level subproblem involves the lower-level variable t_{ij} and is parameterized by the upper-level variables q:

$$\min_{q,t} \quad \sum_{i \in A} (q_i - 1)^2 \tag{1}$$

$$\forall i \in A \quad q_i^{\min} \leq q_i \leq q_i^{\max} \tag{2}$$

$$\forall i < j \in A \quad d^2 \leq \min_{t_{ij} \in [0,T]} \sum_{k \in \{1,2\}} ((x_{ik}^0 - x_{jk}^0) + t_{ij}(q_i v_i u_{ik} - q_j v_j u_{jk}))^2 \tag{3}$$

The upper-level (convex) objective function is the sum of squared aircraft speed changes. This corresponds to finding the feasible solution with the minimum speed change, as mentioned before. It must be minimized w.r.t the variables t and q, with each q_i within the given range $[q_i^{\min}, q_i^{\max}]$.

The objective of each lower-level subproblem is to minimize over $t_{ij} \in [0, T]$ the relative Euclidean distance between the two aircraft it describes; note that this is also a convex function. This minimum distance, reached at t_{ij}^*, must be at least d^2. This corresponds to imposing the minimum safety distance d between aircraft i and j within $[0, T]$.

2.2 KKT Reformulation

We follow standard practice and replace each convex lower-level subproblem by its Karush-Kuhn-Tucker (KKT) conditions. Assuming some regularity condition (e.g.

the Slater's condition) holds, this yields a single-level MP with complementarity constraints. Given the KKT multipliers μ_{ij} and λ_{ij} defined for each lower-level problem, we have:

$$\min_{q,t,\mu,\lambda} \quad \sum_{i\in A}(q_i - 1)^2 \tag{4}$$

$$\text{s.t.} \qquad \forall i \in A \quad q_i^{min} \le q_i \le q_i^{max} \tag{5}$$

$$\forall i < j \in A \quad \sum_{k\in\{1,2\}} (2t_{ij}(q_i v_i u_{ik} - q_j v_j u_{jk})^2 + \\ + 2(x_{ik}^0 - x_{jk}^0)(q_i v_i u_{ik} - q_j v_j u_{jk}) - \mu_{ij} + \lambda_{ij}) = 0 \tag{6}$$

$$\forall i < j \in A \quad \mu_{ij},\ \lambda_{ij} \ge 0 \tag{7}$$

$$\forall i < j \in A \quad \mu_{ij}\, t_{ij} = 0 \tag{8}$$

$$\forall i < j \in A \quad \lambda_{ij}\, t_{ij} - \lambda_{ij}\, T = 0 \tag{9}$$

$$\forall i < j \in A \quad -t_{ij} \le 0,\ t_{ij} \le T \tag{10}$$

$$\forall i < j \in A \quad \sum_{k\in\{1,2\}} ((x_{ik}^0 - x_{jk}^0) + t_{ij}(q_i v_i u_{ik} - q_j v_j u_{jk}))^2 \ge d^2 \tag{11}$$

Constraints (6)–(10) correspond to stationarity, primal and dual feasibility conditions and complementary slackness. The last constraint Eq. (11) is necessary to ensure that each KKT solution t_{ij}^* respects the safety distance.

2.3 Dual Reformulation

We propose another closely related reformulation of the bilevel problem (1)–(3), which arises because the lower-level subproblems are convex Quadratic Programs (QP). Specifically, their duals are also QPs which only involve dual variables [8, 9]. In particular, an upper-level constraint such as Eq. (3) has the form

$$\mathsf{const} \le \min\{\frac{1}{2}\, x^\top Q x + p^\top x \mid Ax \ge b \wedge x \ge 0\}$$

with Q positive semidefinite. By strong duality it can be written as follows:

$$\text{const} \leq \max\{-\frac{1}{2}\, y^\top Qy + b^\top z \mid A^\top z - Qy \leq p \wedge z \geq 0\}, \tag{12}$$

where the maximization QP on right hand side is the dual of the previous minimization one [9].

Proposition 1 *Equation* (12) *can be replaced by*

$$\text{const} \leq -\frac{1}{2}\, y^\top Qy + b^\top z \wedge A^\top z - Qy \leq p \wedge z \geq 0 \; (*)$$

in Eq. (1)–(3).

Proof If Eq. (12) is active, then the maximum objective function value of the QP is const. Because of the max operator, the objective function of the QP cannot attain any larger value. This means that $(*)$ can only be feasible when $-\frac{1}{2}\, y^\top Qy + b^\top z$ attains its maximum over $A^\top z - Qy \leq p$ and $z \geq 0$. If Eq. (12) is inactive, it has no effect on the optimum. Since $(*)$ is a relaxation of Eq. (12), the same holds. □

Given the dual variables y and z of the lower-level subproblems, Proposition 1 yields the following reformulation of (1)–(3).

$$\min_{q,y,z} \quad \sum_{i \in A} (q_i - 1)^2 \tag{13}$$

$$\forall i \in A \quad q_i^{min} \leq q_i \leq q_i^{max} \tag{14}$$

$$\forall i < j \in A \quad -\sum_{k=1}^{2} (q_i v_i u_{ik} - q_j v_j u_{jk})^2\, y_{ij}^2 + (-T) z_{ij} \geq d^2 - \sum_{k=1}^{2} (x_{ik}^0 - x_{jk}^0)^2 \tag{15}$$

$$\forall i < j \in A \quad -\frac{z_{ij}}{2} - \sum_{k=1}^{2} (q_i v_i u_{ik} - q_j v_j u_{jk})^2\, y_{ij} \leq \sum_{k=1}^{2} (x_{ik}^0 - x_{jk}^0)(q_i v_i u_{ik} - q_j v_j u_{jk}) \tag{16}$$

$$\forall i < j \in A \quad z_{ij} \geq 0 \tag{17}$$

3 Cut Generation Algorithm

We introduce a solution algorithm for the bilevel formulation (1)–(3), using a cutting plane approach. We iteratively define the feasible set of the upper-level problem by means of quadratic inequalities in the upper-level variables q_i, q_j.

The algorithm is as follows:

CP Algorithm

1. Let $h = 1$; initialize the relaxation R_h of the bilevel program, obtained considering only the upper-level problem; more explicitly, R_1 is:

$$\min_{q} \quad \sum_{i \in A} (q_i - 1)^2$$
$$\forall i \in A \quad q_i^{\min} \leq q_i \leq q_i^{\max}$$

2. Solve R_h and get q^*.
3. For each aircraft pair (i, j), compute the instant $\tau_{ij}^h \in [0, T]$ for which the distance between i and j is minimum and check if this distance is greater than or equal to the safety value d.
4. If for all the pairs the safety threshold is respected, the algorithm ends (q^* is an optimal solution of the bilevel formulation).
   Else, for all the pairs (i, j) violating the inequality, add to R_h the cut:

$$\sum_{k \in \{1,2,\ldots,k\}} ((x_{ik}^0 - x_{jk}^0) + \tau_{ij}^h (q_i v_i u_{ik} - q_j v_j u_{jk}))^2 \geq d^2 \tag{18}$$

   obtaining R_{h+1}.
5. Put $h = h + 1$ and go back to 2.

Note that in *step 3* of the algorithm τ_{ij}^h is easily computed in closed form.

4 Computational Experiments

We consider the set of instances proposed in [1], where n aircraft are placed on a circle of given radius r, with initial speed v_i and a trajectory defined by a heading angle such that aircraft fly toward the center of the circle (or slightly deviating with respect to such direction). See Fig. 2.

Then we also consider instances always proposed in [1] in which aircraft move along straight trajectories intersecting in n_c conflict points.

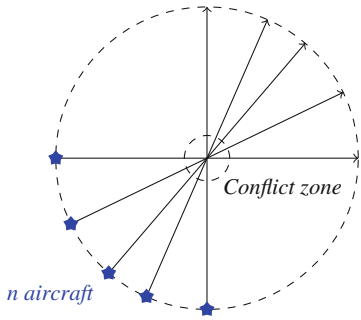


Fig. 2 n conflicting aircraft flying towards the center of a circle

We set: $T = 2\,\text{h}$, $d = 5\,\text{NM}$, $v_i = 400\,\text{NM/h}$ for each $i \in A$. For the "*circle instances*" the heading angles cap_i are randomly generated and parameters x^0_{ik} and u_{ik} are given by

$$u_{i1} = \cos(\mathsf{cap}_i), \quad u_{i2} = \sin(\mathsf{cap}_i), \quad x^0_{ik} = -r\,u_{ik}$$

The bounds q_i^{min} and q_i^{max} are set to 0.94 and 1.03 respectively.

We implement the proposed formulations using the AMPL modeling language [10] and solve the bilevel one with the Cutting Plane algorithm before presented (CP in the Table 1) and the others with the global optimization solver Baron [11] (B in the Table 1) or, when Baron is not successful (exceeding the time-limit set to 15,000 s), with a Multistart algorithm (MS in the Table 1).

The Multistart method for the KKT reformulation uses SNOPT [12] at each iteration (1000 iterations in total), while the one for the Dual reformulation uses IPOPT [13]. Also for the Cutting Plane algorithm (CP), at each iteration we solve the relaxed formulation R_h using IPOPT, that implements an Interior-Point method, for the NLP relaxation.

All the solvers are run with their default settings. The tests are performed on a 2.7 GHz Intel Core i7 processor with 16 GB of RAM and macOS Mojave Operating System.

Our results are reported in Table 1, and compared with those that are the best among the ones obtained with different methods in [1] and [2], that not always guarantee optimal final solutions, using a *matheuristic* approach.

Looking at the solutions obtained on these instances, it appears that they are comparable. The value of the objective function is always very low, given the nature of the problem (q must be in [0.94, 1.03]). We report in bold the best solutions and the minimum time required for each instance.

Table 1 Results obtained solving different formulations of the aircraft deconfliction problem

Instances			Cafieri	Bilevel			KKT reformulation			Dual reformulation		
n	n_c	r	Best obj	Obj	Time (s)	Solver	Obj	Time (s)	Solver	Obj	Time (s)	Solver
Circle												
2	–	100	2.531e−3	2.531e−3	**0.2**	CP	**2.524e−3**	0.3	B	2.526e−3	0.4	B
3	–	200	1.667e−3	1.667e−3	1.6	CP	1.664e−3	**1.5**	B	**1.663e−3**	3.7	B
4	–	200	**4.009e−3**	4.029e−3	**26.3**	CP	4.025e−3	65.4	B	4.017e−3	184.4	B
5	–	300	**3.033e−3**	3.056e−3	**2084.2**	CP	3.052e−3	12,511.1	B	3.050e−3	13,978.3	B
6	–	300	**6.033e−3**	6.557e−3	1549.2	CP	6.088e−3	32.0	MS	6.096e−3	**7.8**	MS
Non-circle												
6	5	–	1.295e−3	**1.254e−3**	**1.1**	CP	**1.254e−3**	53.3	MS	**1.254e−3**	14.9	MS
7	4	–	1.617e−3	**1.1591e−3**	**1.3**	CP	**1.1591e−3**	238.8	MS	**1.1591e−3**	31.2	MS
7	6	–	1.579e−3	**1.566e−3**	**2.7**	CP	**1.566e−3**	86.9	MS	**1.566e−3**	33.2	MS
8	4	–	2.384e−3	**2.384e−3**	**2.5**	CP	**2.384e−3**	1163.4	MS	**2.384e−3**	39.5	MS
10	10	–	1.470e−3	**1.397e−3**	**5.8**	CP	1.469e−3	835.2	MS	**1.397e−3**	78.9	MS

Acknowledgements The project leading to this application has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement MINOA No 764759.

References

1. Cafieri, S., Durand, N.: Aircraft deconfliction with speed regulation: new models from mixed-integer optimization. J. Glob. Optim. **58**(4), 613–629 (2014)
2. Cafieri, S., D'Ambrosio, C.: Feasibility pump for aircraft deconfliction with speed regulation. J. Glob. Optim. **71**(3), 501–515 (2018)
3. Cafieri, S., Omheni, R.: Mixed-integer nonlinear programming for aircraft conflict avoidance by sequentially applying velocity and heading angle changes. Eur. J. Oper. Res. **260**(1), 283–290 (2017)
4. Frazzoli, E., Mao, Z.-H., Oh, J.-H., Feron, E.: Resolution of conflicts involving many aircraft via semidefinite programming. J. Guid. Control. Dyn. **24**(1), 79–86 (2001)
5. Rey, D., Rapine, C., Dixit, V.V., Waller, S.T.: Equity-oriented aircraft collision avoidance model. IEEE Trans. Intell. Transp. Syst. **16**(1), 172–183 (2015)
6. Bilimoria, K.D.: A geometric optimization approach to aircraft conflict resolution. In: 18th Applied Aerodynamics Conference, p. 4265 (2000)
7. Dempe, S., Kalashnikov, V., Prez-Valds, G.A., Kalashnykova, N.: Bilevel Programming Problems: Theory, Algorithms and Applications to Energy Networks. Springer, Berlin (2015)
8. Dorn, W.S.: Self-dual quadratic programs. J. Soc. Ind. Appl. Math. **9**(1), 51–54 (1961)
9. Dorn, W.S.: Duality in quadratic programming. Q. Appl. Math. **18**(2), 155–162 (1960)
10. Fourer, R., Gay, D.M., Kernighan, B.W.: AMPL: A Modeling Language for Mathematical Programming. Cengage learning (2002)
11. Sahinidis, N.V., Tawarmalani, M.: BARON 7.2.5: Global Optimization of Mixed-Integer Nonlinear Programs, User's Manual. The Optimization Firm LLC (2005)
12. Gill, P.E.: User's Guide for SNOPT version 7.2 (2006)
13. Wächter, A., Biegler, L.: On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming. Math. Program. **106**(1), 25–57 (2006)

Computational Evaluation of Data Driven Local Search for MIP Decompositions

Saverio Basso and Alberto Ceselli

Abstract Driven by the perspective use in decomposition based general purpose solvers, we tackle the issue of improving Dantzig-Wolfe decomposition patterns for generic Mixed Integer Programs (MIP). In particular, we consider the scenario in which a MIP instance and its decomposition are given as input and we address the task of manipulating such decomposition by observing only static algebraic components, with the aim of producing better computational performance features (tighter bounds and comparable computing times). We propose a local search algorithm guided by data driven models and evaluate its performance on MIPLIB instances while starting from decompositions given by either static or data driven detectors.

Keywords Dantzig-Wolfe decomposition · General purpose solvers · Machine learning

1 Introduction

The emerging field of data analytics asks for optimization tools which are not only computationally powerful, but also easy to integrate in decision support and information systems. General purpose solvers perfectly fit this need: they offer full modeling flexibility, and their effectiveness has improved exponentially for decades [1].

While branch-and-cut based solvers behave well on generic Mixed Integer Programming (MIP) instances, a few drawbacks are known, limiting their efficacy on selected classes of problems that unfortunately contain also key applications in science and engineering. According to evaluations like [1], it is unlikely that fine tuning and algorithm engineering will be enough to provide a breakthrough for them.

S. Basso (✉) · A. Ceselli
Università degli Studi di Milano, Dipartimento di Informatica, Milan, Italy
e-mail: saverio.basso@unimi.it; alberto.ceselli@unimi.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_19

At the same time, decomposition methods prove to be a valid alternative in several cases. They often allow to exploit more structure and to unlock high potential in terms of massive computing parallelism [2], just to mention two appealing features.

That is why researchers have started to devise general purpose frameworks exploiting problem decomposition [10, 11]. Among them, the state of the art is GCG [7]: based on the framework SCIP, it allows a user to submit *generic MIP instances* together with *decomposition patterns*, reformulates the MIP according to the given decomposition and solves the reformulated MIP by branch-and-price.

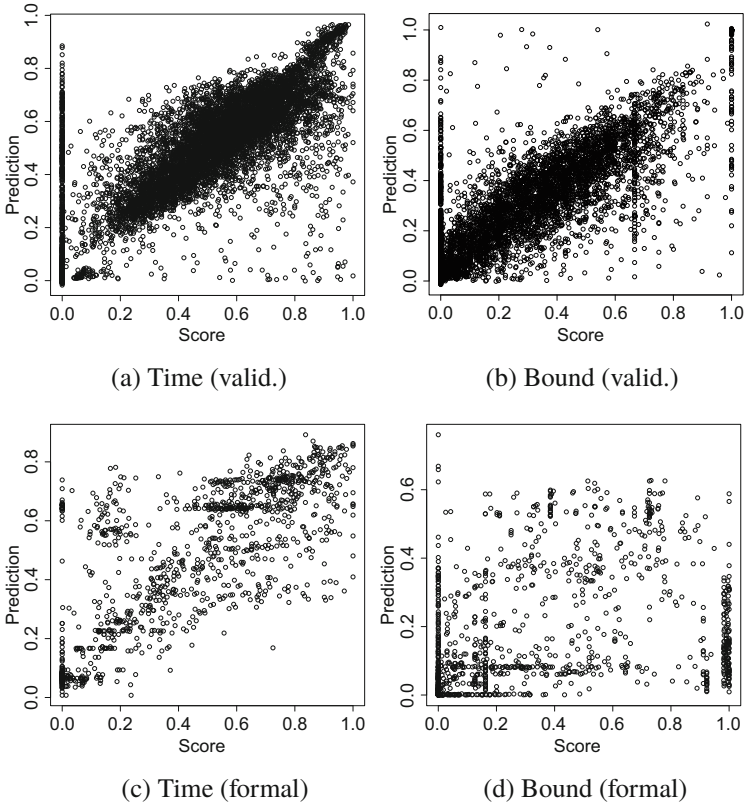
Besides deciding if a MIP is amenable to be decomposed [8] a key issue remains, that is how to find a suitable decomposition pattern, in case the user is not a mathematical programming expert, or is simply unaware if a decomposition approach might be useful or not. Formally, a decomposition pattern can always be induced; attempts to *automatically* detect suitable decomposition (i.e. by means of algorithms) are recent [6] but promising. A common difficulty of these *static* detectors is finding MIP and decomposition features, that is pure (static) algebraic properties of input data, that might guide detection algorithms.

Indeed, in previous works we proposed data-driven approaches aimed at obtaining this type of insights [4] experimenting also, as proof of concept, on fully data driven decomposition methods [3], creating useful decompositions from scratch.

In this paper, instead, we experiment on using data-driven models to guide local search algorithms, improving existing decomposition patterns. In our view these existing decompositions can be those produced by either static detectors or data-driven ones, or even those directly given as input from users to solvers like GCG. While intuitively simple, such a task is made very complex by peculiar, unexpected behaviour of decomposition patterns, as reported for instance in [5]. In Sect. 2 we review some motivation insight from earlier works; in Sect. 3 we describe our local search framework, and in Sect. 4 we report computational findings on generic MIPLIB [9] instances.

2 Preliminary Investigations

In [3] we designed and developed a data driven detector based on supervised machine learning models to predict independently bound and time scores of decompositions (within a 0-1 range) starting from static features. Its most successful setting includes a custom ranking function based on dominance between decompositions in such a score space. In Fig. 1 we report a brief summary of our findings when the following experimental setting is used. A set of MIPLIB is chosen, and several algorithmically generated random decompositions for them are experimented by optimization runs, obtaining score values, and creating a training dataset for our data driven models. Then, a second dataset is created in the same way, including new decompositions for the same MIPLIB instances; it is used as a validation test set. Finally, a third dataset is created by considering *new*



(a) Time (valid.) (b) Bound (valid.)

(c) Time (formal) (d) Bound (formal)

Fig. 1 Time and Bound regression results for the validation set (top) and the formal test set (bottom). Taken from [3]

MIP instances, and decompositions for them; it is also used as a formal test set. That is both validation and formal sets test our data driven models on previously unseen decompositions, but the formal test set has the additional complexity of being created on previously unseen MIP instances. Figure 1 has one point for each decomposition, having on the y-axis the predicted value and on x-axis the real score. We observe that on the validation set both predictors performs well: when the MIP is already known, decomposition bound and time prediction appear possible. When the MIP is unknown, the results are still encouraging: it is still possible to predict time with good accuracy; bound regression is instead suitable only in some scenarios: overall it performs poorly, except on a small subset of the instances.

We therefore focused on the following research question: if a (potentially new) MIP instance and a tentative decomposition of it are given, can we design an algorithm that takes advantage of our data driven predictors to improve such decomposition?

3 Local Search Algorithms

Motivated by the results obtained during our preliminary investigation, we propose a local search algorithm (Fig. 2) aimed to improve the quality of decomposition patterns, that exploits our data driven models to guide the exploration of neighbour regions of the solution space.

Our algorithm starts with a candidate solution, a decomposition that can either be provided by detectors (both static or data driven) or by a simpler random sampling of constraints, and then moves iteratively to neighbor solutions until a termination condition is met.

Generation We define as neighbourhood the set of decompositions that differ, from the candidate one, of just one constraint. In particular, we consider all the solutions that can be generated by convexifying a single additional constraint in the candidate decomposition, that is by moving one relaxed constraint from the border to one of the blocks. In the event that two or more independent blocks share variables with the newly added constraint, the blocks are merged (see Algorithm 1 pseudocode).

The selection of such a neighbourhood has two main advantages. First, it allows to generate and explore a very wide neighbourhood: its cardinality equals the product of the number of constraints in the border and the number of blocks. Second, Dantzig-Wolfe decomposition theory guarantees that the bound of every

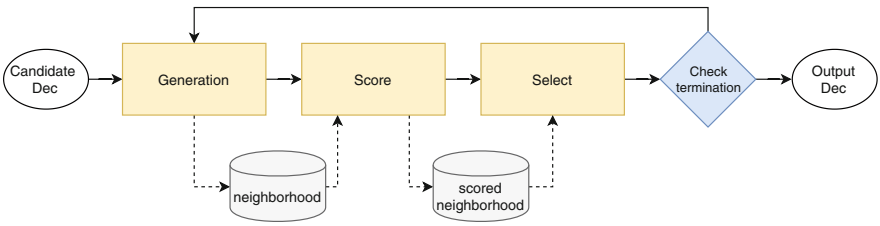


Fig. 2 Local search algorithm

Algorithm 1: Generation pseudocode

```
Data: decomposition D
Result: set Δ of decompositions
Δ = ∅ ;
for i ∈ border(D), k ∈ blocks(D) do
    E = D;
    move(i, k) =
        remove i from border(E);
        add i to block(k, E);
        check merging block(k, E);
    Δ = Δ ∪ E
end
```

decomposition generated with our move will not be worse than the bound of the candidate decomposition. This allows us to disregard bound prediction, which as reported in Sect. 2 proves difficult.

Evaluation and Selection The selection of a move in the neighbourhood is critical, as convexifying a single additional constraint may have a deep impact into the structure of the decomposition blocks. Furthermore, the evaluation of such a large set must be computed quickly, and therefore it is completely impracticable to check even a subset of them by optimization runs. Indeed, no move can worsen the bound: focusing on predicted final computing time, we use our data-driven time regressor as a fitness function. At each iteration, once the neighbourhood has been generated, we evaluate the quality of each decomposition in it using our time regression model. We choose that of minimum predicted time, with a best improvement strategy, as the next candidate decomposition.

Terminating Conditions We repeat the generation of the neighbourhood, its evaluation and the selection of the next candidate solution until termination, which we fix either after a fixed number of iterations or after some quality threshold has been reached, leaving these as parameters of the algorithm. Their selection is discussed in the next section.

4 Experimental Results

For the experimental analysis, we considered three datasets. The first, *Dataset T*, is taken from [4] and consists of 34,565 decompositions for 36 problems from MIPLIB2003 and MIPLIB2010. Each decomposition is described by 117 static features collected during pre-processing. For each decomposition in T, bound and time at the root node have subsequently been obtained by running simulations with GCG 2.1.1, and added as scores to the dataset in post-processing. The second, *Dataset P*, consists of 12 new MIP problems, taken from [3], that were generated by perturbing `rout.mps`, `fiber.mps` and `p2756.mps` of Dataset T. Perturbation was performed either numerically (that is, entries of the MIP problem different from 1 and -1 were multiplied by a random fraction) or structurally (a random number of coefficients in the constraints matrix were swapped). In both scenarios, we considered two levels of perturbation: 6% and a 20%.

The third, *Dataset N*, is composed of 5 entirely new MIP instances from MIPLIB2017: `bienst1`, `bienst2`, `danoint`, `h50x2450` and `newdano`. We chose them because they are listed in MIPLIB as "similar" to `rout.mps`, according to an analysis on static features. In turn, `rout.mps` belongs to Dataset T, and both [4] and [3] suggest it to be suitable to a decomposition approach.

Experimental Setup We performed experiments on a PC equipped with a 16 core AMD Threadripper 1950X 3.4 GHz CPU, 32 GB DDR4 RAM and Ubuntu 18.04 operating system. Our local search algorithm was developed in Python 3.6.7 and

made use of XGBoost library 0.71 to train our supervised learning models on dataset T, with default objective (regression with squared loss), default settings and the following custom settings: maximum tree depth (max_depth) = 13, boosting learning rate (eta) = 0.1 and number of trees to fit (n_estimators) = 100. Additional scripts in Bash and R 3.4.4. were respectively employed to manage the generation of the neighbourhood and to merge blocks with common variables.

We tested our local search algorithms on dataset N and P with the setup explained in Sect. 3. Furthermore, to validate our `data-driven` selection policy, we also tested a configuration with a `random` selection policy: at each iteration, time prediction is not taken into account and the next candidate decomposition is chosen randomly from the neighbourhood. We remark that such a random policy is blind to computing time, but still guarantees to never worsen the bound with local search moves. We repeated tests with random selection 5 times and we aggregated the results in averages. Detection of the starting decomposition was either performed by using GCG 3.0 static detectors or by using our Data Driven Detector [3] on 300 random decompositions for each problem. A single decomposition, obtained at the end of the local search process, was used to optimize the corresponding MIP instance through GCG, setting a 2 h timelimit. Final bound and computing time on that MIP instance were measured accordingly.

Experiment 1: Selection Policies Profiling We report in Table 1 a comparison between selection policies. The table is composed of two blocks. Each value in the first block represents the ratio between the result obtained by the `data-driven` policy and that obtained by the `random` policy. In the second block instead, we report the overall percentage of decompositions produced by data driven local search which are dominated by a random one ($R \rightarrow LS$), that of random

Table 1 Comparison between selection policies

Inst.	Value	Iteration				Overall
		5	10	20	30	
Bienst1	Time	49.83%	11.93%	43.58%	98.82%	51.04%
	Bound	99.36%	109.25%	115.13%	120.13%	110.97%
Bienst2	Time	54.86%	13.55%	24.18%	42.86%	33.86%
	Bound	98.46%	106.03%	113.26%	119.33%	109.27%
Danoint	Time	0.94%	0.33%	0.29%	104.80%	26.59%
	Bound	100.02%	100.05%	100.08%	99.99%	100.03%
h50×2450	Time	56.37%	60.20%	35.15%	70.95%	55.67%
	Bound	100.00%	100.00%	100.00%	100.00%	100.00%
Newdano	Time	26.98%	4.70%	2.46%	3.08%	9.30%
	Bound	101.17%	118.68%	117.63%	115.87%	113.34%
Overall dominance	$R \rightarrow LS$	4.00%	0.00%	0.00%	8.00%	
	$LS \rightarrow R$	76.00%	100.00%	100.00%	64.00%	
	No dominance	20.00%	0.00%	0.00%	28.00%	

decompositions dominated by a data driven one ($LS \rightarrow R$) and that of not dominated decompositions. We consider four iteration limit values: 5, 10, 15 and 30. Technically, the tests with iteration limit of 5, 10 and 15 are obtained as dumps of the 30 iteration limit test. The starting decomposition is always provided by GCG static detectors. We highlight that `data-driven` selection performs better when percentages are below 100 for Time (the lower, the better) and above 100 for Bound (the higher, the better).

Our results show that using data-driven time prediction to explore the neighbourhood has potential and that, as expected, it performs better than a random selection. In fact we observe that, on every instance, `data-driven` selection provides on average better times and bounds. Indeed, the decomposition produced with `data-driven` selection is always better in terms of computing time, except when solving the decomposition for `danoint.mps` after 30 iterations. In this case, both data-driven and random selection perform poorly: the former reaches timeout whilst the latter gets very close to it. However, we report that after the 22nd iteration, our time predictor assigns to the whole neighbourhood very low scores and using a well tuned threshold as a stop condition should be sufficient to avoid this worst case scenario. After 5 iterations, even if neither policy makes assumption on bounds, data-driven selection consistently provides better results. The only exception is instance `h50x2450.mps` in which no improvement of the bound can be obtained, as the starting decomposition is already enough to fully close the duality gap. The results about dominance are even stronger: when stopping at 10 or 20 steps, all random decompositions are worse than the data driven ones in terms of both bound and running time.

Experiment 2: Impact of the Starting Decomposition In a second round of experiments we consider the effect of starting with a decomposition found by our data driven detector [3] (300 random samples), instead of that produced by GCG algorithmic detectors. The local search iterations limit is always set to 30. Our results over Dataset N and P are reported in Table 2. For each instance and each detector, we report the Time (T.) and Bound required to solve the decomposition found, the percentage of constraints in blocks (Cvx) and the number of remaining constraints in the border (Mc). For dataset P, we report each instance with its name followed by the percentage of perturbation that was applied (either 6 or 20) and its type (either n for numerical or s for structural). Being all minimization problems, in case of Bound, higher values correspond to better bounds.

Both detectors mostly produce decompositions that can be solved in seconds. In almost every test, decompositions detected for dataset N are mostly low quality and provide the same bounds of the linear relaxation of the problem. As reported before, the only exception is for `h50x2450.mps`: in this case, GCG is able to find a decomposition producing no duality gap. When facing dataset P instead, GCG finds decompositions with higher bounds for `p2756.mps` and its permutations whilst our data driven detector can generate better decompositions for `rout.mps`.

Table 2 Comparison between GCG static detection and Data driven detection for choosing the starting decomposition

		GCG detection				Data driven detection			
D	Inst.	T. [s]	Bound	Cvx	Mc	T. [s]	Bound	Cvx	Mc
N	Bienst1	3.53	11.72	77.8%	128	4.02	11.72	66.5%	193
	Bienst2	4.28	11.72	77.8%	128	2.17	11.72	65.4%	199
	Danoint	5.25	62.64	67.9%	213	20.94	62.69	54.2%	304
	h50×2450	587.79	32,906.88	98.1%	49	32.33	11,147.73	96.1%	99
	Newdano	3.90	11.72	77.6%	129	2.12	11.72	76.9%	133
	Overall			79.8%	129.4			71.8%	185.6
P	p2756-20n	0.73	1982.69	97.5%	19	0.20	1984.00	79.3%	156
	p2756-20s	3.30	2798.09	58.3%	315	0.73	2774.70	44.5%	419
	p2756-06n	2.76	2888.52	97.5%	19	1.44	2729.66	86.5%	102
	p2756-06s	2.60	2794.17	82.2%	134	0.95	2703.00	70.9%	220
	rout-20n	2.22	654.24	87.6%	36	2.18	686.57	88.7%	33
	rout-20s	2.35	542.01	80.4%	57	405.80	680.82	92.1%	23
	rout-06n	2.26	991.33	87.6%	36	40.79	1024.59	89.7%	30
	rout-06s	4.51	838.01	82.5%	51	13.74	842.34	83.2%	49
	Overall			84.2%	83.4			79.3%	129.0

We also observe that decompositions found for dataset P are, on average, more convexified than the ones detected for dataset N. In this case, the number of constraints in the border is very low and the number of meaningful iterations that can be performed with our local search algorithm is limited. We also report that the average value of constraints in blocks is higher for GCG detection than Data driven detection.

Experiment 3: Local Search Improvement Potential We present a summary of the performance of our algorithm on Dataset N and P when using GCG decompositions as starting ones, in terms of decomposition improvement potential. We report for 5, 10, 20 and 30 iterations limit, the average bound improvement of the final decomposition with respect to the starting one (Fig. 3a) and the required time to solve the final decomposition (Fig. 3b). For the sake of comparison, we have removed timeouts (one for each Dataset). We can observe that overall the local search algorithm presents better results on dataset N for both time and bound. However, when the number of iterations is low, that is 5 o lower, improvements are limited.

We recall that in instances of Dataset P the border has fewer constraints. The impact of local search in such a setting can be expected to be smaller as the corresponding decompositions are already strongly convexified and adding even a single additional constraint to the blocks may yield subproblems whose complexity do not differ significantly from that of the full MIP.

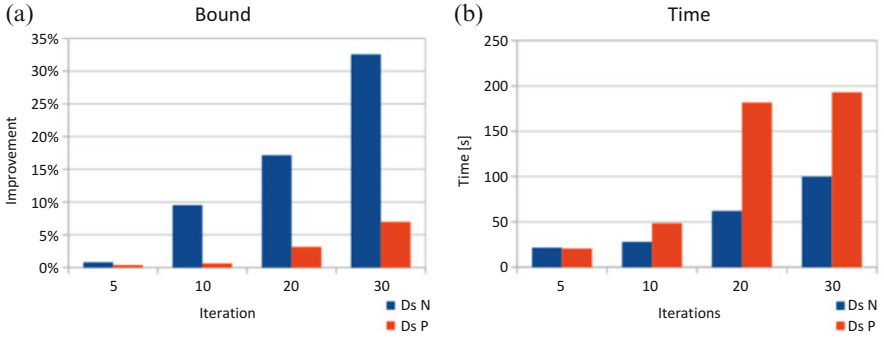


Fig. 3 Performance of local search starting from GCG decompositions on Dataset N and Dataset P. We present, for each dataset, the relative bound improvement and the absolute time in seconds. (**a**) Bound improvement. (**b**) Time

Table 3 Comparison between Ls_{GCG} and Ls_{DD}

			Bound improv.		Time [s]		
Inst.	z^*	z_{LP}	Ls_{GCG}	Ls_{DD}	sGCG	Ls_{GCG}	Ls_{DD}
Bienst1	46.75	11.72	38.06%	5.20%	3.53	75.13	50.12
Bienst2	54.60	11.72	38.71%	10.08%	4.28	124.69	9.33
Danoint	65.67	62.64	0.00%	0.08%	5.25	7344.00	104.82
h50×2450	32,906.88	11,147.73	0.00%	−66.12%	587.79	175.00	32.33
Newdano	65.67	11.72	53.48%	181.47%	3.90	24.66	16.74

For each instance of Dataset N, we report decomposition solving time and the improvement of the bound with respect to static detectors of GCG

Experiment 4: Overall Effect of Local Search Finally, in Table 3 we summarize the performance of our local search algorithm. As a benchmark we consider the decomposition provided by the static detectors of GCG (sGCG). We fix local search iterations limit to 30, and we consider two cases: actually starting local search from the decomposition provided by sGCG (Ls_{GCG}) or fully employing a data-driven approach, starting local search from a decomposition provided by the Data Driven Detection method of [4] (Ls_{DD}). That is Ls_{GCG} evaluates the effect of local search for further improving a decomposition which was produced by an existing optimization algorithm, while Ls_{DD} can be seen as a fully data-driven alternative to algorithmic detectors. For each instance (Inst.) of dataset N, we report its optimal solution value (z^*) and the value of its linear relaxation (z_{LP}). Then, we present the improvement of the bound with respect to sGCG, obtained by solving the output decomposition of both Ls_{GCG} and Ls_{DD}. We also report Time in seconds for Ls_{GCG}, Ls_{DD} and sGCG. We report that Ls_{DD} failed on one instance, `h50x2450.mps`, because bash and R scripts could not handle the generation of a neighbourhood of such a large size.

We observe that on some instances (`bienst1.mps` and `bienst2.mps`) Ls_{GCG} presents best bound improvements while keeping time under control. Even if both detectors start with the same bound, as reported in Table 2, sGCG decompositions are much more convexified and likely structured and Ls_{GCG} provides better bounds after 30 iterations. The same applies for `h50x2450.mps` in which sGCG alone is already able to find a decomposition yielding no duality gap. However, we observe that Ls_{DD} always has lower times and that could potentially perform as well or better when the same overall number of constraints as Ls_{GCG} have been convexified. Furthermore, Ls_{DD} provides better bounds on `danoint.mps` and `newdano.mps`. In the first case, this is due to the fact that Ls_{GCG} reaches timeout and cannot close the root node within a time limit of 2 h. Stopping before 30 iterations, as suggested in Experiment 1 could allow to match Ls_{DD} improvements. In the second, the local search algorithm performs well with both detectors but Ls_{DD} presents a much better bound. This suggests that with these two MIP instances the decomposition pattern found by sGCG does not match the instance very well and limits the performance of our local search algorithm. Finally, we also report that even when sGCG finds an optimal solution, our local search algorithm might still have a positive impact: we can see that the decomposition found on instance `h50x2450.mps` by sGCG requires almost 10 min to be solved whilst using our local search algorithm can lower running time to about 3 min. The counter-intuitive phenomenon on `h50x2450.mps` brings an interesting insight: decompositions yielding lower running times do not always come at the cost of loser bounds; sometimes, the opposite is observed.

Summarizing, given a decomposition, our local search algorithm can consistently find a new solution with better bounds while keeping time under control. However, as expected, the percentage of improvement depends on the detector used to find a starting decomposition.

5 Conclusion and Perspectives

Following the results of [4] and [3], we have explored the idea of using the models of our data driven detectors to improve a given decomposition. We have designed a novel local search algorithm that generates a wide neighbourhood with a move tailored to overcome the shortcomings of bound prediction and we have used our data driven time regressor to evaluate and explore the solution space. Then, we have tested our algorithm on new, unseen instances from MIPLIB2017 and a custom built dataset. Such an approach proves to be effective, consistently improving even those decompositions produced by state-of-the-art algorithmic detectors. Our results suggest that, while the performance of the algorithm is strongly dependent on the starting decomposition and that further testing on an expanded dataset is required to fully explain its behaviour, data driven local search can consistently improve a given decomposition.

References

1. Achterberg, T., Wunderling, R.: Mixed integer programming: analyzing 12 years of progress. In: Facets of Combinatorial Optimization, pp. 449–481 (2013)
2. Basso, S., Ceselli, A.: Asynchronous column generation. In: Proceedings of the Nineteenth Workshop on Algorithm Engineering and Experiments (ALENEX), pp. 197–206 (2017)
3. Basso, S., Ceselli, A.: Computational evaluation of ranking models in an automatic decomposition framework. Electron. Notes Discrete Math. **69**, 245–252 (2018)
4. Basso, S., Ceselli, A., Tettamanzi, A.: Random sampling and machine learning to understand good decompositions. Ann. Oper. Res. (2018)
5. Bastubbe, M., Luebbecke, M.E., Witt, J.T.: A Computational Investigation on the Strength of Dantzig-Wolfe Reformulations. In: Proceeding of SEA (2018)
6. Bergner, M., Caprara, A., Ceselli, A., Furini, F., Lübbecke, M., Malaguti, E., Traversi, E.: Automatic Dantzig-Wolfe reformulation of mixed integer programs. Math. Program. A **149**(1–2), 391–424 (2015)
7. Gamrath, G., Lübbecke, M.E.: Experiments with a generic Dantzig-Wolfe decomposition for integer programs. LNCS **6049**, 239–252 (2010)
8. Kruber, M., Luebbecke, M.E., Parmentier, A.: Learning when to use a decomposition. Proc. CPAIOR **2017**, 202–210 (2017)
9. MIPLIB 2017. http://miplib.zib.de, last access April 2019
10. Vanderbeck, F., Wolsey, L.: Reformulation and decomposition of integer programs. In: Jünger, M., Liebling, Th.M., Naddef, D., Nemhauser, G.L., Pulleyblank, W.R., Reinelt, G., Rinaldi, G., Wolsey, L.A. (eds) 50 Years of Integer Programming 1958–2008. Springer, Berlin (2010)
11. Wang, J., Ralphs, T.: Computational experience with hypergraph-based methods for automatic decomposition in discrete optimization. In: Gomes, C., Sellmann, M. (eds.) Integration of AI and OR techniques in constraint programming for combinatorial optimization problems. LNCS **7874**, 394–402 (2013)

An Integer Programming Formulation for University Course Timetabling

Gabriella Colajanni

Abstract The university timetabling problem is defined as the process of assigning lessons of university courses to specific time periods throughout the five working days of the week and to specific classrooms suitable for the number of students registered and the needs of each course. A university timetabling problem is modeled, in this paper, as an optimization problem using 0-1 decision variables and other auxiliary variables. The model provides constraints for a large number of different rules and regulations that exist in academic environments, ensuring the absence of collisions between courses, teachers and classrooms. The real case of a Department and some instances from the literature are presented along with its solution as resulted from the presented ILP formulation.

Keywords University course timetable · Scheduling · Integer programming problem

1 Introduction

The university course timetabling problem has traditionally and formally been described as "the allocation, subject to constraints of given resources to objects being placed in space time, in such a way as to satisfy as nearly as possible a set of desirable objectives" [20].

Since timetabling problems differ from one university to another, in this paper we propose a new Integer Linear Programming (ILP) problem based on the formulation of a course timetabling for the Department of Mathematics and Computer Science, University of Catania. Therefore, this paper focuses on solving this operational decision-making problem finding the best assignment of lessons to rooms and

G. Colajanni (✉)
Department of Mathematics and Computer Science, University of Catania, Catania, Italy
e-mail: colajanni@dmi.unict.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_20

timeslots, so that a set of constraints is satisfied, to create timetables of high quality for students and teachers.

A lot of research has been done on timetabling, and, being part of optimization problems, several well-known techniques of the computer science and operations research fields have been utilized (see, for instance, [3, 4, 18, 21] for some literature review).

Among the first approaches in mathematical programming, Akkoyunlu in [2] presented linear and integer programming models for some versions of the problem for a university timetabling problem.The same problem was studied in [7] and [16] again with the help of linear programming models. More recently, in [5] the teacher assignment problem is combined with a form of the timetabling problem and solved through commercial software for goal programming. In a similar manner, in [14] a linear programming formulation is provided for the classroom allocation problem, a sub-problem of the university timetabling. In [8–11], authors proposed some Integer Linear Programming models presenting binary variables and using additional variables to describe some type of constraints. Researchers also proposed ILP models based on the decomposition of the problem into two or more stages (see, for instance, [12] and [15]). Also in [19] authors presented a two-stage integer programming approach for building a university course (of the KU Leuven Campus Brussels) timetable that aims at minimizing the resulting student flows. Reference [1] describes the main features of the constraint solver that the authors used to generate a timetable for the Computer Science Department of the University of Munich. The objective of a Course Timetabling problem is to find the best weekly assignment of university course lessons to rooms and time periods, subject to constraints which can be divided into two types: hard constraints that must necessarily be satisfied and soft constraints, where the violation of these should be minimized, determining the objective function (such as room capacity, minimum number of working days, lessons compactness and room stability).

Despite the amount of literature and research dedicated to this problem, a gap still exists between reality and used models, or, in other words, between theory and practice (this is discussed by McCollum and Ireland in [17]). The paper is organized as follows. In Sect. 2 we firstly show the used notation and then we present the mathematical formulation. Section 3 is dedicated to the application of the new model, with appropriate adaptations, to the case study of the Master's Degree Course in Mathematics of the Department of Mathematics and Computer Science (DMI) of the University of Catania. In order to test and validate the model we use some adapted instances available from the literature. Finally, in Sect. 4, we draw some conclusions and discuss some guidelines for future research.

2 Course Timetabling Formulation

We present the mathematical formulation of a new model of a weekly calendar of university lessons, first stating the sets of resources and users, then declaring some auxiliary sets, useful for specifying certain features that intertwine resources and

users, so we describe the main types of variables and the parameters we will use; finally we will name the weights related to the violation of soft constraints.

2.1 Notation

We use a notation that is as close as possible to the papers in the wide literature.

To create our model, we define the sets and subsets described in Table 1 and the parameters shown in Table 2.

We indicate with $c \in C$ a generic course belonging to the set of all the university courses and we denote by $r \in R$ and $h \in H$ the typical room and timeslot, respectively.

We consider $D = \{1, 2, 3, 4, 5\}$ the set of the weekly working days, from Monday to Friday. Lastly, we indicate with $t \in T$ and $s \in S$ the generic teacher and student, respectively.

Table 1 Sets and subsets

C	The set of courses
R	The set of rooms
H	The set of timeslots across the week, that are the time periods
D	The set of days of the planning period
T	The set of teachers
S	Be the set of students
$C(t) \subset C$	The set of courses taught by teacher $t \in T$
$H(d) \subset H$	The set of time periods belonging to the day $d \in D$
F	The set of time periods corresponding to the first timeslot of each day
L	The set of time periods corresponding to the last timeslot of each day
$ND \subset C \times H$	The set which is formed by pairs $(\bar{c}, \bar{h})$ indicating the time period $\bar{h}$ in which the teacher of the course $\bar{c}$ is not available
$HD \subset C \times R$	The set which is formed by pairs $(\bar{c}, \bar{r})$ indicating that the teacher of the course $\bar{c}$ cannot teach any class other than $\bar{r}$

Table 2 Parameters

l_c	The number of lessons for course $c \in C$
m_c	The minimum number of weekdays on which there must be a lesson for course $c \in C$
n_c	The demand for course $c \in C$, that is the number of students in that course
p_r	The capacity of room $r \in R$, that is the number of seats in the room
W^C	The weight corresponding to the penalization of lessons compactness
W^M	The weight corresponding to the penalization of minimum number of working days
W^P	The weight corresponding to the penalization of room capacity
W^S	The weight corresponding to the penalization of room stability
W^D	The weight corresponding to the penalization of distribution

We now present the decision variables.

The first decision variable that we introduce is the binary variable x_{chr} which establishes the possible assignments of the courses to the classrooms and the available time periods.

The variable x_{chr} is formally defined as follows:

$$x_{chr} = \begin{cases} 1 \text{ if a lecture of course } c \text{ is planned in the timeslot } h \text{ and in room } r, \\ 0 \text{ otherwise,} \end{cases}$$

$$\forall c \in C, \quad \forall h \in H, \quad \forall r \in R.$$

We also use some other decision variables:

- v_h: the binary variable which is used to count the number of unused hours between lectures and is expressed by

$$v_h = \begin{cases} 1 \text{ if the timeslos } h \text{ is a free period} \\ 0 \text{ otherwise,} \end{cases} \qquad \forall h \in H;$$

- $dist_{cd} \in \{0, 1\}$: the binary variable which controls the distribution of the lessons of the course $c \in C$ in consecutive days $d, d + 1 \in D$ (it will be clarified later);
- y_{cd}: the binary variable which is used to count the number of lessons of the course $c \in C$ in the day $d \in D$ and is expressed by

$$y_{cd} = \begin{cases} 1 \text{ if at least a lecture of the course } c \text{ is planned in } h \in H(d) \\ 0 \text{ otherwise,} \end{cases}$$

$$\forall c \in C, \forall d \in D;$$

- z_c: the integer variable which, for each course $c \in C$, counts the number of days below the minimum number of working days.

2.2 The Formulation

We begin by describing our hard constraints that are of fundamental importance for the planning of the lessons.

$$\sum_{h \in H} \sum_{r \in R} x_{chr} = l_c, \quad \forall c \in C \tag{1}$$

Constraint (1) requires that all the lessons of each course must be planned.

$$\sum_{c \in C} \sum_{r \in R} x_{chr} \leq 1, \quad \forall h \in H \tag{2}$$

Constraint (2) establishes that do not take place simultaneously.

$$\sum_{c \in C} x_{chr} \leq 1, \quad \forall h \in H, \quad \forall r \in R \tag{3}$$

Constraint (3), on the other hand, requires that at most one class of a course $c \in C$ can be assigned to the same period of time in the same classroom.

Two additional constraints take into account the needs of teachers.

$$\sum_{c \in C(t)} \sum_{r \in R} x_{chr} \leq 1, \quad \forall h \in H, \quad \forall t \in T \tag{4}$$

$$\sum_{r \in R} x_{\bar{c}\bar{h}r} = 0, \quad \forall (\bar{c}, \bar{h}) \in ND \tag{5}$$

First, the impossibility of assigning more than one lesson to be held by the same teacher at the same time, as formalized in constraint (4) which requires that at most a class lesson taught by a teacher can be assigned to a classroom in a time period. Secondly, the non-availability of the teacher of the course $\bar{c} \in C$ is taken into consideration, in the interval of time $\bar{h} \in H$, due to the performance of another teaching in another room for the same degree course or for another one located in the same Department or not. Consequently, constraint (5) establishes that no lesson of a course must be assigned to a classroom in a period of time in which the teacher of that course is not available.

$$\sum_{c \in C} \sum_{h \in H} x_{ch\bar{r}} - \sum_{h \in H} x_{\bar{c}h\bar{r}} = 0, \quad \forall (\bar{c}, \bar{r}) \in HD \tag{6}$$

$$\sum_{h \in H} \sum_{r \in R} x_{\bar{c}hr} - \sum_{h \in H} x_{\bar{c}h\bar{r}} = 0, \quad \forall (\bar{c}, \bar{r}) \in HD \tag{7}$$

Two new additional constraints, (6) and (7), are placed to assign a course to a specific room by virtue of the need to use particular teaching tools, placed only in some classrooms, or to the specific needs of the course teacher. Constraint (6) expresses the following situation: the teacher of the course $\bar{c}$ is the only one, in the set of teachers considered, to be able to do lessons in the classroom $\bar{r}$.

Constraint (7) expresses the condition that the teacher of the course $\bar{c}$ cannot teach any class other than $\bar{r}$.

After defining the hard constraints, we focus now on the objective function which is given by the sum of five terms, each of which contains the penalties for the violation of soft constraints.

A first objective of the model, which is aimed at solving the problem of programming a calendar of university lessons, is to minimize the creation of "free period" within each programming day. It therefore requires that there are no unused

hours between successive lessons of two courses. This is expressed by the first addendum of the objective function:

$$\sum_{h\in H}\left(W^C v_h\right),$$

where the variable v_h is determined by the following constraints:

$$\sum_{c\in C}\left(\sum_{r\in R} x_{c(h-1)r} - \sum_{r\in R} x_{chr} + \sum_{r\in R} x_{c(h+1)r}\right) - 1 \le v_h, \tag{8}$$

$$\forall h \in H - \{F \cup L\},$$

$$\sum_{c\in C}\left(\sum_{r\in R} x_{chr} - \sum_{r\in R} x_{c(h+1)r}\right) \le v_h, \qquad \forall h \in F. \tag{9}$$

From the didactic point of view it would be preferable that the lessons of the same course are not placed on two consecutive days, therefore, in this paper, we formulate this new request in the second addendum of the objective function:

$$W^D \sum_{\substack{d\in D\\ d\neq d_5}} \sum_{c\in C} dist_{cd}.$$

We underline that the relationship between variables $dist_{cd}$ and y_{cd} is given by the following new constraint:

$$y_{cd} + y_{c(d+1)} - 1 \le dist_{cd}, \quad \forall c \in C, \forall d \in D, d \neq d_5 \tag{10}$$

The third addend, $\sum_{c\in C} W^M z_c$, aims to reduce the difference between the number of days used for the weekly distribution of the lessons of a course and the minimum number of working days required for that course. For this purpose the integer variable z_c is used, counting, for each course $c \in C$, the number of days below the minimum number of working days:

$$\sum_{h\in H(d)} \sum_{r\in R} x_{chr} \ge y_{cd} \quad \forall c \in C, \quad \forall d \in D, \tag{11}$$

$$\sum_{d\in D} y_{cd} + z_c \ge m_c \quad \forall c \in C. \tag{12}$$

The fourth addendum of the objective function represents the request that the classroom which the course is assigned to (over a period of time) should have the

capacity to accommodate all the students enrolled in that course and to minimize the number of seats that exceed it. Therefore we use the following soft constraint:

$$\sum_{c\in C}\sum_{h\in H}\sum_{r\in R} W^P |n_c - p_r| x_{chr}.$$

In particular, unlike what has been studied in literature, we underline the insertion of the absolute value in the difference between the two quantities n_c and p_r.

Regarding the soft constraint of the room stability in the model we present, with respect to the models examined in literature, we dealt with the daily permanence in the same classroom, of the students, instead of the previous condition of stability of the classroom concerning each course because in this case it could happen that consecutive lessons are held in different and distant classrooms.

Therefore, the fifth addendum,

$$W^S \sum_{d\in D}\sum_{r\in R}\sum_{\substack{h\in H(d)\\ h\notin L}} \left| \sum_{c\in C} x_{chr} - \sum_{c\in C} x_{c(h+1)r} \right|,$$

aims to minimize the number of daily classroom shifts for students, analyzing daily pairs of consecutive time periods $(h, h+1)$ for each course. To this reason it is necessary to exclude from the periods of time examined the last period of each afternoon, that is $h \notin L$.

The problem can therefore be formulated as an optimization problem as follows:

$$\begin{aligned}\min \Bigg\{ &\sum_{h\in H}\left(W^C v_h\right) + W^D \sum_{\substack{d\in D\\ d\neq d_5}}\sum_{c\in C} dist_{cd} + \\ &+ \sum_{c\in C} W^M z_c + \sum_{c\in C}\sum_{h\in H}\sum_{r\in R} W^P |n_c - p_r| x_{chr} + \\ &+ W^S \sum_{d\in D}\sum_{r\in R}\sum_{\substack{h\in H(d)\\ h\notin L}} \left| \sum_{c\in C} x_{chr} - \sum_{c\in C} x_{c(h+1)r} \right| \Bigg\} \end{aligned} \tag{13}$$

$$\sum_{c\in C}\left(\sum_{r\in R} x_{c(h-1)r} - \sum_{r\in R} x_{chr} + \sum_{r\in R} x_{c(h+1)r}\right) - 1 \leq v_h, \tag{14}$$
$$\forall h \in H - \{F \cup L\}$$

$$\sum_{c\in C}\left(\sum_{r\in R} x_{chr} - \sum_{r\in R} x_{c(h+1)r}\right) \leq v_h, \quad \forall h \in F \tag{15}$$

$$y_{cd} + y_{c(d+1)} - 1 \leq dist_{cd}, \quad \forall c \in C, \quad \forall d \in D, \quad d \neq d_5 \tag{16}$$

$$\sum_{h\in H(d)}\sum_{r\in R} x_{chr} \geq y_{cd}, \quad \forall c \in C, \quad \forall d \in D \tag{17}$$

$$\sum_{d\in D} y_{cd} + z_c \geq m_c, \quad \forall c \in C \tag{18}$$

$$\sum_{h\in H}\sum_{r\in R} x_{chr} = l_c, \quad \forall c \in C \tag{19}$$

$$\sum_{c\in C}\sum_{r\in R} x_{chr} \leq 1, \quad \forall h \in H \tag{20}$$

$$\sum_{c\in C} x_{chr} \leq 1, \quad \forall h \in H, \quad \forall r \in R \tag{21}$$

$$\sum_{c\in C(t)}\sum_{r\in R} x_{chr} \leq 1, \quad \forall h \in H, \quad \forall t \in T \tag{22}$$

$$\sum_{r\in R} x_{\overline{c}\overline{h}r} = 0, \quad \forall(\overline{c},\overline{h}) \in ND \tag{23}$$

$$\sum_{c\in C}\sum_{h\in H} x_{ch\bar{r}} - \sum_{h\in H} x_{\bar{c}h\bar{r}} = 0, \quad \forall(\overline{c},\overline{r}) \in HD \tag{24}$$

$$\sum_{h\in H}\sum_{r\in R} x_{\bar{c}hr} - \sum_{h\in H} x_{\bar{c}h\bar{r}} = 0, \quad \forall(\overline{c},\overline{r}) \in HD \tag{25}$$

$$x_{chr} \in \{0,1\}, \quad \forall c \in C, \quad \forall h \in H, \quad \forall r \in R \tag{26}$$

$$y_{cd} \in \{0,1\}, \quad \forall c \in C, \quad \forall d \in D \tag{27}$$

$$v_h \in \{0,1\}, \quad \forall h \in H \tag{28}$$

$$z_c \in \mathbb{N}, \quad \forall c \in C \tag{29}$$

$$dist_{cd} \in \{0,1\}, \quad \forall c \in C, \quad \forall d \in D, \quad d \neq d_5 \tag{30}$$

The latest constraints families (26)–(30) define the domain of the variables of the problem.

We note that the model results to be an integer non-linear programming model, but it could easily be linearized taking into account that, in this case, an absolute value $|a_i - b_i|$ could be replaced by t_i, adding the following constraints:

$$a_i - b_i \leq t_i, \tag{31}$$

$$a_i - b_i \geq -t_i. \tag{32}$$

Therefore, we consider the integer linear programming model.

3 Application of the Model

Now we apply the model formulated in the previous section, with appropriate adaptations, firstly to the case study of the Master's Degree in Mathematics of the Department of Mathematics and Computer Science (DMI) of the University of Catania and then we use benchmark instances from the literature.

3.1 Case Study: Master's Degree in Mathematics

Since we want to report all the results for transparency purposes, we select the size of problems as reported. The numerical data are real values and are constructed for easy interpretation purposes.

The Department proposes a single 2-year Master's Degree course in Mathematics, organized in three curricula: Theoretical, Application and Didactic. It is also possible to propose individual study plans as an alternative to the proposed curricula. In reference to the courses of the first semester we will not consider the difference between the various curricula chosen by the students and we will adopt the convention to consider all the teachings as if they belonged to a single curriculum. This position is motivated by a "qualitative choice" adopted by the Department: in order to direct students towards an informed choice of study plan they are given the opportunity to attend the first lessons of all courses at the beginning of the academic year (in the first decade of October) and to deliver the study plan 1 month after the beginning of lessons.

The DMI has 19 classrooms available, 16 of which can be used for university courses, and three used for other activities. In the model we present we consider only three of the ten rooms with a capacity of 140 seats in order to guarantee the possibility of assigning a classroom of this capacity to the courses of all three curricula and to decrease the number of model variables and, therefore, decrease the processing time for resolving the problem. We also consider the availability of a room with 50 seats, one of 48, one of 36, one of 32, one of 24 and two of 16 seats.

A special case is room 124, Anile hall, with a capacity of 36 seats, used exclusively for the lessons of one of the courses, also used for seminars.

Lessons take place from Monday to Friday. The time intervals available for each working day range from 8 to 19, considering a free hour for the lunch break and a maximum number of daily lessons of eight (upper-bound constraint). In the model we present we take care of the weekly scheduling of the lessons from Monday to Friday.The time periods available weekly consist of 40 h that can be numbered, but considering that we assign two consecutive hours to the lesson of each course, we adopt the convention of indicating a period of time as an interval of two consecutive hours so as to have the following situation (Table 3).

In such a way, the slots are halved, reducing the number of variables, computational complexity and execution time.

Table 3 Time periods available weekly

Monday	Tuesday	Wednesday	Thursday	Friday
1	5	9	13	17
2	6	10	14	18
3	7	11	15	19
4	8	12	16	20

Table 4 Resulting lesson schedule

Monday	Tuesday	Wednesday	Thursday	Friday
	5	2	8	2
7	3	1	5	3
1	6	7	6	
4	8	4		

The teaching staff of the Master's Degree Course in Mathematics is composed of 24 professors. In particular, 13 of them teach the first year, of which 10 are internal to the DMI, the others come from other departments; 8 of them teach at the first semester, and 10 at the second semester. The periods of non-availability for some of them are known. Considering that a professor holds courses in other departments, it is necessary to distance his/her lessons that take place in distinct locations at least 1 h. Furthermore, the following preferences of three teachers are known: two would like to take the lessons of their courses only during the morning hours and one from 9 am to 1 pm.

For the numerical solution of the mathematical model we use IBM ILOG CPLEX Optimization Studio software, in its IDE version 12.8.0.

The problem related to the first semester is solved by executing the model formulated in the previous section through the CPLEX software, on an HP 255.5 computer, compute cores 2C+3G, 2.60 GHz, RAM: 8 GB, with a processing time of 9 s 46′, after 285 iterations. The model has 1146 variables, of which 992 binary and 330 constraints. The non zero coefficients are 8729 and the optimal value of the objective function is 160.4.

The resulting lesson schedule for the first semester is shown in Table 4 where at each number corresponds a course.

We underline that all the lessons take place in room 36, except for the lessons of course 2 which is held in room 124 (in accordance with constraints (24) and (25)). We also note that the model always assigns course 2 to the first or last time periods of each time, in order to reduce the number of students shifts from a classroom to another (Room Stability).

3.2 Data from the Literature

For further analysis of the performance of the model, 22 additional problem instances were used. We adapted these instances, except the last one "compNEW", from those that have been proposed as a part of the International Timetabling

Competition (ITC-2007, see [6] and [13]). The dataset is composed by real-world instances provided by the University of Udine. Since the objective of the proposed model is different from the one defined for the competition (we focus on lessons distribution and, concerning the rooms, the proposed model aims at minimizing the number of daily classroom shifts for students, not for courses and too large classrooms are not chosen; concerning the compactness, we avoid the free period, not the isolated lectures), we do not intend to compare the results or validate the solutions obtained with those available in the literature. Moreover, since our model is not curriculum-based, we considered the maximum number of courses, among those in the instances, in such a way that each problem admits at least a feasible solution.

All the experiments were conducted on the same computer (HP 255.5 computer, compute cores 2C+3G, 2.60 GHz, RAM: 8 GB).

In Table 5, for each instance we report the name, the number of teaching days in the week, the number of timeslots per day, the number of lectures to be scheduled, the number of available classrooms, the number of constrains, variables and binary variables of each problem and, finally, the number of iterations and the needed time for finding the optimal solutions.

Table 5 Description and results for optimal solutions of the instances tested

Instance	Days	Timeslots per day	Lectures	Classrooms	Constraints	Variables	Binary variables	Iterations	Time (min)
Comp01	5	6	30	6	2490	2080	1930	261,304	0.44
Comp02	5	5	24	16	4454	3625	3305	51,373	0.25
Comp03	5	5	24	16	4434	3625	3305	1,026,911	1.56
Comp04	5	5	24	18	4978	4065	3705	392,015	0.71
Comp05	5	6	28	9	3975	3335	3110	524,869	1.02
Comp06	5	5	24	18	4977	4065	3705	424,840	1.06
Comp07	5	5	24	20	5521	4505	4105	43,359	0.20
Comp08	5	5	24	18	4984	4065	3705	1,084,012	1.90
Comp09	5	5	24	18	4970	4065	3705	112,003	0.29
Comp10	5	5	24	18	4998	4065	3705	183,174	0.38
Comp11	5	9	45	5	4809	4240	4040	2,163,037	3.10
Comp12	5	6	29	11	5179	4385	4110	4,084,683	7.60
Comp13	5	5	24	19	5246	4285	3905	142,944	0.54
Comp14	5	5	24	17	4685	3845	3505	669	0.16
Comp15	5	5	24	16	4434	3625	3305	1,026,911	1.74
Comp16	5	5	24	20	5509	4505	4105	85,037	0.26
Comp17	5	5	24	17	4732	3845	3505	51,831	0.36
Comp18	6	6	36	9	5088	4269	4044	105,591	0.35
Comp19	5	5	24	16	4464	3625	3305	152,837	0.52
Comp20	5	5	24	19	5252	4285	3905	190,345	0.79
Comp21	5	5	23	18	4515	3605	3245	13,632	0.23
CompNEW	6	8	48	25	32,203	29,963	29,088	8,103,026	45.79

We observe that only $33.\bar{3}\%$ of the problems require more than 1 min for resolution. Particularly, the instance named *comp12* requires more than 7 min because it contains the largest number of unavailability constraints.

Note that we also tested the model using an additional instance (*compNEW*) with the maximum number of lectures, days and timeslots per day usually established by universities. We set a number of classrooms of 25 and we used the same unavailability constraints as *comp21* instance. Therefore, the problem that refers to the last instance consists of 32,203 constraints and 29,963 variables of which 29,088 are binary. This additional instance requires more than 45 min. However, in this case, too, the optimal solution is found.

4 Conclusion and Future Research

An integer programming formulation of the university timetabling problem is presented. The model provides constraints for a great number of operational rules and requirements found in most academic institutions. Treated as an optimization problem, the objective is to minimize a cost function.

Therefore, we have proposed a new model for the university course timetabling problem which allows us to determine the best allocation of university course lecturers to classrooms and timeslots. Through the proposed Integer Linear Programming Problem it is possible to find the exact optimal solution. For instance, we introduced the study about lessons distribution and, concerning the rooms, the proposed model aims at minimizing the number of daily classroom shifts for students, not for courses and too large classrooms are not chosen; concerning the compactness, we avoid the free period, not the isolated lectures as is done in most of the papers in the literature. The model is solvable by existing software tools with IP solvers. The real case of a Department is presented along with its solution as resulted from the presented IP formulation. We used benchmark instances from the literature (ITC) adapting them to our model.

Further research can be conducted to determine the optimal solution of the second semester, with a curriculum-based formulation; a school timetabling problem and to find new methods in the case of a large number of courses, teachers and rooms or in more complex cases.

Acknowledgement This work has been supported by the Università degli Studi di Catania, "Piano della Ricerca 2016/2018 Linea di intervento 2".

References

1. Abdennadher, S., Marte, M.: University Timetabling Using Constraint Handling Rules. In: JFPLC, pp. 39–50. LMU, Munich (1998)
2. Akkoyunlu, E.A.: A linear algorithm for computing the optimum university timetable. Comput. J. **16**(4), 347–350 (1973)

3. Aziz, N.L.A., Aizam, N.A.H.: A survey on the requirements of university course timetabling. World Acad. Sci. Eng. Technol. Int. J. Math. Comput. Phys. Electr. Comput. Eng. **10**, 236–241 (2016)
4. Babaei, H., Karimpour, J., Hadidi, A.: A survey of approaches for university course timetabling problem. Comput. Ind. Eng. **86**, 43–59 (2015)
5. Badri, M.A., Davis, D.L., Davis, D.F., Hollingsworth, J.: A multi-objective course scheduling model: combining faculty preferences for courses and times. Comput. Oper. Res. **25**(4), 303–316 (1998)
6. Bonutti, A., De Cesco, F., Di Gaspero, L., Schaerf, A.: Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results. Ann. Oper. Res. **194**(1), 59–70 (2012)
7. Breslaw, J.A.: A linear programming solution to the faculty assignment problem. Socio Econ. Plan. Sci. **10**, 227–230 (1976)
8. Burke, E., Marecek, J., Parkes, A., Rudova, H.: Penalising patterns in timetables: Novel integer programming formulations. In: Operations Research Proceedings 2007, pp. 409–414. Springer, Berlin (2008)
9. Burke, E., Marecek, J., Parkes, A., Rudova, H.: Decomposition, reformulation, and diving in university course timetabling. Comput. Oper. Res. **37**(3), 582–597 (2010)
10. Burke, E., Marecek, J., Parkes, A., Rudova, H.: A supernodal formulation of vertex colouring with applications in course timetabling. Ann. Oper. Res. **179**(1), 105–130 (2010)
11. Burke, E., Marecek, J., Parkes, A., Rudova, H.: A branch-and-cut procedure for the Udine course timetabling problem. Annal. Oper. Res. **194**(1), 71–87 (2011)
12. Daskalaki, S., Birbas, T.: Efficient solutions for a university timetabling problem through integer programming. Eur. J. Oper. Res. **160**(1), 106–120 (2005)
13. Di Gaspero, L., McCollum, B., Schaerf, A.: The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3). Technical Report. School of Electronics, Electrical Engineering and Computer Science, Queens University SARC Building, Belfast (2007)
14. Gosselin, K., Truchon, M.: Allocation of classrooms by linear programming. J. Oper. Res. Soc. **37**(6), 561–569 (1986)
15. Lach, G., Lubbecke, M.: Curriculum based course timetabling: new solutions to Udine benchmark instances. Ann. Oper. Res. **194**(1), 255–272 (2012)
16. McClure, R.H., Wells, C.E.: A mathematical programming model for faculty course assignment. Decis. Sci. **15**, 409–420 (1984)
17. McCollum, B., Ireland, N.: University timetabling: Bridging the gap between research and practice. In: Burke, E., (ed.) Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT), pp. 15–35 (2006)
18. Pandey, J., Sharma, A.K.: Survey on university timetabling problem. In: 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 160–164 (2016)
19. Vermuyten, H., Lemmens, S., Marques, I., Beliën, J.: Developing compact course timetables with optimized student flows. Eur. J. Oper. Res. **251**(2), 651–661 (2016)
20. Wren, A.: Scheduling, timetabling and rostering a special relationship?. In: Burke, E., Ross, P. (eds.) Practice and Theory of Automated Timetabling in Lecture Notes in Computer Science, vol. 1153, pp. 46–75. Springer, Berlin (1996)
21. Yang, X.F., Ayob, M., Nazri, M.Z.A.: An investigation of timetable satisfaction factors for a practical university course timetabling problem. In: 2017 6th International Conference on Electrical Engineering and Informatics (ICEEI), pp. 1–5 (2017)

On the Sizing of Security Personnel Staff While Accounting for Overtime Pay

Patrick Hosein, Victor Job, and Alana Sankar-Ramkarran

Abstract At many universities today, especially those in developing countries, budget cuts are leading to the reduction of staff for various campus services. One such service is campus security. However, a more holistic view may in fact reveal that, at least when it comes to security, workload reductions may actually increase the overall cost to the University since such reductions result in increased criminal activity which could be detrimental to the university either through direct financial losses, harm to students and staff or through a loss of reputation. For a given workload (i.e., the number of personnel assigned to each post for each shift) one must determine the appropriate staff size to satisfy this workload while minimizing overall cost. The desired workload may vary monthly (e.g., additional staff needed during special events) and, in addition, the number of available staff may vary monthly (e.g., because of increased vacation leave requests during the summer break). Staff must be provided at least 40 h per week so too many staff is not cost effective while insufficient staff requires excessive overtime in order to satisfy the required workload which is also not cost effective. We investigate the optimal trade-off so as to minimize the total financial cost for varying workloads and staff availability. We also take into account the various agreements between management and the employees' Union. We use standard Integer Programming techniques to solve the resulting problem. The additional constraints increase the complexity of the problem but we use a lower bound on the optimal solution to show that the solution obtained is close to optimal.

Keywords Staff size optimization · Staff scheduling · Resource allocation

P. Hosein (✉) · V. Job · A. Sankar-Ramkarran
The University of the West Indies, St. Augustine, Trinidad
e-mail: patrick.hosein@sta.uwi.edu; victor.job@sta.uwi.edu; alana.sankar@sta.uwi.edu

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_21

1 Introduction

Universities are always under pressure to improve efficiencies and reduce budgets in an increasingly competitive educational environment. Today many online programs are available and these cost significantly less than traditional programs since many physical infrastructure costs (e.g., for classrooms) are eliminated. In the case of security, a reduction in the security budget may not be wise since it can result in increased on-campus crime and this in turn can lead to additional costs to the university (harm to students, financial costs for repairs, reputation due to incident reporting in avenues such as social media, etc.). Therefore of utmost importance is the balance between workload cost and security risk. Although an important problem in itself, we do not address it here. We instead assume a given workload and focus on the minimization of the budget required to satisfy this workload. In addition, unionized employees have certain rights from bargaining agreements and these must be included in any computed schedules.

Typically the workload requirements are given as the number of security personnel to assign to a specific number of security posts for each shift. Given this workload, one must then determine the number of staff required and finally the specific assignment of staff to post and shift. Our focus is on determination of the optimal staff size. Give this staff size one can then compute the allocations per post/shift and finally other factors can then be used to make specific personnel allocations.

The two major constraints that must be considered are the fact that each employee must be assigned at least 40 h per week and for each shift the minimal allocation must be satisfied. If insufficient staff are available for a shift then one must request other staff members to work the shift even if such staff members already have been allocated their minimum of 40 h for the week. The extra shifts that they work are called overtime shifts and the rate of pay (overtime pay) is much higher than for regular shifts. Therefore too many staff results in excessive salary payments while too few results in excessive overtime payments. Our objective is to minimize the total cost of basic salaries plus overtime.

The above problem can be solved for the baseline workload and a given staff size. However, both of these change on a weekly basis. The required workload will be less during vacation periods while available staff is reduced during periods of high vacation leave requests. We therefore need to look at the total cost for an entire year and determine the staff that minimizes this quantity.

Staff scheduling problems take many forms and have been solved using a variety of approaches since the fifties. In recent times, the easy availability of historical data means that more efficient and optimal means of solving such problems are possible. Many organizations place a substantial amount of investment into manpower management because a strong, skilled workforce is generally the best investment that can be made by a company. Manpower allocation problems address issues of employing, rostering and scheduling to meet operational demands. In practice, it is

extremely difficult to find optimal solutions to these highly complex problems that minimize cost and satisfy all workplace constraints. For this reason one typically resorts to heuristics.

Constraint programming techniques for scheduling and rostering problems have been researched since the fifties. Mathematical programming approaches for solving manpower allocation problems can be formulated as linear or integer programming problems and generally adopt the famous Dantzig set covering formulation [6]. Many real-world optimization problems, such as the days-off shift tour scheduling problem [3] and crew rostering problem [8] can be solved using Dantzig's formulation.

The paper [7] addresses a similar model in that they take into account overtime work. They found that, compared with the standard 40 h per week schedule, even small amounts of premium-pay overtime work provide significant savings. However their model and application and different to ours since they do not address the combined effect. In the papers by Ingels et al. [10] and [9] the focus is on shift allocation with a given number of overtime shifts (or unscheduled time in their terminology). In our approach we do not fix the number of overtime shifts but rather that number is determined through our optimization objective which is the total cost. Another paper that investigates overtime shifts is [5]. They consider two optimization models. In the first model, a bonus term is assigned in the objective function for shifts that are needed to cover an increased workload while in the second model the workload demand is flexible but its violation is included by a penalty in the objective function. Our approach assigns no arbitrary rewards or penalties but instead uses salary values and pay rates to find the minimal overall cost.

Baker and Magazine [2] considered a workforce scheduling problem that examined the cyclic scheduling of days off in 7-day-a-week operations under alternative labour policies. We also include this constraint in our formulation. In their study, closed-form expressions for optimal workforce size were derived based on the demand profile. Herawati et al.[13] constructed a 24-h shift scheduling model for a hotel security department that takes workload and workers' preference into consideration. In their study, a linear integer programming formulation was used with suitable constraints to obtain an equal distribution of allocated shifts among workers (i.e. fairness) but overtime is not included. Ang et al.[1] proposed a university campus security staff model in which the objective is to maximize the satisfaction of the security personnel with respect to their assigned shifts and days off.

The major difference between our work and those described above is that our model takes into account overtime (or unscheduled) shifts in a more practical way by minimizing total cost. Also, in prior work, the assumption was that sufficient personnel were available to satisfy the workload and so the objective is to find an allocation that satisfied all additional constraints. In our model we vary the number of personnel available. If insufficient are available for the workload then we assume that overtime is paid to complete coverage. If too many personnel are available then additional shifts are added in order to ensure all staff receive their 40 h per week. Since the workload requirement as well as available personnel (because of vacation

and sick leave) varies over time then one must find the number of (permanent) personnel that is optimal for the entire year. One can do this by looking at historical data to determine the estimated workload and available staffing (e.g., assume that the same percentage of staff are available for a given period as was available in the same period in the previous year) and determining the average cost over the entire year. In this paper we illustrate the approach for a given workload and number of personnel.

2 Staff Size Optimization for a Fixed Workload

The first problem that must be addressed is the optimal workload to achieve a given level of security. Too few shifts leads to high risk (and cost) while too many leads to high salary costs with little additional risk benefit. One approach, which is not practical, would be to vary the allocated workload over time and compute the cost due to the risk involved as well as the cost of staff salaries. Naturally, as the workload is increased, the cost of the associated risk drops while the salary cost increases. Therefore this summation would be quite high for too few or too many staff but there should be some minimal value in-between these extremes.

Unfortunately this approach is not realistic and other approaches such as those outlined in [12] are required. Factors that affect workload estimates (taken from [12]) include Type of Institution, Student Population, Age and Gender Profiles, Location and Physical Security Requirements, Number of Buildings, Student Housing, Days and Times of Class Sessions, Campus Size, and Institution's Expectations (level of risk). For example, the ratio of officers to students in most universities lies between 1.8 and 3 per 1000 students. In our particular case, the buildings are widely spread and the campus is surrounded by residential homes so let us assume a ratio of 3/1000. The student population size is 18,000 which means a required staffing of 54.

We were provided the workload requirements for a particular university and so assume that sufficient studies were performed by the University in determining this workload to provide an acceptable level of security risk. Later we will find that the required staffing for the provided workload (approximately 80) is significantly larger than 54 which means that the simplistic approach of basing workload on staff and/or student ratios was inadequate for this particular university.

For convenience we focus on optimization for a 1 week period so that staff have the same schedule every week. However, the approach can be applied to any desired period (e.g., 1 month). Let us assume that the required workload for a 1 week period consists of M 8-h shifts. This may consist of different numbers of staff per shift depending on the time of day and day of the week. We assume that allocation of staff to posts within a shift is managed separately. Assume that the pay rate per shift is r_n for regular (normal) shifts and r_o for overtime shifts. Each employee must work 40 h (5 regular shifts) per week and any additional shift (over 40 h per week) is considered as an overtime shift. Let S represent the average salary paid per week

per employee. Finally assume that we have N employees. For now we will ignore vacation and sick leave but account for them later. If $M = 5N$ then there are exactly enough workers to handle the workload. If $M < 5N$ then some workers will have to be allocated shifts for which they are not needed but this must be done to make up their 40 h. If $M > 5N$ then there are insufficient workers and so overtime will have to be paid to cover the workload. The corresponding total weekly salary cost C can therefore be written as

$$C = SN + \max\{0, (K - 5N)r_o\} \tag{1}$$

Now note that if $M \leq 5N$ then the gradient is positive. If $M > 5N$ then the gradient with respect to N is $S - 5r_o$. In most scenarios the salary of an employee for the week will be less than the amount that would have to be paid in overtime to have someone work that employee's week and hence the gradient is typically negative. If $M > 5N$ then as N is decreased, more staff will have to work overtime. However not everyone likes overtime and furthermore there are limits on how many overtime hours one can work per week.

Since overtime is not compulsory, some staff would view it as an inconvenience and avoid it whereas others are willing to work for additional financial gain. One can take a survey of all staff to determine the average number of extra hours they are willing to work per week. We denote by κ the average, over all staff, of the fraction of regular time each would be willing to work overtime. Without a survey one can look at historical data and compute the fraction of time that staff worked overtime and use this as an estimate for κ. Given this parameter we can estimate the optimal number of staff required to satisfy the given workload by:

$$N_1^* = \frac{M}{5(1+\kappa)} \tag{2}$$

Since the above equation can lead to a non-integer value we need to find a nearest integer value. Since we prefer to err on the side of more overtime then this corresponds to smaller values of N, thus

$$N_1^* = \left\lfloor \frac{M}{5(1+\kappa)} \right\rfloor \tag{3}$$

So far we have assumed that the workload is fixed for each week. In practice the actual workload may be more (e.g., during graduation ceremonies) while at other times the required workload might be less (e.g., during public holidays). In general, the total number of shifts required for a year will tend to be larger than $52M$. We therefore use the average workload over the year instead of the baseline workload. If we denote the fraction of excess workload over the prior year by β, then the average

workload per week is given by $(1+\beta)K$ and so we now have

$$N_2^* = \left\lfloor \frac{M(1+\beta)}{5(1+\kappa)} \right\rfloor \tag{4}$$

Note that if the total number of overtime shifts that staff are willing to work equals the excess shifts then $\kappa = \beta$ but in general this is not the case. We can estimate β from historical data but of course it will vary by year because of unforeseen events (e.g., a student protest).

So far we have assumed that each staff member works 40 h every week but of course they are given time for vacation, sick days, holidays, maternity leave, etc. Let α denote the average, over all staff, of the fraction of total regular shifts that an employee actually works. We can estimate this from historical data but it will vary for each employee. Therefore the expected number of shifts within a week that someone needs to work is given by 5α and hence we now have our final estimate for the optimal N which is given by

$$N^* = \left\lfloor \frac{M(1+\beta)}{5\alpha(1+\kappa)} \right\rfloor \tag{5}$$

This is the number of staff required to serve the workload based on the assumption that the workload can be satisfied precisely and will be used to compute a lower bound on cost since the additional constraints can only result in increased costs.

3 A 0/1 Linear Programming Model for Staff Scheduling

In this section we determine how to allocate staff to shifts. Note that in any given week the number of available staff may be less than N^* because of vacation and other types of leave. Furthermore the workload may be above the baseline value of M (e.g., because of an event) or below (e.g., because of a holiday). Therefore both N and M may vary each week.

Assume that N staff members are available for the week and that the required workload for the week is M shifts. We assign staff to one regular shift within a day, and any additional shifts to be allocated (if necessary) are paid as overtime. Let us index the day of the week by $k = 1 \ldots 7$, the shift (on each day) by $j = 1 \ldots 3$ and index staff by $i = 1 \ldots N$. We define the following binary decision variable:

$$x_{ijk} = \begin{cases} 1 \text{ if staff } i \text{ is allocated to shift } j \text{ on day } k \\ 0 \text{ otherwise} \end{cases} \tag{6}$$

Finally, we denote by W_{jk} the number of staff required for shift j on day k, and by A_{jk} the number of additional staff allowed on shift j on day k. This is needed for

the scenario in which one must provide a shift to someone to make up their total of 5 per week. We now state the optimization problem as follows:

$$F(\boldsymbol{x}^*) \equiv \min_{x_{ijk}\in\{0,1\}} \sum_{i=1}^{N}\sum_{j=1}^{3}\sum_{k=1}^{7} x_{ijk} \tag{7}$$

subject to:

$$x_{i1k} + x_{i2k} + x_{i3k} \leq 2 \qquad \forall i,k \tag{8}$$

$$x_{i2k} + x_{i3k} + x_{i1(k \bmod 7+1)} \leq 2 \qquad \forall i,k \tag{9}$$

$$x_{i3k} + x_{i1(k \bmod 7+1)} + x_{i2(k \bmod 7+1)} \leq 2 \qquad \forall i,k \tag{10}$$

$$\sum_{i=1}^{N} x_{ijk} \geq W_{jk} \qquad \forall j,k \tag{11}$$

$$\sum_{i=1}^{N} x_{ijk} \leq W_{jk} + A_{jk} \qquad \forall j,k \tag{12}$$

$$\sum_{j=1}^{3}\left[x_{ij((i-1) \bmod 7+1)} + x_{ij(i \bmod 7+1)}\right] = 0 \qquad \forall i,k \tag{13}$$

$$\sum_{j=1}^{3} x_{ij((i+k) \bmod 7+1)} \geq 1 \qquad k = 1\ldots 5, \qquad \forall i. \tag{14}$$

In the above formulation (a mod b) is defined as the remainder obtained when a is divided by b. Equation (7) is the objective function which represents the total number of shift allocations. If there are no allocated overtime shifts, this objective is minimized when all staff can be allocated exactly 40 h per week, and in this case the total assignment is $5N$. In the case where both regular and overtime shifts are allocated, the minimum (optimal) value of the objective function would exceed $5N$. Inequalities (8)–(10) are constraints which ensure that the number of consecutive shifts worked by each employee in a 24-h period does not exceed two. Inequality (11) is the constraint that on any given shift, the total number of allocated staff is sufficient to meet the workload requirement W_{jk}. The constraint given in line (12) restricts the total number of staff allocated on a shift to at most

$W_{jk} + A_{jk}$. Equation (13) represents the constraint that each staff member should have 2 consecutive days off. Finally, inequality (14) is the constraint that, with the exception of the 2 consecutive days off, each staff member must work at least one shift per day. These are all constraints that are imposed by the agreements between management and the Union.

Note that the constraint that employees get 2 consecutive days off is handled by choosing which 2 days this will be for each employee (Eq. (13)) rather than allowing these 2 days to occur anywhere in the week. This in fact places an additional constraint on the problem but we will see that the resulting solution is still close to optimal. In future work we will investigate the use of a more relaxed constraint.

4 Application to a Real Case Study

In this section we provide numerical results for a workload used at a particular university. The results were obtained on a laptop with 16 GB RAM and a 2 GHz processor. We used the Julia programming language which is a high-level general-purpose dynamic programming language designed for high-performance numerical analysis and computational science [4]. Julia has a package named `JuMP` for modeling optimization problems. This package contains many linear programming solvers and for this problem we used the `CPLEX` optimizer. The `CPLEX` Optimizer was named for the Simplex Method as implemented in the `C` programming language.

We consider the case of the workload described in Table 1, which consists of a total of $M = 395$ shifts. We assume a weekly salary of S and that 30% of this consists of benefits [11]. Therefore the rate per shift is given by $r_n = 0.7S/5$. We assume an overtime rate of $r_o = 1.75r_n = 0.245S$. If we ignore the various constraints and assume perfect scheduling then a lower bound on the cost (normalized by S) can be obtained using Eq. (1):

$$\frac{C}{S} = N + \max\{0, 0.245(395 - 5N)\} \tag{15}$$

In Fig. 1 we plot this lower bound cost and we also plot the cost obtained after solving the associated optimization problem including the various constraints.

These results indicate that the optimal number of employees is 72. For this value of N, when constraints were taken into account, a total of 402 shifts were

Table 1 Workload requirements used for numerical results

Shifts	Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday
Morning (M)	19	16	22	22	22	22	22
Evening (E)	19	16	22	22	22	22	22
Night (N)	14	11	16	16	16	16	16
Total	52	43	60	60	60	60	60

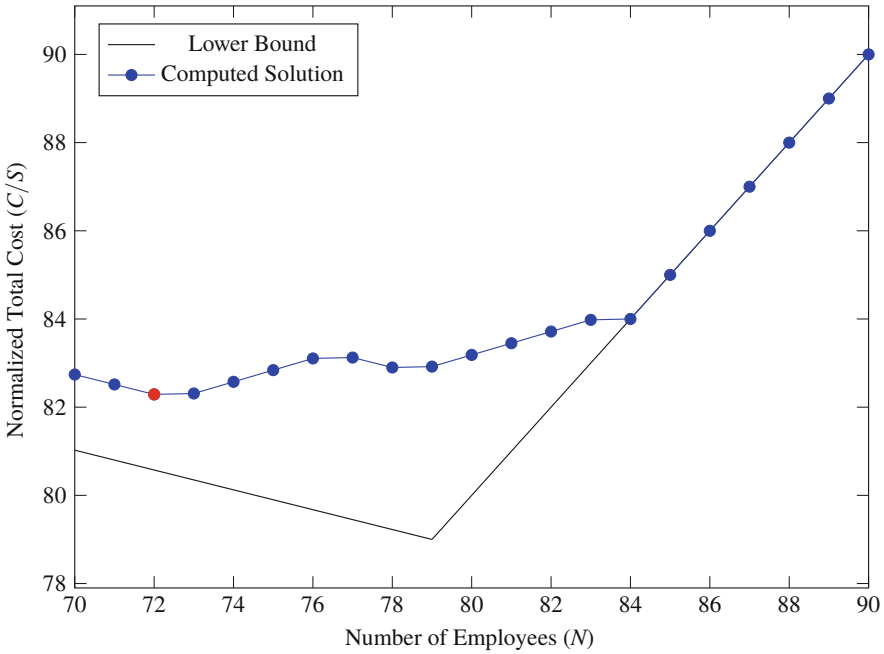


Fig. 1 Variation of total salary with number of employees

Table 2 Shift allocations for $N = 72$ (Regular + Overtime + Excess)

Shifts	Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday
M	$19+0+0$	$16+0+3$	$22+0+0$	$22+0+0$	$22+0+0$	$22+0+0$	$22+0+0$
E	$18+1+0$	$16+0+2$	$14+8+0$	$16+6+0$	$14+8+0$	$18+4+0$	$17+5+0$
N	$14+0+0$	$11+0+2$	$15+1+0$	$14+2+0$	$16+0+0$	$12+4+0$	$13+3+0$
Total	$51+1+0$	$43+0+7$	$51+9+0$	$52+8+0$	$52+8+0$	$52+8+0$	$52+8+0$

needed with 42 of them being overtime shifts. Table 2 shows the total number of persons allocated to each shift. These shifts were classified as regular, overtime and excess. No overtime was allocated on Sunday since Sunday required the smallest total number of allocated staff. Note that the workload is satisfied exactly for all days except Sunday but in many cases overtime shifts are required. On Sunday an additional 7 shifts had to be allocated so that all staff achieve their 40 h per week. However this turns out to be the best alternative.

The number of employees determined, 72, is the number of available employees required to serve the workload. As we saw previously, we need to take into account the fact that employees have vacation, sick days and so the actual number of employees needed can be computed using Eq. (5):

$$N^* = \left\lfloor \frac{72(1+\beta)}{\alpha(1+\kappa)} \right\rfloor \tag{16}$$

Table 3 Run time data

N	No. of decision variables	No. of constraints	Computational time (s)
42	882	1638	0.66
70	1470	2730	0.76
84	1764	3276	0.79
120	2521	4680	0.86
180	3780	7020	0.93

where α, β and κ were previously defined. Let us assume that employees are willing to work a sufficient number of overtime shifts to cover events that require additional manpower and hence that $\beta = \kappa$. The value for α can be determined from historical data but if we assume 3 weeks vacation, an average of 1 week of sick days and 1 week of holidays then we can use the estimate $\alpha = (52 - 5)/52 \approx 0.9$. Substituting these values above we determine that 80 full-time employees are needed for the given workload.

We can also determine a bound on the sub-optimality of the computed solution. The cost computed for scheduling 72 employees is 82.3. The lower bound value for this case is 80.6. Therefore the computed solution is within 2% of the optimal solution and hence the chosen algorithm performs quite well.

5 Tests and Computational Results

The computational time required for this problem is not of great importance since the algorithm needs to be run once per week. However we did monitor run times as well. Table 3 displays computational times for different model sizes, with respect to the number of allocated staff N, number of decision variables and number of constraints. One finds that, even for large problems, run times are less than 1 s.

6 Conclusions

We considered the problem of optimizing the scheduling of staff members for the case of security services. However this approach can be applied to a wide range of rostering or scheduling problems. We demonstrated the trade-off between hiring new staff and having staff work overtime. This trade-off analysis is one of the contributions of the paper. We also took into account various practical constraints which affect the schedule. Some of these are due to negotiated agreements between the Union and Management. Negotiations in these scenarios can become quite heated and so analyses such as those presented here can serve to ease the minds of all parties that a given settlement is fair and efficient.

In future work we plan to use historical data to fine tune the optimization by taking into consideration the variation of workloads and available staff over time. We can then compute the annual cost for a given number of employees and choose the optimal value. In addition, the information gained from this work can be used to better allocate vacation time. We are in the process of developing a cloud-based platform for use by those who presently perform this scheduling function. The historical data used by this platform will be automatically updated so that the solution will self adapt to changes.

References

1. Ang, S.Y., Razali, M., Asyikin, S.N., Kek, S.L.: Optimized preference of security staff scheduling using integer linear programming approach. COMPUSOFT Int. J. Adv. Comput. Tech. **8**, 3103–3111 (2019)
2. Baker, K., Magazine, M.: Workforce scheduling with cyclic demands and day-off constraints. Manag. Sci. **24**, 161–167 (1977)
3. Bechtold, S., Brusco, M., Showalter, M.: Greedy constructive heuristic and local search algorithm for solving nurse rostering problems. Decis. Sci. **22**(4), 683–699 (1991)
4. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.: Julia: a fresh approach to numerical computing. SIAM Rev. **59**, 65–98 (2017)
5. Burgy, R., Michon-Lacaze, H., Desaulniers, G.: Employee scheduling with short demand perturbations and extensible shifts. Omega **89**, 177–192 (2018)
6. Dantzig, G.: A comment on Edies traffic delay at toll booths. Oper. Res. **2**, 339–341 (1954)
7. Easton, F., Rossin, D.F., Brethen, R.H.: Overtime schedules for full-time service workers. Omega **25**(3), 285–299, 6 (1997)
8. Gamache, M., Soumis, F.: A method for optimally solving the rostering problem. Oper. Res. Airline Ind. Internat. Ser. Oper. Res. Manage. Sci. **9**, 124–157 (1998)
9. Ingels, J., Maenhout, B.: The use of overtime to limit the impact of demand variability in personnel scheduling. In: 11th International Conference on Practice and Theory of Automated Timetabling (PATAT 2016), pp. 489–492. PATAT (2016)
10. Ingels, J., Maenhout, B.: The impact of overtime as a time-based proactive scheduling and reactive allocation strategy on the robustness of a personnel shift roster. J. Sched. **21**(2), 143–165 (2018)
11. Santiago, S.: The value of employer benefits (2019). https://www.bankrate.com/finance/financial-literacy/the-value-of-employer-benefits.aspx
12. Woolfenden, S., Stevenson, B.: Establishing appropriate staffing levels for campus public safety departments. COPS US Department of Justice (2011)
13. Yuniartha, D.R., Purnama I.L.I., Herawati, A., Dewi, L.T.: Shift scheduling model considering workload and worker's preference for security department. IOP Conf. Ser.: Mater. Sci. Eng. **337**, 1–6 (2018)

Dynamic Tabu Search for Enhancing the Productivity of a Bottle Production Line

Marie-Sklaerder Vié and Nicolas Zufferey

Abstract Many industries use linear production lines, with buffers between each pair of machines for absorbing small breakdowns or other irregularities. These buffers have different thresholds for triggering the possible speeds of the machines. The goal of this study is to tune the values of these thresholds in order to enhance the productivity of the line. A simulation-optimization approach is proposed and applied to a case study involving a soft-drink plastic bottle company. We show that the production can be increased by a few pallets per day. Such results are appealing for the company, as updating these thresholds does not imply any cost.

Keywords Simulation-optimization · Tabu search · Linear production line

1 Introduction

We consider a representative linear production line of soft-drink plastic bottles for an international company ABC (it cannot be named because of a non-disclosure agreement). The goal is to maximize the production rate of the line over a planning horizon of a shift (8 h). The line consists of a sequence of machines, with a buffer (an accumulation table) between each pair of machines. As the production line is linear, the bottles must be processed by all the machines in the same order, as for a *Permutation Flow-Shop Scheduling Problem* [14]. Each machine can run with a finite number of different speeds (including zero). The speed changes of a machine are triggered by sensors placed on the upstream/downstream buffers, or by physical necessities (e.g., reel change). The only technical constraints are the machine/buffer limitations (e.g., maximum speeds or total capacity). This work aims at tuning the set of possible positions (called the *thresholds*) of the sensors that trigger the speed changes. This optimization is done while taking into account the

M.-S. Vié · N. Zufferey (✉)
GSEM, University of Geneva - Uni Mail, Geneva, Switzerland
e-mail: marie-sklaerder.vie@unige.ch; n.zufferey@unige.ch

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_22

possibility of unplanned stoppages for each machine (e.g., a reel needs changing, a buffer is overloaded, a technicality needs to be corrected, a small breakdown). The probability distributions of such random events are deduced from historical data.

A brief literature review is conducted in Sect. 2. The considered problem is presented in Sect. 3. Relying on tabu search and on a robust production-rate simulator, a simulation-optimization approach is proposed in Sect. 4. Next, some results on a case study are exposed in Sect. 5, and conclusions are highlighted in Sect. 6.

2 Related Literature

In the next paragraph, we investigate the literature connected to our problem (i.e., optimizing a production line, and more precisely with the use of buffers). Second, different pointers on similar simulation-optimization approaches are given. Finally, we present different papers that use a tabu-search method with dynamic adjustments.

Optimizing linear production lines is a well-known topic in the literature. Different case studies have been studied (e.g., schedule a yogurt production line [5], determine the product allocation for a bottling company [2]). There are also numerous papers discussing production lines when buffers are added between pairs of machines (including the situation of unplanned breakdowns), and on the appropriate capacity to choose for each buffer [15, 20]. However, to the best of our knowledge, there is no study on how to calibrate thresholds on these buffers when multiple machine speeds are possible.

The use of simulation-optimization methods for optimizing production lines is often used, in particular when facing machines prone to failure [9], variable replenishment lead-times [16], uncertain demand [10], or uncertain supply [12]. The reader is referred to [1, 17] for an accurate review on: simulation-optimization approaches, the involved algorithms, and the various possible applications. As one can see in these reviews, metaheuristics (e.g., genetic algorithms, simulated annealing, tabu search) have been widely used with simulation-optimization. And, as presented in [13], the descent and tabu-search approaches are particularly adapted when facing perturbations, as they are fast solution methods, and easily adjustable in a changing environment.

In the considered simulation-optimization context, *dynamic tabu search* [22] has been shown to be very efficient. This solution method can be seen as a mix between *tabu search* [6] and *variable neighborhood search* [3, 7], as it adjusts dynamically the neighborhood structures used within the tabu-search framework. Different papers employ this type of dynamic adjustments, as in [11] where the authors use a perturbation that is tuned according to the quality gap between the two last explored solutions, or as in [19] where the authors have proposed strategic oscillations to diversify the search process on the one hand, and adaptive tabu tenures to intensify it on the other hand. A dynamic tabu-search methodology has also been used in [4] for buffer allocation in production lines, but only for

determining the capacity (as previously explained). It is favorably compared with a more standard tabu search.

3 Presentation of the Problem

The components of the production line are first described (i.e., products, machines, and buffers). Next, the optimization problem is formulated (i.e., decision variables, objective function, and constraints).

The considered production line makes soft-drink plastic bottles, which come in different *Stock Keeping Units* (SKU), i.e., in different models. For each SKU, the bottles are batched in packs, then in pallets, but the number of bottles per pack and per pallet differs from one SKU to another, and therefore the possible machine speeds can differ. The line is composed of m different machines, as shown in Fig. 1.

It can for instance involve:

- a *bottle-blowing* machine that forms the bottles,
- a *filler* that fills the bottles with the soft drink,
- a *capper* that closes the bottles,
- a *labeler* that puts the brand sticker on the bottles,
- a *packer* that groups bottles in packs,
- a *palletizer* that groups packs in pallets.

At any time t, each machine i can have its speed $S_i(t)$ in a set of possible speeds Δ_i. Typically, Δ_i is composed of zero, a low speed, a nominal speed, and a high speed. For each machine, a distribution for the production-time interval without any breakdown, and for the duration of a breakdown (both based on historical data), are given. Thanks to that, different breakdown scenarios of 8 h (i.e., a shift) can be computed. Between each pair of consecutive machines, there is a buffer (there are thus $m - 1$ buffers).

The following data is given for each buffer j:

- the maximum capacity C_j,
- the number n_j of thresholds that j has for triggering the different speeds of the upstream/downstream machines,
- the value currently used of each threshold.

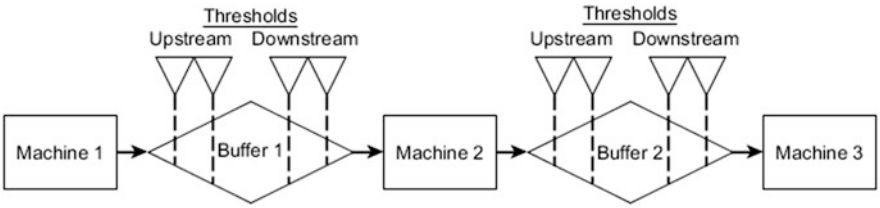


Fig. 1 Considered production line with $m = 3$ machines

For each buffer j, $Q_j(t)$ denotes the number of bottles present on the downstream buffer at time t. Finally, each machine i follows a specific *speed-logic* function that defines its speed considering the current state of the line. A *state* is based on the current speed of the involved machine and on the amount of bottles present in its upstream/downstream buffers (e.g., when the upstream buffer has a low number of bottles or when the downstream buffer has a high number of bottles, the machine reduces its speeds). Formally, each machine i has a specific function

$$g_i(t, S_i(t - \Delta t), Q_{i-1}(t), Q_i(t))$$

that returns its speed $S_i(t) \in \Delta_i$. These functions are not detailed here as they are very specific.

The goal of this study is to tune the thresholds while maximizing the average production rate during a shift. A solution X is defined as the set of threshold values

$$(x_{j,p})_{j\in[1,m-1],\, p\in[1,n_j]}.$$

Each threshold can take values between zero and the total capacity of its associated buffer, while satisfying the *ranking constraint* (i.e., the first threshold must be always lower than the second, etc.). Therefore, a solution has to satisfy the following constraint for each buffer j: $x_{j,1} < x_{j,2} < \ldots < x_{j,n_j}$. Each solution X can be evaluated by computing the average production rate μ_k and its standard deviation σ_k over different scenarios, for all the SKUs k. The following objective function is proposed:

$$f(X) = \sum_{k\in[1,K]} (\mu_k - \sigma_k).$$

It has been designed in order to favor *robust* solutions (i.e., solutions that stay efficient for various breakdowns scenarios), which is an important feature expected by ABC. Indeed, we want solutions that have a high average production rate (see the first part of f)), but with a small standard deviation (see the second part of f), in order to better preserve the planned production rate for most of the possible breakdown scenarios.

4 Simulation-Optimization Approach

The simulator developed to evaluate any solution X is first presented. Next, a *dynamic tabu-search* approach is proposed for the optimization problem.

A solution X can be evaluated for a shift with a chosen breakdown scenario with Algorithm 1. It stimulates the production using a time step $\Delta t = 0.05$ s (as it is the time step used by ABC to save historical data). At each time t, the machine speeds and the number of bottles present in each machine are updated. For each scenario s,

Algorithm 1 Simulator

Input: breakdown scenario s, solution X.

Initialization:

- set time $t = 0$;
- set each buffer $= \emptyset$;
- set each speed $= 0$;
- set $f_s(t) = 0$.

While time $t < 28{,}800$ s (i.e., 8 h), **do**:

1. set $t = t + \Delta t$;
2. implement the breakdowns of scenario s at time t;
3. update the numbers of bottles in each buffer j, according to the current speeds: $Q_j(t) = Q_j(t - \Delta t) + \Delta t \cdot (S_j(t) - S_{j+1}(t))$;
4. update the total number of bottles produced: $f_s(t) = f_s(t) + \Delta t \cdot S_m(t)$;
5. update the speed of each machine i with respect to the speed constraints: $S_i(t) = g_i(t, S_i(t - \Delta t), Q_{i-1}(t), Q_i(t))$.

Output: simulated production rate $f_s(X)$ for scenario s with solution X.

the simulator returns the number $f_s(X)$ of bottles produced. To evaluate a solution X accurately, the simulator is performed with q (parameter) different scenarios for each SKU k, and the average production rate μ_k (with its standard deviation σ_k) is returned at the end. Parameter q is chosen such that the average production rate is guaranteed to a certain precision (typically, a precision of 0.1 pallet is obtained with $q = 100$).

Using the simulator to evaluate any solution, *tabu-search* is proposed to find a hopefully good solution to the problem. The principle of tabu search is to add some diversification to a classic *descent local search*, by allowing some moves (a *move* is a slight modification) that do not improve the current solution, and by avoiding being trapped in local optima. The main idea is to forbid the opposite move of the last performed move during a certain number of iterations (this move being called *tabu*), and to always go to the best non-tabu neighbor solution. When a time limit is reached, tabu search returns the best encountered solution found during its search.

DTS, the proposed *dynamic tabu search*, is summarized in Algorithm 2. It is dynamic in the sense it uses various move amplitudes that are updated in a *variable-neighborhood-search* fashion. Each variable $x_{j,p}$ is associated with a minimum $x_{j,p}^{\min}$ and a maximum $x_{j,p}^{\max}$, based on the ranking constraints and on the C_j's. A move consists of increasing (resp. decreasing) a variable $x_{j,p}$ by an amplitude $\delta_{j,p}^{+}$ (resp. $\delta_{j,p}^{-}$), which is dynamically updated. These amplitudes are first set to $\frac{x_{j,p}^{\max} - x_{j,p}}{2}$ and $\frac{x_{j,p} - x_{j,p}^{\min}}{2}$, which are the average possible amplitudes. This guarantees that the modification of $x_{j,p}$ leads to a feasible solution with respect to the ranking constraint. Also, as the amplitudes will be divided by two at each step they are used, this mechanism will hopefully lead to their appropriate values. Following a sort of dichotomy, the moves become more and more precise as the amplitudes are reduced.

Algorithm 2 Dynamic tabu search *(DTS)*

Initialization:

- set X to the initial solution (i.e., the current threshold values of ABC);
- initialize the best encountered solution and its value: set $X^\star = X$ and $f^\star = f(X)$;
- for each variable $x_{j,p}$, initialize its move amplitudes: $(\delta^-_{j,p}, \delta^+_{j,p}) = (\frac{x_{j,p} - x^{\min}_{j,p}}{2}, \frac{x^{\max}_{j,p} - x_{j,p}}{2})$.

While there is no improvement of $f^\star$ during T minutes, **do:**

1. generate the best non-tabu neighbor solution $X_{j,p}$ by modifying a single decision variable $x_{j,p}$ by $-\delta^-_{j,p}$ or $\delta^+_{j,p}$ (test all the options, i.e., test the augmentation and the reduction of each variable);
2. update the current solution: set $X = X_{j,p}$;
3. update each $(x^{\min}, x^{\max})$ and (δ^-, δ^+) in order to satisfy the ranking constraint;
4. reduce the move amplitudes δ: if $f(X) \leq f^\star$, set $\delta = \omega \cdot \delta$ (for each δ);
5. if $f(X) > f^\star$, set: $X^\star = X$, $f^\star = f(X)$, each δ to its initial value;
6. update the tabu status: $x_{j,p}$ cannot be modified the reverse way (i.e., it cannot be reduced if it has just been augmented, and vice versa) during tab iterations.

From now on, we do not mention the sub/upper-script when it is not necessary to do it. When the values of $(x^{\min}, x^{\max})$ change because of the modification of a variable, the (δ^-, δ^+) are updated (see Step 3) in order to satisfy the ranking constraint. For instance, if $x_{1,2}$ is modified, then $\delta^+_{1,1}$ associated with $x_{1,1}$ needs to be adjusted, as its maximum is now the new value of $x_{1,2}$. Finally, if the current (δ^-, δ^+) did not lead to any improvement, they are reduced thanks to coefficient $\omega \in]0, 1[$ (see Step 4). Otherwise, they are all set back to their initial values (see Step 5). Such a management of the move amplitudes allows to gradually strengthening the search around promising solutions. Indeed the amplitudes are first large, and then smaller and smaller if the solution is not improved. For the experiments, parameters w and tab are set to 0.5 and 5, respectively.

5 Results on a Case Study

First, as computation time is not an issue for this problem (indeed, the thresholds have to be fixed once and for all), we have used large values for the time limit T (typically 2 h). All the algorithms were coded with C++ under Linux, and run on 3.4 GHz Intel Quad-core i7 processor with 8 GB of DDR3 RAM. Real data provided by ABC is used. The studied production line is made of four machines, in the following order:

- a *Combi* (it blows the plastic bottles), its speed can only be in {zero, nominal};
- a *Labeler* (it puts labels on the bottles), its possible speeds are in {zero, low, nominal, high};

- a *Packer* (it groups the bottles in packs), its possible speeds are in {zero, low, nominal, high};
- a *Palletizer* (it groups the packs in pallets), its possible speeds are in {zero, nominal, high}.

To have an idea, the low (resp. nominal and high) speed is around 20,000 (resp. 40,000 and 50,000) bottles per hour. More accurate values cannot be provided because of the non-disclosure agreement. The same remark holds for the other data. For each machine i, its specific speed-logic function g_i is given. For each buffer, there is a typical capacity of 1500 bottles and five thresholds. The current values of these thresholds (i.e., the ones used by ABC nowadays) are initially employed. The considered production line produces six different SKUs, which can have either 8 or 12 bottles per pack, and various numbers of packs per pallet (typically between 150 and 250). Note that both the *Labeler* and the *Packer* work with reels, which have to be changed every 30,000 bottles for the *Labeler*, and every 1500 packs for the *Packer* (this last value being a slightly different for packs of 8 or 12 bottles).

DTS changes mainly the thresholds of the buffers around the *Labeler*. This indicates that this machine was the bottleneck of the line. When compared to the current production rates faced by ABC, the following improvements were achieved. First, for each SKU, the *DTS*-solution always improve the smallest production rate over all the considered scenarios. This shows that the proposed solutions react better to breakdowns than the ABC-solutions. For each SKU k, the percentage gains on the average production rate μ_k and on its standard deviation σ_k are presented in Table 1. On average (i.e., considering all the SKUs), the production rate has been improved by 0.5%. The smallest (resp. largest) encountered improvement over all the considered SKUs is 0.40% (resp. 0.53%). In addition, for each SKU, there exists no scenario for which the *DTS*-solution is worse than the ABC-solution. Moreover, the average standard deviation is reduced by around 4% per SKU. Last but not least, it is important to be aware that a production-rate improvement of 0.5% represents a production augmentation ranging between 1 and 2 pallets per shift, which is very significant for ABC (typically, 6000 additional bottles per day). As this gain is obtained only by changing the thresholds, there is no cost for implementing the proposed solutions.

Table 1 Percentage gains of *DTS*-solutions with respect to ABC-solutions

SKU k	Production-rate gain (μ_k)	Standard-deviation gain (σ_k)
1	0.46%	−4.78%
2	0.44%	−3.93%
3	0.53%	−4.52%
4	0.44%	−4.55%
5	0.40%	−4.29%
6	0.53%	−4.35%

6 Conclusion

In this paper, a simulation-optimization approach was proposed for increasing the production rate of a soft-drink production line. Even if the considered decision variables are limited (i.e., only the thresholds that trigger the speed changes of the machines can be updated), it is showed that promising improvements can be achieved for real instances provided by a company. This optimization is strategic, as it can be deployed to the other production lines of the company (hundreds of production lines in the world, with production rates up to thousands bottles per hour). Among the possible avenues of research, one can investigate other types of decision variables (e.g., set of possible speeds for the machines, number of possible thresholds for the buffers) combined with the use of filtering techniques for determining efficient domain values [8, 21], or the consideration of an *order-acceptance-and-scheduling* [18] version of this problem.

References

1. Amaran, S., Sahinidis, N.V., Sharda, B., Bury, S.J.: Simulation optimization: a review of algorithms and applications. Ann. Oper. Res. **240**(1), 351–380 (2016)
2. Balogun, O.S., Jolayemi, E.T., Akingbade, T.J., Muazu, H.G.: Use of linear programming for optimal production in a production line in Coca Cola bottling company, Ilorin. Int. J. Eng. Res. Appl. **2**(5), 2004–2007 (2012)
3. Bierlaire, M., Thémans, M., Zufferey, N.: A heuristic for nonlinear global optimization. INFORMS J. Comput. **22**(1), 59–70 (2010)
4. Demir, L., Tunal, S., Eliiyi, D.T.: An adaptive tabu search approach for buffer allocation problem in unreliable non-homogenous production lines. Comput. Oper. Res. **39**(7), 1477–1486 (2012)
5. Doganis, P., Sarimveis, H.: Optimal scheduling in a yogurt production line based on mixed integer linear programming. J. Food Eng. **80**(2), 445–453 (2007)
6. Glover, F., Laguna, M.: Tabu search. In: Handbook of Combinatorial Optimization, pp. 2093–2229. Springer, Berlin (1998)
7. Hansen, P., Mladenovic, N.: Variable neighborhood search. In: Handbook of Metaheuristics, pp. 145–184. Springer, Berlin (2003)
8. Hertz, A., Schindl, D., Zufferey, N.: Lower bounding and tabu search procedures for the frequency assignment problem with polarization constraints, 4OR **3**(2), 139–161 (2005)
9. Kenne, J.P., Gharbi, A.: A simulation optimization approach in production planning of failure prone manufacturing systems. J. Int. Manag. **12**(5–6), 421–431 (2001)
10. Lin, J.T., Chen, C.M.: Simulation optimization approach for hybrid flow shop scheduling problem in semiconductor back-end manufacturing. Simul. Model. Pract. Theory **51**, 100–114 (2015)
11. Lü, Z., Hao, J.-K.: Adaptive tabu search for course timetabling. Eur. J. Oper. Res. **200**(1), 235–244 (2010)
12. Osorio, A.F., Brailsford, S.C., Smith, H.K., Forero-Matiz, S.P., Camacho-Rodriguez, B.A.: Simulation-optimization model for production planning in the blood supply chain. Health Care Manag. Sci. **20**(4), 548–564 (2017)
13. Respen, J., Zufferey, N., Wieser, P.: Three-level inventory deployment for a luxury watch company facing various perturbations. J. Oper. Res. Soc. **68**(10), 1195–1210 (2017)

14. Ruiz, R., Stützle, T.: A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. Eur. J. Oper. Res. **177**(3), 2033–2049 (2007)
15. Shi, C., Gershwin, S.B.: An efficient buffer design algorithm for production line profit maximization. Inter. J. Prod. Econ. **122**(2), 725–740 (2009)
16. Silver, E., Zufferey, N.: Inventory control of an item with a probabilistic replenishment lead time and a known supplier shutdown period. Inter. J. Prod. Res. **49**(4), 923–947 (2011)
17. Tekin, E., Sabuncuoglu, I.: Simulation optimization: a comprehensive review on theory and applications. IIE Trans. **36**(11), 1067–1081 (2004)
18. Thevenin, S., Zufferey, N., Widmer, M.: Order acceptance and scheduling with earliness and tardiness penalties. J. Heuristics **22**(6), 849–890 (2016)
19. Wang, Y., Wu, Q., Punnen, A.P., Glover, F.: Adaptive tabu search with strategic oscillation for the bipartite Boolean quadratic programming problem with partitioned variables. Inf. Sci. **450**, 284–300 (2018)
20. Weiss, S., Matta, A., Stolletz, R.: Optimization of buffer allocations in flow lines with limited supply. IISE Trans. **50**(3), 191–202 (2018)
21. Zufferey, N.: Heuristiques pour les Problèmes de la Coloration des Sommets d'un Graphe et d'Affectation de Fréquences avec Polarités. PhD Thesis, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland (2002)
22. Zufferey, N.: Dynamic tabu search with simulation for a resource allocation problem within a production environment. In: Proceedings of 4th International Conference on Metaheuristics and Nature Inspired Computing, META 2012, Sousse, pp. 27–31 (2012)

Swap Minimization in Nearest Neighbour Quantum Circuits: An ILP Formulation

Maurizio Boccia, Adriano Masone, Antonio Sforza, and Claudio Sterle

Abstract Quantum computing (*QC*) represents a great challenge for both academia and private companies and it is currently pursuing the development of quantum algorithms and physical realizations of quantum computers. Quantum algorithms exploit the concept of quantum bit (*qubit*). They are implemented by designing circuits which consider an ideal quantum computer where no interaction restriction between qubits is imposed. However, physical realizations of quantum computers are subject to several technological constraints and adjacency between interacting qubits is one of the most common one. To this end, additional gates, referred to as *swap*, can be added to a quantum circuit to make it *nearest neighbour compliant*. These additional gates have a cost in terms of reliability of the quantum system, hence their number should be minimized. In this paper we first give some hints about this cutting edge topic. Then we provide a review of the literature solving approaches for the swap minimization problem in quantum circuits and propose an integer linear programming formulation for it. We conclude with some preliminary results on small test instances and future work perspectives.

Keywords Nearest neighbor quantum circuit · SWAP minimization · ILP formulation

M. Boccia · A. Masone · A. Sforza
Department of Electrical Engineering and Information Technology, University Federico II of Naples, Naples, Italy
e-mail: maurizio.boccia@unina.it; adriano.masone@unina.it; antonio.sforza@unina.it

C. Sterle (✉)
Department of Electrical Engineering and Information Technology, University Federico II of Naples, Naples, Italy

Institute for Systems Analysis and Computer Science, CNR, Rome, Italy
e-mail: claudio.sterle@unina.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_23

1 Introduction

Non-conventional computing paradigms are continuously drawing the attention of both academia and private companies. They represent the alternative to the classical computing paradigm based on electrical circuits phenomenon. In this context quantum computing (*QC*) represents a great challenge. It replaces the bit with the *quantum bit* (qubit) concept (its mathematical definition will be given in what follows). Roughly speaking a qubit can be defined as the simplest two-state quantum-mechanical system which can be implemented running a quantum physics experiment exploiting the quantum-mechanical properties of particles to perform the required computation. These properties, specifically *superposition* and *entanglement*, are the key enabling factors of the *QC* performance [14].

The theoretical advantages of *QC* paradigm has been largely accepted in recent years [6] and it has already been proven in several applications, such as: Shor's algorithm for factorization; Groover's database search; hard optimization problems; public-key cryptosystem; solution of a large system of linear equations, etc.. Hence, the interest of academia and private companies is rising together with the investments to pursue the research and development in this field. Let us recall that in 2015 D-Wave Systems installed the 2X quantum annealer (with more than 1000 qubits) at the Quantum Artificial Intelligence Lab at NASA Ames Research Center and, recently, free access to several quantum universal processors/simulators is spreading. Among the others we just cite here Quantum Computing Playground, a browser-based WebGL Chrome Experiment offered by Google [9], and IBM Q, a cloud-enabled quantum computing platform offered by IBM, which allows access to several quantum processors [7]. In 2017 IBM launched also the IBM Q Awards, i.e., a series of prizes for professors, lecturers and students who use the IBM Q Experience and QISKit in order to promote teaching and research activities on this cutting edge topic.

The available *QC* resources allow researchers and practitioners to develop and test their own quantum algorithms. A quantum algorithm is described by a quantum circuit, i.e., a sequence of quantum gates (analogues to classical logic gates) operating on one or several qubits and transforming the initial state of the system into its final state [5, 8, 10]. Quantum algorithms usually consider an ideal quantum computer, thus no interaction restriction between qubits is imposed. However, a non-ideal quantum computer has several limitations due to environmental disturbances (noise and decoherence) which can result in computational errors and, more importantly, in physical implementation constraints. In this context it is fundamental the development of tools allowing to achieve quantum circuit synthesis that is scalable, reliable and physically feasible [2, 15].

Several physical implementations of universal quantum computers are currently available and still under investigation, e.g.: superconducting circuits, trapped ions or electrons, optical lattices, nuclear magnetic resonance of molecules in solution, nuclear spin, etc. Some of them present various technological limitations. The limited distance between interacting qubits is one of the most critical and common

one. Indeed, although arbitrary-distance interaction between qubits is possible, in physical implementation of quantum circuits a long-distance interaction between two qubits has to be avoided because it represents a noise source. Therefore multiple quantum technologies require the interacting qubits to be arranged in a nearest neighbour (*NN*) fashion. In other words, when implementing the gates of a quantum circuit, the logical qubits have to be mapped to the physical qubits. Since it is not possible to determine a mapping such that all the interacting qubits are adjacent throughout the whole circuit, this mapping may change over time. To this end additional gates, referred to as *swap*, are added to a quantum circuit to make it *nearest neighbour compliant*, *NNC*. These additional gates have a cost in terms of reliability of the system and they affect the execution time of the quantum algorithm. Hence, their number should be kept as small as possible.

The problem of Minimizing the Number of Swaps to make *NNC* a quantum circuit will be denoted in the following as *MNS-NNC*. It has been widely studied in recent years and several exact methods, mainly based on integer linear programming (ILP) models and heuristic optimization approaches have been proposed. These approaches generally exploit the knowledge of the specific physical layout of a quantum computer. Indeed, the physical qubits of a *QC* technology can be arranged in 1D, 2D and 3D dimensional architectures, differing in the maximum number of neighbours per qubit: 2, 4 and 6 with a respect to a line, two-dimensional square and three dimensional square graph, respectively. The usage of these regular topologies is motivated by their scalability. Regardless of the dimensions of the physical architecture, the *MNS-NNC* problem is *NP-hard* since it may be considered as a particular variant of the minimum linear arrangement problem (*MinLA*), which consists in determining a labeling of the vertices of a graph minimizing the sum of the absolute values of the differences between the labels of adjacent vertices [3]. The 1D case is the most investigated one. Exact approaches can be found in [19, 20] and heuristic approaches in [1, 15, 16, 18], respectively. The 2D case is tackled in [2, 17]. Finally, 3D and multi-dimensional case are studied in [4, 11, 13].

A discussion apart is needed for the work presented in [22]. It is related to the winner algorithm of the IBM QISKit Developer Challenge, whose aim was writing the best compiler code in Python or Cython that takes an input quantum circuit and outputs an optimal circuit for a given arbitrary hardware topology. The authors presented a very effective multi-step heuristic approach which utilizes a depth-based partitioning and a state-space search algorithm. The approach, tested on instances with up to 20 qubits, outperforms IBM own mapping solution either in the quality of the solution or in the computation time [21]. Its implementation is publicly available at http://iic.jku.at/eda/research/ibm_qx_mapping.

In this context, this paper is aimed at presenting some introductory hints about *QC* and its basics, drawing the attention of the operations research community on the *MNS-NNC* problem. Then, it proposes an ILP formulation which optimally solves the problem on small arbitrary architecture topologies.

The paper is structured as follows: in Sect. 2 we provide the *QC* theory background and the problem definition; Sect. 3 is devoted to the presentation of

the proposed ILP formulation; in Sect. 4, preliminary computational results are reported, together with some hints about future work perspectives.

2 Background and Problem Definition

The basic difference between classical and quantum computing paradigm is in the computing unit, i.e., the bit and the quantum bit (qubit), respectively. A bit can exist in one of two states 0 or 1, instead, a qubit can exist in multiple states that can be expressed as a linear combination (*superposition*) of them. In other words, a qubit is a mathematical object representing a two level quantum system described by a two dimensional complex Hilbert space, where the two orthogonal quantum states, $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ (ground state) and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ (excited state), are used to represent the Boolean values 0 and 1. Using the bra-ket notation $|\cdot\rangle$, introduced by Dirac, any state of a qubit may be written as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β, referred to as amplitudes, are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$. The peculiarity of the superposition is in the measurement phase, since it returns the value $|0\rangle$ and $|1\rangle$ with probability $|\alpha|^2$ and $|\beta|^2$, respectively. Two qubits can interact in complex ways due to the superposition and the entanglement, i.e., the physical phenomenon occurring when pairs or groups of particles interact in ways such that the state of each particle cannot be described independently of the state of the others.

Quantum computing relies on manipulating qubits by performing the so called quantum gates, mathematically expressed as unitary operations. Hence, in other words, an n-qubit quantum gate performs a specific $2^n \times 2^n$ unitary operation on n qubits implementing the tensor product of related matrices. Details on these gates and matrices are not relevant in the remainder of this work, but can be found in [14].

A *quantum algorithm* is described by a *quantum circuit*, i.e., a set of quantum gates applied in sequence to transform the initial state of the quantum system into a final state. Each gate can involve an arbitrary number of qubits. The resulting circuit is then compiled into another equivalent quantum circuit based on a library of primitive one- and two-qubit gates [12]. A two-qubit gate is identified by a couple of qubits, referred to as control and target qubit, respectively. Hence, it implicitly defines a direction of the operation from the first qubit to the second. This equivalent quantum circuit is the input to our problem. Moreover a quantum circuit can be decomposed in a series of layers, where each layer includes a set of quantum gates such that any two pair of gates in the same level do not operate on any common qubit. Hence all the gates of a layer can be executed in parallel.

Given a quantum algorithm with only primitive gates, one should map the related quantum circuit into a quantum apparatus, i.e., a physical implementation (architecture) of the experiment realizing the circuit. The quantum circuit can be modelled by graph, referred to as *interaction graph* (*IG*), where nodes denote qubits of the circuit and arcs the two-qubit gates to be implemented. The architecture of a quantum computing system can be described by graph as well, referred to as

connectivity graph (*CG*), where nodes represent qubits and arcs represent adjacent qubit pairs where gates can be implemented on.

Current technologies for *QC* often require that all the two-qubit gates composing an *IG* can be implemented if and only if the interacting qubits are positioned in adjacent nodes of the *CG* under investigation, i.e., the quantum circuit has to be *nearest neighbour compliant* (*NNC*). Accordingly a complete *CG* would be an ideal quantum architecture with no restriction on qubit interactions. However, as explained in Sect. 1, the available physical architectures consider 1D, 2D and 3D arrangements. In these cases it is not possible to determine a mapping such that all the interacting qubits are adjacent throughout the whole circuit. Hence the mapping may change over time. To this end, *swap* gates, physically exchanging the locations of the involved qubits, are added to a quantum circuit to make it *NNC*. These gates represent an additional implementation cost for a quantum circuit. Hence, the problem of Minimizing the Number of Swaps to make a quantum circuit *NNC*, *MNS-NNC*, arises. Given an architecture *CG* and quantum circuit *IG* with related primitive gates classified in layers, two main variants can be defined:

- *MNS-NNC-1*: determine the mapping of the *IG* to the *CG* and the sequence of swap gates required to modify the location of the qubits in order to make the circuit *NNC* in two successive layers.
- *MNS-NNC-2*: determine the mapping of the *IG* to the *CG* and the sequence of swap gates required to modify the location of the qubits in order to make the circuit *NNC* in all the layers composing it.

Obviously problem *MNS-NNC-1* is included in *MNS-NNC-2*. Figure 1 sketches the *MNS-NNC* problem presented in [20] for a 1D architecture. In particular, Fig. 1a reports a quantum circuit composed of five two-qubit gates (CNOT) where g_1, g_4, and g_5 are not adjacent. The control and target qubit of the CNOT gates are represented by $\bullet$ and $\oplus$, respectively. Figure 1b shows the insertion of seven swap gates to make the circuit *NNC*. The qubits involved in the swap are represented by $\times$. Finally, Fig. 1c shows an optimized solution in terms of number of swaps, obtained by a different initial and final positioning of the qubits. It is intuitive to understand that a multi-dimensional architecture and, more in general, arbitrary topology with a high average connectivity degree of its nodes, allows to more easily achieve the adjacency requirement, but, as said above, these graphs can pose several

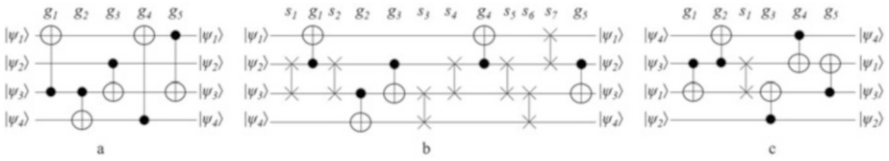


Fig. 1 (**a**) General quantum circuit. (**b**) *NNC* quantum circuit (expensive solution with five swap gates). (**c**) *NNC* quantum circuit (cheap solution with one swap gate)

technological constraints and scalability restrictions. This motivates the interest in simple and regular 1D, 2D and 3D architectures.

3 ILP Formulation for the *MNS-NNC* Problem

In this section we present the proposed integer linear programming (ILP) formulation for the *MNS-NNC* problem. Let us consider a connectivity graph, denoted by $G = (V, A)$, where V and A represent the set of physical nodes and directed links of a quantum computer architecture. For the sake of clarity we highlight that in the following we will denote a generic arc either by a, $a \in A$, or by its extreme vertices (i, j), $i, j \in V$. Moreover, let $G'(V, E)$ be an undirected graph, where $< i, j > \in E \Leftrightarrow (i, j) \in A \vee (j, i) \in A$. Let Q be the set of qubits of the quantum circuit *IG* to be positioned (mapped) in the V nodes. Moreover let us define C as the set of two-qubit gates composing the quantum circuit. As explained above, each gate is represented by an ordered couple of qubit, e.g., $c = (a, b)$, where a is the control qubit and b is the target qubit. Given a generic gate $c = (a, b) \in C$, it can be implemented if and only if the qubits a and b are positioned in adjacent nodes, i.e., being i and j the nodes where qubits a and b are positioned, respectively, then the gate $c = (a, b)$ can be implemented if and only if either arc $(i, j) \in A$ (*direct implementation*) or arc $(j, i) \in A$ (*reverse implementation*). In other words, by direct and reverse implementation we indicate whether the gate is implemented on an arc of *CG* coherently with its direction or in the opposite direction. More formally, direct implementation occurs when $(i, j) \in A$ whereas reverse implementation occurs when $(i, j) \notin A$ but $< i, j > \in E$. This distinction is required since a different computational burden (cost) is associated to each implementation. In the following we will denote these costs as K_c^+ and K_c^- ($K_c^+ < K_c^-$) for direct and reverse implementation, respectively.

The set C can be partitioned in disjoint subsets $C_h \subseteq C$ each of them associated to a layer $h \in H$. It is possible to implement a swap gate between two consecutive layers to exchange the position of two qubits which are adjacent in the graph G'. We recall that each swap gate represents an additional cost (denoted as S) in the implementation of a quantum circuit. Now let us also define the concept of time horizon for the *MNS-NNC* problem. Each layer can be subdivided in time instants. At each instant we can implement in parallel either the gates composing the layer (whose qubits are in adjacent nodes) or the required swap moves. The maximum number of instants of each layer is indicated by m and is defined on trial. Hence, the time horizon of a *MNS-NNC* problem can be defined as: $O = \{t_1^1, \ldots, t_m^1, \ldots, t_1^{|H|}, \ldots, t_m^{|H|}\}$ where the first m instants are related to layer 1, the second m instants to layer 2 and so on. In the following, we will indicate the instants associated to a generic layer h as $O_h = \{t_1^h, \ldots, t_m^h\}$.

Given this setting, *MNS-NNC* consists in determining for each time instant of a time horizon O, the optimal mapping of the quantum circuit *IG* to the architecture

CG, such that all the gates of each layer can be implemented, minimizing the overall computation costs, given by the sum of direct/reverse implementation and swap costs.

In order to mathematically formulate the problem, the following binary variables have to be defined:

- $y^t_{ac}, c \in C, a \in A, t \in O$: binary variable equal to 1 if the gate c is direct implemented on the arc (i, j) at time t , 0 otherwise.
- $y^{*t}_{ac}, c \in C, a \in A, t \in O$: binary variable equal to 1 if the gate c is reverse implemented on the arc (i, j) at time t, 0 otherwise.
- $x^t_{qi}, q \in Q, i \in V, t \in O$: binary variable equal to 1 if the qubit q is positioned in node i at time t, 0 otherwise.
- $z^t_{ij}, (i, j) \in A, t \in O$: binary variable equal to 1 if a swap is implemented between a qubit positioned in i and a qubit positioned in j, or viceversa, at time t, 0 otherwise.

On this basis, the following ILP model can be defined:

$$Min\ z = \sum_{a\in A}\sum_{c\in C}\sum_{t\in T} K^+_c y^t_{ac} + \sum_{a\in A}\sum_{c\in C}\sum_{t\in T} K^-_c y^{*t}_{ac} + \sum_{<i,j>\in E}\sum_{t\in T} S z^t_{ij} \tag{1}$$

subject to

$$x^{t^h_1}_{pi} + x^{t^h_1}_{qj} \geq 2y^{t^h_1}_{ac} \qquad \forall c = (p, q) \in C_h; h \in H; a = (i, j) \in A \tag{2}$$

$$x^{t^h_1}_{pj} + x^{t^h_1}_{qi} \geq 2y^{*t^h_1}_{ac} \qquad \forall c = (p, q) \in C_h; h \in H; a = (i, j) \in A \tag{3}$$

$$\sum_{a\in A}(y^{t^h_1}_{ac} + y^{*t^h_1}_{ac}) = 1 \qquad \forall c = (p, q) \in C_h; h \in H; \tag{4}$$

$$\sum_{q\in Q} x^t_{qi} \leq 1 \qquad \forall i \in V; t \in O \tag{5}$$

$$\sum_{i\in V} x^t_{qi} = 1 \qquad \forall q \in Q; t \in O \tag{6}$$

$$x^{t+1}_{qj} + x^t_{qi} \leq 1 + z^t_{ij} \qquad \forall t \in \{1, .., |O| - 1\}; (i, j) \in A; q \in Q \tag{7}$$

$$x^{t+1}_{qi} + x^t_{qj} \leq 1 + z^t_{ij} \qquad \forall t \in \{1, .., |O| - 1\}; (i, j) \in A; q \in Q \tag{8}$$

$$x^{t+1}_{qi} + x^t_{qj} \leq 1 \qquad \forall t \in \{1, .., |O| - 1\}; i, j \in V, (i, j) \wedge (j, i) \notin A; q \in Q \tag{9}$$

$$z^{t^h_1}_{ij} \leq \sum_{l:(l,i)\in A} z^{t^h_1-1}_{li} + \sum_{r:(j,r)\in A} z^{t^h_1-1}_{jr} \quad \forall t \in \{2, .., |O|\}, t \neq t^h_1; h \in H; (i, j) \in A \tag{10}$$

The objective function (1) minimizes the sum of three cost components: direct, reverse and swap costs. Constraints (2) and (3) impose two conditions: a gate can be implemented on an arc $a = (i, j)$ or on its reverse arc $a = (j, i)$ just in the first time instant of its layer; a gate can be implemented on an arc if and only if the nodes i and j contain the qubits composing it. Constraints (4) impose that each gate composing a layer has to be implemented, either on an arc or on its reverse. Constraints (5) and (6) impose that, at each time instant, each node can contain no more than one qubit and each qubit has to be positioned in just one node. Constraints (7) and (8) impose that at each instant t the qubit contained in a node i can move to an adjacent node j just if the variable $z_{ij}^t = 1$. Constraints (9) impose that at each time instant, the qubit contained in a node i cannot move to a not adjacent node. Finally, constraints (10) are symmetry breaking constraints. These constraints impose that given a layer h, a generic swap between two nodes i and j can be performed in a time instant different from the first if and only if another swap occurred already involving the nodes i and j. Constraints about the domain of the variables are not reported for the sake of brevity.

It is important to note that some modeling decisions are made to reduce symmetries in the solutions provided by the formulation. Indeed, the constraints (2) and (3) avoid the generation of solutions differing just in the time instant when the gates c of a layer h, $c \in C_h$, are implemented. Concerning instead the constraints (10), a small example is provided to explain them. Let us consider two consecutive layers, e.g. the first and the second and let us suppose that, in order to implement the gates of the second layer we have to make the following swaps: $i \rightarrow j$ and $l \rightarrow m \rightarrow n$. This can be achieved performing in parallel the swaps $i \rightarrow j$ and $l \rightarrow m$ in the first time instant of the first layer. Then we can move $m \rightarrow n$. It is easy to understand that several different solutions, equivalent in terms of swap costs can be generated performing the three swap moves in different time instants of the same layer. For this reason, symmetry breaking constraints are needed. In this perspective, with respect to the third cost component, we can also consider to slightly modify the objective function introducing a lexicographic term.

The proposed formulation allows to solve both variants of the *MNS-NNC* problem. It differs from the other formulations present in literature in two main aspects: it is more compact if compared to other formulation, such as the one presented in [4]; it explicitly considers, not only the swap costs, but also the direct and the reverse implementation costs of each gate. Moreover, it does not require the knowledge of an initial positioning of the qubits within the quantum architecture. This requirement, present in most of the exact and heuristic methods present in literature, has a twofold effect. On one side, it reduces the complexity of the problem. On the other side, it significantly affects the quality of the overall solution, since a wrong initial positioning can provide much higher swap costs.

4 Computational Results and Work Perspectives

In this section we present some preliminary results obtained on several instances taken from the IBM Q platform [7]. We have to highlight that a comparison with the performance of other ILP formulations and methods cannot be made since the results are significantly affected by the usage of different objective functions and synthesis of the quantum circuits. A standardization is required to fix this issue in the future. The following parameter values have been used: $K^+ +_c = 10$, $K_c^- = 11/12$ and $S = 34$. These costs take into account the computational burden required to implement the gates using only primitive one- and two-qubit gates.

The model has been experienced on different connectivity graphs of the IBM QISKit Developer Challenge [7]: linear (regular and random); circular (regular and random); Tenerife and Yorktown. All these graphs are composed of 5 nodes and use 5 qubits to implement the gates composing the circuit. Regular and random graphs differ in the orientation of the links. In regular graphs, links are always directed coherently with the topology, from the lower index node to the higher one (i.e. arc (i, j) exists if $i < j$). In random graphs, orientation follows no specific rule.

The value of m has been set equal to the length (in terms of number of links) of the greatest shortest path among all the origin-destination pairs in the non-oriented graph $G'(V, E)$. Even if this value is not an upper bound on the maximum number of time instants to be used for the implementation of all the required swap gates for a layer, in the performed experimentation it was large enough to allow solution feasibility.

The experiments have been performed on an Intel(R) Core(TM) i7-8700k, 3.70 GHz, 16.00 GB of RAM, using Cplex 12.7 with default setting as MIP solver, with a maximum computing time of 3 h.

For each instance we report in Table 1 the following information: the size in terms of number of gates and layers; the optimal solution determined by the proposed formulation; the computation time in seconds.

Table 1 Computational results

Name	Topology	Arcs	Opt sol.	*Time*
Out-circle-rand-q5-2	Circle rand	5	353	339.23
Out-circle-reg-q5-0	Circle reg	5	354	482.95
Out-circle-reg-q5-6	Circle reg	5	354	706.39
Out-linear-rand-q5-5	Linear rand	4	420	6363.89
Out-linear-reg-q5-1	Linear reg	4	386	5614.73
Out-ibmqx2-q5-0	Yorktown	6	419	7162.21
Out-ibmqx2-q5-8	Yorktown	6	419	7446.28
Out-ibmqx2-q5-9	Yorktown	6	419	6333.5
Out-ibmqx4-q5-3	Tenerife	6	353	1527.2
Out-ibmqx4-q5-4	Tenerife	6	419	6298.61
Out-ibmqx4-q5-7	Tenerife	6	419	8643.34

We note that the proposed formulation is able to determine the optimal solution of all the tested instances within the prefixed computation time. Nevertheless, despite the small dimension of the instances, the computation time is between 1.5 and 2 h for most of them. This computational burden is coherent and slightly lower with the one of other ILP formulations present in literature on similar instances (e.g. [4]). However, since the number of quibts in a quantum computer architecture is increasing (up to 50 and more in the future), the usage of the proposed ILP model, to optimally solve the *MNS-NNC* problem on large instances, appears impracticable. In this context, this work has to be considered as an introductory work for the problem. Future research perspectives should be aimed either at strengthening/improving the proposed formulation or proposing more effective optimal solving methods, which can be used in the evaluation of the quality of the solutions provided by the heuristic algorithms present in literature.

References

1. Alfailakawi, M., Alterkawi, L., Ahmad, I., Hamdan, S.: Line ordering of reversible circuits for linear nearest neighbor realization. Quantum Inf. Process. **12**(10), 3319–3339 (2013)
2. Alfailakawi, M.G., Ahmad, I., Hamdan, S.: Harmony-search algorithm for 2D nearest neighbor quantum circuits realization. Expert Sys. Appl. **61**, 16–27 (2016)
3. Andrade, R., Bonates, T., CampÃłlo, M., Ferreira, M.: Minimum linear arrangements. Electron Notes Discrete Math. **62**, 63–68 (2017)
4. Bhattacharjee, D., Chattopadhyay, A.: Depth-optimal quantum circuit placement for arbitrary topologies. arXiv:1703.08540 (2017)
5. Chakrabarti, A., Sur-Kolay, S.: Nearest neighbour based synthesis of quantum boolean circuits. Eng. Lett. **16**, 1–5 (2007)
6. Courtland, R.: Google aims for quantum computing supremacy (news). IEEE Spectr. **54**(6), 9–10 (2017)
7. Gadi, A., et al.: Qiskit: An open-source framework for quantum computing (2019). https://doi.org/10.5281/zenodo.2562110
8. Garcia-Escartin, J.C., Chamorro-Posada, P.: Equivalent quantum circuits. arXiv:1110.2998 (2011)
9. Google LLC: Quantum computing playground (2014). http://www.quantumplayground.net
10. Khan, M.H.A.: Cost reduction in nearest neighbour based synthesis of quantum boolean circuits. Eng. Lett. **16**, 1–5 (2008)
11. Kole, A., Datta, K., Sengupta, I.: A new heuristic for n-dimensional nearest neighbor realization of a quantum circuit. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **37**(1), 182–192 (2015)
12. Li, Z., Chen, S., Song, X., Perkowski, M., Chen, H., Zhu, W.: Quantum circuit synthesis using a new quantum logic gate library of NCV quantum gates. Int. J. Theor. Phys. **56**(4), 1023–1038 (2017)
13. Lye, A.,Wille, R., Drechsler, R.: Determining the minimal number of swap gates for multi-dimensional nearest neighbor quantum circuits. In: 19th Asia and South Pacific Design Automation Conference, ASP-DAC, pp. 178–183 (2015)
14. Nielsen, M.A., Chuang, I.: Quantum Computation and Quantum Information, 10th edn. Cambridge University Press, Cambridge (2010)
15. Saeedi, M., Wille, R., Drechsler, R.: Synthesis of quantum circuits for linear nearest neighbor architectures. Quantum Inf. Process **10**(3), 355–377 (2011)

16. Shafaei, A., Saeedi, M., Pedram, M.: Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures. In: Proceedings of the 50th Annual Design Automation Conference, pp. 41–46. ACM, New York (2013)
17. Shafaei, A., Saeedi, M., Pedram, M.: Qubit placement to minimize communication overhead in 2D quantum architectures. 19th Asia and South Pacific Design Automation Conference, ASP-DAC, pp. 495–500 (2014)
18. Tan, Y.Y., Cheng, X.Y., Guan, Z.J., Liu, Y., Ma, H.L.: Multi-strategy based quantum cost reduction of linear nearest-neighbor quantum circuit. Quantum Inf. Process. **17**(3), 61 (2018)
19. Wille, R., Lye, A., Drechsler, R.: Exact reordering of circuit lines for nearest neighbor quantum architectures. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **33**(12), 1818–1831 (2014)
20. Wille, R., Lye, A., Drechsler, R.: Optimal swap gate insertion for nearest neighbor quantum circuits. In: 19th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 489–494. IEEE, Piscataway (2014)
21. Zulehner, A., Wille, R.: Compiling SU (4) quantum circuits to IBM QX architectures. In: Proceedings of the 24th Asia and South Pacific Design Automation Conference, pp. 185–190 (2019)
22. Zulehner, A., Paler, A., Wille, R.: An efficient methodology for mapping quantum circuits to the IBM QX architectures. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **37**(1), 182–192 (2018)

A Traffic Equilibrium Nonlinear Programming Model for Optimizing Road Maintenance Investments

Mauro Passacantando and Fabio Raciti

Abstract We consider a traffic network in which some of the roads need maintenance jobs. Due to budget constraints not all of the roads can be maintained and a central authority has to choose which of them are to be improved. We propose a nonlinear programming model where this choice is made according to its effects on the relative variation of the total cost, assuming that users behave according to Wardrop equilibrium principle. To assess the network improvement after maintenance we use the Bureau of Public Road link cost functions.

Keywords Traffic network · Wardrop equilibrium · Investment optimization · Braess paradox

1 Introduction

Let us consider a traffic network where some of the roads need maintenance, or improvement jobs. However, the available money to be invested in the improvement of the road network is not sufficient for all the roads and a central authority has to decide which of them is better to maintain. In this regard, it is important to assess the impact that the improvement of a single road, or of a group of roads, has on the overall network efficiency. The efficiency index that we use is the relative variation of total travel time on the network under the assumption that flows are distributed according to Wardrop equilibrium principle. This means that travelers choose the roads so as to minimize their journey time, and all of the paths actually used to reach a certain destination from a given origin give rise to the same travel time. The total

M. Passacantando
Department of Computer Science, University of Pisa, Pisa, Italy
e-mail: mauro.passacantando@unipi.it

F. Raciti (✉)
Department of Mathematics and Computer Science, University of Catania, Catania, Italy
e-mail: fraciti@dmi.unict.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_24

travel time is often considered as a "social cost" because it represents the sum of the time spent by all the travelers in the network, and moreover is obviously connected with the pollution released by all the vehicles. In our analysis we make use of the link cost functions in the form given by the Bureau of Public Roads (BPR) [2] which explicitly contain the flow capacity u_i of each link a_i. We assume that after maintenance the capacity of each link varies from u_i to $\gamma_i u_i$, where $\gamma_i > 1$ is referred to as the *enhancement ratio* of link a_i. The case of a uniform γ, for all links, was considered in [6], where the authors mainly focused on the case $0 < \gamma < 1$ to describe the degradation of the network links. Let I_i be the investment required to enhance the capacity of link a_i by the ratio γ_i, and let I be the amount of money available for the network maintenance. For each set of links that can be upgraded we update the total travel time mentioned above. It has to be noted that the improvement of a link can result in a worsening of the network efficiency. This counterintuitive fact is related to the well known Braess paradox [1] and will be discussed in detail by means of a small test network for which we can compute all the relevant quantities in closed form. Once that the efficiency of the network has been assessed for all the improvements that satisfy the budget constraint, the central authority can make a decision.

The paper is structured as follows. In the following Sect. 2 we provide the main definitions regarding traffic networks and recall the concept of a Wardrop equilibrium and the network efficiency measure that will be used. In Sect. 3 we present the investment optimization model, which is then illustrated in Sect. 4 by means of a small test problem (Braess network) and a realistic traffic network. Further research perspectives are touched upon in Sect. 5. The paper ends with an appendix where we provide some analytical computations related to the small test problem treated in Sect. 4.

2 Traffic Network Equilibrium and Efficiency Measure

For a comprehensive treatment of all the mathematical aspects of the traffic equilibrium problem, we refer the interested reader to the classical book of Patriksson [9]. Here, we focus on the basic definitions and on the variational inequality formulation of a network equilibrium flow (see, e.g. [3, 11]). In what follows, we denote with $a^\top b$ the scalar product between vectors a and b, and with $M^\top$ the transpose of a given matrix M. A traffic network consists of a triple $G = (N, A, W)$, where $N = \{N_1, \ldots, N_p\}$, $p \in \mathbb{N}$, is the set of nodes, $A = \{a_1, \ldots, a_n\}$, $n \in \mathbb{N}$ represents the set of direct arcs (also called links) connecting pairs of nodes and $W = \{W_1, \ldots, W_m\} \subset N \times N$, $m \in \mathbb{N}$ is the set of the origin-destination (O-D) pairs. The flow on the link a_i is denoted by f_i, and we group all the link flows in a vector $f = (f_1, \ldots, f_n)$. A path (or route) is defined as a set of consecutive links and we assume that each O-D pair W_j is connected by r_j, $r_j \in \mathbb{N}$, paths whose set is denoted by P_j, $j = 1, \ldots, m$. All the paths in the network are grouped into a vector $(R_1, \ldots, R_k)$, $k \in \mathbb{N}$. The link structure of the paths can be described by using the link-path incidence matrix $\Delta = (\delta_{ir})$, $i = 1, \ldots, n$, $r = 1, \ldots, k$ with

entries $\delta_{ir} = 1$ if $a_i \in R_r$ and 0 otherwise. To each path R_r it is associated a flow F_r. The path flows are grouped into a vector $(F_1, \ldots, F_k)$ which is called the network path-flow (or simply, the network flow if it is clear that we refer to paths). The flow f_i on the link a_i is equal to the sum of the path flows on the paths which contain a_i, so that $f = \Delta F$. We now introduce the unit cost of traveling through a_i as a real function $c_i(f) \geq 0$ of the flows on the network, so that $c(f) = (c_1(f), \ldots, c_n(f))$ denotes the link cost vector on the network. The meaning of the cost is usually that of travel time and, in the simplest case, the generic component c_i only depends on f_i. In our model we use the BPR form of the link cost function which explicitly take into account the link capacities. More precisely, the travel cost for link a_i is given by:

$$c_i(f_i) = t_i^0 \left[1 + k \left(\frac{f_i}{u_i} \right)^{\beta} \right], \tag{1}$$

where u_i describes the capacity of link a_i, t_i^0 is the free flow travel time or cost on link a_i, while k and β are model parameters which take on positive values. In many applications $k = 0.15$ and $\beta = 4$. Analogously, one can define a cost on the paths as $C(F) = (C_1(F), \ldots, C_k(F))$. Usually, $C_r(F)$ is just the sum of the costs on the links which build that path:

$$C_r(F) = \sum_{i=1}^{n} \delta_{ir} c_i(f),$$

or in compact form $C(F) = \Delta^{\top} c(\Delta F)$. For each pair W_j, there is a given traffic demand $D_j \geq 0$, so that $D = (D_1, \ldots, D_m)$ is the demand vector of the network. Feasible path flows are nonnegative and satisfy the demands, i.e., belong to the set

$$K = \{F \in \mathbb{R}^k : F_r \geq 0, \text{for any } r = 1, \ldots, k \text{ and } \Phi F = D\}, \tag{2}$$

where Φ is the pair-path incidence matrix whose entries, for $j = 1, \ldots, m$, $r = 1, \ldots, k$ are

$$\varphi_{jr} = \begin{cases} 1, & \text{if the path } R_r \text{ connects the pair } W_j, \\ 0, & \text{elsewhere.} \end{cases}$$

The notion of a user traffic equilibrium is given by the following definition.

Definition 1 A network flow $H \in K$ is a user equilibrium, if for each O-D pair W_j, and for each pair of paths R_r, R_s which connect W_j

$$C_r(H) > C_s(H) \Longrightarrow H_r = 0;$$

that is, if traveling along the path R_r takes more time than traveling along R_s, then the flow along R_r vanishes.

Remark 1 Among the various paths which connect a given O-D pair W_j some will carry a positive flow and others zero flow. It follows from the previous definition that, for a given O-D pair, the travel cost is the same for all nonzero flow paths, otherwise users would choose a path with a lower cost. Hence, as an equivalent definition of Wardrop equilibrium we can write that for each W_j,

$$C_r(H) \begin{cases} = \lambda_j & \text{if } H_r > 0, \\ \geq \lambda_j & \text{if } H_r = 0. \end{cases} \tag{3}$$

Hence, with the notation λ_j we denote the equilibrium cost shared by all the used paths connecting W_j. The variational inequality formulation of the user equilibrium is given by the following result (see, e.g., [9]).

Theorem 1 *A network flow $H \in K$ is a user equilibrium iff it satisfies the variational inequality*

$$C(H)^\top (F - H) \geq 0, \quad \forall\, F \in K. \tag{4}$$

Sometimes it is useful to decompose the scalar product in (4) according to the various origin-destination pairs W_j:

$$\sum_{j=1}^{m} \sum_{r \in P_j} C_r(H)\,(F_r - H_r) \geq 0, \quad \forall\, F \in K.$$

The network efficiency measure we consider in this paper is the total travel time when a Wardrop equilibrium is reached:

$$TC = \sum_{j=1}^{m} \sum_{r \in P_j} C_r(H)\, H_r = \sum_{j=1}^{m} \lambda_j D_j. \tag{5}$$

3 Investment Optimization Model

We consider a central authority with an amount of money I available for the network maintenance. Only a subset of links $\{a_i : i \in \mathcal{L}\}$, where $\mathcal{L} \subset \{1, \ldots, n\}$, are involved in the improvement process. Let I_i be the investment required to enhance the capacity of link a_i by a given ratio γ_i.

The central authority aims to find in which subset of links to invest in order to improve as much as possible the total cost (5), while satisfying the budget

constraint. This problem can be formulated within the framework of integer nonlinear optimization.

Let x_i be a binary variable which takes on the value 1 if the investment is actually carried out on link a_i, and 0 otherwise. A vector $x = (x_i)_{i\in\mathcal{L}}$ is feasible if the budget constraint $\sum_{i\in\mathcal{L}} I_i x_i \leq I$ is satisfied. Given a feasible vector x, the capacity of each link a_i, with $i \in \mathcal{L}$, is equal to $u_i(x) := \gamma_i u_i x_i + (1 - x_i)u_i$, i.e., $u_i(x) = \gamma_i u_i$ when $x_i = 1$ and $u_i(x) = u_i$ when $x_i = 0$. We aim to maximize the relative variation of the total cost defined as

$$f(x) = 100 \cdot \frac{TC - TC(x)}{TC},$$

where TC is the total cost (5) before the maintenance job and $TC(x)$ is the total cost corresponding to the improved network. Therefore, the optimization model we propose is

$$\begin{cases} \max\ f(x) \\ \text{s.t.} \sum\limits_{i\in\mathcal{L}} I_i x_i \leq I \\ \qquad x_i \in \{0, 1\} \quad i \in \mathcal{L}. \end{cases} \tag{6}$$

Let us notice that the computation of the nonlinear function f at a given x requires to find a Wardrop equilibrium for both the original and the improved network. Thus, model (6) can be considered as a generalized knapsack problem.

4 Numerical Experiments

This section is devoted to the numerical solution of the proposed model for two networks: the first is a small size network, while the second is the well known Sioux Falls network. The numerical computation of the Wardrop equilibrium was performed by implementing in Matlab 2018a the algorithm designed in [7].

Example 1 We consider the Braess network shown in Fig. 1. There are four nodes, five links labeled by $\{a, b, c, d, e\}$, and one origin-destination pair, from node 1 to node 4 with demand $D = 30$, which can be connected by 3 paths: $R_1 = (a, c)$, $R_2 = (b, d)$, $R_3 = (a, e, d)$. The link cost functions are given by:

$$c_a = 1 + \frac{f_a}{1/2},\ c_b(f_b) = 50\left(1 + \frac{f_b}{50}\right),\ c_c(f_c) = 50\left(1 + \frac{f_c}{50}\right),$$
$$c_d = 1 + \frac{f_d}{1/2},\ c_e(f_e) = 10\left(1 + \frac{f_e}{10}\right).$$

Fig. 1 Braess network

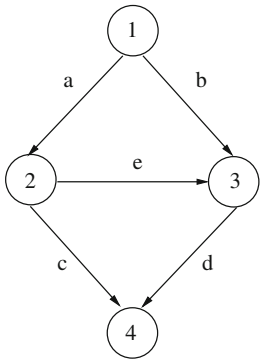


Table 1 Numerical results for Braess network

x	$f(x)$	$I(x)$
(1,0,1,1,0)	6.85	13
(1,1,0,1,0)	6.08	13
(1,0,0,1,0)	5.61	5
(0,1,0,0,1)	−0.19	13
(0,0,0,0,1)	−0.73	5

We assume that the available budget $I = 15$ k€, the subset of links to be maintained is $\mathcal{L} = \{a, b, c, d, e\}$,

$$\gamma_a = 1.2,\ \gamma_b = 1.1,\ \gamma_c = 1.3,\ \gamma_d = 1.2,\ \gamma_e = 1.5,$$
$$I_a = 2, \quad I_b = 8, \quad I_c = 8, \quad I_d = 3, \quad I_e = 5.$$

Table 1 shows the three best feasible solutions and the two worst ones together with the percentage of total cost improvement and the corresponding investment $I(x) = \sum_{i\in\mathcal{L}} I_i x_i$. It is interesting noting that the third best solution needs a much lower investment than the one required by the optimal solution, but the corresponding objective function values are close. Moreover, the two worst solutions reflect the Braess paradox since the values of the objective function are negative. In the Appendix we analyze in more details the Braess paradox for any value of the demand D and of the enhancement factor γ_e.

Example 2 The Sioux Falls network consists of 24 nodes, 76 links and 528 O-D pairs (see Fig. 2). The complete data can be found on [8]. We assume that the available budget $I = 30$ k€ and the subset of links to be maintained is $\mathcal{L} = \{4, 10, 21, 22, 29, 30, 31, 49, 75, 76\}$. We consider two different scenarios: in the first one the average enhancement ratio is around 1.3 (low quality maintenance), while in the second one is 1.55 (high quality maintenance). The values of γ_i and I_i of the two scenarios are shown in Table 2.

Table 3 reports the ten best feasible solutions for the two scenarios. Let us note that the value of the ten best solutions in scenario 1 varies between around 10 and

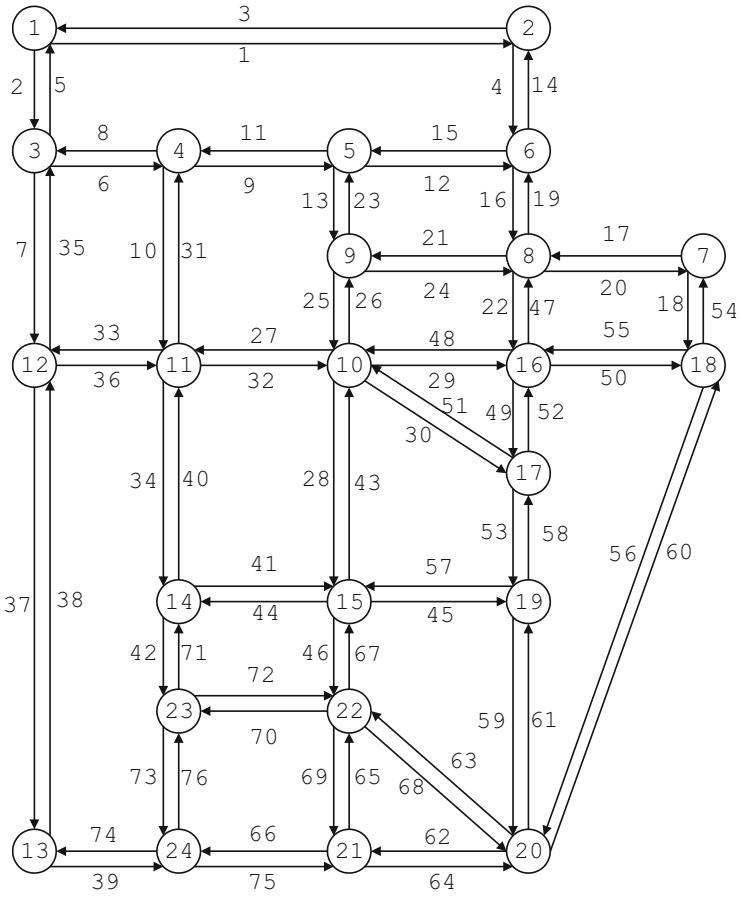


Fig. 2 Sioux Falls network

Table 2 Link capacities and investments for Sioux Falls network

	Scenario 1		Scenario 2	
Links	γ_i	I_i	γ_i	I_i
4	1.2	5	1.4	6
10	1.5	6	1.8	7
21	1.1	10	1.3	12
22	1.3	5	1.5	6
29	1.4	4	1.7	5
30	1.2	8	1.4	10
31	1.1	6	1.3	7
49	1.5	2	1.8	2.5
75	1.4	3	1.7	3.5
76	1.3	2	1.5	2.5

Table 3 Numerical results for Sioux Falls network

Scenario 1			Scenario 2		
x	$f(x)$	$I(x)$	x	$f(x)$	$I(x)$
(0,1,0,1,1,1,0,1,1,1)	10.97	30	(0,1,1,0,1,0,0,1,1,0)	14.30	30
(0,1,0,1,1,1,0,1,1,0)	10.89	28	(0,0,1,1,1,0,0,1,1,0)	14.26	29
(1,0,0,1,1,1,0,1,1,1)	10.64	29	(0,1,0,0,1,1,0,1,1,0)	14.05	28
(1,1,0,0,1,1,0,1,1,1)	10.58	30	(1,1,0,1,1,0,0,1,1,0)	14.00	30
(1,0,0,1,1,1,0,1,1,0)	10.55	27	(0,0,0,1,1,1,0,1,1,1)	13.92	30
(1,1,0,0,1,1,0,1,1,0)	10.50	28	(0,0,0,1,1,1,0,1,1,0)	13.84	27
(0,0,0,1,1,1,1,1,1,1)	10.46	30	(1,0,1,0,1,0,0,1,1,0)	13.80	29
(0,1,1,1,1,0,0,1,1,0)	10.46	30	(0,0,1,0,1,0,1,1,1,0)	13.80	30
(0,0,0,1,1,1,1,1,1,0)	10.38	28	(1,0,0,0,1,1,0,1,1,1)	13.62	30
(0,1,0,0,1,1,1,1,1,0)	10.37	29	(1,0,0,1,1,0,1,1,1,0)	13.54	30

11%, while that in scenario 2 between around 13 and 14%. Therefore, as opposite to Example 1, an improvement of the quality of maintenance implies an improvement of the total cost.

5 Conclusions and Further Perspectives

In this paper we consider the problem of maintaining a road network in an optimal manner. The decision makers are endowed with a given budget and have to decide which roads is better to improve. They make their choice by computing, for each set of possible investments, the corresponding relative improvement of total travel time. The problem is modeled as an integer nonlinear optimization program and some numerical experiments on small and medium scale networks are performed. Such an approach can help the decision makers to select the best possible investments and also displays the counterintuitive fact that some investments can produce a worsening of the traffic.

In the case of large scale networks, further methods have to be developed (e.g., Branch and Bound techniques) to cope the combinatorial nature of the problem. Moreover, since the optimal choice of the decision makers heavily depends on the traffic demand, it would be interesting to consider the realistic case of a randomly perturbed demand (see, e.g., [4, 5]).

Acknowledgements The authors are members of the Gruppo Nazionale per l'Analisi Matematica, la Probabilità e le loro Applicazioni (GNAMPA—National Group for Mathematical Analysis, Probability and their Applications) of the Istituto Nazionale di Alta Matematica (INdAM—National Institute of Higher Mathematics). The work of Fabio Raciti has been partially supported by University of Catania ("Piano della Ricerca 2016/2018 Linea di intervento 2").

Appendix

We consider the Braess network shown in Fig. 1, where the traffic demand from node 1 to node 4 is D and the link cost functions are defined as follows:

$$c_a = 1 + \frac{f_a}{1/2},\ c_b(f_b) = 50\left(1 + \frac{f_b}{50}\right),\quad c_c(f_c) = 50\left(1 + \frac{f_c}{50}\right),$$
$$c_d = 1 + \frac{f_d}{1/2},\ c_e(f_e) = 10\left(1 + \frac{f_e}{10\gamma}\right),$$

where γ is the enhancement factor of arc e. We can find the exact Wardrop equilibrium for any value of parameters D and γ (see e.g. [10]). The path cost functions are then given by:

$$C_1(F) = 3F_1 + 2F_3 + 51,$$
$$C_2(F) = 3F_2 + 2F_3 + 51,$$
$$C_3(F) = 2F_1 + 2F_2 + (4 + \gamma^{-1})F_3 + 12.$$

The Wardrop equilibrium is

$$H = \begin{cases} (D/2, D/2, 0) & \text{if } D > 78, \\ \left(\frac{(2+\gamma^{-1})D-39}{3+2\gamma^{-1}}, \frac{(2+\gamma^{-1})D-39}{3+2\gamma^{-1}}, \frac{78-D}{3+2\gamma^{-1}}\right) & \text{if } \dfrac{39}{2+\gamma^{-1}} \le D \le 78, \\ (0, 0, D) & \text{if } 0 \le D \le \dfrac{39}{2+\gamma^{-1}}, \end{cases}$$

and the corresponding equilibrium cost is

$$\lambda = \begin{cases} \dfrac{3}{2}D + 51 & \text{if } D > 78, \\ \dfrac{(4+3\gamma^{-1})}{3+2\gamma^{-1}}D + \dfrac{39}{3+2\gamma^{-1}} + 51 & \text{if } \dfrac{39}{2+\gamma^{-1}} \le D \le 78, \\ (4+\gamma^{-1})D + 12 & \text{if } 0 \le D \le \dfrac{39}{2+\gamma^{-1}}. \end{cases}$$

We remark that when the demand D varies between 13 and 78, the total cost

$$TC = \frac{(4+3\gamma^{-1})D^2 + 39D}{3+2\gamma^{-1}} + 51D$$

is an increasing function with respect to $\gamma \in [1, 2]$. As a consequence, in this demand range investing for improving the link e capacity results in a growth of

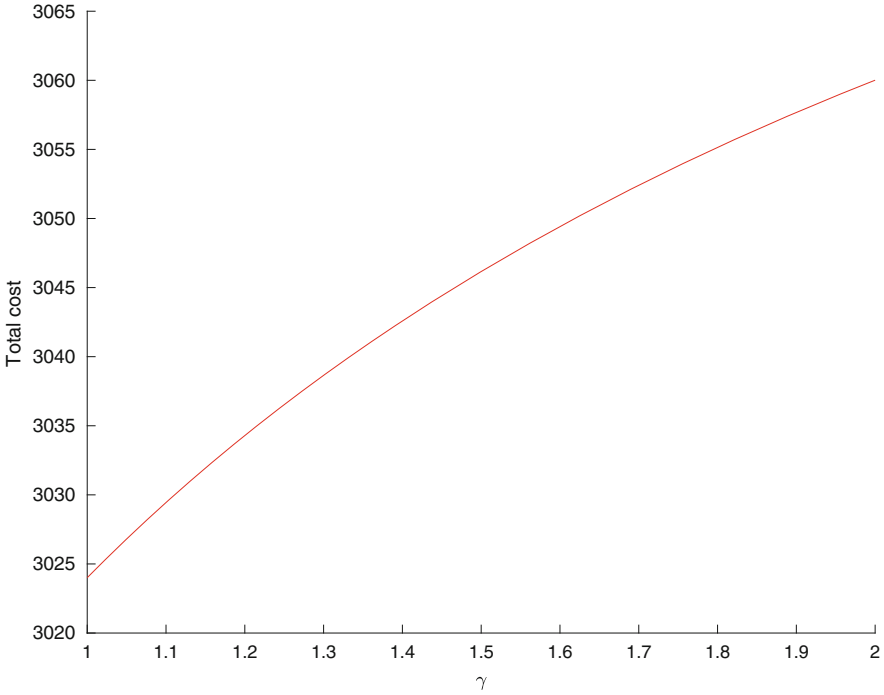


Fig. 3 Total cost vs. enhancement ratio γ for link e

the social cost and pollution. Figure 3 shows the graph of TC as a function of γ for $D = 30$.

References

1. Braess, D.: Über ein Paradoxon aus der Verkehrsplanung. Unternehmenforschung **12**, 258–268 (1968)
2. Bureau of Public Roads: Traffic Assignment Manual. U.S. Department of Commerce, Urban Planning Division, Washington DC (1964)
3. Dafermos, S.: Traffic equilibrium and variational inequalities. Transp. Sci. **14**, 42–54 (1980)
4. Gwinner, J., Raciti, F.: Some equilibrium problems under uncertainty and random variational inequalities. Ann. Oper. Res. **200**, 299–319 (2012)
5. Jadamba, B., Pappalardo, M., Raciti, F.: Efficiency and vulnerability analysis for congested networks with random data. J. Optim. Theory Appl. **177**, 563–583 (2018)
6. Nagurney, A., Qiang, Q.: Robustness of transportation networks subject to degradable links. Europhys. Lett. EPL **80**, Art. 68001, 6 (2007)
7. Panicucci, B., Pappalardo, M., Passacantando, M.: A path-based double projection method for solving the asymmetric traffic network equilibrium problem. Optim. Lett. **1**, 171–185 (2007)
8. Passacantando, M.: Personal web page, Transportation Network Test Problems. http://pages.di.unipi.it/passacantando/test_networks.html. Accessed 29 April 2019

9. Patriksson, M.: The traffic assignment problem. VSP BV, The Netherlands (1994)
10. Raciti, F., Falsaperla, P.: Improved, non iterative algorithm for the calculation of the equilibrium in the traffic network problem. J. Optim. Theory Appl. **133**, 401–411 (2007)
11. Smith, M.J.: The existence, uniqueness and stability of traffic equilibria. Transp. Res. **13B**, 295–304 (1979)

Opinion Dynamics in Multi-Agent Systems Under Proportional Updating and Any-to-Any Influence

Loretta Mastroeni, Maurizio Naldi, and Pierluigi Vellucci

Abstract We study an agent-based model to describe the formation of opinions within a group, where agents belong to classes. In the model any agent influences all the other agents at the same time, and the influence is proportional to the difference of opinions through interaction coefficients. We find that the interaction coefficients must lie within a tetrahedron for the internal consistency of the model. We show that the system of agents reaches a steady state. The long-term opinion of each agents depends anyway on its initial opinion.

Keywords Agent-based models · Opinion dynamics · Social networks

1 Introduction

Agent-based models (ABM) allow us to study all kinds of influence phenomena, in particular to analyze the formation of opinions within a group of people, which is a subject of interest in many areas, e.g. sociology and psychology. Through ABMs we can understand if and how the individuals reach a final consensus or the people polarize around a small number of different opinions [2, 12, 15, 20], by going from the description of the behaviour of individuals at the micro level to the prediction of the macro behaviour of the group.

L. Mastroeni · P. Vellucci
Department of Economics, Roma Tre University, Rome, Italy
e-mail: loretta.mastroeni@uniroma3.it; pierluigi.vellucci@uniroma3.it

M. Naldi (✉)
Department of Civil Engineering and Computer Science, University of Rome Tor Vergata, Rome, Italy

Department of Law, Economics, Politics and Modern Languages, LUMSA University, Rome, Italy
e-mail: maurizio.naldi@uniroma2.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_25

In many cases, the individuals in the group exhibit significant differences, e.g. in their religion, ethnicity, political convictions. It is natural to assume that individuals with similar characteristics behave similarly, so that we can consider a partition of the original group into classes and devise multi-class agent-based models. Examples of such multi-class models are shown in [7, 8, 13, 14, 22, 25] for two classes, in [1, 4, 26] for the classification based on the leader\follower role, and in [17], where a classification into three classes based on political convictions (leftist, centrist and rightists) is considered. A generic multi-class model has recently been studied by Monica and Bergenti [19], but the model considers agents interacting in pairs: at each time step, a single agents influences another single agent, who in turn changes its opinion due to that influence.

What happens when an agent influences many agents at the same time? Or when many agents act at the same time on all other agents? This is the most frequent situation occurring in many contexts. For example, that routinely happens in an online social network (like *Facebook* or *Twitter*), where an agent submitting its post influences all the followers at the same time. It also happens on more traditional media (like TV or the radio) where agents involved in a debate may broadcast their opinions to a large audience.

Our paper answers that question, by extending Monica and Bergenti model to the case of any-to-any interaction, where each agent influences all the other agents at once, by providing the following contributions:

- since the opinion is represented by a variable in the $[-1, 1]$ range, we identify the conditions that allow the opinion of agents to stay within the prescribed range, i.e. the *closure property* (Sect. 2);
- we show that the typical state updating equation may be set in the form of a dynamical linear system (Sect. 2);
- we show that the opinion of each agent converges in the long term, and provide the steady-state solution of the dynamical system, i.e. the steady-state opinion of all agents, provided their initial values (Sect. 3).

2 The Any-to-Any Interaction Model

As stated in the Introduction, we introduce a major difference with the model proposed in [19], by assuming that at any time step each agent influences all the other agents at once, i.e. an *any-to-any* interaction. This happens, e.g., in online forums, where each agent (each individual) is exposed in real time to the opinions of all the other agents and may change its opinion accordingly (see, e.g. [21]). In this section, we describe the resulting model and formulate the interaction described by the state updating equation as a linear dynamical system.

We consider a population of n agents, who belong to one of c classes. The differences among classes may be related to a number of factors, e.g. religion, ethnicity, political convictions and we assume that each class is homogeneous—i.e. in them people are similar to each other or are of the same type (same religion,

same political party etc.). The number of agents in class i is n_i ($n = \sum_{i=1}^{c} n_i$). The opinion of agent j belonging to class i is v_{ij}, with $-1 \leq v_{ij} \leq 1$, as assumed in the literature [1, 3, 19, 22, 25].

The opinion of any agent may change over time due to its interactions with other agents. We assume a linear effect of opinion differences: the change in the opinion of an agent due to another agent is proportional to the difference in their opinions. The intensity of the interaction is described by a set of c^2 coefficients γ_{kl}, $k, l \in \{1, 2, \ldots, c\}$ in the range [0, 1]. The generic coefficient γ_{kl} measures the propensity of an agent of class l to make its opinion closer to that of an agent in class k. The interaction needs not be symmetric, so that $\gamma_{kl} \neq \gamma_{lk}$ in general, but is identical for all the pairs of agents in the same pairs of classes. This any-to-any interaction model with linear interaction has been adopted in [9, 10, 23, 24]. However, none of them considered a multi-class model as we do here.

The time evolution of the opinion of an agent is then described by Eq. (1), where the interaction terms have been grouped by considering separately the contributions due to the agent's own class and those due to other classes:

$$\begin{aligned} v_{ij}(t+1) &= v_{ij}(t) - \sum_{\substack{k=1 \\ k\neq i}}^{c} \gamma_{ik} \sum_{l=1}^{n_k} \left[v_{ij}(t) - v_{kl}(t) \right] + \gamma_{ii} \sum_{\substack{m=1 \\ m\neq j}}^{n_i} \left[v_{im}(t) - v_{ij}(t) \right] \\ &= v_{ij}(t) \left[1 - \gamma_{ii}(n_i - 1) - \sum_{\substack{k=1 \\ k\neq i}}^{c} \gamma_{ik} n_k \right] + \gamma_{ii} \sum_{\substack{m=1 \\ m\neq j}}^{n_i} v_{im}(t) \\ &+ \sum_{\substack{k=1 \\ k\neq i}}^{c} \gamma_{ik} \sum_{l=1}^{n_k} v_{kl}(t) \,. \end{aligned} \tag{1}$$

However, the opinion expressed by Eq. (1) may leak outside the prescribed boundaries. In order to obtain the *closure property* for the transformation represented by that equation (i.e., $-1 \leq v_{ij}(t) \leq 1 \quad \forall t$), we impose conditions on the coefficients γ_{kl}, starting with the sufficient condition of Proposition 1. The closure property was already achieved in [19], but just under the assumption of pair-by-pair interaction (i.e., at each time step just two agents interact with each other).

Under an any-to-any interaction, we can state the following closure condition:

Proposition 1 *If* $-1 \leq v_{ij}(0) \leq 1$ *and the coefficients* γ_{ij} *satisfy the inequality*

$$\gamma_{ii}(n_i - 1) + \sum_{\substack{k=1 \\ k\neq i}}^{c} \gamma_{ik} n_k \leq 1 \,, \; \forall i = 1, \ldots, c \,, \; c \in \mathbb{N} \,. \tag{2}$$

then $-1 \leq v_{ij}(t) \leq 1 \; \forall t \in \mathbb{N}$.

Proof We prove first the proposition concerning the upper bound ($v_{ij}(t) \leq 1$).

Let us proceed by induction on $t \in \mathbb{N}_0$. The starting step, $t = 0$, is ensured by hypothesis. We can now check the claim for $t + 1$. Since $-1 \leq v_{ij}(t) \leq 1 \; \forall i, j$ and v_{ij} multiplies a positive quantity in (1) by Eq. (2) we have:

$$v_{ij}(t+1) \leq \left[1 - \gamma_{ii}(n_i - 1) - \sum_{\substack{k=1 \\ k \neq i}}^{c} \gamma_{ik} n_k \right] + \gamma_{ii} \sum_{\substack{m=1 \\ m \neq j}}^{n_i} 1 + \sum_{\substack{k=1 \\ k \neq i}}^{c} \gamma_{ik} \sum_{l=1}^{n_k} 1$$

$$= 1 - \gamma_{ii}(n_i - 1) - \sum_{\substack{k=1 \\ k \neq i}}^{c} \gamma_{ik} n_k + \gamma_{ii}(n_i - 1) + \sum_{\substack{k=1 \\ k \neq i}}^{c} \gamma_{ik} n_k = 1 \,.$$

Similarly, we prove the condition concerning the lower bound ($v_{ij}(t) \geq -1$). Since $v_{ij}(0) \geq -1$ by hypothesis, we have

$$v_{ij}(t+1) \geq - \left[1 - \gamma_{ii}(n_i - 1) - \sum_{\substack{k=1 \\ k \neq i}}^{c} \gamma_{ik} n_k \right] - \gamma_{ii}(n_i - 1) - \sum_{\substack{k=1 \\ k \neq i}}^{c} \gamma_{ik} n_k$$

$$= -1 + \gamma_{ii}(n_i - 1) + \sum_{\substack{k=1 \\ k \neq i}}^{c} \gamma_{ik} n_k - \gamma_{ii}(n_i - 1) - \sum_{\substack{k=1 \\ k \neq i}}^{c} \gamma_{ik} n_k = -1.$$

□

The condition for closure represented by the inequality (2) is amenable to the following geometric interpretation.

Remark 1 Let us consider the condition (2) $\gamma_{ii}(n_i - 1) + \sum_{\substack{k=1 \\ k \neq i}}^{c} \gamma_{ik} n_k = 1$ and set $\underline{\gamma} := (\gamma_{i1}, \ldots, \gamma_{ic})$, $\underline{x} = (n_1, \ldots, n_{i-1}, n_i - 1, n_{i+1}, \ldots, n_c)$. For a given $\underline{x}$, the solution is a set of the form $\left\{ \underline{\gamma} \,|\, \underline{x}^T \cdot \underline{\gamma} = 1 \right\}$, i.e. an *affine hyperplane* in $\mathbb{R}^c$. Since $\gamma_{ij} \geq 0 \; \forall i, j$, condition (2) represents the volume of a *generalized tetrahedron* in $\mathbb{R}^c$ bounded by the coordinate hyperplanes and the hyperplane $\underline{x}^T \cdot \underline{\gamma} = 1$.

We can now write the state updating equation as a linear dynamical mapping:

$$\underline{v}(t+1) = \mathbf{A}\underline{v}(t) \tag{3}$$

where the opinions of agents belonging to different classes have been stacked in a single vector $\underline{v} = [v_1, \ldots, v_n]^T$, whose components are

$$v_{ij} = \begin{cases} v_{j+\sum_{h=1}^{i-1} n_h} & \text{if } i \geq 2\,,\ j = 1, \ldots, n_i \\ v_j & \text{if } i = 1\,,\ j = 1, \ldots, n_1 \end{cases} \tag{4}$$

and $\mathbf{A}$ is a $n \times n$ matrix of interaction coefficients.

We use the following results (Proposition 2 and Lemma 1) to prove Theorem 1. Proposition 2 shows that the matrix $\mathbf{A}$ is a *stochastic* matrix and it has a special block structure. (A stochastic matrix $M = (m_{ij}) \in \mathbb{R}^{n \times n}$ is a square matrix whose entries are such that $m_{ij} \in [0, 1]$ and $\sum_{j=1}^{n} m_{ij} = 1$.)

Proposition 2 *The matrix $\mathbf{A}$ is a stochastic matrix with the block form:*

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \cdots & \mathbf{A}_{1c} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \cdots & \mathbf{A}_{2c} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{c1} & \mathbf{A}_{c2} & \cdots & \mathbf{A}_{cc} \end{pmatrix}.$$

where each block $\mathbf{A}_{ii}$, for $i = 1, \ldots, c$, is a $n_i \times n_i$ matrix

$$\mathbf{A}_{ii} = \begin{pmatrix} 1 - \underline{x}^T \cdot \underline{\gamma} & \gamma_{ii} & \gamma_{ii} & \cdots & \gamma_{ii} \\ \gamma_{ii} & 1 - \underline{x}^T \cdot \underline{\gamma} & \gamma_{ii} & \cdots & \gamma_{ii} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \gamma_{ii} & \gamma_{ii} & \cdots & \gamma_{ii} & 1 - \underline{x}^T \cdot \underline{\gamma} \end{pmatrix},$$

and $\mathbf{A}_{hk}$, for $h, k = 1, \ldots, c$, is a $n_h \times n_k$ matrix such that

$$\mathbf{A}_{hk} = \begin{pmatrix} \gamma_{hk} & \gamma_{hk} & \cdots & \gamma_{hk} \\ \vdots & \vdots & \vdots & \vdots \\ \gamma_{hk} & \gamma_{hk} & \cdots & \gamma_{hk} \end{pmatrix}.$$

Proof Let us rewrite (1) according to (4). Then, if $i = 1$

$$v_j(t+1) = v_j(t) \left[1 - \gamma_{11}(n_1 - 1) - \sum_{\substack{k=1 \\ k \neq 1}}^{c} \gamma_{1k} n_k \right] + \\ + \gamma_{11} \sum_{\substack{m=1 \\ m \neq j}}^{n_1} v_m(t) + \sum_{k=2}^{c} \gamma_{1k} \sum_{l=1}^{n_k} v_{l+\sum_{h=1}^{k-1} n_h}(t) \tag{5}$$

for each $j = 1, \ldots, n_1$; otherwise, if $i \geq 2$ we have

$$v_{j+\sum_{h=1}^{i-1} n_h}(t+1) = v_{j+\sum_{h=1}^{i-1} n_h}(t) \left[1 - \gamma_{ii}(n_i - 1) - \sum_{\substack{k=1 \\ k \neq i}}^{c} \gamma_{ik} n_k \right] + \\ + \gamma_{ii} \sum_{\substack{m=1 \\ m \neq j}}^{n_i} v_{m+\sum_{h=1}^{i-1} n_h}(t) + \gamma_{i1} \sum_{l=1}^{n_1} v_l(t) + + \sum_{\substack{k=2 \\ k \neq i}}^{c} \gamma_{ik} \sum_{l=1}^{n_k} v_{l+\sum_{h=1}^{k-1} n_h}(t)$$

for each $j = 1, \ldots, n_i$.

Let us consider (5). We can rewrite $v_j(t+1)$ as a dot product of two arrays:

$$\underline{\alpha} = \Bigg[\underbrace{\gamma_{11}, \ldots, \gamma_{11}}_{j-1}, 1 - \gamma_{11}(n_1 - 1) - \sum_{k=2}^{c} \gamma_{1k} n_k, \underbrace{\gamma_{11}, \ldots, \gamma_{11}}_{n_1 - j}, \\ \underbrace{\gamma_{12}, \ldots, \gamma_{12}}_{n_2}, \ldots, \underbrace{\gamma_{1c}, \ldots, \gamma_{1c}}_{n_c} \Bigg]^T$$

and $\underline{v}$. Denoting $v_{j+\sum_{h=1}^{i-1} n_h}(t+1) = \underline{\beta} \cdot \underline{v}$, we have

$$\underline{\beta} = \Bigg[\underbrace{\gamma_{i1}, \ldots, \gamma_{i1}}_{n_1}, \ldots, \underbrace{\gamma_{i,i-1}, \ldots, \gamma_{i,i-1}}_{n_{i-1}}, \underbrace{\gamma_{ii}, \ldots, \gamma_{ii}}_{j-1}, 1 - \gamma_{ii}(n_i - 1) + \\ - \sum_{\substack{k=1 \\ k \neq i}}^{c} \gamma_{ik} n_k, \underbrace{\gamma_{ii}, \ldots, \gamma_{ii}}_{n_i - j}, \underbrace{\gamma_{i,i+1}, \ldots, \gamma_{i,i+1}}_{n_{i+1}}, \ldots, \underbrace{\gamma_{ic}, \ldots, \gamma_{ic}}_{n_c} \Bigg]^T .$$

The sum of all elements of $\underline{\alpha}$ and $\underline{\beta}$ is equal to 1. □

Before introducing Lemma 1, let us recall below a well-known property of a generic square matrix **A**. It is said to be diagonally dominant if, denoting the generic element of **A** by a_{ij} , the following inequality holds:

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}| \quad \text{for all } i \in \{1, \ldots, c\} .$$

We recall theta a matrix $\mathbf{A}$ is said to be strictly diagonally dominant if

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \quad \text{for all } i \in \{1, \ldots, c\}\,.$$

Then:

Lemma 1 *Let us consider the $n \times n$ matrix $\boldsymbol{A} - \boldsymbol{I}$, where $\boldsymbol{A}$ is the matrix defined in (3). Then the following claims hold.*

(i) Matrix $\boldsymbol{A} - \boldsymbol{I}$ is singular.
(ii) Matrix $\boldsymbol{A} - \boldsymbol{I}$ is non-strictly diagonally dominant. More precisely, the following equality holds

$$|b_{hh}| = \sum_{k \neq h} |b_{hk}|\,,$$

where $b_{hk} = (\boldsymbol{A} - \boldsymbol{I})_{hk}$.
(iii) The rank of $\boldsymbol{A} - \boldsymbol{I}$ is $n - 1$.

Proof The first claim follows is true since the sum of the elements in each row of $\mathbf{A} - \mathbf{I}$ is zero (see Proposition 2), so that $\det(\mathbf{A} - \mathbf{I}) = 0$.

As to claim (ii), in our assumptions the parameters γ_{ij}, for each $1 \leq i \leq c$ and $1 \leq j \leq n_i$, are positive; the values of n_i, $1 \leq i \leq c$, represent the number of agents of class i so that they are also positive. Consequently, the non-diagonal elements of $\mathbf{A} - \mathbf{I}$ are positive and $|b_{hk}| = b_{hk}$, $\forall h, k = 1, \ldots, n$ with $k \neq h$. For the same matter the diagonal elements of $\mathbf{A} - \mathbf{I}$ are negative. Therefore, the equality of claim (ii) holds.

According to the first claim, the matrix $\mathbf{A} - \mathbf{I}$ is singular, i.e., not full rank ($\text{rank}(\mathbf{A} - \mathbf{I}) < n$). In order to prove the third claim, we show that the principal minor of order $n - 1$ is full rank, since a strictly diagonally dominant matrix is in turn full rank. That property can be easily checked observing

$$\gamma_{11}(n_1 - 1) + \sum_{k=2}^{c} \gamma_{1k} n_k > \gamma_{11}(n_1 - 1) + \sum_{k=2}^{c-1} \gamma_{1k} n_k + \gamma_{1c}(n_c - 1)$$

when $i = 1$ and $j = 1, \ldots, n_1$, and

$$\gamma_{ii}(n_i - 1) + \sum_{\substack{k=1 \\ k \neq i}}^{c} \gamma_{ik} n_k > \gamma_{ii}(n_i - 1) + \sum_{\substack{k=1 \\ k \neq i}}^{c-1} \gamma_{ik} n_k + \gamma_{ic}(n_c - 1)$$

when $i = 2, \ldots, c - 1$ and $j = 1, \ldots, n_i$, or $i = c$ and $j = 1, \ldots, n_c - 1$. □

3 Steady-State Opinions

The opinions of agents fluctuate over time due to the reciprocal influences described by the model of Sect. 2. We wish to see if such opinions keep changing at every time step or reach a steady state. In particular we wish to see if a consensus forms around one or more opinion values, i.e. if those opinions, though different at the beginning, converge around a limited number of values. In this section, we provide the conditions for such a steady state and the steady-state solution as well.

We recall that the time evolution of opinions over k time steps is

$$\underline{v}(t+k) = \mathbf{A}^k \underline{v}(t), \tag{6}$$

so that knowing the power $\mathbf{A}^k$ allows us to get the vector of opinions at time k if we know the vector of opinions at time 0.

We now prove the following theorem, which holds under the only requirement that the interaction coefficients meet the closure condition of Proposition 1. It requires the definition of positive matrix [16], i.e. $\mathbf{A}$ is a positive matrix whether each $a_{ij} > 0$, and this is usually denoted by writing $\mathbf{A} > 0$.

Theorem 1 *Let $\gamma_{ij} \in (0, 1]$, for each $1 \le i \le c$, $1 \le j \le n_i$. Let also $\gamma_{ii}(n_i - 1) + \sum_{\substack{k=1 \\ k \neq i}}^{c} \gamma_{ik} n_k < 1 \ \forall i = 1, \ldots, c$. Then the matrix $\mathbf{A}$ in (3) is positive and the following statements are true.*

(i) The Perron–Frobenius eigenvalue r is equal to 1.

(ii) There exists a right-hand eigenvector $\underline{u}$ and a left-hand eigenvector $\underline{w}^T$ of $\mathbf{A}$ associated to the Perron–Frobenius eigenvalue 1. *They are: $\underline{u}^T = (u_1, \ldots, u_n) = (1, \ldots, 1)$; $\underline{w}^T = (w_1, \ldots, w_n)$ is such that*

$$\begin{cases} 0 &= -\sum_{k=2}^{c} \gamma_{1k} n_k \, w_1 + \sum_{k=2}^{c-1} \gamma_{k1} n_k w_{1+\sum_{h=1}^{k-1} n_h} + \\ &\quad + \gamma_{c1} \left[(n_c - 1) w_{1+\sum_{h=1}^{c-1} n_h} + w_{\sum_{h=1}^{c} n_h} \right] \\ 0 &= \gamma_{12} n_1 w_1 - \sum_{\substack{k=1 \\ k \neq 2}}^{c} \gamma_{2k} n_k w_{n_1+1} + \sum_{k=3}^{c-1} \gamma_{k2} n_k w_{1+\sum_{h=1}^{k-1} n_h} \\ &\quad + \gamma_{c2} \left[(n_c - 1) w_{1+\sum_{h=1}^{c-1} n_h} + w_{\sum_{h=1}^{c} n_h} \right] \\ &\vdots \\ 0 &= \gamma_{1c} n_1 w_1 + \sum_{k=2}^{c-1} \gamma_{kc} n_k w_{1+\sum_{h=1}^{k-1} n_h} + \\ &\quad - \left(\gamma_{cc} + \sum_{k=1}^{c-1} \gamma_{ck} n_k \right) w_{1+\sum_{h=1}^{c-1} n_h} + \gamma_{cc} w_{\sum_{h=1}^{c} n_h} \end{cases} \tag{7}$$

and

$$\begin{cases} w_{n_1} = \cdots = w_2 = w_1 \\ w_{n_1+n_2} = \cdots = w_{n_1+2} = w_{n_1+1} \\ \vdots \\ w_{\sum_{h=1}^{c-1} n_h} = \cdots = w_{2+\sum_{h=1}^{c-2} n_h} = w_{1+\sum_{h=1}^{c-2} n_h} \\ w_{-1+\sum_{h=1}^{c} n_h} = \cdots = w_{2+\sum_{h=1}^{c-1} n_h} = w_{1+\sum_{h=1}^{c-1} n_h} \end{cases} .$$

(iii) Let $k \in \mathbb{N}$. Then, fixed $t \in \mathbb{N}$,

$$\lim_{k\to\infty} \underline{v}(t+k) = \frac{\underline{u}\,\underline{w}^T}{\underline{w}^T\,\underline{u}}\,\underline{v}(t)\,. \tag{8}$$

Proof The first claim derives from Perron–Frobenius theorem [16], and Proposition 2. In fact, $\sum_j a_{ij} = 1\ \forall i$. From this the expression of right-hand eigenvector $\underline{u}$ in the second claim also follows.

Let us calculate the left-hand eigenvector $\underline{w}^T = (w_1, \ldots, w_n)$. By definition, $\underline{w}^T\mathbf{A} = \underline{w}^T$. It turns out to be a $n \times n$ homogenous system, which can be broken down as follows:

$$\begin{cases} w_1 & = \left(1 - \gamma_{11}(n_1 - 1) - \sum_{k=2}^{c} \gamma_{1k} n_k\right) w_1 + \gamma_{11} \sum_{k=2}^{n_1} w_k + \\ & + \sum_{k=2}^{c} \gamma_{k1} \sum_{l=1}^{n_k} w_{l+\sum_{h=1}^{k-1} n_h} \\ & \vdots \\ w_{n_1} & = \gamma_{11} \sum_{k=1}^{n_1-1} w_k + \left(1 - \gamma_{11}(n_1 - 1) - \sum_{k=2}^{c} \gamma_{1k} n_k\right) w_{n_1} + \\ & + \sum_{k=2}^{c} \gamma_{k1} \sum_{l=1}^{n_k} w_{l+\sum_{h=1}^{k-1} n_h} \end{cases} \tag{9}$$

for $i = 1$, and so on up to $i = c$, i.e.

$$\begin{cases} w_{n_1+\cdots+n_{c-1}+1} & = \gamma_{1c} \sum_{k=1}^{n_1} w_k + \sum_{k=2}^{c-1} \gamma_{kc} \sum_{l=1}^{n_k} w_{l+\sum_{h=1}^{k-1} n_h} + \\ & + \left(1 - \gamma_{cc}(n_c - 1) - \sum_{k=1}^{c-1} \gamma_{ck} n_k\right) w_{n_1+\cdots+n_{c-1}+1} + \\ & + \gamma_{cc} \sum_{k=2}^{n_c} w_{n_1+\cdots+n_{c-1}+k} \\ & \vdots \\ w_{n_1+\cdots+n_c (= w_n)} & = \gamma_{1c} \sum_{k=1}^{n_1} w_k + \sum_{k=2}^{c-1} \gamma_{kc} \sum_{l=1}^{n_k} w_{l+\sum_{h=1}^{k-1} n_h} + \\ & + \gamma_{cc} \sum_{k=1}^{n_c-1} w_{n_1+\cdots+n_{c-1}+k} + \\ & + \left(1 - \gamma_{cc}(n_c - 1) - \sum_{k=1}^{c-1} \gamma_{ck} n_k\right) w_{n_1+\cdots+n_c}\,. \end{cases} \tag{10}$$

The coefficient matrix of the $n \times n$ system is $\mathbf{A} - \mathbf{I}$, with has rank equal to $n - 1$ (see Lemma 1, point (iii)). Following the proof of Lemma 1, we can work on the top-left principal minor of order $n - 1$, i.e., on the c sub-systems from $i = 1$ (9) to $i = c$ (10), but deleting the last equation of the last sub-system, so that we can rewrite the system in a more concise form.

Let $i = 1$ and consider the sub-system (9). If we subtract the first equation from the second, the first from the third and so on, all the way to subtract the first equation from the last one, we obtain $w_{n_1} = \cdots = w_2 = w_1$.

By performing previous step up to $i = c$ (sub-system (10)), where we have $n_c - 1$ equations, we obtain $w_{n_1+\cdots+n_c-1} = \cdots = w_{n_1+\cdots+n_{c-1}+2} = w_{n_1+\cdots+n_{c-1}+1}$ (notice that the chain of equalities does not start from $w_{n_1+\cdots+n_c}$). Substituting these results in the system composed by the first equations of the sub-systems, ranging from $i = 1$ (9) to $i = c$ (10) we obtain the system (7) of c equations in $c + 1$ variables. The third claim derives by claim (ii) and Perron–Frobenius theorem [16] applied to Eq. (6). □

4 Conclusions and Open Problems

We have obtained two major results for a multi-agent multi-class system, concerning respectively some constraints on the interaction coefficients and the long-term evolution of the multi-agent system. The interaction coefficients cannot be set freely, but must lie within a generalized tetrahedron to guarantee that the range of the opinion variables stays bounded in $[-1, 1]$. If the interaction coefficient are to be derived from the observation of a real sample of individuals, this may cause some problem. The most important result is that the opinion of each agent converges in the long term to a steady state. In addition, though the steady-state solution has an apparently simple form, expressed in terms of right and left-hand eigenvectors, its computation remains generally unknown because of the expression of the left-hand eigenvector, whose computation is difficult due to the high number of parameters involved. Further work is therefore envisaged to ease that computation and to examine the behaviour of such a model in sample cases, even with the use of numerical algorithms (such as e.g. QR algorithm and its variants [5, 6, 11, 18]).

It would also be interesting, for the future, to investigate possible connections of the work with spectral graph theory, particularly in the case of non-symmetric matrices (since $\mathbf{A}$, the matrix of interaction coefficients, is in general non-symmetric). Matrix $\mathbf{A}$ is related to the notion of Laplacian of a graph, whose vertices are the agents and the edges represent the interactions between them. Estimating the eigenvalues of Laplacian is related to the collective decision-making as discussed, e.g., in [25].

References

1. Albi, G., Pareschi, L., Zanella, M.: Boltzmann-type control of opinion consensus through leaders. Philos. Trans. R. Soc. Lond. A: Math. Phys. Eng. Sci. **372**(2028), 20140138 (2014)
2. Albi, G., Pareschi, L., Toscani, G., Zanella, M.: Recent Advances in Opinion Modeling: Control and Social Influence, pp. 49–98. Springer International Publishing, Cham (2017)
3. Amelkin, V., Bullo, F., Singh, A.K.: Polar opinion dynamics in social networks. IEEE Trans. Autom. Control **62**(11), 5650–5665 (2017)
4. Borra, D., Lorenzi, T.: A hybrid model for opinion formation. Z. Angew. Math. Phys. **64**(3), 419–437 (2013)
5. Francis, J.G.F.: The QR transformation a unitary analogue to the LR transformation—part 1. Comput. J. **4**(3), 265–271 (1961)
6. Francis, J.G.F.: The QR Transformation—part 2. Comput. J. **4**(4), 332–345 (1962)
7. Galam, S.: Minority opinion spreading in random geometry. Eur. Phys. J. B; Condensed Matter and Complex Sys. **25**(4), 403–406 (2002)
8. Galam, S.: Contrarian deterministic effects on opinion dynamics: "the hung elections scenario". Physica A: Stat. Mech. Appl. **333**, 453–460 (2004)
9. Grabisch, M., Mandel, A., Rusinowska, A., Tanimura, E.: Strategic influence in social networks. Math. Oper. Res. **43**(1), 1–22 (2017)
10. Jia, P., Friedkin, N.E., Bullo, F.: Opinion dynamics and social power evolution over reducible influence networks. SIAM J. Control Optim. **55**(2), 1280–1301 (2017)
11. Kublanovskaya, V.: On some algorithms for the solution of the complete eigenvalue problem. USSR Comput. Math. Math. Phys. **1**(3), 637–657 (1962)
12. Lorenz, J.: Continuous opinion dynamics under bounded confidence: a survey. Int. J. Mod. Phys. C **18**(12), 1819–1838 (2007)
13. Mastroeni, L., Naldi, M., Vellucci, P.: Individual competence evolution under equality bias. In: European Modelling Symposium (EMS). IEEE, Manchester (2017)
14. Mastroeni, L., Naldi, M., Vellucci, P.: An agent-based model on scale-free networks for personal finance decisions. In: Proceedings of the 20th Workshop "from Objects to Agents", WOA (2019)
15. Mastroeni, L., Vellucci, P., Naldi, M.: Agent-based models for opinion formation: a bibliographic survey. IEEE Access **7**, 1–12 (2019)
16. Meyer, C.D.: Matrix Analysis and Applied Linear Algebra, vol. 2. Society for Industrial and Applied Mathematics, Philadelphia (2000)
17. Mobilia, M.: Fixation and polarization in a three-species opinion dynamics model. Europhys. Lett. **95**(5), 50002 (2011)
18. Moler, C.B.: Eigenvalues and Singular Values, chap. 10, pp. 1–39. Siam, Philadelphia (2008)
19. Monica, S., Bergenti, F.: An analytic study of opinion dynamics in multi-agent systems. Comput. Math. Appl. **73**(10), 2272–2284 (2017)
20. Motsch, S., Tadmor, E.: Heterophilious dynamics enhances consensus. SIAM Rev. **56**(4), 577–621 (2014)
21. Naldi, M.: A conversation analysis of interactions in personal finance forums. In: 14th International Conference on statistical Analysis of Textual Data (JADT), pp. 571–577. Rome (2018)
22. Pareschi, L., Vellucci, P., Zanella, M.: Kinetic models of collective decision-making in the presence of equality bias. Physica A: Stat. Mech. Appl. **467**, 201–217 (2017)
23. Parsegov, S.E., Proskurnikov, A.V., Tempo, R., Friedkin, N.E.: Novel multidimensional models of opinion dynamics in social networks. IEEE Trans. Autom. Control **62**(5), 2270–2285 (2017)

24. Proskurnikov, A.V., Tempo, R., Cao, M., Friedkin, N.E.: Opinion evolution in time-varying social influence networks with prejudiced agents. IFAC-PapersOnLine **50**(1), 11896–11901 (2017). 20th IFAC World Congress
25. Vellucci, P., Zanella, M.: Microscopic modeling and analysis of collective decision making: equality bias leads suboptimal solutions. Ann. Univ. Ferrara **64**, 185–207 (2017)
26. Zhao, Y., Zhang, L., Tang, M., Kou, G.: Bounded confidence opinion dynamics with opinion leaders and environmental noises. Comput. Oper. Res. **74**, 205–213 (2016)

A New Formulation of the Single Door Truck Scheduling Problem

M. Flavia Monaco and Marcello Sammarra

Abstract In this paper we propose a new formulation for the truck scheduling problem at a cross-docking terminal with one inbound door and one outbound door. Through the computational experience, we show that the new formulation is more effective than a formulation from the literature.

Keywords Cross-docking · Mixed integer formulation · Scheduling

1 Introduction

Cross-docking terminals play an important role in manufacturing and logistic systems. Actually, transferring goods coming from possibly different suppliers to the retailers without the inventory phase, has the immediate consequence of reducing the distribution costs. On the other hand, this policy requires sophisticated management procedures to synchronize the arrival and departure times of the trucks. As an evidence of the importance of cross-docking policies in the distribution chain, we want to highlight here that more than 150 papers on this research stream have been published in the last 15 years. The reader can refer to [1, 2, 7, 9] for a complete and updated review of the literature related to cross-docking terminal management problems in many different settings.

In this paper we consider the truck scheduling problem at a cross docking terminal with one inbound and one outbound door. We assume that any product arriving at the terminal by the inbound trucks is dedicated to specific outbound trucks, so that the transhipment flow inside the terminal is known. As a result, the truck scheduling

M. F. Monaco (✉)
DIMES, Universitá della Calabria, Rende, Italy
e-mail: monaco@dimes.unical.it

M. Sammarra
ICAR, Consiglio Nazionale delle Ricerche, Rende, Italy
e-mail: sammarra@icar.cnr.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_26

problem can be seen as a two-machine flow-shop scheduling problem with cross-docking (precedence) constraints, the objective being to minimize the makespan. In a very recently published paper [4], the problem has been modeled in this way. Here we propose a new formulation for the same problem. We compare experimentally the two formulations by comparing the linear bounds they provide on a set of instances available from the literature [4]. Finally we report on the performances of a standard MILP solver when used to solve the two models.

The paper is organized as follows. In Sect. 2 we first detail the formulation of Fonseca et al. [4], then we introduce our new formulation. Section 3 reports the computational experience. Finally, conclusions are drawn in Sect. 4.

2 Truck Scheduling Models

It is well known that scheduling problems can be formulated in different ways, depending on the variables one chooses (see [8]). Among them, the most used are: natural-dates variables, time-indexed variables and positional variables. We will briefly recall their definitions, referring, for simplicity, to a non-preemptive single machine scheduling problem.

Assuming we are given a set $N = \{1, \ldots, j, \ldots, n\}$ of jobs to be processed on the machine, and their processing times p_j, the *natural-date* variables are the n continuous variables C_j, $j \in N$, where C_j is the completion time of the job j. Very often such variables are used in conjunction with the *linear ordering* variables, defined as follows:

$$\delta_{ij} = \begin{cases} 1 & \text{if the job } i \text{ precedes the job } j \text{ in the schedule} \\ 0 & \text{otherwise} \end{cases} \qquad \forall i, j \in N,\ i \neq j$$

so that, in each permutation schedule (without idle time), it results:

$$C_j = \sum_{i \in N, i \neq j} \delta_{ij} p_i + p_j \qquad \forall j \in N$$

Otherwise, the scheduling model will necessarily include disjunctive constraints.

When *time-indexed* variables are used, the time horizon T is discretized into a finite number of periods $T = \{1, 2, \ldots, T_f\}$, and the variables are defined as follows:

$$x_{jt} = \begin{cases} 1 & \text{if the processing of job } j \text{ starts in the period } t \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \in N, t \in T$$

The completion time of each job is, then, the following linear function of the x_{jt} variables:

$$C_j = \sum_{t=1}^{T_f - p_j} t x_{jt} + p_j \qquad \forall j \in N$$

A schedule may be also identified by the positions occupied by the jobs in the sequence. Introducing a set $K = \{1, \ldots, k, \ldots, n\}$ of position indexes, the *positional* variables are defined as follows:

$$x_{jk} = \begin{cases} 1 & \text{if the job } j \text{ is processed as the } k\text{-th one in the sequence} \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \in N, k \in K$$

Note that, in this framework, it is easy to identify the completion time of the k-th job in the sequence (*positional* completion time):

$$C_{[k]} = C_{[k-1]} + \sum_{j \in N} p_j x_{jk} \qquad \forall k \in K$$

and the makespan, which is a nonlinear function of the natural C_j's, is simply

$$C_{max} = C_{[n]}$$

On the contrary, the natural completion times C_j, $j \in N$, can not be expressed as linear functions of the variables x_{jk}.

In [6], where positional variables have been introduced, it is shown that the positional formulations of some single-machine scheduling problems can outperform those based on time discretization, while requiring much fewer variables. Our aim is to investigate if similar conclusions can be drawn for scheduling problems in more complex frameworks.

Observe that if jobs have unitary processing times, the time-indexed and positional formulations coincide. The truck scheduling problem with unitary processing times has been investigated in [3] for the single door cross docking setting, and in [5] for the multi-door case. In [4] the authors have proposed a time-indexed formulation for the two-machine cross-docking flow-shop scheduling problem with arbitrary processing times. Here we propose a formulation that makes use of positional variables, instead.

In the following Table 1 we introduce our main notation that will be shared by the two formulations. Additional notation will be properly introduced when needed.

Table 1 Notation

Name	Definition
I	Set of inbound trucks; $\|I\| = n$
J	Set of outbound trucks; $\|J\| = m$
p_i^I	Processing time of the inbound truck $i \in I$
p_j^O	Processing time of the outbound truck $j \in J$
$I_j \subseteq I$	Set of inbound trucks supplying the outbound truck $j \in J$ ($\|I_j\| \geq 1$)

2.1 Time-Indexed Formulation

The Time-Indexed formulation (TI) proposed in [4] requires some additional notation. Let $T = \{0, \ldots, T_f\}$ be the discretized planning horizon, where the ending time T_f is computed as $T_f = \sum_{i \in I} p_i^I + \sum_{j \in J} p_j^O$. Let T_0 be a lower bound on the starting service time for the outbound trucks, defined as follows

$$T_0 = \min_{j \in J} \left\{ \sum_{i \in I_j} p_i^I \right\} \tag{1}$$

By means of the following decision variables,

- $x_{it} = 1$, if the processing of inbound truck $i \in I$ starts at time $t \in T$; 0 otherwise
- $y_{jt} = 1$, if the processing of outbound truck $j \in J$ starts at time $t \in T$; 0 otherwise
- $C_{max} \geq 0$, the makespan

the TI formulation reads as follows

$$Z = \min \quad C_{max} \tag{2}$$

$$\sum_{t=0}^{T_f - p_i^I} x_{it} = 1 \qquad i \in I \tag{3}$$

$$\sum_{i \in I} \sum_{s=\max\{0; t-p_i^I+1\}}^{t} x_{is} \leq 1 \qquad t \in T \tag{4}$$

$$\sum_{t=T_0}^{T_f - p_j^O} y_{jt} = 1 \qquad j \in J \tag{5}$$

$$\sum_{j \in J} \sum_{s=\max\{T_0; t-p_j^O+1\}}^{t} y_{js} \leq 1 \qquad t \in T \tag{6}$$

$$\sum_{t=T_0}^{T_f - p_j^O} t y_{jt} - \sum_{t=0}^{T_f - p_i^I} (t + p_i^I) x_{it} \geq 0 \qquad j \in J, i \in I_j \tag{7}$$

$$C_{max} \geq p_j^O + \sum_{t=T_0}^{T_f - p_j^O} t y_{jt} \qquad j \in J \tag{8}$$

$$x_{it} \in \{0, 1\} \qquad i \in I, t \in T \tag{9}$$

$$y_{jt} \in \{0, 1\} \qquad j \in J, t \in T \tag{10}$$

In model TI, the objective function (2) is the makespan. Constraints (3) impose that the processing of each inbound truck starts at exactly one time-slot t such that it is completed within the time horizon. Constraints (4) ensure that two inbound trucks cannot be processed simultaneously. Constraints (5) and (6) are the counterpart of constraints (3) and (4) on the outbound side. Inequalities (7) are the cross-docking constraints, imposing the precedence relation between the processing of the inbound truck i and the processing of the outbound truck j, if j requires items transported by i ($i \in I_j$). Constraints (8) define the makespan; (9) and (10) are the domain of the variables.

2.2 Position-Indexed Formulation

Let $K = \{1, \ldots, n\}$ and $H = \{1, \ldots, m\}$ be two sets of position indexes for the processing of the inbound and outbound trucks, respectively. We define the following constants

$$M^I = \sum_{i \in I} p_i^I \qquad M^O = \sum_{j \in J} p_j^O \qquad M = M^I + M^O \tag{11}$$

Defining the binary positional variables

- $x_{ik} = 1$, if the truck $i \in I$ is the k-th one in the inbound sequence, 0 otherwise;
- $y_{jh} = 1$, if the truck $j \in J$ is the h-th one in the outbound sequence, 0 otherwise;

and the continuous variables

- $C_i^I \geq 0$, the completion time of the inbound truck $i \in I$;
- $C_j^O \geq 0$, the completion time of the outbound truck $j \in J$;
- $C_{max} \geq 0$, the makespan

the Position-Indexed (PI) formulation of the truck scheduling problem is as follows

$$Z = \min \quad C_{max} \tag{12}$$

$$\sum_{k \in K} x_{ik} = 1 \qquad i \in I \tag{13}$$

$$\sum_{i \in I} x_{ik} = 1 \qquad k \in K \tag{14}$$

$$C_i^I - p_i^I \geq C_l^I - M^I(2 - x_{ik} - x_{l\,k-1}) \qquad i \neq l \in I, k \in K \setminus \{1\} \tag{15}$$

$$C_i^I - p_i^I \geq 0 \qquad i \in I \tag{16}$$

$$\sum_{h \in H} y_{jh} = 1 \qquad j \in J \tag{17}$$

$$\sum_{j \in J} y_{jh} = 1 \qquad h \in H \tag{18}$$

$$C_j^O - p_j^O \geq C_l^O - M(2 - y_{jh} - y_{l\,h-1}) \qquad j \neq l \in J, h \in H \setminus \{1\} \tag{19}$$

$$C_j^O - p_j^O \geq C_i^I \qquad j \in J, i \in I_j \tag{20}$$

$$C_{max} \geq C_j^O \qquad j \in J \tag{21}$$

$$x_{ik} \in \{0, 1\} \qquad i \in I, k \in K \tag{22}$$

$$y_{jh} \in \{0, 1\} \qquad j \in J, h \in H \tag{23}$$

In this model the objective function (12) is the makespan. Constraints (13) and (14) ensure that each inbound truck is assigned to one and only one sequence position. Constraints (15) correctly compute the completion time of the inbound truck i if i is the successor of l in the sequence; otherwise they are dominated by (16). Moreover, constraints (16) play the role of (15) for the inbound truck assigned to the first position ($k = 1$). Constraints (17)–(19) play the same role of constraints (13)–(15) on the outbound side. Note that the constraints $C_j^O - p_j^O \geq 0$, $j \in J$, corresponding to (16), do not appear because they are dominated by (20). Constraints (20) impose the precedence relationships between inbound and outbound trucks involved in transhipment operations. Finally, constraints (21) define the makespan, while the remaining two groups define the domain of the binary variables. The non-negativity constraints on C_i^I and C_j^O are useless because of constraints (16) and (20).

2.3 Strengthening of the PI Formulation

The formulation (12)–(23) can be strengthened by adding some valid inequalities, as described below.

Constraints (16) impose an obvious and weak lower bound on the completion times of the inbound trucks $i \in I$. To get a stronger lower bound, we can relate C_i^I to the processing times of the trucks preceding i in the sequence. This results in the following constraints

$$C_i^I - p_i^I \geq \sum_{\substack{l \in I \\ l \neq i}} \sum_{q=1}^{k-1} p_l^I x_{lq} - M^I (1 - x_{ik}) \qquad i \in I, k \in K \setminus \{1\} \tag{24}$$

By this way, when $x_{ik} = 1$, the corresponding constraint (24) imposes a non trivial lower bound on the starting time of truck i; when $x_{ik} = 0$ the corresponding constraints are dominated by (16).

In order to strengthen the constraints defining the completion times of the outbound trucks, we observe that each $j \in J$ has to wait for the completion of operations on all inbound trucks $i \in I_j$, that is the following are valid inequalities:

$$C_j^O - p_j^O \geq \sum_{i \in I_j} p_i^I \qquad j \in J \tag{25}$$

On the other hand, the counterpart of constraints (24) for the outbound trucks, i.e.:

$$C_j^O - p_j^O \geq \sum_{\substack{l \in J \\ l \neq j}} \sum_{q=1}^{h-1} p_l^O y_{lq} - M^O (1 - y_{jh}) \qquad j \in J, h \in H \setminus \{1\}$$

are useless since they are dominated by (19). Actually, it is sufficient to note that, when $y_{jh} = 1$ and $y_{l\,h-1} = 1$, it is

$$C_l^O \geq \sum_{\substack{l \in J \\ l \neq j}} \sum_{q=1}^{h-1} p_l^O y_{lq}$$

because of possible idle times on the outbound door, due to the cross-docking constraints.

Therefore, we come out with a stronger formulation, say PI^*, obtained by adding to the (PI) model (12)–(23), the constraints (24) and (25).

Note that PI^* is not a fully positional formulation, since it uses the natural-dates completion times C_i^I and C_j^O together with the job-to-position assignment variables x_{ik} and y_{jh}. This is because the cross-docking precedence constraints relate each

outbound truck $j \in J$ with a subset I_j of the inbound trucks, and their completion times cannot be expressed as linear functions of the positional dates and assignment variables (see [6]).

In closing this section we want to observe that the Time-Indexed formulation by [4] is characterized by a number of variables that grows obviously with the number of inbound and outbound trucks, but also with the size of the planning horizon, corresponding to the sum of all the truck processing times. This is the main well-known drawback of all time-indexed formulations of scheduling problems. On the contrary, the formulation PI^* has a lower number of variables, depending only on the total number of trucks, but a higher number of constraints, caused by the combined use of positional and natural-date variables.

3 Computational Results

In order to assess the effectiveness of the positional formulation PI^*, we have used a set of 300 test instances extracted from the set of 500 benchmark instances defined in [4]. They are partitioned in two groups: G1 and G2, sharing the same values of n, m and $|I_j|$, but differing in the size of the processing times. For the instances in G1 p_i^I and p_j^O are in the range [1, 10], while in G2 they are in the range [10, 100].

On these instances we have first compared PI^* with PI and TI in terms of linear programming lower bounds. In order to evaluate the performance of our model in the most critical cases for the TI model, we have also compared the upper bounds obtained by PI^* on the instances in G2 with those reported in [4]. We have used Cplex 12.8 both as LP and MILP solver. The experiments have been run on a machine equipped with a 3.1 GHz CPU and 16 GB of RAM. We have imposed a time limit of 3600 s to Cplex when solving the model PI^*.

From Tables 2 and 3 it is evident that, as expected, PI^* is by far stronger than PI, and dominates TI in all cases but one. Actually, on the G1 instances the lower bounds are uniformly higher than those corresponding to the TI model: the percentage increase, computed as $(AvgLB(PI^*) - AvgLB(TI))/AvgLB(TI)$, ranges between 8.72% (Instances 10–14), and 82.19% (Instances 60–36). Furthermore, the computation times are smaller in all instances, except for the largest ones. A similar behavior can be observed in Table 3 for the G2 instances. The percentage increase in the average lower bound attains its maximum value 82.74% on the Instances 60–36, while its minimum value is −6.44% (Instances 40–24), meaning that on this group of 10 instances TI performs better than PI^*. However the computation times required by PI^* are uniformly lower by at least one order of magnitude.

In Table 4 we report the comparison on the upper bounds in terms of Cplex's optimality Gaps and CPU running times on the G2 instances. As for the PI^* model, Cplex has been able to solve all the instances with $n = 5$ in a negligible time, and almost all the instances with $n = 10$, even though with a remarkable increase in the computation time, probably due to the use of big M in the constraints. As expected, the TI model can not even be solved at optimality on very small size instances

Table 2 Lower bound comparison on G1 instances

		Formulation PI			Formulation PI^*			Formulation TI		
		LB		Time (s)	LB		Time (s)	LB		Time (s)
n	m	Max	Avg	Avg	Max	Avg	Avg	Max	Avg	Avg
5	3	19.00	16.00	0.00	39.00	28.60	0.00	30.17	24.53	0.00
	5	20.00	15.20	0.00	41.00	28.70	0.00	32.55	23.50	0.00
	7	20.00	16.30	0.00	49.00	39.70	0.00	34.71	27.17	0.01
10	6	20.00	17.30	0.00	63.00	47.60	0.00	44.78	36.54	0.02
	10	20.00	18.20	0.00	76.00	54.90	0.00	50.00	39.82	0.05
	14	20.00	17.30	0.00	66.00	52.60	0.01	61.94	48.38	0.09
20	12	20.00	18.90	0.03	132.00	102.60	0.05	54.29	65.68	0.93
	20	20.00	19.40	0.05	130.00	105.20	0.08	80.71	70.72	2.11
	28	20.00	19.60	0.10	115.00	103.70	0.15	98.36	84.94	1.24
40	24	20.00	19.80	0.28	255.00	213.80	0.97	139.00	122.28	4.56
	40	20.00	19.90	0.46	256.00	221.70	1.56	147.79	126.48	5.84
	56	20.00	19.90	0.95	253.00	217.90	3.86	169.43	160.96	9.48
60	36	20.00	20.00	1.11	342.00	321.10	7.95	195.48	176.25	11.91
	60	20.00	20.00	1.84	364.00	322.80	13.96	195.59	183.77	20.83
	84	20.00	20.00	4.04	367.00	324.20	68.93	255.59	238.64	44.57

Table 3 Lower bound comparison on G2 instances

		Formulation PI			Formulation PI^*			Formulation TI		
		LB		Time (s)	LB		Time (s)	LB		Time (s)
n	m	Max	Avg	Avg	Max	Avg	Avg	Max	Avg	Avg
5	3	196.00	145.10	0.00	337.00	239.90	0.00	298.16	225.51	0.44
	5	196.00	149.20	0.00	376.00	253.40	0.00	319.76	232.36	0.71
	7	196.00	159.00	0.00	441.00	294.10	0.00	341.03	271.10	1.01
10	6	197.00	168.80	0.01	623.00	476.70	0.00	442.26	362.79	5.87
	10	197.00	172.90	0.00	628.00	505.50	0.01	478.28	399.33	6.73
	14	197.00	172.80	0.01	656.00	526.90	0.01	599.66	482.23	11.39
20	12	200.00	184.10	0.03	1290.00	1025.00	0.07	753.83	654.33	17.66
	20	197.00	188.10	0.05	1268.00	1047.90	0.09	790.48	701.24	28.67
	28	197.00	190.50	0.09	1242.00	1064.50	0.18	1010.86	872.81	51.06
40	24	198.00	190.40	0.28	1308.00	1128.00	1.16	1349.72	1205.68	80.73
	40	198.00	193.40	0.49	2411.00	2174.00	1.86	1353.50	1237.32	163.07
	56	200.00	194.90	1.12	2463.00	2173.50	5.83	1686.53	1609.82	267.03
60	36	199.00	195.40	1.17	3393.00	3217.10	7.90	1921.71	1760.46	281.56
	60	199.00	197.50	2.00	3578.00	3231.50	14.30	1923.15	1828.36	420.14
	84	199.00	197.40	4.25	3608.00	3243.00	72.82	2542.00	2384.64	776.25

Table 4 Upper bound comparison on G2 instances

		Formulation PI^*				Formulation TI			
		Gap%		Time (s)		Gap%		Time (s)	
n	m	Min	Avg	Min	Avg	Min	Avg	Min	Avg
5	3	0.00	0.00	0.01	0.00	0.00	0.00	6.70	21.00
	5	0.00	0.00	0.02	0.19	0.00	0.80	32.00	1570.00
	7	0.00	0.00	0.04	2.43	0.00	13.10	94.90	2882.50
10	6	0.00	0.00	1.88	4.52	28.20	28.70	2801.10	3509.90
	10	0.00	0.94	831.59	2407.61	35.70	45.20	3600.00	3600.00
	14	0.00	10.26	66.33	3246.89	53.20	73.60	3600.00	3600.00
20	12	8.02	14.26	3600.00	3600.00	NA	NA	3600.00	3600.00
	20	21.43	31.64	3600.00	3600.00	NA	NA	3600.00	3600.00
	28	37.31	45.79	3600.00	3600.00	NA	NA	3600.00	3600.00
40	24	36.58	53.82	3600.00	3600.00	NA	NA	3600.00	3600.00
	40	37.02	40.17	3600.00	3600.00	NA	NA	3600.00	3600.00
	56	47.98	52.11	3600.00	3600.00	NA	NA	3600.00	3600.00
60	36	26.23	31.63	3600.00	3600.00	NA	NA	3600.00	3600.00
	60	43.77	47.79	3600.00	3600.00	NA	NA	3600.00	3600.00
	64	52.09	57.06	3600.00	3600.00	NA	NA	3600.00	3600.00

NA means that the corresponding values are not reported in [4]

($n = 5, m = 7$). For $n = 10$ it returns uniformly larger Gaps in higher average computation times.

On all the remaining instances PI^* outperforms TI, given that in [4] no Gap is reported for these cases. This is not surprising, because the number of variables in TI is $(n + m)T_f + 1$ and for the largest instances of the G2 group it is about 1.5 millions.

4 Conclusions

In this paper we have compared two formulations for a truck scheduling problem in a single door cross-docking terminal. Despite the hybrid nature of our model for a non-standard flow shop scheduling problem, the computational experience seems to confirm the effectiveness of scheduling problem formulations based on positional variables against those using time-indexed variables.

Acknowledgements The research has been supported by Ministero dell'Istruzione, Universitá e Ricerca under the PRIN 2015 research program—Grant 2015XAPRKF—Smart PORt Terminals.

References

1. Boysen, N., Fliedner, M.: Cross dock scheduling: classification, literature review and research agenda. Omega **38**(6), 413–422 (2010)
2. Buijs, P., Vis, I.F., Carlo, H.J.: Synchronization in cross-docking networks: a research classification and framework. Eur. J. Oper. Res. **239**(3), 593–608 (2014)
3. Chiarello, A., Gaudioso, M., Sammarra, M.: Truck synchronization at single door cross-docking terminals. OR Spectr. **40**(2), 395–447 (2018)
4. Fonseca, G.B., Nogueira, T.H., Ravetti, M.G.: A hybrid Lagrangian metaheuristic for the cross-docking flow shop scheduling problem. Eur. J. Oper. Res. **275**, 139–154 (2019)
5. Gaudioso, M., Monaco, M.F., Sammarra, M.: A decomposition-based Heuristic for the truck scheduling problem in a cross-docking terminal. In: Daniele, P., Scrimali, L. (eds.) New Trends in Emerging Complex Real Life Problems: ODS, Taormina. AIRO Springer Series, vol 1, pp. 265–273. Springer, Cham (2018)
6. Lasserre, J.B., Queyranne M.: Generic scheduling polyhedra and a new mixed-integer formulation for single-machine scheduling. In: Balas, E., Cornuejols, G., Kannan, R. (eds.) Proceedings of the 2nd IPCO Conference, Pittsburgh, Carnegie Mellon University, May 1992, pp. 136–149 (1992)
7. Ladier, A.-L., Alpan, G.: Cross-docking operations: current research versus industry practice. Omega **62**, 145–162 (2016)
8. Queyranne, M., Schulz, A.: Polyhedral approaches to machine scheduling. Technical Report Technical University of Berlin (1996). http://web.mit.edu/schulz/www/epapers/queyranne-schulz.pdf
9. Van Belle, J., Valckenaers, P., Cattrysse, D.: Cross-docking: state of the art. Omega **40**(6), 827–846 (2012)

Optimization of Car Traffic in Emergency Conditions

Luigi Rarità

Abstract In this work, the aim is to present a possible methodology to redistribute car traffic, modelled via a fluid dynamic approach, within a part of the Caltanissetta city (Italy), when critical events, such as car accidents, occur. Adopting a decentralized approach, a cost functional, that measures the asymptotic average velocity of emergency vehicles, is maximized with respect to traffic parameters at nodes with two incoming and outgoing roads. Then, the management of high traffic is analyzed through local optimal coefficients at each node of the network. The whole traffic dynamics is studied by simulations, which confirm the correctness of the optimization procedure. It is also shown that optimal parameters allow a fast transit of emergency vehicles on assigned paths on the network.

Keywords Conservation laws · Optimal paths for emergency vehicles · Simulation

1 Introduction

Various phenomena, such as queues formations, long travel times, pollution and so on, often characterize road networks. Sometimes, high traffic levels lead to car accidents, with consequent serious issues in terms of emergency management. In such cases, suitable methodologies for road traffic in emergency conditions are often analyzed. The aim of this work is to consider some optimization results within a part of Caltanissetta urban network, Italy, for the redistribution of car flows so that emergency vehicles could cross assigned roads at the maximum possible speed.

A fluid dynamic model is considered where the dynamics of car densities on roads is modelled by conservation laws [4, 6, 8], while phenomena at $n \times m$ junctions (namely nodes with n incoming roads and m outgoing roads) are treated via rules

L. Rarità (✉)
Dipartimento di Ingegneria Industriale, University of Salerno, Fisciano (SA), Italy
e-mail: lrarita@unisa.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_27

of traffic distributions and right of ways (if $n > m$). Considering the distribution coefficients as control parameters, we aim to redirect traffic at 2×2 nodes in order to manage emergency conditions. Hence, under the assumption that emergency vehicles could cross assigned paths [7], a cost functional $V_{(a,b)}$, that indicates, for 2×2 nodes, the average velocities of such vehicles on the incoming road I_a, $a \in \{1, 2\}$, and the outgoing road I_b, $b \in \{3, 4\}$, is considered. The optimization results provide the distribution coefficients that maximize the functional and allow a fast transit of emergency vehicles to reach hospitals and/or the places of car accidents.

As, for complex networks, the analysis of $V_{(a,b)}$ represents a hard task, a decentralized methodology is adopted, namely: the asymptotic dynamics (for very large times) is considered and an exact solution for $V_{(a,b)}$ is obtained for a node of 2×2 type. Then, we get a global (sub)optimal solution for the whole network by applying simply the obtained local optimal solution at each node with two incoming roads and two outgoing roads. Similar procedures have been also studied for other road junctions and different functionals, see [2, 3], and [5], as well as suitable numerical approaches are described in [9] and [10].

Simulations are useful to test the proposed approach. In particular, two different choices of distribution coefficients are evaluated: Results achieved by the optimization algorithm; Random parameters, namely: at the beginning of the simulation process, values of traffic parameters are randomly chosen and then kept constant during the simulation. For the case study of a part of the Caltanissetta urban network in Italy, the choice of optimal distribution coefficients at 2×2 nodes allows obtaining better performances on the network. Finally, following an algorithm described in [1] to trace car trajectories on networks, further simulations are run to test if distribution coefficients provide variations of the total travelling time for emergency vehicles. It is proved that times to cover a path of a single emergency vehicle decrease if optimal coefficients are adopted.

The paper is structured as follows. Section 2 introduces the model for car traffic. Section 3 focuses on the cost functional for emergency vehicles and the optimization of traffic parameters. Section 4 presents the simulations for the case study. Conclusions end the paper in Sect. 5.

2 A Model for Traffic on Road Networks

A road network is seen as a couple $(\mathscr{R}, \mathscr{N})$, where $\mathscr{R}$ and $\mathscr{N}$ are, respectively, the set of roads, indicated by intervals $[\eta_i, \theta_i] \subset \mathbb{R}$, $i = 1, \ldots, M$, and the set of nodes. Assuming that: $D = D(t, x) \in [0, D_{\max}]$ is the density of cars, where $D_{\max}$ is the highest possible density; $f(D) = Dv(D)$ is the flux where $v(D)$ denotes the average velocity, the traffic on each road is represented by the conservation law (Lighthill-Whitham-Richards model, [6, 8]):

$$\frac{\partial D}{\partial t} + \frac{\partial f(D)}{\partial x} = 0. \tag{1}$$

We assume that: (F) f is a strictly concave C^2 function such that $f(0) = f(D_{\max}) = 0$. Fixing the decreasing velocity function:

$$v(D) = v_{\max}\left(1 - \frac{D}{D_{\max}}\right), D \in [0, D_{\max}], \tag{2}$$

a flux function, that respects (F), is, for $v_{\max} = D_{\max} = 1$:

$$f(D) = D(1 - D), \quad D \in [0, 1], \tag{3}$$

that presents a unique maximum $\sigma = \frac{1}{2}$.

Riemann Problems (RPs), i.e. Cauchy Problems with a constant initial datum for incoming and outgoing roads, are useful to solve the dynamics at nodes.

Focus on a node J of $n \times m$ type (n incoming roads I_a, $a = 1, \ldots, n$, and m outgoing roads, I_b, $b = n+1, \ldots, n+m$) and an initial datum indicated by $D_0 = \left(D_{1,0}, \ldots, D_{n,0}, D_{n+1,0}, \ldots, D_{n+m,0}\right)$.

A Riemann Solver (RS) for J is a map $RS : [0,1]^n \times [0,1]^m \to [0,1]^n \times [0,1]^m$ that associates to D_0 a vector $\widehat{D} = \left(\widehat{D}_1, \ldots, \widehat{D}_n, \widehat{D}_{n+1}, \ldots, \widehat{D}_{n+m}\right)$ so that the wave $\widetilde{D}_a = \left(D_{a,0}, \widehat{D}_a\right)$ is solution for an incoming road I_a, $a = 1, \ldots, n$, while the wave $\widetilde{D}_b = \left(\widehat{D}_b, D_{b,0}\right)$ is solution for an outgoing road I_b, $b = n+1, \ldots, n+m$. The following conditions are required: (C1) $RS(RS(D_0)) = RS(D_0)$; (C2) for a generic incoming (resp. outgoing) road, the wave $\widetilde{D}_a$ (resp. $\widetilde{D}_b$) has negative (resp. positive) speed.

If $m \geq n$, a suitable RS at J is defined via the rules [4]:

(A) Traffic distributes at J according to some parameters, collected in a traffic distribution matrix $A = \left(\alpha_{b,a}\right)$, $a = 1, \ldots, n$, $b = n+1, \ldots, n+m$, $0 < \alpha_{b,a} < 1$, $\sum_{b=n+1}^{n+m} \alpha_{b,a} = 1$. The $a-$th column of A provides the percentages of traffic that, from the incoming road I_a, goes to the outgoing roads;
(B) Respecting (A), drivers maximize the flux through the node J.

If $n > m$, a further rule (yielding criterion) is necessary:

(C) Assume that S is the amount of cars that can enter the outgoing roads. Then, $p_a S$ cars come from the incoming road I_a, where $p_a \in]0,1[$, $\sum_{a=1}^{n} p_a = 1$, represents the right of way parameter for road I_a, $a = 1, \ldots, n$.

For the particular case of a node J of 2×2 type (incoming roads I_1 and I_2, and outgoing roads I_3 and I_4), assume that the densities of cars for incoming and outgoing roads are, respectively, $D_a(t,x) \in [0,1]$, $(t,x) \in \mathbb{R}^+ \times I_a$, $a = 1, 2$, and $D_b(t,x) \in [0,1]$, $(t,x) \in \mathbb{R}^+ \times I_b$, $b = 3, 4$. From condition (C2),

for the flux function (3) and initial datum of an RP at J indicated by $D_0 = \left(D_{1,0}, D_{2,0}, D_{3,0}, D_{4,0}\right)$, we get that the maximal flux values on roads are:

$$\gamma_\chi^{\max} = \begin{cases} f\left(D_{\chi,0}\right), & \text{if } 0 \le D_{\chi,0} \le \frac{1}{2} \text{ and } \chi = 1,2, \\ & \text{or } \frac{1}{2} \le D_{\chi,0} \le 1 \text{ and } \chi = 3,4, \\ f\left(\frac{1}{2}\right), & \text{if } \frac{1}{2} \le D_{\chi,0} \le 1 \text{ and } \chi = 1,2, \\ & \text{or } 0 \le D_{\chi,0} \le \frac{1}{2} \text{ and } \chi = 3,4. \end{cases}$$

In this case, matrix A has the coefficients $\alpha_{3,1}, \alpha_{3,2}, \alpha_{4,1} = 1-\alpha_{3,1}, \alpha_{4,2} = 1-\alpha_{3,2}$, and the assumption $\alpha_{3,1} \neq \alpha_{3,2}$ guarantees the uniqueness of solutions. From rules (A) and (B), the flux solution to the RP at J, $\widehat{\gamma} = (\widehat{\gamma}_1, \widehat{\gamma}_2, \widehat{\gamma}_3, \widehat{\gamma}_4)$, is found as follows: the incoming fluxes $\widehat{\gamma}_a$, $a = 1,2$, are solutions of the problem $\max(\gamma_1 + \gamma_2)$, with $0 \le \gamma_a \le \gamma_a^{\max}$, $a = 1,2, 0 \le \alpha_{b,1}\gamma_1 + \alpha_{b,2}\gamma_2 \le \gamma_b^{\max}$, $b = 3,4$. The outgoing fluxes $\widehat{\gamma}_b$ are simply obtained as $\widehat{\gamma}_b = \alpha_{b,1}\widehat{\gamma}_1 + \alpha_{b,2}\widehat{\gamma}_2$, $b = 3,4$. Once $\widehat{\gamma}$ is known, $\widehat{D}$ is found as:

$$\widehat{D}_\chi \in \begin{cases} \left\{D_{\chi,0}\right\} \cup \left]\tau\left(D_{\chi,0}\right), 1\right], & \text{if } 0 \le D_{\chi,0} \le \frac{1}{2} \text{ and } \chi = 1,2, \\ & \text{or } \frac{1}{2} \le D_{\chi,0} \le 1 \text{ and } \chi = 3,4, \\ \left[0, \frac{1}{2}\right], & \text{if } 0 \le D_{\chi,0} \le \frac{1}{2}, \chi = 3,4, \\ \left[\frac{1}{2}, 1\right], & \text{if } \frac{1}{2} \le D_{\chi,0} \le 1, \chi = 1,2, \end{cases}$$

where $\tau : [0,1] \to [0,1]$ is the map such that $f(\tau(D)) = f(D)\ \forall\ D \in [0,1]$ and $\tau(D) \neq D\ \forall\ D \in [0,1] \setminus \left\{\frac{1}{2}\right\}$.

3 Optimal Coefficients for Traffic Dynamics

Assume that some car accidents occur on a urban network and that emergency vehicles need to reach a hospital and/or the places of the accidents. For the emergency vehicles, the following velocity function is considered:

$$\zeta(D) = 1 - \lambda + \lambda v(D), \tag{4}$$

where $0 < \lambda < 1$ and $v(D)$ follows (2). As $\zeta(D_{\max}) = 1 - \lambda > 0$, then the emergency vehicles have higher velocities than cars. For a node J with incoming roads I_1 and I_2 and outgoing roads I_3 and I_4, for a fixed initial datum $\left(D_{1,0}, D_{2,0}, D_{3,0}, D_{4,0}\right)$, the cost functional $V_{(a,b)}(t)$, that indicates the average velocity of emergency vehicles that cross the incoming road I_a, $a \in \{1,2\}$, and the outgoing road I_b, $b \in \{3,4\}$, is defined as:

$$V_{(a,b)}(t) := \int_{I_a} \zeta\left(D_a(t,x)\right) dx + \int_{I_b} \zeta\left(D_b(t,x)\right) dx. \tag{5}$$

If $a = 1$ and $b = 3$, we get the following theorem, whose proof is in [7] (the statement is similar for different combinations of a and b).

Theorem 1 *Fix a node J with incoming roads I_1 and I_2, and outgoing roads I_3 and I_4. For a time $t >> 0$, the coefficients $\alpha_{3,1}$ and $\alpha_{3,2}$, that maximize $V_{(1,3)}(t)$, are $\alpha_{3,1}^{opt} = 1 - \frac{\gamma_4^{\max}}{\gamma_1^{\max}}$, $0 \leq \alpha_{3,2}^{opt} < 1 - \frac{\gamma_4^{\max}}{\gamma_1^{\max}}$, with the exception of the following cases, where the optimal values do not exist and, for ξ_1 and ξ_2 small, positive and such that $\xi_1 \neq \xi_2$, are approximated as: $\alpha_{3,1}^{opt} = \xi_1$, $\alpha_{3,2}^{opt} = \xi_2$ if $\gamma_1^{\max} \leq \gamma_4^{\max}$; $\alpha_{3,1}^{opt} = \frac{\gamma_3^{\max}}{\gamma_3^{\max}+\gamma_4^{\max}} - \xi_1$, $\alpha_{3,2}^{opt} = \frac{\gamma_3^{\max}}{\gamma_3^{\max}+\gamma_4^{\max}} - \xi_2$ if $\gamma_1^{\max} > \gamma_3^{\max} + \gamma_4^{\max}$.*

4 Simulations

The optimization results, foreseen by Theorem 1, are studied via different control choices, applied locally at each node, on the global behaviour of a real network. This analysis is then completed by computing the travelling times of an emergency vehicle on assigned routes.

We focus on a part of the urban network of Caltanissetta, Italy (see Fig. 1).

The network has: 8 roads, described by 51 segments (see Table 1). Eight of them are incoming roads (1, 5, 23, 27, 35, 39, 46, 51), and nine of them identify outgoing roads (2, 4, 8, 22, 25, 34, 37, 44, 49). Moreover, there are 25 nodes of different types: 2×2, labelled as A_i, $i = 1, \ldots, 11$; 2×1, identified by B_i, $i = 1, \ldots, 6$;

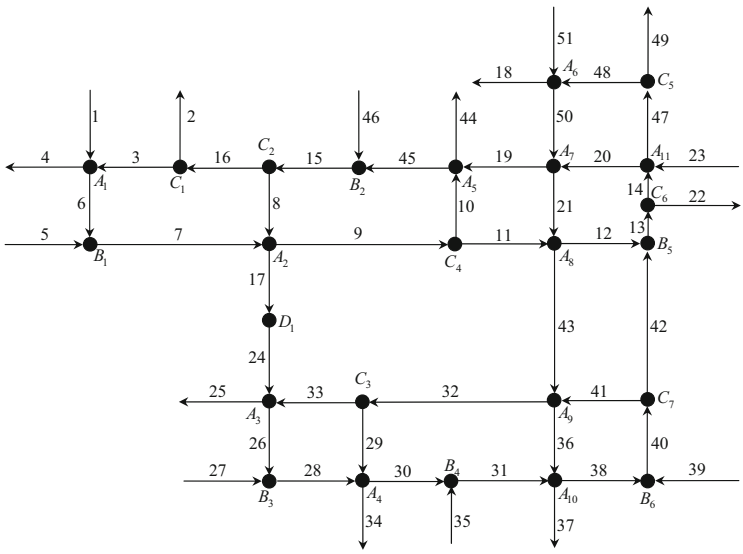


Fig. 1 Topology of the considered network within Caltanissetta, Italy

Table 1 Correspondence among numbers and roads in Fig. 1

Road	Graph segments
Via Giuseppe Mulè	1, 2
Via Luigi Monaco	3–21
Via della Regione	22, 23
Via Due Fontane	24–33
Via SD1	34, 35
Via Leone XIII	36–43
Via Luigi Russo	44, 45, 46
Via Poggio S. Elia	47–51

1×2, indicated by C_i, $i = 1, \ldots, 7$; 1×1, D_1. We assume that emergency vehicles could cross the path $\rho = \Omega_1 \cup \Omega_2 \cup \Omega_3 \cup \Omega_4$, with:

$$\Omega_1 = \{23, 47, 48, 50, 19, 45, 15\}\,,\ \Omega_2 = \{16, 3, 6, 7, 17, 24, 26\}\,,$$
$$\Omega_3 = \{28, 30, 31, 38, 40, 42, 13\}\,,\ \Omega_4 = \{14, 20, 21, 43, 32, 33, 35\}\,.$$

Hence, we analyze the behaviour of the cost functional $E(t) = \sum_{(a,b)\in\Gamma} V_{(a,b)}(t)$, with $V_{(a,b)}(t)$ as in (5) and:

$$\Gamma := \left\{ \begin{array}{l} (23, 47)\,,\ (48, 50)\,,\ (50, 19)\,,\ (19, 45)\,,\ (3, 6)\,,\ (7, 17)\,, \\ (24, 26)\,,\ (28, 30)\,,\ (31, 38)\,,\ (20, 21)\,,\ (43, 32)\,,\ (33, 25) \end{array} \right\}.$$

Traffic flows simulations are made by the Godunov method with $\Delta x = 0.0125$, $\Delta t = 0.5\Delta x$ in a time interval $[0, T]$, with $T = 120$ min. Initial conditions and boundary data for densities are chosen approaching $D_{\max} = 1$, with the aim of simulating a congestion scenario on the network, as follows: initial datum equal to 0.85 for all roads; boundary data 0.95 for roads 1, 5, 23, 27 and 35; 0.9 for roads 39, 46, and 51; 0.85 for roads 2, 4, 18, 22 and 25; 0.9 for roads 34, 37, 44 and 49. Considering some measures on the real network, we have, for nodes B_i, $i = 1, \ldots, 6$, the right of way parameters: $p_{12} = p_{26} = 0.2$, $p_{46} = 0.3$, $p_6 = p_{35} = 0.4$, $p_{38} = p_{39} = 0.5$, $p_5 = p_{30} = 0.6$, $p_{45} = 0.7$, $p_{42} = p_{27} = 0.8$; for nodes C_i, $i = 1, \ldots, 7$, the distribution coefficients: $\alpha_{41,40} = 0.2$, $\alpha_{49,47} = \alpha_{22,13} = 0.3$, $\alpha_{8,15} = \alpha_{33,32} = 0.4$, $\alpha_{2,16} = \alpha_{3,16} = \alpha_{10,9} = \alpha_{11,9} = 0.5$, $\alpha_{16,15} = \alpha_{29,32} = 0.6$, $\alpha_{48,47} = \alpha_{14,13} = 0.7$, $\alpha_{42,40} = 0.8$. Finally, $\lambda = 0.5$ is used.

We consider two different simulation cases: locally optimal distribution parameters (*optimal case*) at each node A_i $i = 1, \ldots, 11$, i.e. coefficients that follow Theorem 1; random parameters (*random case*), namely the distribution coefficients are chosen randomly at each node A_i $i = 1, \ldots, 11$, when the simulation starts and then are kept constant.

Figure 2 presents the behaviour of $E(t)$. The optimal simulation is represented by a continuous curve, while random cases by dashed lines. As expected, random simulations of $E(t)$ are lower than the optimal behaviour. Precisely, when optimal parameters are used, nodes of 2×2 type have a congestions reduction due to the

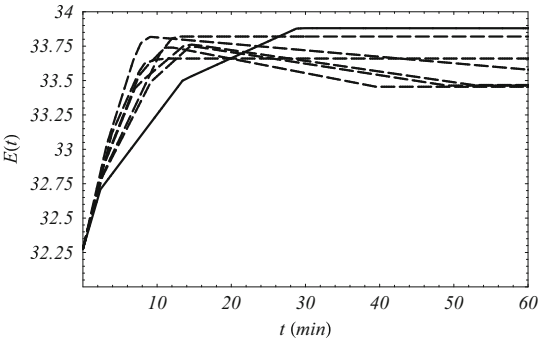


Fig. 2 Evolution of $E(t)$ in $[0, 60]$ using optimal distribution parameters (continuous line) and random coefficients (dashed lines)

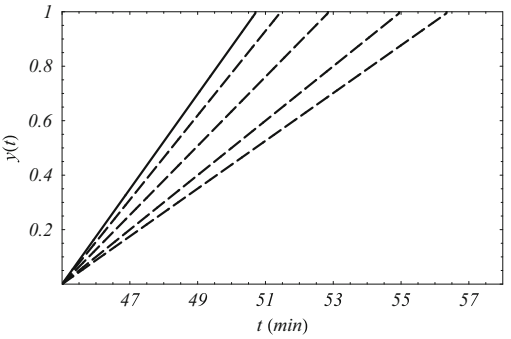


Fig. 3 Trajectory $y(t)$ for an emergency vehicle along road 17 with $t_0 = 45$; optimal coefficients (continuous line) and random choices (dashed lines)

redistribution of flows on roads. Even if right of way parameters of nodes B_i, $i = 1, \ldots, 6$, and distribution coefficients of nodes C_i, $i = 1, \ldots, 7$, are considered by the results of [2] and [3], traffic conditions are indeed almost unaffected.

Suppose that an emergency vehicle crosses a path in a network. Its position $y = y(t)$ is modelled by the Cauchy problem:

$$\begin{cases} \dot{y} = \zeta\left(D(t, x)\right), \\ y(t_0) = y_0, \end{cases} \tag{6}$$

where y_0 indicates the initial position at the initial time t_0. A numerical approach, described in [1], allows estimating the travelling time of the emergency vehicle. First, we compute the trajectory along road 17 and the time to cover it in optimal and random conditions. Then, we consider the path ρ and focus on the exit time versus the initial travel time t_0 (the time that the emergency vehicle needs to enter into the network).

In Fig. 3, we assume that the emergency vehicle starts its own travel at the beginning of road 17 at the initial time $t_0 = 45$ and compute the trajectories $y(t)$ in optimal (continuous line) and random cases (dashed lines).

The behaviour $y(t)$ in the optimal case has always a higher slope than the trajectories in random cases as traffic levels are low. When random parameters

are considered, shocks propagating backwards increase the density values on the network; The velocity for the emergency vehicles is reduced and exit times from road 17 become longer. Assuming $t_0 = 45$ we have the following time instants t_{out} in which the emergency vehicle goes out of road 17, either for the optimal distribution coefficients (t_0^{opt}) or random choices ($t_0^{r_i}, r_i, i = 1, .., 4$): $t_0^{opt} = 50.70$, $t_0^{r_1} = 56.34$, $t_0^{r_2} = 54.94$, $t_0^{r_3} = 52.84$ and $t_0^{r_4} = 51.42$.

5 Conclusions

This paper deals with an optimization study whose aim is to face emergencies for car traffic. Optimal distribution parameters at road nodes with two incoming and outgoing roads are obtained by maximizing a cost functional, which indicates the average velocity of emergency vehicles. Simulations on a real urban network prove the goodness of the optimization procedure as well as, in case of high congestions, the possibility of fast transits through the estimation of the trajectories of emergency vehicles. Future research activities aim to extend the proposed approach by either different types of cost functionals or other optimization procedures, based on genetic algorithms.

References

1. Bretti, G., Piccoli, B.: A tracking algorithm for car paths on road networks. SIAM J. Appl. Dyn. Sys. (SIADS) **7**, 510–531 (2008)
2. Cascone, A., D'Apice, C., Piccoli, B., Rarità, L.: Optimization of traffic on road networks. Math. Model. Methods Appl. Sci. **17**(10), 1587–1617 (2007)
3. Cascone, A., D'Apice, C., Piccoli, B., Rarità, L.: Circulation of car traffic in congested urban areas. Commun. Math. Sci. **6**(3), 765–784 (2008)
4. Coclite, G., Garavello, M., Piccoli, B.: Traffic flow on road networks. SIAM J. Math. Anal. **36**(6), 1862–1886 (2005)
5. D'Apice, C., Manzo, R., Rarità, L.: Splitting of traffic flows to control congestion in special events. Int. J. Math. Math. Sci. **2011**, 1–18 (2011). https://doi.org/10.1155/2011/563171
6. Lighthill, M.J., Whitham, G.B.: On kinetic waves. II. theory of traffic flows on long crowded roads. Proc. Roy. Soc. London Ser. A **229**, 317–345 (1955)
7. Manzo, R., Piccoli, B., Rarità, L.: Optimal distribution of traffic flows at junctions in emergency cases. Eur. J. Appl. Math. **23**(4), 515–535 (2012)
8. Richards, P.I.: Shock waves on the highway. Oper. Res. **4**, 42–51 (1956)
9. Tomasiello, S.: Numerical stability of DQ solutions of wave problems. Numer. Algorithms **57**(3), 289–312 (2011)
10. Tomasiello, S.: Some remarks on a new DQ-based method for solving a class of Volterra integro-differential equations. Appl. Math. Comput. **219**, 399–407 (2012)

The Cumulative Capacitated Vehicle Routing Problem with Profits Under Uncertainty

M. E. Bruni, S. Nucamendi-Guillén, S. Khodaparasti, and P. Beraldi

Abstract In this paper, we introduce the cumulative capacitated vehicle routing problem with profits and uncertain travel times. The aim is to visit a subset of customers maximizing the total collected revenue expressed as a decreasing function of the uncertain arrival times. The selective nature of the problem, the stochasticity of travel times, and the introduction of the capacity of vehicles make the problem quite challenging. We present a risk-averse approach leading to a non-linear mixed integer mathematical model. To solve the model, we develop a very fast and efficient metaheuristic designed to address the selective nature of the problem. The performance of the metaheuristic is shown by preliminary results obtained for two sets of benchmark instances.

Keywords Cumulative capacitated VRP · VRP with profits · Uncertainty · Metaheuristic

1 Introduction

The Cumulative Capacitated Vehicle Routing Problem (CCVRP, for short) is a variant of the classical capacitated vehicle routing problem in which the objective is to minimize the sum of arrival times at customers instead of the total route distance. For this problem, mathematical models [22], exact algorithms [14, 22] and heuristic and metaheuristic approaches [13, 17, 19–21] have been developed. Recently, Nucamendi-Guillén et al. [18] presented two new mathematical models for the

M. E. Bruni · S. Khodaparasti (✉) · P. Beraldi
Department of Mechanical, Energy and Management Engineering, University of Calabria, Rende, Cosenza, Italy
e-mail: mariaelena.bruni@unical.it; sara.khodaparasti@unical.it; patrizia.beraldi@unical.it

S. Nucamendi-Guillén
Facultad de Ingeniería, Universidad Panamericana, Zapopan, Jalisco, Mexico
e-mail: snucamendi@up.edu.mx

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_28

CCVRP in the deterministic context. The authors also proposed an iterated greedy algorithm outperforming other existing methods in the literature. The CCVRP can be also regarded as a generalization of the k-Traveling Repairmen Problem (k-TRP) [17] addressing customer-centric problems in which the need for fast, equitable and fair services, is crucial. A selective variant of the TRP, called TRP with Profits (TRPP), was introduced by Dewilde et al. [10]. Differently from the non selective case, two kinds of decisions have to be taken: which customers to serve and how to route them. The aim is to collect a revenue, defined as a function of the profit associated with each customer and of the customer's waiting time, defined as the elapsed time starting when the vehicle is at the depot, until the customer's service is completed.

In this paper, we present a mathematical model for the selective CCVRP with profits. The problem arises in situations in which, on the one hand, it is not possible to serve all the customers due to the resource and vehicle limitations, and on the other hand, the arrival time has a crucial role on the service performance. Examples of applications can be found in routing problems in relief efforts [7], perishable product delivery [5] or multi-robot routing [16]. We address the problem under uncertainty in travel times, via a mean-risk approach. This brings us with a non-linear mixed-integer formulation which is solved using a very fast and efficient metaheuristic approach. The uncertainty in vehicle routing problems has been widely studied in the literature [3, 8, 11, 12] but a few contributions are present for the single vehicle TRPP [4] and for the k-TRPP [1, 6]. To the best of our knowledge, the selective CCVRP, especially under the uncertainty of travel times, has never been addressed in the literature.

The paper is organized as follows. In Sect. 2, we introduce the CCVRP with profits and uncertain travel times and formulate it as a non-linear mixed-integer mathematical model. In Sect. 3, the metaheuristic solution approach is discussed and the proposed algorithm is presented. Section 4 is devoted to the computational experiments on two sets of benchmark instances. Finally, conclusions and remarks are discussed in Sect. 5.

2 Problem Formulation

Let consider an undirected graph $G = (V, E)$ where $V = \{0, 1, 2, \ldots, n\}$ corresponds to the node set and E denotes the edge set. Node 0 denotes the depot and $V' = \{1, 2, \ldots, n\}$ represents the set of customers. Each edge $(i, j) \in E$ has an associated random travel time $\tilde{t}_{ij}$, with a given mean μ_{ij} and variance σ^2_{ij}. To each customer $i \in V'$, a demand value d_i and a profit π_i are assigned. The profit represents the initial reward to visit the customer at time $t = 0$. Obviously, the demand and the profit of the depot are set to zero. A homogeneous fleet of k vehicles with a limited capacity Q ($\sum_{i \in V'} d_i \leq k \times Q$) are dispatched from the depot. The goal is to find a set of k disjoint tours visiting a subset of customers such that the

total collected revenue is maximized while the capacity of vehicles is not violated. The revenue collected from customer i is defined as the difference between its profit π_i and the customer's waiting time, defined as the elapsed time starting when the vehicle is at the depot, until the customer's service is completed. To account for the uncertainty of travel times, we adopt a mean-risk framework [2, 15] expressing the objective function as a linear combination of the expected revenue and its standard deviation, with weighting parameter $\lambda \in [0, 1]$. The variations in the trade-off parameter λ reflect the attitude of the decision maker with respect to risk. In order to formulate the problem, we present the multi-layer network proposed for the CCVRP in the deterministic context [18], and extend it to address the selective variant with random travel times. The network is briefly described as follows. Let L be the set of levels $L = \{1, \cdots, r, \cdots, N\}$, where $N = n - k + 1$, and each level includes a copy of all the customers amended also with the depot in levels from 2 to n. Each tour in the network is represented by a path that ends in the first level and starts in a copy of the depot in some level. In fact, the level number represents the position of the customer in the tour: the customer in the first level is the last in the tour, the customer in the second level is the last but one, and so on. Two distinct tours cannot visit the same customer, neither in the same level nor in different levels. The model variables are defined as follows. Let x_i^r be a binary variable that takes value 1 iff customer i is visited at level r (i.e. there are $r - 1$ customers to be visited after in the same tour); otherwise, it is set to 0. If $x_i^r = 1$, we say that customer i is active at level r. Let y_{ij}^r be another binary variable that is set to 1 iff edge (i, j) is used to link customer i active at level $r + 1$ with customer j active at level r; otherwise, it takes value 0. Finally, the variable v_{ij}^r denotes the sum of the demands of all the customers to be served after customer i in the same tour when $\sum_{r=1}^{N} y_{ij}^r = 1$ and it is equal to 0 otherwise.

The mathematical formulation is expressed as follows.

$$Max: z = \lambda\left(\sum_{j\in V'}\sum_{r=1}^{N}(\pi_j - r\mu_{0j})y_{0j}^r + \sum_{i\in V'}\sum_{\substack{j\in V'\\ j\neq i}}\sum_{r=1}^{N-1}(\pi_j - r\mu_{ij})y_{ij}^r\right) -$$

$$(1-\lambda)\sqrt{\sum_{j\in V'}\sum_{r=1}^{N} r^2\sigma_{0j}^2 y_{0j}^r + \sum_{i\in V'}\sum_{\substack{j\in V'\\ j\neq i}}\sum_{r=1}^{N-1} r^2\sigma_{ij}^2 y_{ij}^r} \tag{1}$$

$$\sum_{r=1}^{N} x_i^r \leq 1 \quad i \in V' \tag{2}$$

$$\sum_{i\in V'} x_i^1 = k \tag{3}$$

$$\sum_{r=1}^{N} \sum_{j \in V'} y_{0j}^{r} = k \tag{4}$$

$$y_{0i}^{N} = x_{i}^{N} \quad i \in V' \tag{5}$$

$$\sum_{\substack{j \in V' \\ j \neq i}} y_{ij}^{r} = x_{i}^{r+1} \quad i \in V',\ r = 1, 2, \ldots, N-1 \tag{6}$$

$$y_{0j}^{r} + \sum_{\substack{i \in V' \\ i \neq j}} y_{ij}^{r} = x_{j}^{r} \quad j \in V',\ r = 1, 2, \ldots, N-1 \tag{7}$$

$$v_{ij} \geq d_j \sum_{r=1}^{N} y_{ij}^{r} \quad i,\ j \in V', i \neq j \tag{8}$$

$$v_{0j} \leq Q \sum_{r=1}^{N} y_{0j}^{r} \quad j \in V' \tag{9}$$

$$v_{ij} \leq (Q - d_i) \sum_{r=1}^{N-1} y_{ij}^{r} \quad i,\ j \in V', i \neq j \tag{10}$$

$$v_{0i} + \sum_{\substack{j \in V' \\ j \neq i}} v_{ji} - \sum_{\substack{j \in V' \\ j \neq i}} v_{ij} = d_i \sum_{r=1}^{N} x_{i}^{r} \quad i \in V' \tag{11}$$

$$x_{i}^{r} \in \{0, 1\} \quad i \in V',\ r = 1, 2, \ldots, N \tag{12}$$

$$y_{0j}^{r} \in \{0, 1\} \quad j \in V',\ r = 1, 2, \ldots, N \tag{13}$$

$$y_{ij}^{r} \in \{0, 1\} \quad i, j \in V',\ i \neq j,\ r = 1, 2, \ldots, N-1 \tag{14}$$

$$v_{ij} \geq 0 \quad i \in V' \cup \{0\},\ j \in V', i \neq j \tag{15}$$

The objective function (1) maximizes the total stochastic revenue. In order to deal with this more involved objective function, the first term accounts for the expected total revenue, expressed as the sum of the profits collected at nodes minus the expected arrival time at those nodes, whereas the second one for the standard deviation of the total arrival time. Both the terms can be derived by applying the standard formula of the expected value and variance of the sum of independent random variables. The factor λ is used to weight the importance attributed by the decision maker to the two terms: the lower is λ, the greater is the importance attributed to the risk. Constraints (2) ensure that customer i is served at most once. Constraints (3) and (4) ensure that only k starting and ending edges are created, whereas constraints (5)–(7) satisfy connectivity requirements. Constraints (8)–(10) establish the minimum and maximum values for v_{ij}^{k}. In particular, Eqs. (9) and (10)

force the same variables to be 0 when y^r_{0j} and y^r_{ij} are 0, respectively. These constraints in conjunction with (11) estimate the load of each vehicle and act as the sub-tour elimination constraints. Finally, constraints (12)–(15) show the nature of variables.

3 Metaheuristic Procedure

In contrast to the classical CCVRP, the number of visited customers in our problem is not known in advance and, more importantly, the objective function does not show a monotonic behaviour with respect to the number of visited customers. Besides the selective nature of the problem, the introduction of the non-linear term in the objective function (1) makes it quite challenging. To deal with this computational complexity, we propose a very fast metaheuristic approach composed of three main procedures, including a procedure for building an initial solution, an improvement phase and a destructive part. The pseudo code of the proposed approach is shown in Algorithm 1. Here U denotes the set of unrouted customers. The first solution is generated in lines (2)–(10). A *Constructive* procedure, adapted from the iterated greedy approach presented in [18], is called. If not all the nodes are served, γ nodes are randomly deleted from the current solution $\mathbf{s}$ and the procedure is repeated. Since the elimination mechanism is completely random, in a finite number of iterations, a complete solution $\mathbf{s}$ including all customers is built. The *Constructive* procedure is described in Algorithm 2. First, a partial solution $\mathbf{s}$ is built assigning the k best customers with the highest revenue to the k vehicles. Then, the solution is extended to include other customers, respecting the capacity restrictions. Each feasible customer insertion is evaluated using a regret criterion. Denoting with RQ a k-dimensional vector representing the remaining capacity of vehicles, the insertion in route r is feasible only if $RQ_r \geq d_i$. For any customer $i \in U$ the best feasible position in each vehicle route r is evaluated. Among these values the best absolute value is chosen. Then, the differences between that values and the best one are evaluated. These differences are then summed up to obtain, for each customer, the regret value. The customer with the highest regret is selected to be included in the solution and inserted in its best position and tour. If the insertion is not feasible in any route (the demand of the candidate customer is higher than the remaining capacity of all vehicles ($RQ_r < d_i$, $\forall r = 1, \cdots, k$), the current solution is not extended anymore and the procedure quits.

The *Improvement* procedure is a local search mechanism which is appropriately customized to cope with the selective nature of the problem. The local search mechanism includes five different neighborhood structures which are arranged into two classes of intra- and inter-route neighborhoods.

The intra-route neighborhoods used are the *swap*, the *reallocation*, and the *2-opt* move operators. The inter-route neighborhoods are the *exchange* and the *relocation* operators which are implemented on a pair of tours.

Algorithm 1: The proposed metaheuristic

Initialization: $U \leftarrow V'$, $\mathbf{s}, \mathbf{s}', \mathbf{s}''$, $\mathbf{s_{best}} \leftarrow null$, $z(\mathbf{s_{best}}) \leftarrow -\infty$, $iter \leftarrow 0$
while $(U \neq \emptyset)$ **do**
  $\mathbf{s}' \leftarrow$ *Constructive procedure*$(\mathbf{s}, U)$
  if $(U == \emptyset)$ **then**
    break
  end
  else
    $\gamma \leftarrow rand(1, |V'| - |U|)$
    Eliminate from $\mathbf{s}'$ γ nodes randomly selected
  end
end
$\mathbf{s} \leftarrow$ *Improvement procedure*$(\mathbf{s}')$
if $(z(\mathbf{s_{best}}) < z(\mathbf{s}))$ **then**
  $\mathbf{s_{best}} \leftarrow \mathbf{s}$
end
while $(iter < \lfloor 25\%|V'| \rfloor)$ **do**
  $\mathbf{s}' \leftarrow$ *Destructive procedure*$(\mathbf{s}, 1)$
  $\mathbf{s}'' \leftarrow$ *Improvement procedure*$^*(\mathbf{s}')$
  if $(z(\mathbf{s_{best}}) < z(\mathbf{s}''))$ **then**
    $\mathbf{s_{best}} \leftarrow \mathbf{s}''$
  end
  $\mathbf{s} \leftarrow \mathbf{s}''$
  $iter \leftarrow iter + 1$
end
return $\mathbf{s_{best}}$

Algorithm 2: The constructive procedure

$Continue := 1$
while $(U \neq \emptyset \quad \& \quad Continue)$ **do**
  if (*there are empty routes in* $\mathbf{s}$) **then**
    Initialize them with customers $i \in U$ that have the highest values of
    $\lambda(\pi_i - \mu_{0i}) - (1 - \lambda)\sqrt{\sigma_{0i}^2}$
    Update U and RQ
  end
  foreach *customer in* U **do**
    Determine the best feasible insertion points over all partial tours
    Compute the regret value
  end
  if *at least one feasible insertion is possible* **then**
    Insert the customer with the highest regret in its best position and route in $\mathbf{s}$
    Update U and RQ
  else
    $Continue := 0$
  end
end
return $\mathbf{s}$

To be more precise, the *swap* operator exchanges the position of two routed customers i and j; the *reallocation* deletes a customer from its current position and reinserts it into another position on the same route; in the *2-opt* move, two non-adjacent edges $(i, i+1)$ and $(j, j+1)$ in the nominal tour $0, 1, 2, \cdots, i, i+1, \cdots, j, j+1, \cdots$ are deleted and replaced by (i, j) and $(i+1, j+1)$, resulting in the new tour $1, 2, \cdots, i, j, \cdots, i+1, j+1, \cdots$.

The *exchange* operator exchanges two customers belonging to different tours, if possible with respect to the remaining vehicle capacities, and the *relocation* deletes one customer from its current position and its tour and inserts it into another position on a different tour with enough remaining capacity. The *exchange* and *relocation* moves are followed by the update of RQ vector. Each intra- or inter-route neighborhood is explored based on a first improvement strategy until there is an improvement. The intra- and inter-route procedures are executed separately one after the other until the input solution is improved. The improvement procedure (marked by an $*$ in line 18) performs the same local search as described above augmented with an extra move operator (*replace*) that substitutes a routed customer by an unrouted one. After each *replace* move, U and RQ are appropriately updated. As before, the first improvement strategy is taken into account and the neighborhood is explored until there is any improving solution.

The *Destructive* procedure implements a simple but effective idea to find the right number of visited customers. In fact, whenever the destructive procedure is executed, the number of visited customers is decreased by one randomly deleting one node. After each customer deletion, the solution is repaired linking the disconnected endpoints and the set of unrouted customers as well as the remaining capacities are updated. We quit the iterative process when the cardinality of the unrouted customers is below a given threshold, let say $\lfloor 25\% \times n \rfloor$.

4 Computational Results

In this section, we report the computational experiments carried out with the aim of assessing the efficiency of the proposed approach. The metaheuristic was coded in C++ and the mathematical model was solved using the open source SCIP library, release 3.2.0. All the experiments have been performed on an Intel® Core™ i7 2.90 GHz, with 8.0 GB of RAM memory. The performance of our algorithm is evaluated with respect to the best feasible solution reported by SCIP within a time limit of 3600 s, if any; otherwise, it was compared with the *Final Dual Bound* reported by SCIP. To be precise, the heuristic Gap is calculated as $Gap = \frac{OF_{SCIP} - OF_{Heu}}{OF_{SCIP}} \times 100$ where OF_{SCIP} is the best objective value reported by SCIP or its *Final dual Bound* if no feasible solution was found within the time limit (marked in bold). All the Gap values are reported in percentage and the CPU values are measured in seconds. To account for the randomness of the algorithm, we ran each instance 10 times with different seed values and the average over all iterations is reported. As a test bed, we used the set of P- and E-instances [9].

The instance name shows the number of nodes n and the number of vehicle k. For example, the P-instance Pn40k5 has $n = 40$ and $k = 5$. The expected travel time μ_{ij} of the edge (i, j) is set as the rounded Euclidean distance between customers i and j and the travel time variance σ_{ij}^2 is set as $\lceil \mu_{ij} \times \omega \rceil$ where $\omega \sim U[0.1, 0.32)$. The profit values are proportional to the expected distance and its variance. $\pi_i \sim U\left(\min_{i \in V'}\left(\mu_{0i} - \beta\sqrt{\sigma_{0i}^2}\right), \frac{n}{2}\max_{i \in V'}\left(\mu_{0i} + \beta\sqrt{\sigma_{0i}^2}\right)\right], \ \beta \sim U\,(0, 1]$

Tables 1 and 2 show the results for the set of the P- and the E-instances, respectively for different values of $\lambda \in \{0.1, 0.5, 0.9\}$. Column 1 shows the instance name. The results for $\lambda = 0.1$ are summarized in columns 2–6; the columns 2–3 report the average relative gap (Gap) and the computational time (CPU) over 10 different runs. Column 4 reports the best gap over all the runs. Columns 5–6 report the computational time and the SCIP optimality gap. In a similar way, the results for $\lambda = 0.5$ and $\lambda = 0.9$ are arranged.

As the results show, the proposed metaheuristic provides quite promising performance in terms of computational time, with an average solution time of 0.25 s. In addition, the average Gap reported in Columns 2, 7 and 12 varies from 2.73 to -42.92 and -61.18. It is important to note that in 9 out of 16 instances for the case with $\lambda = 0.1$ (4 and 3 instances for $\lambda = 0.5$ and $\lambda = 0.9$, respectively) SCIP was not able to find any feasible solution (in this case the SCIP optimality gaps is set to ∞). The average (maximum) Gap for the instances in which a SCIP feasible solution is available, is limited to -0.53 (1.06), -57.94 (2.19) and -75.65 (2.05). These negative values confirm that, on average, the proposed heuristic outperforms SCIP in terms of the solution quality.

Similar conclusions can be drawn for the set of the E-instances in Table 2. The average CPU time for all the instances and λ values is below 0.28 s; the average Gap in Columns 2, 7 and 12 varies from 1.83 to -27.37 and -27.26. If we consider only the cases for which SCIP is able to provide a feasible solution, the average Gap decreases to 0.34, -39.81 and -39.56, respectively for different λ values.

Table 1 Computational results for P-instances

	$\lambda = 0.1$					$\lambda = 0.5$					$\lambda = 0.9$				
	Avg.		Best	SCIP		Avg.		*Best*	SCIP		Avg.		Best	SCIP	
Instance	*Gap*	*CPU*	*Gap*	*CPU*	*Gap*	*Gap*	*CPU*	*Gap*	*CPU*	*Gap*	*Gap*	*CPU*	*Gap*	*CPU*	*Gap*
Pn40k5	1.06	0.02	0.78	3600	8.44	1.5	0.03	1.29	1912	0	2.05	0.02	2.02	258.81	0
Pn45k5	0.54	0.04	−0.84	3600	7.37	2.19	0.03	2.03	3600	0.65	1.86	0.04	0.92	833.77	0
Pn50k7	−0.45	0.07	−0.91	3600	7.28	1.85	0.06	1.61	3600	0.74	0.84	0.06	0.84	3600	0.28
Pn50k8	**7.23**	0.05	**7.22**	3600	∞	**2.39**	0.05	**2.39**	3600	∞	-0.61	0.06	-0.61	3600	2.62
Pn50k10	**5.5**	0.05	**5.5**	3600	∞	1.07	0.04	1.07	3600	0.75	0.44	0.04	0.44	3600	0.29
Pn51k10	1.03	0.06	0.29	3600	3.33	0.91	0.06	0.91	3600	0.33	1.25	0.06	1.25	1412.5	0
Pn55k7	−1.18	0.09	−1.62	3600	8.25	0.86	0.09	0.53	3600	0.86	1.61	0.1	1.57	3600	0.29
Pn55k8	−4.76	0.1	−4.76	3600	11.68	0.96	0.1	0.92	3600	0.53	1.11	0.1	1.11	1875	0
Pn55k10	0.03	0.08	−0.04	3600	4.89	0.88	0.07	0.88	3600	0.46	1.32	0.07	1.32	1953.2	0
Pn55k15	**4.45**	0.05	**4.1**	3600	∞	**1.71**	0.04	**1.07**	3600	∞	**1.62**	0.04	**1.62**	3600	∞
Pn60k10	**5.08**	0.09	**5.08**	3600	∞	0.87	0.1	0.87	3600	0.72	−0.26	0.11	−0.26	3600	1.16
Pn60k15	**3.68**	0.05	**3.68**	3600	∞	**1.21**	0.07	**1.21**	3600	∞	**0.88**	0.06	**0.88**	3600	∞
Pn65k10	**3.53**	0.13	**3.5**	3600	∞	−25.69	0.11	−25.69	3600	27.37	−22.34	0.13	−22.34	3600	23.95
Pn70k10	**3.24**	0.15	**3.45**	3600	∞	−12.58	0.15	−12.58	3600	14.25	−299	0.17	−299	3600	303.13
Pn76k4	**7.59**	0.19	**7.13**	3600	∞	−668.2	0.19	−669.8	3600	697.41	−671.8	0.21	−675.9	3600	693.56
Pn76k5	**7.11**	0.23	**5.83**	3600	∞	**3.38**	0.25	**3.22**	3600	∞	**2.16**	0.22	**2.07**	3600	∞
Avg.	2.73	0.09	2.4	3600	7.32	−42.92	0.09	−43.13	3494.9	62.01	−61.18	0.09	−61.5	2870.83	78.87
Max.	7.59	0.23	7.22	3600	∞	3.38	0.25	3.22	3600	∞	2.16	0.22	2.07	3600	∞

Table 2 Computational results for E-instances

	$\lambda = 0.1$					$\lambda = 0.5$					$\lambda = 0.9$				
	Avg.		Best	SCIP		Avg.		Best	SCIP		Avg.		Best	SCIP	
Instance	*Gap*	*CPU*	Gap	*CPU*	*Gap*	*Gap*	*CPU*	*Gap*	CPU	*Gap*	*Gap*	*CPU*	*Gap*	*CPU*	*Gap*
En22k4	2.02	0	2.02	93.11	0	2.18	0.01	2.18	131	0	1.57	0	1.57	56.39	0
En23k3	4.85	0	2.81	306.41	0	4.36	0.01	3.38	306	0	6.33	0.01	2.68	305.58	0
En30k4	5.03	0.01	5.03	3600	8.64	2	0.02	2	2815	0	2.04	0.01	2.04	685.38	0
En30k3	8.85	0.01	−0.64	3600	16.35	2.56	0.01	0.94	3080	0	3.03	0.01	1.82	394.79	0
En33k4	−8.98	0.01	−9.36	3600	25.19	1.47	0.02	1.33	3600	1.68	1.01	0.01	1.01	3039.91	0
En51k5	−9.76	0.07	−10.47	3600	19.1	2.62	0.05	2.5	3600	0.96	1.25	0.05	0.22	3600	1.2
En76k7	**4.45**	0.26	**4.08**	3600	∞	**1.95**	0.25	**1.75**	3600	∞	**1.51**	0.28	**1.49**	3600	∞
En76k8	**4.54**	0.25	**4.19**	3600	∞	**1.65**	0.23	**1.11**	3600	∞	**2.02**	0.26	**1.69**	3600	∞
En76k10	**4.26**	0.23	**4.24**	3600	∞	−293.9	0.26	−293.9	3600	300	−292.18	0.24	−292.43	3600	297.6
En76k14	**3.05**	0.19	**3.05**	3600	∞	**1.39**	0.19	**1.39**	3600	∞	**0.86**	0.14	**0.86**	3600	∞
Avg.	1.83	0.1	0.5	2920.25	11.55	−27.37	0.11	−27.73	2793	43.2	−27.26	0.1	−27.91	2249.2	42.69
Max.	8.85	0.26	5.03	3600	∞	4.36	0.26	3.38	3600	∞	6.33	0.28	2.68	3600	∞

5 Conclusions

In this paper, we have introduced the CCVRP with profits and uncertain travel times. We presented a risk-averse mathematical formulation for the problem and proposed an efficient and fast metaheuristic that takes the selective nature of the problem into account. The good performance of the proposed solution algorithm is shown by the computational experiments carried out on the set of the P- and the E-instances. Future work can be focused on the development of a multi-objective approach for the considered problem and on the design of a tailored matheuristic approach.

References

1. Beraldi, P., Bruni, M.E., Khodaparasti, S.: A hybrid reactive GRASP heuristic for the risk-averse k-traveling repairman problem with profits. Comput. Oper. Res. **115**, 104854 (2020)
2. Beraldi, P., Bruni, M.E., Violi, A.: Capital rationing problems under uncertainty and risk. Comput. Optim. Appl. **51**(3), 1375–1396 (2012)
3. Beraldi, P., Bruni, M.E., Laganà, D., Musmanno, R.: The mixed capacitated general routing problem under uncertainty. Eur. J. Oper. Res. **240**(2), 382–392 (2015)
4. Beraldi, P., Bruni, M.E., Laganà, D., Musmanno, R.: The risk-averse traveling repairman problem with profits. Soft. Comput. **23**(9), 2979–2993 (2019)
5. Bruni, M., Forte, M., Scarlato, A., Beraldi, P.: The traveling repairman problem app for mobile phones: a case on perishable product delivery. AIRO Springer Series Volume "Advances in Optimization and Decision Science for Society, Services and Enterprises" Proceedings of ODS 2019
6. Bruni, M., Beraldi, P., Khodaparasti, S.: A heuristic approach for the k-traveling repairman problem with profits under uncertainty. Electron Notes Discrete Math. **69**, 221–228 (2018)
7. Bruni, M.E., Beraldi, P., Khodaparasti, S.: A fast heuristic for routing in post-disaster humanitarian relief logistics. Transp. Res. Procedia **30**, 304–313 (2018)
8. Bruni, M.E., Brusco, L., Ielpa, G., Beraldi, P.: The risk-averse profitable tour problem. In: ICORES 2019–Proceedings of the 8th International Conference on Operations Research and Enterprise Systems, pp. 459–466 (2019)
9. Christofides, N., Eilon, S.: An algorithm for the vehicle-dispatching problem. J. Oper. Res. Soc. **20**(3), 309–318 (1969)
10. Dewilde, T., Cattrysse, D., Coene, S., Spieksma, F.C., Vansteenwegen, P.: Heuristics for the traveling repairman problem with profits. Comput. Oper. Res. **40**(7), 1700–1707 (2013)
11. Eksioglu, B., Vural, A.V., Reisman, A.: The vehicle routing problem: a taxonomic review. Comput. Ind. Eng. **57**(4), 1472–1483 (2009)
12. Gaur, D.R., Mudgal, A., Singh, R.R.: Improved approximation algorithms for cumulative VRP with stochastic demands. Discret. Appl. Math. (2018, in press)
13. Ke, L., Feng, Z.: A two-phase metaheuristic for the cumulative capacitated vehicle routing problem. Comput. Oper. Res. **40**(2), 633–638 (2013)
14. Lysgaard, J., Wøhlk, S.: A branch-and-cut-and-price algorithm for the cumulative capacitated vehicle routing problem. Eur. J. Oper. Res. **236**(3), 800–810 (2014)
15. Markowitz, H.: Portfolio selection. J. Financ. **7**(1), 77–91 (1952)
16. Melvin, J., Keskinocak, P., Koenig, S., Tovey, C., Ozkaya, B.Y.: Multi-robot routing with rewards and disjoint time windows. In: Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2332–2337 (2007)

17. Ngueveu, S.U., Prins, C., Calvo, R.W.: An effective memetic algorithm for the cumulative capacitated vehicle routing problem. Comput. Oper. Res. **37**(11), 1877–1885 (2010)
18. Nucamendi-Guillén, S., Angel-Bello, F., Martínez-Salazar, I., Cordero-Franco, A.E.: The cumulative capacitated vehicle routing problem: new formulations and iterated greedy algorithms. Expert Sys. Appl. **113**, 315–327 (2018)
19. Ozsoydan, F.B., Sipahioglu, A.: Heuristic solution approaches for the cumulative capacitated vehicle routing problem. Optimization **62**(10), 1321–1340 (2013)
20. Ribeiro, G.M., Laporte, G.: An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. Comput. Oper. Res. **39**(3), 728–735 (2012)
21. Rivera, J.C., Afsar, H.M., Prins, C.: A multistart iterated local search for the multitrip cumulative capacitated vehicle routing problem. Comput. Optim. Appl. **61**(1), 159–187 (2015)
22. Rivera, J.C., Afsar, H.M., Prins, C.: Mathematical formulations and exact algorithm for the multitrip cumulative capacitated single-vehicle routing problem. Eur. J. Oper. Res. **249**(1), 93–104 (2016)

Dealing with the Stochastic Home Energy Management Problem

Patrizia Beraldi, Antonio Violi, Maria Elena Bruni, and Gianluca Carrozzino

Abstract This paper focuses on the home energy management problem faced by a smart prosumer equipped with photovoltaic panels and a storage system. Some of the home appliances (the shiftable ones) can be controlled in that the consumer may specify an operating time window within the load should be turned on. The inherent uncertainty affecting the main model parameters (i.e. loads and production from renewable) is explicitly accounted for by adopting the two-stage stochastic programming modeling paradigm. The solution provides the prosumer with the optimal scheduling of the shiftable loads and the using profile of the storage system that guarantee the minimum expected energy procurement cost, taking into account the prosumer's comfort. Preliminary results, collected on three different categories of residential prosumers, have shown the effectiveness of the proposed approach in terms of cost saving and the advantage related to the use of a stochastic programming approach over a deterministic formulation.

Keywords Home energy management · Stochastic programming · Optimal scheduling

P. Beraldi (✉) · M. E. Bruni · G. Carrozzino
Department of Mechanical, Energy and Management Engineering, University of Calabria, Rende CS, Italy
e-mail: patrizia.beraldi@unical.it; mariaelena.bruni@unical.it; gianluca.carrozzino@unical.it

A. Violi
Department of Law, Economics, Management and Quantitative Methods, University of Sannio, Benevento BN, Italy
e-mail: antonio.violi@unisannio.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_29

1 Introduction

In the last decades, energy industries all around the world have been experiencing rapid and deep changes. One of the main driver is represented by the fast development and uptake of the renewable technologies facilitated by the ICT evolution, as well as the growing government initiatives to promote the deployment of clean energy sources. Renewable production systems, such as photovoltaic panels (PV) and wind turbines, are typically distributed in the territory and their integration into the current energy system increases the complexity level and requires the joint effort by all the stakeholders of the energy supply chain. The decentralized configuration also brings new players into the electricity industry leading to a reconfiguration of the role and functions of the main participants and, in particular, of the end-users. They are no longer passive consumers of electricity services, but are increasingly involved as producers. They are now termed as "prosumers" to emphasize their new nature, as consumers who generate renewable energy. If aggregated, prosumers may have even a large potential by increasing the reliability of electrical supply. Different forms of aggregation are emerging in recent years, from virtual power plants to micro grids and integrated community energy systems.

The optimal management and operation of these emerging forms of aggregation poses new challenging optimization problems. Among the others, we mention the management of the shared resources [4], the definition of the optimal tariffs for the coalition members [12], the interaction between the aggregation and the power grid [9]. In all the mentioned problems, the prosumers play a crucial role and the optimal management of their resources represents a critical issue also for the energy coalition they might belong to. This paper focuses on the home energy management problem faced by a prosumer that we assume to be able to exchange the energy locally produced with the distribution grid. We assume that the prosumer owns smart devices able to control electrical appliances whose use can be shifted within predefined time windows. The problem consists in defining the optimal management of the resources and load scheduling that minimizes the total electricity procurement cost, taking into account the prosumer's comfort and the load priority.

Because of its practical relevance, the home energy management problem has been widely studied by the scientific community. However, as highlighted by Benetti et al. in [2] much more effort is needed in the definition of optimization models including real features in order to enhance the accuracy of the provided solutions. The vast majority of the literature proposes deterministic formulations that differ for the real features that are mathematically represented. Most of the papers consider two type of loads: non-controllable and controllable. While the former must be activated at fixed hours of the day (e.g. the refrigerator), the latter (e.g. washing machines, dryers) may operate at any time within a time interval specified by the end-user. Thus, depending on the hourly tariffs, it may result convenient to shift the use of some appliances. Among the different scientific contributions, we cite the paper by Martinez-Pabon et al. who propose in [10] a model for determining the optimal scheduling of the appliances so to minimize the total energy cost. Yahia and

Pradhan extend in [13] the model by incorporating the consumer's preference by a bi-objective function where the first term accounts for the energy cost, whereas the second one for the "inconvenience" measured in terms of disparity between the preferred and the optimal schedule. Some other authors consider the inconvenience issue by introducing in the formulation a constraint on the consumer's preferences [7]. While in the referred papers no local energy sources are assumed to be available in the prosumer's home, some other papers integrate the scheduling with the optimal management of the local resources. Among the contributions dealing with this more involved configuration, we mention the recent paper [1] where the authors propose a mixed integer problem that also accounts for the management of the thermal equipment.

Few papers acknowledge the importance of explicitly accounting for the inherent uncertainty affecting the main problem parameters. We cite the contribution by Chen et al. who propose in [6] a stochastic scheduling technique which involves an energy adaptation variable β to model the stochastic consumption patterns of the various household appliances. Correa-Florez et al. propose in [8] a stochastic programming model for the optimal management of the prosumer's resources without accounting for the scheduling of the controllable loads. Our paper contributes to the literature on the application of stochastic optimization approaches for the home energy management problem by proposing a two-stage stochastic programming formulation that integrates the optimal management of the prosumer's resources with the scheduling of the controllable loads. A preliminary formulation of the problem based on the same modeling framework has been proposed in [5]. It, however, does not consider the decisions related to the management of the storage system as second-stage variables. Moreover, it does not include the regret constraint aimed at limiting to a given threshold the inconvenience related to the shifting of the flexible loads from the preferred starting time. The approach proposed in this paper has been tested on several test cases that represent different prosumer configurations.

The rest of the paper is organized as follows. Section 2 introduces the problem and the stochastic formulation. Section 3 is devoted to the presentation of the numerical results carried out considering a real case study. Concluding remarks and future research developments are discussed in Sect. 4.

2 Problem Definition and Mathematical Formulation

We consider a prosumer's home connected with the power grid and equipped with a smart controller able to manage the electrical appliances eventually postponing their use in more convenient time slots. We assume that the smart home hosts PV panels and storage devices allowing the prosumer to satisfy the demand (at least partially) by the local production and eventually selling the amount of electricity in excess. Figure 1 shows an overview of the prosumer's system we are dealing with. The problem under investigation consists in defining the optimal management of the available resources and the scheduling of the loads so to

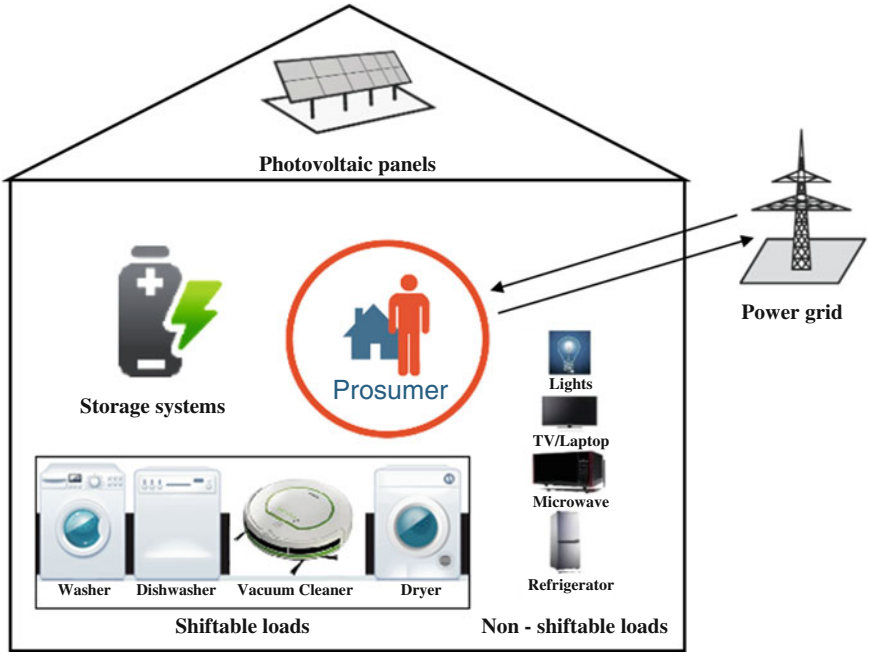


Fig. 1 Overview of the prosumer's home resources

minimize the total electricity procurement cost. The intermittent renewable power generation and the variable loads makes the problem more challenging and calls for the adoption of mathematical frameworks able to explicitly address the inherent uncertainty affecting the model parameters. In this paper, we adopt the stochastic programming framework and, in particular, the recourse paradigm. We assume that uncertain parameters are represented by random variables defined on a given discrete probability space $(\Omega, \mathscr{F}, \mathbb{P})$. We denote by S the number of realization, indexed by s, each occurring with a probability π^s.

We consider a time horizon defined by T hourly time steps ($t = 1, \ldots, T$) and we assume that the hours of the day are divided in three time-of-use blocks: peak, intermediate and off-peak (according to the Italian configuration). The electricity price varies according to the block the specific hour belongs to. We denote by P_t the unitary purchasing price and by W_t the selling price, that we assume to be known in advance. The prosumer loads are classified in controllable and non controllable. The controllable ones (e.g. washing machine, dryers), referred by the set $K = \{1, \ldots, N\}$, can be shifted, provided that they are run within a given time window $[l_k, u_k]$. For each load $k \in K$, let d_k denote the hourly energy consumption and n_k the number of working hours once activated. We also consider a given unit regret rate r_k associated with each hour shifting from the preferred starting time st_k and a maximum cumulative dissatisfaction value V. Some appliance operations are related by precedence relations. For example, the operation of a clothes dryer

follows the operation of a washing machine. To mathematically represent these relations, we introduce a binary parameter f_{kj} taking the value 1 if load k cannot start before load j, and 0 otherwise. Moreover, we denote with g_{kj} the minimum number of hours of delay (if any) between k and j. Differently from the controllable loads, the non controllable ones can not be postponed (e.g. refrigerator, lights) and might be uncertain. For each hour t, we denote by D_t^s the cumulative demand related to uncontrollable loads under scenario s.

The overall hourly demand can be satisfied by purchasing electricity from the market and/or by using the available local resources. We denote by R_t^s the amount of energy produced by the PV panels at time t under scenario s. To bridge the timing gap between power generation from renewable sources and consumption, the prosumer may use the storage system that could also accumulate energy bought from market during off-peak hours. We denote by C^{Max} the storage capacity and by η_{in} and η_{out} the efficiency rate for energy injection and withdrawal from the storage system. Moreover, we indicate by φ^{LB} and φ^{UB} the operative range in terms of minimum and maximum percentage of nominal capacity, and by χ^{LB} and χ^{UB} the minimum ramp-up and ramp-down rate for the energy level from one hour to the next one.

The main decisions that the prosumer is called to take are related to the optimal scheduling of the controllable loads by managing the available resources and eventually purchasing energy from the market. While some decisions should be taken in advance without knowing the realization of the random parameters, other decisions can be postponed and used as corrective actions, if necessary. In the proposed formulation, first-stage decisions refer to the scheduling operation. In particular, we denote by δ_{kt}, the binary variable taking value 1 if the controllable load k starts at time t and 0 otherwise. Once uncertainty realizes, corrective actions referred to the management of the storage system and market operations are taken, in order to guarantee the satisfaction of the stochastic loads. In particular, for each scenario s and time t, we indicate by SL_t^s energy level of the storage system and by SIN_t^s and $SOUT_t^s$ the amount to supply into and supplied from the system. Moreover, x_t^s and y_t^s represent the amount of energy to buy and sell in hour t under scenario s, respectively. The proposed mathematical formulation is the following:

$$\min \sum_{s=1}^{S} \pi^s \sum_{t=1}^{T} (P_t x_t^s - W_t y_t^s) \tag{1}$$

$s.t.$

$$x_t^s + SOUT_t^s - SIN_t^s - y_t^s = D_t^s + \sum_{k=1}^{N} d_k \sum_{h=t-n_k}^{t} \delta_{kh} - R_t^s \qquad \forall t, \forall s \tag{2}$$

$$\sum_{t=l_k}^{u_k} \delta_{kt} = 1 \qquad \forall k \tag{3}$$

$$\delta_{kt} \leq \sum_{h=l_k}^{t-g_{kj}} \delta_{jh} \qquad \forall t \in \{l_k, u_k\}, \forall (k,j) \in K | f_{kj} = 1 \quad (4)$$

$$\sum_{k=1}^{N} r_k | st_k - \sum_{t=1}^{T} t\delta_{kt} | \leq V \quad (5)$$

$$SL_t^s = SL_{t-1}^s + \eta^{in} SIN_t^s - \frac{SOUT_t^s}{\eta^{out}} \qquad \forall t, \forall s \quad (6)$$

$$\varphi^{LB} C^{Max} \leq SL_t^s \leq \varphi^{UB} C^{Max} \qquad \forall t, \forall s \quad (7)$$

$$\chi^{LB} C^{Max} \leq SL_t^s - SL_{t-1}^s \leq \chi^{UB} C^{Max} \qquad \forall t, \forall s \quad (8)$$

$$x_t^s \leq E^{Max} \qquad \forall t, \forall s \quad (9)$$

$$x_t^s, y_t^s \geq 0 \qquad \forall t, \forall s \quad (10)$$

$$SL_t^s, SIN_t^s, SOUT_t^s \geq 0 \qquad \forall t, \forall s \quad (11)$$

$$\delta_{kt} \in \{0, 1\} \qquad \forall k, \forall t \quad (12)$$

The objective function (1) aims at minimizing the expected value of the difference between total cost of energy purchased and the revenue for energy selling. Constraints (2) represents the energy balance for each hour t of the time horizon and every scenario s, while condition (3) imposes that each controllable load k must be activated within its time window. Constraints (4) model the precedence relation and the eventual delay between two shiftable loads. Condition (5) limits to the value V the overall regret due to the shifting from the preferred starting time of the controllable loads shifts. Constraints (6)–(8) model the technological constraints of the storage system. In particular, (6) states the energy level balance from one hour to the next one, constraints (7) bound the energy level within the expected operative range and conditions (8) limit the change in the energy level in each hour. The energy amount that can be absorbed from the grid is limited to E^{Max} by (9). Finally, (10)–(12) define the nature of the decision variables.

We note that regret constraint (5) is non linear. A linearized formulation can be easily derived by including additional nonnegative variables ϵ_k^+ and ϵ_k^- and the following conditions:

$$\sum_{k=1}^{N} r_k (\epsilon_k^+ + \epsilon_k^-) \leq V \quad (13)$$

$$\epsilon_k^+ \geq st_k - \sum_{t=1}^{T} t\delta_{kt} \qquad \forall k \quad (14)$$

$$\epsilon_k^- \geq \sum_{t=1}^{T} t\delta_{kt} - st_k \qquad \forall k \quad (15)$$

The overall problem belongs to class of mixed integer linear problems and depending on the number of considered scenarios the solution process can be computationally demanding. However, the test cases considered hereafter, the use of off-of-the-shelf software is still possible.

3 Computational Experiments

In this section, we describe the computational experience carried out in order to validate the effectiveness of the proposed approach. The numerical code integrates MATLAB R2015[1] for the scenario generation and parameters set-up phases and GAMS 24.5[2] as algebraic modeling system, with CPLEX 12.6.1[3] as solver for mixed integer problems. All the test cases have been solved on a PC Intel Core I7 (2.5 GHz) with 8 GB of RAM.

We have considered three different prosumer configurations, representing a family with one, three or five components, that we refer to as small, medium and large prosumer. The expected values of the overall daily demand are 6.43 kWh, 8.67 kWh and 10.91 kWh, respectively. We assume that each type of prosumer is equipped with building integrated PV panels with a nominal power of 3 kWh, whose average daily production level depends on the season: 5 kWh for Winter, 9.86 kWh for Spring/Autumn and 13.7 kWh for Summer. We also assume that the energy that can be absorbed from the grid in 1 h (E^{Max}) cannot exceed 4.5 kWh. As regards the controllable loads we have considered four devices, whose characteristics are reported in Table 1.

We also assume the presence of one precedence constraint related to the tumble dryer start-up, which cannot start less than 2 h after the washing machine. The storage system we have considered, the same for all the three prosumer configurations, is a Li-Po battery, the standard for a household usage, with a nominal capacity of 3.8 kWh and a starting level equal to 0.8 kWh. Other technical parameters are reported in Table 2.

Scenario generation has been carried out by adopting the Monte Carlo technique (see [3]). The overall demand and production from renewable systems have been determined starting from the hourly expected values and considering random variations. For all the test cases, the numbers of scenarios is 500. We have considered different test cases by combining different prosumers configurations and seasonal variations. For example, Fig. 2 reports the expected values of production and overall demand for each hour for the medium prosumer case in a summer day. As already stated, the purchasing and selling electricity prices are known in advance. While the former is related to the time-of-use block the hour belongs

[1] www.mathworks.com.

[2] www.gams.com.

[3] https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.

Table 1 Controllable loads parameters

Device	Time window	Default starting hour	Regret rate	Working hours	Hourly energy consumption [kWh]
Washing machine	9–13	9	1	1	1
Tumble dryer	9–15	11	2	2	1.5
Dish washer	14–17	15	0.5	2	1.2
Vacuum cleaner	10–16	15	1	1	0.5

Table 2 Storage system parameters

η_{in}	η_{out}	φ^{LB}	φ^{UB}	χ^{LB}	χ^{UB}
0.98	0.99	0.2	0.9	0.5	0.99

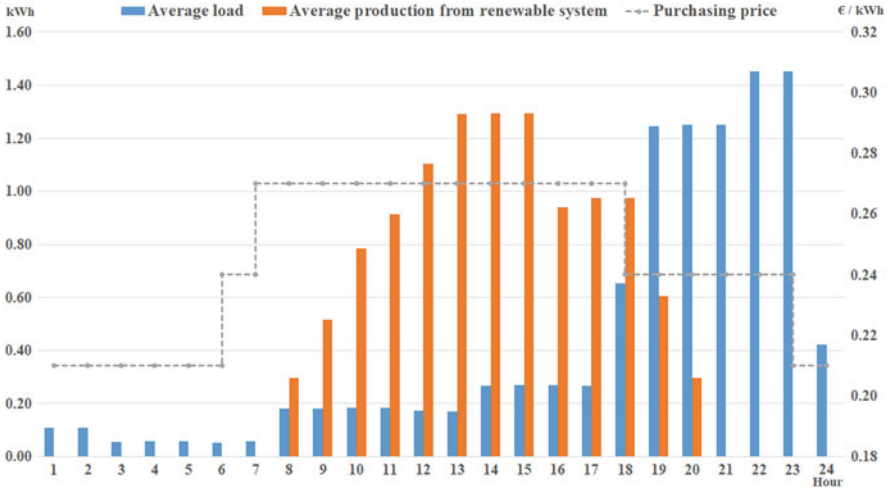


Fig. 2 Production, demand levels and purchasing price for the medium prosumer in a summer day

to (see Fig. 2), the latter is constant and equal to 0.1 €/kWh. The figure clearly shows the misalignment between hourly production and demand, which leads to an economic disadvantage for the prosumer: when the demand is higher than the production some energy must be purchased at a higher price w.r.t. the selling tariff. However, the possibility to effectively use the storage system and to schedule some loads can improve the overall economic efficiency.

Figure 3 reports the solution of the proposed model, in terms of storage system level and market operations, for the same test case under a single scenario. As evident, the controllable loads are planned to start when the PV production is high and the storage system is full, without buying energy from the grid. Moreover, the storage system is charged in the morning, when the purchasing price is lower (off-peak hours) and when the production is greater than the demand, while its energy is used later in the evening.

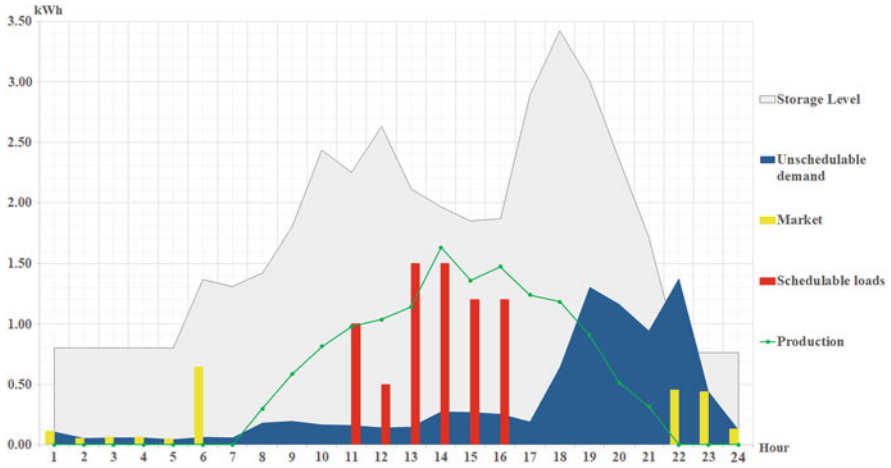


Fig. 3 Energy procurement and management for the medium prosumer in a summer day

In order to further quantify the benefit provided by the proposed approach in terms of monetary savings, we have compared the solutions obtained with different resource configurations. In the following we denote with "HEM Full" the model we have introduced in Sect. 2, assuming the availability of both storage system and controllable loads, with "HEM FS" a formulation with the decision variables related to the storage system assumed to be of first-stage, that is to be defined in advance with respect to the observation of uncertain parameters, with "HEM no SS" the case in which no storage system is available, with "HEM no SL" the situation with loads that cannot be controllable and with "HEM no SS and SL" the case without both of these two features. Figure 4 reports percentage increase of the annual cost of energy,

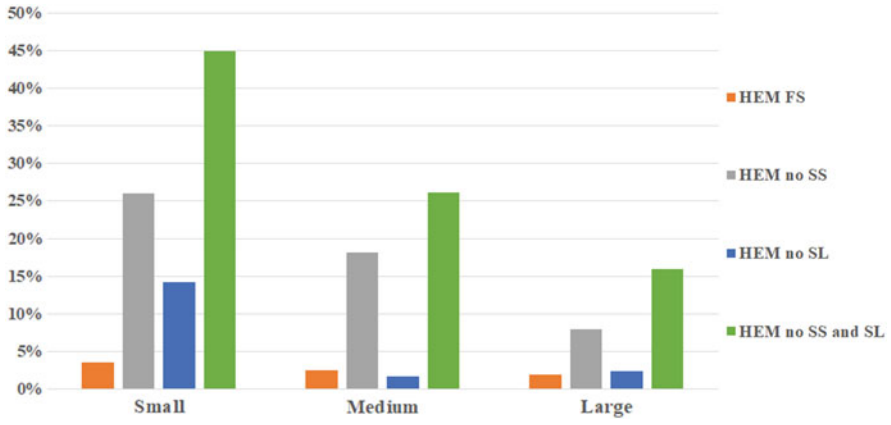


Fig. 4 Annual energy cost increase w.r.t. to HEM Full

obtained by multiplying the daily cost by the number of days falling into the three day types, of the different configurations w.r.t. "HEM Full". It is evident that the possibility to effectively manage some of the loads and the storage system can lead to significant savings. In the test cases we have considered the greater benefit has been registered for the small prosumer, which has the greater flexibility due to the storage system management. For a larger prosumer a system with higher capacity could have the same effect. It is also important to note that the possibility to use the storage system as a "reactive" resource in a quasi real-time fashion allows to have a more effective energy management.

Moreover, in order to evaluate the benefit obtained from explicitly accounting for uncertainty within the optimization process, we have computed the value of the stochastic solution (VSS) (see, for example, [11]), a measure traditionally used to compare deterministic and stochastic solutions. This value is defined as the difference between the objective function value obtained by solving the stochastic model and the one obtained by solving the same stochastic problem with the first-stage variables fixed to the values of the optimal solution of the deterministic (with the expected values) problem (also known as EEV). Table 3 reports the relative VSS expressed in percentage (with the respected to EEV) for the different prosumers on a daily basis. The results show that the medium prosumer is the one who can have the higher benefits from an effective management of uncertainty. Moreover, we note that in the summer, when the production is higher, the VSS values are more relevant.

A final set of experiments has been carried out in order to evaluate the impact of the maximum regret value on the solutions. Table 4 reports the objective function values obtained for different values of V.

Note that for $V = 0$ all the loads can not be controlled. As we increase the value of V, a more flexible scheduling can be obtained achieving higher savings in terms of total electricity procurement cost.

Table 3 Value of stochastic solution (%)

Prosumer	Winter	Spring/Autumn	Summer
Small	5.1	5.65	6.8
Medium	7.73	8.33	10.59
Large	5	5.7	6.2

Table 4 Energy costs for different values of V

	Small			Medium			Large		
V	Winter	Spring/ Autumn	Summer	Winter	Spring/ Autumn	Summer	Winter	Spring/ Autumn	Summer
0	2.42	1.22	0.63	2.91	1.67	0.77	3.54	2.53	1.28
3	2.35	1.11	0.34	2.84	1.65	0.76	3.47	2.21	1.26
5	2.35	1.10	0.31	2.84	1.65	0.76	3.46	2.20	1.26
7	2.35	1.10	0.27	2.84	1.65	0.75	3.46	2.20	1.26

4 Conclusions

The paper addressed the home energy management problem faced by a prosumer hosting PV panels and storage devices. The smart home is also equipped with a smart controller able to manage the electrical appliances eventually postponing their use in more convenient time slots. The inherent uncertainty affecting the main parameters involved in the decision process is explicitly accounted for by adopting the two-stage stochastic programming modeling paradigm. The solution provides the prosumer with the optimal scheduling of the controllable loads and the using profile of the storage system that guarantee the minimum expected energy procurement cost, taking into account the prosumer's comfort. The computational experiments have been carried out by considering three types of residential prosumers. The results have shown that significant savings can be achieved by the optimal use of the storage device and scheduling of the loads. The advantage deriving from the use of a stochastic programming approach over a deterministic formulation, measured in terms of VSS, is also significant.

Acknowledgements This work has been partially supported by Italian Minister of Economic Development, Fund HORIZON 2020 PON I&C 2014–2020, with the grant for research project F/050159/01-03/X32 "Power Cloud: Tecnologie e Algoritmi nell'ambito dell'attuale quadro regolatorio del mercato elettrico verso un new deal per i consumatori e i piccoli produttori di energia da fonti rinnovabili".

References

1. Belli, G., Giordano, A., Mastroianni, C., Menniti, D., Pinnarelli, A., Scarcello, L., Sorrentino, N., Stillo, M.: A unified model for the optimal management of electrical and thermal equipment of a prosumer in a DR environment. IEEE Trans. Smart Grid. **10**(2), 1791–1800 (2019)
2. Benetti, G., Caprino, D., Della Vedova, M.L., Facchinetti, T.: Electric load management approaches for peak load reduction: a systematic literature review and state of the art. Sustain. Cities Soc. **20**, 124–141 (2016)
3. Beraldi, P., De Simone, F., Violi, A.: Generating scenario trees: a parallel integrated simulation optimization approach. J. Comput. Appl. Math. **233**(9), 2322–2331 (2010)
4. Beraldi, P., Violi, A., Carrozzino, G., Bruni, M.E.: A stochastic programming approach for the optimal management of aggregated distributed energy resources. Comput. Oper. Res. **96**, 200–212 (2018)
5. Beraldi, P., Violi, A., Carrozzino, G.: The optimal management of the prosumer's resources via stochastic programming. In: Energy Reports – Proceedings of ICEER 2019, 6th International Conference on Energy and Environment Research (2019)
6. Chen, X., Wei, T., Hu, S.: Uncertainty-aware household appliance scheduling considering dynamic electricity pricing in smart home. IEEE Trans. Smart Grid **4**(2), 932–941 (2018)
7. Cheong Sou, K., Weimer, J., Sandberg, H., Johansson, K.H.: Scheduling smart home appliances using mixed integer linear programming. In: IEEE Conference on Decision and Control and European Control Conference, pp. 5144–5149 (2011)
8. Correa Florez, C.A., Gerossier, A., Michiorri, A., Kariniotakis, G.: Stochastic operation of home energy management systems including battery cycling. Appl. Energy **225**, 1205–1218 (2018)

9. Heredia, J.F., Cuadrado, M.D., Corchero, C.: On optimal participation in the electricity markets of wind power plants with battery energy storage systems. Comput. Oper. Res. **96**, 316–329 (2018)
10. Martinez-Pabon, M., Eveleigh, T., Tanju, B.: Optimizing residential energy management using an autonomous scheduler system. Expert Syst. Appl. **96**, 373–387 (2018)
11. Ruszczyński, A., Shapiro, A.: Stochastic Programming, Handbook in Operations Research and Management Science. Elsevier Science, Amsterdam (2003)
12. Violi, A., Beraldi, P., Ferrara, M., Carrozzino, G., Bruni, M.E.: The optimal tariff definition problem for a prosumers' aggregation. In: Daniele, P., Scrimali, L. (eds.) New Trends in Emerging Complex Real Life Problems. AIRO Springer Series, vol. 1, pp. 483–492. Springer, Cham (2018)
13. Yahia, Z., Pradhan, A.: Optimal load scheduling of household appliances considering consumer preferences: an experimental analysis. Energy **163**, 15–26 (2019)

Optimization Methods for the Same-Day Delivery Problem

Jean-François Côté, Thiago Alves de Queiroz, Francesco Gallesi, and Manuel Iori

Abstract In the same-day delivery problem, requests with restricted time windows arrive during a given time horizon, and it is necessary to decide which requests to serve and how to plan routes accordingly. We solve the problem with a dynamic stochastic method that invokes a generalized route generation function combined with an adaptive large neighborhood search heuristic. The heuristic is composed of destroying and repairing operators. The generalized route generation function takes advantage of sampled-scenarios, which are solved with the heuristic, to determine which decisions should be taken at any instant. Results obtained on different benchmark instances prove the effectiveness of the proposed method in comparison with a consensus function from the literature, with an average decrease of 10.7%, in terms of solution cost, and 24.5%, in terms of runtime.

Keywords Same-day delivery problem · Pickup and delivery problem · Dynamic stochastic algorithm · Adaptive large neighborhood search · Route generation function

1 Introduction

Same-day delivery is a problem that has several real-world applications in online retail. Other similar, related applications of this problem emerge in the delivery of groceries and transportation of patients between their homes and a hospital. This

J.-F. Côté
CIRRELT and Laval University, Quebec City, QC, Canada
e-mail: jean-francois.cote@fsa.ulaval.ca

T. A. de Queiroz (✉) · F. Gallesi · M. Iori
Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Reggio Emilia, Italy
e-mail: taq@ufg.br; francesco.gallesi@unimore.it; manuel.iori@unimore.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_30

problem is related to the classical NP-hard vehicle routing problem and claims research attention, especially due to the complicated, expensive logistic decisions that it requires to take. Assuming a time horizon over which a fleet of identical vehicles should operate, it is necessary to determine routes for these vehicles. The objective is to maximize the number of requests that can be delivered on time and, as a secondary objective, minimizing the traveled distance. Requests arrive dynamically during the time horizon, each one associated with a time window that needs to be respected. Consequently, vehicles may return to the depot after performing delivery to pickup products and continue serving the following requests that will arrive later on.

The Same-Day Delivery Problem (SDDP) is a dynamic problem introduced in [10]. The authors assumed that vehicles could return to the depot to pickup freight only after finishing their current routes. Based on sampled-scenarios, the authors used a consensus function as a way to make dynamic decisions and consequently generate the vehicle routes. To create the routes, they solved a team orienteering problem with time windows and multi-trips [5]. They also considered that a vehicle could wait at the depot while new requests arrive (i.e., they studied a waiting strategy). The idea is to anticipate decisions based on the already known requests and potential future ones that were sampled from known probability distributions. From the experimental results, the authors concluded that considering the uncertainty of future requests had an essential impact on the solution quality.

The SDDP can be viewed as the dynamic version of a one-to-one Pickup and Delivery Problem (PDP) with time windows (TW), with a single pickup location, which is the depot, where vehicles need to return for new pickups during the time horizon. Comprehensive surveys on the PDPTW can be found in [2], for to the transportation of goods, and in [6], for the transportation of people. Some further interesting related works are: [3], in the delivery of groceries, where time windows must be strictly respected, and requests are generally known 1-day in advance; [1], that aims at maximizing the total expected profit, where vehicles can depart from the depot as soon as requests are available (i.e., there is no waiting strategy); [4], in which release dates are associated with requests and a genetic algorithm with local searches is used to solve the problem; [8], that solves a PDPTW in which the pickup and delivery nodes are known in advance but not the time at which requests are available (i.e., requests are arriving dynamically during the time horizon); [9], that solves a multi-period problem in which requests are dynamically integrated into existing decisions, and some requests can be served on the next day.

In this work, we tackle the SDDP with the aim of minimizing the number of rejected requests, and, as a secondary objective, the total routing cost that is incurred from performing all deliveries. Moreover, when a vehicle starts performing its route, we allow it to return to the depot after serving a customer, and before completing its route to pickup more requests. The latter assumption generalizes the proposal in [10], which allows vehicles to return to the depot only after finishing their routes, and enlarges the space of decisions substantially. This is expected to come

at the expense of a greater computational effort. Considering these assumptions, we propose a generalized route generation function to improve how routes are built in [10]. We develop an Adaptive Large Neighborhood Search (ALNS) [7] to iteratively solve sub-instances on the sampled-scenarios.

In the following, we present results for three versions of the SDDP: *static*, in which the problem is solved with the ALNS for all the time horizon, assuming that all requests are known in advance; *dynamic*, in which the ALNS is applied several times during the time horizon in order to update the current solution; and *dynamic-stochastic*, in which the generalized route generation function with the ALNS, considering sampled-scenarios of future information, is used to update the current solution. Results are compared with those of the consensus function in [10], especially in the dynamic-stochastic version, considering 120 instances that such authors have proposed.

This work is organized as follows: Sect. 2 has a formal description of the problem, with its objectives and constraints; Sect. 3 describes the ALNS and the generalized route generation function, highlighting the differences over the methods proposed in [10]; Sect. 4 contains the experimental results of the three versions of the problem; finally, Sect. 5 brings concluding remarks and directions for future works.

2 Problem Definition

The SDDP that we study considers a fleet M of identical vehicles and a set L of customers locations over a geographical area. A central depot, denoted as node 0, is associated with start and end times between which vehicles can depart and arrive. These times are the depot working hours or the time horizon over which the depot and vehicles are in operation. With each pair $i, j \in L$, it is associated a deterministic travel time t_{ij} and a cost c_{ij} (e.g., distance) that are known in advance. During the depot working hours, requests arrive at a rate $\lambda_i \geq 0$ from each location $i \in L$. Let $\mathcal{R}$ be the set of requests that will occur during the time horizon. Set $\mathcal{R}$ is composed of requests that are known in advance and some others that are unknown at the beginning of the time horizon, but will become known as time unfolds. Each request $k \in \mathcal{R}$ has a service time μ_k, a demand d_k, and a delivery time window $[s_k, e_k]$. Request k is revealed at release time r_k and can only be served later on. Requests that are found impossible to deliver on time can be assigned to a third-party logistics operator at the expenses of an additional cost. The delivery costs incurred by the fleet are always lower than the cost of the third-party logistic operator.

Vehicles start and end at the depot according to their working hours and may serve one or more available requests, without violating their capacity Q. The design of the route associated with each vehicle may involve waiting at the depot for new requests or picking up some requests to perform the deliveries. Besides, no

diversion is allowed when a vehicle is on the way to a customer. As soon as a delivery is done, the vehicle can return to the depot to pickup new requests. This means the vehicle is not required to finish serving all its onboard requests before going back to the depot. The objective of the SDDP is to plan routes for vehicles, aiming at first maximizing the number of requests served by the fleet, and secondly minimizing the total cost of performing the routes.

The description above corresponds to the dynamic version of the problem. For this variant, we developed an ALNS to solve partial instances at given times of the time horizon. The ALNS is also used to solve the static version of the problem, in which all requests are known at the start time of the day. The solution of this version serves as an estimation for the other dynamic versions. Aiming at improving solutions of the dynamic version, we consider the dynamic-stochastic version, in which sampled-scenarios are used to help with decisions regarding possible future requests.

3 Solution Methods

This section first describes how the SDDP is modeled, then presents the different events that can occur in real-time. Therefore, it describes the ALNS and the two different approaches for tackling the problem: dynamic and the dynamic-stochastic.

3.1 Modeling

The problem is modeled as a classical PDPTW with the inclusion of release dates for the arrival of new requests. At any instant, the set of known requests is built. Each request is composed of a pickup node at the depot and a drop node at the customer location, besides a restricted time window. Modification of any element that was performed is forbidden, so only choices concerning new requests or nodes that were not visited can be changed. Scenarios containing future requests are generated to help on minimizing costs. Futures requests are dealt like regular requests, with the exception that a vehicle cannot take any action before the release date. This means the vehicle has to stay idle until the release of the request.

It is important to note that our method allows all possible sequences of potential future and known requests. This is not the case in [10], where future requests are also generated, but a vehicle must return to the depot to do the pickup every time a future request is encountered in its route. The drawback of this approach is that only a subset of possible routes can be produced. For example, it cannot create routes where a vehicle picks up a real request but waits at the depot for future requests.

In our approach, this type of routes is allowed, and the vehicle can return to the depot even if it is loaded with known requests still to be delivered.

3.2 Event Management

In [10] two types of events are defined: (1) arrival of a new request when there is at least one vehicle that is waiting at the depot; and, (2) arrival of a vehicle at the depot or completion of the waiting period of a vehicle. Every time a new event happens, instances of the PDPTW are generated and are solved using the ALNS. When allowing vehicles not to complete their routes, we also consider a delivery completion as a new event. Namely, when a vehicle completed a delivery, it can be diverted to the depot to pickup requests and perform the deliveries later. Finally, it is worth noting that this additional event will possibly increase computational time.

3.3 Adaptive Large Neighborhood Search

The proposed ALNS is based on [7], and uses a simulated annealing acceptance probability function to accept worse solutions. It works as follows: (1) it obtains a feasible solution x by a constructive heuristic; (2) it applies a destroy operator on x to obtain x'; (3) it applies a repair operator on x' to obtain x''; (4) it replaces x with x'' if x'' has lower cost or else by applying the acceptance probability function; (5) it goes back to step (2) if the maximum number of iterations is not reached, or otherwise it returns x.

At step (1), the initial solution is constructed by a regret insertion heuristic. At step (2), we consider the *related* and *random* destroy operators. In the first operator, requests that are closely related (i.e., in terms of cost, time, and capacity) are removed. In the second one, requests to be removed are randomly selected. Then, the removed requests are reinserted at step (3) by one of two repair operators. The first one is a greedy operator that reinserts iteratively each of the removed requests into its best position. The other one is a standard regret reinsertion operator. At steps (2) and (3), an operator is chosen according to the roulette wheel selection principle, in which a given weight is associated with each operator. These weights are dynamically updated by using statistics of previous iterations as well as a reaction factor that controls the influence of the weights. Moreover, at the end of step (3), a local search is applied in x'', consisting of determining the best moment to serve each request that has not been served yet. Regarding the acceptance probability function, a given initial temperature is decreased over the ALNS iterations, and thus the probability of accepting worse solutions in comparison with the current one is decreased as well.

3.4 Dynamic Problem

In this SDDP version, a PDPTW instance and its solution are maintained over time. On each new event, the instance is updated with further information (e.g., delivery completion, new requests, etc.) and elements performed in the past are fixed inside their routes. The ALNS is run to obtain a new solution, and it updates the maintained solution. New pickup and departure commands to the vehicles are generated. Requests that remained outside the solution are given to the third-party logistics operator when they become impossible to serve.

3.5 Dynamic-Stochastic Problem

To improve routes that are planned in the dynamic version for any event, sample-scenarios of future requests are used. These scenarios are generated from a probability function, taking into consideration the known requests until the current time. Each sampled-scenario is solved with the ALNS, similarly to what is performed in the dynamic problem, however, now also considering future requests that contemplate a time horizon.

After solving all scenarios, a generalized route generation function is used to identify the best solution among all of them. Then, such a best solution is used to update the current solution. This function works on the following way: (1) for each solution of each scenario, remove the sampled requests and every real requests that lie after at least one sampled request from all routes, since they indicate that a vehicle must wait or return to the depot to pickup some future requests; (2) assign a score to each solution based on the number of times each of its routes are in other solutions, where the solution with the highest score is chosen and implemented. As commented before, requests outside the solution are assigned to the third-party logistics operator when they become impossible to serve.

4 Experimental Results

All the methods were coded in the C++ programming language and run on an Intel 2.667 GHz Westmere EP X5650 processor. The experiments were carried out over a subset of instances from [10]. The instances under consideration are of two types that differ in the way customer locations have been generated, namely, clustered (C) and randomly dispersed (R). For each type, we consider data sets that contain 100 (C_1 and R_1) and 200 (C_2, C_6, R_2, and R_6) customers, as well as five types of time windows that are TW.d1, TW.f, TW.h, and TW.r, with 1-h deadlines, and TW.d2, with a 2-h deadline. Moreover, there are four different request arrival rates, namely, 1, 2, 3, and 4. Therefore, we have a total of 120 instances, where the first

instance is called TW.d1_C_1_hom_1 (and so on). The number of vehicles is fixed to 10 for each instance.

Regarding the parameters of the methods, we carried out preliminary experiments in which the sampling horizon was defined over the entire horizon, and the ALNS had 50 and 250 iterations, assuming 30 scenario samples. These experiments indicated, in terms of solution quality and runtime, that performing 250 iterations for the ALNS is preferable. Thus, such values were adopted when solving all the 120 instances. The results that we obtained are presented in Tables 1 and 2. Each line of these tables has the name of the instance, the solution values of the static, dynamic, and dynamic-stochastic versions as explained in Sect. 3, as well as the solution value of the dynamic-stochastic version obtained by using within the ALNS the route generation method by Voccia et al. [10] with the ALNS. For each problem, it is presented the total solution cost, the number of unserved requests (#Not), and total computational time in seconds.

Observing Table 1, the average solution cost and runtime (in seconds) are, respectively: 2203.5 and 25.4, for the static problem; 3319.6 and 26.0, for the dynamic problem; and, 2592.7 and 14,806.1, for the dynamic-stochastic problem that was solved with the generalized route generation function. We notice that the dynamic-stochastic that was solved with the consensus function in [10], where these values are 2913.2 and 19,430.6, respectively, is outperformed by the proposed method. Indeed, we can observe a decrease of 11.0% and 23.8%, respectively. In terms of the number of unserved requests, the proposed method performed the best with 0.3 more requests on average over [10].

The results of Table 2 are very similar to those of Table 1. In summary, from Table 2, the average solution cost, number of not served requests, and runtime (in seconds) are: 2321.4, 4.8, and 27.1, for the static problem; 3253.0, 14.4, and 27.3, for the dynamic problem; 2633.1, 8.7, and 14,622.6, for the dynamic-stochastic problem that was solved with the generalized route generation function; and, 2937.2, 8.9, and 19,552.5, the dynamic-stochastic that was solved with the consensus function in [10]. Once again, the proposed method can overcome the dynamic. In terms of solution cost and the number of not served requests, there is a decrease of 19.1% and 35.7%, respectively. In comparison with the dynamic-stochastic of the literature, in terms of solution cost and runtime, there are a decrease of 10.3% and 25.2%, respectively. On the other hand, it better approximates the results of the static problem, because, in terms of solution cost and the number of not served requests, they have the smallest percentage deviation.

Finally, with relation to the instances characteristics, comparing the dynamic-stochastic problem with the respective version that was solved with the consensus function in [10], from Tables 1 and 2, we can highlight that the latter performed worse for all types (R and C), time windows (TW.d1, TW.d2, TW.f, Tw.h, and TW.h), and requests arrival rates (1, 2, 3, and 4) in terms of average solution cost and runtime. Thus, we can conclude that the generalized route generation function, which allows vehicles to stop their current routes and return to the depot to pickup requests, performs well in practice.

Table 1 Results on the C instances

	Static			Dynamic			Dyn-Stoc			Dyn-Stoc in [10]		
Instance	Cost	#Not	Time (s)	Cost	#Not	Time (s)	Cost	#Not	Time (s)	Cost	#Not	Time (s)
TW.d1_C_1_hom_1	2264.0	1	2	2976.0	1	2	2878.4	1	704	2980.8	1	1667
TW.d1_C_1_hom_2	3275.2	2	5	4281.6	6	3	3980.8	4	2393	4192.0	6	3198
TW.d1_C_1_hom_3	4211.2	9	15	4913.6	22	6	4684.8	20	4578	4772.8	20	4016
TW.d1_C_1_hom_4	4270.4	13	21	4878.4	30	9	4601.6	26	9815	4620.8	25	9895
TW.d2_C_1_hom_1	1329.6	1	3	2633.6	1	2	1846.4	1	1382	1961.6	1	1938
TW.d2_C_1_hom_2	1820.8	1	12	3820.8	1	5	2185.6	1	5754	2609.6	1	8430
TW.d2_C_1_hom_3	2456.0	5	38	4497.6	10	11	2734.4	8	17,992	3305.6	6	22,907
TW.d2_C_1_hom_4	2515.2	4	61	4942.4	9	14	3009.6	4	35,838	3587.2	5	54,882
TW.f_C_1_hom_1	1398.4	1	3	2851.2	1	2	1891.2	2	1221	2312.0	2	2678
TW.f_C_1_hom_2	1886.4	1	10	3558.4	1	6	2257.6	1	4895	3113.6	1	6416
TW.f_C_1_hom_3	2531.2	5	30	3993.6	7	14	2937.6	8	14,820	3776.0	8	13,258
TW.f_C_1_hom_4	2609.6	4	47	4323.2	8	16	2910.4	5	27,199	3601.6	7	25,843
TW.h_C_1_hom_1	1134.4	1	3	1940.8	5	4	1512.0	2	1863	1676.8	3	1933
TW.h_C_1_hom_2	1624.0	1	12	2160.0	10	16	1910.4	1	7055	2275.2	2	7364
TW.h_C_1_hom_3	2163.2	5	55	2561.6	28	72	2636.8	14	28,556	2622.4	13	25,866
TW.h_C_1_hom_4	2246.4	4	71	2620.8	33	126	2547.2	11	46,525	2611.2	17	56,207
TW.r_C_1_hom_1	1168.0	1	4	1969.6	6	4	1544.0	3	1712	1801.6	3	1814
TW.r_C_1_hom_2	1401.6	1	12	2297.6	12	16	1649.6	1	6703	1972.8	4	7064
TW.r_C_1_hom_3	2438.4	5	39	2684.8	31	50	2683.2	13	21,983	2859.2	12	31,046
TW.r_C_1_hom_4	2401.6	5	68	2590.4	39	97	2681.6	11	43,907	2987.2	13	46,370
TW.d1_C_2_hom_1	2696.7	2	1	3253.0	2	2	3302.3	2	608	3265.1	2	1553
TW.d1_C_2_hom_2	3210.2	3	3	4122.8	4	2	3690.2	4	1972	3917.2	3	3431
TW.d1_C_2_hom_3	4254.0	17	13	5032.6	30	6	4812.1	33	3714	4801.9	30	2933
TW.d1_C_2_hom_4	4174.4	26	17	4982.3	38	7	4447.3	40	8061	4568.2	39	9112

TW.d2_C_2_hom_1	1578.6	2	2	3041.9	2	2	1827.9	2	1122	1947.0	2	2129
TW.d2_C_2_hom_2	1591.6	3	7	3542.3	3	4	1813.0	3	4661	2323.7	3	7272
TW.d2_C_2_hom_3	2251.2	7	36	4815.8	8	11	2527.4	7	17,309	3330.2	7	26,622
TW.d2_C_2_hom_4	2574.0	5	54	4832.1	7	16	2922.1	5	35,374	3570.7	6	49,661
TW.f_C_2_hom_1	1582.3	2	2	3046.5	2	2	1839.1	2	1019	2480.0	2	1868
TW.f_C_2_hom_2	1607.4	3	6	3321.9	3	4	2052.1	3	4108	2454.9	3	7197
TW.f_C_2_hom_3	2280.0	7	29	4353.5	12	14	2686.5	7	14,388	3739.5	9	13,423
TW.f_C_2_hom_4	2674.8	5	43	4281.5	11	23	3182.3	6	27,112	3732.8	7	29,382
TW.h_C_2_hom_1	1375.8	2	3	2266.0	6	3	1586.0	3	1442	1922.8	5	2998
TW.h_C_2_hom_2	1162.8	3	7	2016.7	9	11	1501.4	3	5933	1778.6	5	5184
TW.h_C_2_hom_3	2044.7	7	39	2881.9	17	68	2439.1	11	22,743	2668.8	13	39,875
TW.h_C_2_hom_4	2418.3	6	63	2635.4	33	95	2821.4	15	43,067	2953.3	11	53,510
TW.r_C_2_hom_1	1312.6	2	3	2094.9	6	3	1734.0	2	1429	1848.4	2	2253
TW.r_C_2_hom_2	1385.1	3	7	2160.0	15	8	1709.8	4	5674	2037.2	3	9349
TW.r_C_2_hom_3	2011.2	7	39	2888.4	22	56	2887.4	7	22,960	2935.8	8	37,789
TW.r_C_2_hom_4	2337.7	5	61	2372.5	41	112	2795.7	6	41,491	2876.3	8	69,096
TW.d1_C_6_hom_1	1054.7	0	3	1666.9	1	4	1332.2	0	1779	1468.1	0	3870
TW.d1_C_6_hom_2	1713.8	2	18	2374.7	12	29	2047.5	4	10,909	2139.4	5	11,673
TW.d1_C_6_hom_3	1721.3	3	37	2556.6	16	77	2164.7	5	22,547	2408.4	3	33,890
TW.d1_C_6_hom_4	2223.8	6	83	2705.6	32	131	2796.6	10	52,112	2686.9	11	71,392
TW.d2_C_6_hom_1	1845.9	0	2	2377.5	0	2	2142.2	0	785	2353.1	0	2589
TW.d2_C_6_hom_2	3384.4	3	6	4400.6	6	4	4322.8	6	2878	4215.0	6	3847
TW.d2_C_6_hom_3	3802.5	3	13	4769.1	15	6	4283.4	12	6093	4488.8	14	6891
TW.d2_C_6_hom_4	4254.4	13	27	5012.8	28	9	4596.6	25	11,753	4710.9	21	8514
TW.f_C_6_hom_1	1247.8	0	3	2356.9	0	2	1393.1	0	1462	1589.1	0	3069
TW.f_C_6_hom_2	1928.4	2	13	4281.6	4	5	2294.1	2	7143	2794.7	2	13,094

(continued)

Table 1 (continued)

	Static			Dynamic			Dyn-Stoc			Dyn-Stoc in [10]		
Instance	Cost	#Not	Time (s)	Cost	#Not	Time (s)	Cost	#Not	Time (s)	Cost	#Not	Time (s)
TW.f_C_6_hom_3	2079.4	3	32	4267.5	7	12	2452.5	4	18,950	3011.3	4	33,698
TW.f_C_6_hom_4	2462.8	6	75	4988.4	10	17	2891.3	8	42,447	3343.1	8	62,566
TW.h_C_6_hom_1	1286.3	0	3	2175.0	0	2	1393.1	0	1270	1815.0	0	3592
TW.h_C_6_hom_2	1981.9	2	11	3826.9	5	7	2441.3	3	5948	3285.9	2	8083
TW.h_C_6_hom_3	2167.5	3	26	3829.7	7	15	2541.6	4	15,367	3310.3	4	23,639
TW.h_C_6_hom_4	2595.0	6	60	4156.9	9	30	2879.1	8	33,223	3699.4	8	36,124
TW.r_C_6_hom_1	1080.0	0	3	1455.9	4	3	1319.1	0	1753	1459.7	0	3689
TW.r_C_6_hom_2	1760.6	2	14	2391.6	19	16	2061.6	4	8192	2261.3	5	17,287
TW.r_C_6_hom_3	1772.8	3	38	2472.2	21	56	2046.6	7	23,503	2354.1	8	41,579
TW.r_C_6_hom_4	2180.6	6	108	2672.8	24	207	2502.2	13	71,170	2602.5	19	69,321
Average	2203.5	4.2	25.4	3319.6	12.5	26.0	2592.7	7.0	14,806.1	2913.2	7.3	19,430.6

Table 2 Results on the R instances

	Static			Dynamic			Dyn-Stoc			Dyn-Stoc in [10]		
Instance	Cost	#Not	Time (s)	Cost	#Not	Time (s)	Cost	#Not	Time (s)	Cost	#Not	Time (s)
TW.d1_R_1_hom_1	2078.7	1	2	2798.7	1	2	2562.6	1	706	2674.8	1	2222
TW.d1_R_1_hom_2	1756.4	0	1	2379.5	0	2	2231.9	0	696	2461.0	0	2399
TW.d1_R_1_hom_3	4300.6	12	14	4999.4	24	6	4614.2	27	4717	4800.0	23	4338
TW.d1_R_1_hom_4	4453.9	8	24	5047.0	22	9	4558.3	24	11,475	4714.8	22	11,696
TW.d2_R_1_hom_1	1215.5	1	4	2471.6	1	3	1538.7	1	1374	1740.0	1	3177
TW.d2_R_1_hom_2	1117.9	0	3	2371.1	0	2	1368.1	0	1391	1592.1	0	3790
TW.d2_R_1_hom_3	2640.0	4	38	4654.8	8	14	2926.5	6	18,259	3400.6	7	26,058
TW.d2_R_1_hom_4	2598.3	1	67	4773.9	6	19	2970.4	5	40,201	3407.0	1	61,951
TW.f_R_1_hom_1	1267.7	1	3	2429.0	1	2	1577.4	1	1217	2034.2	1	2902
TW.f_R_1_hom_2	1145.5	0	2	2090.7	0	2	1384.8	0	1246	1799.4	0	2519
TW.f_R_1_hom_3	2707.7	4	31	4277.4	9	17	3031.0	7	14,971	3774.2	6	15,039
TW.f_R_1_hom_4	2688.7	1	52	4053.9	12	26	3165.2	4	30,960	3676.5	7	32,787
TW.h_R_1_hom_1	1105.2	1	4	1807.7	3	7	1306.5	2	1889	1639.4	1	2652
TW.h_R_1_hom_2	1553.0	0	12	2130.4	9	15	1735.7	1	7285	2151.3	0	17,666
TW.h_R_1_hom_3	2266.5	7	46	2990.3	22	66	2537.4	13	22,523	2829.7	16	27,810
TW.h_R_1_hom_4	2488.7	2	93	2768.7	24	173	2770.4	11	62,857	2913.0	11	92,204
TW.r_R_1_hom_1	1101.3	1	4	1906.5	2	6	1445.8	1	1854	1604.5	1	2786
TW.r_R_1_hom_2	1560.0	0	13	2260.9	6	13	1836.5	1	7325	2073.0	1	11,856
TW.r_R_1_hom_3	2498.7	4	55	2549.0	40	79	2707.7	11	28,977	2893.5	10	40,499
TW.r_R_1_hom_4	2375.7	3	71	2702.6	32	106	2733.9	11	46,710	2593.0	19	67,312
TW.d1_R_2_hom_1	2291.4	2	1	2769.5	2	2	2534.3	2	657	2707.6	2	1676
TW.d1_R_2_hom_2	3411.6	5	4	3792.4	15	3	3492.9	13	2035	3763.6	12	3803
TW.d1_R_2_hom_3	4467.8	22	12	4956.7	31	5	4823.8	32	3594	4900.2	34	2742

(continued)

Table 2 (continued)

	Static			Dynamic			Dyn-Stoc			Dyn-Stoc in [10]		
Instance	Cost	#Not	Time (s)	Cost	#Not	Time (s)	Cost	#Not	Time (s)	Cost	#Not	Time (s)
TW.d1_R_2_hom_4	4410.5	37	22	4969.5	60	9	4710.5	54	8440	4741.0	55	8042
TW.d2_R_2_hom_1	1331.4	2	3	2608.6	2	2	1567.6	2	1209	1727.6	2	1971
TW.d2_R_2_hom_2	1738.5	4	8	3521.7	4	6	2112.4	4	5206	2495.2	4	11,402
TW.d2_R_2_hom_3	2881.0	4	33	4787.1	8	11	3333.2	6	14,383	3692.2	5	23,146
TW.d2_R_2_hom_4	3513.3	9	60	4999.0	20	20	3427.6	17	35,536	3831.4	18	38,436
TW.f_R_2_hom_1	1339.0	2	2	2501.9	2	2	1572.4	2	1081	2197.1	2	2930
TW.f_R_2_hom_2	1811.9	4	7	3390.7	4	5	2148.1	4	4512	3087.3	4	8341
TW.f_R_2_hom_3	2954.4	4	29	4494.5	12	15	3373.9	6	12,888	4087.9	5	13,915
TW.f_R_2_hom_4	3265.7	10	50	4347.6	20	25	3592.4	18	27,822	4166.7	17	19,028
TW.h_R_2_hom_1	1268.6	2	2	1869.5	7	3	1466.7	2	1398	1663.8	5	2091
TW.h_R_2_hom_2	1601.7	4	10	2314.7	9	15	2030.1	4	6975	2279.0	5	7954
TW.h_R_2_hom_3	2267.1	6	42	2383.1	34	72	2622.1	11	19,423	2629.1	18	29,474
TW.h_R_2_hom_4	2727.6	16	81	2557.1	61	128	2985.7	25	48,078	3139.0	29	57,724
TW.r_R_2_hom_1	1132.4	2	3	1674.3	7	4	1417.1	2	1475	1541.9	2	1624
TW.r_R_2_hom_2	1503.5	4	10	2578.5	5	12	1842.6	4	6482	2111.4	4	11,709
TW.r_R_2_hom_3	2481.3	5	37	2619.2	35	60	2868.1	14	17,836	2881.0	12	30,735
TW.r_R_2_hom_4	2973.3	15	68	3558.1	38	48	3505.7	25	42,296	3434.3	26	39,780
TW.d1_R_6_hom_1	2413.2	0	1	2932.8	0	2	2873.4	0	631	2916.3	0	1766
TW.d1_R_6_hom_2	2795.1	3	7	4054.2	3	5	3572.3	3	3562	3852.9	3	7034
TW.d1_R_6_hom_3	4496.3	17	17	5027.1	30	7	4804.6	27	4694	4992.0	27	2948
TW.d1_R_6_hom_4	4631.1	16	22	5026.2	35	9	4673.5	29	8408	4814.8	27	6503
TW.d2_R_6_hom_1	1404.8	0	2	2713.8	0	2	1702.0	0	1166	2060.9	0	2494
TW.d2_R_6_hom_2	1625.5	3	13	3352.6	3	6	1881.2	3	6529	2379.7	3	12,211
TW.d2_R_6_hom_3	2743.4	2	50	4914.5	6	14	3071.1	4	20,234	3632.3	4	25,544

TW.d2_R_6_hom_4	2822.8	3	65	4851.7	7	17	3130.2	6	38,330	3579.7	6	55,518
TW.f_R_6_hom_1	1441.1	0	2	2693.9	0	2	1694.3	0	1067	2343.9	0	2393
TW.f_R_6_hom_2	1668.9	3	11	3100.6	3	7	2010.5	3	5474	2678.8	3	8215
TW.f_R_6_hom_3	2847.7	2	39	4391.1	7	20	3136.6	5	16,332	4182.5	4	11,064
TW.f_R_6_hom_4	2880.0	3	52	4292.3	12	24	3274.2	7	29,221	3904.6	7	29,138
TW.h_R_6_hom_1	1268.3	0	2	1761.5	4	4	1452.1	3	1519	1586.4	1	2006
TW.h_R_6_hom_2	1488.0	3	13	2332.6	7	15	1886.8	4	7777	2103.7	4	13,214
TW.h_R_6_hom_3	2520.0	3	60	3202.2	29	75	2968.6	13	28,453	2900.3	13	30,533
TW.h_R_6_hom_4	2359.4	7	76	2254.2	49	160	2502.5	13	49,839	2804.3	15	78,774
TW.r_R_6_hom_1	1174.7	0	3	2448.4	1	3	1520.4	0	1436	1793.4	0	1907
TW.r_R_6_hom_2	1390.2	3	13	2099.1	10	16	1748.3	4	7623	1903.4	3	11,141
TW.r_R_6_hom_3	2698.2	2	75	2882.8	27	106	2992.6	11	30,495	3141.2	9	47,257
TW.r_R_6_hom_4	2321.5	6	78	2691.7	32	126	2660.3	13	46,607	2808.9	15	75,305
Average	2321.4	4.8	27.1	3253.0	14.4	27.3	2633.1	8.7	14,622.6	2937.2	8.9	19,552.5

5 Concluding Remarks and Future Research Directions

In this paper, we studied a same-day delivery problem in which requests can be rejected (i.e., reassigned to a third-party logistics operator by paying a cost). We developed a generalized route generation function and combined it with an ALNS. We also adopted sampled-scenarios to anticipate potential future requests and improve decisions. Aiming at improving results of a recent algorithm in [10], our function allows vehicles to return to the depot to pickup requests even if they have not completed their routes.

The computational results of the static, dynamic, and dynamic-stochastic versions over different types of instances, arrival rates, and time windows have indicated that the proposed method is quite effective to solve the problem when sampled-scenarios are taken into consideration. In general, there is an overall average increase in the solution cost, considering the static problem, of 42.3%, compared with the dynamic, 15.5%, compared with the dynamic-stochastic that uses the generalized route generation function, and 29.3%, compared with the dynamic-stochastic that uses the consensus function in [10]. In terms of runtime, this increase is of 1.5%, 56,061.8%, and 74,295.4%, respectively.

Future works will focus on reducing the total runtime of the proposed method, including a careful study on the number of scenario samples, sampling horizon, iterations of the ALNS, and events. We will also attempt policies to improve insertion algorithms. We are also interested in presenting a formal mathematical formulation of the problem, as well as building a parallel version of the proposed dynamic-stochastic method.

Acknowledgements Thiago Alves de Queiroz would like to thank CNPq (grant number 308312/2016-3) and FAPEG for their financial support. Manuel Iori acknowledges support from the University of Modena and Reggio Emilia, under grants FAR 2017 Multiscale modeling in science, industry, and society and FAR 2018 Analysis and optimization of healthcare and pharmaceutical logistic processes. Jean-François Côté acknowledges support by the Canadian Natural Sciences and Engineering Research Council (NSERC) under grants 2015-04893. We thank Compute Canada for providing high-performance computing facilities.

References

1. Azi, N., Gendreau, M., Potvin, J.Y.: A dynamic vehicle routing problem with multiple delivery routes. Ann. Oper. Res. **199**(1), 103–112 (2012)
2. Battarra, M., Cordeau, J.F., Iori, M.: Pickup-and-delivery problems for goods transportation. In: Toth, P., Vigo, D. (eds.) Vehicle Routing: Problems, Methods, and Applications, 2nd edn., pp. 161–191. SIAM, Philadelphia (2014). Chapter 6
3. Campbell, A.M., Savelsbergh, M.: Incentive schemes for attended home delivery services. Transp. Sci. **40**(3), 327–341 (2006)
4. Cattaruzza, D., Absi, N., Feillet, D.: The multi-trip vehicle routing problem with time windows and release dates. Transp. Sci. **50**(2), 676–693 (2016)

5. Gunawan, A., Lau, H.C., Vansteenwegen, P.: Orienteering problem: a survey of recent variants, solution approaches and applications. Eur. J. Oper. Res. **255**(2), 315–332 (2016)
6. Ho, S.C., Szeto, W., Kuo, Y.H., Leung, J.M., Petering, M., Tou, T.W.: A survey of dial-a-ride problems: literature review and recent developments. Transp. Res. B Methodol. **111**, 395–421 (2018)
7. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. **40**(4), 455–472 (2006)
8. Srour, F.J., Agatz, N., Oppen, J.: Strategies for handling temporal uncertainty in pickup and delivery problems with time windows. Transp. Sci. **52**(1), 3–19 (2018)
9. Ulmer, M.W., Thomas, B.W., Mattfeld, D.C.: Preemptive depot returns for dynamic same-day delivery. EURO J. Transp. Logist. **8**, 327–361 (2018)
10. Voccia, S.A., Campbell, A.M., Thomas, B.W.: The same-day delivery problem for online purchases. Transp. Sci. **53**(1), 167–184 (2019)

Intermodality and Rail Transport: Focus on Port Rail Shunting Operations

Daniela Ambrosino and Veronica Asta

Abstract The presence of many actors interacting in the intermodal transport where rail modality is involved, causes some inefficiencies and some bottlenecks. These bottlenecks make the intermodal transport rigid and less fluid than the road transport. Operation research approaches can be useful as support for the decision makers in order to improve the process and to favour the development of rail transport. Thus, this work aims at describing the entire rail transport process within the intermodal transport in order to highlight the critical points that can be improved; in particular, both port rail shunting activities, and port rail terminal operations are discussed. A focus on a novel critical aspect never investigated before (at least for the author knowledge) is reported: the port rail shunting re-scheduling problem. A discussion on possible approaches for solving this problem is presented, together with a first approach based on a space-time network.

Keywords Port rail shunting activities · Port rail shunting scheduling problem · Space-time network

1 Introduction

Goods are transferred every day from origins to final destinations and, generally, shipments involve more than a single mode of transport. In particular, deep-sea ports play an important role of gateways for both import and export cargoes and the hinterland transportation system too [11].

D. Ambrosino (✉)
Department of Economics and Business Studies, University of Genova, Genova, Italy
e-mail: daniela.ambrosino@economia.unige.it; ambrosin@economia.unige.it

V. Asta
Italian Centre of Excellence on Logistics, Transports and Infrastructures, University of Genova, Genova, Italy
e-mail: veronica.asta@economia.unige.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_31

The hinterland haulage (i.e. the freight transport between the origin/destination ports and the origin/destination of cargoes in the hinterland) is facing the problem to quickly manage increasing cargo volumes (due to mega vessel employment), traffic congestion and traffic emission problems. The most of freight transportation is done by road because of time saving and flexibility, even though rail freight transportation is considered environmentally friendly. The intermodal freight transport seems to be the best answer in order to obtain efficient hinterland transportation systems. The development of intermodalism requires to pay attention to three elements: transport links, transport nodes, and the provision of efficient services [7]. A review of operations research opportunities in intermodal freight transport research is reported in [12].

In order to provide door-to-door service, railways must be integrated with existing logistical networks. Interfaces between railways and other transport modes are essential in order to encourage a modal shift. An example of global freight transport is depicted in Fig. 1, in which it is possible to note the hinterland haulage and the shunting yards that are present for each node of the network where goods have to change transport mode. Shunting yards represent important nodes for improvement, since for example the average dwell time of freight cars in terminals is more than 23 h in the U.S. Therefore, improvements in the operational processes at shunting yards can lead to a better competitiveness of rail transport [8].

In the literature the term shunting process is generally used for the operations in the shunting yards of decoupling the freight cars of inbound trains and form new outbound trains heading to some other shunting yard or to the final destination, see for example [3, 8, 19].

Effective planning and operation of shunting yards is therefore a decisive factor for reliable and punctual freight transportation, and the effectiveness of the shunting yard also affects the overall rail system fluidity. Unfortunately, scheduling the operations of a yard is hard as there are different shunting tasks with complex dependencies.

In this work the shunting operations of freight trains either within the port area or the intermodal inland terminal are considered. In particular, we refer to Italian transportation systems in which there is a shunting company that has the duty of taking the train from the railway station outside the port/inland area and bring it to the dedicated area within the freight terminal. Therefore, the main shunting activities to manage are those concerning the whole train's transfer within the port/inland areas.

The shunting procedure is complex and rail yards constitute bottlenecks in the rail freight network, often causing delays to individual shipments. Thus, thinking that an improvement in shunting yard activities, related to the connection of the rail transport to road and sea ones, can have a great impact on the reliability of rail services, the present work tries to describe the container handling process from the ship discharge, passing through the yard's stop, until the container load on the train and the train departure on the railway network towards the inland terminal of destination, focusing the attention on shunting depots that are present at different points of the network, as depicted in Fig. 1. Moreover, in the present work the critical

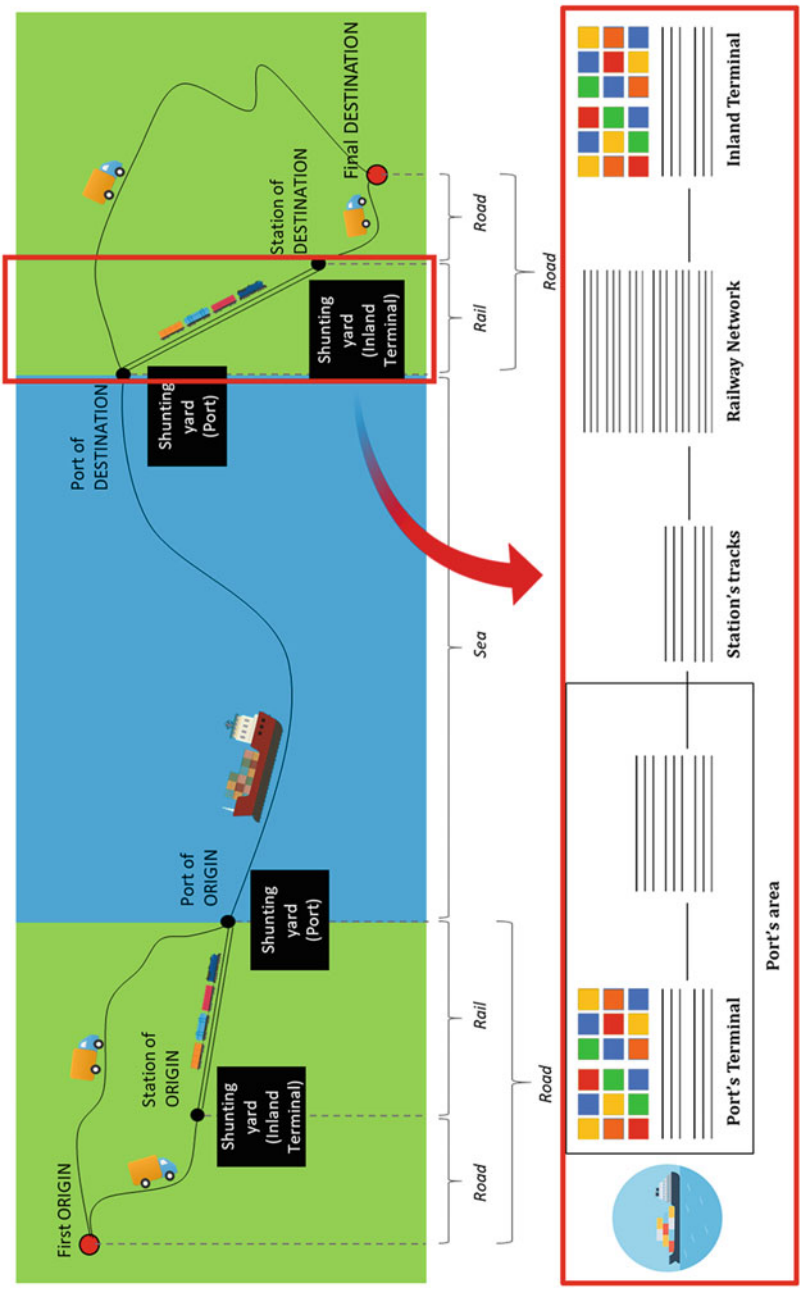


Fig. 1 A maritime based intermodal freight transport system

points of the analyzed process will be highlighted. The entire process is complex both in terms of passages to execute, subjects involved, flows of information and documents to manage.

The paper is organized as follows: Sect. 2 describes optimizing activities to make intermodal transport competitive; Sect. 3 is dedicated to the analysis of the port rail shunting scheduling and re-scheduling problem. It includes the description of a space-time network model for scheduling shunting operations and some very preliminary results. Finally, in Sect. 4 some conclusions are outlined.

2 Optimizing Activities to Make Intermodal Transport Competitive

The presence of many actors interacting in the intermodal transport where rail modality is involved, causes some inefficiencies and some bottlenecks. Let us start describing the import and export cycle.

The shipping company transports goods in import until the port's terminal by ship. The ship docks at the terminal's quay and let the discharge operations starting. The port's terminal has to manage, other than the discharge operations, all the activities related to both the goods stocking at the terminal's yard and the trains loading. All these activities can be guided by a decision support system based on optimization models and methods [17].

When the train is loaded, the shunting manager (in the following SM) intervenes in the process. This actor has the duty to execute all the operations related to the trains transfer from the ports terminal to the tracks outside the port. SM carries out also other accessory operations, e.g. the wagons discarding, transfer of single carriages, introduction/extraction of empty carriages. The goods to be loaded on the train must be subjected to customs controls. The whole train must be controlled by the financial police and it must be subjected to the technical verification by the railway undertaking. This latter is the actor that will manage the train during the transfer on the national network. The first delivery passage of the train is from the terminal manager to SM and it happens when the train is on the terminals tracks. The second delivery passage of the train is from the SM to the railway undertaking and usually happens when the train is positioned on the tracks of the railway station outside the port area. Since this moment, the train will travel on the rail national network, controlled by the infrastructure manager, toward the inland terminal of destination. When the train will arrive at the inland terminal will be taken over by the SM of the inland terminal, which will transfer it inside for the next discharging operations.

The flow of the export cycle is exactly the viceversa. When the train arrives at the last railway station outside the port area, the delivery passage of the train, from the railway undertaking to SM of the port, happens. The SM has the duty to transfer the train within the port area until the delivery on the terminals tracks. It has also to execute the required accessory operations.

By analyzing port rail shunting activities (Sect. 2.1) and port rail terminal operations (Sect. 2.2), the critical points where it is possible to optimize the process can be underlined.

2.1 Port Rail Shunting Activities

In this section the role of the SM in the Italian transportation network is described and the identification of the problems arisen in the port shunting areas will follow.

The port rail SM receives an annual plan of the operations to execute on the arrival and departing trains to and from the port. This specific plan is updated on weekly base keeping into consideration hours variations, and suppression. The main problem arises on daily level when events such as delays, suppression and inserting of new extraordinary trains occur. At this point, SM has to quickly manage and re-plan the operations in order to execute all the requested activities, respecting both the fixed time slot with the terminal and the time tracks of the trains departing on the national rail network. All the activities need specific resources both in terms of types and numbers, in order to be completed, such as shunting teams, locomotives and tracks.

Thanks to optimization approaches for supporting SM during the real time re-scheduling of daily operations the whole rail transport process will gain in terms of reduction of delays and major reliability [4]. Also at tactical and strategic levels optimization methods can support SM for defining the best policy to improve the service quality in favour of intermodal transport. At strategic level SM decides about the infrastructure investments and the resources sizing necessary to execute in the following years forecasted activities.

At tactical level the decisions are related to the assignment of the resources belonging to the port shunting company to the different activities that have to be carried out. Note that, generally, all the operations that the shunting manager must execute are planned 1 year before.

2.2 Rail Activities in a Maritime Terminal

In this section the role of the maritime terminal is described and the identification of the problems arisen in the rail activities in the terminal will follow. Regarding the export flows, the terminal receives the trains with containers that will leave the terminal by ship. Generally, these containers are transferred either from the train to the ship or from the train to the yard. It is important to be able to receive the train in the correct planned time window and to unload it quickly.

As far as the import flow is considered, the terminal activities to manage in such a way to be efficient and to be able to respect the time windows for trains departures are related to the optimal planning and management of train loading and the optimal

use of terminal resources (rail tracks, reach stacker, etc.). Containers unloaded from the vessels and leaving the terminal by rail are stored in a rail yard near tracks. The optimal management of the storage areas dedicated to the terminal rail cycle is really important. In fact, in accordance with the available information the storage in the yard can be more or less well organized and can require or not housekeeping operations before starting the loading operations [10]. Moreover, depending on the terminal layout and operational procedures, it is possible to maximize the performance of the whole rail cycle in terms of re-handles and travel distances by fulfilling timing constraints and safety requirements [6].

The train loading operations must be managed in such a way to optimize the usage of all the handling equipment/resources available in the terminal and to respect the train load plans. The definition of the train load plans has the objective of optimally assigning containers to slots of the train wagons in order to satisfy structural and stability constraints of trains and while maximizing key performance indicators specifically characterizing the terminal operations (i.e., related to the operativity of handling resources or to commercial requirements) [1]. The optimization of loading/unloading operations specifically refers to the optimal definition of all the activities of handling resources necessary for performing the movements of goods on/from the train wagons. The activities must be scheduled in order to respect timing constraints, the features of handling procedures and to obtain the best exploitation of terminal resources that can be shared among several terminal activities [9].

3 Port Rail Shunting Scheduling and Re-scheduling Problem

As far as shunting activities are considered, one of the most impacting task on reliability that SM has to realize is the re-scheduling of the activities, due to delays, suppression and inserted new extraordinary trains. These activities, daily managed by SM, are here below better explained.

The primary shunting (see Fig. 2) consists on both the delivery passage of the train between the railway undertaking and the shunting manager and the transfer of the train for bring it outside (import cycle) or inside (export cycle) the port area. The primary shunting usually needs one shunting team, one locomotive and two tracks, the one for transferring the train inside/outside the port and the one to position the train within the port area.

The secondary shunting (see Fig. 2) consists on both the transfer of the train for bring it outside (import cycle) or inside (export cycle) the terminal area and the delivery passage of the train between the shunting manager and the terminal manager. The resources needed for performing secondary shunting are the same used for the primary shunting, and the two tracks are used to keep the train within the port area and to transfer the train inside/outside the terminal.

Finally, the accessory shunting consists on operations, which may be requested from the actors on wagons, such as wagons' discarding and extraction or intro-

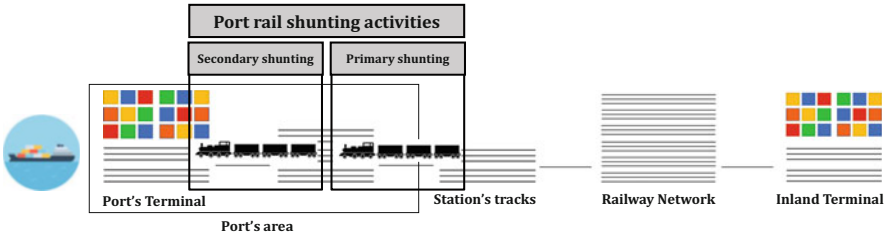


Fig. 2 Primary and secondary shunting operations

duction of empty/full wagons on a track. Each accessory shunting operation has a physical point of origin and a physical point of destination (i.e. terminal, stations and yard tracks). The resources needed vary operation from operation. Generally, the processing time for executing these operations depends on the train on which the activities have to be made.

3.1 Port Rail Shunting Re-scheduling Problem

Given a time horizon (i.e. 1 day), given a set of jobs, i.e. the trains passing through the port area on which some shunting activities, also called operations, have to be performed on, given a set of different types of resources R needed to execute the operations, the problem consists on defining the best sequence of activities to perform on jobs while satisfying some capacity, sequencing, time windows and priority constraints. In the re-scheduling problem the best sequence is the one minimizing the sum of the delays of the completion of the jobs with respect to their due time (derived by the scheduling).

Let be M the set of operations, composed by the primary, the secondary and the unique shunting operations, and the accessory shunting operations.

The operations that have to be performed on job j are known, together with their execution time (i.e. $t_{j,m}$ is the processing time of operation m for job j). Moreover, they have to be performed with a specific order. Let $r_{j,m}$ be the sequence number of operation m on job j.

Set R of the needed resources to execute the shunting activities is the union of three subsets: SQ the shunting teams, L the locomotives and B the tracks. Note that, some operations require specialized resources, i.e. an electrical locomotive instead of a diesel one, and so on; as a consequence, it is necessary to distinguish the different types of resources. Thus, SQ is the union of the set of general shunting teams SQ_G and shunting teams for a specific activity SQ_S; L is the union of diesel locomotives and electrical ones (L_D U L_E), while in B there are track(s) for the entrance in the port area B_E, shunting park tracks B_P and the tracks from the shunting park to the port terminal areas B_T.

For each operation it is necessary to know, not only the amount of needed resources, but also the compatibility with different types of resources. For example, a secondary shunting operation requires and is compatible with a general a shunting team $sq \in SQ_G$, either an electric locomotive $l \in L_E$ or a diesel one $l \in L_D$ and two specific tracks: one $b \in B_P$ and one $b \in B_T$.

Suppose to have to manage a job j_{a_1} requiring the primary and secondary operations. The order of the operations is given, together with their duration.

SM decides the schedule of these operations (here named o_P and o_{S_1}) on the job j_{a_1}, taking into account also the others jobs to manage. For job j_{a_1} he has to define the starting time of the first operation to execute on the job, $(s_{j_{a_1},MP})$, while $e_{j_{a_1},MP}$ the ending time is a consequence. Ending the first operation, the second can start. The starting time of the second operation is $s_{j_{a_1},M_{s_1}}$ while $e_{j_{a_1},M_{S_1}}$ indicates the ending time.

Figure 3 shows the scheduling of the operations together with the used resources. Suppose to have the following available resources: three shunting teams (two general $sq \in SQ_G$ and one specific $sq \in SQ_S$), four locomotives (two electric $l \in L_E$ and two diesel $l \in L_D$) and nine tracks (three tracks of entrance $b \in B_E$, two tracks of the shunting park $b \in B_P$ and four terminal tracks $b \in B_T$). Looking at Fig. 3, operation MP will be performed with the generic shunting team $sq_1 \in SQ_G$, the electric locomotive $l_1 \in L_E$, using the entrance track $b_2 \in B_E$ from the station outside the port area to the track $b_4 \in B_P$ of the shunting park area while operation o_{S_1} with the general shunting team $sq_2 \in SQ_G$, the same electric locomotive

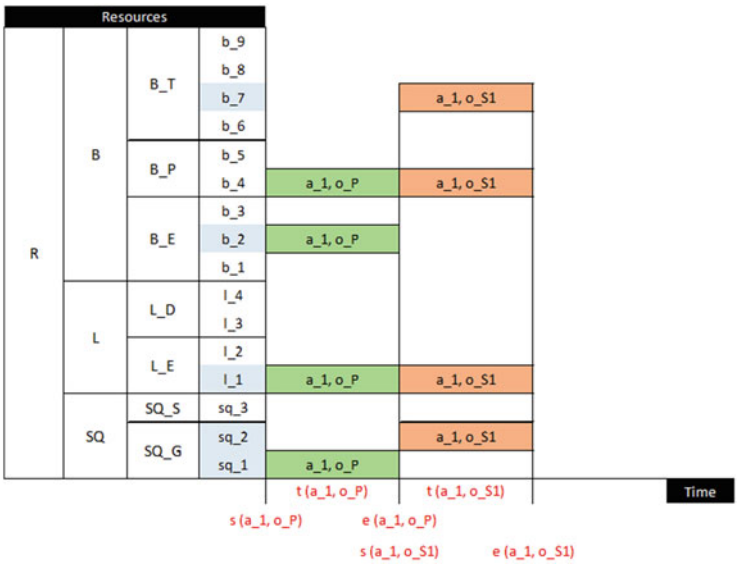


Fig. 3 Resources scheduling for two operations on job j_{a_1}

$l_1 \in L_E$ and the tracks $b_4 \in B_P$, of the shunting park, and $b_7 \in B_T$, of the port terminal area.

Every day SM needs to (re-) schedule on average a lot of operations for a certain number of trains (jobs) passing through the shunting area. A decision support system can be helpful. We need a good formulation for facing and solving quickly this difficult problem. In the next section different formulations proposed in the literature are discussed. Even if, probably its more natural formulation is as job shop, due to the presence of many limited resources to use we propose a network flow formulation based on a space-time network.

3.2 (Re-)scheduling Problems Literature Overview

In general, the railway system is based on a timetable which consists on a set of train's trips. At the beginning, when the activities' scheduling is built, the timetable and the resources duties are without conflicts, but in the real time operations problems are unavoidable. Conflicts can occur for delays due to disturbances or disruptions, thus the railway system's timetable has to be re-scheduled. This problem in literature is called Train Timetable Rescheduling (TTR). The aim of TTR is to quickly re-obtain a feasible timetable of sufficient quality [4]. TTR has been approached with a no-wait job shop scheduling model. In [13] the job-shop scheduling problem with blocking and/or no-wait constraints is addressed; the problem was formulated by means of an alternative graph, a suitable model for this job shop problem, also able to include several real-world constraints. The authors of [13] consider a set of operations which have to be performed on fixed machines while in the problem under inspection we have flexible machines linked to different resources. Also Pacciarelli et al. [15] show how the alternative graph formulation is able to represent in details the train scheduling problem. Differently from our problem, tracks are the only resources taken into account and the speed is an element used to avoid conflicts.

Sometimes TTR is split into a re-timing and a re-routing problem, as in [5] that describes a number of algorithmic improvements implemented in the real-time traffic management system ROMA (Railway traffic Optimization by Means of Alternative graphs), achieved by incorporating effective re-scheduling algorithms and local re-routing strategies in a tabu search scheme.

Then, in [2] is proposed a hybrid approach for solving the resource-constrained project scheduling problem which is an extremely hard combinatorial optimization problem of practical relevance. Jobs have to be scheduled on resources subject to precedence constraints such that the resource capacities are never exceeded and the latest completion time of all jobs is minimized. The specific relations of precedence between the activities used in [2] are not so strictly in our problem.

Meng and Zhou [14] develop an innovative integer programming model based on a time-space network for the problem of train dispatching on N-track network by means of simultaneously re-routing and re-scheduling trains. An integer program-

ming formulation for the train dispatching problem based on a space-time network is also proposed in [16]. These works consider only tracks as resources capacity constraints.

The problem addressed in the present work is focused on the re-scheduling of the shunting operations. In general, this is different from the previous problems since the main decisions are related to the starting time of the operations on the jobs, and the feasibility is strongly linked to the resources that must be used for executing activities. It is closer to the problem of re-scheduling activities either in a station or in a depot.

For station shunting scheduling problems, Tomii and Zhou [18] propose an algorithm combining probabilistic local search and PERT (Program Evaluation and Review Technique).

Depot shunting scheduling problems, however, are again more complicated compared with the station shunting scheduling problems with regard to a larger number of task types, workforce types and more complicated criteria. Tomii and Zhou [18] consider to adopt the genetic algorithms (GA) but the search space becomes too large because in solving the depot shunting scheduling problem, exact timings for the start and end of each task have to be decided. To solve the problem, the authors propose to combine the genetic algorithm with PERT. They introduce a two-stage algorithm, in which a rough shunting schedule without explicitly considering the timings of each task is produced by GA, and the timings of tasks and shunting are decided by using the PERT technique.

3.3 A Time-Space Network Model for Scheduling Shunting Operations

Inspired by Meng and Zhou [14], a time-space network for a simple case of the problem described in Sect. 3.1 is investigated. In particular, a shunting area connected to a terminal, and a set A of trains approaching the shunting area for executing some operations (primary shunt, secondary shunt or unique shunt) are considered.

Given a planning horizon T split into s time periods ($T = \{1, 2, s\}$) and the operations to execute on the trains, the problem consists in determining the starting and ending time of each operation to execute in the shunting area, for transferring the trains approaching the area in ao_j, and having to be inside the terminal respecting their time of arrival ad_j.

The set A of trains include a set of jobs (trains) requiring the unique shunting (A^U) and trains requiring the other operations ($A = A^U \cup A^{NU}$). The set of shunting operations M is here limited to the primary, the secondary and the unique shunting.

It is known the required time for executing each shunting operation m on each job j ($t_{j,m}$) $m \in M$, $j \in A$. We denote $i_{j,m}$ and $f_{j,m}$ the starting and the ending time of each operation m for each job j.

The shunting manager has a limited number of teams (sq) for realizing all the activities and there is also a capacity for the number of operations that can be realized in each period t: let MP_{max}, MS_{max} and MU_{max} be the maximum number of primary, secondary and unique shunting operations that can be realized in t, respectively. Also the number of trucks in the station and in the park is limited, so S_{max} and P_{max} are the maximum number of trains that can wait in the rail station and in the park, respectively.

For solving the above described problem we define a spatial-time network $G = (V, E)$ as follows.

The *set of nodes* is given by the union of different subsets:

The origin/destination nodes (O, D) that represent the arrival of each job to the shunting area and its exit (that correspond to its entrance in the terminal)

$$O = \{O_j | j \in A\}$$
$$D = \{D_j | j \in A\}$$

The operational nodes used for representing both the different zones in the shunting area (i.e. the station S, the park P), and the shunting operations (i.e. MP, MS, MU). These nodes are replicated for each period of time in T, thus are defined as follows:

$$S = \{S_t | t \in T\}$$
$$MP = \{MP_t | t \in T\}$$
$$P = \{P_t | t \in T\}$$
$$MU = \{MU_t | t \in T\}$$
$$MS = \{MS_t | t \in T\}$$

The *set of arcs* is the union of the subsets of arcs used for representing the arrival and the departure of each train in the shunting area, the starting and ending of each operation, the time spent by a job either to execute a shunting operation or to wait to start it, in the station or in the park. They are defined as follows:

$$A_{E1} = \{(O_j, S_t) | j \in A, t \in T, t \geq ao_j\}$$
$$A_{E2} = \{(S_t, MP_t) | t \in T\}$$
$$A_{E3} = \{(S_t, MU_t) | t \in T\}$$
$$A_{E4} = \{(MP_t, P_t) | t \in T \; s.t. \; t > 1\}$$
$$A_{E5} = \{(P_t, MS_t) | t \in T\}$$
$$A_{E6} = \{(MS_t, D_j) | t = ad_j, j \in A^{NU}\}$$
$$A_{E7} = \{(MU_t, D_j) | t = ad_j, j \in A^{U}\}$$
$$A_{T1} = \{(S_t, S_{t+1}) | 1 \leq t \leq s - 1\}$$
$$A_{T2} = \{(MP_t, MP_{t+1}) | 1 \leq t \leq s - 1\}$$
$$A_{T4} = \{(P_t, P_{t+1}) | 1 \leq t \leq s - 1\}$$
$$A_{T6} = \{(MU_t, MU_{t+1}) | 1 \leq t \leq s - 1\}$$
$$A_{T7} = \{(MS_t, MS_{t+1}) | 1 \leq t \leq s - 1\}$$

The network flow model uses the following *decision variables*:

AR_{ik}^{j}, $j \in A$, $(ik) \in A_{E1}$ s.t. $i = O_j$, indicating the arrival of job j to the station;

SOS_{ik}^{j}, $j \in A, (ik) \in A_{T1}$ s.t. $i = S_t | t \geq ao_j$, indicating the jobs waiting at the station;
IMP_{ik}^{j}, $j \in A^{NU}, (ik) \in A_{E2}$ s.t. $i = S_t | t \geq ao_j$, indicating that job j starts the primary shunting;
MAP_{ik}^{j}, $j \in A^{NU}, (ik) \in A_{T2}$ s.t. $i = MP_t | t \geq ao_j$, indicating that job j is executing the primary shunting;
FMP_{ik}^{j}, $j \in A^{NU}, (ik) \in A_{E4}$ s.t. $i = MP_t | t \geq ad_j$, indicating that job j ends the primary shunting;
PAR_{ik}^{j}, $j \in A^{NU}, (ik) \in A_{T4}$ s.t. $i = P_t | t \geq ao_j + t_{j,m=MP}, t \leq ad_j$, indicating the jobs waiting in the park;
IMS_{ik}^{j}, $j \in A^{NU}, (ik) \in A_{E5}$ s.t. $i = P_t | t \geq ao_j + t_{j,m=MP}, t \leq ad_j$, indicating that job j starts the secondary shunting;
MAS_{ik}^{j}, $j \in A^{NU}, (ik) \in A_{T7}$ s.t. $i = MS_t | t \geq ao_j + t_{j,m=MP}, t \leq ad_j$, indicating that job j is executing the secondary shunting;
FMS_{ik}^{j}, $j \in A^{NU}, (ik) \in A_{E6}$ s.t. $i = MS_t | t \geq ad_j$, indicating that job j ends the secondary shunting;
IMU_{ik}^{j}, $j \in A^{U}, (ik) \in A_{E3}$ s.t. $i = S_t | ao_j \leq t \leq ad_j$, indicating that job j starts the unique shunting;
MAU_{ik}^{j}, $j \in A^{U}, (ik) \in A_{T6}$ s.t. $i = MU_t | ao_j \leq t \leq ad_j$, indicating that job j is executing the unique shunting;
FMU_{ik}^{j}, $j \in A^{U}, (ik) \in A_{E7}$ s.t. $i = MU_t | t \geq ad_j$, indicating that job j ends the unique shunting;
$MP^t \geq 0$, number of jobs executing the primary operation in t;
$MS^t \geq 0$, number of jobs executing the secondary operation in t;
$MU^t \geq 0$, number of jobs executing the unique operation in t;
$fMS_{j,m}$, $j \in A^{NU}, m \in Ms.t.\ m = MS$;
$fMU_{j,m}$, $j \in A^{U}, m \in Ms.t.\ m = MU$;
$dfMS_{j,m}$, $j \in A^{NU}, m \in Ms.t.\ m = MS$;
$dfMU_{j,m}$, $j \in A^{U}, m \in Ms.t.\ m = MU$.

Having to minimize the difference between the expected time of entrance of the jobs and their real time of entrance, the *objective function* is the following:

$$\text{MIN } dfM = \sum_{j \in A^{NU}} dfMS_{j,m=MS} + \sum_{j \in A^{U}} dfMU_{j,m=MU}$$

In this network model the constraints are the following:
O-D constraints

$$AR_{O_j, S_{t=ao_j}}^{j} = 1 \quad \forall j \in A \tag{1}$$

$$\sum_{t \geq ad_j}^{s} FMS_{MS_t, D_j}^{j} = 1 \ \forall j \in A^{NU} \tag{2}$$

$$\sum_{t \geq ad_j}^{s} FMU^{j}_{MU_t, D_j} = 1 \quad \forall j \in A^U \tag{3}$$

flow conservation constraints

$$AR^{j}_{O_j, S_t} + SOS^{j}_{S_{t-1}, S_t} = IMP^{j}_{S_t, MP_t} + IMU^{j}_{S_t, MU_t} + SOS^{j}_{S_t, S_{t+1}} \quad \forall t \in T, \forall j \in A \tag{4}$$

$$IMP^{j}_{S_t, MP_t} + MAP^{j}_{MP_{t-1}, MP_t} = MAP^{j}_{MP_t, MP_{t+1}} + FMP^{j}_{MP_t, P_t} \ \forall t \in T, \forall j \in A^{NU} \tag{5}$$

$$PAR^{j}_{P_{t-1}, P_t} + FMP^{j}_{MP_t, P_t} = IMS^{j}_{P_t, MS_t} + PAR^{j}_{P_t, P_{t+1}} \ \forall t \in T, \forall j \in A^{NU} \tag{6}$$

$$IMU^{j}_{S_t, MU_t} + MAU^{j}_{MU_{t-1}, MU_t} = MAU^{j}_{MU_t, MU_{t+1}} + FMU^{j}_{MU_t, DT_j} \quad \forall t \in T, \forall j \in A^U \tag{7}$$

$$IMS^{j}_{P_t, MS_t} + MAS^{j}_{MS_{t-1}, MS_t} = MAS^{j}_{MS_t, MS_{t+1}} + FMS^{j}_{MS_t, DT_j} \ \forall t \in T, \forall j \in A^{NU} \tag{8}$$

capacity constraints

$$\sum_{j \in A^{NU}} IMP^{j}_{S_t, MP_t} + \sum_{j \in A^U} IMU^{j}_{S_t, MU_t} + \sum_{j \in A^{NU}} SOS^{j}_{P_t, MS_t} <= S_{max} \quad \forall t \in T \tag{9}$$

$$\sum_{j \in A^{NU}} IMS^{j}_{P_t, MS_t} + \sum_{j \in A^{NU}} PAR^{j}_{P_t, P_{t+1}} <= P_{max} \ \forall t \in T \tag{10}$$

$$\sum_{j \in A^{NU}} MAP^{j}_{MP_t, MP_{t+1}} + \sum_{j \in A^{NU}} FMP^{j}_{MP_t, P_t} <= MP_{max} \ \forall t \in T \tag{11}$$

$$\sum_{j \in A^{NU}} MAS^{j}_{MS_t, MS_{t+1}} + \sum_{j \in ADT^{NU}} FMS^{j}_{MS_t, D_j} <= MS_{max} \ \forall t \in T \tag{12}$$

$$\sum_{j \in A^U} MAU^{j}_{MU_t, MU_{t+1}} + \sum_{j \in A^U} FMU^{j}_{MU_t, D_j} <= MU_{max} \ \forall t \in T \tag{13}$$

$$\sum_{j \in A^{NU}} MAP^{j}_{MP_t, MP_{t+1}} + \sum_{j \in A^{NU}} FMP^{j}_{MP_t, P_t} = MP^t \ \forall t \in T \tag{14}$$

$$\sum_{j \in A^{NU}} MAS^{j}_{MS_t, MS_{t+1}} + \sum_{j \in A^{NU}} FMS^{j}_{MS_t, D_j} = MS^t \ \forall t \in T \tag{15}$$

$$\sum_{j \in A^U} MAU^{j}_{MU_t, MU_{t+1}} + \sum_{j \in A^U} FMU^{j}_{MU_t, D_j} = MU^t \ \forall t \in T \tag{16}$$

$$MP^t + MS^t + MU^t <= sq \ \forall t \in T \tag{17}$$

constraints defining the duration of each operation:

$$\sum_{t \in T} MAP^{j}_{MP_t, MP_{t+1}} = t_{j,m} \ \forall j \in A^{NU},\ m = MP \tag{18}$$

$$\sum_{t \in T} MAS^{j}_{MS_t, MS_{t+1}} = t_{j,m} \ \forall j \in R^{NU},\ m = MS \tag{19}$$

$$\sum_{t \in T} MAU^{j}_{MU_t, MU_{t+1}} = t_{j,m} \ \ \forall j \in A^{U},\ \ m = MU \tag{20}$$

constraints defining the end of the unique and the secondary shunt and the deviation with respect to the expected time of entrance in the terminal:

$$fMS_{j,m} = \sum_{t \in T} FMS^{j}_{MS_t, D_j} * t \ \forall j \in A^{NU}, m = MS \tag{21}$$

$$fMU_{j,m} = \sum_{t \in T} FMU^{j}_{MU_t, D_t} * t \ \ \forall j \in A^{U}, m = MU \tag{22}$$

$$dfMS_{j,m} = fMS_{j,m} - ad_j \ \forall j \in A^{NU}, m = MS \tag{23}$$

$$dfMU_{j,m} = fMU_{j,m} - ad_j \ \ \forall j \in A^{U}, m = MU \tag{24}$$

3.4 Preliminary Results

Just to test the previous network flow model and its capability to solve the problem under investigation, three different instances have been considered, with 15, 20 and 25 jobs to execute in 1 day split into time periods of 10 min. The model size ranges from 6832 to 13,578 variables and from 5713 to 10,446 constraints. The CPU time for the larger instance is 37.82 s.

In Fig. 4 an example of a solution obtained by solving the flow model with Gurobi 7.5.1 for MPL [nn] is reported: in black the reader can note the path of a job from its entrance to its arrival to the considered terminal. The horizontal sections in the figure represent the different areas in which the job has to pass through. The analyzed train enters the shunting area and immediately starts the primary shunt; then, the train waits many time periods before starting the secondary shunting for, finally, entering in the terminal in the exact due time.

Figure 5 represents the flow related to the same train in the space-time network.

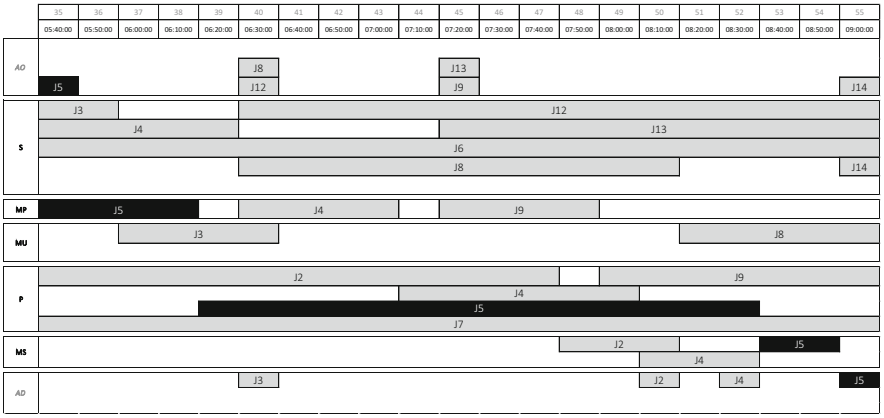


Fig. 4 Focus on a train through the shunting area

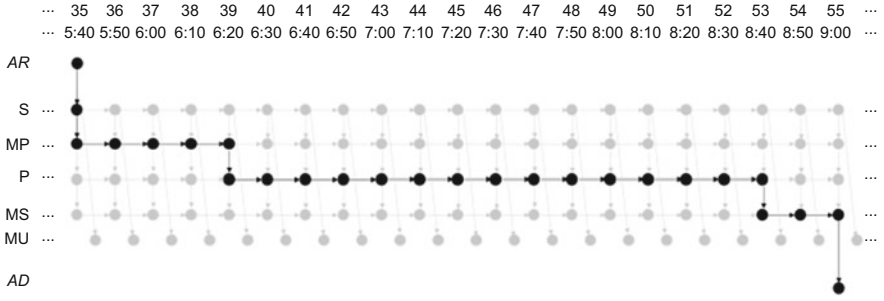


Fig. 5 The train's path on the space-time network

4 Conclusions

Trying to help the rail freight transport mode to became more competitive and more used in the intermodal transport, the rail transport process and the port rail shunting (re)-scheduling problem have been addressed. A first approach for solving the rail shunting scheduling problem has been presented. Due to the complex relations among jobs, operations and resources (different kind of resources must be linked to the operations to schedule) we have decided to propose a formulation based on a time-space network (able to manage both the schedule of the activities and the limited resources).

References

1. Ambrosino, D., Siri, S.: Comparison of solution approaches for the train load planning problem in seaport terminals. Transp. Res. E Logist. Transp. Rev. **79**, 65–82 (2015)
2. Berthold, T., Heinz, S., Lübbecke, M.E., Möhring, R.H., Schulz, J.: A constraint integer programming approach for resource-constrained project scheduling. In: International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming, pp. 313–317 (2010)
3. Bohlin, M., Hansmann, R., Zimmermann, U.T.: Optimization of railway freight shunting. Computers & industrial engineering. In: Borndörfer, R., Klug, T., Lamorgese, L., Mannino, C., Reuther, M., Schlechte, T. (eds.) Handbook of Optimization in the Railway Industry. International Series in Operations Research & Management Science, vol. 268. Springer, Cham (2018)
4. Cacchiani, V., Huisman, D., Kidd, M., Kroon, L., Toth, P., Veelenturf, L., Wagenaar, J.: An overview of recovery models and algorithms for real-time railway rescheduling. Transp. Res. B Methodol. **63**, 15–37 (2014)
5. Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M.: A tabu search algorithm for rerouting trains during rail operations. Transp. Res. B Methodol. **44**, 175–192 (2010)
6. Güven, C., Eliiyi, D.T.: Trip allocation and stacking policies at a container terminal. Transp. Res. Proc. **3**, 565–573 (2014)
7. Hanaoka, S., Regmi, M.B.: Promoting intermodal freight transport through the development of dry ports in Asia: an environmental perspective. Iatss Res. **35**, 16–23 (2011)
8. Jaehn, F., Michaelis, S.: Shunting of trains in succeeding yards. Comput. Ind. Eng. **102**, 1–9 (2016)
9. Legato, P., Mazza, R.M.: A decision support system for integrated container handling in a transshipment hub. Decis. Support Syst. **108**, 45–56 (2018)
10. Lehnfeld, J., Knust, S.: Loading, unloading and premarshalling of stacks in storage areas: survey and classification. Eur. J. Oper. Res. **239**, 297–312 (2014)
11. Li, L., Negenborn, R.R., De Schutter, B.: Intermodal freight transport planning – a receding horizon control approach. Transp. Res. C Emerg. Technol. **60**, 77–95 (2015)
12. Macharis, C., Bontekoning, Y.M.: Opportunities for OR in intermodal freight transport research: a review. Eur. J. Oper. Res. **153**, 400–416 (2004)
13. Mascis, A., Pacciarelli, D.: Job-shop scheduling with blocking and no-wait constraints. Eur. J. Oper. Res. **143**, 498–517 (2002)
14. Meng, L., Zhou, X.: Simultaneous train rerouting and rescheduling on an N-track network: a model reformulation with network-based cumulative flow variables. Transp. Res. B Methodol. **67**, 208–234 (2014)
15. Pacciarelli, D., Mascis, A., Pranzo, M.: Scheduling models for short-term railway traffic optimization. In: Proceedings of the 9th International Conference on CASPT, San Diego, USA (2004)
16. Şahin, G., Ahuja, R.K., Cunha, C.B.: Integer programming based solution approaches for the train dispatching problem. Tech. Rep., ID: SV_FENS_2011/0002. Sabanci University, Istanbul (2010)
17. Stahlbock, R., Voß, S.: Operations research at container terminals: a literature update. OR Spectr. **30**, 1–52 (2008)
18. Tomii, N., Zhou, L.J.: Depot shunting scheduling using combined genetic algorithm and PERT. In: Allen, J., et al. (eds.) Computers in Railways VII, pp. 437–446. WIT Press, Southampton (2000)
19. Wang, J., Lin, B., Jin, J.:Optimizing the shunting schedule of electric multiple units depot using an enhanced particle swarm optimization algorithm. Comput. Intell. Neurosci. **2016**, 1–11 (2016)

The k-Color Shortest Path Problem

Daniele Ferone, Paola Festa, and Tommaso Pastore

Abstract This paper proposes a mathematical model and an exact algorithm for a novel problem, the k-Color Shortest Path Problem. This problem is defined on a edge-colored weighted graph, and its aim is to find a shortest path that uses at most k different edge-colors. The main support and motivation for this problem arise in the field of transmission networks design, where two crucial matters, reliability and cost, can be addressed using both colors and arc distances in the solution of a constrained shortest path problem. In this work, we describe a first mathematical formulation of the problem of interest and present an exact solution approach based on a branch and bound technique.

Keywords Constrained path problem · Edge-colored graph · Branch and bound

1 Introduction

Shortest Path Problems (SPPs) represent one of the most significant and investigated family of problems in Operations Research. The formulations that describe members of this class are often intuitive and easily relatable to real-world scenarios, while their broad applicability implies that often SPPs need to be solved as sub-tasks

D. Ferone
Department of Mechanical, Energy and Management Engineering, University of Calabria, Rende, Italy
e-mail: daniele.ferone@unical.it

P. Festa
Department of Mathematics and Applications, University of Naples Federico II, Naples, Italy
e-mail: paola.festa@unina.it

T. Pastore (✉)
Department of Structures for Engineering and Architecture, University of Naples Federico II, Naples, Italy
e-mail: tommaso.pastore@unina.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_32

in many other combinatorial problems, such as Maximum-Flow Minimum-Cost Problems or Vehicle Routing Problems.

While notably the classical SPPs can be optimally solved using polynomial algorithms, a more recent stream of research focused on the solution of either dynamic [12, 13] or constrained-SPPs [11, 14, 20], studying both exact and heuristic techniques. In this work, we define and formally state a novel problem: the k-Color Shortest Path Problem (k-CSPP), whose objective is to find a shortest path in a weighted edge-colored network, with a constraint on the maximum number of different colors that can be used. In the following we will use interchangeably the terms "colors" and "labels".

Edge-colored networks received a fair share of attention in the scientific literature, given their aptness for the depiction of complex and diverse relations among nodes. This feature proved to be beneficial in a wide variety of application fields, such as: computational biology [9], telecommunications [23], as well as in the analysis of transportation networks [1], and conflicts resolution [21].

In [4], it is proven that to find a path from a source node s to a destination node t with maximum k colors is a **NP**-complete problem, by reduction from the 3-SAT problem. We observe how any instance of the decisional problem of finding an $s-t$ path with at most k colors can be represented as an instance of k-CSPP, where each edge has null cost. Therefore, a polynomial algorithm for the k-CSPP would efficiently solve the decisional problem described in [4]. Consequently, k-CSPP is **NP**-hard.

In the study of edge-colored graphs, many works—of both theoretical and experimental interest—are concerned with the investigation of specific properly-colored edge structures, where a coloring is said to be proper whenever any two adjacent edges differ in color. These structures include for example: paths, trails, trees and cycles; see for example [17]. On the other hand, some classical optimization problems—such as the Minimum Spanning Tree (MST), the Traveling Salesman Problem (TSP), and the Longest Path Problem (LPP)—have all been extended to the case of edge-colored graphs, taking labels into account either in the objective function or in their constraints.

For example, the Minimum Label Spanning Tree is defined in [8] as a variant of the classical MST in which the cost of the spanning tree is given by the number of different edge-labels used. The authors of [8] describe a heuristic technique and an exact method based on the A* algorithm. The problem is further investigated in [6] which present a logarithmic approximation algorithm and a comparative study of several metaheuristic techniques, respectively. A strictly related generalization, the k-labeled Spanning Forest Problem, is studied in [7].

In [19], Jozefowiez et al. present an in-depth analysis of the Minimum Label Hamiltonian Cycle Problem (MLHCP). The MLHCP consists in the determination of an Hamiltonian cycle that presents the minimum total number of different edge-labels used. Moreover, in [19] two variations are introduced: the MLHP with length constraints and the Traveling Salesman Problem with label constraints (LCTSP). Aim of the LCTSP is to minimize the length of the tour—as in the classical TSP—while constraining the maximum number of different colors that

can be used. The authors propose mathematical models and prove valid inequalities that are then included in branch-and-cut algorithms for the MLHCP and its two variants.

As an additional example, a special case of longest path on edge-colored graphs, the Orderly Colored Longest Path Problem, was recently studied in [5] by Carrabs et al. Suitably adapting existing formulations, the authors obtain several mathematical descriptions for the problem, that are then compared over a broad set of instances.

The main ground of interest for the k-CSPP arises in the field of telecommunications. While designing transmission networks, reliability is a crucial matter to ensure good performances and prevent data loss. As deeply discussed in [22] and [23], the robustness of a path-routed communication network can be achieved by means of path protection schemes, which make use of backup paths to ensure reachability in the case of single link failures. The backup path and the primary path are link-disjoint, and share the same source and destination. To prevent traffic loss, the backup path is activated whenever the primary path fails. On the other hand, often a single happening can cause the simultaneous failure of several links in the network. For instance, in WDM networks it is customary to bundle multiple fiber links in the same conduit. Consequently, even if these links are disjoint in the network layer, a damage to the conduit will cause the failure of all the links there bundled. The fibers sharing a common risk factor are said to be in the same Shared Risk Link Group, and are modeled with arcs of the same color. In [23], Yuan et al. address the failure minimization problem as a minimum-color path problem, in which each path is associated with one or more colors and each color is related to a given failure event. The authors support their approach arguing that by minimizing the number of colors involved in the path, the failure probability of the path can consequently be minimized. At the same time, while this argument handles different edge-labels, it does not include lengths in the comparison of different paths. Accordingly, a path with at most k different colors is connected if and only if failures do not occur in any of the k colors traversed in the path. If we make the assumption that color failures are mutually independent, and equiprobable—with probability $p \in [0, 1]$—, then the reliability of the path can be computed as $(1 - p)^k$. Consequently, an upper bound on the number of different colors allows to have a probabilistic estimate on the reliability of the network. A similar argument can be repeated in the case of independent failure events with different probabilities.

With in mind a similar network reliability scenario, the k-CSPP handles risk adversity as a strict requirement, while optimizing path length. Hence, the mathematical models introduced in the present paper include distances in the objective function, while encompassing the use of few colors in a problem constraint.

The main contributions of this work are: (1) a first formal description of the k-CSPP and (2) the design of a branch and bound algorithm. The work is organized as follows: in Sect. 2 some related works on Constrained Shortest Path Problems are presented. The problem is formally introduced in Sect. 3, where a mathematical model is proposed. Section 4 describes the Branch and Bound implementation used to optimally solve the problem. Computational results and a comparative analysis of the performances of our Branch and Bound with respect to the CPLEX solver are presented in Sect. 5. Concluding remarks are given in Sect. 6.

2 A Brief Taxonomy of Shortest Path Problems with Edge Constraints

The goal of this section is to provide a brief classification of some Shortest Path Problem variants that include edge constraints, with the aim of pointing out their differences with the k-CSPP.

One of the most broad and notable classes of edge-constrained SPPs is given by Resource Constrained Shortest Path Problems (RCSPPs) [3, 20].

In RCSPPs, in addition to the customary directed graph $G = (V, E)$ and edge-distance function $d_{ij} : E \rightarrow \mathbb{R}_0^+$, a L-dimensional vector of resources R is defined. Essentially, each resource is related to relevant link attributes that needs to be accounted in the planning of the path. Indeed, for each $l = 1, \ldots, L$, to each $(i, j) \in E$ is associated a resource attribute r_{ij}^l. Accordingly, a path P^* is optimal whenever it is minimal w.r.t. the distance function d, and satisfies the restrictions enforced on the resources r_{ij}^l.

As argued in [2], the resources and the subsequent constraints can be of multifold nature. In fact, resources can model both numerical (either cumulative or non-cumulative) and categorical (or index) attributes. Straightforward examples of cumulative numerical attributes are travel time or fuel consumption, whose total use in a path P is obtained adding up all the travel times or consumption along the edges belonging to P, as in [10].

On the other hand, road width is an example of numerical non-cumulative parameter; in cases like this the feasibility of a path P can be subject to average or bottleneck restrictions, in which either the average of the resource or its minimum (respectively maximum) has to respect some bounds. Finally, RCSPPs can include categorical attributes, that can be used to specify the type of connection among two nodes. Whenever the problem considers arcs with categorical attributes, the formulation can feature constraints such as: a feasible path cannot contain links whose attribute is equal to a specific value (or a set of specific values). For a more detailed discussion see [2, 18].

The fundamental difference between the k-CSPP and RCSPPs lies in the fact that the k-CSPP does not restricts color-sets a priori and there is a strong interdependence among arcs. In our scenario, indeed, the cost of a color as a resource is not constant during the exploration of the solution space: once that an arc with a certain color is traversed, all other arcs sharing the same colors turn free and thus can be inserted in the solution without placing additional burden on the color constraint.

Another related idea studied in path problems on edge colored graphs consists in the use of *reload costs*. For each couples of colors (b, c) a reload cost $\rho_{b,c}$ is the amount to be paid if in the path P an arc of color c is traversed after an arc of color b. Gourvès et al. [16] studied the problem of minimizing reload costs for walks, trails and paths, deriving the resulting computational complexities. On the other hand, [1] consider a general form of objective function that includes both distances and reload cost. Aside from their presence in the objective function rather than in the constraints, the main difference between reload costs and the modeling paradigm

of the k-CSPP is that reload costs are fixed, and have to be taken into account any given time there is a change from a color to another. On the contrary, bounding the maximum number of different colors, as required in the k-CSPP, means to count just once a transition to a specific color, whatever the preceding color (if any) may be.

3 Mathematical Model

Let $G = (V, E)$ be an undirected graph, with n nodes and m edges. Let the functions $C : E \longrightarrow \mathbb{N}$ and $d : E \longrightarrow \mathbb{R}_0^+$ be an edge-coloring and a non-negative distance function defined on E, respectively. The positive integer $C(e)$, $\forall e \in E$, is said to be the *color* of edge e.

Let $c(G)$ be the number of colors used to label the edges of G, for each color $h \in \{1, \ldots c(G)\}$ all edges labeled with h are collected in color class C_h, in a way that $E = \bigcup_{h=1}^{c(G)} C_h$.

The k-CSPP consists in finding a shortest path $P^* = (v_1, v_2, \ldots, v_h)$ from a source node $v_1 = s$ to a destination node $v_h = t$, with $s, t \in V$, such that the number of different colors traversed in the path does not exceed k.

Introducing a Boolean decision variable x_{ij}, for each edge $[i, j] \in E$, such that:

$$x_{ij} = \begin{cases} 1, & \text{if } [i, j] \text{ belongs to } P^*, \\ 0, & \text{otherwise,} \end{cases}$$

and for each possible color h, a Boolean decision variable y_h such that: $y_h = 1$, if color h is traversed in P^*, $y_h = 0$ otherwise. Then, the problem can be formulated as follows:

$$z = \min \sum_{[i,j] \in E} d_{ij} x_{ij} \tag{1a}$$

subject to:

$$\sum_{j \in V \setminus \{i\}} x_{ji} - \sum_{j \in V \setminus \{i\}} x_{ij} = b_i, \qquad \forall i \in V \tag{1b}$$

$$x_{ij} \leq y_h, \quad \forall [i, j] \in C_h, h = 1, \ldots, c(G) \tag{1c}$$

$$\sum_{h=1}^{c(G)} y_h \leq k \tag{1d}$$

$$x_{ij} \in \{0, 1\}, \qquad [i, j] \in E \tag{1e}$$

$$y_h \in \{0, 1\}, \qquad \forall h = 1, \ldots, c(G) \tag{1f}$$

with $b_i = -1$ for $i = s$, $b_i = 1$ for $i = t$, and $b_i = 0$ otherwise.

The objective function (1a) minimizes the total cost $d(p^*)$ of the solution path p^*. The constraints (1b) represent the flow balance constraint at each node. Constraints (1c) correlate arc traversal to color selection, and constraints (1d) impose the maximum number of colors that can be used in the path. Finally, the binary constraints are given in Eqs. (1e) and (1f).

4 Branch and Bound

The basic idea of the branch and bound here proposed consists in the observation that relaxing the color constraints (1d), the problem can be solved very efficiently by a classical shortest path algorithm. Consequently, at each node of the branching tree, a shortest path problem is solved on a given edge-colored graph $G' = (V, A')$. Then, if $p^*_{G'}$ is the optimal solution obtained, and $c(p^*_{G'})$ is the number of different colors used in $p^*_{G'}$, four cases can occur:

- $d(p^*_{G'}) = +\infty$: there is no path from s to d, the feasible region is empty, and the branching node becomes a leaf;
- $d(p^*_{G'}) \geq d(\hat{p})$: where $\hat{p}$ is the incumbent solution. In this case, the branching node is fathomed due to the bounding criterion;
- $c(p^*_{G'}) \leq k$: the solution is feasible for the original problem, and the incumbent is updated if necessary;
- $c(p^*_{G'}) = l > k$: the solution is not feasible for the original problem.

In the last case, a branching operation is performed. Let $C^* = \{c_1, \ldots, c_l\}$ be the colors used by the path $p^*_{G'}$, for each $i = 1, \ldots, l$ a new branching node is generated on the graph $G'' = (V, E'')$, where $E'' = E' \setminus \{e_{vw} \in E' : C(e_{vw}) = c_i\}$.

Moreover, the strategy guiding the exploration of the branching tree is a depth-first mechanism, and the nodes of the tree are generated excluding colors according to their absolute frequencies in $p^*_{G'}$. The lesser used the color, the earlier it is excluded from G'. The main target of this strategy is to obtain a feasible solution in the quickest way possible, in order exploit the bounding operation as much as possible. This aim is reflected by both the choice of the depth first strategy, and in the exclusion criterion considered for colors. The latter, indeed, tries to define a sub-problem favoring the constriction of colors that are less used in the computed path $p^*_{G'}$.

5 Experimental Results

This section presents some computational experiments designed to compare the performances of the two exact solution strategies, the Branch and Bound technique described in Sect. 4, and the model presented in Sect. 3 and solved with ILOG CPLEX 12.9. Both the algorithms were coded in C++ using the flags `-std=c++17 -O3` and compiled with `g++ 8.2`. The experiments were run on a `INTEL i5-6400@2.70 GHz` processor with 8GB of RAM. A time limit of 10 min has been used for both the algorithms.

The instances used in the experiments can be divided in two separate classes, both obtained with the a generator adapted from [15] to suitably introduce edge-colors. More specifically, the networks considered are either grid graphs or fully random networks. The total number of colors introduced in each graph amounts to either the 15% or 20% of the total number of edges.

Table 1 reports a summary of the instances included in the experimental phase. For each type, ten instances have been generated with different seeds.

The computational results obtained by our Branch and Bound (B&B), and CPLEX—executed either with depth first (df) or breadth first (bf) strategy—are reported in Tables 2 and 3 for fully random and grid graphs, respectively. More specifically, for each instance type we report: the average time spent by the algorithms solving instances of that type (avg. time), and the number of instances

Table 1 Instance parameters

Fully random graphs				Grid graphs		
Problem	Nodes	Arcs	Colors	Problem	Size	Colors
R1	75,000	750,000	112,500	G1	100 × 100	5940
R2	75,000	750,000	150,000	G2	100 × 100	7920
R3	75,000	112,500	168,750	G3	100 × 200	11,910
R4	75,000	112,500	225,000	G4	100 × 200	15,880
R5	75,000	150,000	225,000	G5	250 × 250	37,350
R6	75,000	150,000	300,000	G6	250 × 250	49,800
R7	100,000	1,000,000	150,000	G7	250 × 500	74,775
R8	100,000	1,000,000	200,000	G8	250 × 500	99,700
R9	100,000	1,500,000	225,000	G9	500 × 500	149,700
R10	100,000	1,500,000	300,000	G10	500 × 500	199,600
R11	100,000	2,000,000	300,000	G11	500 × 1000	299,550
R12	100,000	2,000,000	400,000	G12	500 × 1000	399,400
R13	125,000	1,250,000	187,500			
R14	125,000	1,250,000	250,000			
R15	125,000	1,875,000	281,250			
R16	125,000	1,875,000	375,000			
R17	125,000	2,500,000	375,000			
R18	125,000	2,500,000	375,000			

Table 2 Results on random graphs

	B&B		CPLEX (df)		CPLEX (bf)	
Instance type	Avg. time	O + F	Avg. time	O + F	Avg. time	O + F
R1	306.72	5 + 4	126.18	10 + 0	**92.43**	10 + 0
R2	308.49	5 + 4	118.66	9 + 0	**89.12**	9 + 0
R3	197.47	7 + 2	159.84	9 + 0	**158.54**	9 + 0
R4	199.00	7 + 2	210.22	8 + 0	**163.05**	9 + 0
R5	**322.67**	5 + 5	600.00	0 + 0	521.92	2 + 0
R6	**324.29**	5 + 5	600.00	0 + 0	513.47	2 + 0
R7	300.54	5 + 4	214.57	8 + 0	**123.57**	10 + 0
R8	300.62	5 + 2	289.92	6 + 0	**218.52**	8 + 0
R9	**360.75**	4 + 6	600.00	0 + 0	600.00	0 + 0
R10	**361.01**	4 + 6	600.00	0 + 0	600.00	0 + 0
R11	**121.40**	8 + 2	600.00	0 + 0	600.00	0 + 0
R12	**121.75**	8 + 2	600.00	0 + 0	600.00	0 + 0
R13	**383.90**	4 + 5	564.25	1 + 0	600.00	0 + 0
R14	**384.42**	4 + 5	557.50	1 + 0	512.84	2 + 0
R15	**420.33**	3 + 6	600.00	0 + 0	600.00	0 + 0
R16	**420.31**	3 + 6	600.00	0 + 0	600.00	0 + 0
R17	**305.85**	5 + 5	600.00	0 + 0	600.00	0 + 0
R18	**305.84**	5 + 5	600.00	0 + 0	600.00	0 + 0
Average	**302.52**		457.84		432.97	

Bold values indicate the algorithm with the shortest computational time average

of that type for which either an optimal (O) or feasible (F) solution has been found. This last information is collected in the column (O + F). Note that since each class is made up by 10 different instances, we have $O + F \leq 10$; whenever the preceding inequality its strict, then for some of the instances not even a feasible solution could be found within the time limit.

The results show how CPLEX is performing well on the smaller instances of the dataset, while the Branch and Bound approach is able to tackle larger instances, where the model becomes too large to be managed by CPLEX. For example, referring to graph types R15–R18, it is worthy to note that CPLEX can obtain just two feasible solutions—that happen to be optimal as well—within the required time limit. On the other hand, the branch and bound approach is able to obtain at least a feasible solution in 56 out of 60 total cases, guaranteeing optimality in 24.

Similarly, we can observe in Table 3 that even in the case of grid graphs our Branch and Bound outperforms CPLEX as the size of the network grows. Additionally, the results evidence higher computational times and a lower number of optimal solution found with respect to those reported in Table 2. For example, comparing the results obtained on instance classes G8 and R13—that have the same number of nodes—we can observe how the time spent by our Branch and Bound increased by 40%, and the number of optimal solution found within the time limit dropped by 50%. This behaviour is probably due to the greater sparsity of grid

Table 3 Results on grid graphs

	B&B		CPLEX (df)		CPLEX (bf)	
Instance type	Avg. time	O + F	Avg. time	O + F	Avg. time	O + F
G1	404.78	4 + 6	41.69	10 + 0	**41.05**	10 + 0
G2	403.93	4 + 6	57.15	10 + 0	**42.16**	10 + 0
G3	485.14	2 + 6	230.77	10 + 0	**200.71**	10 + 0
G4	543.64	1 + 7	230.45	10 + 0	**225.43**	9 + 0
G5	**578.72**	1 + 9	597.25	1 + 8	596.76	1 + 7
G6	**583.95**	1 + 9	587.59	2 + 7	587.55	2 + 7
G7	**532.55**	3 + 7	601.45	0 + 10	602.28	0 + 10
G8	**540.59**	2 + 8	601.28	0 + 10	602.36	0 + 10
G9	**540.24**	1 + 9	601.26	0 + 5	607.63	0 + 9
G10	**510.76**	2 + 7	600.00	0 + 0	607.07	0 + 10
G11	**540.36**	1 + 9	600.00	0 + 0	600.00	0 + 0
G12	**540.46**	1 + 9	600.00	0 + 0	600.00	0 + 0
Average	517.09		445.74		**442.75**	

Bold values indicate the algorithm with the shortest computational time average

graphs, that implies increased difficulties in the construction of a feasible k-color path.

6 Conclusions and Future Work

This paper presented a new variant of constrained shortest path problem, the k-Color Shortest Path (k-CSPP). The problem is formally described and a Branch and Bound is proposed for its solution. The performances of the exact method here described are then compared to those achieved by CPLEX in the solution of the integer programming model. The results evidence how our Branch and Bound can manage larger instances with respect to CPLEX, achieving good performances in terms of both optimal and feasible solutions found.

As future research perspectives—inspired by classical constrained shortest path literature—we plan to investigate the use of an exact dynamic programming algorithm. Additionally, given the complexity of the problem, future investigation will exploit metaheuristic techniques to quickly obtain good sub-optimal solutions. Moreover, a larger set of instances will be generated to properly assess a comparison between the methods, and to establish a shareable benchmark for future works.

References

1. Amaldi, E., Galbiati, G., Maffioli, F.: On minimum reload cost paths, tours, and flows. Networks **57**(3), 254–260 (2011)

2. Avella, P., Boccia, M., Sforza, A.: Resource constrained shortest path problems in path planning for fleet management. J. Math. Model. Algorithms **3**(1), 1–17 (2004)
3. Beasley, J.E., Christofides, N.: An algorithm for the resource constrained shortest path problem. Networks **19**(4), 379–394 (1989)
4. Broersma, H., Li, X., Woeginger, G.J., Zhang, S.: Paths and cycles in colored graphs. Australas. J. Comb. **31**, 299–312 (2005)
5. Carrabs, F., Cerulli, R., Felici, G., Singh, G.: Exact approaches for the orderly colored longest path problem: Performance comparison. Comput. Oper. Res. **101**, 275–284 (2019)
6. Cerulli, R., Fink, A., Gentili, M., Voß, S.: Metaheuristics comparison for the minimum labelling spanning tree problem. In: The Next Wave in Computing, Optimization, and Decision Technologies, pp. 93–106. Springer, New York (2005)
7. Cerulli, R., Fink, A., Gentili, M., Raiconi, A.: The k-labeled spanning forest problem. Proc. Soc. Behav. Sci. **108**, 153–163 (2014)
8. Chang, R.S., Shing-Jiuan, L.: The minimum labeling spanning trees. Inf. Process. Lett. **63**(5), 277–282 (1997)
9. Dorninger, D.: Hamiltonian circuits determining the order of chromosomes. Discrete Appl. Math. **50**(2), 159–168 (1994)
10. Feillet, D., Dejax, P., Gendreau, M., Gueguen, C.: An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems. Networks **44**(3), 216–229 (2004)
11. Ferone, D., Festa, P., Guerriero, F., Laganà, D.: The constrained shortest path tour problem. Comput. Oper. Res. **74**, 64–77 (2016)
12. Ferone, D., Festa, P., Napoletano, A., Pastore, T.: Reoptimizing shortest paths: From state of the art to new recent perspectives. In: 2016 18th International Conference on Transparent Optical Networks (ICTON), pp. 1–5. IEEE, Piscataway (2016)
13. Ferone, D., Festa, P., Napoletano, A., Pastore, T.: Shortest paths on dynamic graphs: a survey. Pesquisa Operacional **37**(3), 487–508 (2017)
14. Ferone, D., Festa, P., Guerriero, F.: An efficient exact approach for the constrained shortest path tour problem. Optim. Methods Softw., 1–20 (2019). https://doi.org/10.1080/10556788.2018.1548015
15. Festa, P., Pallottino, S.: A pseudo-random networks generator. Technical report, Department of Mathematics and Applications "R. Caccioppoli", University of Napoli FEDERICO II, Italy (2003)
16. Gourvès, L., Lyra, A., Martinhon, C., Monnot, J.: The minimum reload s–t path, trail and walk problems. Discrete Appl. Math. **158**(13), 1404–1417 (2010)
17. Gourvès, L., Lyra, A., Martinhon, C.A., Monnot, J.: Complexity of trails, paths and circuits in arc-colored digraphs. Discrete Appl. Math. **161**(6), 819–828 (2013)
18. Irnich, S.: Resource extension functions: properties, inversion, and generalization to segments. OR Spectr. **30**(1), 113–148 (2008)
19. Jozefowiez, N., Laporte, G., Semet, F.: A branch-and-cut algorithm for the minimum labeling hamiltonian cycle problem and two variants. Comput. Oper. Res. **38**(11), 1534–1542 (2011)
20. Pugliese, L.D.P., Guerriero, F.: A survey of resource constrained shortest path problems: exact solution approaches. Networks **62**(3), 183–200 (2013)
21. Xu, H., Kilgour, D.M., Hipel, K.W., Kemkes, G.: Using matrices to link conflict evolution and resolution in a graph model. Eur. J. Oper. Res. **207**(1), 318–329 (2010)
22. Yuan, S., Jue, J.P., et al.: Shared protection routing algorithm for optical network. Opt. Netw. Mag. **3**(3), 32–39 (2002)
23. Yuan, S., Varma, S., Jue, J.P.: Minimum-color path problems for reliability in mesh networks. In: IEEE INFOCOM, vol. 4, p. 2658 (2005)

The Traveling Repairman Problem App for Mobile Phones: A Case on Perishable Product Delivery

M. E. Bruni, M. Forte, A. Scarlato, and P. Beraldi

Abstract Delivering perishable food as soon as possible has been always a challenge for producers, amplified in recent years, by a more and more competitive global market. The problem can be tackled as a routing problem with consideration of the arrival time at the customers' location, taking into account the perishability of the products planned to be delivered. Since in real-world applications customers' requests dynamically arrive during the execution of the transportation process, building vehicle routes in an on-going fashion is a challenge to be addressed. This paper describes a mobile solution that heavily relies on the use of mobile phones and integrates a well known heuristic method for the problem at hand. A case concerning the delivery of perishable food to a set of restaurants will serve as a base for illustrating the potential benefits of such a system.

Keywords Traveling repairman problem · Mobile app · Perishable product

1 Introduction

Delivering perishable food as soon as possible has been always a concern for producers and carriers. For many products that have short life span the element of time is the biggest challenge faced by these companies. In order to gain a competitive advantage over the competitors, the food delivery companies should deliver food within short lead times, accurately and with acceptable quality. Notwithstanding this has been always a challenge for producers, it has been amplified, over recent years,

M. E. Bruni (✉) · P. Beraldi
Department of Mechanical, Energy and Management Engineering, University of Calabria, Rende, Cosenza, Italy
e-mail: mariaelena.bruni@unical.it; patrizia.beraldi@unical.it

M. Forte · A. Scarlato
Department of Mathematics and Computer Science, University of Calabria, Rende, Cosenza, Italy

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_33

by a more and more competitive global market. The term perishable is applied for the products that start deteriorating as soon as they are produced. In general there are two kinds of deterioration. The first one makes the product outdated after a specified time (like blood). The second type, makes the product quality unpleasant as time passes, as in the case of flowers, vegetables and foods which are labelled as highly perishable because they deteriorate significantly fast and, as a by-product, the customers are sensitive about their freshness. For instance, a grocery shop which sells vegetables definitely wants them to be as fresh as if they have been brought from the farm. It is noteworthy that the revenues of food suppliers are dependent on the condition and freshness of the products. As a matter of fact, delivering low-quality products may impose a penalty on the supplier, decreasing the collected revenue. The above-mentioned factors highlight the importance for the suppliers to organize an efficient and effective delivery in order to maximize the freshness of the products delivered to the customers and to maximize the collected profit at the same time. This entails solving a vehicle routing problem with consideration of the arrival time at the customer's location, taking into account the perishability of the products planned to be delivered. This calls for models which aim at maximizing the revenue (profit minus perishability costs). These two objectives are in conflict with each other and an appropriate trade-off is needed.

The Traveling Repairman Problem with profits (TRPP) is an extension of the Traveling Repairman Problem (TRP), where a time-dependent profit is associated with each customer and the objective is to maximize the total collected revenue, which is a decreasing function of the arrival time at the nodes. Hence, it is an appropriate modeling framework for the problem at hand.

In real-world applications, customers' requests dynamically arrive during the execution of the transportation process. Building vehicle routes in an on-going fashion, in such a way that customer requests arriving dynamically are efficiently and effectively served, is a challenge addressed in on-line routing approaches. Although dynamic routing problems and quantitative methods for on-line routing have been discussed in the scientific literature since a seminal paper of Psaraftis [19, 20], the technology required for implementing on-line routing methods is more recent. In particular, we refer to the mobile technology, which emerges as a new business process characterised by mobility, reachability, localisation and personalisation. The user of a mobile device can access networks, products and services while on the move. This is important in context like ours, where the driver must be timely informed about the next stop to approach and the dispatching centre must be informed about the driver's location and the status of the delivery. The availability of a mobile communication system alone is, however, not sufficient. It is important that the communication system is properly and friendly interfaced with the firm system and data bases. If well designed, the mobile system will help the driver in routing decisions, taking also into account real-time traffic information (up-to-date information on weather and road conditions and detours) improving the convenience,the safety and the efficiency of travel.

This paper describes a mobile business solution that heavily relies on the use of mobile phones. An advantage of such a system is its low cost and its ease of use.

Furthermore, the integration of algorithms for the solution of routing problems is relatively straightforward. A case concerning the delivery of perishable food to a set of restaurants will serve as a base for illustrating the potential benefits of such a system.

The next section gives a short overview of the related literature, Sect. 3 describes the problem and discusses in some detail the above mentioned mobile system and the implementation issues. Section 4 presents the application on perishable products delivery and screen snapshots are presented. Finally, conclusions are given in Sect. 5.

2 Related Work

Customer-centric routing problems, where the customer's satisfaction is taken into account mostly through the arrival time at the customer's location, are broadly referred to as the minimum latency problems or traveling repairman problems (TRPs). The aim is to find a tour, starting from a depot node, which minimizes the sum of the elapsed times (or latencies) to reach a given set of nodes. The problem has been extensively studied by a large number of researchers who proposed several exact and non-exact approaches. Lucena [11] and Bianco et al. [3] proposed early exact enumerative algorithms, in which lower bounds are derived using a Lagrangian relaxation. Fischetti et al. [9] proposed an enumerative algorithm that makes use of lower bounds obtained from a linear integer programming formulation. Different mixed integer programming formulations with various families of valid inequalities have been proposed in the last years [4, 8, 12, 18]. Salehipour et al. [16] first proposed a simple composite algorithm based on a Greedy Randomized Adaptive Search Procedure (GRASP) [7, 10], improved with a variable neighborhood search procedure. In [13], Mladenović et al. presented a general variable neighborhood search metaheuristic enhanced with a move evaluation procedure facilitating the update of the incumbent solution. Silva et al. [17] presented a composite multi-start metaheuristic approach consisting of a GRASP for the construction procedure, and a randomized variable neighborhood descent algorithm for the improvement phase. In [1], Avci et al. presented a new mixed-integer linear model capable of solving small size instances and a simple and effective metaheuristic algorithm which combines a GRASP for initial solution construction and Iterated Local Search (ILS) with an adaptive perturbation mechanism for solution improvement. In particular, the developed GRASP-ILS obtained the best known results for the benchmark instances. For this reason, in this paper, this heuristic has been used as solution method to solve the problem at hand.

Recently, some interesting variants of the TRP have been proposed. Nucamendi-Guillén et al. [15] proposed two new models for the capacitated version and an efficient iterated greedy procedure. For the variant with profits, namely for the TRPP, a stochastic programming model with chance constraints has been proposed

in [2]. The aim is to find the travel plan that maximizes the random revenue that can be collected with a given reliability level.

Although many researchers have studied the TRP, the literature on the multi-vehicle TRP (referred to as k-TRP) is surprisingly limited. Recently, Nucamendi-Guillén et al. [14, 15] presented an efficient new formulation, defined on a multi-level network, for the deterministic k-TRP enhanced by an iterative greedy metaheuristic. The k-TRPP under uncertainty has been recently addressed in [5, 6], where a reactive GRASP has been proposed to solve the problem.

3 Problem Description and System Implementation

The perishable product delivery is usually performed by third-party carriers. The carriers receive the orders characterized by a position, a profit (price paid by the customer) and a service time. We assume that the capacity of the vehicle is enough to carry all the orders. The objective of the driver is to find a tour that maximizes the total profit, minimizing at the same time the so called latency, that is the waiting time of the customers, which is a proxy of the product freshness. The two conflicting criteria are then considered into the same objective function in the spirit of the TRPP. Moreover, the driver has a time limit on the route duration, that should not exceed a given time-lapse, normally 4 or 5 h (a delivery tour starts in the morning and ends at midday). The driver collects the data and feeds them into a software system, which reoptimizes heuristically and in real time the route. The route is calculated and defined upstream by the server. On the basis of the objective function trade-off and for the presence of the route duration constraint, some of the customers may not be visited. The driver will then apologise for the inconvenience and assure subsequent delivery. If the customer is selected to be visited, a corresponding delivery order is triggered to the driver performing the delivery. Drivers who already started their delivery tours may then receive additional orders. After the service, the driver notifies the system about the completed delivery and eventually adds new orders. Then, the route is re-optimized by the system, possibly servicing new customers.

The app requires a series of data to work, some of them are present in the server and therefore supplied when necessary, others are supplied through the driver (also called here operator). Potential customers are represented by nodes, characterized by an Id (used only for technical purposes), a defined geographical position through latitude and longitude, a profit and an expected service time.

Figure 1 shows the system's general architecture. During the design process the system requirements were examined and three main components were identified:

- a practical interface which allows the operator to communicate with the server
- a system able to receive requests and process them in real-time
- an external service that provides information on the routes.

An Android device was chosen as the mobile technology to be used by the operator. This means that http requests can be sent to the central system and it also

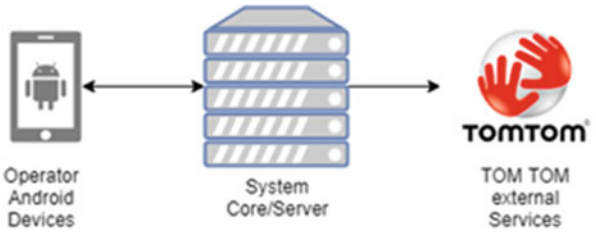


Fig. 1 System architecture

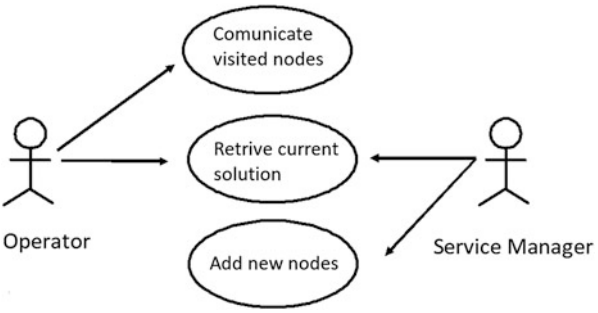


Fig. 2 Use case diagram

provides a series of services and integrations with other systems that facilitate the operator during the execution of his tasks. For the Android devices we used the latest version: 28.0.3 of the "build tool", as the "minimum sdk" we used version 15 and as "target sdk" version 26. This allows the application to be run on devices with an Android operating system greater or equal to the "Ice Cream Sandwich" version (4.0.4). There are two actors involved in the process the operator/driver and the service manager/the product supplier. The service manager is the one who uses the system to plan the work of the operator. His main interaction with the system is to add various points to an existing route. Figure 2 shows the use case diagram.

The core of the system is clearly the central server, which performs various tasks:

- it processes all (http) requests submitted to the system
- it executes the heuristic algorithm
- it queries the external services to find the data necessary for the execution of the algorithm.

A fundamental requirement for the performance of the previous tasks is the ability to access data regarding the possible route from a starting point to a destination, the distance, and above all, the time needed for the tour. This information can be obtained from different providers such as "Google Routes" or "TOM TOM Routing API". TOM TOM was chosen because it is the only one that offers a free version of the service. The only version currently usable and available of the TOM TOM

API is 1.0. All the communication take place via Web Rest in https with http basic authentication method. Server side Java 1.8 was chosen as it was one of the most mature and advanced technologies. Considering the tasks performed by the server, the Spring Boot framework was the best solution to adopt as it is excellent for the management of back-end systems with API. The framework handles all the technical aspects, allowing developers to focus on the application core instead of the technicalities. The version of Spring Boot used is 2.0.4.

The server is composed of three layers: the communication layer, the logic layer and the model layer. The functional division of the components into different levels makes each independent of each other. There is a physical dependence of the Java classes belonging to the higher levels compared to the lower level classes. The divisions of the components due to functionality and class dependence make the system highly modular and consequently improve. Some Java classes do not belong to any of the previous levels, as they do not have a functional collocation but rather only deal with technical functions of system configuration.

The communication layer, as the name suggests, deals with the external communication of the system. It has the primary task of receiving incoming requests and, once they are verified, it forwards these requests to the logical level. The classes in this level use Spring Boot annotations, turning a simple class into a restless service controller. Through the injection function, provided by the framework, each controller has a reference to the instance of the logical component to which forward requests.

The logic layer, also called “business layer” or “core layer”, is the main component of the system. All the logic of the processes is contained in this section which in turn is divided into different packages: services, algorithm and external. The services component is the entry point for requests sent from the upper level (layer communication), each request is processed in full in this section. During the processing of requests, the services component often uses the other components to deal with specific tasks, such as the generation of a new solution through the algorithm package and/or the acquisition of new data from external services through the external classes package.

The model layer is a simple container of the “objects” used by the system, this component does not contain any logical role or process.

As far a the heuristic called by the server is concerned, as already mentioned, we have implemented from scratch in Java the GRASP-ILS the heuristic proposed in [1]. The algorithm has been tailored to address the route duration constraint by adding into the objective function evaluation a term penalizing the infeasibility of the route duration constraint. We have observed that the algorithm is very fast and the solution time is almost neglectable, in agreement with the real time information that it should provide. Each time the heuristic is called the information is updated: travel times are updated with real-time traffic data of road segments and incoming requests are eventually added to the set of nodes. Using the two kinds of information previously mentioned, the GRASP-ILS algorithm is proposed to either calculate an initial route or to re-optimize the route and the server updates the visual guidance information for the driver in real-time.

The Android application has a single main screen which contains a Fragment which in turn contains a GoogleMap. From this interface the operator can perform different tasks: he can consult the solution (a series of points visited or to visit) and communicate a node already visited. The communication takes place via the http protocol and the APIs exposed by the server are RESTful type. The application contains a specific package that deals with communication, where all the remote requests are asynchronous and are managed by a special class that extends AsyncTask. During the execution of requests, the user is shown a ProgressDialog that indicates the status of the request. The interface shows the points of interest with markers and the route to follow via polylines. The operator can choose whether to display only the nodes belonging to the solution or also the nodes that have been discarded. This option can be activated through a Toggle Button. The interface also includes another button that shows, through a window, the values of the current solution: the profit collected and the total time in seconds ("HH: mm: ss" format).

4 Results

The system was tested considering 78 restaurants in Milan and a deposit (close to the Cathedral of Milan) with profit 0 and service time 0. As a server was used a Windows 10, CPU i7 quad-core, 16 GB RAM ddr4. For each node the service time was generated between 500 and 900 s. The profit of a node was calculated according to the restaurant rating (the final values of the profits lie between 10,500 and 13,500).

Clicking on the "Report" button the window with the details of the solution will be opened. In Fig. 3 the report of the initial solution is reported. Analyzing the proposed solution in more detail, the chosen nodes are 15 and the order is as follows:

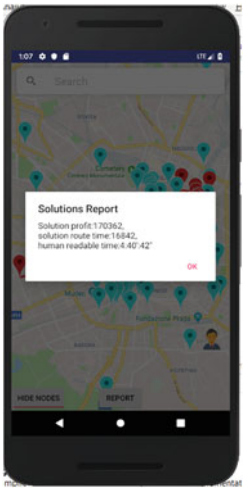


Fig. 3 Initial solution report

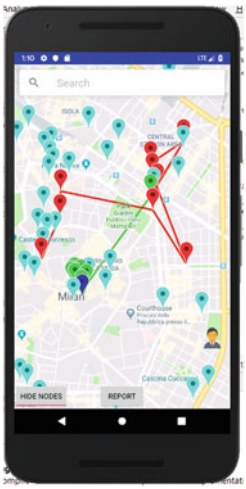


Fig. 4 Updated solution

0, 35, 74, 41, 29, 60, 26, 4, 70, 63, 33, 9, 20, 15, 54, 46. The operator starts the route. After servicing the first seven nodes, a new request arrives with a profit of 16,000, slightly higher than the average (12,000). Then the driver confirms the visit of the already visited nodes and communicates to the system the presence of this new customer. The system updates the route as shown in Fig. 4. Here, the restaurants to be visited are highlighted in red. The blue marker indicates the deposit, while the markers in light blue indicate discarded restaurants. The red line shows the route to follow. The markers and the lines that connect the nodes already visited are represented in green. In the new solution, the profit increases from 170,362 to 184,485.

5 Conclusions

This paper introduces a mobile communication system that enables route re-optimization. The paper exemplifies that today's communication and information technology allows to quickly build mobile communication systems based on easy-to-handle system components. Such systems can be used to implement mobile business processes for supporting logistic processes by means of optimisation and quantitative methods. The development, investigation and implementation of efficient and fast heuristic methods is then a complementary research area, which can be a promising future research area.

References

1. Avci, M., Avci, M.G.: A GRASP with iterated local search for the traveling repairman problem with profits. Comput. Ind. Eng. **113**, 323–332 (2017)
2. Beraldi, P., Bruni, M.E., Laganà, D., Musmanno, R.: The risk-averse traveling repairman problem with profits. Soft Comput. **23**(9), 2979–2993 (2018)
3. Bianco, L., Mingozzi, A., Ricciardelli, S.: The traveling salesman problem with cumulative costs. Networks **23**(2), 81–91 (1993)
4. Bigras, L.P., Gamache, M., Savard, G.: The time-dependent traveling salesman problem and single machine scheduling problems with sequence dependent setup times. Discrete Optim. **5**(4), 685–699 (2008)
5. Bruni, M.E., Beraldi, P., Khodaparasti, S.: A fast heuristic for routing in post-disaster humanitarian relief logistics. Transp. Res. Proc. **30**, 304–313 (2018)
6. Bruni M.E., Beraldi, P., Khodaparasti, S.: A heuristic approach for the k-traveling repairman problem with profits under uncertainty. Electron. Notes Discrete Math. **69**, 221–228 (2018)
7. Bruni, M.E., Guerriero, F., Beraldi, P.: Designing robust routes for demand-responsive transport systems. Transp. Res. E Logist. Transp. Rev. **70**(1), 1–16 (2014)
8. Ezzine, I.O., Semet, F., Chabchoub, H.: New formulations for the traveling repairman problem. In: Proceedings of the 8th International Conference of Modeling and Simulation (2010)
9. Fischetti, M., Laporte, G., Martello, S.: The delivery man problem and cumulative matroids. Oper. Res. **41**(6), 1055–1064 (1993)
10. Guerriero, F., Bruni, M.E., Greco, F.: A hybrid greedy randomized adaptive search heuristic to solve the dial-a-ride problem. Asia Pac. J. Oper. Res. **30**(1), 1250046–1250067 (2013)
11. Lucena, A.: Time-dependent traveling salesman problem-the deliveryman case. Networks **20**(6), 753–763 (1990)
12. Méndez-Díaz, I., Zabala, P., Lucena, A.: A new formulation for the traveling deliveryman problem. Discrete Appl. Math. **156**(17), 3223–3237 (2008)
13. Mladenović, N., Urošević. D., Hanafi, S.: Variable neighborhood search for the travelling deliveryman problem. 4OR **11**(1), 57–73 (2013)
14. Nucamendi-Guillén, S., Martínez-Salazar, I., Angel-Bello, F., Moreno-Vega, J.M.: A mixed integer formulation and an efficient metaheuristic procedure for the k-Travelling Repairmen Problem. J. Oper. Res. Soc. **67**(8), 1121–1134 (2016)
15. Nucamendi-Guillén, S., Angel-Bello, F., Martínez-Salazar, I., Cordero-Franco, A.E.: The cumulative capacitated vehicle routing problem: New formulations and iterated greedy algorithms. Expert Syst. Appl. **113**, 315–327 (2018)
16. Salehipour, A., Sörensen, K., Goos, P., Bräysy, O.: Efficient GRASP+ VND and GRASP + VNS metaheuristics for the traveling repairman problem. 4OR **9**(2), 189–209 (2011)
17. Silva, M.M., Subramanian, A., Vidal, T., Ochi, L.S.: A simple and effective metaheuristic for the minimum latency problem. Eur. J. Oper. Res. **221**(3), 513–20 (2012)
18. Van Eijl, C.A.: A polyhedral approach to the delivery man problem. Department of Math. and Computing Science, University of Technology (1995)
19. Psaraftis, H.N.: Dynamic vehicle routing problems. IN: Golden, B., Assad, A. (eds.) Vehicle Routing: Methods and Studies, pp. 223–248. Elsevier Science Publishers BV, Amsterdam (1988)
20. Psaraftis, H.N.: Dynamic vehicle routing: status and prospects. Ann Oper Res **61**, 143–164 (1995)

Integrating Vehicle Routing and Resource Allocation in a Pharmaceutical Network

Nicolas Zufferey and Roxanne Tison

Abstract A problem integrating resource allocation and transportation is considered here as an extension of the well-known *vehicle routing problem*. It was proposed by a pharmaceutical company, for which items (blood samples) have to be typically transported from doctor offices to laboratories (where the items are analyzed by machines) by a fleet of identical vehicles. The following specific features are considered: multiple pickups, multiple depots, multiple products, transfers, due dates, service level, and dropout level. The goal consists in minimizing the employed resource (i.e., machines, vehicles) and the traveling times of the vehicles. We propose an iterative solution method based on the following sequence of steps: build sectors, build routes, assign vehicles to routes, try to find a solution with fewer machines. Numerical experiments are reported and discussed from a managerial standpoint.

Keywords Vehicle routing · Heuristics · Resource allocation · Integrated logistics · Pharmaceutical network

1 Introduction

The project is motivated by a pharmaceutical company denoted here as $COMP$. It cannot be named because of a non-disclosure agreement. The planning horizon is a working day. During the day, items (blood samples in this study) are released at the client locations (i.e., doctor offices, clinics, hospitals) and they have then to be delivered at the destination locations (i.e., medical labs), where the items have

N. Zufferey (✉)
GSEM, University of Geneva, Geneva, Switzerland
e-mail: n.zufferey@unige.ch

R. Tison
ProcSim, Lausanne, Switzerland
e-mail: roxanne.tison@procsim.ch

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_34

to be analyzed by machines. Identical and uncapacitated vehicles (i.e., cars) are considered, and there is no parking limitation in the network. In contrast with the broad existing literature on the *vehicle routing problem* (VRP) [9, 17], and with the scarce literature on related pharmaceutical networks [2, 10], the combination of the below-listed features makes the considered problem new and very challenging. It is denoted as PVRP (for pharmaceutical VRP). In line with the increasing literature on integrated logistics [5], two fields are integrated in a common problem (here, vehicle routing and resource allocation in a transportation-production context). The considered problem is characterized by the following features.

- Multiple demands: for each client location, many items are released during the whole day.
- Multiple pickups: when an item is released at a client location, a vehicle can pick it up anytime.
- Multiple depots: there are various labs that can be delivered as often as desired.
- Service level: an item is available when it is released at its client location, then it must be possibly pre-analyzed (in a client location or in a lab), and finally, it must be analyzed (in a lab) before its due date. An item can be late, but a daily *service level* (i.e., the percentage of on-time items) has to be respected.
- Dropout level: the proportion of items that are not collected in a day should not exceed a predefined threshold. Such items are frozen for the next day.
- Multiple products: there are various item types (called *products*). In addition to its inherent chemical properties, a product is characterized by the need to be pre-analyzed or not. Each product must be analyzed by a single machine in any lab. There is a dedicated machine type for each product.
- Positioning of the vehicles: the initial position of each vehicle is known, and at the end of the day, some vehicles have to come to their initially assigned locations, which could be either a client or a lab.
- Transfers: the items can be delivered to a lab thanks to the use of different vehicles [4, 6]. The transfer of items from a vehicle to another can be performed in each location. A location for which transfers occur is called a *hub* in this paper.
- Resource optimization: use as few machines and as few vehicles as possible.

Many algorithms have been developed for the VRP and the best performance is achieved by metaheuristics [8, 12]. The considered real application for the PVRP is characterized by the following order of magnitude: 4 products, 50 clients, less than 5 labs, and 60 vehicles. There are roughly 11,000 items per day, and the time step is 1 min. As a consequence, exact methods are not appropriate for real instances, and heuristics/metaheuristics have to be employed.

The paper is organized as follows. The problem is presented accurately in Sect. 2. A solution method is proposed in Sect. 3. Experiments are discussed in Sect. 4. Conclusions and avenues of research are provided in Sect. 5.

2 Presentation of the Problem

Consider a transportation network made of clients (origins) and labs (destinations). Each client location is considered as a hub in the sense that any vehicle can transfer its load (i.e., boxes) to another vehicle. The planning horizon is a working day.

For each location of the network (i.e., a lab or a client), its assigned set of vehicles is known. All the vehicles can be considered as identical (from a technical standpoint), but there are various vehicle *types*. First, a *constrained* vehicle must start and end its day in the same location, but the return time can exceed the closing time of the involved location. A constrained vehicle belongs to the vehicle fleet of $COMP$. Second, an *external* vehicle is used by $COMP$ as a taxi service. It does not belong to the $COMP$ fleet and thus, it is not necessary to come back to its initial location at the end of the day.

The daily demand consists of thousands of items coming from the client locations. The same client can be visited (by any vehicle) several times during the day, and the item release times are known in advance (i.e., before the planning horizon). Any vehicle can be used to pickup items and deliver them to a lab, and capacity constraints can be ignored (indeed, small boxes of items are involved). For each lab, the following information is given: its specific opening hours, its assigned set of machine types, and the number of machines available for each machine type.

Anytime a driver arrives at a client location, it takes the box of all the released items up to this arrival time. A box can thus contain all the product types. Each box must then be delivered on-time to a lab (i.e., satisfying the due date of all the involved items, which is determined by the first released item of the box). Some items can be late, but an overall *service level* $SL = 90\%$ has to be satisfied. Similarly, it is allowed that some items are not collected at all, but such a *dropout level* cannot exceed $DL = 3\%$. The SL and DL values are imposed by $COMP$. There is a constant service time t^s (set to 10 min) at the client location (for the pickup), at the hub location (if any, for the transfer), and at the destination (for the delivery). There are four item types (i.e., products) denoted here as A, B, C and D. Each item has to be analyzed in a lab by a dedicated machine, and a machine can only process one dedicated product (i.e., there are four machine types).

For each item i, we know its release time r_i (i.e., the time at which it is available at the involved client location) and its type k_i. It is not allowed to interrupt the processing of an item (i.e., preemptions are forbidden). Anytime an item is released, it should be analyzed (in a lab) within a due time of $D = 6$ h (i.e., the due date $d_i = r_i + D$), otherwise the item chemical properties have a risk to be deteriorated. Each machine can handle only one item at a time, and uses a FIFO rule (first-in-first-out: the items are processed according to the delivery sequence). For each machine of type $k \in \{A, B, C, D\}$, we know its speed defined as the number a_k of items that it can analyze per hour. Each machine can also have independent items to analyze, coming from a few clients that are out of our control. Such independent items reduce slightly the machine availabilities for some periods. The proportion of independent items is approximately 7% of the daily demand (observed input data).

Each *regular* item (i.e., of type B, C or D in this work) has to satisfy the *pre-processing* and the *pre-analysis* constraints. More precisely, each regular item i cannot be processed by any machine during a waiting time $s_i^{(w)}$ of 30 min after its release time r_i (pre-processing constraint). Next, it has to be pre-analyzed by a pre-analysis machine during $s_i^{(c)} = 15$ min. For products B and D, the pre-analysis can occur in any location of the network (i.e., a client, a hub or a lab); in contrast, for product C, it must occur in a lab (pre-analysis constraint). Formally, each regular item i has a setup time of $s_i^{(w)} + s_i^{(c)} = 45$ min before it can be processed in a lab. From a practical standpoint, we can assume that there is always an available pre-analysis machine, and that if pre-analysis occurs in a lab, the pre-analysis operation is triggered as soon as possible (i.e., the above-mentioned FIFO rule is overruled).

Let D_i (resp. R_i) be the time at which item i is delivered (resp. released, i.e., available for the analysis) to its associated lab. R_i can be computed as follows.

- $R_i = D_i$: if i is non-regular, or if i is of type B or D and the pre-processing ($s_i^{(w)}$) and the pre-analysis ($s_i^{(c)}$) have already occurred before D_i.
- $R_i = \max\{D_i + s_i^{(c)}, r_i + s_i^{(w)} + s_i^{(c)}\}$: if i is regular and the pre-analysis machine is employed in the lab.

Consider the following elements:

- m_k^L: number of machines of type k in lab L.
- $h_k^L(t)$: number of items of type k waiting, at time t, to be processed in lab L.
- $w_k^L(t)$: waiting time, at time t, of an item of type k in lab L (as other items have to be processed first).
- S_i: starting time of item i (time at which item i starts to be analyzed in its lab).
- C_i: completion time of item i (time at which the associated analysis is finished).
- a_k: speed of machine k (number of items that it can analyze per hour).

As a consequence, we have: $w_k^L(t) = \frac{h_k^L(t)}{a_k \cdot m_k^L}$; $S_i = R_i + w_{k_i}^L(R_i)$; $C_i = S_i + w_{k_i}^L(R_i) + \frac{1}{a_{k_i}}$; $C_i < r_i + D$ (due date constraint).

The optimization problem (i.e., decision variables, constraints, objective functions) can be formulated as follows.

Decision Variables

- number m_k^L of machines for each type k that is present in each lab L;
- number v^l of vehicles (constrained and external) assigned to each location l;
- route s^v of each vehicle v (constrained and external);
- pre-analysis location of each item $i \in \{B, D\}$.

Constraints

- each item i has to be delivered to a lab;
- for each lab L, there is a set M^L of machines that cannot be removed (typically, for each lab, each machine type must be always represented);
- each machine can handle only one item at a time;

- pre-processing and pre-analysis constraints (for the regular items);
- service level: the percentage LI of late items (i.e., exceeding the due date) cannot exceed the service level SL imposed by $COMP$;
- dropout level: the percentage NCI of non-collected items cannot exceed the dropout level DL imposed by $COMP$;
- respect the opening hours of the lab;
- each vehicle v must start in its initial location l^v, and each constrained vehicle v must return to l^v at the end of the day (even out of the opening hours of l^v);
- there is a constant service time t^s when picking up, when transferring, and when delivering boxes;

Objective Functions to Minimize
(f_1) number of employed machines; (f_2) number of external vehicles; (f_3) number of constrained vehicles; (f_4) total time spent by the vehicles on the road. (f_4) is computed for each vehicle v as the difference between (1) the time of its last delivery (or its return time to l^v if v is constrained), and (2) its first departure time from l^v. As for many other situations in practice (e.g., [15]), company $COMP$ has imposed a lexicographic consideration of these objectives (i.e., no improvement on a lower-level objective f_{o+1} can compensate a degradation of a higher-level objective f_o). This prioritization is straightforward as these objectives are ranked according to the impact on the overall costs that $COMP$ has to face. Without lexicographic optimization, the consideration of nonlinear costs might be investigated with the use of a variable neighborhood search [1].

3 Solution Method

First of all, it is important to be aware that company $COMP$ aims at having a quick solution method for tackling the PVRP. Indeed, in a real context, random events (e.g., unexpected items appear during the day, a vehicle has a breakdown) can occur, and the involved decision maker should be able to generate an updated solution quickly. For this reason, we do not investigate cumbersome metaheuristics, but we rather focus on streamlined heuristics. The overall approach is made of the four below steps, where steps (S1) to (S3) focus on all objectives f_2 to f_4, whereas step (S4) is specifically designed for reducing f_1.

- (S1) build sectors (i.e., decompose the network into smaller parts);
- (S2) build routes (i.e., construct the collecting routes in each sector and the delivery routes from sectors to labs);
- (S3) assign vehicles to routes;
- (S4) try to find a solution with fewer machines (i.e., restart with step (S1)).

3.1 Build Sectors

Let $G = (X, Z, U)$ be the considered oriented network, with client set X, lab set Z, and arc set U (containing the path between each pair of locations). As a pre-processing phase, we propose to split the network G into sectors $(G_1, G_2, \ldots, G_q)$. The following elements are associated with each sector G_j: a set $X_j \subset X$ of clients (where $(X_1, X_2, X_3, \ldots)$ is a partition of X, thus $X_j \cap X_{j'} = \emptyset,\ \forall j, j'$), a single hub $y_j \in X$ (the same hub can be used for different sectors), and a single lab z_j (the same lab can be used for different sectors).

There is obviously no perfect procedure for building sectors. Each G_j is built based on the following ideas: (1) group clients that are close to each other; (2) group clients with significant demand peaks at similar times; (3) determine the hub y_j in X that has the smallest average distance with the clients of X_j (break ties with the proximity of the closest lab); (4) determine the lab $z_j \in Z$ that is the closest to y_j; (5) the working load of a lab should depend on its capacity (i.e., number of machines of each type); (6) a big client (i.e., triggering many items) should be favored to be a hub. Note that if a lab z_j is already located close enough to a sector G_j, we can set $y_j = z_j$ and thus we avoid managing transfers for sector G_j.

3.2 Build Routes

Two types of trips are designed for each sector: collection routes and delivery routes. A *collection route* consists in picking up boxes of items in the client locations and in bringing them to the associated hub. A *delivery route* only involves a hub and a lab. As a consequence, each box is first picked up by a collection route (by the involved vehicle), and it is next delivered to a lab by a delivery route (with a different vehicle). As a consequence, two functions are possible for a vehicle: collect boxes through a collection route or deliver boxes through a delivery route.

An important feature of the problem is that the picking time in each location is a decision variable. As items can be released anytime during the day in any client location, we propose that each location (either a client, a hub or a lab) should be visited with a rather stable frequency during the day. Consider a sector G_j. Let A_j be an analysis parameter (tuned to 30 min) corresponding to the allocated time for performing the analysis at lab z_j. It means that any item i should be delivered to its associated lab $A_j - t^s$ minutes before its due date $d_i = r_i + D$. Let $\hat{t}(y_j, z_j)$ be the standard travel time (in minutes) between hub y_j and lab z_j. Therefore, if item i is released at time r_i, it must be picked up from its associated hub y_j within $H_j = D - A_j - \hat{t}(y_j, z_j) - 2 \cdot t^s$ minutes (note that $2 \cdot t^s$ captures the service time at the hub and at the lab). As a consequence, a delivery route can be planned from y_j to z_j every H_j minutes. In addition, as the demand is not a uniform function during the day, the last *cycle* (i.e., collection routes followed by the subsequent delivery routes) is delayed in order to meet the SL and DL constraints. For instance, if a

cycle is planned every 3 h but the working day is 10 h, then the durations of the two first cycles are 3 h, but the last cycle has a duration of 4 h (i.e., the last collecting routes are delayed by 1 h).

Let T_j be a transportation parameter (tuned to 30 min) that allocates a safety time period for performing any collection route in G_j. Therefore, each collection route of G_j cannot exceed a planning window of $W_j = H_j - T_j$ minutes. Based on this information, we can determine the number of collection routes that are theoretically necessary for G_j, and thus, the theoretical number V_j of required vehicles can be deduced. The computation of V_j is obtained as follows. For each sector G_j, we consider the classic VRP with depot y_j and client set X_j. The simple and famous Clarke and White algorithm [3] can be performed, with the constraint that each route cannot exceed W_j minutes.

At the end of the route building process, each route is adjusted (i.e., some departures are delayed in some client locations) if pre-analysis machines can be used (for item types B and D) without exceeding the planning window W_j. All the remaining pre-analysis operations occur in the hubs (but without delaying the subsequent delivery routes) or in the labs.

3.3 Assign Vehicles to Delivery/Collection Routes

We have to be aware that the fleet of $COMP$ is expected to be oversized. Ideally, we propose to first assign the constrained vehicles to collection routes, and next the external vehicles to delivery routes.

There are two types of sectors. G_j is *overloaded* if it contains V_j or more constrained vehicles initially located on it. The overloaded sectors are first considered. For each overloaded sector (considered in a random order), only assign to it the vehicles that are the closest to y_j. The unassigned vehicles are thus available for other sectors if necessary. Next, for each non-overloaded sector G_j, do the following. On the one hand, assign to G_j all the constrained vehicles that are already there. On the other hand, iteratively assign a missing vehicle in a greedy fashion, by favoring the closest unassigned vehicle from the neighboring sectors. If some vehicles are still missing at the end of this process, use external vehicles in a greedy fashion as well (i.e., based on proximity).

After the above assignment phase, the set of unassigned vehicles can contain both external and constrained vehicles. What is important in a delivery route is the trip from y_j to z_j (because items are transported), but not the trip right after the delivery (indeed, the vehicle becomes empty afterwards). For each sector G_j, we have to determine the number of vehicles that are needed to perform all the scheduled trips $y_j \rightarrow z_j$. This computation is straightforward: one vehicle is enough if the return trip from z_j back to y_j can be completed before the next departure from y_j. Otherwise, a second vehicle has to be employed, and the same argument can be used to determine if additional vehicles are needed.

Given a sector G_j, for each vehicle involved in the collection of items, there are two types of route: (1) $y_j \rightarrow$ clients $\rightarrow y_j$; (2) initial location (if different from y_j) $\rightarrow$ clients $\rightarrow y_j$. Type (1) represents the regular situation that is likely to be repeated during the day anytime there is a departure of a delivery route from y_j. This situation can again be solved with the Clarke and White algorithm. Type (2) occurs as the first morning route of each vehicle that is initially not located in y_j. In this case, each vehicle moves in a greedy fashion: at each step, it moves to the closest unserved client. If no more client is unserved, all the vehicles move to y_j, resulting in the start of the regular situation (i.e., type (1)).

3.4 Reduce the Resource Levels

The previous subsections aim at minimizing f_2 to f_4 if the overall machine capacity is fixed to machine set M (also called as a *resource level*). Let $s(M)$ be the solution returned by the above routing procedures, denoted from now as *ROUTE(M)*. In order to reduce f_1 as well, it is proposed to work level by level (i.e., focusing on one level at a time).

Solution methods based on levels were successfully adapted to various combinatorial optimization problems (e.g. [11, 14]). We have $f(s) < f(s')$ if s is better than s' with respect to the imposed lexicographic approach $(f_1) > (f_2) > (f_3) > (f_4)$ (i.e., no improvement on a lower-level objective f_{o+1} can compensate a degradation of a higher-level objective f_o). Let $s^\star$ be the best solution encountered during the search process, and let $f^\star = (f_1(s^\star), f_2(s^\star), f_3(s^\star), f_4(s^\star))$ be its associated objective-function values. The f_1-reduction procedure is depicted in Algorithm 1.

A m-drop move consists of removing machine m from machine set M. It is important to be aware that anytime a m-drop move is investigated for a lab L_j, it has the following consequences. First, its analysis parameters A_j are slightly augmented. Indeed, if the capacity of L_j is reduced, more time should be allocated to it in order to perform its assigned tasks. Second, if a lab has a smaller capacity, it might be assigned with fewer sectors (see point (4) of Sect. 3.1). Third, more vehicles might be used (see Sects. 3.2 and 3.3).

4 Experiments

The experiments were performed on real data (slightly perturbed to preserve the confidentiality agreement). All the algorithms were coded with C++ under Linux, and run on 2.8 GHz Intel Quad-core i7-7700HQ processor with 16 GB of DDR4 RAM. As the proposed algorithms are quick because of their very natures, computing times are very small (i.e., always less than a minute for planning a whole day) and thus not reported here. The network has a structure like the RC benchmark instances [16]. That is, there are clustered clients on the one hand, and zones

Algorithm 1 Reduction of f_1

Initialization

1. set M to the resource level employed by $COMP$ (or to any straightforward upper bound);
2. generate an initial feasible solution s with *ROUTE(M)*;
3. set $f^\star = (f_1(s), f_2(s), f_3(s), f_4(s))$ and $s^\star = s$;

While $f^\star$ can be reduced, do:

1. perform the best m-drop move: generate a solution with $ROUTE(M - \{m\})$ for each possible m-drop move (respecting M^L for each lab L), and keep the best solution $s(M - \{m^\star\})$ associated with the removal of machine $m^\star$ from level M;
2. if s is feasible and if $f(s) < f^\star$, set: $s^\star = s$, $f^\star = (f_1(s), f_2(s), f_3(s), f_4(s))$, $M = M - \{m^\star\}$;

Return $s^\star$ with value $f^\star$; best encountered level M.

for which the clients are randomly distributed on the other hand. The following approach is proposed to benchmark our solution. We simulate the solution provided by $COMP$ (such solutions are built by hand based on the $COMP$ decision-maker experience), and then we measure how our own solutions (generated by the above-proposed algorithms) can improve the existing solutions (percentage gaps with respect to each f_o).

The considered application is characterized by the following data: the planning horizon is a working day (roughly 12 h), 11,000 items, 50 client locations, less than 5 lab locations, 40 constrained vehicles, 20 external vehicles, 3500 (resp. 4500, 500, 2500) items of type A (resp. B, C, D). The a_k values (in items/hour) are approximately the following: $(a_A, a_B, a_C, a_D) = (100, 800, 200, 200)$.

The $COMP$ solution is characterized by a service level of 90.4% and a dropout level of 2.9%. The average waiting time of an item in a lab (i.e., before being processed) is 28.8 min. Considering the same set of machines (i.e., the same f_1-value), our method is able to achieve the following improvements: a f_2-reduction of 16 external vehicles, a f_3-reduction of 5 constrained vehicles, a f_4-reduction of 22 h (of time spent on the road by the vehicles, corresponding to a 22%-gap). Moreover, the service level and the dropout level are 94.9% and 2.7%, respectively. The average waiting time of an item in a lab is 20.3 min. Obviously, the improvements are very significant, while preserving (and even augmenting!) service level indicators. If f_1 is also optimized (using Algorithm 1), it can be reduced by seven machines without augmenting f_2, f_3 and f_4. Moreover, the service level and the dropout level are 92.7% and 2.7%, respectively. The average waiting time of an item in a lab is 29.8 min.

5 Conclusion and Future Works

In this paper, we have proposed a solution method for an integrated-logistic problem (PVRP) combining vehicle routing and resource allocation. The resource is made of two types: vehicles for performing pickups and deliveries of items, and machines for processing the delivered items. The problem was motivated by a real company, and its originality relies on the specific features and dimensions that are added to the well-known vehicle routing problem (VRP) (e.g., multiple demands/pickups/depots/products, due dates, service level, dropout level, possibility of transfers, resource optimization). The experiments conducted on real data show the significant benefits of the integrated approach with respect to the solutions provided by the company. Various avenues of research can be envisioned.

- Non-identical vehicles might be employed. For instance, use scooters in addition to cars, as scooters are faster/cheaper than cars, but they might be limited by capacity constraints.
- Allow some items to be rejected. In such a case, pay an external taxi service to deliver such items, and penalize these rejections with a dedicated objective function that is appropriately ranked in the lexicographic optimization. The idea is to avoid having an additional vehicle only for a few items occurring in some peak demand periods.
- As proposed in [13] for the VRP, an on-line version of the PVRP could be investigated (e.g., variable traffic conditions, items are revealed progressively during the day, routes are updated anytime a new event/item occurs).
- Decisions can also be made on the network design (e.g., where to locate the labs, [7]) and on the vehicle allocation (i.e., where to initially locate the vehicles [18]).

References

1. Bierlaire, M., Thémans, M., Zufferey, N.: A heuristic for nonlinear global optimization. INFORMS J. Comput. **22**(1), 59–70 (2010)
2. Ciavotta, M., Meloni, C., Pranzo, M. Scheduling dispensing and counting in secondary pharmaceutical manufacturing. AIChE J. **55**(5), 1161–1170 (2009)
3. Clarke, G., Wright, J.R.: Scheduling of vehicles from a central depot to a number of delivery points. Oper. Res. 12 (4), 568–581 (1964)
4. Cortes, C., Matamala, M., Contardo, C.: The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. Eur. J. Oper. Res. **200**, 711–724 (2010)
5. Darvish, M., Coelho, L.: Sequential versus integrated optimization: Production, location, inventory control, and distribution. Eur. J. Oper. Res. **268**(1), 203–214 (2018)
6. Drexl, M.: Synchronization in vehicle routing: a survey of VRPs with multiple synchronization constraints. Transp. Sci. **46**(3), 297–316 (2012)
7. Farahani, R.Z., Asgari, N., Heidari, N., Hosseininia, M., Goh, M.: Covering problems in facility location: a review. Comput. Ind. Eng. **62**(1), 368–407 (2012)
8. Gendreau, M., Potvin, J.Y.: Handbook of Metaheuristics. International Series in Operations Research & Management Science, vol. 146. Springer, Cham (2010)

9. Golden, B.L., Raghavan, S., Wasil, E.A. (eds.) The Vehicle Routing Problem: Latest Advances and New Challenges. Springer US, New York (2008)
10. Grasas, A., Ramalhinho, H., Pessoa, L., Resende, M., Caballé, I., Barba, N. On the improvement of blood sample collection at clinical laboratories. BMC Health Serv. Res. **14**, 12 (2014)
11. Hertz, A., Schindl, D., Zufferey, N.: Lower bounding and tabu search procedures for the frequency assignment problem with polarization constraints. 4OR **3**(2), 139–161 (2005)
12. Labadie, N., Prins, C., Prodhon, C. (eds.) Metaheuristics for Vehicle Routing Problems. Wiley, New York (2016)
13. Respen, J., Zufferey, N., Potvin, J.Y.: Impact of vehicle tracking on a routing problem with dynamic travel times. RAIRO Oper. Res. **53**(2), 401–414 (2017)
14. Schindl, D., Zufferey, N. A learning tabu search for a truck allocation problem with linear and nonlinear cost components. Naval Res. Logist. **62**(1), 32–45 (2015)
15. Solnon, C., Cung, V., Nguyen, A., Artigues, C.: The car sequencing problem: Overview of state-of-the-art methods and industrial case-study of the ROADEF 2005 challenge problem. Eur. J. Oper. Res. **191**(3), 912–927 (2008)
16. Solomon, M.M.: Algorithms for the vehicle routing and scheduling problem with time window constraints. Oper. Res. **35**, 254–265 (1987)
17. Toth, P., Vigo, D. (eds.): Vehicle Routing: Problems, Methods and Applications. MOS – SIAM Series on Optimization. SIAM, Philadelphia (2014)
18. Vasco, R.A., Morabito, R.: The dynamic vehicle allocation problem with application in trucking companies in Brazil. Comput. Oper. Res. **76**, 118–133 (2016)

A Real Case on Making Strategic Logistics Decisions with Production and Inventory Optimization Models

E. Parra

Abstract A real case is described of the use of industrial optimization models (linear/integer programming) for logistics decisions in the medium term (annual planning) and strategic decisions. The application of these models to optimize annual operations and make strategic decisions on the sizing of storage capacity is studied. The optimization models are built with the author's software, which is capable of proposing and solving problems of the required size: thousands of equations, tens of thousands of variables and hundreds of thousands of non-zero coefficients. The models demonstrate the enormous power of this methodology and its potential savings in production/transport costs, which were as much as 20% in the case in this work.

Keywords Logistics · Strategic decisions · Optimization

1 Introduction

This work describes the use of industrial optimization models (linear/integer programming) for both medium term (annual planning) and strategic logistics decisions [1] in a real case.

Specifically, we study the use of models for optimizing the planning of annual operations and adopting strategic decisions for the sizing of storage capacity. The optimization models in the case study were built with the author's software and resolved with a commercial software capable of proposing and resolving problems of the required size: 8000 equations, 80,000 variables and 300,000 non-null coefficients.

Logistic problems has been studied widely: in well known books on industrial modeling like Winston [2], Williams [3], Baker [4] or Kallrath [5] and in papers with

E. Parra (✉)
Department of Economics, University of Alcala, Madrid, Spain
e-mail: enrique.parra@uah.es

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_35

applications to specific industries:: Martin et al. [6] presented a linear programming model for planning production, distribution and inventory operations in the glass sector industry. Chen and Wang [7] proposed a model to solve integrated supply, production and distribution planning for the steel sector. Ryu et al. [8] suggested a bi-level modeling approach comprising two linear programming models, one for production planning and one for distribution planning. Kallrath [9] presented a model applied to the process industry. Jung et al. [10] compared models for centralized and decentralized production and transport planning. Aghezzaf [11] explores capacity planning and warehouse location. Mula et al. [12] is a good survey on mathematical programming models for supply chain production and transport planning. Kallrath [13] explains the important role of Modeling Languages in Mathematical Optimization: compulsory tool for model building. These models and tools, usually need a mathematical modeling expert.

In this work, a very versatile original approach is used since the planner (tactical or strategic) is the person who can change the model and apply it to different uses modifying the model that is constructed by the data provided not writing mathematical equations.

This methodology shows the great benefit of its use through the application to a real case of medium-term planning and strategic planning.

2 The Problem

The case study concerns an industry that manufactures an intermediate product from a raw material that can either be stored or transformed into a final product, which in turn can be shipped exactly as it is obtained -bulk-, or in different packaging formats. The most important costs are those involved in transforming the raw material and especially in transporting the product to the customers, who receive the product on a CIF (cost, insurance and freight) basis.

The company had 11 factories that received raw material, transformed it in the intermediate product and obtained final product from the intermediate one. Final product can be packaged in different formats. It is possible to store intermediate and final product in some of the factories. Figure 1 is a factory scheme.

The firm could use other seven warehouses (without production capacity) to receive product, store it and dispatch to other warehouses or to customers. The transformation rate of the raw material into intermediate product is faster than the process of obtaining the final product from the intermediate product, and when the raw material is being received, more intermediate product may be obtained than is processed to achieve the final product, meaning the inter-mediate product needs to be stored. Intermediate and final product in any of the packaging options can be transported between factories and warehouses.

Final product is stored in eight factories and can be dispatched as it is obtained or in the different packaging formats. Table 1 shows the installed capacity in thousands of tons (kt) to store bulk—the most important- before the optimization and the

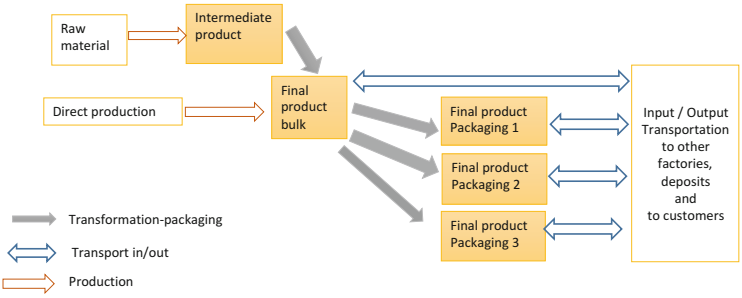


Fig. 1 Factory scheme

Table 1 Installed storage capacity

Factory	Installed cap. (kt)
Fac01	25
Fac02	20
Fac03	35
Fac04	20
Fac05	40
Fac06	20
Fac10	70
Fac11	45
Total	275

strategic study. Some of the warehouses can also be used to store either bulk or packaged product. Packaging operations may take place in any of the factories (although not all types in all factories) and within certain limits. There are shipping constraints in each factory. There are also constraints on the production rate (raw material-intermediate product and intermediate-bulk product) in addition to the aforementioned packaging constraints.

The customers are distributed over a broad geographic area, so it was considered enough to use an approach that groups them by proximity (postcode, municipalities, provinces), except for the most important customers and/or those that for some reason represent another type of constraint in terms of the source of the supply.

The product is manufactured and stored in the factories and sent to the deposits; it is then sent from the deposits and/or factories to the customers on a CIF basis. As the cost of transport is supported by the company, its optimization is of crucial importance.

The procurement of the raw material is subject to strong seasonality, which varies from one geographic region to another. There is also significant seasonality in the demand and production, which makes the choice of storage capacity critical for minimising the cost of transporting the products to the customers.

In addition to resolving medium-term operations, a strategic discussion is afforded: could be said to be whether increased investment in storage capacity could reduce transport costs, which requires an analysis of the investment in warehouse

space from the standpoint of potential reductions in transport costs. Later it is explained how this strategic decision is addressed in this case.

Using optimization models, proposed and resolved with the author's software described below, a study is made of where and by how much the factories' storage capacity must be increased. The software shows the optimal transport solution available to the company in each scenario and provides savings of up to 20%, as will be seen.

The following is an explanation of the initial situation of the company, the mathematical model, the software and the methodology used in the strategic analysis of new investments, and concludes with the results obtained. Specific data related with the real case are used to make the methodology easier to follow, although with re-scaled data to guarantee confidentiality.

3 Optimization of the Operations

3.1 Introduction

The company opted to use linear programming models for planning all its transport operations and to study the expected savings from possible warehouse expansions.

Given the seasonality of the demand and production, it is essential that the model should have good seasonal detail. It therefore needed to be multi-period; specifically, each case study covers a whole year as a planning horizon with monthly periods.

The models were built and resolved using the author's software (TPOS), whose latest version is described in Parra [14]. TPOS is a set of programs for personal computers that allow the combined resolution of production and distribution optimization. TPOS places greater emphasis on the modelling of the multilevel transport network.

The problem resolved by TPOS (details in Parra [14, 15]) can be summarised in Sect. 3.2.

3.2 Mathematical Model (TPOS)

Given **N** nodes (that model suppliers, factories, deposits, customers) connected by transport arcs (A) in a network, **P** products produced and transported over the network during **T** planning horizon periods of variable length. Let i, i´ ∈ **N** (Nodes, set), j, j´ ∈ **P** (Products, set), k ∈ **T** (Periods, set).

The generated mathematical optimization model purpose is to find the optimal value of the following *variables*:

1. *PX (i, j, k).* Quantity produced of product "j" at node "i" during period "k".

2. *TX (i, j, j', k).* Quantity transformed from product "j" into product "j´" at node "i" during period "k".
3. *X (i, l, j, k).* Quantity (if it is generated as a continuous variable) transported from origin node "i" to destination node "l" of product "i" during period "k", or (see below). Transportation arcs are explicitly declared buy the modeller.
4. *CS (i, j, k).* Quantity consumed (sales = demand satisfaction) of product "j" at node "i" during period "k"
5. *OF (i, j, k).* Quantity of product "j" leaving node "i" during period "k".
6. *StoF (i, j, k).* Quantity stored at the end of period "k", of product "j" at node "i".

Main *data*:

1. Price (i, j, k). Price paid by costumer for a unit of product "j" at node "i" during period "k".
2. CostProd (i, j, k). Unit cost to produce product "j" at node "i" during period "k".
3. CostSto (i, j, k). Unit cost to stock product "j" at node "i" during period "k".
4. CostTra (i, i', j, k). Unit cost to transport product "j" from node "i" to node "i'" during period "k".
5. Cost TR (i, j, j',k). Unit cost to transform product "j" into product "j' "at node "i" during period "k".
6. Yield (i, j, j', k). Quantity of product "j' "obtained when a unit of product "j" is transformed into product "j' "at node "i" during period "k".

Other data used are introduced in the constraint where is used.

The objective function is to "Maximize the variable margin: Revenues from product sales minus Variable Costs (sum of supply, manufacturing, stock and transportation costs)":

$$\sum_i \sum_j \sum_k \mathrm{Pr}\,ice\,(i,\,j,\,k)\,.CS\,(i,\,j,\,k) - \sum_i \sum_j \sum_k CostProd\,(i,\,j,\,k)\,.PX\,(i,\,j,\,k)$$
$$- \mathrm{Cos}t Sto.\sum_i \sum_j \sum_{k=1}^{T-1} StoF\,(i,\,j,\,k) - \sum_i \sum_{i' \neq i} \sum_j \sum_k \mathrm{Cos}tTra\left(i,\,i',\,j\right).X\left(i,\,i',\,j,\,k\right)$$
$$- \sum_i \sum_j \sum_{j'} \sum_k \mathrm{Cos}tTR\left(i,\,j,\,j',\,j\right).TX\left(i,\,j,\,j',\,k\right)$$

Constraints:

1. Balance for Initial stock, productions, transformations, consumption, shipments and final stock for each node (i), product (j) and period (k).

   These are the core equations of the model. Each product: can be produced (variables PX) in a node each period, can be transformed (variables TX) in other product in the same node each period, can be transported to other node (X variables to other nodes), received from other node (X variables from other nodes), can be stored in the node where is produced (StoF variables) or can be consumed in the node (CS variables).

Selecting which of these operations are allowed in each node/period it is a very versatile method to represent a factory with or without storage, a warehouse, a transhipment center, or a customer.

$$\begin{aligned}&\text{StoF}\ (i,j,k) - \text{PX}\ (i,j,k) + \text{CS}\ (i,j,k) - \sum_{i'} \text{X}\ \left(i',i,j,k\right) + \sum_{i'} \text{X}\ \left(i,i',j,k\right)\\&+ \sum_{j'} \text{TX}\ \left(i,j',j,k\right) - \sum_{j'} \text{Yield}\ \left(i,j,j',k\right).\text{TX}\ \left(i,j',j,k\right)\\&= \text{StoIni}\ (i,j) \quad \forall (i,j,k=1)\end{aligned} \tag{1}$$

$$\begin{aligned}&\text{StoF}\ (i,j,k) = \text{StoF}\ (i,j,k-1)\ \text{PX}\ (i,j,k) + \text{CS}\ (i,j,k) - \sum_{i'} \text{X}\ \left(i',i,j,k\right)\\&+ \sum_{i'} \text{X}\ \left(i,i',j,k\right) + \sum_{j'} \text{TX}\ \left(i,j',j,k\right)\\&- \sum_{j'} \text{Yield}\ \left(i,j,j',k\right).\text{TX}\ \left(i,j',j,k\right) = 0 \quad \forall (i,j,k>1)\end{aligned} \tag{1'}$$

2. Quantities Pr (i, j, k) are produced in node i. Limits can be set: a lower limit, ProdMin (i, j, k), and an upper limit, ProdMax (i, j, k), for each product "j" and during each period "k" with unitary cost CostProd (i, j, k). These quantities can be seen like supplies at the node. This is the only way to *generate* material in the model.

$$\text{ProdMin}\ (i,j,k) \le \text{PX}\ (i,j,k) \le \text{ProdMax}\ (i,j,k) \quad \forall (i,j,k) \tag{2}$$

3. Lower limit (ProdCMin) and upper limit (ProdCMax) can be set for joint production (sum for all products) for each period y node.

$$\text{ProdCMin}\ (i,k) \le \sum_{j} \text{PX}\ (i,j,k) \le \text{ProdCMax}\ (i,k) \quad \forall (i,k) \tag{3}$$

4. Lower limit (ProdPMin) and upper limit (ProdPMax) can be set for a product production in the whole horizon and at each node.

$$\text{ProdPMin}\ (i,j) \le \sum_{k} \text{PX}\ (i,j,k) \le \text{ProdPMax}\ (i,j) \quad \forall (i,j) \tag{4}$$

5. Any product can be transformed into another with a yield, a unitary cost (CostTR values, included in the objective function) and different limits can be set:

a. TRMin/TRMax (minimum/maximum for each transformation from product "j" to product "j'" in node i and period "k")

$$\text{TRMin}\ \left(i, j, j', k\right) \leq \text{TR}\left(i, j, j', k\right) \leq \text{TRMax}\ \left(i, j, j', k\right) \quad \forall \left(i, j, j', k\right) \tag{5}$$

b. TR1Min/TR1Max (minimum/maximum for total transformation of product "j" to other products in node "i" and period "k")

$$\text{TR1 Min}\ (i, j, k) \leq \sum_{j'} \text{TR}\ \left(i, j, j', k\right) \leq \text{TR1 Max}\ (i, j, k) \quad \forall (i, j, k) \tag{6}$$

c. TR2Min/TR2Max (minimum/maximum for total transformation of product "j" to other products in node "i" in the sum of periods),

$$\text{TR2 Min}\ (i, j) \leq \sum_{j'} \sum_{k} \text{TR}\ \left(i, j, j', k\right) \leq \text{TR2 Max}\ (i, j) \quad \forall (i, j) \tag{7}$$

d. TR3Min/TR3Max (minimum/maximum for total transformation of all the products in each node "i" and in each period "k"),

$$\text{TR3 Min}\ (i, k) \leq \sum_{j} \sum_{j'} \text{TR}\ \left(i, j, j', k\right) \leq \text{TR3 Max}\ (i, k) \quad \forall (i, k) \tag{8}$$

e. TR4Min/TR4Max (minimum/maximum for total transformation of all products for all periods in node "i"),

$$\text{TR4 Min}\ (i) \leq \sum_{j} \sum_{j'} \sum_{k} \text{TR}\ \left(i, j, j', k\right) \leq \text{TR4 Max}\ (i) \quad \forall (i) \tag{9}$$

6. Any product can be stored from a period to the next one. Lower and upper stock limits can be set (StoMin y StoMax) for each pair product/node. At the beginning of the planning horizon, the nodes have an initial stock of each of the products (StoIni). There is a unitary stock cost (CostSto)

$$\text{StoMin}\ (i, j) \leq \text{StoF}\ (i, j, k) \leq \text{StoMax}\ (i, j) \quad \forall (i, j, k) \tag{10}$$

7. Lower and upper limits can be set for the sum of all product stocks at node "i" in each of the periods (StoCMin and StoCMax)

$$\text{StoCMin}\ (i) \leq \sum_{j} \text{StoF}\ (i, j, k) \leq \text{StoCMax}\ (i) \quad \forall (i, k) \tag{11}$$

8. Lower and upper demand limits can be set for each period, product and node (DemMin and DemMax); the revenues are the product of the price ("Price" values included in objective function) and the demand. This is the way to represent the sales. They are the income in the model.

$$\text{DemMin}\ (i, j, k) \leq \text{CS}\ (i, j, k) \leq \text{DemMax}\ (i, j, k) \quad \forall (i, j, k) \tag{12}$$

9. Lower and upper transportation limits can be set for each arc (from node "i" to node " i' "), product "j" and period "k" (XMin and XMax). This is the way to model the transportation between nodes. There is a unitary transport cost (CostTra)

$$\text{XMin}\ \left(i, i', j, k\right) \leq \text{X}\ \left(i, i', j, k\right) \leq \text{XMax}\ \left(i, i', j, k\right) \quad \forall (i, j, k) \tag{13}$$

10. Lower and upper limits for the inflow to a node can be set (InfMin and InfMax). To avoid very low sendings it is posible to impose a threshold level defining the outflow variable as a semi continuous variable (MIP version).

$$\text{InfMin}\ (i, j, k) \leq \sum_{i'} \text{X}\ \left(i', i, j, k\right) \leq \text{InfMax}\ (i, j, k) \quad \forall (i, j, k) \tag{14}$$

11. Lower and upper limits for the outflow from a node (variables OF) can be set (OutMin and OutMax).

$$\text{OF}\ (i, j, k) = \sum_{i'} \text{X}\ \left(i, i', j, k\right) \quad \forall (i, j, k) \tag{15}$$

$$\text{OutMin}\ (i, j, k) \leq \text{OF}\ (i, j, k) \leq \text{OutMax}\ (i, j, k) \quad \forall (i, j, k) \tag{16}$$

12. Also, it is possible to include special constraints using the variables already generated in the model (productions, shipments, ...)

$$\text{R\_Specials : Those included explicitly by the user using generated variables} \tag{17}$$

TPOS builds the model from data supplied by the user in the form of alphabetic codes that guide the construction of the model's equations by the software. The user do not write any equation, only fill codes: production, transformation, shipment with keywords: PROD, TR, NET, ... (details in Parra [14, 15]).

The data can be stored, for example, on a spreadsheet or linked to databases. The software optimizes the model and creates a report of the results that can be processed using any database or spreadsheet software.

Models built by TPOS are multi-node, multi-product and multi-period. In general, on any node a product can be produced, transformed into another, stored, received from other nodes, or sent to other nodes and consumed. It can represent a factory (everything except consumption), a deposit (which receives, stores and ships) or a customer (only receives and consumes). All the operations described

can be modelled with no limits or with minimum and maximum limits. More detail about the possibilities of TPOS are shown in the references.

3.3 Case Study

In the case study, 76 nodes were considered for the basic model (named ALM0): the factories, the deposits, the client groupings ("provinces") and some special clients. The model uses 12 periods (months) and five products: intermediate product, and final bulk product which is shipped either as is or in three packaging formats. The practical considerations in this case meant that the ALM0 model consisted of approximately 8000 constraints, 80,000 variables (with 5000 individual limits) and around 250,000 non-null coefficients. The model was built using the TPOS codes, resolved with an optimizer—for example FICO [16]—integrated in TPOS, and a written report was then created with all the details of the model and the solution, with an output file in CSV format. This can be done in very little time in current personal computers under Windows. Another options for optimiser selection can be IBM [17], LINGO [18] or others listed in Fourer [19].

Once the base model (ALM0) had been resolved, the optimal solution could be observed and compared with the real transport expenses. The use of optimization pointed to a new annual planning solution: the optimal solution reduced real transport costs by around 20%. This first stage allowed the company to reorder its production and transport structure. Since its introduction, this method of planning operations has become standard in the company. This first phase also identified possible additional improvements if investments were made in storage looking at the dual values of the solution, so a second phase in the use of the optimization was begun in order to identify the most profitable investments in storage capacity. In a third phase, the company even embarked on a wholesale restructuring of the business, including factory expansions and closures as it will be explained in Sect. 4.

Essentially the same optimization model was used in each scenario, with variations in the data as explained below.

4 Strategic Study of Warehouse Expansion

4.1 Scenarios

Various scenarios were contemplated, with variations in customer demand and factory output limits (due to expansions planned). Specifically, two versions were used for demand, the first corresponding to the data for the last real year available ("Historic") and the second to the estimated demand for the following year ("Future"). There were two production options: we will refer to them as "Current", which was the historic situation; and "Future" which considered the finalisation of the investments currently underway to increase manufacturing capacity. Four scenarios are obtained from the combination of demand and production.

Table 2 Savings compared to the historic plan

Case	Demand	Production	Cost	Cost reduction over base cost (%)
Historic	Current	Historic	17,588	
Base (optimal)	Current	Historic	14,672	17
Historic	Future	Historic	18,770	
Base (optimal)	Future	Historic	15,658	17
Historic	Future	New	17,588	
Base (optimal)	Future	New	14,672	12

In order to have a point of comparison, the base model which has constraints in the storage and production capacity, among others, was optimized to the historic figures and the real data for the last year was used as demand.

The result obtained (re-scaled figures because confidentiality) was a transport cost of 14.670 K$, representing a 17% improvement over the real costs (almost 3 M$ with the data used). Thus, as expected, the first significant result is that the use of an optimization model leads to a highly significant improvement in the company's economic results.

These same calculations made with the following year's demand reveal a similar saving of 17% when continuing with the historic plan vs. adopting the solutions proposed by the optimization model.

If, in addition to changing the demand, a new production scheme is added to approximate the most probable future scenario, the cost saving is still 12%. These results can be seen in Table 2.

The optimal solution also suggests incentives (via dual values) to increase bulk storage capacity. Specifically, the most important incentives are: Fac01: −45 $/t, Fac07, −35 $/t, Fac04, −28 $/t.

Based on this result, we continue to the second strategic phase, in which we study whether this result can be further improved by expanding storage.

4.2 Resolution of the "Unlimited" Case

To find a "floor cost" for transport costs and observe the optimal levels of storage, another model was created (ALM0I) containing the same elements as ALM0 except that any amount of stock is permitted in factories with warehouses. In other words, the real storage capacities are eliminated. This reveals the best possible situation.

It was observed that this release from the real limits led to the completion of the entire plan with a cost that was 3.5 M$ lower than under the historic demand scenario, representing a saving of 20% compared to the starting situation, or 4% of the optimized solution with TPOS. When the optimization was done with the new production and future demand scenario, a very similar potential improvement of 18% was found compared to the historic case, or 8% (1.3 M$) compared to the optimal solution in TPOS. We will call the savings in this "unlimited" case the "maximum theoretical savings" (MTS) (Table 3).

Table 3 Maximum theoretical savings, MTS

Case	Demand	Production	MTS/base	MTS/optimum
Unlimited	Current	Historic	3.5 M$ (20%)	0.6 M$ (4%)
Unlimited	Future	New	3.5 M$ (18%)	1.3 M$ (8%)

4.3 Expansion Options

The results of the ALM0 model (as it was mentioned in Sect. 4.1) revealed that Factories 1, 7 and 4, in that order, had the greatest incentive to expand. The methodology followed was therefore to study the cases in the order suggested by the duals.

Based on this new demand and production scenario, the storage limit is only released in Factory 1 (Fac01). The resolution of this case implies an optimum with 16% lower costs compared to the base case, or 4% lower compared to the optimized case, but with a capacity limit.

In other words, 62% of the maximum theoretical savings (MTS) is achieved by only expanding the storage capacity in Factory 1 in the future demand and production case. This saving is achieved by expanding the storage in Factory 1, which would need to store up to 72 kt, and would therefore require an expansion of 47 kt, almost three times the installed capacity.

To improve this result, the storage capacities of the factories were released in the order suggested by the duals mentioned above. Starting from the previous case (Fac01) in which unlimited storage was allowed in Factory 01, the storage is also released in Factory 4 (next highest dual). This case (Fac01 + 04) provides an additional 270 k$ in savings compared to the Fac01 case, achieving 83% of the MTS. In this case Factory 1 requires less expansion of its capacity, which would be limited to an increase of 34 kt instead of 47 kt, but it implies adding 27 kt of storage to Factory 4. This means expanding the capacity by 61 kt, 22% of the total.

Finally, 99% of the MTS was achieved by releasing the storage capacity of Factory 10. This last case suggested allocating the capacities as follows (Table 4):

The same procedure was used to study more cases in the different demand and production scenarios, although due to their importance in the final decision only the above cases are included. All these data are shown in Table 5.

Table 4 Expansions with three expansions

Factory	Expansion (kt)
Fac01	32
Fac04	27
Fac10	40

Table 5 Possible expansions

Scenario: new demands and productions									
				Fac01 expansion		Fac01 + Fac04 expansion		Fac01 + Fac04 + Fac10 expansion	
	Storage Cap.	Unlimited	Expansion	Unlimited	Expansion	Unlimited	Expansion	Unlimited	Expansion
Fac01	25	45	20	72	47	59	34	57	32
Fac02	20	0	–	0	–	0	–	0	–
Fac03	35	54	19	35	–	35	–	35	–
Fac04	20	45	25	20	–	47	27	47	27
Fac07	45	40	–	45	–	45	–	44	–
Fac08	20	17	–	19	–	20	–	20	–
Fac10	40	80	40	40	–	40	–	80	40
Fac12	70	42	–	70	–	69	–	42	–
Total	275	324	104	301	47	314	61	325	99
			38%		17%		22%		36%
Costs (K $)	16.560	15.300	1.260	15.790	770	15.520	1.040	15.310	1.250
Ratio					61%		83%		99%
			1.214		1.652		1.716		1.267

4.4 Investment Analysis

These data were enough to apply the corresponding process of analysis of the investment. The cost of expansion was the amount that needed to be invested, and the annual cost saving was assumed to be the cash flows enabled by the investment.

The high return on investment in terms of IRR (internal rate of return) and NPV (net present value), and some non-quantitative considerations, pointed to the advisability of first expanding only in Factory 1, although this was postponed until after the expansion of Factory 4.

This method proved to be so versatile that the company almost immediately embarked on a third phase of strategic rethinking of all its production capacities in all its factories, leading to a wholesale reorganisation of the business. But ... that is another case.

5 Conclusions

This work describes a successful real case of the use of industrial optimization models (linear/integer programming) for both medium term (annual planning) and strategic logistics decisions. The models were built with the author's software and enabled a significant saving in transport costs, after which their use became standard for the company's planning operations.

However, the study also examined the profitability of strategic investments for the expansion of the company's storage capacity.

The use of models has demonstrated the vast power of this methodology and the cost saving it represents, as much as 20% in the case described in this work.

Acknowledgment I would like to thank to two anonymous referees for their wise advices that have improved the paper.

References

1. Powers, R.F.: Optimization models for logistics decisions. J. Bus. Logist. **10**(1), 106 (1989). https://www.ibm.com/es-es/analytics/cplex-optimizer
2. Winston, W.L.: Practical Management Science: Spreadsheet Modeling and Applications. Wadsworth Publishing, Belmont (2000)
3. Williams, H.P.: Model Building in Mathematical Programming, 5th edn. Wiley, Hoboken (2013)
4. Baker, K.: Optimization Modeling with Spreadsheets, 3rd Revised edn. Wiley, Hoboken (2015)
5. Kallrath, J., Wilson, J.M.: Business Optimisation: Using Mathematical Programming. McMillan, Basingstoke (1997)

6. Martin, C.H., Dent, D.C., Eckhart, J.C.: Integrated production, distribution, and inventory planning at Libbey–Owens–Ford. Interfaces. **23**, 78–86 (1993)
7. Chen, M., Wang, W.: A linear programming model for integrated steel production and distribution planning. Int. J. Oper. Prod. Manag. **17**, 592–610 (1997)
8. Ryu, J.H., Dua, V., Pistikopoulos, E.N.: A bilevel programming framework for enterprise-wide process networks under uncertainty. Comput. Chem. Eng. **28**, 1121–1129 (2004)
9. Kallrath, J.: Solving planning and design problems in the process industry using mixed integer and global optimization. Ann. Oper. Res. **140**(1), 339–373 (2005). https://doi.org/10.1007/s10479-005-3976-2
10. Jung, H., Jeong, B., Lee, C.G.: An order quantity negotiation model for distributor-driven supply chains. Int. J. Prod. Econ. **111**, 147–158 (2008)
11. Aghezzaf, E.: Capacity planning and warehouse location in supply chains with uncertain demands. J. Oper. Res. Soc. **56**(4), 453–462 (2005)
12. Mula, J., Peidro, D., Díaz-Madroñero, M., Vicens, E.: Mathematical programming models for supply chain production and transport planning. Eur. J. Oper. Res. **204**(3), 377–390 (2010)
13. Kallrath, J.: Modeling Languages in Mathematical Optimization (Applied Optimization). Kluwer, Dordrecht (2004)
14. Parra, E.: A software for production-transportation optimization models building. In: Daniele, P., Scrimali, L. (eds.) New Trends in Emerging Complex Real Life Problems, AIRO Springer Series, vol. 1. Springer, Cham (2018)
15. Parra, E.: Transfer prices assignment with integrated production and marketing optimization models. J. Ind. Eng. Manag. **11**(2), 262–275 (2018)
16. FICO (2019). https://www.fico.com/en/products/fico-xpress-optimization
17. IBM (2019) IBM ILOG CPLEX Optimiser. https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer
18. LINGO (2019) LINGO 18.0 - Optimization Modeling Software for Linear, Nonlinear, and Integer Programming. https://www.lindo.com
19. Fourer R (2015) Linear Programming: Software Survey. OR MS Today. Informs, vol. 42.3

A Bi-objective Mixed Integer Model for the Single Link Inventory Routing Problem Using the ϵ-Constraint Method

Arianne A. S. Mundim, Maristela O. Santos, and Reinaldo Morabito

Abstract In this paper, we study an Inventory Routing Problem for the Single Link case. In this problem, products must be transported from an origin point to a destination in order to meet the demand. The products can be delivered in a finite number of periods and the destination has a constant rate in each period. There are two costs associated with the problem: transportation cost and inventory cost. In the literature, the approaches usually are developed to the mono-objective problem, i.e., minimize both the inventory and transportation costs in a single function. However, for real companies, an analysis of these different costs is extremely important to define new policies. In order to deal with this literature gap, we develop a bi-objective method that considered the ϵ-constraint approach to deal with these two objectives. In numerical experiments, new instances based on the literature are presented and solved to optimality using an optimization solver. The experiments show that the model returns an efficient set of non-dominated solutions. Finally, the results indicate that using the proposed method, decision makers will have a powerful tool to construct the Pareto front.

Keywords Single link inventory routing problem · ϵ-Constraint method · Pareto front

A. A. S. Mundim (✉) · M. O. Santos
Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, SP, Brazil
e-mail: arianne@usp.br; mari@icmc.usp.br

R. Morabito
Departamento de Engenharia de Produção, Universidade Federal de São Carlos, São Carlos, SP, Brazil
e-mail: morabito@ufscar.br

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_36

1 Introduction

The problem studied considers a set of n products available at an origin point and demanded at a destination point at a constant rate (or demand for the products) in each period. The products can be transported from the origin to the destination regularly in a set of specific frequencies (defined by the inverse of the period) using a limited number of trucks with given capacity. The periods are integers from 1 to the number of frequencies, i.e., the maximum number of periods is the size of the planning horizon T. The problem aims to decide how much of each product to ship at each frequency in order to minimize transportation and inventory cost. This problem is referred to here as the bi-objective Single Link Problem. For example, the origin may represent a consolidation center and the destination a depot. Some origins regularly send products to the consolidation center, and then, the products are shipped to the depot [7]. This problem with discrete frequencies is shown to be NP-hard in [19]. The practical situations for the case in which the set of possible shipping frequencies is given can be found in [12–14] and [16].

Hall [12] proposed a new method to determine the optimal frequency for the single link problem. This optimal dispatch schedule is very flexible and allows for the combined inventory and transportation costs of the collection to be reduced. The method has been developed analytically and illustrated with a toy problem. In [14] is presented a model that can be used to find consistent and realistic reorder intervals for each item in large-scale production-distribution systems. They presented an algorithm to solve it. The model that results from these assumptions is an integer nonlinear programming problem. Two models are provided in [13]. The first one with a single constrained work center and the other one with multiple constrained work centers, and both have limited capacity. They provided necessary and sufficient conditions that characterize the solution and showed that the optimal partition of nodes in the production-distribution network is invariant to an arbitrary parameters. The chapter [16] reviews optimization-based methods for solving small IRPs. It proposes algorithms to solve these models. The chapter examines several different multi-stage generalizations of the model. A system having a serial structure is studied. A system with an assembly system structure is examined, and a model and an algorithm for a distribution system were discussed.

The Single Link Problem (SLP)—mono and bi-objective—is a variant of the Inventory and Routing Problem (IRP), introduced by Bell et al. [4]. The IRP involves both the origin's and the destination's inventory, in additional the routing to ship the products from origin to destination. Review to IRP is presented in [6] with an overview of the SLP, with examples, characteristics and different models and policies for the class of problems where the crucial decision is when to serve destinations. The IRP consist of a set of origins which must distribute a set of products to several destinations and origins have vehicles with a delivery capability and destinations, a demand.

In the IRP there are different techniques to control the origin's and destination's inventories. Thus, to meet the market's need for activities, initiatives have emerged in order to facilitate decisions made by the manager. Bertazzi and Speranza [6] present some existing techniques, the Retailer Managed Inventory (RMI) and the Vendor Managed Inventory (VMI), of which VMI stands out for providing good practical results in companies by providing a quick response to the destination [2]. In this paper, we studied the technique VMI.

The technique VMI consists of the control of the origin on the supply of the destination's stock, aiming the monitoring of the stock of the consumers by means of previously obtained data analyses. This strategy reduces uncertainty about inventory, increases efficiency through automation of activities and decreases expenses with transportation and storage of products. However, in order to obtain good confidence between the two parties involved, especially during the data collection stage. Some studies using VMI can be found in [3, 20, 21].

In [18] is introduced a mono-objective Single Link Problem version for this problem, where is summed the inventory and transportation cost; a version of this problem was introduced together with other models for shipping products from an origin to a destination. Speranza and Ukovich[18] and Bertazzi et al. [7] providing exact solutions for real instances of the SLP. Speranza and Ukovich [19] solve the problem using a greedy algorithm and a Branch-and-Bound algorithm, whose computational results from the last algorithm show the possibility of solving realistic sized problems.

The mono-objective Single Link Problem is deal in [5]. They present a general framework of analysis from which they derive the known approaches with a continuous frequency and with a set of given frequencies. They propose a new model for the case with discrete shipping times in which a shipment can take place at each discrete time instant.

To the SLP, the authors usually consider the mono-objective problem. However, in practice, the managers want more information about the relations between these objectives, how much the cost of inventory increases, when the cost of transport is minimal or exactly contrary. For this, in this paper, we proposed a bi-objective approach to deal as the SLP, where we can obtain a front of Pareto. A approach bi-objective to the SLP can be found in [9, 17] and [1].

A multi-objective algorithm embedded with column generation to solve a green bi-objective IRP is presented in [9]. They minimize the total cost overall supply chain network and CO_2 emission. They proposed the use of Noninferior Set Estimation algorithm combined with column generation to reduce the number of variables in the problem. Sadok et al. [17] determine the multi-tours of a homogeneous fleet of vehicles covering a set of sales-points and minimizing the distribution and inventory cost per hour. They analyze this problem as a bi-objective IRP in which the transportation cost and the delivery cost are considered separately. Two approaches are proposed to approximate the Pareto front of this bi-objective problem. Both methods are an adaptation of the hybrid grouping genetic algorithm. In [1] is modeled the problem with the aim of minimizing bi-objectives, namely the total system cost and risk-based transportation. This problem belongs to a

class of NP-hard ones. Then, a multi-objective imperialist competitive algorithm. Furthermore, the computational results are compared to show the performance of the algorithm.

Therefore, in this paper, we model the bi-objective Simple Link Problem using the technique VMI and the ϵ-constraint method to deal with the bi-criteria problem.

Thus, this paper is structured as follows. Section 2 presents the description of the problem and the mathematical model of the problem. In Sect. 3, the ϵ-constraint method is presented. Section 4 describes the tests carried out of the bi-objective approach described in Sect. 3 and their results. This paper is concluded with Sect. 5, where a conclusion and some remarks are made with respect to the work presented.

2 Description of the Problem and Mathematical Model

A set of products I is made available at an origin A and demanded at a destination B at a constant rate of q_i. A set of frequency, indexed by J, is considered and each frequency F_j corresponds to the delivery time $t_j = 1/F_j$, then, deliveries are made in discrete times. Trucks, with given capacity r_j, are available for shipping products, and each product can be partly shipped by trucks traveling at different frequencies F_j. Each product $i \in I$ with a unit of volume v_i can be continuously divided. It is assumed that, for each product, the production rate is equal to the demand rate. Two cost factors are considered, namely the transportation, c_j, and the inventory cost, h_i, (which is charged in the same way both at the origin and at the destination). All shipments having the same frequency are supposed to be simultaneous, i.e., performed at the same time. For each frequency, an unlimited number of trucks is available. The inventory level of any product $i \in I$ in the origin and the destination must be non-negative. The objective of this model is to decide the fraction of each product which has to be shipped at each frequency in such a way that the sum of the transportation and the inventory costs are minimized, i.e., find the delivery policy, which, in this case, its solution are the frequencies, the quantity with which the products are delivered in frequency F_j and, additionally, the number of routes required in frequency F_j.

The model presented in this subsection is based on [6] and [7]. A description of the problem is given, and the model considering the objective function to be minimizing the sum of the costs is presented as follows:

Parameters:

$I \rightarrow$ set of products that have to be shipped from an origin point A to a destination point B;
$J \rightarrow$ set of given frequencies;
$q_i \rightarrow$ constant rate at which product i is made available at the origin and absorbed at the destination;
$v_i \rightarrow$ volume of product i;
$h_i \rightarrow$ cost of inventory of product i;

$F_j \rightarrow j$-th frequency, with $j \in J$;
$t_j = 1/F_j \rightarrow$ period, discrete time in frequency F_j;
$r_j \rightarrow$ capacity of each truck traveling at frequency F_j;
$c_j \rightarrow$ cost of a single trip of a truck traveling at frequency F_j.

Decision variables:

$y_j \rightarrow$ number of vehicles to use at frequency j;
$x_{ij} \rightarrow$ percentage of product i to ship at frequency j.

Minimize

$$\sum_{i \in I} \sum_{j \in J} h_i \, q_i \, t_j \, x_{ij} + \sum_{j \in J} \left(\frac{c_j}{t_j} \right) y_j \tag{1}$$

Subject to:

$$\sum_{j \in J} x_{ij} = 1, \quad i \in I \tag{2}$$

$$t_j \sum_{i \in I} v_i \, q_i \, x_{ij} \leq r_j \, y_j, \quad j \in J \tag{3}$$

$$0 \leq x_{ij} \leq 1, \quad i \in I, \quad j \in J \tag{4}$$

$$y_j \geq 0, \quad y_j \in \mathbb{Z}, \quad j \in J \tag{5}$$

In objective function (1) we have $f = h_i \, q_i \, t_j \, x_{ij}$ related to minimize the average transportation cost and $g = (c_j/t_j) \, y_j$ related to minimize the inventory cost, both per unit time. The set of constraints (2) ensures that the products are fully delivered from the origin to the destination using one or more frequencies. The set of constraints (3) states that the number of vehicles used at each frequency is sufficient. And the constraints (4) and (5) define non-negative decision variables, the last being integer.

3 Bi-objective Approach

A bi-objective optimization problem involves two objective functions. In this paper, the SLP was formulated as follows:

$$\text{Minimize} \left\{ \sum_{i \in I} \sum_{j \in J} f \, , \sum_{j \in J} g \right\} \tag{6}$$

The objective function in (6), where $f = h_i\, q_i\, t_j\, x_{ij}$ is related to minimize the average transportation cost and $g = (c_j/t_j)\, y_j$ is related to minimize the inventory cost, both per unit time, aims at Pareto-optimal solutions that are both at minimum costs of inventory and transportation. In Sect. 3.1 we present the ϵ-constraint method that will be used to solve the bi-objective problem.

3.1 The ϵ-Constraint Method

The ϵ-constraint method was first proposed by Haimes et al. [11] and widely discussed in [8, 15]. In this method, one of the objective functions is selected to be optimized while the other(s) are converted into additional constraints, leading to the Pareto optimal solution. Systematic modification of the values of the objective functions forming the additional constraints leads to the generation of an evenly distributed Pareto frontier.

In this paper, we select the transportation cost as the objective and the inventory cost is converted into a constraint with the upper bound ϵ. Thus, the complete Mixed-Integer Programming (MIP) model with ϵ is given as follow:

$$\min \sum_{i \in I} \sum_{j \in J} f \tag{7}$$

Subject to:

$$\sum_{j \in J} g \leq \epsilon \tag{8}$$

Constraints (2)–(5)

For finding the ϵ value, the first step in solving the problem as mono-objective for the transportation cost without additional constraint, from the second step onwards is considered the constraints (8) with ϵ value the equal number of vehicles available, until it reaches a sufficiently small number of vehicles. With this, the number of vehicles decreases together with the transportation cost and the inventory cost increase. The results of this analysis will be shown in Sect. 4.

4 Computational Experiments

The algorithm was implemented in the C++ programming language and the computational experiments were performed on a machine with Intel® Core™ i7 − 4790 CPU @ 3.60 GHz ×8, 16 GB RAM and Ubuntu 18.04.1 as the operating

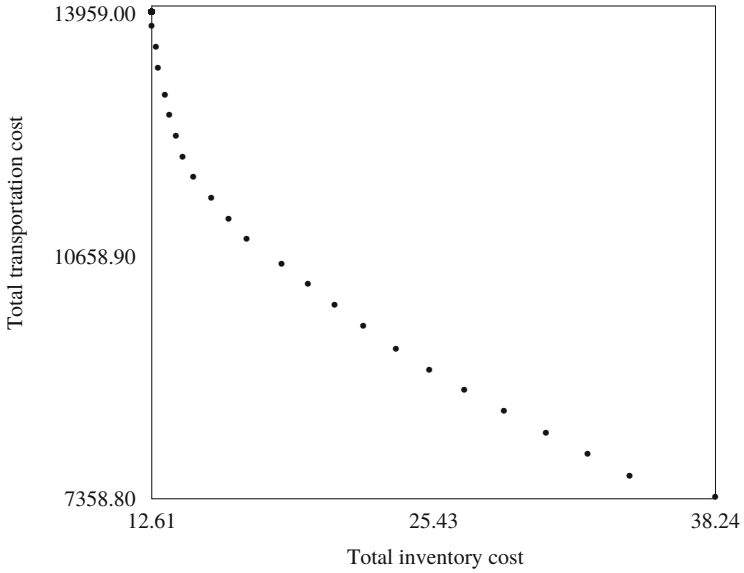


Fig. 1 Non-dominated points solution for the Pareto front

system. The MIP model was solved with the IBM ILOG CPLEX 12.7 considering its default settings. We impose a time limit of 3600 s to solve each instance as the stopping criterion.

The instances are strongly inspired by [7]. To wit:

- Number of products (set I): 10.
- Number of frequencies (set J): 5, 10 and 15.
- q_i: two sets randomly generated in two different intervals (0.1–5) and (5–100);
- v_i: randomly generated between 0.001 and 0.01;
- h_i: randomly generated between 0.001 and 1;
- R_j: randomly generated between 0.05 and 1.
- c_j: randomly generated between 275 and 325.

Figure 1 shows the Pareto front for an instance with 10 products and 5 frequencies. As expected, all computational experiments demonstrate that reducing the total transportation cost increases the total inventory cost, as can be seen in the Pareto front. It is worth mentioning that the inventory values are much lower than the routing values because we generate the instances randomly based on the values of [7]. Other works such as that of [10] have inventory values well above transportation costs. Therefore, we reinforce that dealing with a bi-objective problem in practice brings many advantages to managers.

Table 1 Points from the Pareto front for 30 instances

J	q_i	Ins.	MinF		Median		MinG	
			f	g	f	g	f	g
5	0.1–5	1	9.65	25,738.50	10.06	24,234.00	28.90	2234.42
		2	13.38	12,160.00	13.49	11,769.54	26.75	3553.50
		3	8.47	111,228.00	8.48	111,007.25	42.35	5999.60
		4	10.90	3920.00	11.34	3621.00	21.79	3090.00
		5	14.07	24,447.00	14.08	24,320.50	28.15	5562.00
5	5–10	6	41.39	30,056.00	42.50	29,846.40	206.93	14,235.20
		7	40.06	26,560.00	40.10	26,339.66	121.39	10,862.94
		8	37.92	28,910.00	38.18	28,848.00	38.18	28,848.00
		9	39.45	186,113.00	39.45	186,113.00	193.43	9960.65
		10	57.23	222,780.50	57.23	222,780.50	112.98	13,047.50
10	0.1–5	11	9.00	46,455.00	9.02	45,747.50	27.27	4130.28
		12	9.74	16,006.00	9.91	15,240.50	19.88	4672.82
		13	15.94	99,738.92	15.94	99,738.92	79.56	5167.10
		14	11.42	4768.00	14.21	4256.04	65.08	3609.48
		15	10.45	17,580.00	10.50	17,146.50	84.26	2776.96
10	5–10	16	30.50	9060.00	30.92	9053.00	30.92	9053.00
		17	33.18	31,284.00	33.18	31,122.00	165.88	9817.20
		18	43.06	14,280.00	60.02	13,318.83	130.50	12,340.35
		19	33.90	19,564.00	34.46	18,798.50	282.52	14,573.28
		20	41.92	12,810.00	58.61	11,711.68	299.47	10,038.88
15	0.1–5	21	13.88	2880.00	13.94	2688.03	13.94	2688.03
		22	16.49	55,476.00	16.65	54,562.74	50.56	5620.63
		23	8.55	4284.00	14.20	3950.96	94.99	3019.50
		24	14.29	7670.00	17.88	7149.63	84.44	4259.36
		25	19.54	31,000.00	19.57	30,749.00	39.78	2968.56
15	5–10	26	38.97	18,005.00	39.35	17,342.50	78.18	9292.71
		27	35.87	10,944.00	36.11	10,863.50	144.52	9509.12
		28	38.49	25,696.00	38.65	25,168.00	154.06	6673.66
		29	27.63	15,750.00	27.73	15,620.70	83.76	11,046.28
		30	36.92	25,160.00	36.97	25,015.00	112.15	7361.89

We generate a set of 30 instances following the parameters described above. For ease of comparison, we provide these instances in website.[1] In Table 1 the results of the 30 instances solved to optimality are presented, divided into six groups according to the frequency number.

Each line of the Table 1 has the results of one instance, showing points from the Pareto front. In the first column, we have the frequency number (J). In the second column the interval of the (q_i) the constant rate that product i is made available at the

[1]Website with instances: https://github.com/ariannesilvamundim/instanceods2019.

origin and absorbed at the destination. In the sequence we present three solutions, to know: (1) minimal value to f solution (MinF) and maximal value to g; (2) median solution between f and g from 100 points solution from the Pareto front (Median); and (3) minimal value to g solution (MinG) and maximal value to f. All instances were solved in less than 60 s.

The results of Table 1 show that for all instances the function f increases 291.62% in average with 203.46% in median, 55.94 in absolute value median. While the function g decreases 63.86% in average and 54.92% in median with absolute value median equal 13,068.11. This analysis shows the trade-off of the two functions, while one cost grows the other decreases. In addition, we can see the mean and median values are close. This indicates a symmetry of the distribution of the solutions of the instances, especially in the first five instances that have a standard deviation on the percentage variation of 6.80 to f and 0.35 to g.

5 Conclusion

In this paper, we presented a bi-objective mixed integer model for the SLP. We used the ϵ-constraint method to solve the bi-objective mixed integer model. The computational experiments show that applying the ϵ-constraint is efficient to construct the Pareto front. The computational experiments show that the proposed method is efficient to solve the bi-objective Single Link Problem and is suitable for small companies and managers.

This article presents preliminary results of the bi-objective study of SLP. Future studies will consider the bi-criteria problem with additional constraints in order to compare with the literature. In addition, we propose math-heuristics using the proposed method here to solve instances with a large number of items.

We make the instances based on the literature available in our website for comparison for others authors and future works.

Acknowledgement The authors would like to thank CAPES for their financial support.

References

1. Arab, R., Ghaderi, S.F., Tavakkoli-Moghaddam, R.: Solving a new multi-objective inventory routing problem by an imperialist competitive algorithm. Int. J. Transp. Eng. (2018). https://doi.org/10.22119/IJTE.2018.108299.1378
2. Arora, V., Chan, F.T.S., Tiwari, M.K.: An integrated approach for logistic and vendor managed inventory in supply chain. Expert Syst. Appl. **37**, 39–44 (2010)
3. Azuma, R.M.: Otimização multiobjetivo em problema de estoque e roteamento gerenciados pelo fornecedor, M.S. thesis, Universidade Estadual de Campinas, Campinas (2011)
4. Bell, W.J., et al.: Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. Interfaces **13**, 4–23 (1983)

5. Bertazzi, L., Speranza, M.G.: Continuous and discrete shipping strategies for the single link problem. Transp. Sci. **36**, 314–325 (2002)
6. Bertazzi, L., Speranza, M.G.: Inventory routing problems: an introduction. EURO J. Transp. Logist. **1**, 307–326 (2012)
7. Bertazzi, L., Speranza, M.G., Ukovich, W.: Exact and heuristic solutions for a shipment problem with given frequencies. Manag. Sci. **46**, 973–988 (2000)
8. Chankong, V., Haimes, Y.Y.: Multiobjective Decision Making: Theory and Methodology. Elsevier Science Publishing, New York (1983)
9. Franco, C., López-Santana E.R., Méndez-Giraldo, G.: A column generation approach for solving a green bi-objective inventory routing problem. In: Ibero-American Conference on Artificial Intelligence, vol. 10022, pp. 101–112 (2016)
10. Geiger, M.J., Sevaux, M.: The biobjective inventory routing problem: problem solution and decision support. In: Proceedings of the 5th International Conference on Network Optimization, pp. 365–378 (2011)
11. Haimes, Y., Lasdon, L., Wismer, D.: On a bicriterion formulation of the problems of integrated system identification and system optimization. IEEE Trans. Syst. Man Cybern. **3**, 296–297 (1971)
12. Hall, R.W.: Determining vehicle dispatch frequency when shipping frequency differs among origins. Transp. Res. B Methodol. **19**, 421–431 (1985)
13. Jackson, P.L., Maxwell, W.L., Muckstadt, J.A.: Determining optimal reorder intervals in capacitated production-distribution systems. Manag. Sci. **34**, 938–958 (1988)
14. Maxwell, W.L., Muckstadt, J.A.: Establishing consistent and realistic reorder intervals in production-distribution systems. Oper. Res. **6**, 1316–1341 (1985)
15. Miettinen, K.: Introduction to multiobjective optimization: noninteractive approaches. Multiobjective Optimization, pp. 1–26. Springer, Berlin (2008)
16. Muckstadt, J.A., Roundy, R.O.: Analysis of Multistage Production Systems. Logistics of Production and Inventory, vol. 4, pp. 59–131. Elsevier, North-Holland (1993)
17. Sadok, A., Teghem, J., Chabcoub, H.: Grouping genetic algorithms for a bi-objective inventory routing problem. Int. J. Multicriteria Decis. Making **3**, 256–276 (2013)
18. Speranza, M.G., Ukovich, W.: Minimizing transportation and inventory costs for several products on a single link. Oper. Res. **42**, 879–894 (1994)
19. Speranza, M.G., Ukovich, W.: An algorithm for optimal shipments with given frequencies. Naval Res. Logist. **43**, 655–671 (1996)
20. Standerski, N.: Aplicação de Algoritmos Genéticos Paralelos a Problemas de Grande Escala de VMI - Vendor Managed Inventory. In: XXV Simpósio Brasileiro de Pesquisa Operacional, pp. 1184–1195 (2003)
21. Znamensky, A.: Heurísticas para o problema de Distribuição com Estoques Geridos pelo Fornecedor, M.S. thesis, Escola Politécnica da Universidade de São Paulo, São Paulo (2011)

Models for Disassembly Lot Sizing Problem with Decisions on Surplus Inventory

Meisam Pour-Massahian-Tafti, Matthieu Godichaud, and Lionel Amodeo

Abstract We consider a single-product Disassembly Lot Sizing Problem with Disposal (DLSPD) which is a problem arising in the context of disassembly systems. This is the problem of determining the quantity and time of the returned products to be disassembled while satisfying the demand of their parts or components over a planning horizon. Disassembly operation generates several components simultaneously. And, the demands are independent and not balanced which can generate unnecessary surplus inventory during planning horizon. Aggregate formulation (AGG) can be used to model this problem by considering disposal decision. Linear-Programming (LP) relaxation of this model doesn't give very good lower bound, especially for the large size instances. We aim to improve lower bound of the problem. Facility Location-based formulation (FAL) and additional constraints (Valid Inequalities (VIs)) for the LP relaxation of AGG model are proposed. Computational results on generated test instances show that LP relaxation of FAL and AGG with additional constraints can obtain very strong lower bound within a very short computational time which is useful for the varied DLSPD (multi-level, multi-product, . . .).

Keywords Reverse logistics · Disassembly lot sizing problem · Inventory · Disposal option · Linear Programming (LP) relaxation · Inequalities

1 Introduction

Over the last decade, many companies who did not devote much time to consider and implement reverse logistics, have begun to pay considerable attention to it. This is because of increasing concerns over the environmental impacts of End-

M. Pour-Massahian-Tafti (✉) · M. Godichaud · L. Amodeo
Logistique et Optimisation des Systèmes Industriels (LOSI), Université de technologie de Troyes (UTT), Troyes, France
e-mail: Meisam.pour_massahian_tafti@utt.fr; matthieu.godichaud@utt.fr; lionel.amodeo@utt.fr

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_37

Of-Life (EOL) products and the residual value of their components. An efficient management of reverse logistics activities to obtain economic benefits have also been gaining significant attention in recent years [11, 20]. Among the various activities of reverse logistics, disassembly has become increasingly important as a central activity which connects the collection of EOL products at customer's location to the recycling centers of components in order to obtain residual value (for re-use, remanufacturing, recycling or even proper disposing of) [5]. Despite the positive impact of disassembly activities on the environment, economic gain between revenues and disassembly costs can be low. This prompted some researchers to formulate more effective and appropriate models to increase opportunities for cost savings and make disassembly operations more profitable.

In the present paper, we focus on the planning problem called disassembly scheduling in the literature [10]. Disassembly systems have special characteristics that make them challenging for planning decisions: (1) The product diverges into multiple demand sources of parts or components, (2) Independent demands between components that are not necessary well balanced, can generate unnecessary surplus inventory of parts or components, (3) In addition, disassembly operation generates all the components, simultaneously [2]. These features make disassembly scheduling problem different than other lot sizing problems [7]. The surplus inventory can be held to satisfy future demands or be disposed of in a conscious environmental way in real industrial cases. Disassembly lot sizing models with lost sales, purchasing (of components) or disposal options are interested to take into account the management of surplus inventory [6, 10, 11].

Since Gupta et Taleb research [7], few articles studied disassembly lot sizing problem, compared with the huge number of studies on the ordinary lot sizing problems. In general, disassembly scheduling can be classified by parts commonality/no parts commonality, single-product/multi-product, and capacitated/uncapacitated. For the basic case, i.e. single-product type without parts commonality, a reverse version of MRP is proposed in [7], further they extend to include parts commonality for disassembly of multiple product types [19]. In [13], a heuristic algorithm is studied in which an initial solution is obtained by the algorithm of Gupta and Taleb for the objective of minimizing various costs related with disassembly operations. Disassembly scheduling problem can be modeled with an Integer Programming (IP) model as in [14] for the case of single product with capacity constraints with considering various cost factors in the objective function. Some researches consider setup costs and inventory costs together in the objective function so that lot sizing decision have to be made. In this case, a lot sizing heuristic study is addressed to improve the solutions of reverse MRP algorithm [2]. A Mixed-Integer Programming (MIP) model for the problem with parts commonality is proposed in [10] and a multi-product problem in which LP relaxation based heuristic gives the good solutions in reasonable times is considered in [9]. Then a branch and bound algorithm is suggested in [12] that incorporates a Lagrangian heuristic to address a single product type without parts commonality. Recently, a capacitated single-item multi-period disassembly scheduling problem with random parameters which is formulated as a mixed-integer nonlinear program, is studied in [15].

In ordinary lot sizing problem, there are many studies that address different formulations for the problem to compare their advantages [3]. There are not a lot of works who consider different formulations for the problem of disassembly lot sizing with disposal. A single-product disassembly lot sizing problem with disposal decision is proposed in [18]. They propose a new model to consider disposal decision into disassembly lot sizing problem and two heuristics are presented for the DLSPD. They also extend this work by proposing new models to consider disposal decision [17]. They show that the proposed models can improve lower bound (LB) of the problem. LP relaxation of the model AGG gives a very weak LB for the DLSPD. By adding valid inequalities to AGG-LP relaxation model of lot sizing problem, it is possible to obtain very strong LB for the lot sizing problem [4]. Some authors introduced (l, S) inequalities and show that combining (l, S) inequalities with LP relaxation of the AGG formulation provides a complete polyhedral description of the convex hull of the uncapacitated single-item lot sizing problem [1, 4].

As far as we know, in disassembly lot sizing problem, there are not a lot of works taking into account disposal decision in order to handle the problem of surplus inventory accumulation. Motivated by above discussion, we are interesting to disassembly planning problem with the decisions taken for surplus inventory of parts or components. The motivation of this research is twofold. (1) We propose different models for the DLSPD to handle the issue of surplus inventory. We compare the lower bound of LP relaxation of each proposed models. (2) Moreover, we adapt the proposed valid inequalities for ordinary lot sizing problem into disassembly lot sizing problem. This is because the LB obtained by LP relaxation of AGG model is not very good. The reminder of the paper is organized as follows: After presenting the problem, we present two mathematical models as well as valid inequalities cuts as a way to drive strong lower bound for the DLSPD. We will then present computational results of new generated benchmark in Sect. 3.

2 Problem Statement

A new single-product disassembly lot sizing problem considering disposal decision with two-level disassembly structure is modeled in this section. An example of this structure is given in Fig. 1. The number in parenthesis is the yield of component when one unit of root item (A) is disassembled. The first level represents leaf items, while the second level represents root item (EOL Product). The DLSPD can be defined as follow: For a given disassembly structure, the problem is to determine the quantity and timing of disassembling each EOL product in order to satisfy the demand of leaf items over a planning horizon, while unnecessary surplus inventory of leaf items can be disposed of.

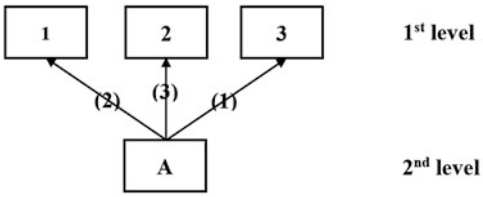


Fig. 1 An example of a two-level disassembly structure

The assumptions made in this paper are summarized as follows: (a) EOL product can be obtained whenever it is ordered and there is no holding cost for it; (b) backlogging and lost sales are not allowed, and hence demands should be satisfied on time; (c) demand of components are given and deterministic; (d) disassembled leaf items are considered of equal quality; (e) we assume, without loss of generality, the stock of the root and leaf items at the beginning of the planning horizon are zero. (f) we suppose that there is no disposal cost for the unnecessary leaf items. The following notations are used in this paper:

Indices

i Index for leaf items ($1 \ldots N$)
t Index for periods (T is the planning horizon)

Parameters

M_t Arbitrary big number considered in period t
s_t Setup cost of root item in period t
p_t Disassembly operation cost of root item in period t
a_i Number of unit of item i obtained from disassembling one unit of root item
h_{it} Inventory holding cost of leaf item i in period t
H_{ijt} Cumulative holding cost of leaf item i from period j to t ($j \leq t$)
d_{it} Demand of leaf item i in period t
D_{ijt} Cumulative demand of leaf item i from period j to t

Decision Variables

Y_t 1 if there is a setup in period t, and 0 otherwise
X_t Disassembly quantity of root item in period t
E_{it} Disposed quantity of leaf item i in period t
I_{it} Inventory level of leaf item i at the end of period t
Z_{ikt} Quantity of leaf item i disassembled in period k to satisfy demand of period t

The disposal decision is considered into the ordinary disassembly lot sizing problem using a new variable of disposed quantity of leaf items (E_{it}). The single-product disassembly lot sizing problem with disposal (DLSPD) can be formulated as a MIP model. Two formulations are proposed with considering disposal decision into the disassembly lot sizing problem. A natural formulation of the problem which

is called Aggregate formulation (AGG) is as follow:

$$\textbf{[P1]} \quad Min \ \{\sum_{t=1}^{T} s_t \cdot Y_t + \sum_{t=1}^{T} p_t \cdot X_t + \sum_{i=1}^{N}\sum_{t=1}^{T} h_{it} \cdot I_{it}\} \tag{1}$$

$$I_{it} = I_{it-1} + a_i.X_t - E_{it} - d_{it} \qquad \forall i = 1 \ldots N \,\&\, t = 1 \ldots T \tag{2}$$

$$X_t \leq M_t \cdot Y_t \qquad \forall t = 1 \ldots T \tag{3}$$

$$X_t \geq 0 \,\&\, integer \qquad \forall t = 1 \ldots T \tag{4}$$

$$E_{it} \geq 0 \qquad \forall i = 1 \ldots N \,\&\, t = 1 \ldots T \tag{5}$$

$$I_{it} \geq 0 \qquad \forall i = 1 \ldots N \,\&\, t = 1 \ldots T \tag{6}$$

$$Y_t = 0 \, or \, 1 \qquad \forall t = 1 \ldots T \tag{7}$$

Objective function (1) is to minimize the sum of setup, disassembly operation, and inventory holding costs over the whole T-period horizon. Constraints (2) are the inventory balance equations for the leaf items. Constraints (3) guarantee that a setup cost is performed in period t if any disassembly operation is done in that period. Constraints (4–7) impose the non-negativity and binary restrictions on the variables.

Disaggregate or Facility Location-based formulation (FAL) is commonly used in lot sizing problem because its LP relaxation provides for the uncapacitated problem an optimal solution in which setup variables are integer and it has stronger LBs for the capacitated lot sizing problem. An additional disassembly variable Z_{ikt} is considered, which corresponds to the quantity of leaf items i disassembled in period k to fulfill the demand of period t. The disaggregate formulation for the DLSPD is presented in [P2]:

$$\textbf{[P2]} \quad Min \ \{\sum_{t=1}^{T} s_t \cdot Y_t + \sum_{t=1}^{T} p_t \cdot X_t + \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{k=1}^{t} H_{ikt-1} \cdot Z_{ikt}\} \tag{8}$$

$$\sum_{k=1}^{t} Z_{ikt} = d_{it} \qquad \forall i = 1 \ldots N \,\&\, t = 1 \ldots T \tag{9}$$

$$Z_{ikt} \leq d_{it} \cdot Y_k \qquad \forall i = 1 \ldots N \,\&\, t = 1 \ldots T \,\&\, k \leq t \tag{10}$$

$$a_i \cdot X_t \geq \sum_{j=t}^{T} Z_{itj} \qquad \forall i = 1 \ldots N \,\&\, t = 1 \ldots T \tag{11}$$

$$X_t \geq 0 \,\&\, integer \qquad \forall t = 1 \ldots T \tag{12}$$

$$Z_{ikt} \geq 0 \qquad \forall i = 1 \ldots N \,\&\, t = 1 \ldots T \,\&\, k \leq t \tag{13}$$

$$Y_t = 0 \, or \, 1 \qquad \forall t = 1 \ldots T \tag{14}$$

Objective function (8) is to minimize the sum of setup, disassembly operation, and inventory holding costs over the whole T-period horizon. Constraints (9) represent that the demand of leaf items should be satisfied. Constraints (10) relate the disassembled quantity of leaf items (variable Z_{ikt}) to the binary setup variable Y_t. Constraints (11) express that the total quantity of leaf item i obtained in period t, after disassembly of product, will be delivered to satisfy the demand or will be disposed of. Constraints (12–14) define the domains of decision variables.

Much research has been devoted to obtain the tight LP relaxations and improve corresponding LBs. In particular, Valid inequalities (VIs) which reduce the volume of the linear relaxation solution space by cutting off irrelevant parts. We adapt the expression of the inequalities proposed by Pochet and Wolsey [16] for the DLSPD. The following constraints define the more general class of so-called (l,S) inequalities: For any $1 \leq l \leq T$, $L = \{1 \cdots l\}$, and $S \subseteq L$, the following inequalities are valid inequalities for the LP relaxation of the DLSPD:

$$\sum_{s \in S} a_i \cdot X_s - \sum_{s \in S} E_{is} \leq \sum_{s \in S} D_{isl} \cdot Y_s + I_{il} \qquad \forall i = 1 \ldots N \,\&\, 1 \leq l \leq T, S \subseteq L \tag{15}$$

By adding these constraints to the LP relaxation formulation of AGG, we can obtain a tight linear description of the DLSPD. The idea underlying constraints (15) is to compute a lower bound on the inventory level of a leaf item i at the end of a period l. In the computation experiments, we use a cutting-plane generation algorithm using separation algorithm to add most violated inequalities of family of Eq. (15). Since in practice the number of such inequalities is limited, the separation algorithm in this case is effectively performed by enumeration. Note that VIs cuts can not be used for the FAL formulation because of requiring inventory balance equation.

3 Computational Experiments

In this section, we discuss the results of computational tests to evaluate the effectiveness of the proposed models and methods. We randomly generate various instances of the problem. The Benchmark of [8] is adapted to obtain a more interesting average of cycle Time Between Order (TBO$\geq$2). We use the proposed formula of optimal TBO for a disassembly system by Godichaud and Amodeo [5] to modify the parameters (i.e. inventory holding). Note that the new data generator gives an average of TBO equal to 2.

For the tests, we generate 150 instances with different sizes, i.e. 10 instances for each combination of three levels of the number of children ($C \Rightarrow$ low ($L = 10$), medium ($M = 100$), large ($H = 1000$)), and five levels of the number of periods ($T = 10, 20, 30, 40, 50$). The yield, demand of each children, disassembly operation cost and setup cost are generated form $DU(1, 4)$, $DU(50, 250)$, $DU(38, 62)$ and $DU(2500, 3500)$, respectively. Also, inventory holding cost for each level of children (Low, Medium, High) are generated form $DU(0.3, 0.5)$, $DU(0.03, 0.05)$ and $DU(0.003, 0.005)$. Here, $DU(a, b)$ means the discrete uniform distribution with a rage of $[a, b]$. The proposed models P1 and P2 are MIPs, so we can use CPLEX solver to obtain optimal solution. Then, we propose LP relaxation approach to obtain a strong lower bound for the DLSPD. The LP relaxation of the MIP problem (P1&P2) is the problem that arises by removing the integrality constraint of each variable. After solving LP relaxation problem of P1 and P2 by using CPLEX solver, its solution quality and computational time will be compared with those obtained from other proposed methods. Afterwards, we apply cutting plane algorithm by using a separation algorithm to add the VIs cuts to the LP relaxation of model P1, in order to improve corresponding LBs.

All tests are run on a system with an Intel Core i7-7700T, 2.9 GHz, and 16 Go RAM on windows 10. We use CPLEX solver 12.8 to solve the problems. The test results are summarized in Tables 1 and 2, which show the performance of the AGG-MIP, AGG-LP relaxation, AGG-Valid Inequalities, FAL-MIP, and FAL-LP relaxation for the DLSPD. These results show that LP relaxation of FAL has a very strong LBs for the problem within very short computational time so that it can find the optimal solution for 84% of the problems. The proposed inequalities (AGG-Valid Inequalities) are very efficient at strengthening formulation P1 so that It can find the optimal solution for 80% of the problems. Comparison of the results obtained by LP relaxation of FAl and the proposed inequalities (AGG-Valid Inequalities) show that the proposed AGG-Valid Inequalities is faster than LP relaxation of FAL with overall average computational time of 42.47 (s) but both can obtain optimal or near optimal solution with the same overall average Gap of 0.01%. LP relaxation of AGG can solve the problems in a very short computational time (overall average of 0.42 (s)) but the overall average Gap is considerable (11.98%). In this case, the proposed inequalities work very well. For instance, for the problems

Table 1 Gap (%) of the proposed methods and models

	AGG model						FAL model		
	LP relaxation			Valid inequalities			LP relaxation		
C-T	$\underline{M}$	$\hat{M}$	$\overline{M}$	$\underline{M}$	$\hat{M}$	$\overline{M}$	$\underline{M}$	$\hat{M}$	$\overline{M}$
L-10	5.91	8.73	11.67	0.00	0.00	0.00	0.00	0.00	0.00
L-20	9.50	12.10	14.34	0.00	0.00	0.00	0.00	0.02	0.10
L-30	10.31	13.17	15.88	0.00	0.02	0.04	0.00	0.01	0.04
L-40	12.60	16.35	27.45	0.00	0.02	0.06	0.00	0.01	0.05
L-50	12.90	13.96	15.74	0.00	0.00	0.00	0.00	0.00	0.00
M-10	7.59	9.09	11.41	0.00	0.03	0.16	0.00	0.03	0.16
M-20	9.01	11.27	13.67	0.00	0.00	0.01	0.00	0.00	0.01
M-30	10.88	11.91	13.47	0.00	0.01	0.05	0.00	0.01	0.05
M-40	11.85	13.05	14.92	0.00	0.02	0.10	0.00	0.01	0.05
M-50	11.69	13.10	14.89	0.00	0.00	0.01	0.00	0.00	0.01
H-10	6.87	9.05	12.00	0.00	0.00	0.00	0.00	0.00	0.00
H-20	9.43	10.92	12.16	0.00	0.01	0.05	0.00	0.01	0.04
H-30	11.08	11.93	12.84	0.00	0.00	0.00	0.00	0.00	0.00
H-40	9.69	12.42	15.50	0.00	0.00	0.02	0.00	0.00	0.02
H-50	12.08	12.69	13.99	0.00	0.00	0.02	0.00	0.00	0.02
Avg.	10.09	11.98	14.66	0.00	0.01	0.03	0.00	0.01	0.04

with medium number of children and 40 periods, average Gap is reduced from 13.05% (for LP relaxation of AGG) to 0.02% (for AGG-Valid Inequalities). We note that Gap is the percentage deviation from the optimal solution obtained by MIP model (P1&P2).

4 Conclusion

The single-product disassembly lot sizing problem with considering disposal decision to the management of surplus inventory is investigated. Two new MIP models are proposed to consider disposal decision. The results showed that both LP relaxation and proposed inequalities are efficient at obtaining tight linear description of model and they obtained the strong LBs for the DLSPD with reducing overall computational time. As future study, it might be interesting to consider more complex product structure such as multi-product with part commonality and to use other methods to handle the issue of surplus inventory such as demand balancing by pricing methods.

Table 2 CPU seconds of the proposed methods and models

	AGG model									FAL model					
	MIP			LP relaxation			Valid inequalities			MIP			LP relaxation		
C-T	$\underline{M}$	$\hat{M}$	$\overline{M}$	$\underline{M}$	$\hat{M}$	$\overline{M}$	$\underline{M}$	$\hat{M}$	$\overline{M}$	$\underline{M}$	$\hat{M}$	$\overline{M}$	$\underline{M}$	$\hat{M}$	$\overline{M}$
L-10	0.03	0.05	0.09	0.01	0.01	0.01	0.00	0.00	0.00	0.02	0.02	0.02	0.01	0.01	0.01
L-20	0.09	0.14	0.20	0.01	0.01	0.01	0.01	0.02	0.03	0.05	0.07	0.17	0.02	0.02	0.03
L-30	0.23	0.55	0.84	0.01	0.01	0.01	0.06	0.06	0.07	0.09	0.19	0.42	0.05	0.06	0.09
L-40	0.65	1.21	4.28	0.01	0.01	0.01	0.12	0.15	0.16	0.22	0.38	0.66	0.08	0.11	0.18
L-50	0.57	1.02	1.58	0.01	0.01	0.01	0.14	0.20	0.26	0.24	0.40	0.66	0.13	0.18	0.25
M-10	0.11	0.15	0.21	0.01	0.01	0.02	0.41	0.71	1.18	0.22	0.32	0.43	0.08	0.12	0.20
M-20	0.86	1.38	2.78	0.03	0.03	0.40	4.05	5.88	7.97	1.14	1.41	2.05	0.71	0.82	0.92
M-30	2.84	5.60	7.65	0.04	0.04	0.06	2.15	2.90	3.48	3.34	4.50	7.53	1.97	2.39	2.85
M-40	6.81	9.96	13.96	0.04	0.05	0.06	3.05	4.24	7.94	9.85	14.17	23.80	5.07	6.55	9.10
M-50	6.55	11.56	15.46	0.06	0.08	0.10	6.43	7.87	9.89	18.13	25.33	35.82	7.49	12.09	14.69
H-10	1.99	2.79	5.35	0.17	0.23	0.27	5.96	10.56	14.85	5.04	6.18	7.77	2.41	3.39	5.13
H-20	27.21	74.54	178.33	0.55	0.57	0.67	21.12	33.61	42.68	38.41	48.99	75.86	18.25	21.39	25.04
H-30	129.00	258.34	335.56	1.00	1.20	1.48	59.71	130.31	174.68	129.49	152.79	183.67	58.13	73.54	101.61
H-40	277.15	416.91	698.73	1.82	2.03	2.16	96.50	146.58	322.66	319.08	602.63	1293.36	174.48	204.25	237.54
H-50	347.29	517.25	795.88	1.79	1.98	2.13	184.75	293.93	439.87	601.06	1242.21	3600.00	325.54	447.10	807.91
Avg.	53.43	86.76	137.39	0.37	0.42	0.47	25.63	42.47	68.38	75.09	139.97	348.95	39.63	51.47	80.37

Acknowledgements We thank the departmental council of Aube, along with the European Union (FEDER) for funding the project

References

1. Barany, I., Van Roy, T., Wolsey, L.A.: Uncapacitated lot-sizing: the convex hull of solutions. In: Mathematical Programming at Oberwolfach II, pp. 32–43. Springer, Amsterdam (1984)
2. Barba-Gutiérrez, Y., Adenso-Diaz, B., Gupta, S.M.: Lot sizing in reverse mrp for scheduling disassembly. Int. J. Prod. Econ. **111**(2), 741–751 (2008)
3. Brahimi, N., Dauzere-Peres, S., Najid, N.M., Nordli, A.: Single item lot sizing problems. Eur. J. Oper. Res. **168**(1), 1–16 (2006)
4. Brahimi, N., Absi, N., Dauzère-Pérès, S., Nordli, A.: Single-item dynamic lot-sizing problems: an updated survey. Eur. J. Oper. Res. **263**(3), 838–863 (2017)
5. Godichaud, M., Amodeo, L.: Economic order quantity for multistage disassembly systems. Int. J. Prod. Econ. **199**, 16–25 (2018)
6. Godichaud, M., Amodeo, L., Hrouga, M.: Metaheuristic based optimization for capacitated disassembly lot sizing problem with lost sales. In 2015 International Conference on Industrial Engineering and Systems Management (IESM), pp. 1329–1335. IEEE, Piscataway (2015)
7. Gupta, S., Taleb, K.: Scheduling disassembly. Int. J. Prod. Res. **32**(8), 1857–1866 (1994)
8. Kim, H.-J., Xirouchakis, P.: Capacitated disassembly scheduling with random demand. Int. J. Prod. Res. **48**(23), 7177–7194 (2010)
9. Kim, H.-J., Lee, D.-H., Xirouchakis, P., Züst, R.: Disassembly scheduling with multiple product types. CIRP Ann. Manuf. Technol. **52**(1), 403–406 (2003)
10. Kim, H.-J., Lee, D.-H., Xirouchakis, P.: Two-phase heuristic for disassembly scheduling with multiple product types and parts commonality. Int. J. Prod. Res. **44**(1), 195–212 (2006)
11. Kim, H.-J., Lee, D.-H., Xirouchakis, P.: Disassembly scheduling: literature review and future research directions. Int. J. Prod. Res. **45**(18–19), 4465–4484 (2007)
12. Kim, H.-J., Lee, D.-H., Xirouchakis, P., Kwon, O.: A branch and bound algorithm for disassembly scheduling with assembly product structure. J. Oper. Res. Soc. **60**(3), 419–430 (2009)
13. Lee, D.-H., Xirouchakis, P.: A two-stage heuristic for disassembly scheduling with assembly product structure. J. Oper. Res. Soc. **55**(3), 287–297 (2004)
14. Lee, D.-H., Xirouchakis, P., Zust, R.: Disassembly scheduling with capacity constraints. CIRP Ann. Manuf. Technol. **51**(1), 387–390 (2002)
15. Liu, K., Zhang, Z.-H.: Capacitated disassembly scheduling under stochastic yield and demand. Eur. J. Oper. Res. **269**(1), 244–257 (2018)
16. Pochet, Y., Wolsey, L.A.: Production Planning by Mixed Integer Programming. Springer Science & Business Media, New York (2006)
17. Tafti, M. P. M., Godichaud, M., Amodeo, L.: Models for the single product disassembly lot sizing problem with disposal. IFAC-PapersOnLine **52**(13), 547–552 (2019)
18. Tafti, M. P. M., Godichaud, M., Amodeo, L.: Single product disassembly lot sizing problem with disposal. pp. 135–140 (2019)
19. Taleb, K.N., Gupta, S.M., Brennan, L.: Disassembly of complex product structures with parts and materials commonality. Prod. Plan. Control **8**(3), 255–269 (1997)
20. Tian, X., Zhang, Z.-H.: Capacitated disassembly scheduling and pricing of returned products with price-dependent yield. Omega **84**, 160–174 (2019)

Learning Inventory Control Rules for Perishable Items by Simulation-Based Optimization

Remigio Berruto, Paolo Brandimarte, and Patrizia Busato

Abstract We consider an inventory control problem for quickly perishable items, such as fresh produce, at the retail store level, assuming a fixed shelf life. Demand is affected by both uncertainty and seasonality within the week, as sales feature a peak close to weekends. Another complicating factor is customer behavior and inventory issuing: In the case of a first-in-first-out (FIFO) pattern, older items are sold first, whereas a last-in-first-out (LIFO) pattern is more critical as newer items are sold first, which may increase scrapping. These and possibly other complicating factors make elegant mathematical modeling not quite feasible. Hence, we experiment with simulation-based optimization approaches integrating a discrete-time simulation model with direct search methods like simplex and pattern search. The approach is rather flexible, and learning simple rules has a definite advantage in terms of management acceptance. One aim is to compare simple order-up-to rules, based on overall available inventory, against more complex rules that take inventory age into account. Since more complex rules require more effort in implementation, it is important to understand under which circumstances their use is justified. We also want to study the effect of economic parameters, demand uncertainty and skewness, as well as FIFO/LIFO behavior. Preliminary computational experiments are reported, including a comparison with simple newsvendor-based heuristics.

Keywords Perishable inventory · Simulation-based optimization · Learning decision rules

R. Berruto · P. Busato
Dipartimento di Scienze Agrarie, Forestali e Alimentari, Università di Torino, Grugliasco, TO, Italy
e-mail: remigio.berruto@unito.it; patrizia.busato@unito.it

P. Brandimarte (✉)
Dipartimento di Scienze Matematiche, Politecnico di Torino, Torino, Italy
e-mail: paolo.brandimarte@polito.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_38

1 Introduction

Supply chain management (SCM) involves difficult inventory control decisions, subject to demand uncertainty. Complicating factors may be seasonality and obsolescence or perishability risk (see the reviews [6, 11]). Obsolescence risk is more and more significant because of the increasing pace in product innovation. Perishability is relevant, e.g., for fresh produce and blood. Here, we consider items with fast perishability, like certain varieties of fruit and vegetables. The concrete setting is a retail store of a large distribution chain, ordering items from a logistic platform in a hub-and-spoke network structure (see, e.g., [13] for a related problem). The supply efficiency is such that the delivery lead time is virtually zero (if we assume that items are ordered in the evening and reach the store by the next morning).

There are two possible objectives when dealing with inventory control: (1) Minimizing expected cost, subject to service level constraints; (2) Maximizing expected profit. The first objective is more relevant, e.g., in blood banks, or when item availability is needed to support sales of other items [10]. Here, we consider expected profit maximization. We do not consider risk measures, as the inventory management problem is repetitive in nature. Nevertheless, tight control is needed in order to avoid missing profit opportunities when stockout occurs, as well as losses due to scrapping perished items.

If the shelf-life and the ordering period are the same, the problem essentially boils down to the well-known newsvendor problem (see [3]), where the optimal ordered amount q^* satisfies the equation

$$F_D(q^*) = \frac{m}{m + c_u} = \frac{\gamma}{\gamma + 1}, \tag{1}$$

where: $F_D(x)$ is the cumulative distribution function of demand, which we assume continuous for simplicity, over the sales time window; m is the profit margin $m = p - c$, where p is the selling price and c is the purchase cost per item; c_u is the cost of unsold items, $c_u = c - p_u$, where p_u is the salvage value of a scrapped item (or a markdown price). The above equation links the problem economics to the service level, i.e., the probability $\mathrm{P}\{D \leq q^*\} \equiv F_D(q^*)$ of not having a stockout. What really matters, in defining the service level, is the ratio $\gamma = m/c_u$, i.e., the profit margin relative to the cost of unsold items. For instance, if demand distribution is symmetric and $\gamma = 1$, then q^* is just the expected value of demand, but the optimal order quantity might be significantly different in other cases.

In the newsvendor model, there is one ordering opportunity for an item with a limited time window for sales. This is one extreme of the spectrum of classical inventory control models; inventory control policies, such as continuous-review (Q, R) or periodic-review (s, S), are the opposite extreme, where infinite ordering opportunities for items with unlimited life are considered. In this paper, we consider a case where items are delivered each day and the shelf-life is 3 days. This is an interesting setting, as it is an intermediate but challenging point between the two extremes above. Further complicating issues that we address here are demand

seasonality within a week (sales on Saturdays are much larger than sales on Mondays) and customer behavior. When items with different residual shelf-life (RSL) are available, customers may choose one according to different patterns, such as first-in/first-out (FIFO, an item with minimal RSL is selected, i.e., the most mature item) or last-in/first-out (LIFO, an item with maximal RSL is selected). This may also be the result of an inventory issuing policy, rather than customers' preferences. We do not consider random behavior in this paper, which is of course of interest (as we discuss in the last section) as we want to analyze the impact of this factor by considering the two extreme cases. The newsvendor rule does not apply here, even though it might be used as a heuristic, and we have to look for different rules. A simple base-stock policy relies on a set of order-up-to levels, one per day. Alternative policies may use information about the age of available stock, and they may perform better than simple-minded rules. However, their comparative advantage should be assessed, given the increased implementation effort at retail store level.

Within this general framework, the (limited) aims of the paper are: (1) To develop insights about the factors that may affect the choice of an inventory management policy and the resulting profit. (2) To compare different inventory control rules in terms of effectiveness, ease of learning, and informational requirements. Emphasis is on managerial insight; hence, we do not investigate in depth different optimization strategies that could be used.

2 Problem Statement and System Dynamics

Here we list the basic assumptions behind our study:

- We deal with a single retail store, where orders for fresh produce are issued each day, with the exception of Sundays, when the store is closed.
- Since a huge mix of different products (perishable and not) are delivered through the transit point, transportation charges are shared among many items. Hence, we do not consider this cost component.
- The product we consider has a quite short shelf-life of $L = 3$ days. Due to the limited shelf-life, we do not consider inventory holding cost.
- We assume that the objective is to maximize long-run average profit.
- Replenishment orders are issued at the end of each business day, and the new items are on the shelves, with remaining shelf-life L, just before the store opens the next day.
- We consider two extreme patterns of consumer behavior: first-in/first-out (FIFO) and last-in/first-out (LIFO).
- Demand is uncertain and subject to daily seasonality. We assume a multiplicative seasonality model, where β_k is the seasonal factor of weekday $k = 1, \ldots, 6$; $k = 1$ corresponds to Mondays and $k = 6$ to Saturdays. A typical weekly pattern is shown in Table 1.

Table 1 Weekly demand pattern

Weekday ($k = 1, \ldots, 6$)	Mon	Tue	Wed	Thr	Fri	Sat
Seasonality factor (β_k)	0.7500	0.8333	0.8333	0.8333	1.0833	1.6667

To describe (and simulate) the evolution of the system under control, we need a state variable representing the inventory in terms of amount available for each value of remaining shelf-life. The time bucket in our case corresponds to a single day. For instance, let $\mathbf{I}_t^m$ be a vector corresponding to inventory state at the beginning of day t (the superscript m stands for *morning*), after receiving the quantity ordered and before opening the retail store. This vector consists of L components,

$$\mathbf{I}_t^m = \left[I_{t,1}^m,\ I_{t,2}^m,\ \cdots,\ I_{t,L}^m\right],$$

where $I_{t,l}^m$ is the inventory level of items with remaining shelf-life l, $l = 1, \ldots, L$. The system state variable evolves as follows:

$$\mathbf{I}_t^m \quad \overset{\text{sell}}{\Longrightarrow} \quad \mathbf{I}_t^{e^-} \quad \overset{\text{scrap}}{\Longrightarrow} \quad \mathbf{I}_t^{e^+} \quad \overset{\text{order}}{\Longrightarrow} \quad \mathbf{I}_{t+1}^m, \tag{2}$$

where all vectors consist of L components and:

- $\mathbf{I}_t^{e^-}$ is the inventory at the end of day t (the superscript e stands for *evening*), after sales and *before* scrapping items.
- $\mathbf{I}_t^{e^+}$ is the inventory *after* scrapping $I_{t,1}^{e^-}$ items. The RSL of remaining items is updated by the following up-shift:

$$\mathbf{I}_t^{e^+} = \left[I_{t,1}^{e^+},\ I_{t,2}^{e^+},\ I_{t,3}^{e^+},\ \cdots,\ I_{t,L-1}^{e^+},\ I_{t,L}^{e^+}\right] = \left[I_{t,2}^{e^-},\ I_{t,3}^{e^-},\ I_{t,4}^{e^-},\ \cdots,\ I_{t,L}^{e^-},\ 0\right].$$

- Ordering decisions are based on $\mathbf{I}_t^{e^+}$. Let Q_t be the amount ordered in the evening of time bucket t; in the morning of time bucket $t+1$ the inventory state is

$$\mathbf{I}_{t+1}^m = \left[I_{t+1,1}^m,\ I_{t+1,2}^m,\ \cdots,\ I_{t+1,L-1}^m,\ I_{t+1,L}^m\right] = \left[I_{t,1}^{e^+},\ I_{t,2}^{e^+},\ \cdots,\ I_{t,L-1}^{e^+},\ Q_t\right]$$

  The received items fill the bottom-level position in the state vector. On Saturday evenings, the up-shift from $\mathbf{I}_t^{e^-}$ to $\mathbf{I}_t^{e^+}$ takes place as well, but no item is ordered.

An important issue is whether such a detailed information about the inventory state is available when making ordering decisions. Inventory deterioration due to manipulation can be difficult and costly, if not impossible, to assess during a business day, especially for items that are not packaged and labeled. Even if items are properly packaged and labeled, we may only have information on the total number of items in stock. Nevertheless, it is quite useful to assess the value of this information by comparing control policies that use detailed and exact state information against simpler policies that just use aggregate information.

3 Learning Ordering Rules by Simulation-Based Optimization

It is well-known [1] that the optimal policy for a discrete-time inventory model where demands are identically and independently distributed is a periodic-review, order-up-to (base-stock) policy with target level S. In our problem, this need not be the case, but we may consider variations of order-up-to policies because of their simplicity and intuitive appeal. Hence, we may consider ordering an amount

$$Q_t = \max\left\{S_k - \sum_{l=1}^{L-1} I_{tl}^{e+}, 0\right\}. \tag{3}$$

We are going to refer to this rule as `SixS`, since we assume a retail store operating 6 days per week and we use a different S_k for each one. The `SixS` rule uses information about scrapped items, but not about the RSL of items on-hand. If we assume that detailed information about the RSL of each individual item is available, we could devise rules exploiting such an additional knowledge (see, e.g., [12]). One possibility is a linear rule that, for $L = 3$, looks like

$$Q_t = \max\left\{S_k - a_{k1} I_{t,1}^{e+} - a_{k2} I_{t,2}^{e+}, 0\right\}.$$

Now we have to learn $6 \times 3 = 18$ coefficients; hence, we refer to this rule as `Linear18`. Clearly `SixS` is obtained if all the coefficients are set as $a_{k,l} = 1$. Hence, if we are able to find the optimal setting of coefficients, `Linear18` cannot perform worse than `SixS`. In related settings, age-based policies have been shown to perform better than pure inventory-based policies [4, 10]. Needless to say, the more coefficients we use, the more difficult is setting them optimally. A more relevant point, arguably, is the increased effort in implementing the rule. In order to compare `SixS` and `Linear18` against some sensible benchmark, we also consider alternative newsvendor-based policies. These policies are obtained by setting levels S_k in `SixS` on the basis of quantiles of the demand distribution over some reference period, using the logic of Eq. (1). If the inventory level is set by considering only the next day, we obtain policy `News1`. In policy `News3`, we consider the demand distribution over the next 3 days, with some adjustments to account for Sundays.

To set the coefficients of `SixS` and `Linear18` policies, we resort to simulation-based optimization (see, e.g., [5] for an overview). We use a simulator as a black box estimating the expected average profit over some reference period (day or week, it is really inconsequential); a search engine aims at setting coefficients so as to maximize expected profit. We have experimented with derivative-free algorithms such as simplex search, pattern search, simulated annealing, and genetic algorithms. These are not the only possibilities, as we could use particle swarm optimization or more refined alternatives based on metamodels. However, our aim is to understand the comparative advantage of rule structures and the impact of problem features,

rather than comparing solution algorithms. Dynamic programming is proposed in [7], and [10] uses an analytical modeling approach. However, these strategies are limited to problems with a simple structure. On the contrary, simulation-based optimization is less elegant but more flexible; see, e.g., [4, 8] for applications to perishable items, and [9] for a more general survey for inventory control problems.

4 Computational Experiments

We have addressed the following questions: How do the `SixS`, `Linear18`, `News1`, and `News3` rules perform in both absolute and relative terms? Can we learn good settings of rule coefficients efficiently and effectively? What is the impact of problem features? We consider: LIFO vs. FIFO customer behavior; symmetric vs. skewed demand, for a given level of variability; high vs. low profit margins.

Let $\overline{\pi}$ be the average profit of a policy over a long out-of-sample, test scenario. Rather than just comparing rules in relative terms, we use the ideal profit π^* that would be obtained on the same test scenario, when assuming perfect demand information. Thus, we evaluate a policy by considering the (percentage) performance ratio ρ defined as

$$\rho \equiv \frac{\overline{\pi}}{\pi^*}. \tag{4}$$

In an easy problem setting the performance ratio should be close to 100%, whereas in a difficult setting, we might fail short of obtaining such a satisfactory performance.

In order to check the difficulty of learning a rule, we also compare the performance predicted in-sample, on a relatively short learning sample (scenario), against the actual one on the test sample. Let $\overline{\pi}_L$ and $\overline{\pi}_T$ denote the average profits in the learning and test samples, respectively. The reliability of the learning horizon is evaluated by the following performance error measure:

$$\epsilon_l \equiv \frac{|\overline{\pi}_L - \overline{\pi}_T|}{|\overline{\pi}_T|}. \tag{5}$$

After some experimentation, we have used a learning horizon of 30 weeks. As to the test horizon, we have used a single test horizon of $200 \times 30 = 6000$ weeks. Furthermore, when learning and testing different rules, we reset the state of the random number generators in such a way that the same scenarios are always used, which amounts to use variance reduction by common random numbers [2].

4.1 Problem Instance Features

The first problem feature is related to FIFO/LIFO behavior. We should expect that FIFO behavior is much easier to deal with in terms of profit performance. The second feature is the distribution of daily demand, which we assume continuous for the sake of simplicity. As we pointed out before, we consider a multiplicative demand model, with factors given in Table 1. We generate demand D_t for time bucket t on the basis of some probability distribution and multiply it by the seasonality factor β_k, where $k = 1, \ldots, 6$ is the weekday corresponding to time bucket t. We do not consider either autocorrelation or any other form of dependency in demand over time. Our base case is a normal distribution, with expected value 200 and standard deviation 60, i.e., $\mathsf{N}(200, 60^2)$. In order to induce some left- and right-skewness, we could resort to a beta or a gamma distribution. However, a gamma distribution is associated with right-skewness only, and a beta distribution has only two parameters, which means that we cannot play with both variability and skewness at will. Hence, we resorted to simple mixtures of normal distributions:

- To generate right-skewed demand, we sample a $\mathsf{N}(300, 60^2)$ variable with probability 0.9 or a $\mathsf{N}(600, 100^2)$ variable with probability 0.1; the coefficient of variation for this distribution is 0.34, similar to that of $\mathsf{N}(200, 60^2)$, and skewness is 1.79.
- To generate right-skewed demand, we sample a $\mathsf{N}(650, 120^2)$ variable with probability 0.85 or a $\mathsf{N}(300, 100^2)$ variable with probability 0.15; the coefficient of variation for this distribution is 0.29, and skewness is -0.65.

Note that skewness is not affected by a multiplicative seasonality factor. When applying `News1` and `News3` policies, we need quantiles of a skewed distribution, which we estimate by straightforward Monte Carlo sampling. The third feature refers the economic parameters, which influence the γ ratio in Eq. (1). We have experimented with two quite different (and somewhat extreme) settings:

- The high profit margin problem setting is characterized by a purchase cost of 4 per unit, a sales price of 12, and a markdown price of 2.
- The low profit margin problem setting is characterized by a purchase cost of 8 per unit, a sales price of 10, and a markdown price of 0 (no salvage value).

4.2 Experiment 1: Comparison of Alternative Search Strategies

In the first experiment, we want to compare the results of the four optimization algorithms in order to check their viability (simplex search—Splx; pattern search—Pat; simulated annealing—SA; genetic algorithms—GA). Since our main interest is in the impact of problem features, rather than solution algorithms, we always used default settings from MATLAB's Global Optimization Toolbox. Clearly, performance would be affected by changing the termination conditions. We have

Table 2 Experiment 1: Performance ratio and CPU time (seconds) for different problem settings and search algorithms

	Performance ratio				CPU time			
	Splx	Pat	SA	GA	Splx	Pat	SA	GA
High FIFO	98.61%	98.63%	98.63%	98.67%	23.53	32.04	320.45	48.20
Low FIFO	93.67%	94.66%	94.03%	94.60%	16.30	33.97	289.11	47.96
High LIFO	95.50%	95.43%	95.57%	95.66%	55.01	23.16	189.36	47.55
Low LIFO	81.21%	80.51%	81.05%	79.07%	35.90	54.00	269.68	54.91

Table 3 Experiment 1: Performance evaluation error for different problem settings and search algorithms

	Splx	Pat	SA	GA
High FIFO	1.9%	1.9%	1.9%	2.1%
Low FIFO	0.34%	1.3%	0.72%	1.4%
High LIFO	1.2%	1.6%	1.5%	1.5%
Low LIFO	1.2%	2.0%	1.5%	3.6%

used these algorithms to learn a `SixS` rule in the normal demand case, in the four combinations of high/low profit margin and FIFO/LIFO behavior. To initialize the algorithms, we set S_k to the expected demand on weekday k; the initial population of genetic search is created by randomly generating values of S_k between 50% and 200% of expected demand. In Table 2, we report the performance ratio and the CPU time for each problem setting. The table suggests that, indeed, problem features have a significant impact on the performance ratio. In the nice setting, we get close to the ideal profit, but, on the other end of the spectrum, low profit margin and LIFO behavior make a much more difficult problem setting. In terms of solution quality, there seems to be no clear winner, but there is a difference in terms of CPU time. The point, actually, is not the time in itself, but the relative performance, showing that simulated annealing looks definitely less efficient. We also observe that, in the case of the `SixS` rule, the global search abilities of stochastic search do not look very helpful. Finally, in Table 3 we list the performance evaluation error as defined in Eq. (5). The errors are fairly limited, with the possible exception of the 3.6% error incurred by genetic search in the low profit/LIFO case. This suggests that selecting a suitable learning scenario is important.

4.3 Experiment 2: Comparing Decision Rules

In the second experiment, we aim at comparing `SixS` against `News1` and `News3` policies. In Table 4 we report the performance ratio for the case of a normal distribution. For the sake of brevity, we do not report the results for right- and left-skewed mixtures, which are quite similar (arguably, due to rather limited amount of skew). The results have been obtained by learning the `SixS` policy with simplex search. Once again, high profit/FIFO is a rather easy problem setting, whereas low

Table 4 Experiment 2: Performance ratio for different rules in the case of normal demand

	Six6	News1	News3
High FIFO	98.61%	96.29%	95.00%
Low FIFO	93.67%	71.10%	85.44%
High LIFO	95.50%	94.93%	85.29%
Low LIFO	81.21%	70.56%	−26.77%

profit/LIFO is tough to deal with. What is certainly striking, even though not quite surprising, is that in the most difficult setting `News3` results in a negative profit. In fact, the `News3` policy maintains a much larger inventory than the other rules. In fact, we have observed that `News1` orders the smallest amounts, whereas `News3` orders most, which is quite dangerous in this problem setting. This supports the view that a bit of sophistication helps, and that the problem is not trivial for certain feature settings.

4.4 Experiment 3: Is a More Complicated Policy Warranted?

As a final experiment, we have checked whether there is any advantage in learning a more complicated policy, like `Linear18`. After some experimentation we found that the following learning strategy is the most efficient:

- Learn the coefficients of a `SixS` policy with simplex (or pattern) search.
- Use the S_k coefficients to initialize an equivalent `Linear18` policy, where $a_{k1} = a_{k2} = 1$.
- Try improving the `Linear18` policy by genetic search.

The results are given in Table 5. We only illustrate the easiest and the most difficult problem settings (FIFO.High and LIFO.Low) for normal, left-skewed, and right-skewed demand.

- The columns err.Sixs and err.Lin18 give the performance evaluation errors of the `SixS` and `Linear18` policy, respectively.
- The columns impr.opt and impr.test give the percentage improvement of the performance of the `Linear18` over the `SixS` policy, respectively, as predicted

Table 5 Experiment 3: Checking the improvement of `Linear18` with respect to `SixS`

	err.Sixs	err.Lin18	impr.opt	impr.test	ratio.SixS	ratio.Lin18
Norm.FIFO.High	1.9%	1.1%	−0.016%	−0.80%	98.61%	97.82%
Norm.LIFO.Low	1.2%	7.0%	3.8%	−1.9%	81.21%	79.65%
Left.FIFO.High	0.49%	0.62%	0.037%	−0.097%	97.05%	96.95%
Left.LIFO.Low	4.1%	5.1%	0.12%	−0.89%	83.95%	83.20%
Right.FIFO.High	0.76%	0.44%	−0.17%	−0.49%	98.96%	98.47%
Right.LIFO.Low	4.0%	5.0%	−1.4%	−2.4%	84.06%	82.09%

in terms of objective function value from the optimization problem and evaluated over the test horizon.
- The columns ratio.SixS and ratio.Lin18 give the performance ratio of the two policies on the test sample.

The bottom line is clear from the last two columns, suggesting that there is no apparent gain from using the more complicated rule. The data in the previous columns help to understand why. The second line shows a possibly disturbing pattern. In the low profit/LIFO case with normal demand, we do improve the objective function on the learning sample (by 3.8%), but in the test sample we obtain a worse performance from the more sophisticated rule. This is due to a large evaluation error (7.0%) between learning and test sample, whereas this error is only 1.2% with `SixS`. This weird pattern can be explained in terms of overfitting the policy coefficients to the learning scenario.

5 Critical Remarks and Directions for Further Research

Clearly, no definitive conclusion can be drawn on the basis of such a limited set of experiments. The basic messages are that simulation-based optimization is an effective and flexible way to tackle the class of problems we consider, and that we should not take for granted that an increased sophistication in decision rules, using age information, is worth the price of a more difficult implementation. However, as shown in [10], this could be the case when dealing with non-zero lead times. Further issues are related with the plausibility of the learned rules. It often happens that different settings have a similar performance, but they may not be equivalent in terms of user acceptance (using quite different target levels, especially for similar days). To avoid this issue, we may use regularization terms shaped after ridge and lasso regression. A further issue is related with robustness, i.e., uncertainty about the model itself. Consider random customer behavior, i.e., a random mix of LIFO and FIFO patterns according to some probability model: What about the uncertainty of the involved parameters (e.g., the fraction of LIFO vs. FIFO customers)? And what about uncertainty about seasonal factors? We are investigating the use of a worst-case robust framework, which may need more sophistication in the learning procedures, possibly based on metamodeling approaches.

References

1. Bertsekas, D.P.: Dynamic Programming and Optimal Control, vol. I, 4th edn. Athena Scientific, Belmont (2017)
2. Brandimarte, P.: Handbook in Monte Carlo Simulation: Applications in Financial Engineering, Risk Management, and Economics. Wiley, Hoboken (2014)
3. Brandimarte, P., Zotteri, G.: Introduction to Distribution Logistics. Wiley, Hoboken (2007)

4. Duan, Q., Liao, T.W.: A new age-based replenishment policy for supply chain inventory optimization of highly perishable products. Int. J. Prod. Econ. **145**, 658–671 (2013)
5. Fu, M.C. (ed.): Handbook of Simulation Optimization. Springer, New York (2015)
6. Goyal, S.K., Giri, B.C.: Recent trends in modeling of deteriorating inventory. Eur. J. Oper. Res. **134**, 1–16 (2001)
7. Haijema, R.: A new class of stock-level dependent ordering policies for perishables with a short maximum life. Int. J. Prod. Econ. **143**, 434–439 (2013)
8. Haijema, R., Minner, S.: Stock-level dependent ordering of perishables: a comparison of hybrid base-stock and constant order policies. Int. J. Prod. Econ. **181**, 215–225 (2016)
9. Jalali, H., van Nieuwenhuyse, I.: Simulation optimization in inventory replenishment: a classification. IIE Trans. **47**, 1–19 (2015)
10. Minner, S., Transchel, S.: Periodic review inventory-control for perishable products under service-level constraints. OR Spectr. **32**, 979–996 (2010)
11. Nahmias, S.: Perishable inventory theory: a review. Oper. Res. **61**, 680–708 (1982)
12. Tekin, E., Gurler, U., Berk, E.: Age-based vs. stock level control policies for a perishable inventory system. Eur. J. Oper. Res. **134**, 309–329 (2001)
13. van Donselaar, K., van Woensel, T., Broekmeulen, R., Fransoo, J.: Inventory control of perishables in supermarkets. Int. J. Prod. Econ. **104**, 462–472 (2006)

A Genetic Algorithm for Minimum Conflict Weighted Spanning Tree Problem

Carmine Cerrone, Andrea Di Placido, and Davide Donato Russo

Abstract The Minimum Conflict Weighted Spanning Tree Problem is a variant of the Minimum Spanning Tree Problem in which, given a list of conflicting edges modelled as a conflict graph, we want to find a weighted spanning tree with the minimum number of conflicts as main objective function and minimize the total weight of spanning trees as secondary objective function. The problem is proved to be NP-Hard in its general form and finds applications in several real-case scenarios such as the modelling of road networks in which some movements are prohibited. We propose a genetic algorithm designed to minimize the number of conflict edge pairs and the total weight of the spanning tree. We tested our approach on benchmark instances, the results of our GA showed that we outperform the other approaches proposed in the literature.

Keywords Minimum Conflict Weighted Spanning Tree · Genetic algorithm · Conflict graph

1 Introduction

The classical minimum spanning tree (MST) problem and its generalizations, can be applied to many real-world application scenarios. The minimum spanning tree problem with conflicts on edge pairs (MSTC) is an NP-hard variant of the classical MST problem in which are considered incompatibility between pairs of edges. There are several real-world applications for this problem: model the connection

C. Cerrone
University of Genova, Genova, Italy
e-mail: carmine.cerrone@unige.it

A. Di Placido (✉) · D. D. Russo
University of Molise, Campobasso, Italy
e-mail: a.diplacido2@studenti.unimol.it; d.russo7@studenti.unimol.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_39

layout of wind turbines, where we want to minimize the number of connections among them avoiding overlapped cables [9]; solve a quadratic bottleneck spanning problem [17]; model a road network in which some movements are prohibited [8], and install an oil pipeline system connecting several countries [5].

Incompatibility between pairs of edges means that, given the set of conflicting pairs, at most one takes place in the spanning tree. Darmann et al. [5] exposes that the conflict relations can be represented in an undirected *conflict graph*, where each vertex uniquely identifies an edge of the original graph and each edge in the conflict graph implies the incompatibility between two edges. Thus we want to identify the conflict-free MST of minimum cost, given a connected, undirected and weighted graph and a set of conflicting edges pairs.

The problem was introduced by Darmann et al. [5, 6], describing the MST variant as NP-hard. Other authors studied the variant in several optimization problems such as minimum cost perfect matching problem with conflict pair constraints [11], knapsack problem with conflict graphs [13], maximum flow problem with disjunctive constraints [14], and bin packing with conflicts [13, 15].

Regarding the MSTC resolution, Zhang et al. [17] presented several meta-heuristic approaches to solve the problem. They were able to produce, if possible, the MST conflict free or the MST with conflict of minimum cost. Samer and Urrutia [16] proposed a branch-and-cut algorithm for the exact solution of the problem. The authors exploited an equivalent definition of the conflicting edge pairs using an auxiliary conflict graph and provided two sets of inequalities for both of the spanning tree and the stable set polytopes. Recently another branch-and-cut algorithm for MSTC was introduced by Carrabs et al. [4]. Thanks to a new set of three classes of valid inequalities, based on combined properties belonging to any feasible solution, the last algorithm outperforms the ones proposed by Samer and Urrutia [16]. Finally Carrabs et al. [1] exposed a multi-ethnic genetic algorithm (MEGA) with three local search procedures. The results obtained by MEGA on benchmark instances outperforms the other heuristic approaches proposed in the literature, for this problem.

In this work, we face a variant of the MSTC, proposed by Zhang et al. [17], named Minimum Conflict Weighted Spanning Tree problem (MCWST). Given an undirected graph $G = G(V, E)$ and a set $C \subseteq E \times E$ of conflicting edge pairs, the primary goal consists in finding a spanning tree T of G having the minimum number of conflicts edge pairs and, in case of conflict-free solution, the secondary goal consists in finding spanning tree of G with minimum weight. We propose a genetic algorithm approach for the MCWST and test its effectiveness and performance on the benchmark instances proposed by Samer and Urrutia [16]. Our results have been compared to the ones obtained by the multiethnic genetic approach of Carrabs et al. [1].

2 Problem Description and Definitions

Let $G = G(V, E, C)$ be an undirected weighted graph, where V is the set of vertices, E is the set of edges with weight $w(e), \forall e \in E$ and $C \subseteq E \times E$ is the set of *conflicting edge pairs*.

A *spanning tree* T of a connected graph G is a set of edges $E_T \subseteq E$ such that $|E_T| = |V| - 1$, and the subgraph induced from G by E_T is connected. The weight of T is denoted by $W(T)$ and it is the sum of all $w(e), \forall e \in E_T$. The conflict edges set is formally defined as follows:

$$C = \{(e_i, e_j) : e_i \in E, e_j \in E, e_i \text{ and } e_j \text{ are in conflict}\}$$

This means that $\forall (e_i, e_j) \in C$ only one belongs to E_T. Since the couples in C are not ordered, $\{e_i, e_j\} = \{e_j, e_i\}$. Furthermore, $\forall e_k \in E$ we define $C(e_k)$ the set of conflicts induced by e_k and $C(E_T)$ the set of conflicting edge pairs contained in T. When $|C(E_T)| = 0$, we say that T is *conflict-free*.

An equivalent definition which we exploit here uses the concept of conflict graph $G^C(E, C)$: by denoting each edge in the original graph as a node in G^C, we represent each conflict constraint by an edge connecting the corresponding nodes in G^C [16].

The *minimum spanning tree problem with conflict constraints* (MSTC) consists in finding a minimum weighted spanning tree T of G which is conflict-free. The MSTC problem is infeasible if no conflict-free solution T exists for it.

In this work we face a variant of the MSTC, proposed by Zhang et al. [17], named Minimum Conflict Weighted Spanning Tree problem (MCWST), in which the main goal is to find an MST of G where $C(E_T)$ is minimum and if $|C(E_T)| = 0$, the secondary goal is to minimize the total weight of T.

3 Genetic Algorithm

In this section, we expose our genetic algorithm (GA) designed to solve the MCWST problem. Genetic algorithms, introduced by Holland [7], are a family of metaheuristics based on the natural selection theory of Darwin. Starting from an initial population formed by a group of individuals, known as chromosomes, it evolves over the generations through crossover and mutation operators. In literature, there is already MEGA, proposed by Carrabs et al., that provides excellent results for the selected problem [1]. Despite this, there are several dissimilarities between our approach and the one recently proposed. The main difference concerns the population: while MEGA works on a set of parallel populations to provide the multi-ethnicity concept, our GA operates on a single set of individuals. Moreover, MEGA uses three local searches to improve the solution of the last population only, while our GA has two different operators to refine the individuals at each generation. Further details are provided in the following sections.

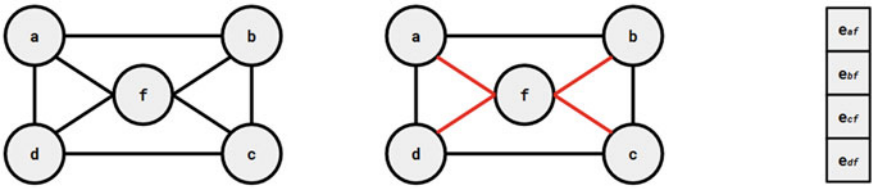


Fig. 1 Example of chromosome. Given the graph G on the left, we define a spanning tree of G, marked red in the center, and the related chromosome on the right

3.1 Chromosome Encoding and Fitness Function

In our implementation, a chromosome describes a feasible solution for the MCWST, a spanning tree of G. Each chromosome is represented as a set of genes, where each gene is an edge that forms the spanning tree, as shown in Fig. 1. Furthermore, the goodness of each individual is evaluated by the fitness function. Since we want to minimize the number of conflicts and the weight of the produced spanning tree, the fitness function is based on the total weight and the number of conflicting edge pairs of the spanning tree: Let T be the spanning tree described by the chromosome and M be a sufficient big number, the fitness function $f(x)$ is defined as follow:

$$f(x) = (\sum_{e \in T} w(e)) + |C(E_T)| \times M$$

The main objective is to reduce the number of conflicts. The fitness function guarantees this condition. In fact, if a solution has conflicts, the penalty introduced will be higher. In our computational experiments, we set the parameter M equals to the cost of the minimum spanning tree of G. On the contrary, if a solution is conflict-free, the fitness function will give more importance to reduce the weight of the spanning tree.

3.2 Initial Population

The initial population is composed of pop_{size} randomly generated individuals. In our experimental tests, pop_{size} is equal to 100. Each chromosome is produced as the following: starting from an empty set of edges T, a random edge e_i that connects two disconnected components is added to it, until we obtain a spanning tree. This procedure assures that the obtained tree is certainly an admissible spanning tree. In case that, the produced chromosome is already present in the population, the other is discarded.

3.3 Generation of New Individuals

A set of individuals is kept to the next generation through the selection strategy. The procedure aims to preserve individuals with the best fitness value. A ranked list system is used as a selection strategy to carry out the best chromosomes: the population is sorted in increasing order by fitness value and the first t elements are chosen for the next generation. In our implementation, t corresponds to $\frac{pop_{size}}{2}$.

3.4 Crossover Operator

The remaining $\frac{pop_{size}}{2}$ individuals of the population are generated by the crossover operator. Starting from two individuals defined as parents, an offspring is produced by recombining the genetic heritage of the two parents. The idea behind our crossover is to produce a spanning tree that does not necessarily reduce the conflicts edge pairs, of the parents, but the aim of exploring as much research space as possible. To this end, the chromosome c_1 is chosen from those preserved, while c_2 is selected from the previously discarded part of the population. The Crossover Operator uses four random indexes i_1, i_2 and k_1, k_2. Starting from i_1, k_1 genes are selected from c_1, and added into the offspring c_{child}. Starting from i_2, k_2 genes are taken from c_2. From the subset of edges, we take only the ones that reduce the number of disconnected components of c_{child}. If the c_{child} size is lower than $|V|-1$, the tree is completed by random edges that connect the component.

A variant of this crossover is called when $|C(c_1)| = 0$ instead. Given c_1 and c_2, the operator tries to define a third chromosome c_{child} which is conflict-free. The logic behind this variant is the same but the selected genes from c_2 still make c_{child} conflict-free. If the c_{child} size is lower than $|V| - 1$, the operator tries to complete the chromosome adding edges one by one which doesn't introduce conflicts into c_{child}, until the solution is feasible. If it's not possible, the tree is completed adding random edges that connect the component.

3.5 Mutation

Once the population is filled, the mutation operator is performed on all the individuals. We distinguish two cases: either the chromosome c has conflicts and or it is devoid of them. If $|C(c)| > 0$, the *ConflictReduction* operator is applied on c with a certain probability mut_{prob}. Otherwise the *LenghtReduction* operator is performed on c.

3.6 ConflictReduction Operator

The operator is designed to reduce the number of conflicts of a feasible solution T. For each $e \in T$, the number of conflicts $|C(e)|$ in which this edge is involved in T is computed. If $|C(e)| > 0$, e is removed from T, makes the tree disconnected. Then, an edge $e_k \in E - \{E_T\}$ that reconnects T is added to it s.t. $|C(e_k)| < |C(e)|$. If it's not possible, e is added again. At this point, since it has not been possible to remove e, the operator tries to reduce the conflicts by removing the edges which are in conflict with e. Therefore, we repeat the previous step $\forall e_i \in C(e)$ and when it's not possible to complete the tree, the selected edge is added again to grant a feasible solution.

3.7 LengthReduction Operator

Given a feasible conflict solution T, where $|C(E_T)| = 0$, the operator tries to minimize the weight of the tree while keeping the conflict-free condition. The procedure starts sorting the set of edges T by weight in nonincreasing order and $\forall e \in T$, e is removed from T. Every time an edge is removed, T defines two disconnected components. Hence, a second edge $e_i \in E \backslash T$ that connects them is added to T s.t. $|C(T + \{e_i\})| = 0 \wedge w(e_i) < w(e)$. If e_i is not found, we add again e instead.

3.8 Stopping Criteria

The genetic algorithm ends when a fixed number of iterations (gen_{size}) is reached or when there are no further improvements in the best fitness value after a certain number of generations gap_{gens}. The best individual of the current population is provided as the output of the algorithm. Figure 2 reports the flow chart of the GA.

4 Computational Experiments

Several computational experiments were performed to verify the effectiveness of the proposed approach. The benchmarks were provided by Samer and Urrutia [16] and include *type1* and *type2* instances. The first set is the harder one since several instances have neither feasibility nor optimality certified. The presence of a conflict-free solution is also not verified. Differently, all *type2* instances have at least one conflict-free solution by construction. The algorithm is written in Java with JDK ver.11 and all experimental tests have been performed on a PC with 2.30 GHz Intel Core i5-6200U processor and 8.0 GB memory running Ubuntu 18.04.2 LTS operating system. We compared the results of GA to the multiethnic

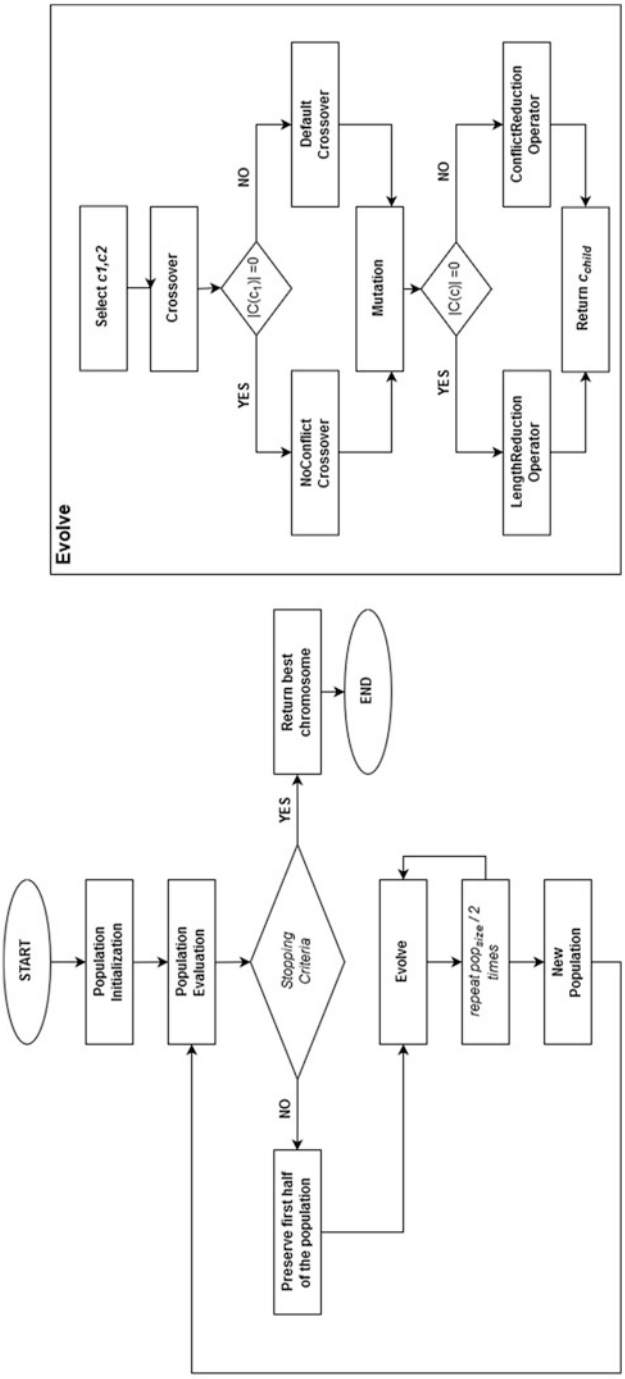


Fig. 2 Flow chart of GA

Table 1 Computational comparison between GA, MEGA and TS on type1 instances

	GA		MEGA		TS		GAP	
Instance	$\|C(E_T)\|$	w(T)	$\|C(E_T)\|$	w(T)	$\|C(E_T)\|$	w(T)	$\|C(E_T)\|$	w(T)
50_200_199	**0**	**708**	**0**	**708**	0	711	0.00%	0.00%
50_200_398	**0**	**770**	**0**	**770**	0	785	0.00%	0.00%
50_200_597	**0**	**917**	**0**	**917**	0	1086	0.00%	0.00%
50_200_995	**0**	**1324**	0	1336	0	1629	0.00%	0.90%
100_300_448	**0**	**4042**	0	4088	0	4207	0.00%	1.13%
100_300_897	**0**	**5687**	0	6095	–	–	0.00%	6.70%
100_300_1344	**8**	–	10	–	13	–	20.00%	–
100_500_1247	0	4282	**0**	**4275**	0	4593	0.00%	−0.16%
100_500_2495	**0**	**6053**	0	6199	0	6812	0.00%	2.35%
100_500_3741	0	8397	**0**	**7665**	0	8787	0.00%	−9.55%
100_500_6237	**7**	–	8	–	11	–	12.50%	–
100_500_12474	**34**	–	35	–	41	–	2.86%	–
200_600_1797	**0**	**16,162**	**0**	–	2	–	0.00%	–
200_600_3594	**55**	–	57	–	67	–	3.50%	–
200_600_5391	**136**	–	142	–	149	–	4.23%	–
200_800_6392	**18**	–	23	–	39	–	21.73%	–
200_800_9588	**83**	–	87	–	95	–	4.59%	–
200_800_15980	**171**	–	172	–	178	–	0.58%	–
300_800_3196	**47**	–	52	–	63	–	9.62%	–
300_1000_4995	**18**	–	21	–	38	–	14.28%	–
300_1000_9990	**166**	–	176	–	207	–	5.68%	–
300_1000_14985	**321**	–	329	–	351	–	2.43%	–

The last column reports the GAP between MEGA and GA. For 200_600_1797 instance, there is no reference for w(T) of MEGA by the authors [1]
The proposed values on the table are the best values obtained after five runs of the algorithms for each instance. In bold are reported the best values overall the approaches

genetic approach proposed by Carrabs et al. [1] and to the tabu search proposed by Zhang et al. [17]. It is important to point out that the computational experiments for the MEGA algorithm are performed on an OSX platform (Imac mid 2011), running on an Intel Core i7-2600 3.4 GHz processor with 8 GB of RAM. Computational results on *type1* instances are reported in Table 1 with relative execution times shown in Table 2. Regarding *type2* instances, GA obtains the same results of the other approaches, so they are not reported below. Table 1 is structured on five columns. The column *Instance* contains the name of the input files in the format: $|V|\_|E|\_|C|$. The remaining columns report the results obtained by the different approaches, genetic algorithm (GA), multiethnic genetic approach (MEGA) and tabu search (TS) respectively, in terms of the number of conflicts present in the obtained solution, reported as $|C(E_T)|$, and in terms of weight of the spanning tree, reported as $|w(T)|$. For no conflict-free result, only $|C(E_T)|$ is reported. Finally, the last column shows the gap. The gap is calculated between GA and MEGA since MEGA outperforms

Table 2 CPU times of GA, MEGA and TS on type1 instances

Instance	GA	MEGA	TS
50_200_199	3.10	0.71	1.17
50_200_398	3.37	0.68	1.13
50_200_597	2.78	0.63	0.98
50_200_995	3.10	0.66	1.16
100_300_448	15.91	2.39	6.33
100_300_897	13.34	1.81	5.99
100_300_1344	3.64	2.69	6.77
100_500_1247	13.86	5.18	17.71
100_500_2495	12.41	5.12	17.09
100_500_3741	21.46	3.72	14.94
100_500_6237	3.79	4.98	15.17
100_500_12474	8.74	6.68	14.64
200_600_1797	63.50	12.23	72.48
200_600_3594	19.12	21.71	70.24
200_600_5391	28.18	29.43	80.21
200_800_6392	20.60	28.2	98.01
200_800_9588	30.41	35.32	97.1
200_800_15980	40.31	44.48	104.93
300_800_3196	41.22	62.68	239.63
300_1000_4995	20.93	83.68	303.04
300_1000_9990	100.79	117.58	345.25
300_1000_14985	90.51	134.42	381.28

Times are reported in seconds

TS overall [1]. It is estimated with the formula $\frac{MEGA(|C(E_T)|)-GA(|C(E_T)|)}{MEGA(|C(E_T)|)} \times 100$ for $|C(E_T)|$, and $\frac{MEGA(w(T))-GA(w(T))}{MEGA(w(T))} \times 100$ for $w(T)$. Table 2 follows the same structure with the time in the last columns instead. Computational times are reported in seconds.

The GA parameters for all the tests are: 100 individuals as pop_{size}, 5000 as gen_{size}, $\frac{gen_{size}}{10}$ as gap_{gens} and 40% as mut_{prob}.

5 Considerations and Conclusions

The computational results show that our genetic algorithm outperforms all the other approaches present in literature, as far as providing a spanning tree with the least number of conflicting edge pairs. When it comes to define the minimum spanning tree, GA usually provides better solutions with respect to the other algorithm. The running time of our algorithm is comparable to MEGA, in particular, although GA is slightly slower on small instances, it becomes competitive on larger ones, compared to the other approach, despite the difference in terms of programming languages

and hardware. To understand further the differences in computational times, we compared the computing systems used for the experimentations, in particular, the CPUs. According to PassMark Software benchmark results [12], Intel Core i5-6200U processor, which was used for GA tests is 23% less powerful than Intel Core i7-2600 in single-thread computing. Moreover, in multi-thread computing the power gap increases to 51%. To improve the performances of our genetic algorithm, an idea for future work involves the partitioning of the input graph into subgraphs [10], since we have supposed if it is possible to execute a partition transforming the problem into the rainbow spanning forest problem [2, 3]. Considering our preliminary results, we think that this transformation will be the core of our future work concerning this problem.

References

1. Carrabs, F., Cerrone, C., Pentangelo, R.: A multiethnic genetic approach for the minimum conflict weighted spanning tree problem. Networks **74**(2), 134–147 (2017)
2. Carrabs, F., Cerrone, C., Cerulli, R., Silvestri, S.: On the complexity of rainbow spanning forest problem. Optim. Lett. **12**(3), 443–454 (2018)
3. Carrabs, F., Cerrone, C., Cerulli, R., Silvestri, S.: The rainbow spanning forest problem. Soft Comput. **22**(8), 2765–2776 (2018)
4. Carrabs, F., Cerulli, R., Pentangelo, R., Raiconi, A.: Minimum spanning tree with conflicting edge pairs: a branch-and-cut approach. Ann. Oper. Res., 1–14 (2018). https://link.springer.com/journal/10479/onlineFirst/page/23
5. Darmann, A., Pferschy, U., Schauer, J.: Determining a minimum spanning tree with disjunctive constraints. In: International Conference on Algorithmic DecisionTheory, pp. 414–423. Springer, Berlin (2009)
6. Darmann, A., Pferschy, U., Schauer, J., Woeginger, G.J.: Paths, trees and matchings under disjunctive constraints. Discrete Appl. Math. **159**(16), 1726–1735 (2011)
7. Holland, J.H.: Adaptation in Natural and Artificial Systems Ann Arbor, vol. 1, p. 975. The University of Michigan Press, Cambridge (1975)
8. Kanté, M.M., Laforest, C., Momege, B.: Trees in graphs with conflict edges or forbidden transitions. In: International Conference on Theory and Applications of Models of Computation, pp. 343–354. Springer, Berlin (2013)
9. Klein, A., Haugland, D., Bauer, J., Mommer, M.: An integer programming model for branching cable layouts in offshore wind farms. In: Modelling, Computation and Optimization in Information Systems and Management Sciences, pp. 27–36. Springer, Cham (2015)
10. Lum, O., Cerrone, C., Golden, B., Wasil, E.: Partitioning a street network into compact, balanced, and visually appealing routes. Networks **69**(3), 290–303 (2017)
11. Öncan, T., Zhang, R., Punnen, A.P.: The minimum cost perfect matching problem with conflict pair constraints. Comput. Oper. Res. **40**(4), 920–930 (2013)
12. PassMark Software: Benchmarks, CPU and performance, vol. 2. (2017). http://www.cpubenchmark.net/
13. Pferschy, U., Schauer, J.: The knapsack problem with conflict graphs. J. Graph Algorithms Appl. **13**(2), 233–249 (2009)
14. Pferschy, U., Schauer, J.: The maximum flow problem with disjunctive constraints. J. Combinat. Optim. **26**(1), 109–119 (2013)

15. Sadykov, R., Vanderbeck, F.: Bin packing with conflicts: a generic branch-and-price algorithm. INFORMS J. Comput. **25**(2), 244–255 (2013)
16. Samer, P., Urrutia, S.: A branch and cut algorithm for minimum spanning trees under conflict constraints. Optim. Lett. **9**(1), 41–55 (2015)
17. Zhang, R., Kabadi, S.N., Punnen, A.P.: The minimum spanning tree problem with conflict constraints and its variations. Discrete Optim. **8**(2), 191–205 (2011)

Algorithmic Strategies for a Fast Exploration of the TSP 4-OPT Neighborhood

Giuseppe Lancia and Marcello Dalpasso

Abstract The 4-OPT neighborhood for the TSP contains $\Theta(n^4)$ moves so that finding the best move effectively requires some ingenuity. Recently, de Berg et al. have given a $\Theta(n^3)$ dynamic program, but the cubic complexity is still too large for using 4-OPT in practice. We describe a new procedure which behaves, on average, slightly worse than a quadratic algorithm. This is much faster than the DP procedure, achieving speedups of two to three orders of magnitude on all instances we tested.

Keywords Traveling Salesman Problem · Local search · K-opt neighborhood

1 Introduction

In this work we consider the symmetric Traveling Salesman Problem (TSP), which consists in finding a Hamiltonian cycle (or *tour*) of minimum length in a complete graph $G = (V, E)$ of n nodes, weighted on the edges. Let us denote by $c(i, j) = c(j, i)$ the distance between any two nodes i and j. A tour is identified by a permutation of vertices $(v_1, \ldots, v_n)$. We call $\{v_i, v_{i+1}\}$, for $i = 1, \ldots, n-1$, and $\{v_n, v_1\}$ the *edges of the tour*. The length of a tour T, denoted by $c(T)$ is the sum of the lengths of the edges of the tour. More generally, for any set F of edges, we denote by $c(F)$ the value $\sum_{e \in F} c(e)$.

G. Lancia (✉)
DMIF, University of Udine, Udine, UD, Italy
e-mail: giuseppe.lancia@uniud.it

M. Dalpasso
DEI, University of Padova, Padova, PD, Italy
e-mail: marcello.dalpasso@unipd.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_40

A large number of applications over the years have shown that *local search* is often a very effective way to tackle hard combinatorial optimization problems [1, 8]. The local search paradigm for a general problem $\min\{f(x) : x \in X\}$ starts by defining a *neighborhood* function which associates to any solution x a set $N(x) \subset X$ of solutions reachable from x with a *move*. A popular neighborhood for the TSP is the K-OPT, where $K \in \mathbb{N}$ is a constant. Given a tour T, a tour T' belongs to $N(T)$ if T' and T differ by at most K edges. Hence, a K-OPT move on T consists in removing a set R of K edges and then inserting a set I of K edges so as $(T \setminus R) \cup I$ is still a tour. A K-OPT move is *improving* if $c(R) > c(I)$, i.e., if for the new tour it is $c((T \setminus R) \cup I) < c(T)$. A move is *best improving* (or, simply, a *best* move) if $c(R) - c(I)$ is maximum over all possible choices of R, I with $|R| = |I| = K$.

The local search starts at any feasible tour and then performs a series of improving moves, each time replacing T with a better $T' \in N(T)$ until T is the best tour in its neighborhood (i.e., it is a *local optimum*). If at each iteration we perform a best move, the strategy is called *best-improvement* or *steepest-descent* convergence. The alternative, called *first-improvement* is to perform the first improving move that we can find. In this paper we focus on the more challenging problem of finding the best move at each time. The changes to make our procedure work also for first improvement are minimal and left to the reader.

The first use of K-OPT dates back to 1958 with the introduction of 2-OPT in [2]. In 1965 Lin [6] described the 3-OPT neighborhood, and experimented with the complete enumeration algorithm, of complexity $\Theta(n^3)$, which finds the best 3-OPT move by trying all possibilities. The instances which could be tackled at the time were fairly small ($n \leq 150$). Later in 1968, Steiglitz and Weiner [9] described an improvement over Lin's method which made it two or three times faster (although still cubic).

The exploration of the K-OPT neighborhood, for a fixed K, might be considered "fast" from a theoretical point of view, since the most obvious algorithm (complete enumeration) has complexity $\Theta(n^K)$ and is therefore polynomial. However, despite being polynomial, this algorithm cannot be used in practice already for $K = 3$ (if n is large enough, like 3000 or more). In our previous work [5] we have described some algorithmic ideas to speed-up the exploration of the 3-OPT neighborhood in order to lower its complexity and make it practical. The result is a procedure which takes on average a subcubic time (estimated experimentally around $O(n^{2.5})$) to find the best 3-OPT move. This paper is a follow-up to [5] in which we try to apply similar ideas to the 4-OPT neighborhood in order to make it practical on graphs on which it could have never been applied before.

An important result in [3] proves that, under a widely believed hypothesis similar to the $P \neq NP$ conjecture, it is impossible to find the best 3-OPT move with a worst-case algorithm of time $O(n^{3-\epsilon})$ for any $\epsilon > 0$ so that complete enumeration is, in a sense, optimal for 3-OPT. The work [3], however, does not rule out the

possibility of algorithms with an average-case complexity lower than $O(n^3)$ and indeed our procedure [5] takes less than $O(n^3)$ time on average on several classes of random instances. Furthermore, in [3] it is proved that it is possible to find the best 4-OPT move in *worst-case* time $\Theta(n^3)$ by a dynamic programming procedure. This is already a huge improvement over the $\Theta(n^4)$ complete enumeration procedure, but still the cubic complexity is not practical for large values of n. The goal of our paper is to describe a procedure which behaves, on average, better than the cubic dynamic programming procedure. The computational results section will show how this goal has indeed been achieved.

Assessing the effectiveness of 3-OPT and 4-OPT, and possible adjustments to make them even better (such as warm restarts and perturbations) with respect to the quality of the local optima they find is beyond the scope of this paper, and will be the matter of future research in which these neighborhoods will be compared with some more involved heuristics (such as, e.g., Lin and Kernighan's procedure [7]). For a very good chapter comparing various heuristics for the TSP, see Johnson and Mc Geich [4].

1.1 The Main Contribution

The contribution of this paper is the description of a procedure to find the best 4-OPT move which outperforms the $\Theta(n^3)$ dynamic programming (and, obviously, is incredibly faster than the $\Theta(n^4)$ enumeration) on all instances that we tried (both random and from the TSPLIB repository).

Let us give a flavor of the type of results that we can achieve (a discussion of the computational results can be found in Sect. 5). On an average PC, finding the best 4-OPT move for a given tour of 2000 nodes by listing all possible moves takes more than 1 day. The dynamic programming procedure finds the best move in about 400 s. Our procedure finds it in 4 s. With larger n the improvements are even more dramatic, e.g., it is about 500 times faster than DP when $n = 10{,}000$.

2 The 4-OPT Neighborhood

Let $G = (V, E)$ be a complete graph on n nodes, and $c : E \mapsto \mathbb{R}^+$ be a cost function for the edges. We assume $V = \{0, 1, \ldots, \bar{n}\}$, where $\bar{n} = n - 1$. In this paper we will describe an effective strategy for finding the best move for a given current tour T which, without loss of generality, will always be $T = (0, 1, \ldots, \bar{n})$.

Since we will be using modular arithmetic frequently, for each $x \in V$ and $t \in \mathbb{N}$ we define

$$x \oplus t := (x + t) \mod n, \qquad x \ominus t := (x - t) \mod n.$$

A 4-OPT move is fully specified by two sets, i.e., the set of removed and of inserted edges. We call a *removal set* any set of four tour edges, i.e., four edges of type $\{k, k \oplus 1\}$. A removal set is identified by a quadruple $S = (i_1, i_2, i_3, i_4)$ with $0 \leq i_1 < i_2 < i_3 < i_4 \leq \bar{n}$, where the edges removed are $R(S) := \{\{i_j, i_j \oplus 1\} : j = 1, \ldots, 4\}$. We call any such quadruple S a *selection*. A selection is *complete* if $i_s \neq i_t \oplus 1$ for each s, t, otherwise we say that S is a partial selection. We denote the set of all complete selections by $\mathcal{S}$. Furthermore, for each $1 \leq a < b \leq 4$, we denote by

$$\mathcal{S}_{ab} = \{(x, y) : \exists (i_1, i_2, i_3, i_4) \in \mathcal{S} \text{ with } i_a = x, i_b = y\}$$

$\mathcal{S}_{24}$, for instance, contains all pairs of values (x, y) such that we can replace the "*" with numbers in $(*, x, *, y)$ so as to make it a valid selection. We call any such pair a *pivot*. Pivots will be important in our enumeration procedure described later. Clearly $|\mathcal{S}_{ab}| = \Theta(n^2)$ for all a, b.

Let S be a selection and $I \subset E$ with $|I| = 4$. If $(T \setminus R(S)) \cup I$ is still a tour then I is called a *reinsertion set*. Given a selection S, a reinsertion set I is *pure* if $I \cap R(S) = \emptyset$, and *degenerate* otherwise. The 4-OPT moves when the reinsertions are degenerate and/or the selections are partial are, in fact, either 2-OPT or 3-OPT moves. Therefore, the most computationally expensive task is to determine the best move when *the selection is complete and the reinsertion set is pure*. In the remainder of the paper, by 4-OPT moves we will implicitly mean only moves for which the selection is complete and the reinsertion set is pure. Similarly, by "selection" we will in fact mean a complete selection.

2.1 Reinsertion Schemes, Orbits and Moves

Let $S = (i_1, \ldots, i_4)$ be a selection. When the edges $R(S)$ are removed from a tour, the tour gets broken into four consecutive segments which we can label by $\{1, \ldots, 4\}$. The segment labeled l has i_l as its last vertex. A reinsertion set reconnects back these segments into a new tour. If we adopt the convention that a tour starts always with segment 1 traversed clockwise, the reinsertion set: (1) determines a new ordering in which the segments are visited along the tour and (2) may cause some segments to be traversed counterclockwise. In order to represent this fact we can use a notation called a *reinsertion scheme*. A reinsertion scheme is a signed

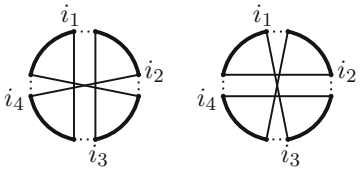


Fig. 1 The reinsertion schemes $< +4, -2, -3 >$ (left) and $< -3, -4, +2 >$ (right)

permutation of $\{2, 3, 4\}$. The permutation specifies the order in which the segments 2, 3, and 4 are visited after the move. The signing $-s$ tells that segment s is traversed counterclockwise, while $+s$ tells that it is traversed clockwise. For example, the reinsertion set depicted in Fig. 1 (left) is also represented by the reinsertion scheme $< +4, -2, -3 >$.

There are potentially $2^3(3)! = 48$ reinsertion schemes, but for many of these the corresponding reinsertion sets are degenerate. A scheme for a pure reinsertion must not start with $+2$, nor end with "$+4$", nor contain consecutive elements "$+t, +(t + 1)$" or "$-t, -(t - 1)$" for any t in $1, \ldots, 4$. It turns out that the pure reinsertion schemes are exactly 25:

Proposition 1 *There are 25 pure reinsertion schemes for* 4*-OPT.*

Proof We prove the claim by listing the schemes $r_1, \ldots, r_{25}$, since we will be needing them when we discuss how to find the best 4-OPT move. The schemes are enumerated by first looking at the permutation of the segments and then the signings:

$$
\begin{array}{llll}
(2,3,4): & r_1 =< -2,-3,-4 > & r_2 =< -2,+3,-4 > & \\
(2,4,3): & r_3 =< -2,-4,+3 > & r_4 =< -2,+4,-3 > & r_5 =< -2,+4,+3 > \\
(3,2,4): & r_6 =< -3,+2,-4 > & r_7 =< +3,-2,-4 > & r_8 =< +3,+2,-4 > \\
(3,4,2): & r_9 =< -3,-4,-2 > & r_{10} =< -3,-4,+2 > & r_{11} =< -3,+4,-2 > \\
 & r_{12} =< -3,+4,+2 > & r_{13} =< +3,-4,-2 > & r_{14} =< +3,-4,+2 > \\
(4,2,3): & r_{15} =< -4,-2,-3 > & r_{16} =< +4,-2,-3 > & r_{17} =< -4,-2,+3 > \\
 & r_{18} =< +4,-2,+3 > & r_{19} =< -4,+2,-3 > & r_{20} =< +4,+2,-3 > \\
(4,3,2): & r_{21} =< -4,+3,-2 > & r_{22} =< -4,+3,+2 > & r_{23} =< +4,-3,+2 > \\
 & r_{24} =< +4,+3,-2 > & r_{25} =< +4,+3,+2 > &
\end{array}
$$

□

Although the reinsertion schemes are 25, we can see how some of them are equivalent in the sense that we can obtain the (unlabeled) drawing of one from another by either a rotation or a flip around an axis of symmetry in the plane. For instance in Fig. 1 we see the schemes r_{16} (left) and r_{10} (right) and it is clear how r_{10} can be obtained from r_{16} by a rotation of 90°. If we consider the classes of this equivalence, the 25 reinsertion schemes get partitioned into 7 classes (called

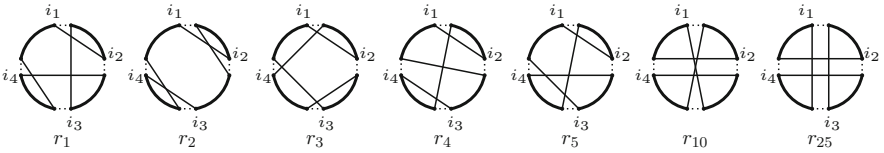


Fig. 2 Orbits of 4-OPT

orbits) denoted by $\mathcal{O}_1, \ldots, \mathcal{O}_7$. For each orbit $\mathcal{O}_k$ it is then enough to consider one particular scheme r (the orbit *representative*). We will then say that $\mathcal{O}_k = \mathcal{O}(r)$. The grouping into orbits simplifies our analysis, since it is enough to specify how to deal with the orbit representative instead of considering all the reinsertions schemes of the orbit.

Proposition 2 *The pure reinsertion schemes are partitioned into the following seven orbits* $\mathcal{O}_1, \ldots, \mathcal{O}_7$*:*

$$\mathcal{O}_1 = \mathcal{O}(r_1) = \{r_1, r_{24}, r_{23}, r_{22}\}, \qquad \mathcal{O}_2 = \mathcal{O}(r_2) = \{r_2, r_{21}\}$$
$$\mathcal{O}_3 = \mathcal{O}(r_3) = \{r_3, r_7, r_{13}, r_{17}\}, \qquad \mathcal{O}_4 = \mathcal{O}(r_4) = \{r_4, r_{19}, r_{11}, r_6\}$$
$$\mathcal{O}_5 = \mathcal{O}(r_5) = \{r_5, r_{20}, r_{14}, r_{15}, r_{12}, r_{18}, r_9, r_8\}, \qquad \mathcal{O}_6 = \mathcal{O}(r_{10}) = \{r_{10}, r_{16}\}$$
$$\mathcal{O}_7 = \mathcal{O}(r_{25}) = \{r_{25}\}$$

In Fig. 2 we illustrate the representatives of the seven orbits of 4-OPT.

3 Previous Approaches and Our Strategy

Each 4-OPT move is identified by a pair $\mu = (r, S)$ where r is a reinsertion scheme and S is a selection. Let $\mathcal{M}$ be the set of all 4-OPT moves. We have the following theorem (proof omitted for space reasons).

Theorem 1 *The number of* 4*-OPT moves for a tour of* n *nodes is*

$$|\mathcal{M}| = 25 \times \binom{n-3}{4} - \binom{n-5}{2} = 25 \times \frac{n^4 - 18n^3 + 107n^2 - 210n}{24} \tag{1}$$

To get an idea of how fast this number grows, we remark that it is roughly 86 million for $n = 100$, 1 billion for $n = 1000$ and 10 million billions for $n = 10{,}000$.

To find the best move overall, it is sufficient to describe how to find the best selection for a given reinsertion scheme r, and then iterate the same process over all the reinsertion schemes. Moreover, we can limit ourselves to consider the seven schemes r that are representatives of the seven orbits. Let us assume then that r is

fixed and we want to find the best move (i.e., best selection for r. From now on, "move" and "selection" mean, basically, the same thing).

The obvious polynomial algorithm (brute force) is a 4-level, nested-for loop which iterates over all selections i_1, i_2, i_3, i_4 and takes $\Theta(n^4)$ time (both on average and in the worst case).

3.1 A $\Theta(n^3)$ Dynamic Programming Procedure

A big improvement over the brute force algorithm is the dynamic programming procedure introduced in [3]. For a given reinsertion scheme r, and selection (i_1, i_2, i_3, i_4), let $F = \{f_1, \ldots, f_4\}$ be the set of edges inserted by r. We say that two nodes i_s and i_t are *independent* if no edge of F is incident on both $\{i_s, i_s \oplus 1\}$ and $\{i_t, i_t \oplus 1\}$. For instance, in Fig. 1 (left), the nodes i_1 and i_2 are independent, while i_1 and i_3 are not, etc. It can be shown that the selection nodes can be partitioned into two groups say $A = \{a_1, a_2\}$ and $B = \{b_1, b_2\}$ of independent nodes, so that the cost of the move is

$$c(a_1, a_1 \oplus 1) + c(a_2, a_2 \oplus 1) + \tilde{c}_1(b_1|a_1, a_2) + \tilde{c}_2(b_2|a_1, a_2)$$

with $\tilde{c}_1(b_1|a_1, a_2) := c(b_1, b_1 \oplus 1) - c(f') - c(f'')$ and $\tilde{c}_2(b_2|a_1, a_2) := c(b_2, b_2 \oplus 1) - c(f''') - c(f'''')$ for a suitable partitioning of F into $\{f', f''\}$ and $\{f''', f''''\}$. To find the best move, we consider all possibilities for the choice of a_1 and a_2 (there are $\Theta(n^2)$ choices). For each such choice, we find the completion of the selection by optimally placing the remaining two nodes b_1 and b_2. In [3] it is shown how this can be done via a dynamic programming procedure in time $\Theta(n)$ rather than $\Theta(n^2)$, by exploiting the fact that the costs $\tilde{c}$ for b_1 and b_2 do not depend on each other but only on a_1 and a_2.

3.2 Our "Smart Force" Approach

Our idea for an effective search is based on the following considerations, similar to those utilized successfully for 3-OPT in [5] and which were called *smart force* in opposition to the *brute force* enumerative approach.

Suppose there is a magic box which contains all the best moves, but these moves have been masked by deleting 2 indices out of 4. So each entry in the box (a *masked move*) is something like $(*, 6, 9, *)$, which says that there is a best move for which $i_2 = 6, i_3 = 9$ but we don't know i_1 and i_4. We can inquire the box by specifying two indices $a, b \in \{1, 2, 3, 4\}$ and the box, in time $O(1)$ returns us a masked move (x_1, x_2, x_3, x_4) such that $x_i \neq *$ iff $i = a, b$. How can we use such a box to find the best move?

One way would be to ask, e.g., for a masked move revealing the first two nodes of a best move. Say we get back $(5, 17, *, *)$. At this point we could enumerate the values for the two missing nodes i_3 and i_4 in the range $[19, \ldots, \bar{n}]$ and determine the best completion possible. This way, finding a best improving 4-OPT move would take $\Theta(n^2)$. Indeed, we could do even better than $\Theta(n^2)$ by calling the magic box more than once. Let's say that we make a call for the first two indices and obtain the masked move $(v_1, v_2, *, *)$. Now we would keep calling the box asking for all the masked moves of type $(*, , *, w_3, w_4)$ in the box. We would then determine the best solution among all the quadruples which do in fact represent feasible selections (i.e., $(v_1, v_2, w_3, w_4) \in \mathcal{S}$). If the number of best moves is considerably smaller than n^2, this procedure would take less than $O(n^2)$ time. As a matter of fact, the number of best moves is in general *much* smaller than n^2 (many times the best move is unique, in which case we would determine it in time $O(1)$ with two calls). Let us say that there are B best moves overall. Then the above approach would take time $O(B)$. It is safe to say that B is in general a very small number (it can almost be considered a constant), as we noticed in our computational experiments.

The bulk of our work has then been to simulate a similar magic box, i.e., a data structure that can be queried and should return two out of four nodes of a best move much in a similar way as described above. Being heuristic, our box, rather than returning a pair of nodes that are certainly in a best move, returns a pair of nodes that *are likely to be in a best move*. In order to assess the likelihood of two specific indices to be in a best solution, we will use suitable two-arguments functions. Loosely speaking, these functions will be used to determine, for each pair of indices of a selection, the contribution of that pair to the value of a move. The rationale is that, the higher the contribution, the higher the probability that a particular pair is in a best selection.

The two functions are called $\tau^+()$ and $\tau^-()$. For each $a, b \in \{0, \ldots, \bar{n}\}$, we define $\tau^+(a, b)$ to be the difference between the cost from a to its successor and the cost from a to the successor of b, i.e.,

$$\tau^+(a, b) = c(a, a \oplus 1) - c(a, b \oplus 1)$$

while $\tau^-(a, b)$ is be the difference between the cost from a to its predecessor and the cost from a to the predecessor of b, i.e.,

$$\tau^-(a, b) = c(a, a \ominus 1) - c(a, b \ominus 1)$$

Clearly, each of these quantities can be computed in time $O(1)$, and computing their values for a subset of possible pairs can never exceed time $O(n^2)$.

4 Finding the Best 4OPT Move

Let then r be a given reinsertion scheme (representative of an orbit). For each selection $S = (i_1, i_2, i_3, i_4)$, the evaluation of a move requires to compute the difference $\Delta_r(i_1, i_2, i_3, i_4)$ of costs between the removed edges and the inserted edges. We start by showing how to break up the function $\Delta_r()$, that has $\Theta(n^4)$ possible arguments, into a sum of functions of two parameters each. In particular, we have

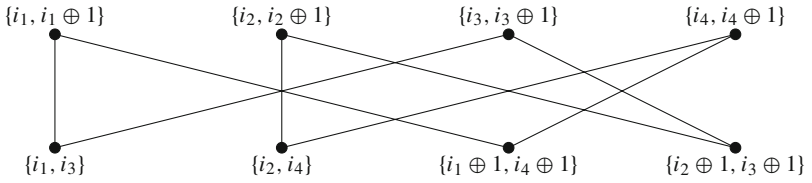
$$\Delta_r(i_1, i_2, i_3, i_4) = f_r^1(i_{\pi(1)}, i_{\pi(2)}) + f_r^2(i_{\pi(3)}, i_{\pi(4)}) + f_r^3(i_{\sigma(1)}, i_{\sigma(2)}) + f_r^4(i_{\sigma(3)}, i_{\sigma(4)}) \tag{2}$$

for suitable functions $f_r^1()$, $f_r^2()$, $f_r^3()$, $f_r^4()$, each representing the contribution of two specific indices to the value of the move, and permutations π, σ of the set $\{1, 2, 3, 4\}$.

Theorem 2 *For every* 4*-OPT reinsertion scheme there exist functions* $f^1()$, $f^2()$, $f^3()$, $f^4() : \mathbb{Z} \times \mathbb{Z} \mapsto \mathbb{R}$*, and permutations* π *and* σ *of the set* $\{1, 2, 3, 4\}$ *such that for each selection* (i_1, i_2, i_3, i_4) *it is*

$$\Delta_r(i_1, i_2, i_3, i_4) = f^1(i_{\pi(1)}, i_{\pi(2)}) + f^2(i_{\pi(3)}, i_{\pi(4)}) + f^3(i_{\sigma(1)}, i_{\sigma(2)}) + f^4(i_{\sigma(3)}, i_{\sigma(4)})$$

Proof (Sketch) Let us consider a bipartite graph B with four vertices on top and four on bottom. The top vertices are the tour edges $R = \{e_1, e_2, e_3, e_4\}$ removed by the selection. The bottom vertices are the edges $I = \{e'_1, e'_2, e'_3, e'_4\}$ inserted by the scheme. In B there is an edge ee' between every $e \in R$ and $e' \in I$ such that $e \cap e' \neq \emptyset$. It is immediate to see that B is in fact a length-8 cycle. As an example, in the figure below we depict B for the reinsertion scheme $r_{11} =< -3, +4, -2 >$.



The cycle is the disjoint union of two perfect matchings. From either one of them we can obtain the functions $f^1()$, $f^2()$, $f^3()$, $f^4()$. For example, consider the matching

$$\{i_1, i_1 \oplus 1\} \leftrightarrow \{i_1, i_3\}, \quad \{i_2, i_2 \oplus 1\} \leftrightarrow \{i_2, i_4\}, \quad \{i_3, i_3 \oplus 1\} \leftrightarrow \{i_2 \oplus 1, i_3 \oplus 1\},$$
$$\{i_4, i_4 \oplus 1\} \leftrightarrow \{i_1 \oplus 1, i_4 \oplus 1\}$$

From each edge of the matching we derive either a τ^+ or a τ^- expression as follows. Assume the edge is $e_x e'$, with $x \in \{1, 2, 3, 4\}$. If $e' = \{i_x, i_y\}$ then

$$c(e_x) - c(e') = c(i_x, i_x \oplus 1) - c(i_x, i_y) = c(i_x, i_x \oplus 1) - c(i_x, (i_y \ominus 1) \oplus 1) = \tau^+(i_x, i_y \ominus 1)$$

Otherwise, it is $e' = \{i_x \oplus 1, i_y\}$ and

$$c(e_x) - c(e') = c(i_x, i_x \oplus 1) - c(i_x \oplus 1, i_y) = c(i_x \oplus 1, i_x) - c(i_x \oplus 1, (i_y \oplus 1) \ominus 1) = \tau^-(i_x \oplus 1, i_y \oplus 1)$$

The sum of these values, for $x = 1, 2, 3, 4$, is then the cost of the move. In the move of the above example it is

$$\Delta_r(i_1, i_2, i_3, i_4) = \tau^+(i_1, i_3 \ominus 1) + \tau^+(i_2, i_4 \ominus 1) + \tau^-(i_3 \oplus 1, i_2 \oplus 2) + \tau^-(i_4 \oplus 1, i_1 \oplus 2).$$

By reading the order in which the various types of indices appear in the first two addends of the above sum we get the permutation π of the theorem statement. Similarly, we get σ from the last two addends. □

In the above example, the permutations are $\pi = (1, 3, 2, 4)$ and $\sigma = (3, 2, 4, 1)$. Furthermore,

- $f^1(x, y) := \tau^+(x, y \ominus 1)$, with domain $\mathcal{S}_{13}$ $f^2(x, y) := \tau^+(x, y \ominus 1)$, with domain $\mathcal{S}_{24}$
- $f^3(x, y) := \tau^-(x \oplus 1, y \oplus 2)$, with domain $\mathcal{S}_{32}$ $f^4(x, y) := \tau^-(x \oplus 1, y \oplus 2)$, with domain $\mathcal{S}_{41}$

Notice how the theorem implies that each function $f^i(x, y)$ is either a τ^+ or a τ^-. By using Theorem 2, we can give the functions $f^1, \ldots, f^4$ and the permutations π and σ for the representatives of all orbits, as reported in Table 1.

Table 1 Expressing $\Delta_r(i, j, k, h)$ as a sum $f^1() + f^2() + f^3() + f^4()$

r	π	σ	$f^1()$	$f^2()$	$f^3()$	$f^4()$
r_1	(1, 2, 4, 3)	(3, 1, 2, 4)	$\tau^+(i, j \ominus 1)$	$\tau^-(h \oplus 1, k \oplus 2)$	$\tau^+(k, i)$	$\tau^-(j \oplus 1, h \oplus 1)$
r_2	(1, 2, 3, 4)	(2, 1, 4, 3)	$\tau^+(i, j \ominus 1)$	$\tau^+(k, h \ominus 1)$	$\tau^-(j \oplus 1, i \oplus 2)$	$\tau^-(h \oplus 1, k \oplus 2)$
r_3	(1, 2, 3, 4)	(4, 1, 2, 3)	$\tau^+(i, j \ominus 1)$	$\tau^+(k, h)$	$\tau^+(h, i)$	$\tau^-(j \oplus 1, k \oplus 2)$
r_4	(1, 2, 4, 3)	(3, 1, 2, 4)	$\tau^+(i, j \ominus 1)$	$\tau^+(h, k \ominus 1)$	$\tau^-(k \oplus 1, i \oplus 2)$	$\tau^-(j \oplus 1, h \oplus 2)$
r_5	(1, 2, 4, 3)	(3, 1, 2, 4)	$\tau^+(i, j \ominus 1)$	$\tau^-(h \oplus 1, k \oplus 1)$	$\tau^-(k \oplus 1, i \oplus 2)$	$\tau^-(j \oplus 1, h \oplus 1)$
r_{10}	(1, 3, 2, 4)	(3, 1, 4, 2)	$\tau^+(i, k \ominus 1)$	$\tau^+(j, h)$	$\tau^-(k \oplus 1, i \oplus 2)$	$\tau^+(h, j)$
r_{25}	(1, 3, 2, 4)	(3, 1, 4, 2)	$\tau^+(i, k)$	$\tau^+(j, h)$	$\tau^+(k, i)$	$\tau^+(h, j)$

4.1 How to Find the Best Selection

Suppose we want to find a selection $S = (i_1, i_2, i_3, i_4)$ better than a selection $S^* = (\bar{v}_1, \bar{v}_2, \bar{v}_3, \bar{v}_4)$ of value V (the current "champion"). Then S must satisfy

$$\left(f^1(i_{\pi(1)}, i_{\pi(2)}) + f^2(i_{\pi(3)}, i_{\pi(4)}) > \frac{V}{2}\right) \vee \left(f^3(i_{\sigma(1)}, i_{\sigma(2)}) + f^4(i_{\sigma(3)}, i_{\sigma(4)}) > \frac{V}{2}\right) \tag{3}$$

For the sake of example, assume $\pi = (1, 3, 2, 4)$ and $\sigma = (1, 4, 2, 3)$. We then run a two-phase algorithm. In the first phase we look for all selections (i, j, k, h) such that $f^1(i, k) + f^2(j, h) > \frac{V}{2}$ and then check if indeed $\Delta_r(i, j, k, h) > V$. In the second phase we look for all selections such that $f^3(i, h) + f^4(j, k) > \frac{V}{2}$ and then check if indeed $\Delta_r(i, j, k, h) > V$. Whenever we improve the champion, we immediately update V so that the two conditions become harder to satisfy.

The procedure for each phase has in input two max-heaps H' and H'' and a permutation ϕ of $\{1, 2, 3, 4\}$. The heap H' contains pivots in the domain $\mathcal{S}_{\phi(1)\phi(2)}$ while H'' contains pivots in the domain $\mathcal{S}_{\phi(3)\phi(2)}$. Each heap corresponds to one of the addends of (2) and is sorted according to the value $f_H(x, y)$ of its pivots (x, y) (where $f_H()$ is one of $f^1, \ldots, f^4$, depending on the heap). The goal of the phase is to form all quadruples with value $> V/2$ by picking one element from each heap. We then run a loop to identify all pairs (x, y), (z, w) of pivots, taken from H' and H'' respectively, such that $f_{H'}(x, y) + f_{H''}(z, w) > V/2$. The loop terminates as soon as the sum of the maxima of the heaps is $\leq V/2$. To perform this search effectively, given that the maximum of H' is (x_1, y_1) of value $f_{H'}(x_1, y_1)$, we first extract from H'' all elements (z_c, w_c) such that $f_{H'}(x_1, y_1) + f_{H''}(z_c, w_c) > V/2$. Note that this way we have in fact created a sorted array of those elements from H'', i.e., H'' now can be replaced by an array $A'' = [(z_1, w_1), \ldots, (z_Q, w_Q)]$ such that

$$f_{H''}(z_1, w_1) \geq \cdots \geq f_{H''}(z_Q, w_Q) > \frac{V}{2} - f_{H'}(x_1, y_1)$$

Creating this array has cost $O(Q \log n)$. In a similar way, in time $O(P \log n)$ we create a sorted array $A' = [(x_1, y_1), \ldots, (x_P, y_P)]$ containing all the elements of H' such that

$$f_{H'}(x_1, y_1) \geq \cdots \geq f_{H'}(x_P, y_P) > \frac{V}{2} - f_{H''}(z_1, w_1)$$

Now we combine elements from the first array and the second array to form all quadruples of value $> V/2$. For doing so we keep two indexes a and b, one per array. Initially $a = b = 1$. If $f_{H'}(x_a, y_a) \geq f_{H''}(z_b, w_b)$ we say that a is the master and b

is the slave, otherwise b is the master and a the slave. We run a double loop which ends as soon as $f_{H'}(x_a, y_a) + f_{H''}(z_b, w_b) \leq V/2$. At each iteration, one index runs through all the elements from the slave down, as long as the sum of their values and the master's value is still $> V/2$. For example, if the master is a, then we would consider all elements $c = b, b+1, b+2, \ldots$ such that $f_{H'}(x_a, y_a) + f_{H''}(z_c, w_c) > V/2$. For each quadruple (x_a, y_a, z_c, w_c) thus obtained we would, in time $O(1)$ sort the indices so as to obtain values i, j, k, h with $i \leq j \leq k \leq h$ and check if indeed (i, j, k, h) is a valid selection and $\Delta(i, j, k, h) > V$. In that case, we would update the current champion and its value V (note that this might in turn cause an earlier termination of the loop).

Notice that once a quadruple is formed there is nothing more to do than compute its value, in time $O(1)$. If the total number of quadruples evaluated is L, the complexity of the loop is $O(L)$ so that, overall, the procedure takes time $O((P + Q) \log n + L)$ where $P = \texttt{size}(A')$ and $Q = \texttt{size}(A'')$. In our computational experiments, by the least square fitting of the running times we noticed that this procedure behaves, in practice, like $O(n^\alpha \log n)$ with $\alpha \simeq 2.1$.

5 Computational Results and Conclusions

For this extended abstract we have run a preliminary set of tests, but large enough to show the effectiveness of the method. In particular, we have first considered random instances of n nodes, in which the edge costs are drawn UAR in $[0, 1]$, for $n = 1000, 2000, \ldots, 10{,}000$. For each n we generated ten instances. For each of them, starting from a random tour we found the best move by DP and by smart force. We also computed (an estimate of) the time needed by the $\Theta(n^4)$ enumeration algorithm. The results (averaged over the ten instances for each n and rounded up) are reported in Table 2. We can see that the speedups of smart force w.r.t. dynamic programming range from about 30 times to about 500 times faster.

We have then selected from the TSPLIB repository all geometric instances of size $1500 \leq n \leq 6000$ and performed the same experiment, this time starting from ten random tours per instance. The results are reported in the bottom half of Table 2. It can be seen that the speedups are comparable to before, with improvements from about 100 times faster to more than 300 times faster.

There is a lot of work left such as running extensive tests on different families of random costs, as well as trying to prove in a formal way that the expected running time is lower than cubic, at least for graphs with costs drawn UAR. All this work will be the matter of future research.

Table 2 Average running times for random graphs and TSPLIB instances

Name	Size n	Dynamic program	Smart force	Speedup	Brute force
Random	1000	26 s	1 s	26×	2 h 46 min 40 s
Random	2000	418 s (6 min 58 s)	4 s	104×	1 d 20 h 26 min 40 s
Random	3000	1771 s (29 min 31 s)	12 s	147×	9 d 9 h 0 s
Random	4000	4793 s (1 h 19 min 53 s)	24 s	199×	29 d 15 h 6 min 40 s
Random	5000	10,258 s (2 h 50 min 58 s)	41 s	250×	72 d 8 h 6 min 40 s
Random	6000	19,095 s (5 h 18 min 15 s)	67 s (1 min 7 s)	285×	150 d 0 s
Random	7000	39,981 s (11 h 6 min 21 s)	93 s (1 min 33 s)	429×	277 d 21 h 26 min 40 s
Random	8000	54,316 s (15 h 5 min 16 s)	174 s (2 min 54 s)	312×	1 y 109 d 1 h 46 min 40 s
Random	9000	75,151 s (20 h 52 min 31 s)	163 s (2 min 43 s)	461×	2 y 29 d 9 h 0 s
Random	10,000	106,046 s (1 d 5 h 27 min 26 s)	216 s (3 min 36 s)	490×	3 y 62 d 9 h 46 min 40 s
TSPLIB fl1577	1577	218 s (3 min 38 s)	2 s	109×	17 h 10 min 48 s
TSPLIB d1655	1655	265 s (4 min 25 s)	3 s	88×	20 h 50 min 22 s
TSPLIB vm1748	1748	321 s (5 min 21 s)	4 s	80×	1 d 1 h 56 min 1 s
TSPLIB u1817	1817	368 s (6 min 8 s)	4 s	92×	1 d 6 h 16 min 38 s
TSPLIB rl1889	1889	438 s (7 min 18 s)	4 s	109×	1 d 11 h 22 min 9 s
TSPLIB d2103	2103	610 s (10 min 10 s)	6 s	101×	2 d 6 h 19 min 54 s
TSPLIB u2152	2152	697 s (11 min 37 s)	6 s	116×	2 d 11 h 34 min 31 s
TSPLIB u2319	2319	886 s (14 min 46 s)	7 s	126×	3 d 8 h 20 min 3 s
TSPLIB pr2392	2392	976 s (16 min 16 s)	8 s	122×	3 d 18 h 56 min 14 s
TSPLIB pcb3038	3038	2246 s (37 min 26 s)	13 s	172×	9 d 20 h 37 min 6 s
TSPLIB fl3795	3795	4879 s (1 h 21 min 19 s)	21 s	232×	24 d 9 min 43 s
TSPLIB fnl4461	4461	8616 s (2 h 23 min 36 s)	32 s	269×	45 d 20 h 5 min 7 s
TSPLIB rl5915	5915	22,606 s (6 h 16 min 46 s)	65 s (1 min 5 s)	347×	141 d 16 h 17 min 39 s
TSPLIB rl5934	5934	23,041 s (6 h 24 min 1 s)	66 s (1 min 6 s)	349×	143 d 12 h 11 min 40 s

References

1. Aarts, E., Lenstra, J.K. (ed.): Local Search in Combinatorial Optimization, 1st edn. Wiley, New York (1997)
2. Croes, G.A.: A method for solving traveling-salesman problems. Oper. Res. **6**(6), 791–812 (1958)
3. de Berg, M., Buchin, K., Jansen, B.M.P., Woeginger, G.J.: Fine-grained complexity analysis of two classic TSP variants. In: 43rd ICALP, pp. 1–14 (2016)
4. Johnson, D., McGeoch, L.A.: The traveling salesman problem: a case study in local optimization. In: Local Search in Combinatorial Optimization, vol. 1, pp. 215–310. Wiley, Chichester (1997)
5. Lancia, G., Dalpasso, M.: Speeding-up the exploration of the 3-OPT neighborhood for the TSP. In: Daniele, P., Scrimali, R. (eds.) New Trends in Emerging Complex Real Life Problems. AIRO Springer Series, vol. 1, pp. 345–356. Springer, Cham (2018)
6. Lin, S.: Computer solutions of the traveling salesman problem. Bell Syst. Tech. J. **44**(10), 2245–2269 (1965)
7. Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the traveling-salesman problem. Oper. Res. **21**(2), 498–516 (1973)
8. Papadimitriou, C., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity, vol. 01. Prentice Hall, Englewood Cliffs (1982)
9. Steiglitz, K., Weiner,P.: Some improved algorithms for computer solution of the traveling salesman problem. In: Proceedings of the 6th Annual Allerton Conference on System and System Theory (1968)

A Computational Evaluation of Online ATSP Algorithms

Michele Barbato, Alberto Ceselli, and Filippo Mosconi

Abstract We prove that state-of-the-art online asymmetric traveling salesman problem algorithms can successfully be used in real time practical systems, in terms of both solutions quality and computational efficiency. At the same time, we show that such a good behaviour can only be obtained by a careful fine tuning of the algorithms, often clashing with their theoretical analysis.

Keywords Asymmetric traveling salesman · Online algorithms · Experimental analysis

1 Introduction

Server routing problems involve a set of requests to be served by a server in the minimum amount of time. Usually, every request is identified with a point in a space. A request is then served if the server visits the corresponding point. In the *online asymmetric traveling salesman problem* (OL-ATSP), requests are generated in sequence along time and each request must be served after it has been generated. In general, the generation time of the next request is unknown to the server. The goal in the OL-ATSP is to minimize the time at which all requests have been served.

The OL-ATSP and its variations arise in a number of real-world applications. Our interest in the OL-ATSP actually stems from its application to the management of automated warehouses [3], which is critical in highly custom production contexts like cosmetics manufacturing [1].

A warehouse consists of racks. They are typically identical, growing in vertical shelves of the same height, with an aisle between them which is traversed both horizontally and vertically by a stacker crane. The stacker crane moves containers to and from the racks: we refer to any such a movement as a *task*. One may assume

M. Barbato (✉) · A. Ceselli · F. Mosconi
Università Degli Studi di Milano, Dipartimento di Informatica, Milano, Italy
e-mail: michele.barbato@unimi.it; alberto.ceselli@unimi.it; filippo.mosconi@studenti.unimi.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_41

that the complete set of possible tasks is known in advance, and so are both the cost to perform each task and the time needed to start a task immediately after another one has been executed. Given a reasonable time horizon, only a small subset of all possible tasks appear: the goal is to find a sequence of them that can be performed by the stacker crane minimizing the total completion time.

Often in practical circumstances there is no knowledge on the temporal distribution of the tasks. That is, the order in which tasks will be revealed, as the time at which this will happen, is unknown. Algorithms operating the stacker crane in the above situation must therefore act in an online fashion: decisions at a given instant are taken by looking only at the sequence of requests generated until that instant. Under this assumption, the problem of optimally sequencing the completion of the tasks is exactly an OL-ATSP where the requests are the tasks to be performed and the server is the stacker crane.

The quality of OL-ATSP solutions, as in general with online algorithm, is assessed by comparing its cost to that potentially achievable by an exact offline algorithm, that is one where decisions are taken with complete knowledge on the sequence of requests. The worst-case ratio between the former and the latter is called competitive ratio. The lower the competitive ratio, the better the corresponding online algorithm. In this regard, the OL-ATSP is well understood, since algorithms providing optimal competitive ratio are known for several variants [5].

However these proven competitive algorithms do not necessarily exhibit good performance in practice, and a corresponding suitable experimental validation is often missing. The OL-ATSP is a relevant case: despite its high potential in applications, no experimental evaluation is carried out in the literature. Additionally, the complexity of the sub-problems that need to be solved undermines their practical applicability, finally imposing to resort to heuristics, thereby losing quality guarantees.

In this paper we fill in such a gap by performing an extensive experimental evaluation of OL-ATSP algorithms. We prove that state-of-the-art OL-ATSP algorithms can successfully be used in real time practical systems, in terms of both solutions quality and computational efficiency. At the same time, we show that such a good behaviour can only be obtained by a careful fine tuning of the algorithms, often clashing with the theoretical analysis.

More precisely, we experimentally study SMARTSTART, a family of algorithms for the OL-ATSP presented in [5] (Sect. 2). The theoretical analysis performed in [5] shows that, for several variations of the OL-ATSP, SMARTSTART yields algorithms with the best possible competitive ratio. We extensively test SMARTSTART algorithms, designing instances with specific spatial and temporal distribution of the requests (Sect. 3). We analyze the discrepancy between an experimental fine tuning of SMARTSTART and the theoretically predicted guarantees as the choice of these distributions changes (Sect. 3.1). We also consider lower bounding procedures, and use them to evaluate the actual gap obtained by SMARTSTART under various settings, thereby checking the tightness of the theoretical analysis in practical instances (Sect. 3.2). Finally, we evaluate the potential performance of SMARTSTART algorithms when employed in real-time applications (Sect. 3.3).

2 Definitions and SmartStart Algorithms

Throughout let V be a set of points in a space with a *distance* $d\colon V \times V \to \mathbb{R}_{>0}$ satisfying the *triangular inequality* $d(u,v)+d(v,w) \geq d(u,w)$ for every $u, v, w \in V$. In general, $d(u,v) \neq d(v,u)$, hence we model such a space as a complete digraph $D = (V, A)$, having weight $d(u,v)$ on arc $(u,v) \in A$. According to [5], asymmetry makes the problem harder. We select an *origin* vertex $O \in V$, where a server is located at time 0. To reach $v \in V$ from $u \in V$ the server employs a time $t_{uv} = d(u,v)$. A *request* is a pair $r = (v,t)$ with $v \in V \setminus \{O\}$ and $t \in \mathbb{R}_+$. Given request $r = (v,t)$, we define $\nu(r) = v$ and $\tau(r) = t$. We say that request r is *served* if the server visits $\nu(r)$ at any time $t \geq \tau(r)$, *unserved* otherwise. Let $\mathcal{R} = \{r_1, r_2, \ldots, r_k\}$ be a sequence of requests such that $k \leq |V|$ and $\tau(r_i) \leq \tau(r_j)$ whenever $1 \leq i < j \leq k$. In the *homing* OL-ATSP, the server must minimize the total *completion time*, that is, the time required to serve all requests in $\mathcal{R}$ and subsequently return at O. Note that the homing OL-ATSP where $k = |V|$ and $\tau(r) = 0$ for every $r \in \mathcal{R}$ is the classical offline asymmetric traveling salesman problem (ATSP), thoroughly described e.g., in [7]. Online algorithms operate the server by taking decisions on the fly, considering at time $t \geq 0$ only the requests r such that $\tau(r) \leq t$. Let $c_{\mathcal{A}}(\mathcal{R})$ be the completion time of a server operated by algorithm $\mathcal{A}$, defined as the time to serve all requests in $\mathcal{R}$ and return to O afterwards. Let $c_{\text{BEST}}(\mathcal{R})$ be the best possible completion time for the request sequence $\mathcal{R}$. Given $K \geq 0$, algorithm $\mathcal{A}$ is *K-competitive* if $c_{\mathcal{A}}(\mathcal{R})/K \leq c_{\text{BEST}}(\mathcal{R})$ for every request sequence $\mathcal{R}$. The real value K is the *competitive ratio* of $\mathcal{A}$.

SmartStart Algorithms SMARTSTART is a family of algorithms first introduced in [4] in the context of online dial-a-ride problems. It has been used for the OL-ATSP in [5]. Let us assume that requests are generated over time at some vertices of the digraph D described above. The behavior of SMARTSTART is parametric on a real value: for a fixed $\alpha > 0$ the corresponding algorithm $\mathcal{A}(\alpha)$ works as follows. At every $t \geq 0$ let $\mathcal{R}_t \subseteq \mathcal{R}$ be the subset of unserved requests r with $\tau(r) \leq t$. The server computes $d(\mathcal{R}_t)$, the value of the tour S needed to visit all requests of $\mathcal{R}_t$ starting from and returning to O. At the minimum t' such that $t' \geq \alpha d(\mathcal{R}_t)$ the server executes S serving all requests in $\mathcal{R}_t$ and ignoring all requests generated meanwhile. When the server is back at O the above process is repeated.

If the weights on the arcs of D obey the triangular inequality, the competitive ratio of $\mathcal{A}(\alpha)$ only depends on whether the last request arrives while the server is idle at O. Exploiting this fact the authors of [5] prove the following:

Theorem 1 ([5]) *For every $\alpha > 0$, $\mathcal{A}(\alpha)$ is $\max\{1+\alpha, 2+1/\alpha\}$-competitive for the homing OL-ATSP. The best* SMARTSTART *algorithm is $\mathcal{A}(\phi)$ where $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio.*

The value ϕ minimizes $\max\{1+\alpha, 2+1/\alpha\}$. Since $1+\phi = 2+1/\phi$, algorithm $\mathcal{A}(\phi)$ is $(1+\phi)$-competitive. The study in [5] also shows that $(1+\phi)$ is a lower bound on the competitive ratio of any possible OL-ATSP algorithm. Thus, $\mathcal{A}(\phi)$ yields the best competitive ratio among online algorithms for the homing OL-ATSP.

3 Experimental Analysis

In this section we address two experimental questions. The first concerns solutions quality, that is low completion time objective values, as defined in Sect. 2. In particular we investigate whether the theoretical best setting of α is effective also in experiments (Sects. 3.1 and 3.2). We stress that, by definition, SMARTSTART algorithms need an exact solver for the ATSP as a subroutine to serve partial sets of requests every time the server leaves the origin. Therefore, the second question concerns computing times, that is if the need of an exact ATSP solver turns out to be a bottleneck for the applicability of SMARTSTART or not (Sect. 3.3).

Dataset Design Our investigation started from preliminary experiments on TSPLib instances [9]. We found that the behavior of SMARTSTART on these instances was affected by requests appearing in clusters. In fact, assuming constant inter-arrival times, if requests are generated spatially very close to each other and far from the origin O, a good heuristic for the OL-ATSP would be to never return to O before having served all of them, as this would imply unnecessary round-trips between the origin and their locations. The same applies to the temporal distribution of the requests: if all data was known in advance a good heuristic would merge requests generated in short time intervals, exploiting large time gaps for returning back to O.

Therefore, we decided to build a more reliable experimental setting, by controlling both spatial and temporal distributions of the requests. Indeed, spatial and temporal distributions are intertwined in determining the complexity of an instance.

Concerning the spatial distribution, SMARTSTART algorithms are designed to start a tour serving subsets of requests based on the trade-off between distance to be traveled and elapsed time. As sketched above, we argue space cluster tendency to be predictive of easy instances. Oppositely, when requests are uniformly scattered, no particular rationale is obviously yielding good policies; in particular, SMARTSTART algorithms has no design elements to perform a wise choice of the subsets of requests to serve.

Motivated by the above arguments, digraphs $D = (V, A)$ in our dataset are constructed as follows. In the bi-dimensional Euclidean space, we initially consider a point O and six points $m_1, \ldots, m_6$ uniformly disposed on a large circumference with centre in O and radius 500. Next, we generate clusters $C_1, \ldots, C_6$ each including 20 points drawn at random from a bi-variate Gaussian distribution centred at m_i, with covariance matrix $[[50^2, 0], [0, 50^2]]$, for every $i = 1, \ldots, 6$. The origin of D is located at O, and all other vertices are located at the points of the clusters (note that the m_i's may not be vertices of D).

In order to have asymmetric distances between vertices, we define $d\colon A \to \mathbb{R}_{\geq 0}$ as:

$$d(v, w) = \left\lceil \left(||p(v) - p(w)||^2 + G \cdot \max\{0, y(w) - y(v)\} \right)^{1/2} \right\rceil \quad \forall v, w \in V \tag{1}$$

where for every $v \in V$, $p(v)$ (resp. $y(v)$) is the Euclidean point (resp. its y-coordinate) where vertex v is located, $||p - q||$ is the Euclidean distance between points p and q and $G > 0$ is an *asymmetry parameter*. It is not difficult to show that the arc weight function d defined in (1) satisfies the triangular inequality. Moreover, by varying the asymmetry parameter, we can easily control the total amount of *asymmetry degree* of D defined in [5] as $\sup_{v,w \in V} \frac{d(x,y)}{d(y,x)}$, which is relevant in the competitive analysis of online algorithms for the OL-ATSP. We remark that, given this setting, O is not guaranteed to be the center of the smallest circle containing all the points. On the whole, we created 50 weighted digraphs (D, d) with d having asymmetry parameter $G = 1$ and 50 additional digraphs with $G = 10$.

In the following we assume that the clusters $C_1, \ldots, C_6$ are disposed in clockwise order around O with C_i being the predecessor of C_{i+1}, for every $i = 1, \ldots, 5$. We consider three types of order in which the requests are generated: (1) CLUSTER order: 20 requests are generated consecutively in the same cluster, one at each vertex of that cluster. Then the operation repeats on successive clusters by following the cluster order $C_1, C_2, C_3, C_4, C_5, C_6$; (2) JUMPING order: 20 requests are generated *consecutively in the same cluster*, one at each vertex of that cluster. The operation is iterated following the cluster order $C_1, C_4, C_6, C_2, C_5, C_3$; (3) RANDOM order: *one request is generated in one vertex of a cluster*, among the vertices where no request has been generated yet. The operation is iterated cyclically following the cluster order $C_1, C_4, C_6, C_2, C_5, C_3$.

Concerning the temporal distribution we designed three scenarios: (a) Uniform: requests appear at constant time intervals; (b) Dense-first: let T be the time at which the last request appears. The first 75% requests are generated uniformly between 0 and time $T/2$; the remaining 25% requests are generated uniformly between time $T/2$ and T. (c) Sparse-first: as in Dense-first case but generating the first 25% requests before time $T/2$ and the last 75% between $T/2$ and T. In all three temporal distributions above, we let the time of the last generated request be equal to the length of the optimal ATSP solution on the corresponding instance, computed in preprocessing, in such a way that the inter-arrival times are of the same order of magnitude of the traveling time between requests.

For each of the 100 digraphs created as above we combine each spatial order from types (1)–(3) with each temporal distributions from types (a)–(c). Hence, overall, our dataset consists of 900 instances.

3.1 Effect of α

We first analyze how the behavior of SMARTSTART algorithm $\mathcal{A}(\alpha)$ varies according to the choice of the parameter α. According to Theorem 1, the value of α yielding the best competitive ratio on generic instances is $\alpha = \phi \simeq 1.6$. We performed tests with $\alpha \in \{0.2, 0.4, 0.8, 1.0, 1.4, 1.6\}$ on our dataset.

In Table 1 we provide the average completion times corresponding to several values of α in our set. In this table the first column corresponds to the three

Table 1 Average completion times for varying values of α

Spatial distribution	Temporal distribution	α					
		0.2	0.4	0.8	1	1.4	1.6
C	Uniform	11,063.22	**11,027.58**	11,154.02	13,883.85	16,699.46	18,091.09
	Dense-first	**10,476.31**	10,895.53	12,760.20	14,903.36	16,699.46	18,091.09
	Sparse-first	**11,783.40**	11,923.03	11,854.01	11,831.93	12,696.33	17,664.01
J	Uniform	**12,143.51**	12,303.67	12,186.78	13,999.04	16,699.46	18,091.09
	Dense-first	**11,507.02**	11,654.39	12,862.41	14,877.36	16,699.46	18,091.09
	Sparse-first	**13,102.64**	13,214.90	13,096.30	13,176.66	14,018.01	17,782.99
R	Uniform	15,175.50	15,503.38	15,728.75	**14,215.77**	16,699.46	18,091.09
	Dense-first	15,122.27	**13,193.61**	15,847.49	14,990.32	16,699.46	18,091.09
	Sparse-first	14,680.25	15,928.97	14,809.05	**14,336.07**	16,699.46	18,091.09

orders of request generation (C is for the CLUSTER order, J for JUMPING, R for RANDOM); similarly, the second column of Table 1 specifies the temporal distribution followed to generate the instances. The best average completion times of Table 1 are reported in boldface for every combination of spatial and temporal distributions. Table 1 highlights that on seven out of the nine types of instances the best average completion times correspond to $\alpha \in \{0.2, 0.4\}$. The other two types of instances are instead optimized in average by the value $\alpha = 1$ and belong to the random spatial distribution. We report that the theoretically optimal algorithm $\mathcal{A}(\phi)$ yielded the best completion time only on 1 instance out of the 900 tested instances.[1] Oppositely, for about 77% of instances the best α belongs to $\{0.2, 0.4, 0.8\}$. This suggests that our instances are better solved by SMARTSTART algorithms serving small sets of unserved requests, almost immediately after their generation. From these results we conclude that when considering structured OL-ATSP instances, the theoretical analysis of [5] can be imprecise in predicting the value of α giving the best SMARTSTART algorithm in practice. We mention that a similar result was reported in [2] on the online *symmetric* traveling salesman problem. On the other hand, for less structured instances, our experiments seem to indicate that the theoretically best α is closer to the best experimental α.

[1]The instance has $G = 1$, following a CLUSTER ordering of the requests and a sparse-first distribution.

Table 2 Comparison of SMARTSTART with a theoretical lower bound

Spatial distribution	Temporal distribution	$\mathcal{L}$	$\mathcal{A}$	Gap%
C	Uniform	7423.65	11,027.58	48.55
	Dense-first	7500.01	10,476.31	39.68
	Sparse-first	7422.70	11,783.40	58.75
J	Uniform	7392.34	12,143.51	64.27
	Dense-first	7468.70	11,507.02	54.07
	Sparse-first	7391.39	13,102.64	77.27
R	Uniform	7392.34	14,215.77	92.30
	Dense-first	7468.70	13,193.61	76.65
	Sparse-first	7391.39	14,336.07	93.96

3.2 Lower Bound Comparison

We now compare the results described in Sect. 3.1 with a theoretical lower bound for the OL-ATSP. Given an instance of the OL-ATSP, let $\bar{r}$ be its last generated request. Then a valid lower bound on the optimal completion time is $\tau(\bar{r}) + d(\nu(\bar{r}), O)$. Indeed, a server at least waits until time $\tau(\bar{r})$ for the last request to arrive and uses at least $d(\nu(\bar{r}), O)$ time units to go from $\nu(\bar{r})$ to O (visiting $\nu(\bar{r})$ is required to serve $\bar{r}$).

In Table 2, we compare SMARTSTART and the theoretical lower bound. Every row of the table represents a fixed instance type, obtained combining a spatial and a temporal distribution, as indicated by the first two columns while column $\mathcal{L}$ reports the average lower bounds on instances belonging to a given type. For each instance type, column $\mathcal{A}$ reports the average completion time of the SMARTSTART algorithms yielding the best average completion times on that instance type. Note that the values in column $\mathcal{A}$ correspond to the boldface values of Table 1. The last column of the table reports the relative gap between the values of column $\mathcal{A}$ and column $\mathcal{L}$, computed as $100 \cdot \frac{\mathcal{A}-\mathcal{L}}{\mathcal{L}}$.

On instances with sparse-first temporal distribution or random ordering generation of requests, the best SMARTSTART algorithm in average yields large relative increasing with respect to the average lower bound. However, on all other instances the best SMARTSTART algorithm in average yields completion times which are less than 60% greater than the corresponding average lower bound.

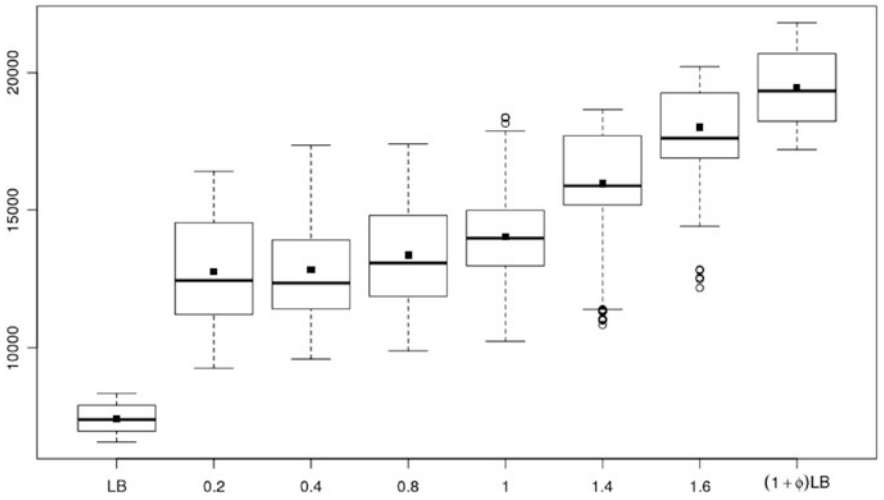


Fig. 1 Comparison of several SMARTSTART algorithms and the lower bounds computed on our dataset

We recall that, by results in [5], the worst-case competitive ratio of $\mathcal{A}(\alpha)$ is $1+\phi$ for every $\alpha > 0$. Since $\phi \simeq 1.6$ this results in a worst-case relative increasing with respect to an optimal offline algorithm of roughly 160%, for every SMARTSTART algorithm. Table 2 indicates that, in practice, SMARTSTART algorithms never attain this worst-case competitive ratio. In fact, this ratio is not reached even by comparing with the average lower bound in place of the optimum.

In Fig. 1 we provide a more detailed view of the same experiment. There, the y-axis represents values of completion times; one boxplot summarizes the distribution of solution values for each choice of $\alpha \in \{0.2, 0.4, 0.8, 1.0, 1.4, 1.6\}$, as indicated on the x-axis. The leftmost (resp. rightmost) boxplot refers to LB values (resp. $(1+\phi) \cdot LB$ values, that is, the *best* competitive ratio of SMARTSTART). Average values are marked with filled squares. First, we observe that the trend of each quartile reflects that of average values (see Table 1). Second, we notice that $\mathcal{A}(1.6)$ solution values are in general lower than $(1+\phi) \cdot LB$; in turn, we know the theoretical analysis to be tight, so we expect at least random instances to exist in our dataset on which $\mathcal{A}(1.6)$ approaches the theoretical worst case $(1+\phi) \cdot OPT$: in those instances LB is approaching OPT; at the same time, the dispersion of LB values is low. This makes us conjecture that the LB is of good quality in our instances.

3.3 Realtime Applicability

As discussed, SMARTSTART involves the iterative resolution of ATSPs. While being an **NP**-hard problem, the size of ATSP instances manageable by current state-of-the-art solvers is matching the size of instances arising from real-world applications. Hence, it is of interest to evaluate the actual applicability of SMARTSTART in real-world OL-ATSPs.

We proceed as follows. For all $t \geq 0$, let $\mathcal{R}_t$ be the set of unserved requests, the last of which appearing at instant t and let $S(\mathcal{R}_t)$ be the time $\mathcal{A}(\alpha)$ employs to solve the ATSP on $\mathcal{R}_t \cup \{O\}$. It never pays off to postpone the resolution of such an ATSP. Then $\mathcal{A}(\alpha)$ has a *timeout* whenever there exists a new request r appearing between t and $t + S(\mathcal{R}_t)$. That is, in a real system, a new request appears while the algorithm is still evaluating the previous ones. We use the probability of occurrence of a timeout as a measure of applicability of SMARTSTART in a real-time context.

In fact, the occurrence of a timeout is a random variable $Y = 1$ if $T < S$, $Y = 0$ otherwise where in turn T is a random variable modeling the time interval between two consecutive requests, and S is a random variable modeling the ATSP resolution time using the ATSP solver. In our experiments we decided to map ATSP instances to instances of the symmetric traveling salesman problem by means of a standard Karp reduction [8], subsequently solved with Concorde [6], a state-of-the-art solver for the symmetric traveling salesman problem. We approximate the cumulative distribution function of S by its empirical counterpart through numerical simulation, considering all ATSP runs in our experiments (that is 401.925 heterogeneous Concorde single thread calls on a 4.00 GHz Intel(R) Core(TM) i7-6700K and 32 GB RAM). In Fig. 2 (top) we report such an empirical cumulative distribution function $F(x)$, that is estimating the probability that a Concorde run employs at most x seconds. For what concerns T, instead, we assumed an exponential distribution for the time intervals between two consecutive requests, whose density function is $q(x) = \lambda e^{-\lambda x}$, that is a standard modeling of independent inter-arrival times. We finally estimated

$$\mathbb{E}[Y] = P[Y = 1] = P[S > T] =$$
$$= \int_{x=0}^{+\infty} P[S > T | T = x] P[T = x] dx = \int_{x=0}^{+\infty} P[S > x | T = x] P[T = x] dx$$

by numerically computing $\int_{x=0}^{+\infty} (1 - F(x)) \cdot q(x) \cdot dx$.

That is, we statistically determine the probability of a SMARTSTART timeout in our setting, for varying values $1/\lambda$ of the average time intervals between consecutive requests. Results are plotted in Fig. 2 (bottom). We deduce that the timeout probability approaches 10% already for average time intervals not smaller than 7 s. As a conclusion, SMARTSTART can be considered a viable solution to solve OL-ATSPs whose requests are generated at a high frequency (of the order of $\simeq$5 s) and hence successfully embeddable in modern systems handling real-time order satisfaction like automated warehouses.

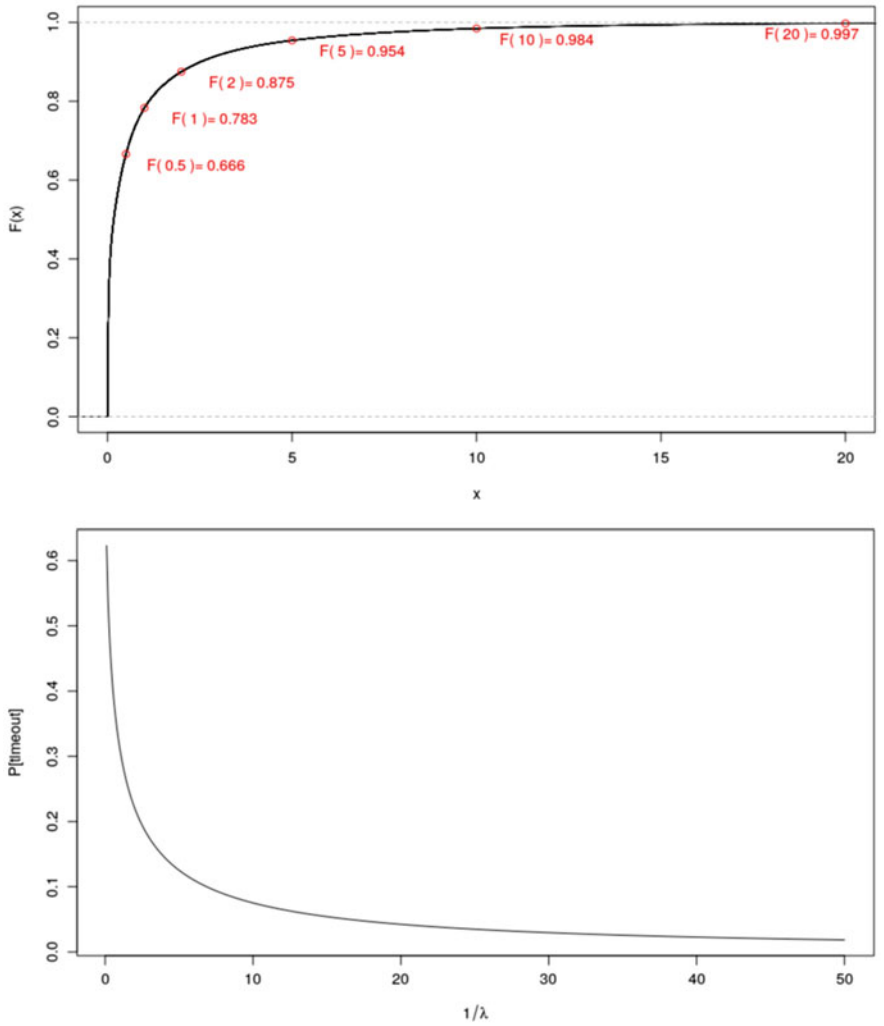


Fig. 2 Distribution of Concorde run times (top) and prob. of SMARTSTART timeouts (bottom)

4 Conclusions

In this paper we have performed an experimental analysis of SMARTSTART, a family of algorithms for the OL-ATSP. First, we have shown that the parameter α determining the SMARTSTART behavior requires careful fine-tuning to take advantage of a prior knowledge on instance structure. In particular, often on structured instances the best theoretical α resulted in SMARTSTART algorithms behaving poorly in practice. Even if more aggressive settings of α always pay off in

our experiments, the theoretical analysis of [5] becomes closer to our results when requests are spatially more randomly distributed. We have additionally observed that well-tuned SMARTSTART algorithms may produce solutions whose value is nearly optimal. This result was shown by comparing the quality of SMARTSTART solutions with a theoretical lower bound. Finally, we have evaluated the usability of SMARTSTART in real-time applications by means of a statistical framework, concluding that, also thanks to the performance level reached by exact ATSP solvers and modern hardware, SMARTSTART can currently be embedded in realtime systems dealing with OL-ATSP applications.

Acknowledgement This research was partially funded by Regione Lombardia, grant agreement n.E97F17000000009, project AD-COM.

References

1. AD-COM: Project website. https://ad-com.net/. Last access April 2019
2. Aprea, M., Feuerstein, E., Sadovoy, G., de Loma, A.S.: Discrete online TSP. In: International Conference on Algorithmic Applications in Management, pp. 29–42. Springer, Berlin (2009)
3. Ascheuer, N., Grötschel, M., Abdel-Hamid, A.A.-A.: Order picking in an automatic warehouse: solving online asymmetric TSPs. Math. Methods Oper. Res. **49**(3), 501–515 (1999)
4. Ascheuer, N., Krumke, S.O., Rambau, J.: Online dial-a-ride problems: minimizing the completion time. In: Reichel, H., Tison, S. (eds.) STACS 2000, Berlin, Heidelberg, pp. 639–650. Springer, Berlin (2000)
5. Ausiello, G., Bonifaci, V., Laura, L.: The on-line asymmetric traveling salesman problem. J. Discrete Algorithms **6**(2), 290–298 (2008)
6. Concorde, Applegate, D.L., Bixby, R.E., Chvatal, V., Cook, W.J.: http://www.math.uwaterloo.ca/tsp/concorde.html (2003)
7. Gutin, G., Punnen, A.P.: The Traveling Salesman Problem and its Variations, vol. 12. Springer Science & Business Media, New York (2006)
8. Roberti, R., Toth, P.: Models and algorithms for the asymmetric traveling salesman problem: an experimental comparison. EURO J. Transp. Logist. **1**(1), 113–133 (2012)
9. TSPLib: Maintained by G. Reinelt. https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/ (2013)

Modeling of Traffic Flows in Internet of Things Using Renewal Approximation

Florian Wamser, Phuoc Tran-Gia, Stefan Geißler, and Tobias Hoßfeld

Abstract This paper proposes a versatile approach to model aggregated traffic flows in the Internet of Things (IoT) using renewal approximation. The modeled traffic originates from a large number of sources or devices consisting of a set of sensors mixed with classical elastic random traffic modeled as Poisson arrival process. The work shows the exact derivation in the simple case for periodic sensors. It shows further results in the mixed case with periodic sensors and a background process. The renewal approximation allows to derive the required number of sensors such that the aggregated traffic can be approximated as Poisson process.

Keywords Internet of Things · Traffic modeling · Renewal approximation

1 Introduction

Internet of Things (IoT) is a growing area in mobile communication applications [1, 2]. It is expected that millions of devices will be found on the networks in the near future, each sending independently or via gateways over the mobile network. In such a scenario, IoT devices encompass all types of physical nodes or objects that are connected to the Internet to receive and respond to requests, or to store data. IoT devices can be subdivided into (1) stand-alone devices with independent Internet connectivity, (2) device groups that communicate in an aggregated manner with servers through Internet gateways on the Internet, or (3) devices that communicate with one another based on direct peer-to-peer connections.

A typical situation is the aggregation of traffic streams from many independent sensors to an Internet gateway as described in [3, 4]. This class of sensors send data at periodically time-fixed intervals to store measurements or to request input

F. Wamser (✉) · P. Tran-Gia · S. Geißler · T. Hoßfeld
University of Würzburg, Institute of Computer Science, Würzburg, Germany
e-mail: florian.wamser@informatik.uni-wuerzburg.de; trangia@informatik.uni-wuerzburg.de; stefan.geissler@informatik.uni-wuerzburg.de; hossfeld@informatik.uni-wuerzburg.de

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_42

data and updates. The time-fixed intervals result from mechanisms to conserve power or from continuous measurements at specific time intervals such as in Smart Grids [4, 5]. As in the classic Internet, this traffic overlaps with background traffic from various other sources, which can be considered independent due to the large number of sources.

In previous work, aggregated IoT traffic was modeled for periodic traffic with a fixed sending period of the individual sensor node intervals [3, 4, 6]. We extend this idea with an additional component, a random and heterogeneous background traffic, as described in [4]. To this end, a closed-form expression for the approximation of the inter-arrival time distribution of the superposition of different arrival processes of IoT devices is derived in this work using renewal approximation. We consider a class of sensors that send data at consecutive, time-fixed intervals combined with a continuous-time Markov arrival stream in form of a background Poisson random process. We provide a detailed derivation for the approximation of the inter-arrival time distribution based on the renewal approximation including an exact determination of the coefficient of variation, which can be well approximated by a Poisson process such that the statistical differences are below a threshold ϵ.

The rest of the paper is structured as follows. After this introduction, in Sect. 2 related work is discussed. In Sect. 3, the aggregated IoT traffic is modeled for an Internet gateway. We introduce the used notation and definitions and provide a detailed description of the approach. Further on, two cases are described in detail in Sects. 3.1 and 3.2. After showing numerical results in Sect. 4, we conclude the work in Sect. 5.

2 Related Work

The superposition of a number of deterministic flows is a subject of various papers in the last decades, especially during the development of Asynchronous Transfer Mode (ATM) technology [6, 7]. The resulting process of n deterministic flows, each of rate $1/T$, is a periodic non-renewal process of the same period T [8, 9]. Assuming now that the traffic sources are independent, e.g. due to a very large number of sources, one can model the superposition of deterministic point processes as a Poisson process as limiting case [8]. There are papers [10, 11] that discuss the renewal assumption that holds true in the Poisson case and does not hold for deterministic processes when the number is small. In this paper we apply the renewal approximation and check whether and when it is valid. Further work on traffic modeling can be found in e.g. [3, 4, 12]. Directly linked to our work are the works of Metzger et al. [4] and Hoßfeld et al. [3]. They both refer to the same modeling context as the present work. These papers are pioneer in this area, facing the same problem, and providing basic ideas for IoT traffic flows modeling. Our work is based on these approaches and complements them with further definitions. Our approach is different in that we apply the renewal approximation to derive a closed form. In [4] a comprehensive list of IoT traffic models is given, showing how important periodic traffic characteristics are in the IoT environment.

3 Modeling Aggregated Traffic Flows

The modeling of aggregated traffic flows in the IoT environment relies on a fundamental consideration and definition of the arrival processes of the individual sources. As described in [4], the predominant consideration of these traffic flows in the literature is the Poisson arrival process. This is in contrast to the work from ATM times [6], which specifies the need for more detailed consideration of periodic traffic. In the following, periodic traffic flows are defined and modeled in detail. We use a description consistent with [4], from which we derive the distribution function and its moments. The latter serves to answer the question of how many devices aggregated traffic flows can be approximated as Poisson process.

Variables and Notation As the resulting processes of flows in IoT environments are generally point processes, we employ a renewal approximation technique to derive the inter-arrival distribution function of the resulting flow, assuming it follows renewal input process properties [13]. The main steps of the renewal approximation used in the analysis are:

1. Consider an independent outside observer looking at the process at an independent point in time.
2. Derive the residual time distribution of the resulting process, i.e. the interval from observation instance until the next arrival to occur.
3. With the assumption that the resulting process is a renewal process we then derive the inter-arrival distribution function out of the forward residual time.

In the following, we consider the scenario described in Fig. 1. There is a group of different sensors. Each device sends periodically messages with period T_1. There are n_1 devices in this class. A node k starts randomly at time $t_{1,k} \in [0; T_1]$ and thus, the sending times are $t_{1,k} + z \cdot T_1$ with $z \in \mathbb{N}$. We denote the distribution function for this process from the sensor nodes as $A_1(t)$, respectively $a_1(t)$ as density distribution function. The traffic pattern for this group is repeated after period T which is the least common multiple of the sending periods for this class. The resulting stream

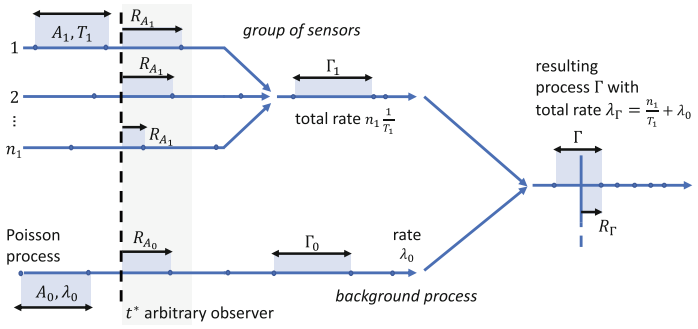


Fig. 1 General model with n_1 deterministic sensor sources plus Poisson source

Table 1 Notation

T_1	$\triangleq$	Sending period of a sensor; without loss of generality, we assume $T_1 \in \mathbb{N}$
n_1	$\triangleq$	Number of devices within the group of sensors
A_1	$\triangleq$	Inter-arrival time of one input process $A_1(t)$, $a_1(t)$
$R_{A_1}(t)$	$\triangleq$	Distribution function of residual time $R_{A_1}(t) = P(R_{A_1} \leq t)$
$r_{A_1}(t)$	$\triangleq$	Density function of residual time
$R^C_{A_1}$	$\triangleq$	Complementary distribution function of residual time $R^C_{A_1} = P(R_{A_1} > t)$
Γ	$\triangleq$	Inter-arrival time of the resulting process
R_Γ	$\triangleq$	Residual time of the resulting process

originating from the group of sensors is described with Γ_1 as inter-arrival time of the resulting process. We model a background random Poisson process as Γ_0 and rate λ_0. The final aggregated process of the sensors and the background process is denoted as Γ with a total rate of $\lambda_\Gamma = n_1/T_1 + \lambda_0$ (Table 1).

Residual Time Distribution The residual time or the forward recurrence time is the time between any random observation time until the next arrival. We consider a random observer looking at the process, the interval to the next observed arrival is denoted by the random variable R_Γ. The residual time until the next message arrival is the minimum of the residual time of participating processes:

$$R_\Gamma = \min(\underbrace{R_{A_1}, R_{A_1}, \ldots, R_{A_1}}_{n_1 \text{ times}}, R_{A_0}). \tag{1}$$

This leads to the complementary cumulative distribution function (CCDF) of the resulting process

$$P(R_\Gamma > t) = 1 - R_\Gamma(t) = \underbrace{P(R_{A_1} > t) \cdot P(R_{A_1} > t) \cdot \ldots \cdot P(R_{A_1} > t)}_{n_1 \text{ times}} \cdot P(R_{A_0} > t). \tag{2}$$

We obtain subsequently the distribution function of the residual time of the resulting process. In assuming the resulting process to be a renewal process, we can use the basic result of renewal theory $r(t) = \frac{1}{E[\Gamma]}(1 - \Gamma(t))$ to derive the inter-arrival time distribution with Eq. (2)

$$\Gamma(t) = 1 - E[\Gamma] \cdot r(t) = 1 - E[\Gamma] \cdot \frac{d}{dt} R_\Gamma(t). \tag{3}$$

It is obvious that the aggregated stream of deterministic traffic processes is non-renewal. However, with the superposition of a very large number of processes, like in IoT environments with huge sets of sensors, the inter-arrival time occurs

in microscopic scale compared to the periodicity of a single participating process. We expect that the resulting process is "more renewal" with a growing number of superimposed processes. In this paper, we investigate under which conditions the results using renewal approximation is accurate enough for practical use in IoT systems and try to quantify accuracy of the renewal approximation. In the following we consider two consecutive cases and model their properties.

3.1 *n D: Deterministic Case for a Group of Periodic Sensors*

This case outlines the aggregation of n_1 deterministic flows solely to an aggregated stream of IoT traffic, in our case Γ_1 see Fig. 1. In the IoT context this model is employed to describe a (large) number of measurement data flows from a set of sensors. This process was also often used to model ATM traffic flows on aggregated cell patterns [6], where it is often denoted as nD.

This basic model is depicted in Fig. 2 with an arbitrary observer at t^*. Each of the input processes, e.g. to represent traffic emitting from a sensor, is a deterministic process with inter-arrival time A_1 with distance T_1 and flow rate $\frac{1}{T_1}$. The corresponding CDF $R_{A_1}(t)$ and the probability density (PDF) of the recurrence time of A_1 are:

$$R_{A_1}(t) = P(R_{A_1} \leq t) = \begin{cases} 0 & \text{for } t < 0 \\ t/T_1 & \text{for } 0 \leq t \leq T_1 \\ 1 & \text{for } t > T_1 \end{cases}, \tag{4}$$

$$\frac{d}{dt} R_{A_1}(t) = r_{A_1}(t) = \begin{cases} 1/T_1 & \text{for } 0 \leq t \leq T_1 \\ 0 & \text{otherwise} \end{cases}. \tag{5}$$

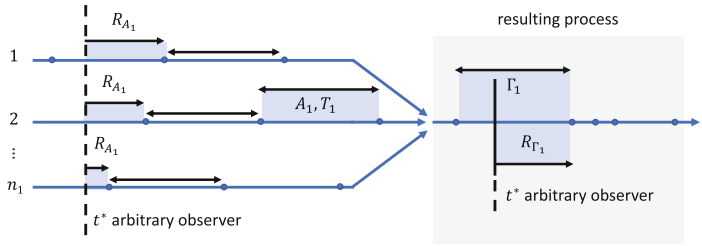


Fig. 2 Model Case 1 with arrival processes according to A_1, fixed T_1, and n_1 streams

With Γ_1 denoting the random variable of the inter-arrival time of the resulting process with corresponding residual time R_{Γ_1}, we obtain the residual time distribution function (of the superposition Γ_1 of n_1 deterministic flows) with density as

$$P(R_{\Gamma_1} \leq t) = R_{\Gamma_1}(t) = \begin{cases} 0 & t < 0 \\ 1 - (1 - \frac{1}{T_1}t)^{n_1} & 0 \leq t < T_1 \\ 1 & t \geq T_1 \end{cases}, \tag{6}$$

$$\frac{d}{dt} R_{\Gamma_1}(t) = r_{\Gamma_1}(t) = \begin{cases} \frac{n_1}{T_1}(1 - \frac{1}{T_1}t)^{n_1 - 1} & 0 \leq t < T_1 \\ 0 & \text{otherwise} \end{cases}. \tag{7}$$

Assuming the renewal property for the resulting process, we arrive at

$$\Gamma_1(t) = 1 - \frac{T_1}{n_1} r_{\Gamma_1}(t) = \begin{cases} 0 & t < 0 \\ 1 - (1 - \frac{1}{T_1}t)^{n_1 - 1} & 0 \leq t < T_1 \\ 1 & t \geq T_1 \end{cases}. \tag{8}$$

As discussed above, in general, the resulting process is non-renewal. During an interval of length T_1, there are exactly n_1 arrivals, which form a periodic pattern depending on the starting constellation of the flows. Thus, microscopically, the process is periodic, with infinite number of possible patterns. If n_1 is sufficiently large, from microscopic views, during a time interval sufficiently smaller than T_1, the inter-arrival process appears more random and a renewal process approximation appears more sensible. In the IoT context this model is employed to describe a (sufficiently large) number of measurement data flows from a set of sensors. We expect that if n_1 becomes large enough, the resulting process will quickly approach Poisson. We try here to compute this limit analytically.

From Eq. (8), we can assess the accuracy of the renewal approximation in more detail. The variance and the coefficient of variation of the resulting process are:

$$Var[\Gamma_1] = \frac{T_1^2 (n_1 - 1)}{n_1^2 (n_1 + 1)}, \qquad c_{\Gamma_1} = \frac{\sqrt{Var[\Gamma_1]}}{E[\Gamma_1]} = \sqrt{\frac{n_1 - 1}{n_1 + 1}}. \tag{9}$$

It can be seen from this expression that the coefficient of variation of the resulting flow just depends on the number n_1 of aggregated flows, not from the inter-arrival distance T_1. Furthermore for the case of one flow, $c_{\Gamma_1} = 0$ as expected for a deterministic process. For the limiting case $n_1 \to \infty$, we obtain $c_{\Gamma_1} \to 1$, which corresponds to the Markovian property. The resulting process approaches a Poisson process.

If we set a threshold $c_{\Gamma_{95\%}} = 0.95$ to answer the question, how many flows we need to deliver for a process with 95% of the randomness of a Poisson stream, we arrive at $n_1 = 19.51$, i.e. with just $n_1 = 20$ flows, the superposition on average can already approximated with a Poisson process with more than 95% accuracy.

3.2 *nD + M: Mixed Case with Periodic Sensors and a Background Process*

The derivation of the case with an additional Poisson background traffic is analogous to the basic case above. This case is shown in Fig. 1. There are n_1 sensors periodically sending data. The sending period is T_1. The nodes start randomly within $[0, T_1]$. There is also background traffic with rate λ_0 with inter-arrival times A_0 following a negative-exponential distribution function: $A_0(t) = 1 - e^{-\lambda_0 t}$. The residual time R_0 has the same expression as for A_0.

The residual time of a sensor is $R_{A_1} \sim U(0, T_1)$ with CDF $R_{A_1}(t) = t/T_1$ for $0 \leq t \leq T_1$. The residual time for the aggregated traffic is $R_\Gamma = min(R_{A_1}, R_{A_1}, \ldots, R_{A_1}, R_{A_0})$. With $R_0 = A_0$, the CDF is given with

$$R_{\Gamma_1}(t) = \begin{cases} 0 & t < 0 \\ 1 - (1 - \frac{1}{T_1}t)^{n_1} \cdot e^{-\lambda_0 t} & 0 \leq t < T_1 \\ 1 & t \geq T_1 \end{cases} . \tag{10}$$

The interarrival time distribution $\Gamma(t)$ can be derived using Eq. (3) with $E[\Gamma] = \frac{T_1}{n_1 + \lambda_0 T_1}$. The CDF for $\Gamma(t)$ for $0 \leq t < T_1$ derives to

$$\Gamma(t) = 1 - \frac{T_1 e^{-\lambda_0 t} \left(1 - \frac{t}{T_1}\right)^{n_1} (n_1 + \lambda_0 (T_1 - t))}{(T_1 - t)(n_1 + \lambda_0 T_1)}. \tag{11}$$

The coefficient of variation c_Γ can be derived in this case with standard mathematical tools analogous to the result of the basic case described above. The coefficient of variation is shown and explained below in the result section.

4 Numerical Results

In order to substantiate and validate our results, we compare the results obtained by renewal approximation with (1) an event-by-event simulation of an exact point process and (2) results from previous works [4]. The simulation randomly generates a sufficiently long point process according to the given properties, over which the same statistical measures can be derived after many iterations as obtained

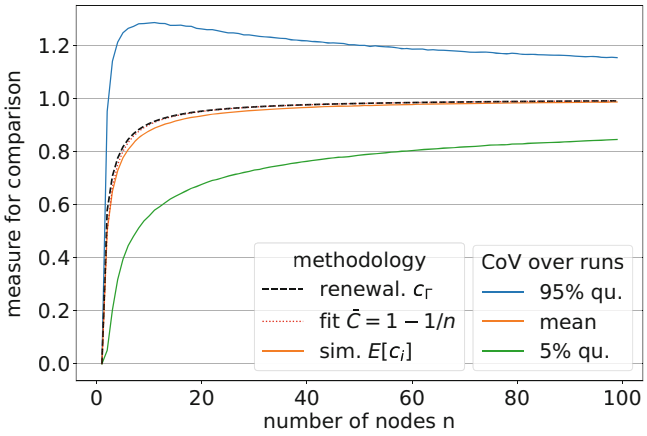


Fig. 3 Comparison of coefficients of variation: simulation, results from [4] and renewal approximation for different number of sensor nodes

analytically. For smart city use cases, typical sensor periods are 1 h and 12 h as e.g. proposed by 3GPP, see [4] for an overview on IoT traffic models. Hence, we assume $T_1 = 1\,\text{h}$ and $T_2 = 12\,\text{h}$.

In Fig. 3, the coefficient of variation (CoV) is depicted for the deterministic case, where the CoV is shown as function of the number of sensor nodes. It can be seen when the coefficient of variation reaches certain values to justify the approximation by a Poisson process, e.g. $c_{\Gamma_1} = 0.95$ or $c_{\Gamma_1} = 1$. Here, only the aggregated periodic traffic of n_1 nodes is considered with period T_1. Since the simulation generates a point process with a specific deterministic pattern for each run, it is possible to plot the CoV as a distribution over all appearing instances. With this, and in addition to the analytic result, Fig. 3 shows (1) the coefficient of variation according to the renewal approximation, (2) the mean of the coefficient of variation from the simulation of 1000 random superpositions, (3) the quantiles of this simulation, and (4) the fitted result as specified by paper [4] with $\bar{C} = 1 - \frac{1}{n}$.

Both the empirically fitted formula from [4] and the value of the renewal approximation are close to the values of the simulation of many instances of the exact process. Furthermore, all curves run together with a large number of sensor nodes n_1. Nevertheless, the quantiles show that the mean values conceal the extreme cases. With a small number of devices, e.g. $n_1 = 20$, the mean increases to 1, but the 5% and 95% quantiles are still more than 20% away from the mean value. Overall, the renewal approximation can be used as a simple closed-form expression if one considers a high number of nodes and also takes into account the quantiles, which show that there are some highly variable occurrence of arrivals in individual cases.

For the numerical results of the mixed case with deterministic arrivals and a Poisson background arrival process, we consider a scenario with $\beta = 10\%$ of background traffic. The aggregated periodic traffic leads to an arrival rate n_1/T_1.

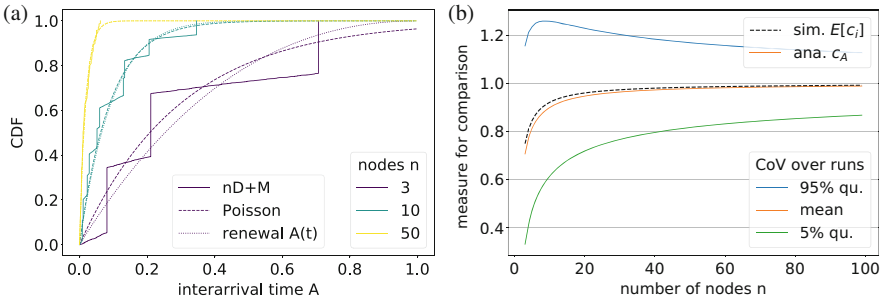


Fig. 4 Mixed case $nD + M$, periodic traffic of sensor nodes and background Poisson traffic. (**a**) Comparison of $nD + M$ with different number of devices n_1 in the deterministic case for Poisson process, renewal approximation, and a single simulation run. (**b**) Coefficients of variation for the aggregated case with background traffic from simulation and renewal approximation on the number of sensor nodes

Hence, the arrival rate of the background traffic is $\lambda_0 = \beta\lambda_\Gamma = \beta(n_1/T_1 + \lambda_0)$ which leads to $\lambda_0 = \frac{\beta}{1-\beta}\frac{n_1}{T_1}$.

Figure 4a shows a comparison between a single simulation run of $nD + M$, Poisson process, and renewal approximation. All results are plotted for $n = 3, 10, 50$ nodes and $T_1 = 1$ h with a ratio of $\beta = 0.1$ of Poisson background traffic. It shows the convergence of both the simulation runs to the renewal approach and the convergence of all approaches with a large number of nodes. With a larger number of devices, the single simulation run curve loses its steps, i.e., visually the convergence to the curve of the renewal approximation can be viewed in this figure.

On the basis of this, the coefficient of variation over the number of sensors is shown in Fig. 4b to discuss the approximation using renewal assumption in the mixed case $nD + M$. The black dashed line is the analytic solution using renewal approximation. The numerical results are derived using numerical integration. The solid lines are from simulation runs. We use $T_1 = 1$ h and vary n_1. We keep a constant ratio of background traffic which is again $\beta = 0.1$.

The simulation and analytic solution from the renewal approximation coincide; in fact, they converge for a large number of sensor nodes. Hence, the renewal approximation can also be used here for a large number of nodes. The results in this case, however, again show large distances to the 5% and 95% quantiles of the simulation runs, which is also due to the low ratio of background traffic with $\beta = 0.1$. If β increases, the curve from the analytic solution approaches 1 more quickly, which means the process becomes more random and converges faster to a random process where the renewal approximation can be employed.

5 Conclusion

This paper had the objective to describe an aggregated traffic mix of IoT devices with (1) periodic traffic patterns and (2) background traffic using renewal approximation. It is based on the papers [3, 4]. In contrast to them, in this paper a closed-form expression of the approximation of the distribution function for the aggregated traffic mix is derived using renewal approximation. Both the simple case with periodic-sending sensors and the mixed case with Poisson background traffic were calculated. The numerical results demonstrate the consistency of this approach with simulated instances of an exact point processes for a large number of devices. In the analytical form, it is shown in this paper that the coefficient of variation for $n \geq 20$ goes sufficiently against 1 and allows to quantify the required nodes, such that the aggregated traffic can be approximated by a Poisson process.

References

1. Shancang Li, Li Da Xu, and Shanshan Zhao. 5g internet of things: A survey. Journal of Industrial Information Integration, 10, 1–9, 2018.
2. I-Scoop, 5G and IoT in 2018 and beyond: the mobile broadband future of IoT. https://www.i-scoop.eu/internet-of-things-guide/5g-iot/. Accessed 03 Apr 2019.
3. Tobias Hoßfeld, Florian Metzger, and Poul E Heegaard. Traffic modeling for aggregated periodic IoT data. In 2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), pages 1–8. IEEE, 2018.
4. Florian Metzger, Tobias Hoßfeld, André Bauer, Samuel Kounev, and Poul E Heegaard. Modeling of aggregated iot traffic and its application to an iot cloud. Proceedings of the IEEE, 107(4), 679–694, 2019.
5. Rapeepat Ratasuk, Athul Prasad, Zexian Li, Amitava Ghosh, and Mikko Uusitalo. Recent advancements in M2M communications in 4G networks and evolution towards 5G. In 2015 18th International Conference on Intelligence in Next Generation Networks, pages 52–57. IEEE, 2015.
6. James Roberts and Jorma Virtamo. The superposition of periodic cell arrival streams in an atm multiplexer. IEEE Transactions on Communications, 39(2), 298–303, 1991.
7. Ottmar Gihr and Phuoc Tran-Gia. A layered description of atm cell traffic streams and correlation analysis. In IEEE INFCOM'91. The conference on Computer Communications. Tenth Annual Joint Conference of the IEEE Computer and Communications Societies Proceedings, pages 137–144. IEEE, 1991.
8. David Roxbee Cox and Walter Smith. On the superposition of renewal processes. Biometrika, 41(1–2), 91–99, 1954.
9. Aleksandr Yakovlevich Khinchin, DM Andrews, and Maurice Henry Quenouille. Mathematical methods in the theory of queuing. Courier Corporation, 2013.
10. Susan L Albin. On Poisson approximations for superposition arrival processes in queues. Management Science, 28(2), 126–137, 1982.
11. Ward Whitt. Approximating a point process by a renewal process, i: Two basic methods. Operations Research, 30(1), 125–147, 1982.
12. Muhammad Asad Arfeen, Krzysztof Pawlikowski, Andreas Willig, and Don McNickle. Internet traffic modelling: from superposition to scaling. IET networks, 3(1):30–40, 2014.
13. David Roxbee Cox and Hilton David Miller. The theory of stochastic processes. Chapman and Hall, New York, 1965.

Flow Assignment in Multi-Core Network Processors

Franco Davoli, Mario Marchese, and Fabio Patrone

Abstract In modern telecommunication networks, the trend toward "softwarization" is shifting the execution of switching and protocol functionalities from specialized devices to general purpose hardware located in datacenters or at the network edge. Incoming flows generated by User Equipment are processed by different functional modules executed in Virtual Machines (VMs) or containers. The paper considers a modeling and control architecture in this environment, for the assignment of flows to the first functional blocks in a chain of Virtual Network Functions (VNFs) and the balancing of the load among the VMs where they are executed.

Keywords Virtual network functions · Network flow optimization · Load balancing

1 Introduction

Telecommunication networks are undergoing a profound evolution, which is bringing part of their infrastructure ever closer to that of computing systems. With the advent of Software Defined Networking (SDN) [1] and Network Functions Virtualization (NFV) [2], Network Service Providers (NSPs) have started considering an increasing level of "softwarization" of the functionalities to be performed, especially as regards the access segment [3]. This trend has been further strengthened by Mobile Edge Computing (MEC) [4, 5], and by the consolidation of the fifth generation of mobile networks (5G) [6], providing a much stronger integration

F. Davoli (✉) · M. Marchese · F. Patrone
Department of Electrical, Electronic and Telecommunications Engineering, and Naval Architecture (DITEN), University of Genoa, Genoa, Italy

National Laboratory of Smart and Secure Networks (S2N), National Inter-University Consortium for Telecommunications (CNIT), Genoa, Italy
e-mail: franco.davoli@unige.it

M. Paolucci et al. (eds.), *Advances in Optimization and Decision Science for Society, Services and Enterprises*, AIRO Springer Series 3,
https://doi.org/10.1007/978-3-030-34960-8_43

between the wireless mobile access and the fixed transport network and enhancing configuration flexibility through the concept of network slicing [7].

In this scenario, more and more often resource allocation and network control problems are encountered that present analogies with similar settings in computing systems and datacenters. Typically, given a set of general-purpose computing machinery, deployed by an Infrastructure Provider (InP) – or by the NSP itself over the networking infrastructure of the InP – they will host multiple tenants that act as NSPs for their (fixed or mobile) customers; the latter run applications on their User Equipment (UE) that may need computing resources that are partly local (on the very same UE) and partly residing in a datacentre or at the mobile edge (with the latter subject to possible latency constraints that may require resource reallocations to follow users on the move).

What we address in this paper is the modelling and control architecture of a fairly general problem in this framework, where multiple incoming flows with Quality of Service (QoS) constraints (typically, on latency) share the computing resources of multi-core network processors, which perform some specific functionality in the form of Virtual Network Functions (VNFs) in the NFV environment. By modelling the incoming traffic generated by each flow in the form of bursts of packets, we adopt a simple but general model for the queueing systems that represent packet-level processing. On top of this, we construct an optimization scheme to implement the assignment and load balancing of incoming flows, characterized by statistical models with much longer time scales than the packet traffic they generate, to the processing queues, over time periods within which they are served with constant rates. Finally, in a hierarchical organization, where an SDN controller may decide upon a reallocation of processing speeds, the possible reallocation of the latter over the next time period could be considered.

The paper is organized as follows. We formalize our general problem in the next Section, along with the description of the control architecture in the case of homogeneous traffic. The third Section contains a formulation suitable for heterogeneous flows with different requirements. We report some preliminary numerical results based on the model in the fourth Section and the conclusions in the fifth one.

2 Problem Statement and Homogeneous Flows Case

We consider a queueing system as depicted in Fig. 1. The queues represent the operations performed by Virtual Machines (VMs)[1] hosting VNFs that implement some specific functionality on packets generated by the flows (representing audio/video/data streams stemming from applications running on UEs). We do not

[1]We refer to VMs in the following, but the control architecture could be implemented with reference to containers, as well.

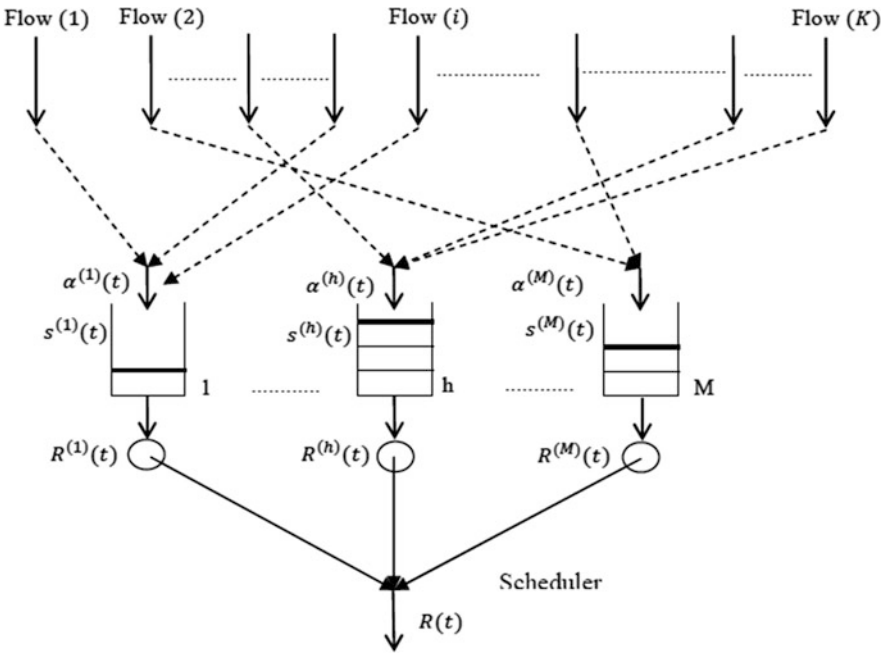


Fig. 1 Flow assignment problem

enter into any specific detail on the types of applications and network functions; our purpose here is to provide a fairly general model that could be applied to different situations by tuning the model's parameters, e.g., on the basis of available traffic traces. The service rates $R^{(1)}(t), \ldots, R^{(M)}(t)$, satisfying $\sum_{i=1}^{M} R^{(i)}(t) = R(t)$, represent the amount of processing capacity dedicated to the specific VM by assigning one or multiple cores on a multi-core network processor provided or hosted by the InP, with total processing capacity $R(t)$. Each VM realizes a specific VNF instance and, to fix ideas, we suppose them to be associated with a specific slice of a single tenant.

It is worth noting that the VMs may reside in the same physical processor, in different processors inside the same datacenter, or in different datacenters. For this reason, the assigned processing capacities may be different, and may correspond to different pricing schemes. The task of steering the traffic is performed by an SDN controller, to which the first packets of a flow are directed for classification when the flow is activated.

Assuming the processing capacities $R^{(1)}(t), \ldots, R^{(M)}(t)$ to have been fixed, we consider each queue with its own independent buffer in stationary conditions (and we drop the dependence on t in the following). Incoming flows are distributed among the processors on the basis of coefficients $\zeta^{(1)} > 0, \ldots, \zeta^{(M)} > 0, \sum_{i=1}^{M} \zeta^{(i)} = 1$ (to be determined through an optimization procedure that will be described later; for the time being, they are considered fixed), in the sense that each

incoming flow is assigned randomly to a processor upon its birth, according to the probability distribution determined by the coefficients.

We suppose the generation of flows to be such that each flow corresponds to a source, following a birth-death model. Packet bursts within each active flow are generated according to a Poisson model with Long-Range-Dependent (LRD) burst length. In order to take into account the traffic generation at the flow level (i.e., that the LRD traffic entering the queue is the aggregate of LRD traffic streams produced by individual flows), for each queue i we consider the average waiting time $W^{(i)}(a^{(i)})$, $a^{(i)} = \zeta^{(i)} m\lambda\beta$ calculated by means of an M$^{\mathrm{x}}$/G/1 [8] queueing model, when the aggregate burst rate is determined by the presence of m total active flows, each with burst generation rate equal to λ and average burst length β. Namely, from [8], we have

$$W^{(i)}\left(a^{(i)}\right) = \frac{{\rho^{(i)}}^2}{2\zeta^{(i)} m\lambda\beta\left(1+\sigma^2_{S^{(i)}}/\overline{S^{(i)^2}}\right)\left(1-\rho^{(i)}\right)} + \frac{\rho^{(i)}\left(\overline{X^2}/\beta - 1\right)}{2\zeta^{(i)} m\lambda\beta\left(1-\rho^{(i)}\right)} \tag{1}$$

where $S^{(i)}$ is the service time (depending on the distribution of the amount of operations to be performed per packet and on the processing speed $R^{(i)}$), with $E\{S^{(i)}\} = 1/\mu^{(i)}$ and mean square value and variance $\overline{S^{(i)^2}}$ and $\sigma^2_{S^{(i)}}$, respectively, $\rho^{(i)} = \zeta^{(i)} m\lambda\beta/\mu^{(i)}$ is the utilization, and $\overline{X^2}$ is the mean square value of the burst length. We note, in passing, that more general models could be also considered; for instance, if energy consumption is to be included as another Key Performance Indicator (KPI) to be traded off with latency, the M$^{\mathrm{x}}$/G/1/SET could be adopted to account for set up times for processor wakeup (as done in [9, 10] in the case of deterministic service times).

Note that, for the time being, we suppose the cluster of VMs under consideration to be dedicated to serve a single class of traffic, characterized by equal generation parameters. We will extend the model to multiple classes in the next section.

As the time scales at the burst- and flow-level are widely different, it makes sense to consider that variations in the number of flows occur on a much longer time scale with respect to that of events in the Markov chain describing the dynamics of packets in the queue. Based on this consideration, we can ignore non-stationary behaviours, and assume that a stationary state in the queue probabilities is reached almost instantaneously between birth and death events at the flow level (a precise treatment of a somehow related problem can be found in [11]).

Under the above flow distribution strategy and the assumption of homogeneous flows, the same burst generation model holds for the flows being assigned to each processor. Therefore, we can examine each queue in isolation, conditioned to the presence of m total flows in the system, as an M$^{\mathrm{x}}$/G/1 queue with input rate $\zeta^{(i)} m\lambda\beta$ [pkts/s], $i = 1, \ldots, M$. As mentioned above, the situation of flows with unequal burst generation rates (or diverse QoS requirements) will be outlined further on; however, we can already note that the more general case can be handled in a similar way if

service separation with static partitions [12] is applied, i.e., services giving rise to flows with similar service rates and QoS requirements are grouped into classes and assigned to a subset of processors for each class.

In order to avoid instability, the following condition must be satisfied for each queue:

$$\rho^{(i)} = \zeta^{(i)} m \lambda \beta / \mu^{(i)} < 1, \text{i.e.} \, m^{(i)} \equiv m\zeta^{(i)} < \frac{\mu^{(i)}}{\lambda\beta} \tag{2}$$

so that the maximum number of flows $m_{\max}^{(i)}$ acceptable by queue i is equal to $\lfloor \mu^{(i)}/\lambda\beta \rfloor$.

This also imposes the presence of a Call Admission Control (CAC) on the system, such that the maximum number of flows totally acceptable be limited to

$$m_{\max} = \sum_{i=1}^{M} \left\lfloor \frac{\mu^{(i)}}{\lambda\beta} \right\rfloor \tag{3}$$

At this point, we can average out the delay over the distribution of the flows. To this aim, we suppose that both interarrival times and durations of flows can be described by independent exponential distributions, with parameters λ_f and μ_f, respectively. Let $A_f = \lambda_f/\mu_f$ [Erlangs] denote the traffic intensity of the flows. Then, the probability $p_k^{(i)}$ that k flows are active (producing bursts) on the i-th processor's queue is given by

$$p_k^{(i)} = \Pr\left\{m^{(i)} = k\right\} = p_0^{(i)} \prod_{j=0}^{k-1} \frac{\left(\zeta^{(i)} A_f\right)^j}{j!} = \frac{\left(\zeta^{(i)} A_f\right)^k / k!}{\sum_{j=0}^{m_{\max}^{(i)}} \frac{(\zeta^{(i)} A_f)^j}{j!}} \tag{4}$$

$$k = 0, 1, \ldots, m_{\max}^{(i)}$$

Thus, we can write

$$\overline{W}^{(i)} = \frac{1}{\left(1 - p_0^{(i)}\right)} \sum_{k=1}^{m_{\max}^{(i)}} p_k^{(i)} W^{(i)} \left(\zeta^{(i)} k \lambda \beta\right) \tag{5}$$

for the average (with respect to the total number of flows) delay per queue (considering the presence of at least one active flow at the i-th VM) and

$$\overline{W} = \sum_{i=1}^{M} \overline{W}^{(i)} \zeta^{(i)} \tag{6}$$

for the total average delay over all flows. The upper limit of the sum in (5) is necessary as a consequence of condition (2).

At this point, an optimization problem can be posed for the selection of the traffic spreading coefficients as

$$\begin{array}{c} \min \quad \overline{W} \\ \zeta^{(1)} \geq 0, \ldots, \zeta^{(M)} \geq 0 \\ \sum_{i=1}^{M} \zeta^{(i)} = 1 \end{array} \tag{7}$$

3 Heterogeneous Flows with Different Requirements

Averaging with respect to the incoming flows might be useful also in the presence of traffic with different statistical characteristics. The flow model would then correspond, in general, to a stochastic knapsack [12]. As noted, in this case the most advisable and manageable model is that of service separation, whereby only flows with the same statistical characteristics are multiplexed together and feed the same buffer with their bursts.

To fix ideas, let us suppose to have K such classes. Then, the overall processing capacity resource pool of R units can be partitioned into K groups, with R_k units assigned to the k-th group, $k = 1, \ldots, K$, according to some criterion. In particular, let $\theta^{(k)}(m^{(k)})$ be a function that represents the minimum processing capacity that is required to satisfy packet-level QoS requirements for $m^{(k)}$ permanent class-k flows multiplexed in a buffer. In principle, there are two possible ways to do the assignment, which we report from [12].

- *Service Separation with Static Partitions (SSSP)*: Let $R_1, \ldots, R_K$, with $R_1 + \ldots + R_K = R$, be a partition of the capacity. Under SSSP, an arriving class-k flow is admitted iff

$$\theta^{(k)}\left(m^{(k)} + 1\right) \leq R_k \tag{8}$$

  with $\theta^{(k)}(\cdot)$ corresponding, for instance, to the criteria defined by (2) or to the constraint of not exceeding a maximum average delay for the class.

- *Dynamic Partitions (DP)*. The processing capacity fractions assigned to classes are now given by $\theta^{(1)}(m^{(1)}), \ldots, \theta^{(K)}(m^{(K)})$, so that they are changing, but on a much longer time scale with respect to the packet-level dynamics. A new class-k

flow would be admitted iff

$$\theta^{(k)}\left(m^{(k)}+1\right)+\sum_{\substack{j=1\\ j\neq k}}^{K}\theta^{(j)}\left(m^{(j)}\right)\leq R \tag{9}$$

In any case, it is interesting to note that the availability of analytical packet-level models makes relatively easy here to define a *packet level* criterion, and naturally lends a notion of capacity of the underlying statistical multiplexer (namely, the stability preserving bound on the utilization, or the delay bound), which allows a clear definition of the flow state space.

Given the presence of a CAC, there is actually another performance index that might become of interest; namely, the blocking probability of flows (Grade of Service, GoS). The blocking probabilities at individual queues are easily calculated in the SSSP case, as done in the preceding section: the queuing model outlined above for the flow level would indeed be of type M/M/ $m^{(k)}_{\max}(R_k)$ / $m^{(k)}_{\max}(R_k)$, $m^{(k)}_{\max}(R_k)$ being the maximum number of acceptable flows as a function of R_k, so that the blocking probabilities just correspond to the Erlang B formula, i.e.,

$$P_B^{(k)}=EB\left[\rho_f^{(k)},m^{(k)}_{\max}(R_k)\right]=\frac{\left(\rho_f^{(k)}\right)^{m^{(k)}_{\max}(R_k)}/m^{(k)}_{\max}(R_k)!}{\sum_{j=0}^{m^{(k)}_{\max}(R_k)}\frac{\left(\rho_f^{(k)}\right)^{j}}{j!}} \tag{10}$$

On the other hand, in the DP case the blocking probabilities should be derived by the general stationary distribution of a stochastic knapsack.

In both situations, a general criterion could be minimizing an overall index of the type $\overline{P}_B=\sum_{k=1}^{K}P_B^{(k)}$, or $P_B^{\max}=\max_{k=1,\ldots,K}P_B^{(k)}$, with respect to the number of active processors and their allocation among classes, under given low-level constraints on delay (and, possibly, on power consumption, if we want to add this KPI to the optimization, by suitably changing the queueing models).

4 Numerical Results

We consider an example with respect to the case of a single traffic class (homogeneous flows). To get an idea of the objective function, we plot it in the case $M=2$, as a function of $\zeta^{(1)}$, $\zeta^{(2)}$, with the following numerical values of the parameters: $A_f=10$, $\lambda=20$ [burst/s], $\beta=1.5$ [pkts/burst], $\overline{X^2}=3$ (we have assumed a continuous approximation of the burst length, with a Pareto distribution

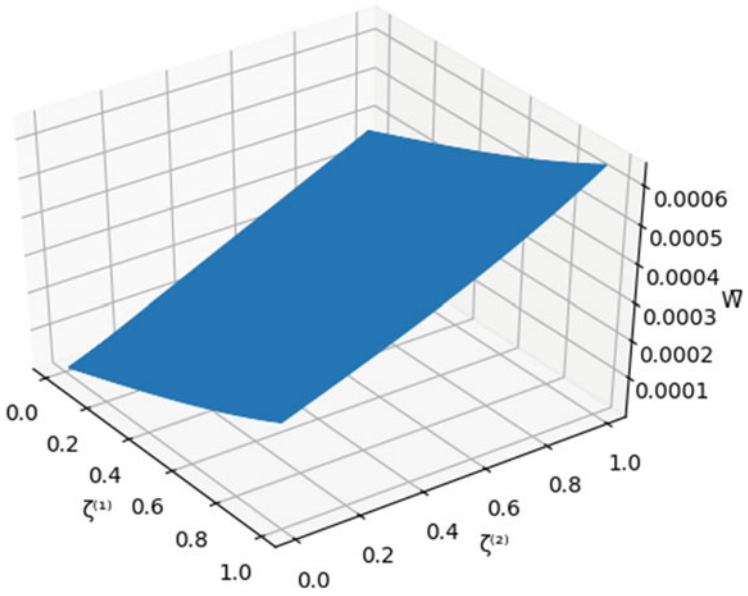


Fig. 2 Plot of the unconstrained objective function in the case $M = 2$

with location parameter $\delta = 1$ and shape parameter $\alpha = 3$), $R^{(1)} = 2{,}100{,}000$, $R^{(2)} = 1{,}600{,}000$ [operations/s], average number of operations per packet 1000 (whence $1/\mu^{(1)} \cong 476$ μs, $1/\mu^{(2)} = 625$ μs), $\overline{S^{(1)^2}} \cong 229,408 \cdot 10^{-12}$, $\overline{S^{(2)^2}} \cong 395,507 \cdot 10^{-12}$ (also here we have assumed a Pareto distribution of the service time, with shape parameter $\alpha = 10$ in both cases and location parameters $\delta^{(1)} \cong 428$ μs and $\delta^{(2)} \cong 562$ μs, respectively), $\sigma^2_{S^{(i)}} = \overline{S^{(i)^2}} - 1/\mu^{(i)^2}, i = 1, 2$. The plots of the objective function are shown in Figs. 2 and 3, for the unconstrained case and over the plane $\zeta^{(1)} + \zeta^{(2)} = 1$, respectively. Figure 4 reports the result of the optimization procedure in this simple case, with the minimum value at $\zeta^{(1)} = 0.836$ corresponding to 271.5 μs. We have used a standard optimization tool available in the Python library (www.scipy.org), with optimization method SLSQP (Sequential Least SQuares Programming). However, it is worth noting that the form of the objective function, which is separable in the optimization variables, may suggest the use of Dynamic Programming. The possible advantages in its application will the subject of further investigation.

Considering now the case $M = 3$, we perform the optimization for a set of different values of the load generated per flow, by varying the burst arrival rate λ in the range [10, 200] with discrete steps of 10 bursts/s. In this case, we have kept all the previous values, and set $R^{(3)} = 1{,}200{,}000$ [operations/s] ($1/\mu^{(3)} \cong 833\ \mu s, \delta^{(3)} \cong 750$ μs, $\overline{S^{(3)^2}} \cong 703,125 \cdot 10^{-12}$). The results are reported in Fig. 5, showing the tendency to a relatively stable distribution of the flows according to the processing capacities for increasing load.

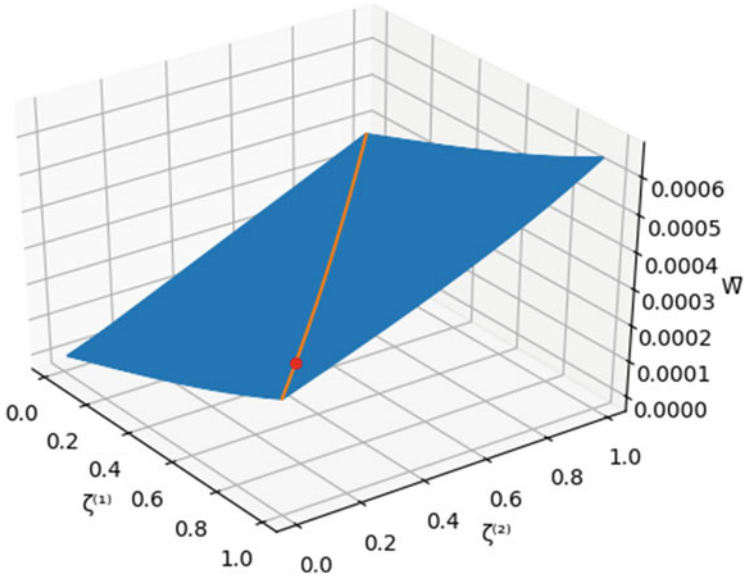


Fig. 3 Plot of the constrained objective function in the case $M = 2$

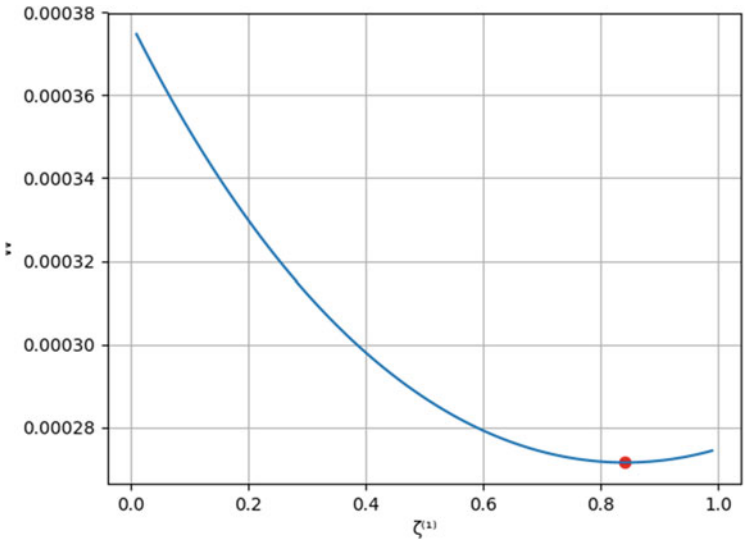


Fig. 4 Constrained cost function against $\zeta^{(1)}$ in the case $M = 2$

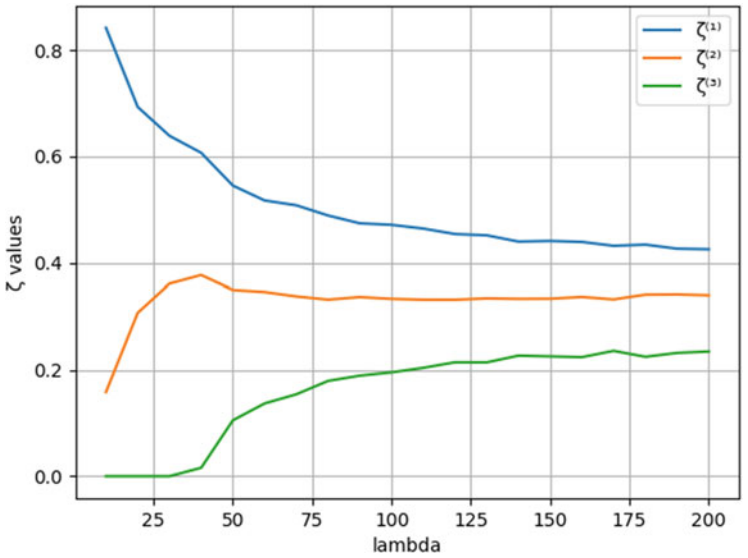


Fig. 5 Plot of the optimal allocations against the average load per flow λ [bursts/s]

5 Conclusions

We have considered an optimization problem in the context of multi-core network processors that provide a set of VNFs performing network operations (which may be related to network switching or MEC functionalities) on the packets generated by multiple incoming flows. The latter may be homogeneous or heterogeneous in the traffic parameters or in their requirements in terms of delay or loss. We have defined two possible optimization schemes in the two cases. Numerical results have been reported in the case of homogeneous flows. Further work will consider the numerical implementation in both cases and comparison with other assignment methods.

Acknowledgments This work was partially supported by the European commission, under the H2020 5G PPP project MATILDA (contract no. 761898).

References

1. Kreutz, D., Ramos, F.M.V., Veríssimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. Proc. IEEE. **103**(1), 14–76 (2015)
2. Mijumbi, R., Serrat, J., Gorricho, J.-L., Bouten, N., De Turck, F., Boutaba, R.: Network function virtualization: state-of-the-art and research challenges. IEEE Commun. Surv. Tutorials. **18**(1), 236–262 (2016)
3. Manzalini, A., et al.: Towards 5G software-defined ecosystems – technical challenges, business sustainability and policy issues. IEEE SDN initiative whitepaper (2016). http://resourcecenter.fd.ieee.org/fd/product/whitepapers/FDSDNWP0002

4. ETSI GS MEC 002 2016: Mobile Edge Computing (MEC); Technical requirements. http://www.etsi.org/deliver/etsi_gs/MEC/001_099/002/01.01.01_60/gs_MEC002v010101p.pdf
5. MEC deployments in 4G and evolution towards 5G. ETSI white paper (2018)
6. The 3GPP Association: System architecture for the 5G system. 3GPP technical specification (TS) 23.501, Stage 2, Release 16, version 16.0.2 (2019)
7. Ordóñez-Lucena, J., et al.: Network slicing for 5G with SDN/NFV: concepts, architectures, and challenges. IEEE Commun. Mag. **55**(5), 80–87 (2017)
8. Tijms, H.C.: A first course in stochastic models. Wiley, Chichester (2003)
9. Bolla, R., Bruschi, R., Carrega, A., Davoli, F.: Green networking with packet processing engines: modeling and optimization. IEEE/ACM Trans. Network. **22**(1), 110–123 (2014)
10. Bolla, R., Bruschi, R., Carrega, A., Davoli, F., Pajo, J.F.: Corrections to: "green networking with packet processing engines: modeling and optimization". IEEE/ACM Trans. Network. (2017). https://doi.org/10.1109/TNET.2017.2761892
11. Ghani, S., Schwartz, M.: A decomposition approximation for the analysis of voice/data integration. IEEE Trans. Commun. **43**(7), 2441–2452 (1994)
12. Ross, K.W.: Multiservice loss models for broadband telecommunication networks. Springer, Secaucus (1995)