Miguel Mujica Mota
Idalia Flores De La Mota
Daniel Guimarans Serrano *Editors*

Applied Simulation and Optimization

In Logistics, Industrial and Aeronautical Practice

Springer

Applied Simulation and Optimization

Miguel Mujica Mota · Idalia Flores De La Mota
Daniel Guimarans Serrano
Editors

Applied Simulation and Optimization

In Logistics, Industrial and Aeronautical Practice

Springer

Editors
Miguel Mujica Mota
Aviation Academy
Amsterdam University of Applied Sciences
Amsterdam, Noord-Holland
The Netherlands

Daniel Guimarans Serrano
Optimisation Research Group, NICTA
Sydney, NSW
Australia

Idalia Flores De La Mota
Facultad de Ingeniería
Universidad Nacional Autónoma de México
Mexico City
Mexico

ISBN 978-3-319-15032-1 ISBN 978-3-319-15033-8 (eBook)
DOI 10.1007/978-3-319-15033-8

Library of Congress Control Number: 2015934043

Springer Cham Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

To my parents and Christina

Miguel Mujica Mota

Foreword

There was a time when problem solving relied on analytical models purely grounded in mathematical optimisation. As problems became more complex, analytical models reached their limits due to the lack of mathematical formulation or intractable solutions. Benefiting from the exponential increase in computer power, numerical models progressively took over to estimate near-optimal solutions, especially for problems that could be distributed across a topological mesh. This first generation of numerical models achieved significant outcomes in chemistry, physics, soil science and climate research. But it is only when network theory, queuing models and microsimulation reached maturity that operational research was able to address complex issues involving time and space-dependent decision-making processes applied to supply chain or logistics management.

As modern supply chains become more sophisticated and more intimately intertwined with logistical constraints, performance optimisation is increasingly concerned with the stochastic and dynamic nature of the relationship between offer and demand. Likewise, optimal solutions often need to accommodate multi-objective, and sometimes conflicting, decisions. In other words, we have moved into a world of complex problem solving characterised by ill-defined predicates that necessitate constant probing of the modelled environment in order to evaluate the robustness of the optimum solution. Simulation-optimisation techniques aim at creating this dyadic and dynamic relationship between optimisation and evaluation.

This book comes at its own time as it provides a broad and rich set of examples of simulation-optimization techniques. From a methodological perspective, coupled optimisation procedures include queuing models, tree search or evolutionary algorithms and linear programming. Domains of application include aviation, land

transportation, health management, communication and manufacturing industries. Each chapter provides a clear justification of the proposed method and comprehensive evidence of its performance. The overall result is a rich but coherent picture of state-of-the-art applications of simulation-optimisation techniques.

Prof. Pascal Perez
Research Director
SMART Infrastructure Facility
Faculty of Engineering and Information Sciences
University of Wollongong, Wollongong
NSW, Australia

Acknowledgments

The editors would like to thank the anonymous reviewers of the book; it has greatly benefited from their very valued comments and suggestions.

We would also like to thank the following institutions which supported us during the development process of the book:

- The Aviation Academy of the Amsterdam University of Applied Sciences.
- The National Autonomous University of Mexico for supporting the research through the project DGAPA-PAPIIT IN-116012.
- The National Council of Science and Technology of Mexico (CONACYT).
- NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program. NICTA is also funded and supported by the Australian Capital Territory, the New South Wales, Queensland and Victorian Governments, the Australian National University, the University of New South Wales, the University of Melbourne, the University of Queensland, the University of Sydney, Griffith University, Queensland University of Technology, Monash University and other university partners.

Contents

Contributors

Michael Affenzeller Heuristic and Evolutionary Algorithms Laboratory, School of Informatics, Communications and Media, University of Applied Sciences Upper Austria, Research Center Hagenberg, Hagenberg, Austria; Institute for Formal Models and Verification, Johannes Kepler University Linz, Linz, Austria

Assil Al-Ibrahim Department of Economics and Business, University of Amsterdam, Amsterdam, The Netherlands

Catya Zuniga Alcaraz Logistics & Supply Chain Management Department, Universidad Popular Autonoma del Estado de Puebla, Puebla, Puebla, Mexico

Pol Arias Smart Logistics and Production Group, Internet Interdisciplinary Institute (IN3-UOC), Barcelona, Spain

Andreas Beham Heuristic and Evolutionary Algorithms Laboratory, School of Informatics, Communications and Media, University of Applied Sciences Upper Austria, Research Center Hagenberg, Hagenberg, Austria; Institute for Formal Models and Verification, Johannes Kepler University Linz, Linz, Austria

Vitaly Bolshakov Riga Technical University, Riga, Latvia

Idalia Flores De La Mota Facultad de Ingeniería, Universidad Nacional Autónoma de México, Mexico City, Mexico

Ranjit Kumar Ettam Xerox Corporation, Webster, NY, USA

Eric Gross Xerox Corporation, Webster, NY, USA

Daniel Guimarans Optimisation Research Group, National ICT Australia (NICTA), Sydney, Australia

René Haijema Department of Economics and Business, University of Amsterdam, Amsterdam, The Netherlands; Operations Research and Logistics Group, Wageningen University, Wageningen, The Netherlands

Leonardo G. Hernández-Landa Graduate Program in Systems Engineering, Universidad Autónoma de Nuevo León (UANL), San Nicolás, Mexico

Stephan Hutterer Heuristic and Evolutionary Algorithms Laboratory, School of Informatics, Communications and Media, University of Applied Sciences Upper Austria, Research Center Hagenberg, Hagenberg, Austria; Institute for Formal Models and Verification, Johannes Kepler University Linz, Linz, Austria

Monika Kofler Heuristic and Evolutionary Algorithms Laboratory, School of Informatics, Communications and Media, University of Applied Sciences Upper Austria, Research Center Hagenberg, Hagenberg, Austria

Michael Kommenda Heuristic and Evolutionary Algorithms Laboratory, School of Informatics, Communications and Media, University of Applied Sciences Upper Austria, Research Center Hagenberg, Hagenberg, Austria

Nikky Kortbeek Department of Economics and Business, University of Amsterdam, Amsterdam, The Netherlands; University of Twente, Enschede, The Netherlands

Gabriel Kronberger Heuristic and Evolutionary Algorithms Laboratory, School of Informatics, Communications and Media, University of Applied Sciences Upper Austria, Research Center Hagenberg, Hagenberg, Austria

Carlos Henrique Mariano Department of Electrical Engineering—DAELT, Federal Technological University of Paraná—UTFPR, Curitiba, PR, Brazil

Galina Merkuryeva Riga Technical University, Riga, Latvia

Miguel L. Morales-Marroquín Graduate Program in Systems Engineering, Universidad Autónoma de Nuevo León (UANL), San Nicolás, Mexico

Miguel Mujica Mota Aviation Academy, Amsterdam University of Applied Sciences, Amsterdam, The Netherlands

Romeo Sánchez Nigenda Graduate Program in Systems Engineering, Universidad Autónoma de Nuevo León (UANL), San Nicolás, Mexico

Carlo Alessandro Zanetti Pece Department of Electrical Engineering—DAELT —Postgraduate Program in Biomedical Engineering, Federal Technological University of Paraná—UTFPR, Curitiba, PR, Brazil

João Pedro Pedroso INESC TEC and DCC-FCUP, Porto, Portugal

Pascal Perez Faculty of Engineering and Information Sciences, University of Wollongong, Wollongong, NSW, Australia

Erik Pitzer Heuristic and Evolutionary Algorithms Laboratory, School of Informatics, Communications and Media, University of Applied Sciences Upper Austria, Research Center Hagenberg, Hagenberg, Austria

Sudhendu Rai Xerox Corporation, Webster, NY, USA

Rui Rei INESC TEC and DCC-FCUP, Porto, Portugal

Yasmín Á. Ríos-Solís Graduate Program in Systems Engineering, Universidad Autónoma de Nuevo León (UANL), San Nicolás, Mexico

Esther Segura Pérez Instituto de Ingeniería, Universidad Nacional Autónoma de México, Mexico City, Mexico

Nico M. van Dijk Department of Economics and Business, University of Amsterdam, Amsterdam, The Netherlands; University of Twente, Enschede, The Netherlands

Erik van der Sluis Department of Economics and Business, University of Amsterdam, Amsterdam, The Netherlands

Jan van der Wal Department of Economics and Business, University of Amsterdam, Amsterdam, The Netherlands

Alexander Vindel Garduño Facultad de Ingeniería, Universidad Nacional Autónoma de México, Mexico City, Mexico

Stefan Vonolfen Heuristic and Evolutionary Algorithms Laboratory, School of Informatics, Communications and Media, University of Applied Sciences Upper Austria, Research Center Hagenberg, Hagenberg, Austria; Institute for Formal Models and Verification, Johannes Kepler University Linz, Linz, Austria

Stefan Wagner Heuristic and Evolutionary Algorithms Laboratory, School of Informatics, Communications and Media, University of Applied Sciences Upper Austria, Research Center Hagenberg, Hagenberg, Austria

Stephan M. Winkler Heuristic and Evolutionary Algorithms Laboratory, School of Informatics, Communications and Media, University of Applied Sciences Upper Austria, Research Center Hagenberg, Hagenberg, Austria

Part I
Tools and Techniques Using SimOpt

Simulation-Based Optimization with HeuristicLab: Practical Guidelines and Real-World Applications

Michael Affenzeller, Andreas Beham, Stefan Vonolfen, Erik Pitzer, Stephan M. Winkler, Stephan Hutterer, Michael Kommenda, Monika Kofler, Gabriel Kronberger and Stefan Wagner

Abstract Dynamic and stochastic problem environments are often difficult to model using standard problem formulations and algorithms. One way to model and then solve them is simulation-based optimization: Simulations are integrated into the optimization process in order to evaluate the quality of solution candidates and to identify optimized system configurations. Potential solutions are evaluated with a simulation model, which leads to new challenges regarding runtime performance, robustness, and distributed evaluation. In order to design, compare, and parameterize algorithmic approaches it is beneficial to use an optimization framework for algorithm design and evaluation. On the one hand, this chapter shows how arbitrary simulators can be coupled with the open-source HeuristicLab optimization framework. This coupling is implemented in a generic way so that the simulators act as external evaluators. On the other hand, we demonstrate how arbitrary optimizers available within HeuristicLab can be called from a simulator in order to perform complex optimization tasks within the simulation model. In order to illustrate the applicability of these approaches, real-world examples investigated by the authors are discussed. We show here application examples from different fields, namely logistics network design, vendor managed inventory routing, steel slab logistics, production optimization with dispatching rule scheduling, material flow simulation, and layout optimization.

M. Affenzeller (✉) · A. Beham · S. Vonolfen · E. Pitzer · S.M. Winkler · S. Hutterer · M. Kommenda · M. Kofler · G. Kronberger · S. Wagner
Heuristic and Evolutionary Algorithms Laboratory, School of Informatics, Communications and Media, University of Applied Sciences Upper Austria, Research Center Hagenberg, Softwarepark 11, 4232 Hagenberg, Austria
e-mail: michael.affenzeller@fh-hagenberg.at

M. Affenzeller · A. Beham · S. Vonolfen · S. Hutterer · M. Kofler
Institute for Formal Models and Verification, Johannes Kepler University Linz, Altenberger Straße 69, 4040 Linz, Austria

M. Mujica Mota et al. (eds.), *Applied Simulation and Optimization*,
DOI 10.1007/978-3-319-15033-8_1

1 Introduction

The field of simulation optimization [11, 16, 17, 20] is still a rather young flavor in operations research. One of its key enablers is the efficient utilization of parallel computing infrastructures which allows modeling and optimizing not only toy problems, but also more complex real-world scenarios from the field of production and logistics as well as other domains. So far, metaheuristic optimization approaches have been applied successfully in solving combinatorial optimization tasks such as vehicle routing, production scheduling, and layout optimization. However, when using standardized problem formulations, only singular aspects of the real-world can be modeled and optimized in a quite restrictive way—which is often not capable to represent the real-world and its complex interrelations and constraints appropriately.

Discrete event simulation approaches allow modeling complex and interrelated production and logistic scenarios in a more sophisticated and realistic way. However, the optimization capabilities of recent discrete simulation packages [17] are still quite limited and rather aimed to offer robust "broadband" optimizers which are not capable to explore the full optimization potential of concrete scenarios.

The approach presented in this chapter aims to couple a powerful meta-heuristic optimization framework offering a huge variety of optimization algorithms with diverse simulators acting as evaluators of solution candidates in a generic way. By this means, the user shall be enabled to choose and parameterize an appropriate optimization method in order to explore more optimization potential compared to when using built-in solvers (when available). Pursuing this approach we adhere to the *no free lunch theorem of optimization* [57] which postulates that a general purpose, universal optimization strategy cannot be implemented and that the only possibility for a strategy to outperform another one is to be more specialized to the structure of the tackled problem.

The generic approach described in this chapter is to couple diverse specific simulation models representing real-world scenarios with the open-source heuristic optimization framework HeuristicLab [53]. Google protocol buffers act as a generic interface between optimization algorithms and concrete simulation models that here act as an external evaluator.

From an algorithmic point of view, the main challenges of the proposed approach are algorithm selection and parameterization, runtime consumption, robustness, and stability of calculated solutions. One of the major issues in this context is the runtime consumption aspect: When solving combinatorial optimization problems, the evaluation of a solution candidate usually only takes small fractions of seconds, whereas in the context of simulation-based optimization the evaluation of a solution candidate might take several seconds or even minutes. For the combination of optimization and simulation it is necessary to scale back the complexity of both simulation and optimization to obtain good results in reasonable time. The requirement of this balance has led to a renaissance of optimization methods that require fewer evaluations such as evolution strategies [40] and simulated annealing [27].

Stochastic elements of the simulation model create the need for robustness analysis of solution candidates. This requires multiple evaluations and enhanced multi-objective fitness functions that also take into account robustness and stability of solution candidates. These additional requirements motivate on the one hand the use of massively parallel computing infrastructures for solution evaluation, and on the other hand the application of enhanced techniques for algorithm selection and parameterization in order to choose the appropriate degree of greediness of the algorithm, which is the basis for exploiting the achievable optimization potential.

This chapter is structured in the following way: Sect. 2 describes the technical basis and implementation of the coupling between simulation models and algorithms offered by HeuristicLab. Thus, for the reader this section also presents guidelines on how to couple simulation models with optimization frameworks. Section 3 summarizes several concrete application example scenarios, in which simulation-based optimization has been used in combination with HeuristicLab for solving real-world problems. Finally, Sect. 4 summarizes this chapter and points out the future research topics and challenges in the field of simulation-based optimization.

2 Methodology and Approach

Over the last decade, a great deal of research has been devoted to couple simulation models with optimization. In order to meet the increasing demand for optimization of simulation model parameters, commercial simulation software packages frequently offer integrated optimization, for example, OptQuest® [41] or the WITNESS® Optimizer [39].

These commercial software solutions frequently apply metaheuristic algorithms for optimization, for instance, genetic algorithms, evolution strategies, tabu search, simulated annealing, scatter search, or hill-climbers. Optimization interfaces usually show only a limited set of tunable algorithm parameters in order to simplify the user interface. These interfaces often do not expose characteristics of the optimization run, e.g., convergence behavior, but focus on statistical analysis of optimization results. This black box approach is certainly favorable with respect to robustness and usability of an embedded optimization tool, but it also limits the potential of the applied optimization method. Optimization environments on the other hand provide a more complex user interface, more algorithms, and allow analyzing the algorithm behavior in more detail in order to improve results or convergence speed.

But, parameter optimization is only one possibility for simulation-based optimization. As far as an embedded optimization approach is concerned commercial packages are not in widespread use. Simulation models that encounter decision problems which would be suited for optimization will have to include their own algorithms or link optimization frameworks into the model. We aim to describe in this section common interaction patterns between simulation and optimization, the HeuristicLab software architecture that is suitable for this kind of optimization, and the interfaces that mate simulation and optimization.

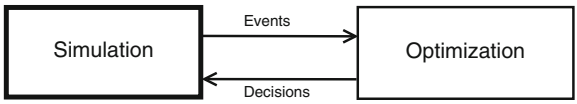


Fig. 1 Interaction pattern for control optimization. The simulation model (here shown *bold*) is the initiating part

2.1 Interaction Patterns Between Simulation and Optimization

Generally, two main interactions patterns, control optimization and parametric optimization can be identified [20]:

- **Control Optimization**: The optimization problem might arise within the simulation model, a decision has to be made given the state of the model.
- **Parametric Optimization**: The simulation model might act as a fitness function, which will take a number of parameters and calculate the resulting fitness value.

The main difference between these patterns is the role of the initiating part that steers the control flow. In this section we will also describe a third pattern which can be seen as a combination of the other two. The application of HeuristicLab has been successful for these patterns [38, 49] of which real-world examples will be given in Sect. 3.

Control Optimization

Control optimization is schematically depicted in Fig. 1. The optimization here is concerned with decision making in changing and uncertain environments. The simulated problem scenario changes over time as new events emerge and previous decisions get executed. It is not possible to undo or change decisions that have been implemented already. Sometimes, a rolling time horizon is allowed to plan ahead, in other cases only the actual situation may be taken into account for making decisions. This category can thus also be described as *online optimization*.

The simulation can be seen as a placeholder for a real-world environment and models the dynamics of the real system. The optimization or decision-making procedure has to react to these dynamics. Eventually, events from the simulation environment can be replaced with events from the real system. The simulation is mainly used in place of the real system to test and validate new optimization approaches. Experiments in the real system would be too costly and too slow to evaluate.

Parametric Optimization

Parametric optimization is illustrated in Fig. 2. A candidate solution (usually a parameter vector) is passed to the simulation which then returns a quality value by running the simulation model using the given parameter set. This approach can be applied when, for example, a closed-form representation of the evaluation function is not feasible because it contains complex stochastic elements or dynamic interactions. Especially, the application of metaheuristics has proven fruitful in this context [45].

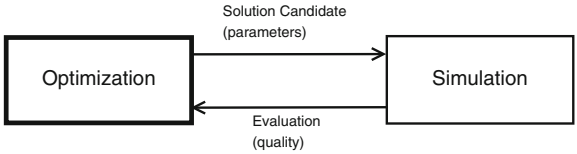


Fig. 2 Simulation-based parametric optimization. The optimization model is the initiating part (here shown *bold*)

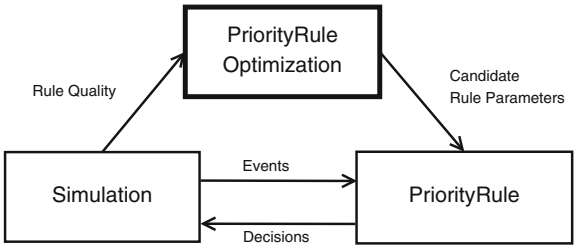


Fig. 3 Generation of priority rules combining parameter and control optimization. The control flow is steered by the priority rule optimization (here shown *bold*)

The candidate solutions are iteratively improved by the metaheuristic algorithm, often totalling a large number of simulation runs. This category can also be described as *offline optimization*. For stochastic simulation models the quality has to be seen as a random variable and often multiple simulation runs of the same parameter set need to be performed to estimate the expected quality.

Generation and Parameterization of Heuristic Policies

This approach combines parametric and control optimization. Policies can be seen as a control strategy that can be used for making decisions online. The simulation model uses such a policy according to the control optimization pattern. When events require a decision, the policy is called to make that decision. The policies themselves are improved offline using the quality that is returned by the simulation model. Policies should be designed such that they can efficiently compute the next decision based on the current situation in dynamic environments. Examples of such policies are priority rules that combine and weight different domain features to rank and prioritize a number of alternative decisions. Different representations for priority rules are possible, for example, vector and tree representations. Figure 3 illustrates this pattern schematically.

2.2 Software Architecture

HeuristicLab[1] [53] is a software environment for heuristic and evolutionary algorithms, developed and successively applied by members of the *Heuristic and Evolutionary Algorithms Laboratory*[2] since 2002. Being licensed under the GNU GPL,[3] HeuristicLab has a growing community of researchers and practitioners using the tool in a wide range of scientific as well as commercial areas. It provides a vast number of already implemented algorithms and problems for optimization and data analysis, an experiment designer, and support for algorithm and results analysis. Furthermore, a sophisticated graphical user interface distinguishes HeuristicLab from other heuristic optimization frameworks [35], which usually require substantial programming skills to extend algorithms and apply them to a given problem. HeuristicLab offers not only programming-based extensibility, but also allows to add and modify algorithms and problems using the graphical user interface and a graphical algorithm modeling language. In HeuristicLab, algorithms are described as operator graphs where an operator represents a node and the connections denote the execution flow. Changing or rearranging operators can be done by drag-and-drop without actually writing code [54]. The framework thereby enables users and practitioners to perform complex tasks such as algorithm development. The possibility to extend the framework on the code level remains, and software engineers benefit from the plugin-based architecture (Fig. 4) allowing them to develop custom algorithms, data structures for solution representations, or custom optimization problems. This has led to a significant level of code reuse across metaheuristic variants and gradually gives users an understanding of algorithm development [52, 55].

Base and Core Layer

The base layer contains plugins that provide essential functionality required by all other plugins of the above layers. Every plugin in HeuristicLab is based on the *PluginInfrastructure* which loads plugins and checks their dependencies. The base layer also includes the *Persistence* which allows to save and load files.

The core layer is situated atop the base layer and includes the algorithm modeling language. It contains core interfaces, data objects, parameters, operators, and engines. The algorithm modeling language uses operators to describe small, individual steps in an algorithm [52]. Algorithms are created by chaining together these operators. This is called an operator graph and *engines* are used to execute that graph by applying one operator after another sequentially. In general, the operators process data that is stored in the memory of an algorithm which is represented in form of a scope tree. Each scope can hold several variables, such as a quality value, the current iteration of an algorithm, or a complex data type like a solution candidate. If it contains subscopes, it can also represent a population. Operators can be applied on any level in the scope tree and may modify its structure as well as read and write variables. Some

[1] http://dev.heuristiclab.com.

[2] http://heal.heuristiclab.com.

[3] https://www.gnu.org/copyleft/gpl.html.

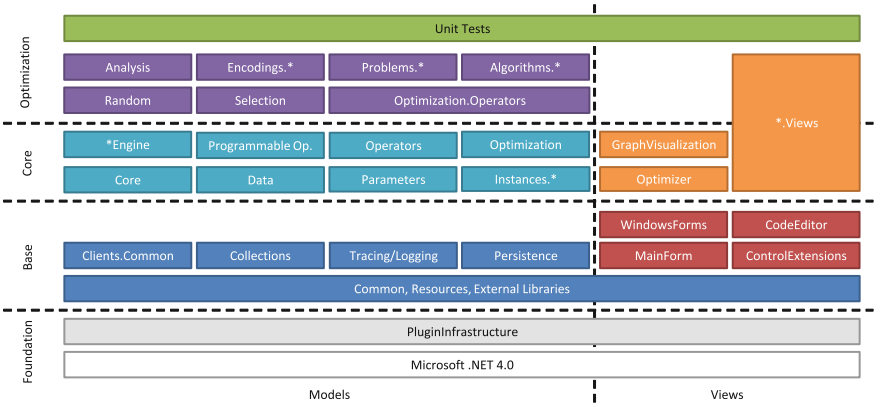


Fig. 4 Block diagram of the architecture of HeuristicLab with a separation into different logical layers. HeuristicLab is composed of a number of plugins each containing a defined set of functionality. Generally, each of the boxes in this figure represent a plugin, or multiple plugins if a "*" is added to the name. The *horizontal layers* are *base*, *core*, and *optimization*. The *vertical layers* show the separation between *models* and *views*. In the block diagram, a plugin always depends on the plugins on the layer below as well as the plugins to its *left* on the same layer

operators allow the application of subsequent operators on a number of sub-scopes in parallel. The *ParallelEngine* is able to execute those operators in different threads and further engines exist that allow to make use of distributed computing resources, such as HeuristicLab Hive [53]. Therefore, HeuristicLab provides an easy way to incorporate data-based parallelism into algorithms. To obtain the values of variables, operators specify *parameters* which can either directly contain a value or provide only the name of a variable which is used to find a match among the algorithm's parameters, the problem's parameters, and the scope tree. An example of such an operator is an evaluator that is applied on a solution scope. Evaluators typically read those variables that contain the solution encoding, those that provide the problem data and, after computing the fitness, add a quality variable to the scope. Similar to any other operator, it could also contain additional parameters that would, e.g., read the algorithm's state such as the current iteration or a collection which acts as another memory.

Optimization Layer

The topmost layer in the architecture includes the algorithms, problems, different standard encodings, and various other plugins for algorithm analysis and random number generation. HeuristicLab is shipped with several algorithms such as genetic algorithm [31], evolution strategy [10, 21], offspring selection genetic algorithm [2], local search, simulated annealing [27], tabu search [19], particle swarm optimization [26], and many more. Among the list of available problems in HeuristicLab are real-valued test functions, combinatorial problems such as the traveling salesman, vehicle routing, and the quadratic assignment problem, as well as data analysis problems such as regression and classification. The optimization layer also contains

analyzers that allow to study the performance of algorithms. Basic analyzers provide a quality progress, more sophisticated analyzers enable an inspection of the algorithms' behavior.

2.3 Interfacing with Simulation

In this section we describe three different strategies for performing simulation-based optimization with HeuristicLab. These strategies differ from each other with respect to their effort to set up the optimization and their possible applications.

- The first strategy (Sect. 2.3.1) is to create interfaces that couple HeuristicLab with specific simulation frameworks such as MATLAB or Scilab.
- Another possibility is to define a general inter-process communication protocol for data exchange that allows the coupling of arbitrary software with HeuristicLab (Sect. 2.3.2).
- The third strategy is to implement the simulation model within HeuristicLab, which results in the tightest coupling between the simulation model and the optimization algorithm (Sect. 2.3.3).

2.3.1 Specific Interface

A possibility to couple a simulation environment with an optimization framework is to provide a specific interface layer which is responsible for handling the communication between the optimization algorithm and the simulation model. Most simulation frameworks provide several ways to couple them with other applications using various kinds of technologies, ranging from direct calls to specialized application programming interfaces (API), component object model (COM) interfaces for interprocess communication, or web services. A drawback of specific interfaces is that they have to be implemented for each simulation environment and technology. However, they can be implemented in a generic way to execute arbitrary commands instead of running a specific simulation model.

HeuristicLab provides specific interfaces for MATLAB and Scilab out of the box. Both these frameworks excel at numeric computation, which makes them especially suited to perform continuous simulation. Furthermore, both frameworks provide specific modules to ease the development of simulation models. The interfaces for these two software systems have been implemented in a generic way allowing the execution of arbitrary scripts in the respective programming language. The MATLAB interface is based on the COM technology to enable communication with HeuristicLab, whereas the Scilab interface calls directly a native `C++` API.

A common problem in continuous simulation is the identification of appropriate parameter values to adapt the simulation model to the circumstances of the real-world—a use case is detailed in Sect. 3.6. Therefore, real-vector encoded parameter optimization problems with a coupling to MATLAB and Scilab have been imple-

```
//parameter assignment
p.m  = param1;
p.d1 = param2;
p.Fc = param3;

//import and execution of simulation model
importXcosDiagram('simulationModel.zcos');
xcos_simulate(scs_m,4);

//read original data and compute quality metric
originalValues = csvRead("data.csv",",",".");
simulatedValues = simulationModel.values(:,2);
quality = sum((values(:,1) - original(:,3))^2);
```

Fig. 5 Example of an evaluation script implemented in Scilab. The simulation is started using the defined model and the parameters given by the optimization algorithm (`param1` – `param3`), the quality is calculated as sum of squared differences between the output of the simulation model and previously measured values

mented in HeuristicLab. The evaluation of a solution candidate uses the aforementioned specific interfaces and executes a script in the simulation environment that calculates the quality of a solution candidate. As a result, every algorithm that can handle real-vector encoded solutions, such as for example evolution strategies, evolutionary algorithms, or simulated annealing, can be used to solve such parameter optimization problems.

A benefit of the chosen strategy is that the effort for configuration and programming is minimized for the user. The only part that has to be provided by the user to run the optimization is the evaluation script; the user can create this script in a familiar environment (MATLAB or Scilab) and does not have to learn the specifics of the HeuristicLab framework to execute the optimization. However, this minimal configuration effort comes with a price, namely that by default only real-valued parameters can be optimized with this approach.

Figure 5 shows an evaluation script for Scilab that parameterizes and runs a simulation model. Solution candidates, in this case combinations of real numbers (`param1` – `param3`), are generated by the algorithm and the quality of the parameter combination is calculated by the simulation framework. The here shown script loads an existing simulation model, sets its parameters to the values supplied by the algorithm, runs the simulation model, and finally extracts the quality of this given parameter vector by converting results of the simulation to a numerical value.

2.3.2 Generic Interface

A generic exchange protocol has been integrated in HeuristicLab that enables communication with external processes, such as simulators, and allows the encoding of several types of parameters. The protocol has been first described in [8]; in this section a summary will be given. It has also been enhanced with a cache which prevents the execution of simulation runs for solutions that have already been evaluated [38]. Since the execution of simulations can consume relatively large amounts of runtime,

```
message SolutionMessage {
  message DoubleVariable {
    required string name = 1;
    optional double data = 2;
  }
  message DoubleArrayVariable {
    required string name = 1;
    repeated double data = 2;
    optional int32 length = 3;
  }
  //... further sub-messages omitted ...
  required int32 solutionId = 1;
  repeated IntegerVariable integerVars = 2;
  repeated IntegerArrayVariable integerArrayVars = 3;
  repeated DoubleVariable doubleVars = 4;
  repeated DoubleArrayVariable doubleArrayVars = 5;
  repeated BoolVariable boolVars = 6;
  repeated BoolArrayVariable boolArrayVars = 7;
  repeated StringVariable stringVars = 8;
  repeated StringArrayVariable stringArrayVars = 9;
  repeated RawVariable rawVars = 10;
}
message QualityMessage {
  required int32 solutionId = 1;
  required double quality = 2;
  extensions 1000 to max;
}
```

Fig. 6 Definition of the generic interface messages in .proto format [8]

parallelization is a major aspect. The generic interface allows to specify several target machines that are running the given simulation with provided parameter settings and returns the results. The evaluation is thus distributed and allows even longer running simulation models to be optimized in reasonable time. The generic interface has been integrated in the form of a customizable problem definition in HeuristicLab and is explained in this section.

External Evaluation Problem

As the name implies, instances of this problem type have to be evaluated by an application that is external to HeuristicLab. This problem has no preconfigured representation or operators, but it can be customized. The *RealVectorEncoding* plugin contains operators that can be added if the simulation exposes real-valued parameters for optimization; if the parameters are discrete values, operators of the *IntegerVectorEncoding* plugin can be used to create and optimize the solutions. These encodings can also be used in combination if there are simulation parameters of both types.

Interoperability

In HeuristicLab, an evaluator that is applied on a solution scope calculates the solution's quality and adds this quality back to the scope. The evaluator of the *ExternalEvaluationProblem* collects a user specified set of variables, adds them to a *SolutionMessage*, and transmits this message to an external application. The evaluation operator then pauses and awaits the reply in form of a *QualityMessage*.

The quality value given in this message is then stored into the corresponding scope. This generic definition allows the use of many algorithms that are designed to optimize single-objective problems in HeuristicLab. To encode and transmit messages, the protocol buffer[4] framework has been used; the messsages' structure is shown in Fig. 6.

The protocol buffer format is designed to work with very compact files that are serialized, which minimizes transmission time. Furthermore, the serialization itself is also very quick. Google provides implementations of protocol buffers for Java, C++, and Python, and open-source ports also have been created for other languages such as C#, R, and many others.[5] The solution message buffer is a so-called "union type" protocol buffer, which is a very generic message for potentially unknown use cases. It includes fields for storing Boolean variables, integers, doubles, and strings as well as arrays of these types. In HeuristicLab, the *SolutionMessageBuilder* class translates the variables in a scope into variables in a *SolutionMessage*; this message builder can use custom converters for transcoding custom data types into a field of the solution message.

Parallelization and Caching

Parallelization is an effective means to reduce overall runtime if the necessary time to run a simulation model becomes very long. The overhead of the communication and the optimization procedure then become a negligible part. In HeuristicLab this is supported through the use of the above-mentioned *parallel engine*. This engine allows multiple evaluators to be executed concurrently, which in turn make use of multiple channels defined in the *Clients* parameter. In the background the *ThreadPool* in .NET is used to provide threads for efficient operations. To further decrease runtime an *EvaluationCache* and the respective evaluator can be used that hashes each visited solution and prevents further simulation runs. The cache can later be persisted to a file or exported as a comma-separated-values (CSV) file for further analysis [38].

Protocol Extensions

The *QualityMessage* can also be extended if more results or variables are included in the solution scope and shall be read and interpreted by an analyzer. This extension of the quality message can be achieved by creating a new message which extends the *QualityMessage*. Field numbers 1000 and higher can be used to declare extension fields. Figure 7 shows a .proto message definition that adds another field storing the number of repetitions.

Example Interface with AnyLogic 6

AnyLogic[6] is a simulation environment implemented in Java that allows to add Java code in various parts of the modeling process. In general, users are able to create model as *ActiveObjects* that might contain other *ActiveObjects*. A model can then be

[4] http://code.google.com/p/protobuf.

[5] http://code.google.com/p/protobuf/wiki/ThirdPartyAddOns.

[6] http://www.xjtek.com.

```
message MyResponse {
  extend HeuristicLab.Problems.ExternalEvaluation.QualityMessage {
    required int32 repetitions = 1000;
  }
}
```

Fig. 7 Extension of the quality message to return also the number of repetitions that have been performed

run in different experiments such as a *SimulationExperiment*. To couple the model with HeuristicLab and make use of the generic data-exchange interface, a special type of experiment is used. In AnyLogic the so-called *ParametersVariation* experiment allows to perform a set of simulation runs for certain parameters. These parameters can be varied automatically given certain bounds and a step size, and they can also be varied freely by an external program such as HeuristicLab. For this purpose a Java library *HL3ExternalEvaluation.jar* was added to the model, which is also available on the HeuristicLab website.[7] This Java library abstracts the data-exchange part and allows to set up the simulation model either as a push or a poll service for HeuristicLab. When choosing the push service, the model needs to implement an interface which is passed to the library; when opting for the poll service, the library can be polled for incoming solution messages and subsequently a quality can be returned.

2.3.3 Integrated Simulation and Optimization

While the methodologies presented in Sects. 2.3.1 and 2.3.2 are dedicated to parameter optimization of models that have been created using external simulation environments, an alternative approach is to implement simulation models directly in HeuristicLab and integrate them with optimization algorithms. The HeuristicLab architecture is generic in the sense that not only optimization algorithms, but arbitrary algorithms including simulations can be modeled. The applicability of this approach has been shown especially in the context of dynamic vehicle routing and various practical case studies [48, 50, 51].

The direct implementation of simulation models in HeuristicLab allows a tight coupling with optimization algorithms, which is beneficial in cases where efficient or sophisticated interactions between simulation and optimization are required. Integrative approaches require that the simulation framework and the optimization framework share a common platform or programming language. For HeuristicLab, this means that .NET based simulation frameworks such as Repast.net[8] or Sim#[9] should be utilized. This allows using the HeuristicLab API in simulation models and vice versa.

[7] http://dev.heuristiclab.com/howtos.

[8] http://repast.sourceforge.net.

[9] http://github.com/abeham/SimSharp.

When tackling control optimization problems an efficient and flexible integration is needed. In this case standard simulation software usually cannot be applied out of the box and only few generic and extensible modeling infrastructures exist for control optimization, e.g., dynamic vehicle routing [36]. Implementing control optimization problems with HeuristicLab requires that the simulation model references the HeuristicLab plugins. When running the model, the events that require a decision parameterize the corresponding optimization problem, the solver algorithm, and await the result. This can be done in a state where the simulation model is paused, or, in a real-time simulation, while the simulation model continues to run. The added complexity of real-time operation, such as the occurrence of changes while the optimization is running, has to be taken into account. Real-time simulation optimization requires a very tight combination of simulation and optimization that may require implementing customized optimization algorithms.

In parametric optimization problems, where the runtime of the simulation model is a critical and limiting factor, an integrated simulation and optimization approach reduces the inter-process communication overhead. Implementing such an approach requires the definition of a customized problem in HeuristicLab with a custom evaluator. In order to evaluate a candidate solution, the evaluator makes use of the simulation framework API, initializes the model, executes it, and creates the fitness values out of the model's performance indicators [7].

3 Real-World Examples

3.1 Simulation-Based Design of a European-Wide Logistics Network for Bio Residues

3.1.1 Problem Description

Increasing prices of fossil fuels and other nonrenewable energy sources have led to an increased interest in the development of alternatives. On the one hand, renewable energy sources, such as energy crops, are employed more often. On the other hand, we see an increased use of so-called second-generation bio-fuels which can be obtained by processing organic residues such as straw, wood chips, lifestock waste, or malt spent grains that do not compete with other food crops and have a more limited impact on greenhouse gas emissions [15, 25]. When processing these waste products, two problems are solved at the same time: Waste amounts are reduced, and precious resources are replenished. The inherent problem of this idea, however, is that it is typically rather uneconomical to further process or transport waste products, which is why they are considered waste in the first place, i.e., the effort invested in their transport is bigger than the expected revenue. The key to the mitigation of this problem is to increase the value density of these products by de-central and cheap

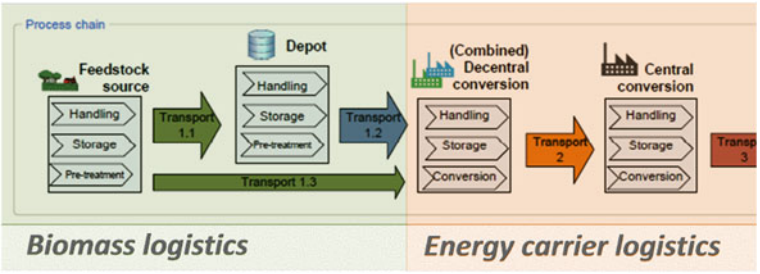


Fig. 8 Echelons in the logistics network

"upgrading" to intermediate energy carriers which can be transported over longer distances more economically.

This is why we have developed an optimized multi-echelon logistics network for the transport of feedstock, intermediate, and final products on a large scale within BioBoost.[10] In this research project, biomass potentials and key data on conversion facilities available in Europe have been compiled; this information forms the basis for planning a large logistics network. Suitable conversion plant locations and capacities as well as transport routes and product amounts are then optimized using simulation-based optimization: Many scenarios are iterated and evaluated, and metaheuristics are used to tune free variables so that the quality of the resulting logistics network is optimized. The approach described in this section encompasses several modeling optimizations to enable faster calculations.

3.1.2 Simulation Model

Several free variables are optimized: At each location, a certain *amount of feedstock* has to be obtained which is then *transported* to a *certain plant* for de-central processing. The intermediate energy carrier is then *transported* to a central plant where it is *converted* into heat, fuel, or other end products. This overall process is shown in Fig. 8.

The initial solution space size in a naive model would be determined by the number of free variables. In the case of the logistics network, the following factors have to be considered:

- Locations of feedstock sources and the utilization levels for each of these sources. In this case more than 1000 level 3 NUTS regions [14] have been used as possible source locations.
- Locations of the intermediate and central processing plants placed on any combination of more than 1000 regions.

[10] http://www.bioboost.eu.

- Connections of sources and targets in each echelon of the logistic network. The logistics network can be modeled as an adjacency matrix containing the links between source and target locations. This matrix would then have a size greater than $s = 1000 \times 1000$. Moreover, every subset of connections will then be a valid solution candidate giving a total of $s! = 8 \times 10^{5565708}$ possible routing networks for each type of feedstock.

The large size of this original solution space renders a direct solution of the problem infeasible. For this reason it was attempted to model the solution space so that it remains susceptible to optimization by providing a meaningful neighborhood definition, and at the same time removing as many unnecessary scenarios as possible.

3.1.3 Optimization

A very strong simplification can be made by allowing only one target region per source region. This immediately reduces the number of choices per feedstock to approximately 10^9. However, the solution space for one complete scenario still comprises at least two echelons and at least three different feedstock types each of which contains a choice for source and target region, their connectedness, and the amount of acquired feedstock. Therefore, the solution space size is still in the area of around 10^{21} per product.

A second reduction of the solution space size can be achieved by eliminating variables or variable choices that would lead to solutions that can be guaranteed to be inferior. One naive possibility would be to allow only transport targets that are directly adjacent to the source region. The variant we have employed was to automatically select plant locations and capacities based on the transport targets of different feedstock types as this further reduces the number of free variables and, simultaneously, the solution space size to 10^{15} possibilities per feedstock type. After these transformations, the resulting solution space is manageable with current metaheuristic optimization methods.

In a third round, computationally expensive calculations of routes between source and target locations have been replaced with precalculated estimates to reduce the computation time. Furthermore, a speed-up has been achieved through aggregation into yearly calculations instead of step-wise event-driven simulation [30].

Finally, while the solution space is perceived as fixed for the whole optimization process, some combinations of variable choices can lead to meaningless, equal, or inferior results. For this purpose we have developed a mechanism that dynamically reduces the solution space only during the application of variation operators, hiding inferior options, which again leads to a significant runtime reduction.

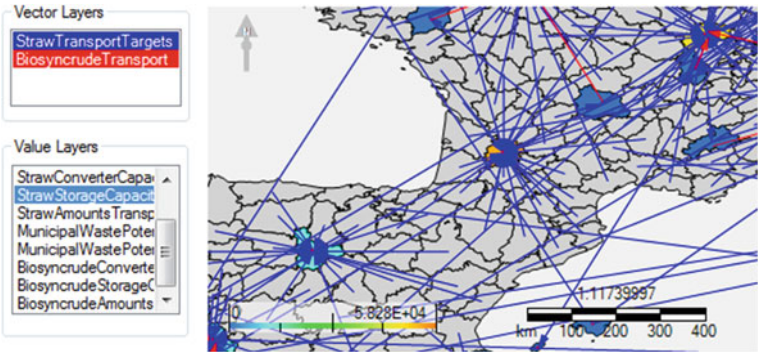


Fig. 9 Solution visualization within HeuristicLab

3.1.4 Conclusion

The simulation model has been implemented based on the HeuristicLab optimization environment [52, 53] where the evaluation of a two-echelon scenario takes about 1 ms on average for one feedstock type and meaningful optimization results are available after 2–12 h when executing the optimization on a single computer with a Core 2 Duo processor. Figure 9 shows a screenshot of the implementation in HeuristicLab with a visualization of the feedstock utilizations and transport vectors.

Using only a few simplifications and the powerful and flexible optimization algorithms available within HeuristicLab, an optimization task that initially seemed intractable has been reformulated to allow its optimization and might help to reduce the amount of waste while increasing the amount of energy available in the future.

3.2 Simulation-Based Priority Rule Optimization for Scheduling Production Systems

Problem Description

Scheduling plays a key role in industrial systems to ensure the efficient use of resources and timely completion of orders. Briefly summarized, a scheduling problem can be described as a set of jobs, each having a collection of operations that are tied to a machine. By specifying the start time of each operation a schedule is constructed. A number of different types of shops have been described in the literature such as job shop, flow shop, and open shop [37]. In the flow shop problem, the sequence of operations is the same for all jobs, while in the job shop, the sequence may be different for each job. In the open shop there is no predefined sequence. Generally, scheduling problems such as the job shop scheduling problem are NP hard as shown in [18, 37].

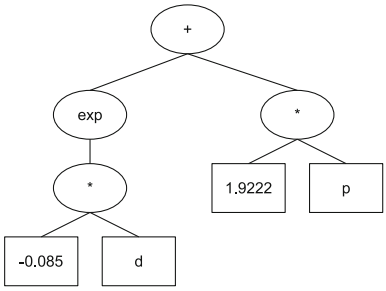


Fig. 10 Example of a tree-encoded priority rule. The variables d and p represent job properties, e.g., due date, processing time, batch size

When solving scheduling problems one can make a decision very early and *plan ahead* or rather late and *decide in time* as in the saying "we'll cross that bridge when we get to it." The first case can be seen as an *offline* approach to determine the optimal schedule in advance. This yields a high quality schedule, but requires high runtime and adaption in the face of changing conditions. The second case is denoted as an *online* approach, which considers the actual state of the system and which thus has to to make decisions much more quickly [3]. One popular method in such an online approach is the use of simple rules [34]: First-in-first-out, earliest-duedate-first, and many more have been proposed to rank and prioritize the pending order queue. A combination of simple heuristics to more complex priority rules, as can be seen in Fig. 10, allows creating tailored and customized rules for specific scheduling scenarios.

Simulation Model

Simulation is particularly suited to support the optimization of complex priority rules. When a machine becomes idle all items in its queue are ranked and the best ranked item is processed next. If a job is finished the performance metrics are updated. At the end of the simulation run the remaining jobs are also rated; this is important as otherwise "problematic" jobs, e.g., that require long setup, would potentially starve in the system and never get picked up.

The model itself describes the flow of the jobs, the entities available in the production system, as well as the interactions between them. Workers arrive at the production plant in the morning and pick up work. They will process jobs on the machines, make a pause, go for lunch, and continue to work. In the simulation model the decision of which item the workers are to pick up next is made using the priority rule as follows:

1. The set of possible decisions is constructed;
2. For each decision a dictionary is created that contains state variables, item characteristics, and more, see Table 1 for examples;
3. An interpreter reads the priority rule and computes a rank using the variables in the dictionary;
4. The set of decisions is sorted according to their rank;
5. The best ranked decision is implemented.

Table 1 State variables of a production system [38]

p…batch size of an order	n_r…number of remaining steps
p_r…remaining batch size after this step	l…queue length of a machine
d…job due date	s…setup required (± 1)
q…job quantity	Q…qualification
t…number of required tools	t_r…remaining processing time

Some situations require that multiple decisions should be made at the same time, e.g., selecting a worker and an item that she should process. This can be useful if the worker qualification or worker satisfaction is taken into account. In this case all combinations of workers and items constitute the set of possible decisions. However, as the evaluation of a decision is not without cost, this provides an additional performance hit and prolongs the simulation run. In the concrete studies in [5, 38] the interpretation of the priority rules was compiled to Microsoft intermediate language code prior to running the model in order to speed up the simulation runtime.

The performance metrics are translated into a fitness of the priority rule by a rating model. The formula may vary depending on the case, but timeliness, troughput, cash flow, or waste production are important factors. For instance, a linear step function can be designed to give a slight penalty for jobs that are finished too early and a heavier penalty for delayed delivery. It is advised to avoid highly discontinuous or flat transformation functions as this creates a very rough or very flat search space that may be difficult to optimize.

Optimization

Typically, the enterprise resource planning system of a production company provides the necessary data such as jobs, working plan, bill of materials, resources, due dates, processing times, and more to parameterize the simulation model. The importance of separating this data in training and test scenarios must be emphasized. Optimized rules may become highly specific to the scenario for which they have been trained and may not generalize very well. Simulating the optimized rules with data from test scenarios allows identifying generalizable rules that show good performance in both training and test. Additionally, maintaining an archive of rules allows exploring the Pareto front between complexity and quality; simple rules may not appear to perform well in the training, but might generalize better [38]. Such an archive may also be used to avoid re-evaluation of already known solutions. Simulation models of such production systems can become quite complex and, therefore it takes a few seconds to finish a run of several weeks of the production plant. Distributed evaluation of the simulation model is quite important in order to obtain good priority rules in reasonable time.

Using genetic programming [4, 29] tree-based priority rules can be generated and evolved to match the scenarios at hand. For this purpose trees are generated randomly and crossed with other trees in a population; better trees have a higher chance of being used for crossover, and therefore their features will prevail in the

following generations. While genetic programming traditionally employs a standard genetic algorithm, variants such as offspring selection genetic algorithm [1, 2] have worked quite well. Offspring selection introduces an additional selection step that discards offspring that are inferior to their parents.

Conclusion

Production systems benefit from suitable decision rules that are tuned to improve long-term goals such as timeliness or throughput. These rules take some time to optimize, but are quick to evaluate in the production system. Volatility in those systems is a concern that can be addressed through a continued reoptimization of these rules and by maintaining a comprehensive set of production scenarios that show many different characteristics. In practical implementations it is also important to assure that the inputs to the rules are available to the rule and that the data is accurate. For instance, if queue length is a highly relevant variable in such a rule, but it cannot be obtained in the real system the priority rule cannot be implemented. However, it would not be wise to omit these variables altogether as it might indicate that additional sensory data should be acquired.

3.3 Simulation-Based Optimization of Inventory Replenishment Rules

This section is based on a study that was previously published by Vonolfen et al. [48] and deals with the simulation-based generation of inventory replenishment rules for stochastic inventory routing problems. The evolved rules are evaluated and tested in the context of retailing based on real-world data with a large number of different products that are replenished at supermarkets. The methodology is an example of simulation-based priority rule generation using an integrated simulation optimization approach (as outlined in Sect. 2) applied to a real-world scenario.

Problem Description

The inventory routing problem (IRP) integrates inventory management and transportation. It is a mathematical model for the concept of vendor managed inventory (VMI) where the vendor has the responsibility for the replenishment of the customers, which requires information about the inventory levels to be available. According to Waller et al. [56], VMI was first popularized by Walmart and then implemented in various other companies. The goal is to minimize the inventory and transportation costs while maintaining a certain service level.

The IRP was first presented by Bell et al. [9] who considered the distribution of industrial gases; since then, many variants of the IRP have been studied. Moin and Salhi [32] provide an overview where they state that most problem formulations do not consider stochastic demand patterns and are deterministic. In contrast, the stochastic IRP (SIRP) considers product usages as probability distributions, but this also increases the problem complexity, which motivates a simulation-based approach.

The considered IRP here is a mixed formulation in which some of the customers choose VMI, while the other customers keep an order-based strategy. Multiple

products P are distributed from a central depot to a set of order-based customers (O) and a set of VMI-customers (N) using a homogeneous fleet of vehicles (V) with a known capacity C_v for each vehicle $v \in V$. Each VMI-customer $n \in N$ has a known storage capacity C_{np} for each product. The planning process is performed on discrete time steps t, which are days in real operation.

A joint probability distribution P^w_{np} is given for each weekday $w \in [0..6]$, customer n and product p from which the product consumption can be sampled for a given time step during the simulation. The inventory level x^t_{np} at a customer of a certain product can be measured every day. The order-based customers place fixed orders $d^t_{op} \in D^t_f$ which they derive from an ordering strategy that is not influenced by the vendor.

Each day, the vendor replenishment d^t_{np} and customer orders d^t_{op} are combined into vehicle routes R^t. The objective is to minimize the required vehicle fleet size $|M|$ as well as the driven distance $d = \sum L_r$ while maintaining the service level by preventing out-of-stock situations where x^t_{np} falls below a certain safety stock.

The motivation to apply simulation-based priority rule generation stems from the high problem complexity resulting from the practical case study performed in cooperation with an Austrian retailer. The considered scenario consists of 84 supermarkets which are served from a central depot. In total, the supermarkets serve 5113 different fast-moving consumer goods that have stochastic demand distributions with high fluctuations during a week. The supermarkets are served with a fleet of homogeneous vehicles. A main challenge is to flatten the peaks in the resource usage and to balance it more evenly to achieve constant resource usage. This is complicated by the very limited storage capabilities at the individual supermarkets, while each individual product contributes to the service quality since out-of-stock situations may lead to a potential loss of revenue.

Simulation Model

The simulation model developed for this inventory replenishment problem is an agent-based formulation consisting of vendor, customer, and vehicle agents. We have implemented the simulation model based on Repast.net[11] agent-based simulation framework and integrated it into HeuristicLab.

The interactions between the agents are illustrated in Fig. 11: Each customer agent has an inventory which is updated by sampling from the demand distributions. Depending on whether the customer has a vendor-managed inventory or places the orders by itself, different interactions with the vendor occur. In the case of VMI, the vendor uses a replenishment policy to determine the replenished demands. In the case of order-based customers, the customer uses an order policy and the vendor has no access to the inventory. In that case, a classical threshold-based order strategy is used which keeps a certain security buffer.

For each day of operation, the accumulated demands are converted into a standard capacitated vehicle routing problem (CVRP) instance which can be solved with any VRP algorithm available in HeuristicLab. The calculated tours are then executed by

[11] http://repast.sourceforge.net.

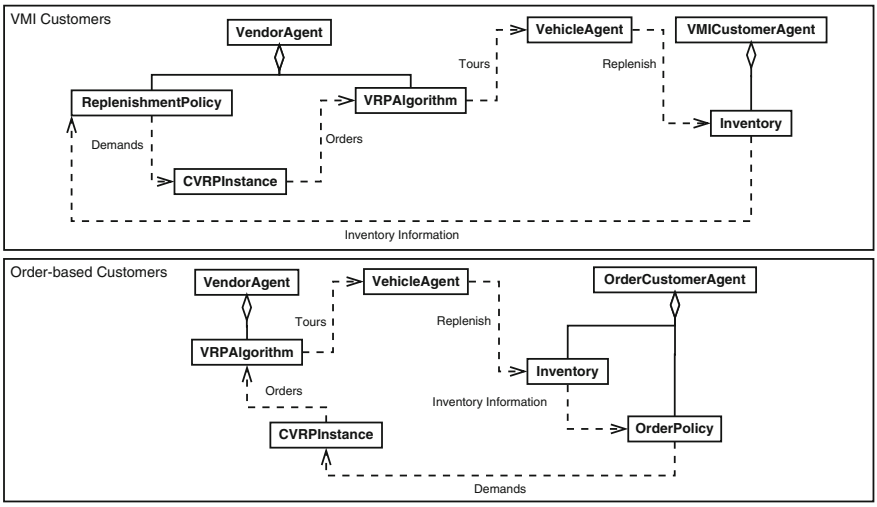


Fig. 11 Interactions between the agents for VMI and order-based customers

vehicle agents which execute the tours and replenish the inventories of the customers. This results in a two-stage approach where first the replenished goods are determined and then the tours are calculated for each day of operation.

Optimization

The aim is to automatically evolve inventory replenishment policies that are able to balance the resource usage with fluctuating demands maintaining a given service quality by preventing out-of-stock situations for each individual product. In order to reach this goal we apply simulation-based generation of priority rules as described in Sect. 2.

The inventory replenishment policy has the main goal of constant resource utilization with fluctuating demand distributions. It consists of two priority rules: The first rule is responsible for choosing a set of customers that should be visited, the second rule determines the amount of replenishment for each product at these customers. This two-stage approach aims at taking into consideration both the routing (e.g., avoiding to visit customers that are geographically far away) and replenishment (e.g., avoiding out-of-stock situations). The description of the policies is based on Vonolfen et al. [48] where the reader is referred to for details.

The replenishment policy is represented as a real-valued vector that consists of general parameters and parameters of the first and second priority rule. The two general parameters are *CapacityUtilization* and *PriorityThreshold*. The capacity utilization parameter determines the capacity that should be used over time to replenish the customers. This capacity is used as a basis for the replenishment rule and aims at constant resource utilization. The *PriorityThreshold* parameter determines the minimum priority a customer must have to be considered for replenishment to avoid unnecessary detours.

The first priority rule calculates a priority for each customer n to be replenished by weighting $m = 6$ factors f_{ni} with an parameterized weight a_i to a priority score $p_n = (\sum_{i=1}^{m} f_{ni} * a_i)/m$. The factors considered for a given customer n are (as discussed in detail in [48]):

f_{n1} : The minimum expected amount of days a stock-out will occur
f_{n2} : The average expected amount of days for stock-outs
f_{n3} : Number of days since the last delivery
f_{n4} : Total inventory size
f_{n5} : Minimum detour to incorporate the customer in existing routes
f_{n6} : Geographic isolation

The second priority rule determines the amount of a product that should be delivered to a given customer. If the expected days a customer would run out of a product falls below a certain *RefillThreshold* (derived from the stochastic product consumption rate information) or the available stock falls below a certain *RefillBarrier* (predefined amount of safety stock) it is refilled to a certain level determined by the *RefillFactor* which defines this level as a ratio of the maximum storage capacity for the respective product.

In total, the parameter vector for the replenishment strategy to be optimized contains 11 parameters: *CapacityUtilization*, *PriorityThreshold*, $a_i (1 \leq i \leq 6)$, *RefillThreshold*, *RefillBarrier*, and *RefillFactor*. These parameters are optimized using an evolution strategy as an priority rule optimization algorithm. Each candidate solution is evaluated by running a simulation run of a time frame of 60 days and evaluating the resource usage as well as the service quality. For each day of the simulation, the replenishment amounts are calculated using the parameterized replenishment strategy and the resulting tours are optimized using a routing algorithm (push-forward insertion heuristic) implemented in HeuristicLab. After each day, the inventory level at each customer is reduced by using the predefined demand distribution.

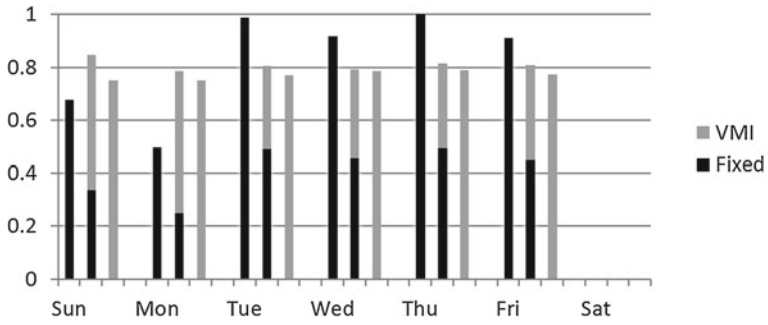


Fig. 12 Average resource utilization over different weekdays for different scenarios [48]. The x-axis represents the weekday and the y-axis the relative resource utilization. The resource utilization is divided for the fixed and VMI customers

Conclusion

As illustrated in Fig. 12, we are able to balance the resource utilization using a vendor managed inventory. When the customers place the orders themselves using a classical threshold-based order strategy, the fluctuations in demands over weekdays lead to a corresponding fluctuation in resource usage, which is undesirable for determining needed capacities. The biggest smoothing effect could be achieved if all customers chose to apply a VMI; however, even if only part of the customers are switched to a VMI, this effect can also be observed.

As a practical guideline for applying simulation optimization it can be concluded that simulation-based priority rule generation is a powerful method for making operational decisions in high-dimensional and stochastic problem environments. By modeling different scenarios and evaluating the optimization potentials, simulation-based optimization aids in making tactical and strategical decisions.

3.4 Simulation Optimization of Transport Activities Within Steel Slab Logistics

This section deals with transport optimization within steel slab logistics and is based on a previously published study of Vonolfen et al. [50]. The main aim is to evaluate optimization potentials in the transportation of steel slabs in terms of throughput maximization within cold-charge. The core of the approach is a detailed simulation model considering constraints of the cold-charge steel production process. According to the classification presented in Sect. 2, it is an example of parameter optimization where the parameters are the transport sequence in this case and the simulation model is integrated with the route optimization algorithm in HeuristicLab.

Problem Description

Steel production is a multi-stage process and is generally geographically distributed and energy as well as capital intensive [43]. The typical production process starts with raw materials and the melted steel produced in the furnace is transformed into slabs at the continuous casting machine. The slabs are then rolled into plates or coils in the rolling mill. Generally, there are two pathways for steel slabs. In the hot-charge process they are transported directly from the caster to the rolling mill, while in the cold-charge process a slab yard is used as an intermediate buffer storage.

This work focuses on scheduling the transport activities within the cold-charge process which are linked to the lifecycle of a steel slab illustrated in Fig. 13. Within cold-charge three transport activities can be identified: transportation from the caster to the slab yard, transportation to processing aggregates, and transportation to the rolling mill. Scheduling the individual transport activities is not a trivial task, since upstream and downstream processes of steel production have to be considered and transport links them together.

In the presented case study, straddle carriers transport the slabs and the activities are scheduled by a human expert. The straddle carriers can carry up to 105 tons

which usually correspond to around four slabs. An outside slab yard is used as an intermediate storage which decouples casting and rolling. The slab yard consists of several fields which are organized in lanes each containing several stacks of slabs. If a slab that lies beneath other slabs is retrieved, shuffling operations have to occur. At the continuous caster as well as at the processing aggregates and the rolling mill, stacks are retrieved and stored in stacks at handover places.

Simulation Model

Simulation optimization is applied to evaluate the optimization potential in scheduling the three types of transport activities optimizing total throughput while considering all relevant operational constraints. The motivation to use a simulation model are the dynamic interactions between the individual activities which would be difficult to represent as a static model. There are several operational constraints for the individual transport activities (cf. [50]):

- Shuffling constraints concern the retrieval and storage of slabs at handover places and the slab yard. No shuffling is possible at handover places which means only the topmost slabs at handover stacks can be retrieved. In the slab yard, only a single straddle carrier can operate at an individual handover place.
- Rolling constraints ensure the correct transportation of slabs to the rolling mill. A certain security buffer of slabs scheduled to be rolled has to be present at the rolling mill which cannot be underrun. Additionally, the rolling sequence has to be followed to a certain degree, otherwise the cranes at the rolling mill have to reshuffle the slabs.
- Temporal constraints concern availability of straddle carriers where each straddle carrier has scheduled maintenance tasks as well as driver breaks. Additionally, the processing schedule has to be considered which means slabs have to arrive earlier than their scheduled processing time.
- Capacity constraints are that each carrier can transport up to 105 tons and slabs must have similar dimensions to be transported together.

The simulation is carried out in discrete time steps where each step represents a minute of operation. The three types of events that occur are actions performed by straddle carriers, update of handover places by simulated movements caused by upstream and downstream activities and the rolling of slabs using the predefined rolling sequence.

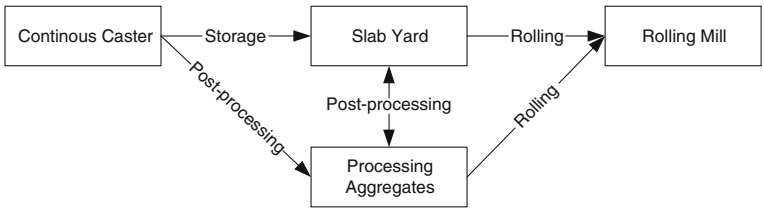


Fig. 13 Lifecycle of a slab in the cold-charge steel production process (cf. [50])

Optimization

For the optimization of the transport schedules, a unified tabu search heuristic is used which can be applied to several routing problems and was presented for pickup and delivery problems by Cordeau and Laporte [12]. By neighborhood exploration it systematically improves the current solution using a tabu list to prevent cycling. The basic neighborhood operation is moving a request to another vehicle at the best possible insertion position. Moving back the request to the original vehicle is made tabu for a given number of iterations.

The quality of a given schedule is evaluated with a simulation run where key values concerning throughput and constraint violations are considered. The key values are combined in a single quality value (cf. [50]):

$$\begin{aligned} quality = {} & travelTime + shufflingTime \\ & + \alpha * shufflingViolations + \beta * rollingViolations \\ & + \gamma * temporalViolations + \delta * capacityViolations \end{aligned} \tag{1}$$

The objective is to maximize the total throughput and thus to minimize the total travel and shuffling time of the straddle carriers. Constraint violations are penalized using penalty factors which are adapted during the search process. If the current solution is feasible, the penalty is decreased and if it is infeasible it is increased. This allows the algorithm to move through infeasible regions of the search space.

During an iteration, a large number of possible insertion positions have to be evaluated. Since a detailed simulation is very time-consuming a combination of a static evaluation and a simulation evaluation is used. The full simulation is only run for interesting insertion positions while the static evaluation serves as a lower bound. However, the static evaluation does not consider the dynamic interactions which include the shuffling constraints and the interactions between the straddle carriers (locking of rows). In preliminary experiments, the best found tradeoff between runtime and achieved quality was to evaluate the best 10 insertion positions for each neighborhood operation using a full simulation.

Conclusion

After exporting 10 historical shifts and comparing the optimized schedule with the schedule created by the domain expert, and by analyzing the optimized schedules, the main bottleneck that was identified were the shuffling operations in the slab yard. The algorithmic solution works around this problem by performing more trips to the storage area and thus avoiding stacking and shuffling operations. The effort to create temporary stacks, which are required when picking multiple slabs at once from the yard, is reduced significantly. The reduction of capacity utilization is compensated by minimizing the time traveled empty by sequencing the individual trips more efficiently.

To tackle the main identified bottleneck of shuffling operations, an efficient storage assignment is needed that considers the rolling schedule [28]. Also, when creating the

rolling schedule the current storage assignment in the slab yard should be considered to reduce shuffling which is known as the slab stack shuffling problem [44]. In the long term, a holistic model should be created that combines the optimization of casting, transport, storage, and rolling due to strong interdependencies.

In terms of practical guidelines, this real-world case study has shown that bottlenecks in the process can be identified by applying simulation optimization. The optimized schedules are evaluated and compared with original schedules using a simulation model to identify potential process improvements. When considering a process where the individual activities are strongly interconnected, it makes sense to create a holistic model instead of optimizing the activities independently. A possible approach is to integrate several individual simulation and optimization models [51].

3.5 Material Flow Simulation and Layout Optimization

Problem Description

The design of manufacturing plants is a complex process where several criteria determine the feasibility and suitability of a certain arrangement. The floor layout contains the storage zones, paths, and workstations required to store, transport, and process the materials into intermediate or end products. Simulating the processes on the computer presents several insights onto the performance of the future plant and may guide the planners' decisions. Often, a large amount of data is available when rearranging the internals of an existing plant which can be obtained from enterprise resource planning (ERP) systems. In designing new plants from the ground up, some assumptions will have to be made. A good overview of facility layout problems is given in [13].

Simulation Model

Common to most ERP systems is the notion of a *job*, which is split into one or more *operations* which in turn demand zero or more *resources*. The outcome of a job is either an end or intermediate product which is either shipped to the customer or placed in the company warehouse. Operations describe basic tasks and are executed in a certain order. Operations are not limited to production tasks; they may also include management tasks such as monitoring or coaching [6]. In a simulation model the material flows can be calculated by taking into account the production plan, that is, the start and processing times of the operations and their assignment to a machine. In rearrangement tasks it is possible to rely on past data if the future outlook is similar, but quite a large amount of data must be accommodated as the time period can stretch over one or several years. In a busy plant this means that hundreds of thousands operations have to be considered.

Three different kinds of flows can be identified when running a simulation model that can later be combined to form a similarity matrix between the locations [6]:

1. A *sequential flow* occurs when there is a transition from one operation to the next.

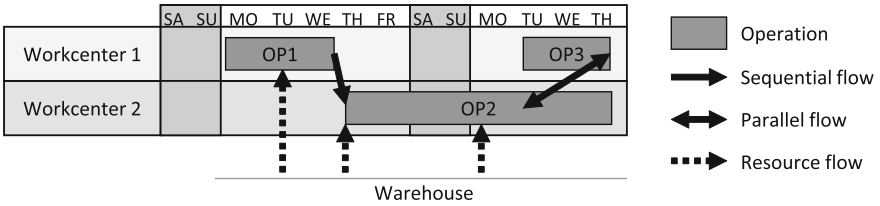


Fig. 14 Examples of sequential, parallel, and resource flows on three operations in a job

2. A *parallel flow* arises when an operation starts while another operation of the same job is still running. Parallel flows are less common than sequential flows, but they can indicate strong collaboration if an assembly part requires the cooperation of several work centers.
3. A *resource flow* occurs when an operation demands materials. This demand can then be satisfied directly through a flow from the last operation of a producing job or by an unknown source, typically the warehouse.

These flow types are visualized in Fig. 14. Sequential and parallel flows represent the flows of products or intermediate products and are passed on or shared between operations. Resource flows represent the flows of items used in the manufacturing of the products. In a layout optimization scenario the sequential and parallel flows would indicate a closeness due to the sequence in the manufacturing process. Resource flows indicate requirements for buffer capacities or closeness to the storage area of which they receive the material. However, resource flows also indicate supplier-producer relationships within the plant.

The strength of a flow is calculated as the sum of weighted transitions between two operations. These weights can be the occurrence or the number of materials that are transported or a number of containers. A special case during flow simulation also occurs in 1:N and N:1 transitions, i.e., when an operation has several possible successors or predecessors. In such cases it is possible to specify additional data such as process flowcharts or to either split and combine or duplicate the handling events. An appropriate assumption depends largely on the problem scenario. It is necessary to discuss and decide on these possibilities in the preparation stage if the strength of the flow should be a valid approximation of the closeness between two work centers.

Optimization

The faced optimization problem here is to arrange a set of rectangular shapes R on a flat surface G such that they lie completely within a boundary polygon P with p_i being the points of the closed polygon. The problem further contains the set B of fixed blocks, which are immobile locations in the layout. Each shape in R represents a work center and is specified by the location of the center coordinates, the dimensions of the rectangle, and a rotation, e.g., in 90° intervals. Each shape in B is specified by the lower left and upper right points. Finally, the matrix F specifying the flow strength is given as a $N \times N$ matrix with $N = |R|$. Elements of this matrix are called f_{ij} and denote the strength of the flow from i to j.

A solution to this problem specifies the location as x and y position, the width w, and rotation of each shape in R. The solution thus can be encoded in the form of multiple vectors of integer values if G is assumed to be discrete. Two vectors $\mathbf{x}$ and $\mathbf{y}$ encode the location on the plane, one vector $\mathbf{w}$ denotes the width—the area A_i remains fixed, so the height is given as $\frac{A_i}{w}$, and the last vector denotes the rotation $\boldsymbol{\omega}$. The distance matrix D with elements d_{ij} between all shapes $i, j \in R$ represents the shortest Manhattan-distance between the rectangles' edges.

The main fitness characteristic, the flow-distance-fitness Q_{flow} is defined as

$$Q_{flow} = \sum_{i=1}^{N} \sum_{j=1}^{N} d_{ij} * f_{ij} \tag{2}$$

The second fitness characteristic, the relayouting costs $Q_{relayout}$, represent the costs of transforming the given layout into the optimized layout. For this purpose each shape $i \in R$ can be attributed with a movement cost mm_i that depends on the distance that the shape is moved as well as a fixed cost ms_i, e.g., for packing and unpacking or calibration. For this purpose a vector of transition distances t_i is calculated that contains the Manhattan-distance between initial and optimized location.

$$Q_{relayout} = \sum_{i=1}^{N} x_i * (ms_i + mm_i * t_i) \tag{3}$$

where x_i is a decision variable that is 1 if $t_i > 0$ and 0 otherwise.

Constraints can be modeled as a penalty that is added to each evaluated layout. The first penalty $C_{overlap}$ deals with the constraint of nonoverlapping elements; this can occur frequently as an element's location is modified in a manner that does not consider feasibility. The second penalty $C_{boundary}$ puts a penalty on shapes that lie at least partly outside P by summing the area that falls beyond the boundary weighted with the distance to the boundary. The third constraint $C_{distance}$ is violated when the distance of two shapes is smaller or larger than the bounds specified on their mutual distance. The last constraint C_{aspect} adds a penalty to solutions in which the aspect ratio of the shape is outside the given bound. The fitness value is then computed as weighted sum of the qualities and constraints as formulated by

$$\begin{aligned} fitness = {} & \alpha_1 * Q_{flow} + \alpha_2 * Q_{relayout} \\ & + \alpha_3 * C_{overlap} + \alpha_4 * C_{boundary} + \alpha_5 * C_{distance} + \alpha_6 * C_{aspect} \end{aligned} \tag{4}$$

Conclusions

Optimizing such a layout presents many alternatives into possible rearrangements with a strong influence of the connectedness between work centers. The layouts solved by the current model do not lead to immediate practical layouts, as many real-world issues, such as infrastructure connections, security, social, and legal

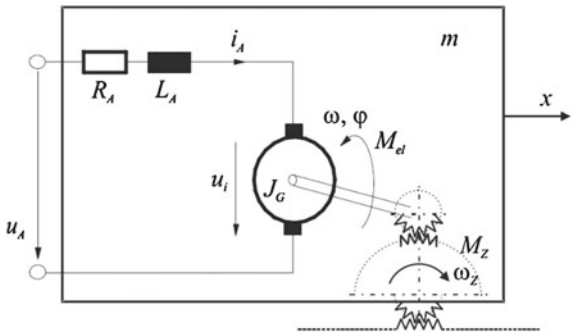


Fig. 15 Schematic picture of the electric cart system

requirements are not taken into account. Still, in a planning process together with a human planner starting points can be identified and good arrangements may be found that have not been thought of before.

3.6 Parameter Optimization of Continuous Simulation Models

Problem Description

Parameter identification in this context refers to the identification of the best possible parameter values of a simulation model: A simulation model has to be adapted to the concrete circumstances of the modeled system in order to match the real-world as exactly as possible. If this adaptations were not performed, the whole effort of creating the simulation model would be pointless because its predictions would be inaccurate. For example, the friction coefficients of a cart vary depending on the surface it is moved on and have to be adapted in the according simulation model. The only prerequisite of this parameter optimization approach is that the structure of the simulation model has to match the system which is modeled since otherwise, regardless of the effort used for parameter optimization, an adaptation to the real system would certainly fail.

Simulation Model

As a reference application we here consider the well-known cart system with an electric motor, where the vehicle mass m_1, the linear friction coefficient d_1, and the static friction coefficient F_C are unknown but constant. The simulation model is implemented in Scilab/XCos with the three free parameters m_1, d_1 and F_C, which have to be identified based on a known current U_A and measurements of the position x of the cart. Figure 15 schematically shows the electric motor and the corresponding differential equations of the continuous simulation model are displayed below.

$$\dot{x} = x$$
$$\dot{v} = -\frac{d_1}{\tilde{m}} \cdot v - \frac{1}{\tilde{m}} \cdot F_C \cdot sign(v) + \frac{k_m \cdot n}{r \cdot \tilde{m}} \cdot i_A$$
$$\dot{i_A} = -\frac{k_m \cdot n}{L_A \cdot r} \cdot v - \frac{R_A}{L_A} \cdot i_A + \frac{u_A}{L_A}$$
$$\tilde{m} = m_1 + J_A \cdot \left(\frac{n}{r}\right)^2$$

Furthermore, a simulation model of a simplified cart system and of a cart with a pendulum attached were used to test the suitability of the approach.

Optimization

In the context of parameter optimization, HeuristicLab is used to identify to parameter values of continuous simulation models which are implemented in Scilab/Xcos. Therefore, a generic coupling between HeuristicLab and Scilab has been implemented (Sect. 2.3.1) that allows the execution of arbitrary Scilab scripts. When a parameter optimization problem for simulation models should be solved, the script is responsible for executing the simulation model with suggested parameter values and calculating a quality value. The quality value expresses the accordance between the results of the simulation model with the currently used parameter values and the observed measurements in the real-world. Most of the times the sum of the squared errors at predefined time steps is calculated and used as quality value.

Every algorithm which is able to handle real-vector encoded problems could be used to solve this parameter optimization problem for continuous simulation models. HeuristicLab provides several algorithms which are suitable for this task: Genetic algorithms, evolution strategies, simulated annealing, etc. However, it was observed that the best results regarding solution quality, convergence speed, and robustness were obtained using the covariance matrix adaptation evolution strategy (CMA-ES) [21].

Conclusion

The presented approach for parameter identification has the great advantage that no information about the simulation model is needed, as the only information exchanged is a parameter vector generated by the optimization algorithm and its according quality value calculated by the simulation model. Therefore, a whole new range of optimization algorithms become applicable and one can refrain from implementing parameter optimization algorithms anew.

3.7 Electric Power System Optimization with Policy Functions

Problem Description

The electric power systems research society early identified the necessity of optimization both for planning and operation tasks, formulations such as the optimal power flow (OPF) problem shape this research domain ever since [33]. At the same

time, technological changes to electric power grids challenge new methods, requiring optimization in both dynamic as well as uncertain systems. In this context, heuristic optimization methods have evolved which are capable of managing many of those upcoming needs. Simulation optimization with metaheuristics provides a promising ground for handling uncertain dynamic problems, which makes it attractive to dynamic stochastic optimal power flow (DSOPF) issues [22, 23].

Such multi-period considerations (i.e., dynamic problem formulations) are often necessary for control issues [42, 46, 47], where optimal decisions have to be made over time while satisfying constraints robustly under stochastic conditions. Especially in future smart electric grids, among others the control of huge amounts of distributed devices (e.g., for the sake of load control) is a crucial challenge. Here, the optimal control of electric vehicles' charging processes has been identified as one of the hot spots in actual smart grids research. It can be formulated as a DSOPF problem and shall here be considered as an exemplary case.

The main idea of controlled charging is that some kind of central or decentral control influences the individual charging behavior of each single electric vehicle (EV) within a given fleet. Common objectives are system-wide peak-load avoidance, correlated charging with renewable supply, and in general the protection of existing distribution grid equipment. Such charging control decisions would need to be derived online (e.g., when an EV reaches a charging infrastructure) through consideration of the system's actual state (e.g. the EV's actual battery state-of-charge, the power grid condition, or the current supply from solar/wind power plants). Hence, similar to the approach of priority-rule optimization, a general function is needed that provides (near-) optimal charging decisions at runtime.

This is the aim of so-called policy function approximation, where an analytic function shall be identified that returns a control decision given a state without the need for embedded optimization. As demonstrated in detail in [23], such a policy function can be approximated using simulation-based metaheuristic optimization.

Simulation Model

Charging control decisions need to consider the power grid's point of view on the one hand (e.g., for avoidance of peak load values, satisfaction of secure power grid operation), but additionally have to satisfy the end-users' needs (recharge the needed energy for the next tour). While especially in modern considerations the uncertain supply provided by wind and solar power plants has to be included into load-control formulations, the resulting simulation model needs to contain three parts: the load flow simulation for deriving the grid's physical state, the traffic simulation that mimics the users' EV usage, and finally the renewable supply simulation that probabilistically describes the uncertain power injection from solar or wind power plants.

In order to derive a valid charging control decision from a given state, a policy function has to consider information from all parts and finally derive the resulting real-valued charging power for the respective EV. This principle is shown in detail in Fig. 16.

While EV-specific parameters concern the EV's driving behavior and charging demand, including its residence time at the actual charging station or its likelihood

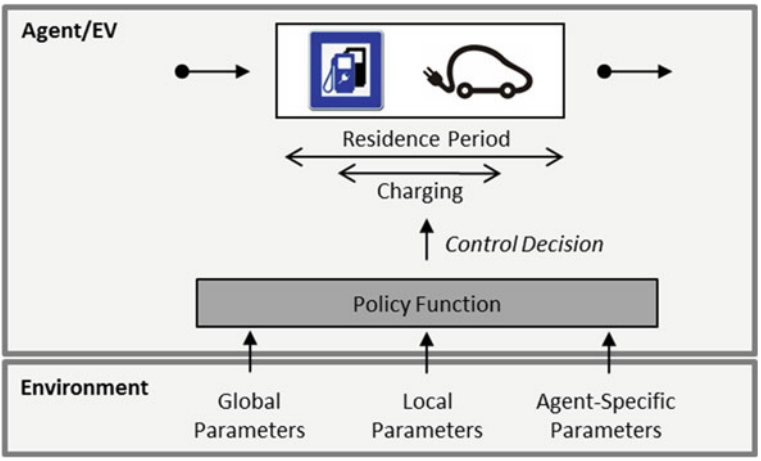


Fig. 16 Principle of policy function based control and simulation

of getting parked at another charging spot later on, local parameters also consider other EVs immediately affecting the local situation in the power grid. For example, if the power grid is stressed locally because of a high amount of EVs charging at the same grid node, their charging power may has to be reduced in the next time step in order to avoid critical power flow conditions. Finally, global parameters consider information describing the entire system's state, such as the total load to the distribution grid, total expected supply provided by renewable sources, and financial aspects considering costs of electrical power supply. Of these information entities, the policy function finally derives the approximate optimal charging decision for a given EV at a defined time step. While the mentioned parameters deliver specific information for each EV, the same policy function can be applied for all EVs in a fleet and still lead to individual decisions.

Optimization

In order to find such policy functions, genetic programming (GP) provides a fruitful method for function approximation that does not need a-priori knowledge on the aspired mathematical function, but only has to know the input variables (parameters as given above) as well as a specific grammar for combining them. Applying a metaheuristic search process (based on a genetic algorithm), GP searches for high-performance policies within a space of analytic functions. Similar to the application of priority rule optimization in the job dispatching example described in the previous section, formula trees are evolved by GP where leafs represent input variables describing the system's state (parameters as given in Fig. 16) that are combined by given mathematical operators incorporated by inner nodes. This kind of solution representation allows the evolution of arbitrary analytic functions without knowing their structure beforehand, which overcomes a severe restriction of existing works on policy function approximation in the literature.

Conclusion

Dynamic optimization with policy functions has the great advantage that it avoids the necessity for computing a specific solution to each state the dynamic system exhibits over time. Instead, an approximate optimal function is optimized offline that takes a system's state and returns control actions online. Unifying this approach with simulation optimization, the treatment of both dynamic and uncertain problems is enabled. Furthermore, the usage of GP for function approximation avoids the need for defining a function's structure beforehand, and thus overcomes a major shortcoming of policy function approximation described in the literature. While we consider here the application of EV charging control as in [23], the same methodology has been applied successfully to applications such as generation unit scheduling in power grids [24].

4 Conclusion

The fruitful combination of simulation and optimization provides mutual benefits for each field. On the one hand simulation engineers are able to improve their models using optimized parameters. On the other hand, optimization experts are able to model systems that are much closer to the real-world. Yet, often the initiating part in simulation-based optimization is the solver with the simulator being merely an evaluation function. However, a growing number of cases emerge where simulation will be used to describe optimization in dynamic environments. These cases are highly interesting from the point of view of optimization as it creates a setup that is much closer to real-world applicability. As in the simulated environment, optimization algorithms in live systems have to deal with changing conditions, uncertainty about the future, and have to make one decision at a time. In the future it will become more and more interesting to study and improve these algorithms using simulated environments.

Interfacing between simulation and optimization has also been a topic that was much discussed. Interprocess communication and different programming languages provide technical difficulties. We have described specific as well as generic interfaces that can be used to overcome these difficulties and allow exchanging candidate solution data as well as a quality feedback. We have motivated how the HeuristicLab architecture is highly suited for these tasks and given more insight into the implementation of these interfaces. The topic of integrated simulation and optimization has also been discussed and is highly relevant for future activities. In several real-world examples we have aimed to describe successful applications that may be interesting and motivating to do further research. These examples can also be seen as guidelines for a generic approach in simulation-based optimization.

Due to the steady increase of available parallel computing resources, the authors are convinced that the simulation-based optimization approach has high potential to model interrelated decision situations, leading again to a more holistic view of production and logistics optimization. The emerging fields *internet of things* and *cyber-physical systems*, which are a matter of recent research in production and

logistics optimization, are expected to benefit from the availability of such enhanced simulation optimization approaches in the future.

Acknowledgments Part of the work described in this chapter were sponsored by the *European Regional Development Fund* and by *Upper Austrian public funds* (within the the Regio 13 program - project 4EMobility), by the *Austrian Research Promotion Agency (FFG)* (within the the Josef Ressel Centre for Heuristic Optimization, the project "NPR" #829679, and the K-project "HOPL" #843532), by the *University of Applied Sciences Upper Austria* (within the basic research program), and by the *seventh framework programme* (within the project BioBoost). HeuristicLab is developed by the Heuristic and Evolutionary Algorithm Laboratory and can be downloaded from the official HeuristicLab homepage http://dev.heuristiclab.com.

References

1. M. Affenzeller and S. Wagner. Offspring selection: A new self-adaptive selection scheme for genetic algorithms. In B. Ribeiro, R. F. Albrecht, A. Dobnikar, D. W. Pearson, and N. C. Steele, editors, *Adaptive and Natural Computing Algorithms*, Springer Computer Series, pages 218–221. Springer, 2005.
2. M. Affenzeller, S. Winkler, S. Wagner, and A. Beham. *Genetic Algorithms and Genetic Programming - Modern Concepts and Practical Applications*. Numerical Insights. CRC Press, 2009.
3. S. Albers. Better bounds for scheduling. *SIAM Journal on Computing*, 29(2):459–473, 1999.
4. W. Banzhaf, P. Nordin, R. Keller, and F. Francone. *Genetic Programming: An Introduction*. Morgan Kaufmann, 1998.
5. A. Beham, M. Kofler, S. Wagner, M. Affenzeller, H. Heiss, and M. Vorderwinkler. Enhanced priority rule synthesis with waiting conditions. In *22nd European Modeling and Simulation Symposium EMSS 2010*, 2010.
6. A. Beham, M. Kofler, S. Wagner, M. Affenzeller, and W. Puchner. Using erp-driven flow analysis to optimize a constrained facility layout problem. In *22nd European Modeling and Simulation Symposium EMSS 2010*, pages 71–76, 2010.
7. A. Beham, G. K. Kronberger, J. Karder, M. Kommenda, A. Scheibenpflug, S. Wagner, and M. Affenzeller. Integrated simulation and optimization in heuristiclab. In *Proceedings of the 26th European Modeling and Simulation Symposium EMSS 2014*, Bordeaux, France, September 2014.
8. A. Beham, E. Pitzer, S. Wagner, M. Affenzeller, K. Altendorfer, T. Felberbauer, and M. Bäck. Integration of flexible interfaces in optimization software frameworks for simulation-based optimization. In *Companion Publication of the 2012 Genetic and Evolutionary Computation Conference, GECCO'12 Companion*, pages 125–132, Philadelphia, PA, USA, July 2012.
9. W. Bell, L. Dalberto, M. Fisher, A. Greenfield, R. Jaikumar, P. Kedia, R. Mack, and P. Prutzman. Improving the distribution of industrial gases with an online computerized routing and scheduling optimizer. *Interfaces*, 13:4–23, 1983.
10. H.-G. Beyer and H.-P. Schwefel. Evolution strategies - A comprehensive introduction. *Natural Computing*, 1(1):3–52, March 2002.
11. Y. Carson and A. Maria. Simulation optimization: methods and applications. In *Proceedings of the 29th conference on Winter simulation*, pages 118–126. IEEE Computer Society, 1997.
12. J.-F. Cordeau and G. Laporte. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579–594, 2003.
13. A. Drira, H. Pierreval, and S. Hajri-Gabouj. Facility layout problems: A survey. *Annual Reviews in Control*, 31(2):255–267, 2007.
14. Eurostat, European Union. Nomenclature of territorial units for statistics.

15. G. Evans. International biofuels strategy project. liquid transport biofuels - technology status report, nnfcc 08-017. Technical report, National Non-Food Crops Centre, 2008.
16. M. Fu, F. Glover, and J. April. Simulation optimization: A review, new developments, and applications. In *Proceedings of the 2005 Winter Simulation Conference*, pages 83–95, 2005.
17. M. C. Fu. Optimization for simulation: Theory vs. practice. *INFORMS J. on Computing*, 14(3):192–215, Summer 2002.
18. M. R. Garey, D. S. Johnson, and R. Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2):117–129, May 1976.
19. F. Glover. Tabu search – part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.
20. A. Gosavi. *Simulation-based optimization: parametric optimization techniques and reinforcement learning*, volume 25. Springer, 2003.
21. N. Hansen. The CMA evolution strategy: a comparing review. In J. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, editors, *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer, 2006.
22. S. Hutterer and M. Affenzeller. Probabilistic electric vehicle charging optimized with genetic algorithms and a two-stage sampling scheme. *International Journal of Energy Optimization and Engineering*, 2:1–15, 2013.
23. S. Hutterer, M. Affenzeller, and F. Auinger. Evolutionary computation enabled controlled charging for e-mobility aggregators. In *Proceedings of the IEEE Symposium Series on Computational Intelligence, Workshop on Computational Intelligence Applications in Smart Grid (IEEE CIASG 2013*, pages 115–121, 2013.
24. S. Hutterer, S. Vonolfen, and M. Affenzeller. Genetic programming enabled evolution of control policies for dynamic stochastic optimal power flow. In *Companion Publication of the 2013 Genetic and Evolutionary Computation Conference*, pages 1529–1536, 2013.
25. O. R. Inderwildi and D. A. King. Quo vadis biofuels? *Energy Environ. Sci.*, 2:343–346, 2009.
26. J. Kennedy and R. C. Eberhardt. Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE Press, 1995.
27. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.
28. M. Kofler, A. Beham, S. Vonolfen, S. Wagner, and M. Affenzeller. Modelling and optimizing storage assignment in a steel slab yard. In *Proceedings of the 4th IEEE International Symposium on Logistics and Industrial Informatics (LINDI 2013)*, pages 101–106, Smolenice, Slovakia, September 2012.
29. J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.
30. A. M. Law. *Simulation Modeling and Analysis*. McGraw-Hill, 2007.
31. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 3rd edition, 1999.
32. N. Moin and S. Salhi. Inventory routing problems: a logistical overview. *Journal of the Operational Research Society*, 58:1185–1194, 2007.
33. J. Momoh. *Electric Power System Applications of Optimization*. CRC / Taylor & Francis, 2009.
34. S. S. Panwalkar and W. Iskander. A survey of scheduling rules. *Operations Research*, 25(1):45–61, Jan-Feb 1977.
35. J. Parejo, A. Ruiz-Cortés, S. Lozano, and P. Fernandez. Metaheuristic optimization frameworks: a survey and benchmarking. *Soft Computing*, 16(3):527–561, 2012.
36. V. Pillac, C. Guéret, and A. L. Medaglia. An event-driven optimization framework for dynamic vehicle routing. *Decision Support Systems*, 2012.
37. M. Pinedo. *Scheduling: Theory, Algorithms and Systems*. Prentice-Hall, 1995.
38. E. Pitzer, A. Beham, M. Affenzeller, H. Heiss, and M. Vorderwinkler. Production fine planning using a solution archive of priority rules. In *Proceedings of the IEEE 3rd International Symposium on Logistics and Industrial Informatics (Lindi 2011)*, pages 111–116, Budapest, Hungary, August 2011.

39. i. Rawles. The WITNESS toolbox - A tutorial. In D. Medeiros, E. Watson, J. Carson, and M. Manivannan, editors, *Proceedings of the 1998 Winter Simulation Conference*, pages 223–226, 1998.
40. I. Rechenberg. *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, 1973.
41. D. Sadowski and V. Bapat. The Arena product family: Enterprise modeling solutions. In P. Farrington, H. Nembhard, D. Sturrock, and G. Evans, editors, *Proceedings of the 1999 Winter Simulation Conference*, pages 159–166, 1999.
42. E. Sortomme, M. M. Hindi, S. D. J. McPherson, and M. Venkata. Coordinated charging of plug-in hybrid electric vehicles to minimize distribution system losses. *IEEE Transactions on Smart Grid*, 2:198–205, 2011.
43. L. Tang, J. Liu, A. Rong, and Z. Yang. A review of planning and scheduling systems and methods for integrated steel production. *European Journal of Operational Research*, 133(1):1–20, 2001.
44. L. Tang, J. Liu, A. Rong, and Z. Yang. Modelling and a genetic algorithm solution for the slab stack shuffling problem when implementing steel rolling schedules. *International Journal of Production Research*, 40(7):1583–1595, 2002.
45. E. Tekin and I. Sabuncuoglu. Simulation optimization: A comprehensive review on theory and applications. *IIE Transactions*, 36(11):1067–1081, 2004.
46. G. K. Venayagamoorthy. Dynamic, stochastic, computational, and scalable technologies for smart grids. *IEEE Computational Intelligence Magazine*, 6:22–35, 2011.
47. J. G. Vlachogiannis. Probabilistic constrained load flow considering integration of wind power generation and electric vehicles. *IEEE Transactions on Power Systems*, 24:1808–1817, 2009.
48. S. Vonolfen, M. Affenzeller, A. Beham, E. Lengauer, and S. Wagner. Simulation-based evolution of resupply and routing policies in rich vendor-managed inventory scenarios. *Central European Journal of Operations Research*, 21(2):379–400, March 2013.
49. S. Vonolfen, M. Affenzeller, A. Beham, S. Wagner, and E. Lengauer. Simulation-based evolution of municipal glass-waste collection strategies utilizing electric trucks. In *Proceedings of the IEEE 3rd International Symposium on Logistics and Industrial Informatics (Lindi 2011)*, pages 177–182, August 2011.
50. S. Vonolfen, A. Beham, M. Kofler, M. Affenzeller, and K. Dörner. Simulation-based optimization of transport activities within cold charge steel production. In *Proceedings of the 5th IEEE International Symposium on Logistics and Industrial Informatics (LINDI 2013)*, pages 67–73, Wildau, Germany, September 2013.
51. S. Vonolfen, M. Kofler, A. Beham, M. Affenzeller, and W. Achleitner. Optimizing assembly line supply by integrating warehouse picking and forklift routing using simulation. In *Proceedings of the Winter Simulation Conference*, page 339. Winter Simulation Conference, 2012.
52. S. Wagner. *Heuristic Optimization Software Systems - Modeling of Heuristic Optimization Algorithms in the HeuristicLab Software Environment*. PhD thesis, Johannes Kepler University, Linz, Austria, 2009.
53. S. Wagner, G. Kronberger, A. Beham, M. Kommenda, A. Scheibenpflug, E. Pitzer, S. Vonolfen, M. Kofler, S. Winkler, V. Dorfer, and M. Affenzeller. *Advanced Methods and Applications in Computational Intelligence*, volume 6 of *Topics in Intelligent Engineering and Informatics*, chapter Architecture and Design of the HeuristicLab Optimization Environment, pages 197–261. Springer, 2014.
54. S. Wagner, G. Kronberger, A. Beham, S. Winkler, and M. Affenzeller. Modeling of heuristic optimization algorithms. In *Proceedings of the 20th European Modeling and Simulation Symposium*, pages 106–111. DIPTEM University of Genova, 2008.
55. S. Wagner, S. Winkler, R. Braune, G. Kronberger, A. Beham, and M. Affenzeller. Benefits of plugin-based heuristic optimization software systems. In R. Moreno-Diaz, F. Pichler, and A. Quesada-Arencibia, editors, *Computer Aided Systems Theory - EUROCAST 2007*, volume 4739 of *Lecture Notes in Computer Science*, pages 747–754. Springer, 2007.
56. M. Waller, M. Johnson, and T. Davis. Vendor-management inventory in the retail supply chain. *Journal of Business Logistics*, 20:181–203, 1999.
57. D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

Simulation Optimization Approach to Solve a Complex Multi-objective Redundancy Allocation Problem

Carlos Henrique Mariano and Carlo Alessandro Zanetti Pece

Abstract This chapter addresses the problem of redundancy and reliability allocation in the operational dimensioning of an automated production system. The aim of this research is to improve the global reliability of the system by allocating alternative components (redundancies) that are associated in parallel with each original component. By considering a complex componential approach that simultaneously evaluates the interrelations among subsystems, conflicting goals, and variables of different natures, a solution for the problem is proposed through a multi-objective formulation that joins a multi-objective elitist genetic algorithm with a high-level simulation environment also known as simulation optimization (SIMO) framework.

1 Introduction

The simulation/optimization framework (SIMO) is an iterative and stochastic technique for generating multiple optimization scenarios where the optimization process occurs simultaneously with system simulation analysis. In general, there are two approaches to SIMO framework. On the first the simulation process is used as a validation tool or test for the effectiveness of any optimization method as seen in [9, 32, 46]. The second presents an optimization process through the simulation applied to the resolution of a complex problem as we see in the texts [22, 36, 44].

The whole process can be summarized in two questions. What happens if? and How do I get? The first is applicable to the analysis of multiple simulation possibilities or scenarios. The second is applicable to optimization analysis, where we can maximize or minimize important criteria or objectives to streamline the effectiveness of the system [30].

C.H. Mariano (✉)
Department of Electrical Engineering - DAELT, Federal Technological University of Paraná - UTFPR, Curitiba, PR, Brazil
e-mail: mariano@utfpr.edu.br

C.A.Z. Pece
Department of Electrical Engineering - DAELT - Postgraduate Program in Biomedical Engineering, Federal Technological University of Paraná - UTFPR, Curitiba, PR, Brazil

M. Mujica Mota et al. (eds.), *Applied Simulation and Optimization*,
DOI 10.1007/978-3-319-15033-8_2

2 Considerations About the Process

In this chapter, the idealized case under consideration assumes the existence of an automated system characterized by an operational scenario (e.g., machine configuration, maintainers, costs, and individual reliabilities). To obtain an optimal scenario by considering the inherent analytical difficulties, random factors, and conflicting multiple objectives, a simulative process combined with an optimization process is proposed.

This process makes intensive use of computational power and can be defined as an iterative stochastic technique also known as simulation optimization framework. The technique generates multiple operational scenarios and those that best meet the problem objectives and constraints are chosen. A generation of new operational scenarios is evaluated after each iteration and, after many iterations, a set of scenarios and solutions can be selected from many possible combinations. The evaluation process starts at the simulator. In the idealized automated system—a discrete complex system—the following relationships are observed:

(1) Nonlinear relationships;
(2) Relationships with feedback loops;
(3) Interchanged relationships (input/output) with the environment;
(4) Past-state dependent relationship;
(5) Concatenated relationships; and
(6) Relationships consisting of different types of variables with different natures.

The modeling of such a system requires an appropriate simulation environment. Among the several options available, the ARENA commercial simulation software was chosen because it is oriented to discrete events. The present study adopted a variant of the simulation optimization process, where the simulator is the core of the process [1]. Due to the high dimensional aspect and complexity of the system, it is virtually impossible to represent it as a single optimization function (conventional case). Therefore, the simulation model enables representation of complex system relationships.

In this variant of the simulation optimization process, system performance is tested by using indicators and variables captured from the simulation model [1, 37]. Thus, the simulator produces inputs required by the optimizer to guide the selection of the operational scenarios (solutions) that best fit the objectives and constraints of the problem.

The term “optimization”, which is commonly used in the computer science field, is used broadly in this chapter. Optimization can be defined as an iterative process of global search, where the optimum is approximated through stochastic processes.

The elitist multi-objective genetic algorithm (MOGA) that was chosen for the present study is the NSGA-II (Nondominated Sorting Genetic Algorithm II), whose ordination principle is based on the notion of Pareto dominance [6, 7]. In this algorithm, the simulation optimization process is completed once a set of optimal Pareto solutions is obtained.

In addition to the optimal Pareto set, a procedure to aggregate quantitative and qualitative information is necessary to determine the best solution. The qualitative information usually reflects the opinion of the specialists involved in system operation and maintenance. The quantitative information is obtained from technical and administrative specifications [5].

3 Some Aspects of Simulation Model

It is worth highlighting the stochastic nature of the simulation model. This characteristic is the basis of the main argument of this study, particularly the unlikelihood of building a single function that represents all of the stochastic system relationships. Therefore, this study proposes a different approach. Significant random aspects of the model, such as the failure and repair process, the loss of machine production speeds after a certain number of failures have occurred, and the processing of lines produced in the machines, are ruled by several variables:

(1) Machine operating time;
(2) Waiting time for a part;
(3) Waiting time for another operation;
(4) Machine failure time; and
(5) Machine repair time.

The events related with failure and repairs of each machine or robot were exponentially distributed with different means. Exponential distributions feature risk functions with constant failure rates throughout the life span of the system. Therefore, they are frequently used to represent failure or repair of complex systems. Different times for scheduled maintenance inspection (preventive) were assigned to each machine. Such times depend on the particular type of inspection, which is determined by a certain number of parts produced.

The topology and assignment of values in the simulation model are extremely dependent on the experience of the modeler. Consequently, a detailed description of the simulation model used in this study would be lengthy and beyond the scope and purpose of this paper.

The automatized system adopted and represented by the idealized case under consideration is discrete. The approach for this type of system requires specific languages and modeling tools. In the present study, Petri networks were adopted to specify the conceptual model of the production process.

The Petri conceptual model facilitated the planning of all involved operations and clearly established system transitions and operations. The model also allowed visual and facilitated changes to be made in the simulation model and defined hierarchies in the sequences of operations.

The advantage of this method, as compared with other existing methods, is that its formalism is based on a simplified graphic model with few syntactic rules [28, 31, 33]. Another interesting characteristic of the Petri model is that its logical development

resembles the block-oriented programming logic of current simulation environments such as ARENA, PROMODEL, and SIMUL8.

It is worth highlighting the stochastic nature of the simulation model. This characteristic is the basis of the main argument of this study, particularly the unlikelihood of building a single function that represents all of the stochastic system relationships. Therefore, this paper proposes a different approach. Significant random aspects of the model, such as the failure and repair process, the loss of machine production speeds after a certain number of failures have occurred, and the processing of lines produced in the machines, are ruled by several variables:

- Machine operating time;
- Waiting time for a part;
- Waiting time for another operation;
- Machine failure time; and
- Machine repair time.

The events related with failure and repairs of each machine or robot were exponentially distributed with different means. Exponential distributions feature risk functions with constant failure rates throughout the life span of the system. Therefore, they are frequently used to represent failure or repair of complex systems. Different times for scheduled maintenance inspection (preventive) were assigned to each machine. Such times depend on the particular type of inspection, which is determined by a certain number of parts produced.

The topology and assignment of values in the simulation model are extremely dependent on the experience of the modeler. Consequently, a detailed description of the simulation model used in this study would be lengthy and beyond the scope and purpose of this chapter.

4 Multi-objective Optimization Models

A multi-objective optimization model strives to minimize or maximize a group of functions that are usually in mutual conflict. The existence of multiple objective functions suggests a fundamental difference between multi-objective and mono-objective optimizations. Therefore, there is no one single solution to the problem, but a cluster of solutions that invoke different compromises among the values of the functions to be optimized. Within a set of efficient solutions, the desired solution is one that exhibits the best compromise for the objectives (e.g., Pareto-optimal and nondominated solutions). The identification of the best compromise solution requires that the decision-maker establish a preference because many objectives are not only in conflict with each other but also represent different phenomena described by deterministic and stochastic mathematical functions.

The classic multi-objective optimization methods, according to [8] and [5], can be divided into two categories: complete enumeration methods and preference-based methods. The former category presents a full group of nondominated solutions to

the decision-maker, which allows him to choose the solution that best suits his goals. The latter category is based on choosing a solution that offers the best compromise from a group of nondominated solutions, through the use of an explicit or implicit criterion. Such compromises indicate that the solution must simultaneously satisfy the analytical functions of the objectives and the degrees of preference established by the decision-maker. Deb [5] presents another classification that also divides the methods into two categories: preference and nonpreference. Deb [5] expands the preference-based methods into three subcategories: posteriori selection methods, priori methods, and interactive methods. The following is a list of distinct preference-based methods:

a. Weighted sum method;
b. Utility function method;
c. Constraint method—ϵ (where ϵ is the upper limit of the scenario function inserted as a constraint to the problem);
d. Weighted metrics method;
e. Benson's method;
f. Value function method;
g. Goal programming method;
h. Pareto method; and
i. Lexicographic method.

The natural complexity inherent to multi-objective optimization problems poses a challenge to exact algorithms.[1] For this reason, evolutionary algorithms are becoming increasingly popular as robust and effective methods to solve single- and multi-objective optimization problems [4].

The operation of Evolutionary Algorithms (EA) is based on the simulation of natural evolution [11]. The EA use an iterative technique that applies stochastic operators to a group of individuals (population) to improve their levels of adaptation to the problem, performance, or fitness. In most applications, this measure is related to the objective functions of the problem being considered.

Multi-objective evolutionary algorithms (MOEA) are capable of treating multiple objectives naturally because they operate naturally in parallel on the group of solutions and develop a set of solutions similar to the Pareto boundary or frontier at each execution [5]. Such characteristics allow MOEA to approach problems with a large solution space.

According to Nesmachnow [29], MOEA exhibit two special operators that do not appear in the generic structure of EAs: a diversity operator and a fitness attribution operator. The diversity operator represents a technique used to avoid premature convergence to a sector of the Pareto boundary (e.g., niche, fitness, sharing, and crowding) and assess diversity. The fitness attribution operator is aimed at ensuring the permanence of individuals with the best characteristics for future generations by considering the values of the objective functions and the results of the metrics.

[1] Enumerative local search algorithms based on gradients or that use standard techniques of deterministic programming, such as greedy algorithms, or branch and bound techniques [14, 29].

For these reasons above, the MOEA are very suitable to apply in the majority of actual optimization problems in engineering. In recent decades multi-objective optimization models have gained a crescent acceptance for multiple applications in various areas of engineering, Coello and Lamont [4], Zio and Zille [48], Çunkas [25], Zhuo et al. [47], Tian et al. [42], Tzu-Chieh and Kuei-Yuan [43]. The difficulty in these cases lies in representing the complex interrelations contained in such systems, as reported by the following authors below, which are better defined on high-level simulation environment:

Mattila and Virtanen [23]—Airplane maintenance workshop—To maximize the availability of airplanes to ensure fleet operation capability, while minimizing the disparity between scheduled and nonscheduled maintenance operations.

Merkuryeva and Napalkowa [26]—Supply/logistics chain—To minimize total average cost corresponding to the total cost of storage, production, and procurement and maximize customer service satisfaction.

Hani et al. [10]—Train maintenance workshop—To maximize the production rate represented by the number of vehicles that leave the workshop per year, minimize the waiting time of vehicles, and the occupation rate of maintainers in the different groups.

5 Reliability Optimization—Redundancy Allocation Problem to Improve System Reliability—A Review

Analysis of the reliability of engineering systems, particularly industrial systems, can be performed by two ways: by the classical/binary approach, where, according to [2], a system subjected to failure can assume only two different states, total standstill or fully functional. And the multistate approach, where, according to [21], a system might assume several states between fully functional and total failure.

The multistate approach better describes actual systems because they are subjected to a series of factors that diminish their life span [21]. The occurrence of failure in such systems may not necessarily cause complete standstill of the activity despite the increasing tendency for such standstill.

Whether the approach adopted for the analysis and optimization of the reliability of a system is binary or multistate, the optimization techniques are typically classified by the following four methods [17, 18]:

1. Redundancy allocation (RAP)—where the decision variables represent the number of redundancies;
2. Reliability allocation—in this case, the decision variables are the reliabilities of the components or subsystems;
3. Redundancy/reliability allocation—the decision variables are a combination of the number of redundancies and reliabilities of the components or subsystems; and
4. Components or subsystems allocation—where the system configuration (component arrangement) represents the decision variables.

Redundancy allocation problems (RAP) are problems related to the search for specific combinations of alternative components that, when properly associated with the components whose reliabilities needs to be improved, enhance the global reliability of the system. Their nature is eminently combinatorial; they can be considered to be nonlinear mixed integer programming problems or a special case of integer programming, and they can be solved by the traditional cut or search methods, or by a combination of both, as presented by Jianping and Xishen [12], Misra and Sharma [27]. The problems are usually solved by mono- or multi-objective formulations and, more recently, exploration of multilevel and multistate RAP has been observed, as presented by [16, 41, 45]. Consequently, these problems have become a focus of many studies, which adds an even greater degree of complexity to traditional problems.

The use of RAP-solving methods such as algorithm-hybridization processes, which combine heuristic methods, neuronal networks, fuzzy techniques, and local search methods with metaheuristic types, are increasing. The processes are utilized alone to improve computational efficiency or with exact methods to reduce the search space. Their use creates the potential for combining two metaheuristics, annealing-genetic (AG) algorithms and simulated annealing, as indicated by [3, 13, 15, 18, 19, 24, 35, 39].

EA exhibit interesting characteristics that enable them to manage RAP in mono-objective, multi-objective, and multilevel formulations. They can control noncontinuous, nonconvex, and/or nonlinear spaces, as well as unknown objective functions. Although genetic algorithms are frequently used, current RAP-solving is moving toward the application of EA that differ from genetic algorithms, as presented by Salazar et al. [34], Taboada and Coit [38], Lins and Droguett [20], Taboada et al. [40].

To test the algorithms applied to multi-objective RAP-solving, a general formulation is utilized that maximizes the reliability of the system, minimizes costs associated with the allocation of components, minimizes the weight and volume of the system, and it is subjected to constraints in cost, volume, weight, number of components to be allocated, and system reliability limits. Mathematically, this formulation is depicted in Fig. 1
where

$\boldsymbol{m}$—number of subsystems or stages
$\boldsymbol{i}$—subsystem index, $i = 1, 2, \ldots, m$
$\boldsymbol{j}$—index of the components of each subsystem, $j = 1, 2, \ldots, n$
$\boldsymbol{r}_{ij}$—reliability of component j in subsystem i
$\boldsymbol{c}_{ij}$—cost of component j in subsystem i
$\boldsymbol{w}_{ij}$—weight of component j in subsystem i
$\boldsymbol{R}_s$—total reliability of the parallel-series system
$\boldsymbol{C}_s$—total cost of the parallel-series system
$\boldsymbol{W}_s$—total weight of the parallel-series system
$\boldsymbol{C}_o$—allowed system cost
$\boldsymbol{W}_o$—allowed system weight
$\mathbf{a}_i$—number of component choices available for subsystem i

Fig. 1 RAP—general multi-objective formulation

$$Max \quad R_s = \prod_{i=1}^{m} \left(1 - \prod_{j=1}^{a_i} (1 - r_{ij})^{xij}\right)$$

$$Min \quad C_s = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$Min \quad W_s = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} x_{ij}$$

$$s.t. \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \leq C_o$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} x_{ij} \leq W_o$$

$$\sum_{j=1}^{a_i} x_{ij} \leq n_{\max}, \quad i = 1,2,\ldots,m$$

$$\sum_{j=1}^{a_i} x_{ij} \geq n_{\min}, \quad i = 1,2,\ldots,m$$

$$x_{ij} \in Z^+, \quad i = 1,2,\ldots,m; \quad j = 1,2,\ldots,n$$

$\mathbf{x}_{ij}$—amount of component j used in subsystem i
$\boldsymbol{n}_i$—total number of components that might be in parallel
$\boldsymbol{n}_{max}$—maximum number of components that can be in parallel
$\boldsymbol{n}_{min}$—minimum number of components that can be in parallel.

Thus, RAP can be classified according to the type of redundancy, such as cold standby, warm standby, hot standby, parallel redundancy, or active redundancy. The type of components to be allocated can be classified as identical or nonidentical components; however, nonidentical components are more complex and similar to the actual systems. The redundancy levels of items can include the component redundancy level (individual component level), modular redundancy level (subsystem level), or system redundancy level, which produces a multilevel RAP. Based on its functioning status, a component can be classified as binary, where the component might assume one of two possible states (e.g., zero for total failure or one for fully functional), or multistate, where the component might assume multiple states between total failure and total functioning. The RAP can also be classified according to the type of system and whether its nature features repairable or nonrepairable components.

In the present study, classification was performed with either a basic componential attribute, where all components were described in terms of three attributes: weight, volume, and individual reliability, or a complex componential attribute including other attributes such as failure and repair distributions, inspections, and process speeds. The last feature increases the degree of complexity of the problem in terms of computational effort because the number of possible combinations increases.

This chapter considers RAP as linked with subsystems characterized by attributes not typically found in combinatory problem solving instead of simple components.

In basic and complex componential approaches, a component with an adequate set of attributes is required that, when combined with the remaining allocated components, will result in the total final reliability of the system. The total final reliability value depends on the structure of the system. A review of current research indicates that analysis in this area is primarily focused on a classical parallel–series structure. The present study approached the analysis similar to that of [18], whose general statement is represented by the following equations:

$R_i = -\prod_{j=1}^{n_i}(1 - r_{ij})$ and the system reliability is defined by $R_s = \prod_{i=1}^{k} R_i$ where

$\boldsymbol{R_i}$—reliability of subsystem i
$\boldsymbol{r_{ij}}$—reliability of a component j, $1 \leq j \leq n_i$ of subsystem i
$\boldsymbol{n_i}$—components in parallel
$\boldsymbol{k}$—subsystems in series.

Joining both expressions yields the following equation:

$$R_s = \prod_{i=1}^{k} \left[1 - \prod_{j=1}^{n_i}(1 - r_{ij})\right].$$

The reliability value only acquires its full meaning when it is associated with a definite timepoint; however, a literature review reveals that this aspect is seldom addressed. This chapter is rooted in this context and adds an appropriation of time to the complex componential approach by considering that the values of reliability are measured in time. The component is replaced by a subsystem whose main attributes are not only related to weight, volume, and individual reliability but also to time.

Conversely, the component is now a machine that performs a certain process within a production line characterized by the following attributes:

(a) operation speed;
(b) repair time;
(c) production cost;
(d) individual reliability; and
(e) production capacity.

The present study considers realistic problems of superior complexity. The modeling of such problems considers different types of variables (real, binary, stochastic, and nonstochastic) concomitantly as more than one objective is optimized. Given the high dimension and complexity of the system and the impracticality of representing the system analytically, a simulative and evolutionary optimizing process combined with remarkable characteristics was selected to solve this type of problem [7].

To dimension a group of machines in an idealized automated production line, the process was applied to the project stage as a function of its inherent advantages and the control of the conjunctural variables of the system (e.g., production standstill due to strikes and lack of raw materials). The expected practical objective is to demonstrate the feasibility of the process rather than to describe an empirical implementation example. The result is a pragmatic method that can be applied to actual manufacturing, especially in automated environments.

6 Formulation of the Variables, Objectives, and Constraints of the Multi-objective Model

The variables of the simulation optimization process can be classified as both exogenous and endogenous to the main code, i.e., the MOGA. The exogenous variables are processed inside the simulator and based on a given operational scenario. The elements of each scenario (e.g., configuration, maintainers, costs, and reliabilities) are represented in the model by a set of endogenous variables that form a chromosome or individual of a specific age.

The objectives of optimization are calculated from exogenous and endogenous variables. In the idealized case under consideration, four conflicting objectives, which must be optimized concurrently, are used:

(1) *TCSM*—Total cost of system maintenance (minimization)—exogenous variable;
(2) R_s—Total reliability of the system (maximization)—exogenous variable;
(3) *TOC*—Total cost of system operation (minimization)—exogenous variable; and
(4) *Numan*—Number of maintainers (minimization)—endogenous variable.

Certain variables are either part of the optimization objectives or represent the objectives themselves, as is the case of TMFC (total maintenance fixed cost) and numan (number of maintainers), respectively. It must still be emphasized that variable $\boldsymbol{a}_{ij}$ represents the allocation of a machine/robot i in stage j of the system. Such a

variable plays a crucial role in the production variability of the operational scenarios and can be presented as follows:

$$\boldsymbol{a}_{ij}$$

$$\begin{cases} 0, \textit{ for non allocated machine/robot} \\ 1, \textit{ for allocated machine/robot} \end{cases}$$

Thus, the functioning of the full process is based on the correct definition of the constraint limits. This task is highly dependent on the degree of familiarity of the analyst with the system; without such knowledge appropriate limits would not be imposed on the process. In other words, the limits represent the horizons of the solution spaces and the problem variables that guide the search process in the desired direction. So, below is the list of variables used on the models:

(1) ($cmttpm_{ij}$)—*cost of materials, tools, and outsourced manpower applied to the maintenance of machine i in subsystem j*
(2) ($cinst_{ij}$)—*cost of allocation of redundancy maintenance in machine i in subsystem j*
(3) ($cplf_{ij}$)—*cost of production loss due to failure of i in subsystem j*
(4) *General management cost (GMC) corresponding to 10% of the sum of costs* $cmttpm_{ij}$,$cinst_{ij}$ *and* $cplf_{ij}$. $\left[\sum_{i=1}^{k}\sum_{j=1}^{n} a_{ij} * (cinst_{ij} + cmttpm_{ij} + cplf_{ij})\right]$
(5) dtm_{ij}—*total standstill time of machine i in subsystem j*
(6) $TTPS_{ij}$—*total standstill time of machine i in subsystem j,*
(7) ot_{ij}—*operation time of machine i in subsystem j*
(8) $csmm_{ij}$—*cost of specialized manpower to maintain machine i in subsystem j.*
(9) *crama—cost of applied raw material*
(10) *cmpol—cost of manpower for operating the line*
(11) *ceeoc—cost of electricity and other fuel*
(12) *cpsat—cost of packaging, storing, and transport*
(13) *TPPS—total number of products produced by the system* (*exogenous variable*)
(14) *TNSF—total number of system failures*
(15) *TSA—total system availability*
(16) *SDT—system downtime*
(17) *TGPPS—total of good products produced by the system*
(18) *TMFC—total fixed costs of maintenance*
(19) *CMTTPM—total cost of maintenance tools, parts, and manpower of the machines* (*exogenous variable*)
$CMTTPM = \sum_{i=1}^{n}\sum_{j=1}^{m} cmttpm_{ij} * a_{ij} \leq UB$
(20) *CINST—total cost of machine installation* (*exogenous variable*)

$$CINST = \sum_{i=1}^{n}\sum_{j=1}^{m} cinst_{ij} * a_{ij} \leq UB$$

(21) *CCPLF—total cost of production loss by the machines due to standstill caused by failure* (*exogenous variable*)

$$CPLF = \sum_{i=1}^{n}\sum_{j=1}^{m} cplf_{ij} * a_{ij} \leq UB$$

(22) *cost of production loss by the machine i in subsystem j due to standstill caused by failure*

$$cpfl_{ij} = \left(ocm_{ij} \times \left(\frac{ppm_{ij}}{eppm_{ij}} \right) \times rt_{ij} \right) * a_{ij}$$

(23) ocm_{ij}*—operation cost of machine i in subsystem j*
(24) ppm_{ij}*—total products produced by machine i in subsystem j*
(25) $eppm_{ij}$*—expected total of products produced without occurrence of failures of machine i in subsystem j*
(26) rt_{ij}*—repair time of machine i in subsystem j.*
(27) *total General Maintenance Cost, GMC, corresponding to 10% of the sum of costs* $cmttp_{mij}$, $cins_{tij}$, *and* csp_{lij}. (*exogenous variable*)

$$GMC = \sum_{i=1}^{n}\sum_{j=1}^{m} (cmttp_{ij} + cinst_{ij} + cplf_{ij}) * a_{ij} \leq UB$$

The complete multi-objective model is shown as

$$\textbf{\textit{MIN TCSM}} = \left[\frac{\left[\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij} * \left[(numan * e^{(\frac{1}{70} * TTPS_{ij})} + (ot_{ij} + dtm_{ij})) * csmm_{ij} \right] * a_{ij} \right] + (\sum_{i}^{n}\sum_{j}^{m} CPLF_{ij} * a_{ij}) + 1.1 * (\sum_{i=1}^{k}\sum_{j=1}^{n} a_{ij} * (cinst_{ij} + cmttpm_{ij} + cplf_{ij}))}{TSA} + TMFC \right]$$

$$\textbf{\textit{MIN TOC}} = TPPS * 0,05 * crama + TPPS * 0,15 * cmpol + TPPS * 0,10 * ceeoc + TPPS * 0,06 * cpsat + TMVC$$

$$\textbf{\textit{MAX}}\ \boldsymbol{R_s} = \prod_{i=1}^{k} \left[1 - \prod_{j=1}^{n_j} (1 - r_{ij} * a_{ij}) \right]$$

where:

$$CMTTPM = \sum_{i=1}^{n}\sum_{j=1}^{m} cmttpm_{ij} * a_{ij} \leq UB;$$

$$CINST = \sum_{i=1}^{n}\sum_{j=1}^{m} cinst_{ij} * a_{ij} \leq UB;$$

$\sum_{i=1}^{n}\sum_{j=1}^{m} cplf_{ij} * a_{ij} \leq UB$

$CGM = \sum_{i=1}^{n}\sum_{j=1}^{m}(cmttp_{ij} + cinst_{ij} + cplf_{ij}) * a_{ij} \leq UB;$

$LB \leq R_s \leq UB; LB \leq TOC \leq UB; LB \leq TMC \leq UB; LB \leq numan \leq UB;$
$LB \leq TMFC \leq UB$

$LB \leq SDT \leq UB; LB \leq TNSF \leq UB; LB \leq TSA \leq UB.$

LB—lower bound—UB—Upper bound

6.1 Declaration of the Nonlinear, Nonstochastic, and Mono-Objective Test Model

In addition to the multi-objective problem, a mono-objective model was included for the purpose of comparison to the full model, which was constructed to reduce its complexity by removing several stochastic features present in the full model.

Thus, this model was developed on the grounds of ϵ-*constraint methods or approach*, according to [5]. Three of the main objectives were transformed into constraints to maximize the system reliability through allocation of redundancies. The costs related to the use of tools, parts, and manpower in the maintenance of machine j in subsystem I ($cmttpm_{ij}$); with the installation of a machine j in subsystem i ($cinst_{ij}$), and with the loss of production of machine j in subsystem i due to standstill caused by failure ($cppm_{ij}$) were predefined for each machine, thus forming a fixed cost matrix.

The amount of products produced by the system per period of system operation (720, 1,440, and 2,160 h) was calculated and supplied. Events related to failure or repair time of the system were not considered. A total system availability (TSA) of 0.90 was established for each simulation period. This model ensures that at least one machine will be allocated to each subsystem. Constant *A* is a fixed value in each simulation period, corresponding to 3,973.99, 7,959.12, and 11,989.40 currency units (CU) for 720, 1,440, and 2,160 h, respectively. The model was encoded in A Mathematical Programming Language (AMPL) and the KNITRO 6.0 solver was used. A total of forty-five binary and five integer variables with thirteen linear constraints was used.

The maximized model is shown as

$$\boldsymbol{Max}\ R_s = \prod_{i=1}^{k}\left[1 - \prod_{j=1}^{n_j}(1 - r_{ij} * a_{ij})\right]$$

where

$$\boldsymbol{TCSM} = \left[\frac{\boldsymbol{numan*BESETTINGA} + \boldsymbol{1.1}*(\sum_{i=1}^{k}\sum_{j=1}^{n} a_{ij}*(\boldsymbol{cinst_{ij}} + \boldsymbol{cmttpm_{ij}} + \boldsymbol{cplf_{ij}}))}{\boldsymbol{TSA}} + \boldsymbol{TMFC}\right] \leq \boldsymbol{3500 \times 10^3}$$

$$\boldsymbol{TOC} = TPPS * 0,05 * crama + TPPS * 0,15 * cmpol + TPPS * 0,10 * ceeoc + TPPS * 0,06 * cpsat + TMVC \leq \mathbf{8000 * 10^3}$$

$$\sum_{i=1}^{k} \sum_{j=1}^{n} a_{ij} \leq n$$

$$CPPROS = \sum_{i}^{k} \sum_{j}^{n} (cppm_{ij} * a_{ij}) \leq 900 \times 10^3 \text{ Currency units}$$

$$CINST = \sum_{i=1}^{k} \sum_{j=1}^{n} cinst_{ij} * a_{ij} \leq 900 \times 10^3 \text{ Currency units}$$

$$CPPM = \sum_{i=1}^{k} \sum_{j=1}^{n} cppm_{ij} * a_{ij} \leq 10 \times 10^3 \text{ Currency units}$$

$$CGM = \sum_{i=1}^{k} \sum_{j=1}^{n} a_{ij} * (cinst_{ij} + cmttpm_{ij} + cppm_{ij}) \leq 260 \times 10^3 \text{ Currency units}$$

$$5 \leq \boldsymbol{numan} \leq 20$$
$$0.1 \leq \boldsymbol{crama} \leq 1.0$$
$$1.0 \leq \boldsymbol{cmpol} \leq 3.0$$
$$1.0 \leq \boldsymbol{ceeoc} \leq 7.0$$
$$1.0 \leq \boldsymbol{cpsat} \leq 5.0$$
$$250 \times 10^3 \leq TMFC \leq 350 \times 10^3 \text{ Currency units (CU)}$$

7 Process of Interaction Between Simulator and Optimizer

Unlike the traditional optimization process, which is based on the construction of objective functions by [17, 18], the optimization process described in this chapter does not employ such functions in the usual manner. The full process is represented by a set of variables defined in both MOGA and the simulator. Certain variables are incorporated in the constraints whereas other variables become the objectives of optimization.

Evolution is guided by a strategy that selects the individuals best adapted for the solution of the problem and best satisfies the constraints and objectives of the problem. The MOGA is responsible for producing variability in the operational scenarios, which is performed automatically with crossover, mutation, and selection operators.

Each operational scenario, or each MOGA individual, is randomly generated and consists of the following endogenous variables:

(1) Number of maintainers (number of persons devoted to tasks related to system maintenance)—$numan \in \mathbb{Z}$;
(2) Allocation of machines or robots—binary variable a_{ij};

(3) Expected reliability of each machine, regardless of whether it is allocated or nonallocated in the system—real variable between 0 and 1;
(4) The four components of the total cost of system operation—real variables *crama, cmpol, ceeoc, and cpsat*; and
(5) TMFC—real variable *TMFC*.

Thus, the operational scenario for MOGA is a chromosome whose encoding represents a list of vectors containing:

(1) Forty-five binary variables for the allocation of machines: a_{ij}, *for* $i = 1, \ldots, n$ *components and* $j = 1, \ldots, k$ *stages or subsystems*, which can assume a value of 0 or 1 and that correspond to the nonallocated and allocated machines, respectively.
(2) Forty-five individual reliabilities, real variables that represent, r_{ij} *for* $i = 1, \ldots, n$ *components*, and $j = 1, \ldots, k$ *stages or subsystems*;
(3) Four real variables, c_p, $p = 1, \ldots, z$ basic production costs;
(4) One real variable *numan* that represents the number of the system maintainers;
(5) One real variable representing the *TMFC* of the system for a given operational scenario.

Thus, a vector with a 96-variable longitude is obtained a follows:

$$\upsilon = \boldsymbol{a}_{ij}, \ldots, \boldsymbol{a}_{nk}/\boldsymbol{r}_{ij}, \ldots, \boldsymbol{r}_{nk}/\boldsymbol{c}_p, \ldots, \boldsymbol{c}_z/\boldsymbol{TMFC}/\boldsymbol{numan}.$$

The interface between the tools is established by a function inserted in the MOGA code, which coordinates the full process of reading and exchange of endogenous and exogenous variables between the programs. The entire simulation is a single process that is coordinated within the main routine of MOGA. The simulation process plays an important role in the testing of the fitness of the operational scenario because it supplies the variables that represent the dynamics and randomness of the process.

The simulator generates partial inputs for the assessment of the scenario; however, it is not a function of fitness. The stop criterion introduced in the simulation model is the functional period of the automated system, i.e., the number of hours that the system must function during simulation.

The position of the simulator in the flowchart of the optimizer (NSGA-II), as explained by [6], has paramount importance for the employed simulation process. As Fig. 2 indicates, the MOGA is responsible for sequentially producing operational scenarios. The simulator receives one scenario at a time and simulates its operation more realistically the more detailed its respective model is. In a later stage, the MOGA assesses the fitness of the scenario according to the nondominance criterion mentioned above.

When an operational scenario is created and sent to the simulator, the configuration of the machines that will be activated in the process is established, and the following system exogenous variables are calculated as the system evolves over time: *TMVC,* R_s, *TPPS, SDT, TSA, TNSF, TGPPS, CPLF, CINST, CMTTPM, CMODC,* and *CGM.*

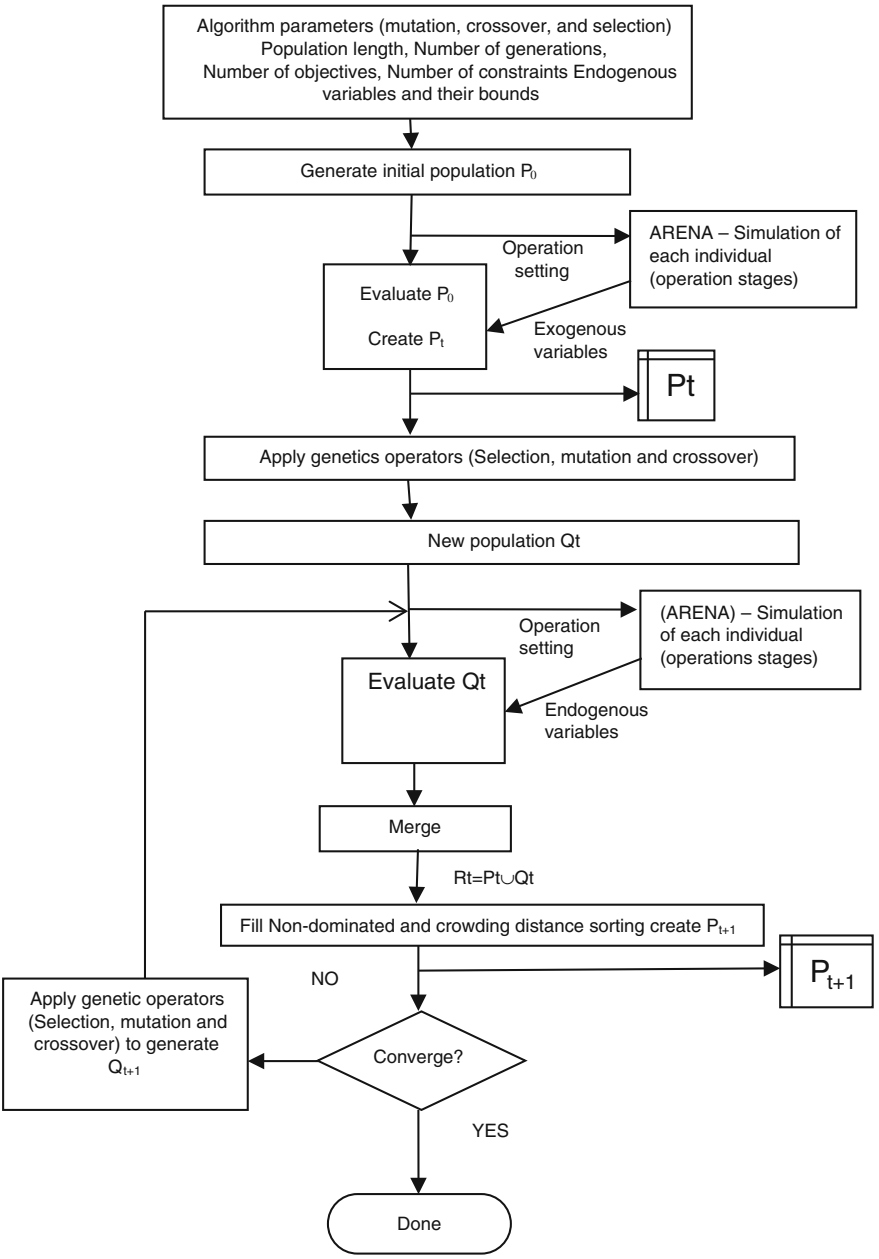


Fig. 2 Flowchart of simulation optimization framework

Some of these variables are included in the calculation of the objectives to be optimized, whereas other variables describe features related to system efficiency and the configuration of machines established in the operational scenario. Nevertheless, all of the variables are either directly or indirectly included in the MOGA-routine of the verification of the problem constraints.

In the routine, the values of the exogenous and endogenous variables described above are compared to limits that are preset by the analyst in the problem constraints. The satisfaction of the limits defines the fitness of the scenario under consideration. Scenarios are discarded when one or more restrictions are violated and admitted when all constraints and optimization objectives are satisfied.

Once the fitness of a scenario is established, the simulation process restarts with a new individual or operational scenario (previously generated in the MOGA). The global stop criterion of the process is the number of generations, where each generation consists of a set of individuals or operational scenarios. The MOGA is capable of handling only one generation at a time. This characteristic requires the inclusion of a waiting mechanism in the MOGA to allow for the simulator to perform the simulation of each individual or scenario of the corresponding generation.

8 Description of the Idealized Case

Actual systems, especially automated ones, are characterized by a wide interrelation of subsystems, such as pneumatic, hydraulic, microelectronic, and electrical systems. Actual systems are essentially complex sets with low human interference (automated). Due to such complexity, these systems exhibit several types of failure modes (e.g., electronic and mechanical) with random failure rates. Failure causes production loss or complete standstill.

Considering the high productivity of these systems, failure mitigation becomes a crucial activity that justifies the use of redundant operations. A well-allocated set of redundancies may ensure availability and high levels of systemic reliability.

To dimension a group of machines in an idealized automated production line, the process was applied to the design stage. Therefore, the present study considered an automated system represented by a generic and continuous production process. The process may contain as many as forty-five operations; nine of these operations form the core of the system functioning. There are also two warehouses; one warehouse contains production parts (left), and the other warehouse stores finished products (right), as shown in Fig. 2. There is also a conveyor that interconnects the system with a set of robots.

Each machine in the system corresponds to a different operation, such as milling, erosion, assembly, and adjustment, which must be performed in a specific sequence. The robots perform the transference and storage operations of the parts and finished products. The system as a whole (machines and robots) performs nine operations, as shown in Fig. 1.

The system can be represented schematically by a set of nine stages in series, corresponding to the nine basic operations of the system, as illustrated in Fig. 3.

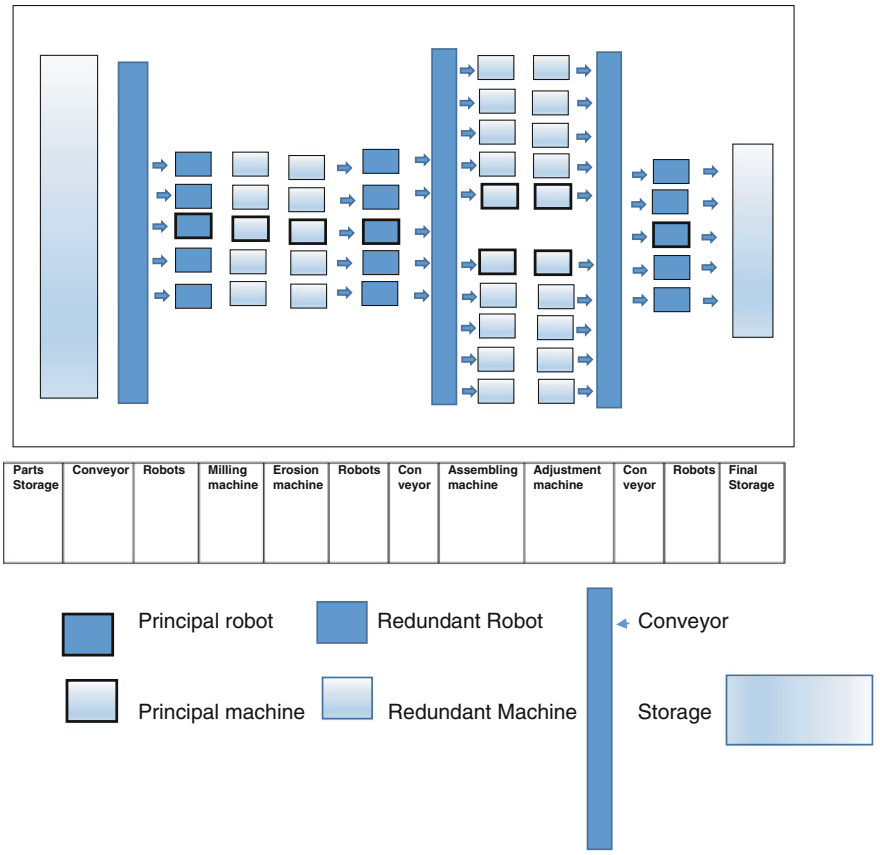


Fig. 3 Schematic diagram of the system

The series of stages represents a parallel system commonly used in RAP. Each stage or subsystem can be performed by as many as five machines that each have distinct operational characteristics. At least one of the five machines is assumed necessary for the basic functioning of the subsystem, whereas the remaining machines are redundancies that may or may not be allocated in the subsystem. The utilization and allocation of these redundancies is precisely what makes it possible to increase the entire availability of the system. The system can have a minimum configuration of nine machines and a maximum configuration of forty-five machines. The resulting system contains a large number of combinations, which warrants evaluation.

Each configuration or set of machines and robots chosen for operation has an associated number of maintainers, their corresponding unitary production costs,[2]

[2] The costs of production include

- CRAMA—cost of the raw materials applied
- CMPOLl—cost of the manpower of the operation of the line

individual reliability, and fixed maintenance costs corresponding to the identified configuration. This set of variables is called the operational scenario (configuration of machines, number of maintainers, individual production costs, and total fixed maintenance costs of the system). Operational scenarios with the same characteristics differ due to random events such as failures.

Therefore, the problem addressed in this study might be expressed with the following questions:

What is the optimal scenario of operation that maximizes total system reliability and minimizes simultaneously the use of maintainers, and the costs of maintenance and operation of the system?

9 Experiments

A total of six experiments were conducted, of which five were multi-objective and one was mono-objective:

- Three multi-objective experiments with 720, 1,440, and 2,160 simulated hours, respectively, and 50 generations, each comprised of 20 individuals (total of 1,000 individuals).
- One multi-objective experiment with 720 simulated hours and 100 generations, each comprised of 20 individuals (total of 2,000 individuals).
- One multi-objective experiment with 720 simulated hours and 150 generations, each comprised of 20 individuals (total of 3,000 individuals).
- One mono-objective experiment, which followed the model described in Sect. 6, with 720, 1,440, and 2,160 h of uninterrupted system operation.

Each individual corresponds to an operational scenario and each 720 h set corresponds to one month of uninterrupted system operation. The individuals were subjected to the process of evolution for the age that is guided by the reproduction operators. The probability of occurrence for the operators was 0.7 for the crossover operator and 0.02 for the mutation operator (real variables).

Such operators are responsible for the magnitude and diversity of the search and their occurrence probabilities were kept constant to ensure some homogeneity in the search across the experiments. Thus, substantial differences between the results of the five multi-objective experiments were avoided.

By keeping the occurrence probabilities of the genetic operators and the number of individuals per generation (20) in each experiment constant, the present study sought to demonstrate the influence of the increase in generations and the influence of the duration of the simulations within the boundaries of the experiments conducted.

(Footnote 2 continued)

- CEEOC—cost of electricity and other combustibles
- CPSAT—cost of packaging, storing, and transport.

In the case of multi-objective optimization, the algorithms used in the problems are expected to exhibit a set of solutions that satisfy the objectives. However, because there is a set of solutions, the final selection of a solution depends on a process to aggregate information, which can be performed *a priori*, *during the process*, or *a posteriori* [5].

In the *a priori* procedure, the criteria are established before the process is developed and guide the construction of the optimal Pareto set.

In the *during-the-process* procedure, the criteria are inserted during the development of the process and may include variables resulting either from previous iterations or inserted by the user.

In the *a posteriori* procedure, the process develops without interference by the decision-maker and, in the end, the information required to select a solution is added to the obtained Pareto set.

A procedure that aggregates information can exhibit a technical quantitative nature or a nontechnical qualitative nature. The former type of procedure treats the technical data as process specifications and variables related to the adjustment of the entire system, whereas the latter type of procedure expresses the opinions of the individuals who participate in decision-making.

In the study, the *a posteriori* procedure was chosen because the resulting Pareto set, which was prepared for the application of the selection criteria, became available at the end of the all experiments, as presented on the tables at Sect. 10.

10 Results—Analysis and Discussion

The main analysis of results is organized into two subanalysis: the firsts subanalysis concerns the convergence of the simulation optimization process and the second subanalysis concerns the process of selecting the best operational scenario. The latter subanalysis also sought to answer the questions posed at the end of section two:

> What is the optimal operational scenario for the idealized case under consideration? Is the system that maximizes total system reliability while minimizing operation and maintenance costs and the use of maintainers the ideal scenario?

Four objectives were assessed in each experiment. The assessment included indexes of reliability, cost, and number of maintainers. The reliability indexes varied between 0 and 1 whereas cost and number of maintainers could have any value, although the maintainers are integer variables. To compare the objectives, which have different natures, adjustments were needed. As a result, the following normalization process was adopted:

$$x_n \sqrt{\frac{\bar{x}_i}{\sum_{i=1}^{n} \bar{x}_i^2}}$$

where

x_n—value of a normalized point

$\bar{x}_i$—value of an average individual of generation i with $i = 1 \ldots n$
n—total number of generations.

10.1 Analysis of AGE Convergence

Figures 4c, 5b, 6b, 7b, and 8b were plotted by using the averages of the objectives for each generation to draw a line of tendency toward convergence, which illustrates reduction of the variability of the responses as a function of the evolution of the process. The MOGA approached its stop criterion (which in this case was the number of generations). The plotted graphs of the resulting Pareto boundaries' are shown in Figs. 4a,b, 5a, 6a, 7a, and 8a for each experiment and reveal an increase in the concentration of individuals and the number of generations. The experiments did not result in a larger number of feasible scenarios. An increase in the number of MOGA generations and computation times resulted in enhanced responses, similar to scenarios with lower costs and higher reliability indexes. The convergence of the genetic process is related to the number of generations, the probabilities of the genetic operators, the type of selection, and the stop criterion. Thus, the convergence of the process is certain, and a larger number of generations guarantees better results.

By keeping constant the number of generations under various simulation times, greater oscillation in the values of the objectives was observed in experiments with 1,440 and 2,160 h as compared with the experiment with 720 h, as shown in Figs. 7b and 8b, respectively. The outcome was due to the presence of undesirable stochastic events, such as failures, which altered the average availability of the system between simulations. Oscillation was expected because some modalities of failures were defined by exponential models with averages greater than 1,000 h. This finding indicates that the system exhibits natural degradation as a function of longer functioning periods, which is due to the presence of failures and other stochastic events that degenerate the system (e.g., reduced operation times and more frequent lines). Longer periods also cause degradation of the reliability of the system, as shown by the boundaries in Figs. 4b, 7a, and 8a, which represent the global system variables of reliability versus maintenance cost.

10.2 Analysis of the Selecting Process of the Best Operational Scenario

The mono-objective model (MOOM) was surpassed by the multi-objective model (MUOM) at SIMO framework for several reasons. The main reason is the impracticality of representing the stochastic characteristics inherent to the investigated system using the mono-objective model, although it produced reliability results similar to the averages of the optimal boundaries obtained by MOGA in each of the multi-objective experiments performed, as shown in Table 1. Reliability tended to decrease with longer simulation times because it is a function that indirectly reveals the aging

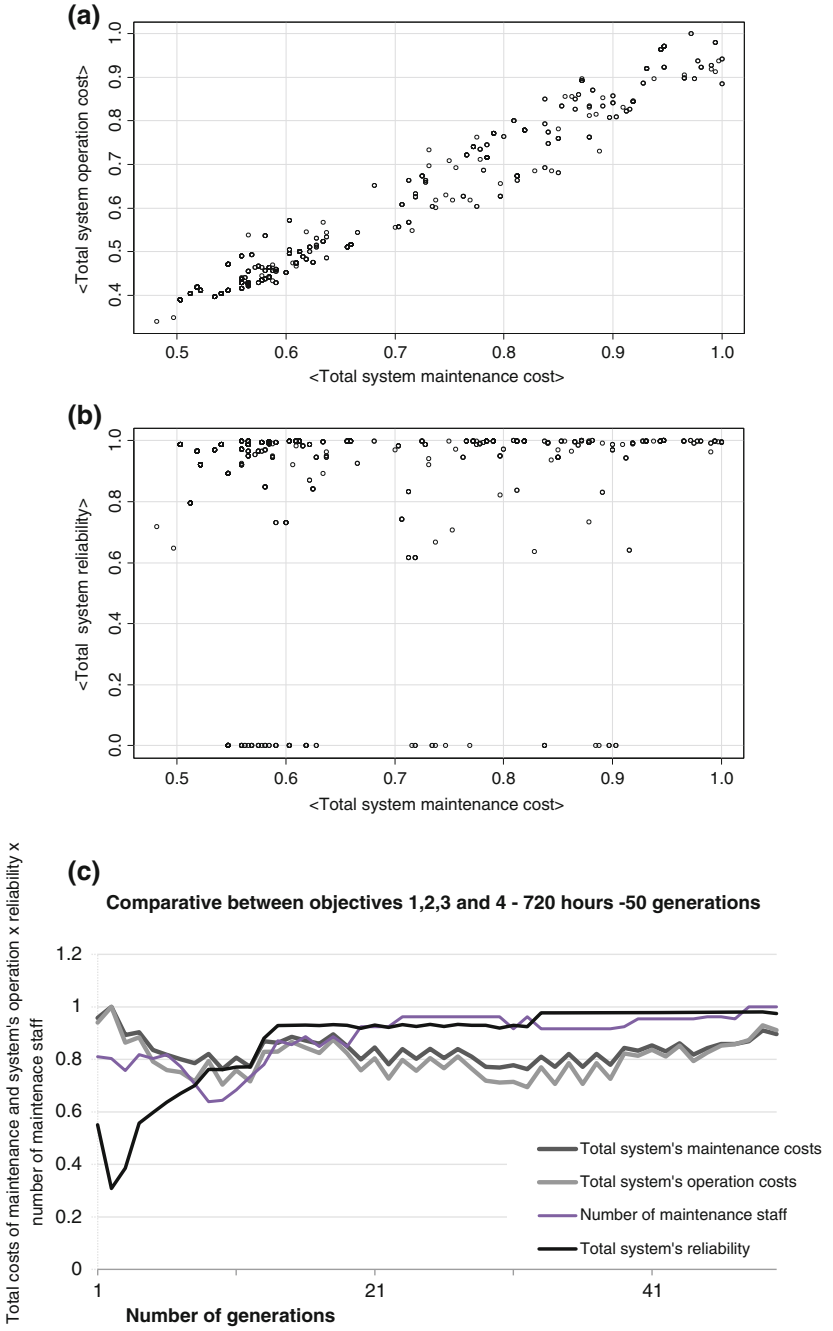


Fig. 4 **a** Pareto boundary objective 1 versus objective 3 for 720 h—50 generations; **b** Pareto boundary objective 2 versus objective 1 for 720 h—50 generations; **c** Experiment (simulation) for 1,720 h—50 generations

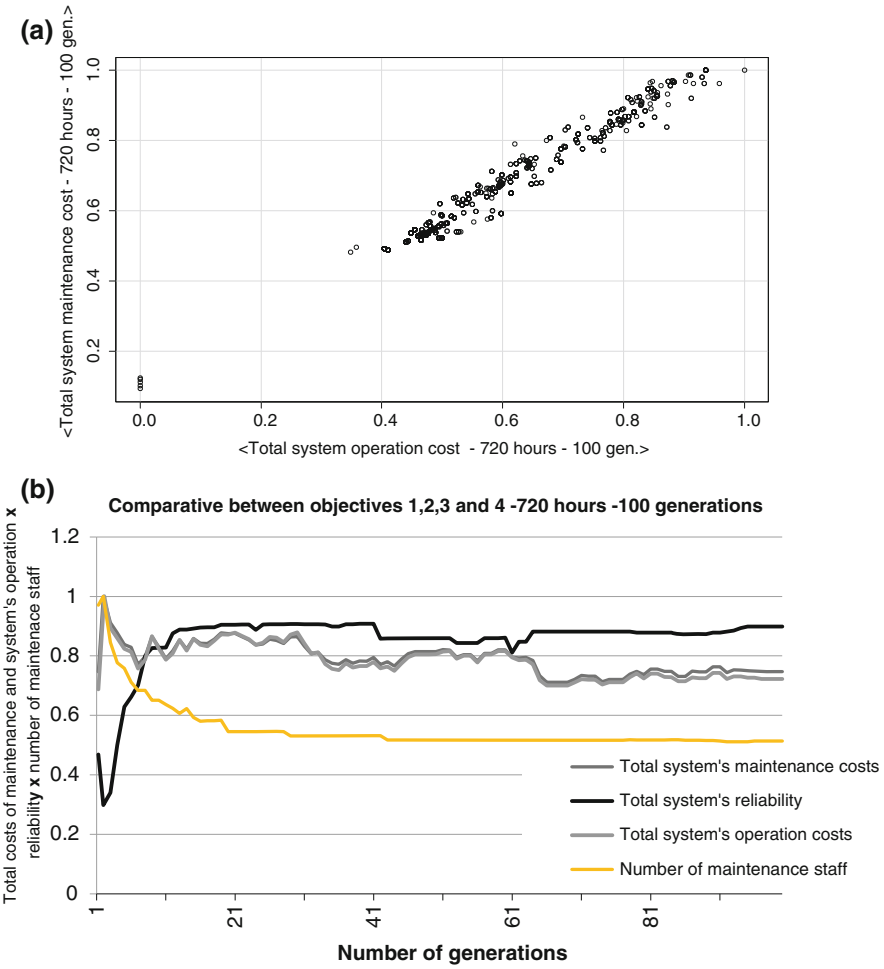


Fig. 5 **a** Pareto boundary objective 1 versus objective 3 for 720 h—100 generations; **b** Experiment (simulation) for 4,720 h—100 generations

of the system. The system is at a higher exposure to future risk of failure with a reduction in performance because the probability of failure increases with the passage of time. This situation can be prevented by adding redundancies to the system and improving the global reliability of the system as a function of the increase of alternative pathways for the process.

The costs obtained in the simplified mono-objective model were higher than the averages of the multi-objective simulation, as shown in Table 2.

Although a simplified model is easier to solve, it is not necessarily less complex. If the development of such a model required many simplifications, the accuracy of its responses would be impacted and the model would not serve its purpose. The consistency of the simplified model, as described in Sect. 6.1, is based on the

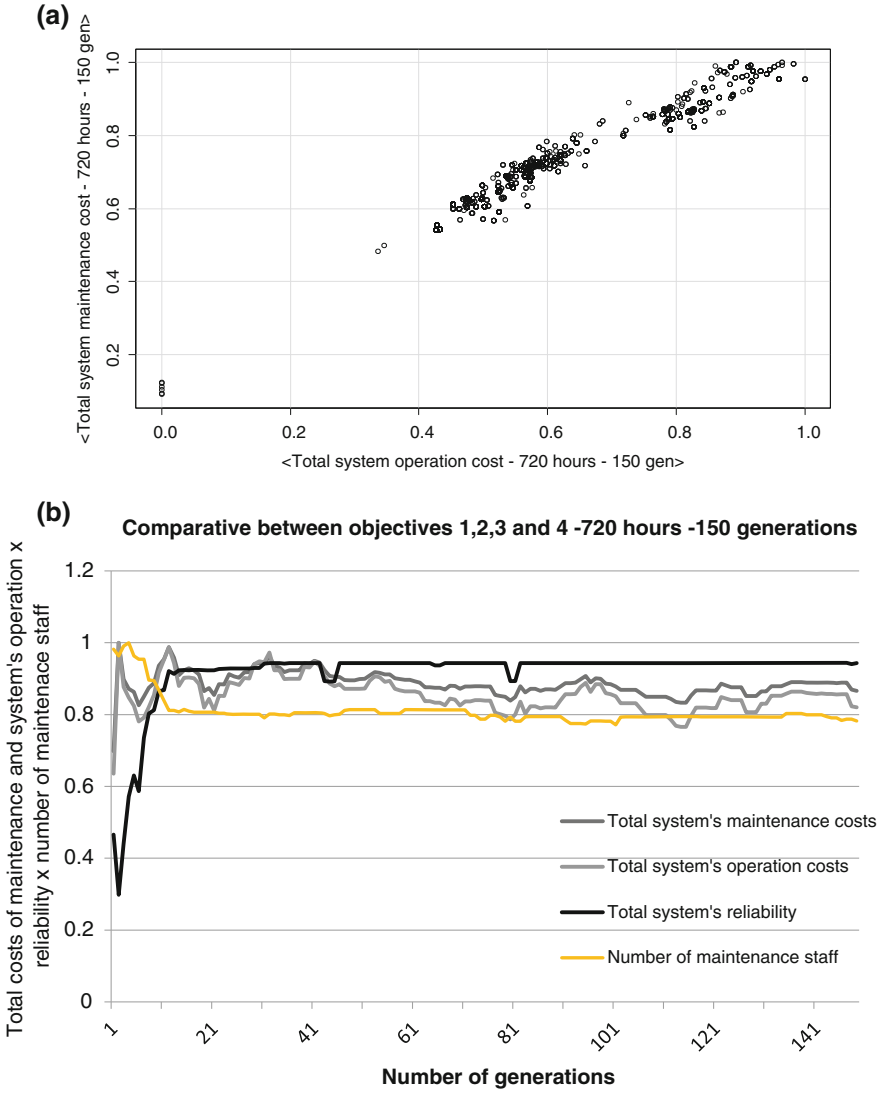


Fig. 6 **a** Pareto boundary objective 1 versus objective 3 for 720 h—150 generations; **b** Experiment (simulation) 5 for 720 h—150 generations

appropriation of the same cost formulation applied to the simulation model; however, without the influence of the full stochastic conjuncture (e.g., failures and reduction of process speeds), which the simulation sought to emulate. Therefore, the process costs in the simplified model originated exclusively from one specific configuration of machines and solely from the number of allocated machines. Figure 9 represents the average values of the main objectives by generation and illustrates that the results of

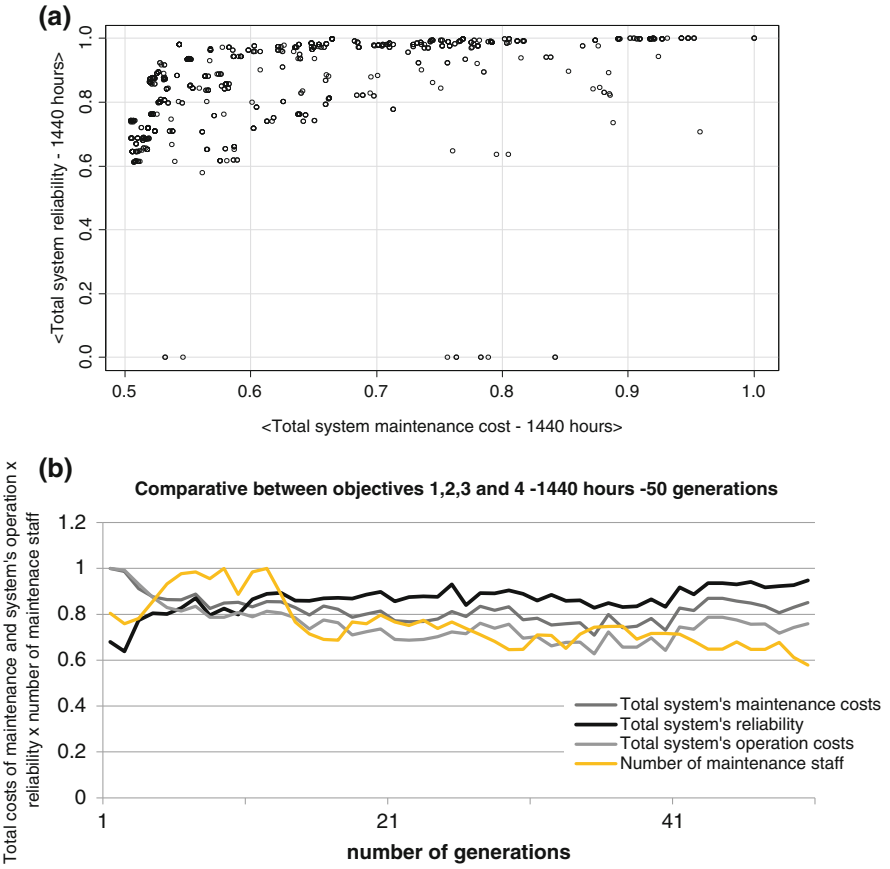


Fig. 7 **a** Pareto boundary objective 1 versus objective 3—1,440 h—50 generations; **b** Experiment (simulation) 2 for 1,440 h—50 generations

the simulation optimization process, which included the above-mentioned stochastic context, surpass the results of the simplified model (mono-objective) in terms of costs. The results accurately reflect the evolution of the responses not only by the level of randomness in the model but also by the level of randomness in the evolutionary process itself.

Of the 8,000 operational scenarios generated in the study, forty-four scenarios were selected by the MOGA that satisfied all objectives and constraints of the problem, as shown in Tables 3 and 6. In spite of the low number of simulations, satisfactory results were achieved. The selected scenarios were initially clustered according to the number of machines allocated in the process. According to that criterion, the minimum number of allocations was fifteen and the maximum number of allocations was forty-two, within a universe that ranged from a total of nine to forty-five

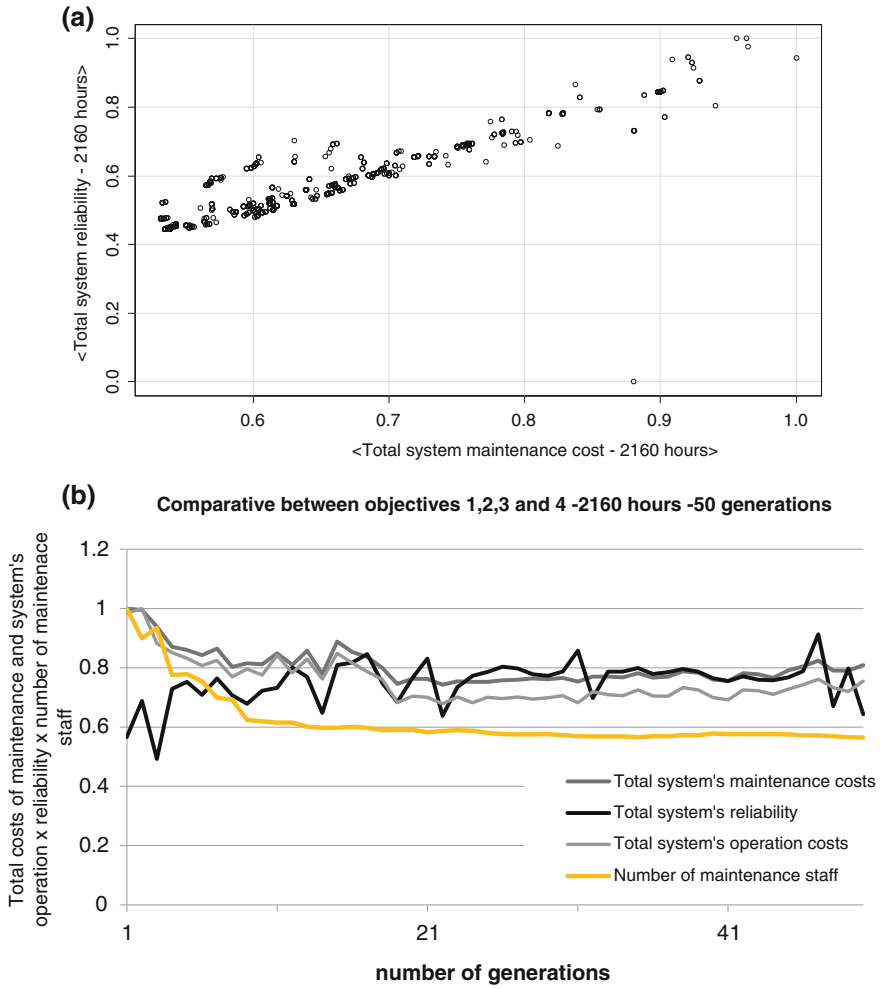


Fig. 8 **a** Pareto boundary objective 1 versus objective 3 for 2,160 h—50 generations; **b** Experiment (simulation) 3 for 2,160 h—50 generations

machines. By itself; however, this criterion did not facilitate the selection of the most appropriate operational scenario.

Another criterion that was considered was the reliability of the scenario. The reliability of the selected scenarios oscillated slightly, with a tendency toward improvement as the number of allocated machines increased (greater redundancy). All selected experiments exhibited high reliability (>0.98), which makes that criterion insufficient for decision-making.

The decision-maker might consider other variables of interest such as cost/product, maintenance costs (objective 1), and operational costs (objective 3), thus creating a novel criterion. Upon application of this criterion, the resulting minimum

Table 1 Reliabilities—multi-objective versus mono-objective experiments

	MUOM—SIMO FRAMEWORK		MOOM	
		Means*	720/1,440 h	2,160 h
Objective 2	720/50	0.870737	0.88	0.78
	1440/50	0.863031	0.88	0.78
	2160/50	0.752815	0.78	0.78
	720/100	0.999761	0.88	0.78
	720/150	0.999959	0.88	0.78

Table 2 Maintenance (objective 1) and operation (objective 3) costs of the system—multi- and mono-objective experiments

	MUOM—SIMO framework (mean values) Multi-objective model		MOOM Mono-objective model	
Simulations	Objective 1	Objective 3	Objective 1	Objective 3
720/50	2,071,950.00	2,257,797.00	2,522,403.00	4,123,482.00
720/100	2,014,479.00	2,166,809.00	2,522,403.00	4,123,482.00
720/150	2,371,427.00	2,592,107.00	2,522,403.00	4,123,482.00
1,440/50	1,962,392.00	2,022,221.00	2,700,065	5,902,222.00
2,160/50	1,892,339.00	2,010,936.00	2,930,879.00	7,734,114.00

cost was 5.4 C.U. (experiment 2 720/100/3 and 720/100/4), as shown in Table 2, and the resulting maximum cost was 11.19 C.U. (experiment 1 720/50/6), as displayed in Table 3, corresponding to 36 and 32 allocated machines, respectively (C.U. = currency units). When 36 machines were allocated, the six maintainers

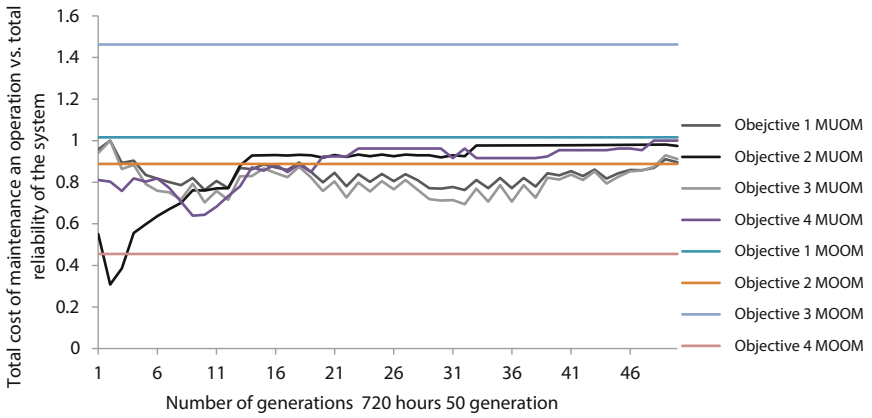


Fig. 9 Comparison between the average fitness of the objectives 1, 2, 3, and 4—multi-versus mono-objective simulation

Table 3 Experiment (simulation) for 1,720 h—50 generations

Experiment 1 (run 1)		720 h—50 generations Objective 1—Total system's maintenance cost Objective 2—Total system's reliability Objective 3—Total system's operation cost Objective 4—Total number of maintenance staff				
Operation set	Obj. 1	Obj. 2	Obj.3	Obj. 4	Machines assignments	Cost/ product
1	2,820,206	0.998928	3,624,749	16	35	6.45
2	2,792,698	0.998885	3,709,403	16	35	5.92
3	1,959,979	0.997946	2,079,463	16	35	8.23
4	1,606,541	0.985626	1,622,457	5	31	9.76
5	1,733,869	0.987398	1,676,912	16	31	10.39
6	1,809,541	0.996716	1,770,991	16	32	11.19
7	1,786,686	0.996919	1,775,866	16	33	10.11
8	2,468,443	0.998576	3,081,645	16	34	6.69
9	2,585,959	0.998631	3,326,377	16	35	6.28
Means	2,167,227.2	0.995818	2,481,908	12	33.5	8.33

were found in objective 4, as compared with sixteen maintainers found when 32 machines were allocated.

By adding to this criterion the requirement that the lowest desired number of maintainers must be less than seven with cost per product lower than 6.00 C.U. Considering the solutions described in Tables 3, 4, 5 and 6 and striving to answer the questions in Sect. 2, it was concluded that the most appropriate (optimal) scenarios for the idealized case were as shown in Table 7.

The costs differed for similar and identical numbers of allocated machines. This finding reveals an interesting characteristic of the process; specifically, that the quantity of machines, the position of each machine in the system, and the consequences of the random events to which they were subjected (failures) were all considered. This consideration causes alterations in production and exposes the system to failures that might result in longer standstill of an operation. In such cases, despite their quantitative accuracy, the results of the combination of machines or of the operational scenario to which they belong differ; consequently, their performance depends on the stochastic conjuncture that was produced.

11 Remarks

This chapter discusses the problem of redundancy and reliability allocation in an automated production system. The purpose of the study is to improve the global reliability of the system by allocating alternative components (redundancies) that

Table 4 Experiment (simulation) for 4, 720 h—100 generations

Experiment 4 (run 4)		720 h—100 generations Objective 1—Total system's maintenance cost Objective 2—Total system's reliability Objective 3—Total system's operation cost Objective 4—Total number of maintenance staff				
Operation set	Obj. 1	Obj. 2	Obj. 3	Obj. 4	Machines assignment	Cost/ product
1	1,967,423	0.999746	2,162,364	6	36	6.23
2	1,641,490	0.999063	1,801,222	5	35	5.65
3	1,744,774	0.999701	1,955,186	6	36	5.40
4	1,696,361	0.999691	1,893,462	6	36	5.40
5	1,691,796	0.999581	1,885,273	6	36	5.42
6	2,014,479	0.999761	2,166,809	6	35	7.00
Means	1,792,721	0.999591	1,977,386	6	35.6	5.85

Table 5 Experiment (simulation) 2–1,440 h—50 generations

Experiment 2 (run 2)		1440 h—50 generations Objective 1—Total system's maintenance cost Objective 2—Total system's reliability Objective 3—Total system's operation cost Objective 4—Total number of maintenance staff				
Operation set	Obj.1	Obj. 2	Obj. 3	Obj. 4	Machines assignment	Cost/ product
1	2,914,198	0.9993065	3,147,688	6	27	5.81
2	2,043,527	0.9985866	2,074,511	7	27	6.81
3	2,742,798	0.9987716	2,901,673	6	27	6.13
4	1,669,170	0.9797631	1,641,688	6	20	7.11
5	1,992,993	0.9837161	2,040,098	15	21	6.30
6	2,010,620	0.9800117	1,993,446	7	20	7.19
7	2,863,146	0.9991417	3,064,383	6	29	5.83
8	2,771,984	0.9990900	2,931,642	6	28	6.18
9	2,896,160	0.9993100	3,130,280	6	29	5.77
10	2,788,534	0.9988900	2,915,434	6	27	7.10
Means	2,469,313	0.9936590	2,584,084	7.1	25.50	6,42

are associated in parallel with each original component and to minimize the use of maintainers and the associated system maintenance and operation costs.

The simulation optimization process employed uses a model constructed with variables and constraints. The process differs from conventional optimization problems because there is no function (or set of functions) that represents the system.

Therefore, the model can assume various degrees of complexity that allow it to accurately represent actual industrial situations.

The method used is intimately related to the constraints imposed on the model, which result from the judgment of the modeler. The process benefits from and depends on the accuracy (quality) of the work of the modeler.

The setting of the operational scenarios and the choice of the best results is entirely automatic within an evolutionary process that is part of a genetic algorithm and based on the production of a large amount of simulations. The optimizing evolutionary process that was considered revealed few simulations in the discrete system (total of 8,000) that achieved satisfactory results. Of these 8,000 simulations, forty-four feasible scenarios were chosen by the MOGA, according to the imposed constraints. Thus, in spite of the small number of simulations, the proposed approach facilitates decision-making by permitting the generation of a reduced and feasible set of solutions, which is sufficient to alter the constraint limits and minimize the number of operational scenarios.

The method used was efficient in developing feasible solutions, in spite of the existence of multiple conflicting objectives, constraints, systemic interrelations, and random factors. For this reason, and considering the resulting high degree of reliability of the solutions (>0.98) for 720, 1,440, and 2,160 h of system functioning and the degree of generalization of the developed model, one might assert that the suggested method exhibits a wide scope of potential applications for actual industrial situations.

Table 6 Experiment (simulation) for 3–2,160 h—50 generations

Experiment 3 (run 3)		2160 h—50 generations Objective 1—Total system's maintenance cost Objective 2—Total system's reliability Objective 3—Total system's operation cost Objective 4—Total number of maintenance staff				
Operation set	Obj.1	Obj. 2	Obj. 3	Obj. 4	Machines assignment	Cost/ product
1	1,658,352	0.981308	1,623,417	6	23	10.11
2	1,923,583	0.985386	1,899,560	6	20	11.15
3	2,157,052	0.993085	2,287,982	6	24	7.84
4	1,965,154	0.989701	2,058,740	6	21	7.91
5	2,217,531	0.994180	2,399,669	6	25	7.24
6	2,405,210	0.995831	2,726,515	6	25	6.10
7	2,145,893	0.993032	2,287,960	6	24	7.56
8	2,031,733	0.991144	2,119,967	6	22	8.29
9	2,239,416	0.994185	2,418,995	6	25	7.30
10	2,404,594	0.995893	2,728,048	6	25	6.10
Means	2114856	0.991330	2,255,058	6	23.4	7.96

Table 7 Final set of solutions after application of the decision-maker preference criteria

Simulation time/generations	Operation set number	Obj.1	Obj. 2	Obj. 3	Obj. 4	Machines assignment	Cost/product
720/100	2	1,641,490	0.999063	1,801,222	5	35	5.65
720/100	3	1,744,774	0.999701	1,955,186	6	36	5.40
720/100	4	1,696,361	0.999691	1,893,462	6	36	5.40
720/100	5	1,691,796	0.999581	1,885,273	6	36	5.42
1440/50	1	2,914,198	0.9993065	3,147,688	6	27	5.81
1440/50	7	2,863,146	0.999142	3,064,383	6	29	5.83

Although the results are satisfactory given the constraints and the small number of simulations used, the developed process and model requires further analysis due to the multiple possibilities that are feasible with more specific variables (to be included in the model).

The use of production systems with more complex structures or the selection of alternate objectives should be considered in future studies, such as:

- Optimization of the stock of maintenance parts;
- Optimization of the number of maintenance inspections to reduce periods of stand-still;
- Optimization of the sequencing of tasks; and
- Optimization of the operational parameters to handle changes in productivity levels.

Acknowledgments To Dr. Deb and his team at the Kanpur Genetic Algorithms Laboratory (KanGAL) for providing the source code of the MOEA NSGA-II used in this work.

References

1. Azadivar, F. (1992). A tutorial on simulation optimization. In *Proceedings of the, (1992). 24th winter simulation conference*, pp. 198–204. doi:10.1145/167293.167332.
2. Billiton, R. A., Ronald N. (1992). Reliability evaluation of engineering systems: Concepts and techniques (2nd ed). Springer. ISBN-10 0306440636; ISBN-13 978–0306440632.
3. Coelho, L. S. (2009). Reliability-redundancy optimization by means of a chaotic differential evolution approach. *Chaos, Solitons and Fractals, 41*(2), 594–602. doi:10.1016/j.chaos.2008.02.028.
4. Coello, C. A., Lamont, G. B. (2004). Applications of multi-objective evolutionary algorithms. Advances in Natural Computation (Vol. 1). World Scientific Publishing. ISBN 981-256-106-4.
5. Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. England: Wiley. ISBN 047187339X, 9780471873396.
6. Deb, K., Pratap, A., & Agarwal, S. (2002). Fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE-Transactions on Evolutionary Computation, 6*(2), 182–197. doi:10.1109/4235.996017.
7. Ergott, M. & Gandibleux, X. (2004). Approximate solution methods for multi-objective combinatorial optimization. Sociedad de Estadística e Investigación Operativa. Madrid. Spain TOP Vol. 12. No. 1. 1–90 June. doi:10.1007/BF02578918.
8. Gen Mitsuo, Cheng Runwei (2000). Genetic Algortihms & Engineering Optimization. Canada. Wiley. A Wiley-interscience publication. ISBN 0471315311, 9780471315315.
9. Ghiani Gianpaolo, Legato Pasquale, Musmanno Roberto, Vocaturo Francesca. A combined procedure for discrete simulation-optimization problems based on the simulated annealing framework. Computational Optimization and Applications, September 2007, Volume 38, Issue 1, Pages 133.145. doi:10.1007/s10589-007-9010-7
10. Hani, Y., et al. (2008). Simulation base optimization of a train maintenance facility. *Journal of Intelligent Manufacturing, 19*(3), 293–300. doi:10.1007/s10845-008-0082-8.
11. Holland, John Henry (1975). Adaptation in natural and artificial systems: an introductory analysis eith applications to biology, control and artificial intelligence. University of Michigan Press. ISBN 0472084607, 9780472084609.
12. Jianping, L. I., & Xishen, J. (1997). A new partial bound enumeration technique for solving reliability redundancy optimization. *Microelectronics Reliability, 37*(2), 237–242. doi:10.1016/S0026-2714(96)00109-6.

13. Khalili-Damghani, K., & Maghsoud, A. (2012). Solving binary-state multi-objective reliability redundancy allocation series-parallel problem using efficient epsilon-constraint, multi-start partial bound enumeration algorithm, and DEA. *Reliability Engineering and System Safety, 103*, 35–44. doi:10.1016/j.ress.2012.03.006.
14. Kulturel-Konak, S., & Coit, D. W. (2007). Determination of Pruned Pareto sets for the multi-objective system redundancy allocation problem. *IEEE Symposium on computational intelligence in multicriteria decision making*, pp. 390–394. doi:10.1109/MCDM.2007.369118.
15. Kulturel-Konak, S., Smith, A. E., & Norman, B. A. (2006). Multi-objective tabu search using a multinomial probability mass function. *European Journal of Operational Research, 169*, 918–931. doi:10.1016/j.ejor.2004.08.026.
16. Kumar, R., Izui, K., Yoshimura, M., & Nishiwaki, S. (2009). Multi-objective hierarchical genetic algorithms for multilevel redundancy allocation optimization. *Reliability Engineering and System Safety*, *94*(4), 891–904. doi:10.016/j.ress.2008.10.002.
17. Kuo, W., Prasad, V. R., Tillman, F. A., & Hwang, C.-L. (2001). Optimal reliability design fundamentals and applications. Cambridge University Press. ISBN 0521781272, 9780521781275.
18. Kuo, W., Zuo, M. J. (2003). Optimal reliability modeling. Principles and applications. New Jersey: Wiley. ISBN 047139761X, 9780471397618.
19. Liang, Y.-C., & Lo, M.-H. (2010). Multi-objective redundancy allocation optimization using a variable neighborhood search algorithm. *Journal of Heuristics, 16*, 511–535. doi:10.1007/s10732-009-9108-4.
20. Lins, Isis Didier, & Droguett, Enrique López. (2011). Redundancy allocation problems considering systems with imperfect repairs using multi-objective genetic algorithms and discrete event simulation. *Simulation Modelling Practice and Theory, 19*, 362–381. doi:10.1016/j.simpat.2010.07.010.
21. Lisnianski, A., & Levitin, G. (2003). Multi-state system reliability Assessment optimization and application (Vol. 6). Singapore: World Scientific Publishing. Series on Quality reliability and Engineering statistics. ISBN 981-238-306-9.
22. Loo Hay Lee, Ek Peng Chew, Kee Hui Chua, Zhuo Sun, Lu Zhen. A Simulation Optimisation Framework for Container Terminal Layout Design. Multi-objective Evolutionary Optimisation for Product Design and Manufacturing 2011, Pages 385–400. doi:10.1007/978-0-85729-652-8_14.
23. Mattila, V., Virtanen, K. (2005). A simulation-based optimization model to schedule periodic maintenance of a fleet of aircraft. In *Proceedings 2005. European simulation and modeling conference*, Porto, Portugal, October 24–26, pp. 479–483.
24. Meedeniya, I., Aleti, A. & Buhnova, B. (2009). Redundancy allocation in automotive systems using multi-objective optimization. Jacobs school UCSD symposium on automotive/avionics systems engineering SAASE 2009. http://www.jacobsschool.ucsd.edu/GordonCenter/g_leadership/l_summer/s_saase/schedule.shtml
25. Mehmet, Çunkaş. (2010). Intelligent design of induction motors by multiobjective fuzzy genetic algorithm. *Journal of Intelligent Manufacturing, 21*(4), 393–402. doi:10.007/s10845-008-0187-0.
26. Merkuryeva, G., & Napalkova, L. (2008). Development of multi-objective simulation-based genetic algorithm for supply chain cyclic planning and optimization. In *International conference 204th EURO mini conference continuous optimization and knowledge-based technologies (EurOPT) May 20.23*. Neringa, Lithuania. ISBN 978-9955-28-283-9.
27. Misra, K. B., Sharma, U. (1991). An efficient approach for multiple criteria redundancy optimization problems. *Microelectronics Reliability, 31*(2/3), 303–321. doi:10.016/0026-2714(91)90216.
28. Murata. T. (1989). Petri Nets: Properties, Analysis and applications. Proceedings of the IEEE. Vol. 77. No 4. pp 541-580. doi: 10.1109/5.24143.
29. Nesmachnow, Sergio (2004). A parallel version of evolutionary algorithm for multiobjective optimization NSGA-II. Proceedings of X Argentine Computation Science Congress, Buenos Aires, Argentina, 2004.

30. Olafsson Sigurdur, Kim Jumi. Simulation Optimization. Proceedings of the IEEE Winter Simulation Conference, Pages 79–82, 2002.
31. Peterson. J.L. (1981) Petri Net Theory and the Modeling of Systems. Prentice-Hall. N.J. ISBN 0136619835.
32. Robin F, Orzati A, Moreno E, Homan O J, Bachtold W. Simulation and evolutionary optimization of electron-beam lithography with genetic and simplex-downhill algorithms. IEEE Transactions on Evolutionary Computation, Volume 7, Issue 1, February 2003, Pages 69–82.
33. Rozenberg Grzegorz, Reisig Wolfgang, Desel Jörg (2004). Lectures on concurrency and Petri Nets. Advances in Petri Nets. Lecture notes in Computer Science. Vol. 3098. doi: 10.1007/b98282.
34. Salazar, D., Rocco, C. M., & Galvan, B. J. (2006). Optimization of constrained multiple-objective reliability problems using evolutionary algorithms. *Reliability Engineering and System Safety, 91*(9), 1057–1070. doi:10.1016/j.ress.2005.11.040.
35. Shelokar, P. S., Jayaraman, V. K., & Kulkami, B. D. (2002). Ant algorithm for single and multi-objective reliability optimization problems. *Quality and Reliability Engineering International, 18*, 497–514. doi:10.1002/qre.499.
36. Soundharajan, B; Sudheer K.P. Sensitivity analysis and auto-calibration of ORYZA2000 using simulation-optimization framework. Paddy and Water Environment, January 2013 - Volume 11, Issue 1–4, Pages 59–71. doi:10.1007/s10333-011-0293-z.
37. Swisher, J. R., Hyden, P. D., Jacobson, S. H., & Schruben, L. W. (2000). A survey of simulation optimization techniques and procedures. In *Proceedings of 2000. Winter simulation conference* (Vol. 1, No. 10–13, pp. 119–128). Dec 2000. doi:10.1109/WSC.2000.899706.
38. Taboada, H. A. & Coit, D. W. (2008). Development of multiple objective genetic algorithm for solving reliability design allocation problems. In J. Fowler, & S. Masons (Eds), *Proceedings of the 2008 industrial engineering research conference*. Available at: http://ie.rutgers.edu/resource/research_paper/paper_08-004.pdf
39. Taboada, H. A., Baheranwala, F., Coit, D. W., & Wattanapongsakorn, N. (2007). Practical solutions for multi-objective optimization: An application to system reliability design problems. *Reliability Engineering and System Safety, 92*, 314–322. doi:10.1016/j.ress.2006.04.014.
40. Taboada, H. A., & Coit, D. W. (2012). A new multiple objective evolutionary algorithm for reliability optimization of series-parallel systems. *Applied Evolutionary Computation, 3*(2), 1–18. doi:10.4018/jaec.2012040101.
41. Tian, Shigang, & Suo, Ming J. (2006). Redundancy allocation for multi-state systems using physical programming and genetic algorithms. *Reliability Engineering and System Safety, 91*, 1049–1056. doi:10.1016/j.ress.2005.11.039.
42. Tian, Z., Lin, D., & Bairong, W. (2012). Condition based maintenance optimization considering multiple objectives. *Journal of Intelligent Manufacturing, 33*(2), 333–340. doi:10.1007/s10845-009-0358-7.
43. Tzu-Chieh, H. & Kuei-Yuan, C. (2011). Uncertainity quantifications of Pareto optima in multiobjective problems. *Journal of Intelligent Manufacturing*. doi:10.1007/s10845-011-0602-9.
44. Ullrich, Christian. Exchange Rate Hedging in a Simulation/Optimization Framework. Forecasting and Hedging in the Foreign exchange Markets. Lectures Notes in Economics and Mathematical Systems, Volume 623, 2009, Pages 185–188. doi:10.1007/978-3-642-00495-7_17.
45. Wang, Z., & Chen T., Tang K., Yao X. (2009). A multi-objective approach to redundancy allocation problem in parallel-series systems. *IEEE Congress on evolutionary computation–CEC*, pp. 582–589. doi:10.1109/CEC.2009.4982998.
46. Yang Taho, Chou Pohung. Solving a multiresponse simulation-optimization problem with discrete variables using a multiple-attribute decision-making method. Mathematics and Computers in Simulation, Volume 68, Issue 1, 3 February 2005, Pages 9–21.

47. Zhou, L., Yoke San, W., & Kim Seng, Lee. (2011). A manufacturing-oriented approach for multi-platforming product family design with modified genetic algorithm. *Journal of Intelligent Manufacturing*, *22*(6), 891–907. doi:10.1007/s10845-009-0365-8.
48. Zio, E. & Zille, V. (2007). Multi-objective optimization of network systems by using ANT algorithms. *Quality Technology of Quantitative management. 4*(2), 211–224. Available at: http://web.it.nctu.edu.tw/qtqm/qtqmpapers/2007V4N2/2007V4N2_F4.pdf

OR and Simulation in Combination for Optimization

Nico M. van Dijk, René Haijema, Erik van der Sluis, Nikky Kortbeek, Assil Al-Ibrahim and Jan van der Wal

Abstract This chapter aims to promote and illustrate the fruitful combination of classical operations research (OR) and computer simulation. First, a highly instructive example of parallel queues will be studied. This simple example already shows the necessary combination of OR (queueing) and simulation that appears to be of practical interest such as for call center optimization. Next, two more 'real life' applications are regarded: (1) blood platelet production and inventory management at blood banks, and (2) train conflict resolution for railway junctions. Both applications show the useful combination of *simulation* and *optimization methods from OR*, in particular stochastic dynamic programming (SDP) and Markov decision theory (MDP), to obtain simple rules that are nearly optimal. The results are based on real-life Dutch case studies and show that this combined OR-simulation approach can be most useful for 'practical optimization' and that it is still wide open for further application.

1 Introduction

Discrete event simulation is well known as a most powerful tool for logistical process computation and performance evaluation in a vast majority of fields. Standard applications are found in the production sector, the service industry (call centers, administrative logistics), transportation (public transportation systems, road traffic, airports,

N.M. van Dijk (✉) · R. Haijema · E. van der Sluis · N. Kortbeek · A. Al-Ibrahim · J. van der Wal
Faculty of Economics and Business, University of Amsterdam, Amsterdam, The Netherlands
e-mail: N.M.vanDijk@uva.nl

N.M. van Dijk · N. Kortbeek
Stochastic Operations Research group, University of Twente, Enschede, The Netherlands

R. Haijema
Operations Research and Logistics Group, Wageningen University, Wageningen, The Netherlands

M. Mujica Mota et al. (eds.), *Applied Simulation and Optimization*,
DOI 10.1007/978-3-319-15033-8_3

harbors, maritime logistics, express delivery systems, and so on), computer networks, communication networks and, over the last decade with fast growing attention, in health care logistics. In most of these applications simulation is required, due to

- the complexity of the system,
- the uncertainties (stochastics) involved at micro up to macro level.

On the one hand, analytic techniques, most notably OR (operations research) techniques such as queueing analysis and dynamic programming, are generally restricted to simplified models and simplifying underlying assumptions that have to be made. On the other hand, simulation by itself does not standardly provide underlying insights nor techniques for optimization. Clearly, in simple situations in which an optimization problem can be parameterized by one parameter, such as by the number of staffing or storage capacity to be determined, simulation search approaches can be suggested to expedite and automate the search for an optimal value. An elegant exposé of such methods can be found in Krug [20]. Case-specific references are found in Sects. 2–4.

Unfortunately, in most realistic logistical situations there will be multiple parameters and problem aspects that complicate the optimization. In these situations, at best a number of different scenarios might be proposed to be evaluated and to be compared by simulation. Alternatively, fast and extensive simulation search approaches might be developed for optimization, as studied in the last decade. As such, simulation is to be regarded as a most practical and almost unlimited tool for scenario "optimization". But, as mentioned, it remains to be realized that simulation by itself does not provide any of these scenarios nor an automatic tool for optimization. This is where OR might help out in either of two directions

(i) To suggest scenarios based upon general OR results and insights.
(ii) To provide an OR-optimization technique for the problem included.

Clearly, analytic or OR models are generally too simplistic for realistic modeling. Simulation in contrast, hardly seems to have any limitations on the modelling complexity at all. But to the price of loosing general insights due to this complexity or at least having to simulate extensively to gain such insights. Here the simplistic OR model might play an important role of just being generic and providing essential insights. A similar statement can be made for insights on modeling the underlying stochasticity. OR models strongly rely upon distributional assumptions, most notably of exponential nature, which can easily be parameterized by an arrival or a service rate. Such strict simplifying assumptions can be relaxed by simulation, but to the price of requiring detailed input data on very specific input distributions. In addition, the outcomes of a simulation dependent on the sampled random data. For a fair comparison of slightly different scenarios of the same system requires many and long simulation runs to obtain accurate confidence intervals that allow for hypothesis testing. In contrast, to say the least, analytic or OR models, even though simplistic and whether exact or approximate, provide expressions or algorithms that are 100 % verifiable and replicable.

Table 1 Combined advantages

OR advantages	OR disadvantages
Optimization	Simple models
By techniques	Strict assumptions
Also by insights	
Scenario development	
Analytic approximate results	
SIM-advantages	SIM-disadvantages
Evaluation only	Real-life stochastics
By numbers only	Real-life complexities
By scenarios only	
Computationally expensive	
Requires highly detailed data	

In short, a combination of OR and simulation might thus become highly beneficial to compensate for the disadvantages of one another and to exploit the advantages of either

- OR for insights from simple computation and optimization,
- Simulation for more realistic evaluation and validation.

The disadvantages and combined advantages are summarized in Table 1.

This chapter aims to promote and illustrate the practical potential of the combination of simulation and optimization (OR). It therefore collects and exposes three applications based on more detailed and technical papers by the authors, as outlined in Table 2. The chapter is organized by its separate applications in Sects. 2–4. In each of these sections, the same structuring is used by its specific problem formulation and background literature, and also by a presentation of the combined OR-simulation approach and by its concrete practical numerical results and conclusions. The chapter will be concluded by an evaluation in Sect. 5.

2 Should We Pool or Not?

This section studies the question as simple as whether queues (or line ups) in front of service desks should be combined (pooled) into a single queue (line up) or not. The example in this section is based on van Dijk and van der Sluis [30] and van Dijk and van der Sluis [31].

Table 2 Combination of techniques for three applications

Sections	Topic	Combination
Sect. 2	Pooling in call centers	SIM + Q insights
Sect. 3	Blood banks	SIM + MDP
Sect. 4	Railways	SIM + Q + semi-MDP

Legend SIM: Simulation; Q: Queueing; MDP: Markov Dynamic Programming

In this section, it will be shown that both OR insights and simulation are essential for an improvement of either a pooled or an unpooled situation and further optimization.

2.1 Motivation and Literature

The question relates to a variety of daily-life situations such as at banks, information desks, ticket offices, up to manufacturing with parts and tools lined up for parallel machines. A question that seems too simple to be asked as the answer seems so obvious. In contrast, however, it appears to be surprisingly intriguing.

Capacity pooling is a common concept. The general perception seems to be in favor of pooling (e.g., see Borst et al. [7], Cattani and Schmidt [8]). From an OR-, or rather queueing-, point of view, it seems less obvious, if not highly intriguing (e.g., Bell and Williams [5]). Counterintuitive examples can already been found in the book of Wolff [34] and Smith and Whitt [26]. Particularly, motivated by present-day developments of so-called skill-based routing; over the last decade, it has been given considerable attention within the application field of call centers (see Gans and Zhou [14], Wallace and Whitt [33]). The insights and results from the field of queueing appear to be essential to steps for performance improvement and optimization, such as by overflow and threshold policies (see van Dijk and van der Sluis [30], Osogami et al. [23], Squillante et al. [27], Wallace and Whitt [33]). Here, overflow mechanisms come in for which simulation becomes necessarily required.

2.2 Queueing Insights

2.2.1 A First Queueing Insight

Indeed, the last perception seems supported by the most standard queueing expression $D = 1/(\mu - \lambda)$ with

μ: the service rate (or capacity) of the server (per unit of time)
λ: the arrival rate (per unit of time) and
D: the average (or mean) delay

with the implicit assumption of exponential service times. (This assumption can be regarded as formal justification for speaking in terms of a service rate.) Pooling two of such servers as if it becomes a twice as faster server with double arrival rate thus seems to reduce the mean delay by 50 % according to $D = 1/(2\mu - 2\lambda)$. This delay reduction seems to result from the efficiency gained by pooling individual queues. The inefficiencies in the nonpooled case are avoided, as one server can no longer be idle while there are still customers waiting at another. The intuitive reasoning above thus seems to be supported by queueing theoretic results and in-line with the general perception that pooling is beneficial.

More precisely, by straightforward calculation from standard M/M/s-expressions, we find

$$\frac{W_E(2,\rho,\tau)}{W_E(1,\rho,\tau)} = \frac{\tau\rho^2/(1-\rho^2)}{\tau\rho/(1-\rho)} = \frac{\rho}{1+\rho}$$

with $W_E(s,\rho,\tau)$: the mean waiting time for an exponential server group with s servers, (that is an M/M/s-queue) with traffic load $\rho = \lambda/s\mu$ per server, with λ the arrival rate and $\tau = 1/\mu$ the mean service time.

This shows that the effect of pooling two parallel servers depends on the traffic load ρ and will indeed lead to a reduction of at least 50 %.

2.2.2 A Second Queueing Insight: Variability Factor

This reasoning, however, relies upon the implicit assumption of statistically identical jobs, identical servers, and equal server loads; however, when services are pooled with different service means, there is also another elementary queueing result that becomes important. This is the factor of the variability of the service times (in addition to just the mean). For example, for the case of a single-server system this is expressed by Pollaczek–Khintchine's famous formula

$$W_G = (1+c^2)W_D - \frac{(1+c^3)}{2}W_E$$

where W_G, W_D and W_E are the expected (average or mean) waiting times for the situation of a general service distribution (G) with squared coefficient of variation c^2, respectively, for a deterministic (or fixed) service time D (hence with $c^2 = 0$) and for an exponential distribution E (for which $c^2 = 1$), and where

c^2 the squared coefficient of variation $= \sigma^2/\tau^2$
σ^2 the variance of the service time;
τ the mean service time.

In words, this formula tells us that also the variation (as expressed by σ^2) around (relatively to) the mean τ of the service times plays an essential factor for the average delay (as compared to the situation of fixed service times).

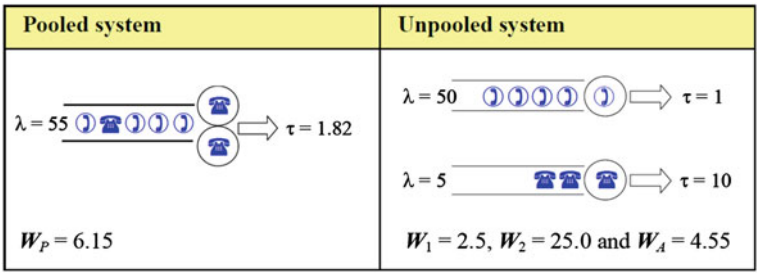


Fig. 1 Two-server example ($k = 10$, $\rho = 0.83$) by OR (queuing)

By pooling different services, due to the mix variability introduced and as by the Pollaczek–Khintchine formula, the effect will thus be less beneficial.

2.2.3 Instructive Example

Consider the situation of two arrival streams of service (e.g., call) requests, referred to as of type 1 and 2, with arrival rate λ_1 and λ_2, and mean service time τ_1 and τ_2, with equal workload $\rho = \lambda_1\tau_1 = \lambda_2\tau_2$, and two servers which can handle either type of service. Let

$k = \lambda_1/\lambda_2 = \tau_2/\tau_1$,
$\overline{\tau} = p_1\tau_1 + p_2\tau_2$, with $p_i = \lambda_i/(\lambda_1 + \lambda_2)$,
W_A Average waiting time for all jobs,
W_1 Average waiting time for type 1 jobs,
W_2 Average waiting time for type 2 jobs,
W_P Average waiting time for the pooled case.

Figure 1 then illustrates the effect of pooling, for example, with $k = 10$, $\lambda_1 = 50$ and $\lambda_2 = 5$ per hour, $\tau_1 = 1$ and $\tau_2 = 10$ min (hence 10 times more short jobs which are 10 times shorter). Furthermore, the service times are assumed to be deterministic and the waiting times are expressed in minutes.

2.2.4 A Pooling Formula

With c^2_{mix} denoting the mix coefficient for the pooled case, as computed by:

$$c^2_{mix} = \frac{p_1(\tau_1 - \bar{\tau})^2 + p_2(\tau_2 - \bar{\tau})^2}{\bar{\tau}^2} = p_1\left(\frac{\tau_1}{\bar{\tau}^2}\right)^2 + p_2\left(\frac{\tau_2}{\bar{\tau}^2}\right)^2 - 1$$

the effect of pooling these two servers is then given by

$$c^2_{mix} = \frac{k}{k+1}\left(\frac{k+1}{2k}\right)^2 + \frac{1}{k+1}\left(\frac{k+1}{2}\right)^2 - 1 = \frac{(k-1)^2}{4k}.$$

$$\frac{W_P}{W_A} \approx \frac{1/2(1+c_{mix}^2)W_E(2,\rho,\bar{\tau})}{1/2W_E(1,\rho,\bar{\tau})} = (1+c_{mix}^2)\left(\frac{\rho}{1+\rho}\right) = \frac{(k+1)^2}{4k}\left(\frac{\rho}{1+\rho}\right)$$

$$\frac{W_P}{W_1} \approx \frac{1/2(1+c_{mix}^2)W_E(2,\rho,\bar{\tau})}{1/2W_E(1,\rho,\tau_1)} = (1+c_{mix}^2)\frac{2k}{k+1}\left(\frac{\rho}{1+\rho}\right) = 1/2(k+1)\left(\frac{\rho}{1+\rho}\right)$$

These expressions directly lead to the following conclusions:

Conclusions 1 (As based on OR (queueing), for the two-server case, deterministic services and identical loads)

1. Pooling is always beneficial for type 2.
2. There can be an increase for a fraction $k/(k+1)$ of the calls for $k > 3$.
3. Pooling is not beneficial for $k > 5$.

Similar results can be just as well for larger server numbers; say instead of 2 single servers for 2 groups of servers each of size s = 5, 10, 20 servers, as of realistic interest for call center dimensioning. For more details, we refer to van Dijk and van der Sluis [30, 31].

More generally, by these rather basic but essential OR (queueing) insights and results, it would seem advantageous to combine the advantages of:

- No (or minimum) idleness as for the pooled case
- No (or minimum) service variability as for the unpooled case.

2.3 Improvement and Optimization by OR and Simulation

2.3.1 Overflow

An overflow system is proposed to further improvement for the overall mean waiting time. This is where *simulation* comes in necessarily. Overflow systems are virtually unsolvable analytically. In Fig. 2, the results by simulation are shown for a two-way overflow (2WO) and a one-way (1WO-1) scenario as specified by

- *Two-way overflow* (*2WO*): A separate queue for each type. An idle server, when there are no jobs of its own type waiting, will take a job waiting of the other type, if any.
- *One-way overflow* (*1WO-1*): A separate queue for each type. Only an idle server of type 2 and if there are no jobs waiting of type 2 will take a job from the other queue, if any.

Figure compares for $s = 1$ (two parallel servers), $k = 10$ (hence with 10 times more short jobs which are 10 times shorter) four basic scenarios of the pooled (P), the unpooled (U), a two-way overflow (2WO) and a one-way (1WO-1) scenario.

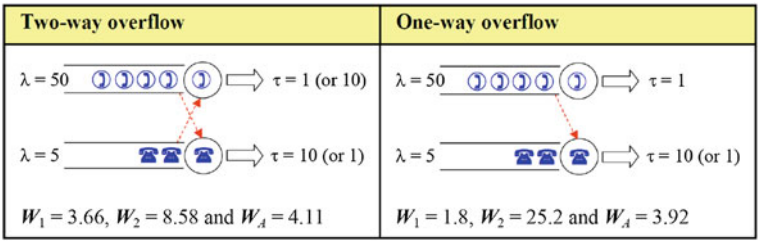


Fig. 2 Two-server example ($k = 10$, $\rho = 0.83$) by OR insights and simulation

Two more simple scenarios to improve the overall average waiting time, as also based on queueing insights, are to prioritize type 1 jobs, either without preemption (service interruption) or with preemption of type 2 jobs when a type 1 arrives. In either way, service for a type 2 job only starts (or is resumed) when no more type 1 jobs are waiting

- *Nonpreemptive-Priority-1* (*NP1*): As in the pooled case and with priority for type 1 jobs when a server idles. Type 2 jobs are served only if there is no type 1 job waiting.
- *Preemptive-Priority-1* (*PP1*): As scenario NP1. In addition, when a type 1 job arrives, a type 2 job is preempted. When no more type 1 jobs are waiting, type 2 jobs are resumed.

By simulation the possible improvement is illustrated in Fig. 3 for the situation with $k = 10$, $\rho = 0.9$ and $s = 10$ (20 servers in total). (To focus on type 1 jobs, the two-way scenario, which would rank in between the unpooled and one-way scenario for the allover average, is left out.)

Rank	Scenario	W_A
5	Pooled	0.71
4	Unpooled	0.63
3	One-way	0.52
2	NP1	0.38
1	PP1	0.20

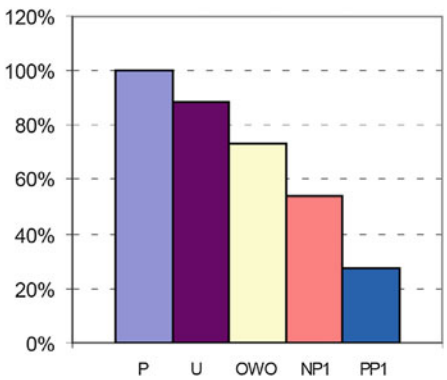


Fig. 3 Average waiting times for different scenarios

Conclusion 2 As based upon OR insights and by simulation, it appears that overflow and priority rules perform substantially better.

2.3.2 Single Threshold Optimization

As shown in Sect. 2.3.1, a simple priority rule, particularly the preemption scenario for short (type 1) jobs, generally seems to perform quite well and to be "optimal" among simple scenarios. Unfortunately, preemption (interruption) of service will generally be impractical. A further improvement step by queueing and Simulation might therefore be proposed by using threshold policies, where $T(\theta_1)$ indicates a threshold policy as specified by

$$T(\theta_1) = \begin{cases} \text{a server of either type serves jobs from queue 1,} \\ \quad \text{if either } (i) \quad m_1 \geq \theta_1 \quad \text{or} \quad (ii) \quad m_2 = 0 \wedge m_1 \geq 1; \\ \text{it serves jobs from queue 2,} \\ \quad \text{otherwise.} \end{cases}$$

More precisely, by exploiting simulation even an optimization can take place by determining

$$W^* = \min_{\theta_1} W(\theta_1).$$

The results, as had to be obtained by simulation, for some optimal threshold values compared to the results for the pooled and NP1 scenarios are shown in Table 3.

In fact, it has also been concluded by simulation that these single threshold policies, with only a threshold value θ_1 for type 1, are nearly optimal among all policies that use threshold values for both type 1 and type 2 servers.

Table 3 Optimal threshold values

s	Pooled	NP1	$\mathbf{T}(\theta_1)$	
	W_P	W_A	W_A	θ_1^*
1	11.53	4.58	4.37	3
2	5.27	2.44	2.27	4
3	3.33	2.63	1.51	4
4	2.34	1.19	1.12	4
5	1.76	0.93	0.88	5
10	0.71	0.38	0.38	1
15	0.40	0.21	0.21	1
20	0.26	0.14	0.14	1
30	0.13	0.07	0.07	1

2.3.3 Double Optimization

So far, an improvement of the average overall average waiting time was obtained by particularly improving the mean waiting time for type 1 jobs. Here the average was computed proportional to the arrival rate ratio for type and type 2 jobs. Though this averaging makes natural sense, the choice of weights for type 1 and 2 jobs is still arbitrary. In all scenarios so far, a price still had to be paid by type 2 jobs (even though for just a small percentage of the jobs).

As an another objective, and supported by insight from queueing, we can address the question whether a scenario can be found that strictly improves the pooled scenario, that is, in mean waiting time, for both type 1 and type 2 jobs. As mentioned in Sect. 2.3.2, for the overall average mean waiting time, an optimization by threshold values basically boiled down to just one threshold value θ_1. For the purpose of a strict improvement, in contrast, also a prioritization of type 2 jobs might thus be expected and be required. Instead of one threshold value θ_1, we will consider threshold rules with one threshold values θ_1 and θ_2 for either type of jobs. More precisely, let the threshold rule $\mathbf{S}(\theta_1, \theta_2) = \mathbf{Thr}(\theta_1, \theta_2, \theta_1, \theta_2, 1, 1)$, i.e., as specified by:

$$\mathbf{S}(\theta_1, \theta_2) = \begin{cases} \text{a server of type 1 serves jobs from queue 2,} \\ \quad \text{if either } (i) \quad m_2 \geq \theta_2 \wedge m_1 < \theta_1 \text{ or } (ii) \quad m_1 = 0 \wedge m_2 \geq 1; \\ \quad \text{otherwise, it serves jobs from queue 1.} \\ \text{a server of type 2 serves jobs from queue 1,} \\ \quad \text{if either } (i) \quad m_1 \geq \theta_1 \wedge m_2 < \theta_2 \text{ or } (ii) \quad m_2 = 0 \wedge m_1 \geq 1; \\ \quad \text{otherwise, it serves jobs from queue 2.} \end{cases}$$

Among these dynamic (or queue length dependent) rules $\mathbf{S}(\theta_1, \theta_2)$ by simulation a rule is sought which strictly improves the pooled scenario. An $\mathbf{S}$(Opt)-rule is determined that takes into account the waiting times of both job types by:

Step 1: Solve

$\min_{\theta_1,\theta_2} \max \{\mathbf{W}_1[S(\theta_1, \theta_2)], \mathbf{W}_2[S(\theta_1, \theta_2)]\}$

This leads to an optimal threshold combination $(\theta_1, \theta_2)^*$

and overall average waiting time under $(\theta_1, \theta_2)^*$: $\mathbf{W}_A[S(\theta_1, \theta_2)^*]$.

Step 2: Solve

$$\mathbf{W}_A[S(\theta_1, \theta_2)^{**}] = \min_{\theta_1,\theta_2} \mathbf{W}_A[S(\theta_1, \theta_2)] \quad \text{s.t.} \quad \max \{\mathbf{W}_1[S(\theta_1, \theta_2)], \mathbf{W}_2[S(\theta_1, \theta_2)]\} \leq \mathbf{W}_{Pooled}$$

Step 3: If $(\theta_1, \theta_2)^{**}$ exists, then

$$\mathbf{W}_A[S(\text{Opt})^{**}] = \mathbf{W}_A[S(\theta_1, \theta_2)^{**}],$$

otherwise

$$\mathbf{W}_A[S(\text{Opt})^{**}] = \mathbf{W}_A[S(\theta_1, \theta_2)^{*}].$$

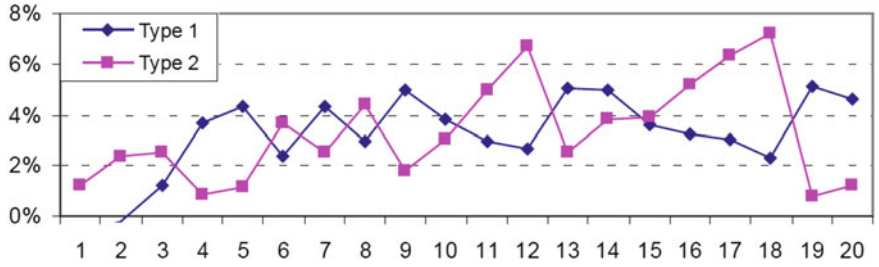


Fig. 4 Relative improvements over the pooled scenario

k \ s	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	(4,2)	(4,2)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(3,2)	(3,2)	(3,2)	(3,2)	(3,2)
3	(4,1)	(4,1)	(3,1)	(3,1)	(3,1)	(5,2)	(5,2)	(5,2)	(5,2)	(5,2)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)	(2,1)
4	(5,1)	(7,2)	(4,1)	(4,1)	(6,2)	(6,2)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(5,2)	(5,2)	(2,1)	(2,1)	(2,1)	(2,1)
5	(9,2)	(6,1)	(5,1)	(5,1)	(4,1)	(4,1)	(4,1)	(7,2)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(6,2)	(6,2)	(6,2)
6	(8,1)	(7,1)	(6,1)	(5,1)	(5,1)	(5,1)	(4,1)	(4,1)	(4,1)	(4,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)
7	(13,2)	(8,1)	(7,1)	(6,1)	(10,2)	(5,1)	(5,1)	(4,1)	(4,1)	(4,1)	(4,1)	(4,1)	(4,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)
8	(14,2)	(9,1)	(8,1)	(7,1)	(6,1)	(11,2)	(5,1)	(5,1)	(4,1)	(4,1)	(4,1)	(4,1)	(4,1)	(9,2)	(3,1)	(3,1)	(9,2)	(9,2)	(9,2)	(9,2)
9	(12,1)	(15,2)	(9,1)	(8,1)	(7,1)	(6,1)	(6,1)	(5,1)	(5,1)	(5,1)	(5,1)	(4,1)	(4,1)	(4,1)	(4,1)	(4,1)	(4,1)	(3,1)	(3,1)	(3,1)
10	(18,2)	(12,1)	(15,2)	(8,1)	(7,1)	(7,1)	(6,1)	(6,1)	(5,1)	(5,1)	(5,1)	(5,1)	(4,1)	(4,1)	(4,1)	(4,1)	(4,1)	(4,1)	(3,1)	(3,1)

(θ1, θ2)** solution (θ1, θ2)* solution (θ1, θ2)* solution, pooling average better

Fig. 5 Optimal threshold combinations $(\theta_1, \theta_2)^{**}$ or $(\theta_1, \theta_2)^{*}$ for strict improvements

The improvements are only in the order of a few % but consistently outside 95 % confidence intervals with a range of 0.5 %. Figure 4 shows the relative improvements (mean waiting time reduction) that can so be obtained for both type 1 and type 2 jobs over the pooling scenario for $k = 10$ and $s = 1$ up to 20.

Figure 5 lists optimal threshold combinations $(\theta_1, \theta_2)^{**}$ (if existing), for which the pooled scenario is improved allover, and optimal threshold combinations $(\theta_1, \theta_2)^{*}$ otherwise, for different values of s, mix ratios k and $\rho = 0.9$. It shows that $(\theta_1, \theta_2)^{**}$ does not always exist. For example, for $k = 10$ and $s = 2$, at least one of the two job types will always be worse than for the pooled case. However, for most (s, k)-values $(\theta_1, \theta_2)^{**}$ appears to exist.

Conclusion 3 By OR (queueing) insights and Optimization using Simulation a strict improvement over both short and long jobs might be feasible.

2.4 Summary of Combined Queueing and Simulation

To summarize this first application section of combined queueing and simulation, tailored to question of pooling and possible improvements and optimization, we may thus conclude that:

- Pooling is not necessarily optimal in all situations.
- Queueing insights appear to be essential.

- Simulation is required.
- Queueing insights and further optimization by simulation may lead to substantial and even strict improvements.

3 Blood Inventory Management

Blood management is of worldwide and generic concern. This includes the production (or rather acquisition of donors) and the inventory management of perishable blood platelets. No general and practical approach seems to be available. In this section, therefore, an integrated OR-simulation approach is provided.

3.1 Problem Motivation

Blood inventory management is a problem of general human interest with a number of concerns and complications. Our problem of interest will concentrate on the production and inventory management of blood platelets. Here there are a number of conflicting aspects. On the one hand, the demand is highly "uncertain" and apart from planned surgeries (if such information is used) roughly 50 % of the demand is unpredictable. Clearly, as lives may be at risk, shortages are to be minimized. On the other hand, the supply is voluntary, and also for ethical reasons blood has to be considered as highly precious. Any spill, by outdating, of blood (products) is thus highly "undesirable" if not to be avoided at all. As an extra complicating factor, blood platelets (thrombocytes) have a limited life time or rather "shelf life" of at most 6 days, this in contrast to red blood cells and plasma in all sorts of blood types that can be kept for months up to over a year. In addition, regular production of a platelet pool takes about 1 day. Hence production volumes should be set carefully. Another complicating factor is that part of the patients need the youngest platelets available, whereas other patients can be transfused with any platelets that do not exceed their shelf life of 5 or 6 days. Figure 6 shows the product of interest; to help one patient, platelets of five donors are needed.

3.2 Literature

The above perishable inventory management problem is studied in literature using various techniques. In the late 1960s and 1970s of last century, the problem is first analyzed by mathematical analysis of rather simple models that assume zero lead time, stationary demand, and that neglect the existence of different groups of patients, etc., see Nahmias [22], Prastacos [24]. More realistic studies use simulation models to gain insights in the performance of base stock policies, see o.a. Katsaliaki and Brailsford [18], Sirelson and Brodheim [25].

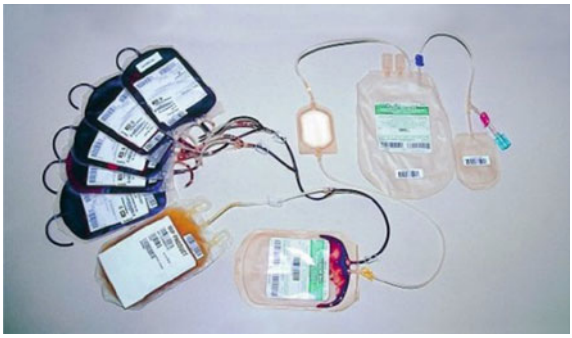


Fig. 6 A single platelet pool consists of platelets of five donors

Base stock policies set the order quantity equal to the difference between the actual stock level (or stock position) and a fixed-up-to level S. The value of parameter S is weekday dependent as the practical problem is nonstationary: Mean demand are weekday dependent and no production happens during the weekends. Optimizing the parameters of order policies is usually very time consuming as the for each day of the week an optimal parameter setting has to be found and these parameters are correlated. The number of combinations is often too large to apply enumerated search using simulation. A recent study that applies simulation based optimization using meta heuristics is Duan and Liao [12].

It is known that an optimal order policy should consider the ages of the products in stock, see Fries [13], Nahmias [22]. Base stock policies neglect stock ages but nevertheless they are commonly applied for being relatively easy to implement and to analyze. The optimality gap is hardly studied for realistic problem settings with positive lead time, nonstationary/periodic demand, and multiple types of patients who require different issuing policies. Main reason is the computational complexity involved in determining an optimal stockage-dependent policy Blake et al. [6]. This gap is investigated in the following papers: Haijema et al. [16, 17], Van Dijk et al. [32]. In these papers, optimal stockage-dependent order policies are derived and methods are presented that use simulation in combination with optimization to derive improved but simple ordering policies as well as a way for finding nearly optimal order-up-to-levels. In this chapter we summarize and integrate the findings of these studies.

3.3 Combined Optimization-Simulation Approach

In Haijema et al. [16] a combined approach for the blood platelet inventory problem has therefore been followed, which combines OR and simulation by the following steps:

Step 1: Optimization model: First, a stochastic dynamic programming (SDP) formulation is provided, which neglects the existence of blood types. This latter assumptions will be validated in Step 5.

Step 2: Optimal solution: The dimension of the (SDP) formulation is then reduced (downsized) by aggregating the state space and demands so that the downsized (SDP) problem can be solved numerically (using successive approximation). That is, the optimal value and an optimal strategy is determined for the downsized SDP.

Step 3: Simulation for investigation: Then, as essential tying step, this optimal policy is (re)evaluated and run by simulation in order to investigate the structure of the optimal strategy. In this simulation, one registers the frequency of (state, action)-pairs for the downsized problem.

Step 4: Simulation for re-optimization: The results of step 3 are used to derive practical order rules, like improved base stock policies and to obtain nearly optimal parameter values. By a heuristic search procedure parameter values of these rules are fine tuned for the full-size problem.

Step 5: Simulation for validation: The quality (near-to-optimality) of this practical simple order-up-to strategy is evaluated by detailed simulation. In this step it is also justified, for Dutch blood banks, that blood types are ignored in the previous steps.

As the technical (mathematical) details of steps 1 and 2 are somewhat 'standard' but also 'complicated' and worked out in detail in Haijema et al. [16] and related references, we present here a compact presentation of the essential OR and Simulation Steps. The results of Step 5 (validation by simulation) are reported for two cases in Sects. 3.4 and 3.5.

3.3.1 Steps 1 and 2 Optimization by SDP

To give an SDP formulation, the state of the system is described by (d, x) with

d: the day of the week $(d = 1, 2, \ldots, 7)$

and

$\mathbf{x} = (x_1, x_2, ., x_m)$ the inventory state

with x_r = the number of pools with a residual life time of r days (maximal $m = 6$ days) (A pool is one patient-transfusion unit containing the platelets of five different donations).

Let $V_n(d, x)$ represent the minimal expected costs over n days when starting in state (d, x). The optimal inventory strategy and production actions are then determined by iteratively computing (solving) the SDP-equations for $n = 1, 2, \ldots$

$$\mathbf{V}_n(d, \mathbf{x}) = \min_k \left[c(\mathbf{x}, k) = \sum_b p_d(b) \mathbf{V}_{n-1}(d, t(\mathbf{x}, k, b)) \right] \tag{1}$$

Table 4 Optimal productions by SDP for a selection of states

Production	Inventory (old . . . young)
7	(0, 0, 5, 0, 0, 9)
8	(0, 0, 6, 0, 0, 8)
9	(0, 0, 8, 0, 0, 6)
10	(0, 6, 2, 0, 0, 6)
10	(5, 0, 3, 0, 0, 6)

with

k	the production action,
$c(\mathbf{x}, k)$	the one day expected costs in state $\mathbf{x}$ under production k,
$pd(b)$	the probability for a (composite) demand b,
$t(\mathbf{x}, k, b)$	the new inventory state depending on k, b, $\mathbf{x}$, and some issuing policy,

and

$\mathbf{V}_0(d, \mathbf{x}) = 0$ to start up the iterative computations.

However, for a realistically sized problem for one of the Dutch regional blood banks the computational complexity of this SDP for a one-week iteration already becomes of an order 1014, which makes the computation times prohibitively large. Therefore, we have downsized the demands and inventory levels by aggregating the pools into quantities of four. This strongly reduces the computational complexity, so that an optimal strategy can be computed for this downsized problem by the optimizing actions of the SDP. However, in practice one needs a simple rule and this optimal strategy has no simple structure. See, for example, Table 4 which prescribes the production volumes on Tuesday for five different states, which all have the same total inventory level of 14 pools, but of varying ages.

3.3.2 Steps 3 and 4 Simulation for Investigation and Re-optimization

In order to derive a simple order-up-to strategy which only depends on the total predicted inventory, the actual platelet production-inventory process is therefore simulated for 100,000 replications so as to register how often which total predicted final inventory level (I) and corresponding action occurs under the optimal strategy (as determined by SDP) for the downsized problem. As an illustration, for a particular day of the week and the dataset of the regional blood bank, this led to the "simulation table" in Fig. 7. For example, it shows by row 15 and column 7 that during the 100,000 replications 2593 times a state was visited with a total final inventory (I) of 7 followed by a production decision of 8 (order-up-to 15). Order-up-to-level 15 occurs in 74.5 % of the states visited, however, often a higher production is optimal. The order-up-to level can be seen as a target-inventory level for Wednesday mornings.

I	2	3	4	5	6	7	8	9	10	11	12	13	14	cum.
Order-up-to														
23													4	4
22												28		28
21											96			96
20										267				267
19								2	748	3				753
18							18	1928	31	1				1978
17						6331	4490	353	26	1				11201
16					8260	2078	783	7						11128
15	3131	14123	20926	23646	10087	2593	39							74545
:														
0														
cum.	3131	14123	20926	23646	18347	11002	5330	2290	805	272	96	28	4	100000

Fig. 7 Simulation frequency table of (State, Action)-pairs for tuesdays from simulation of optimal SDP solution for 100,000 weeks

We conclude that a simple order-up-to rule might perform well. By investigating the states at which the optimal production volume is higher we have derived even better rules that closely resembles the optimal production strategy. For example, a base stock policy that first estimate the quantity that is left upon replenishment is doing better as it compensates for estimated waste during the lead time. Such a policy is called in Haijema et al. [16], the final stock rule. Another improved policy applies two order-up-to levels, one for the demand for young platelets, and one for the total demand. Both these improved policies are discovered and tested by simulation of the optimal stockage-dependent policy.

3.3.3 Step 5 Validation by Simulation

The results of Step 5 are reported for two real-life applications that differ in their motivation. Application 1 was selected for validation of the method with the premier objective of reducing waste. In Application 2 the focus is on applying the method such that one issues younger product while maintaining low levels of waste and a high product availability. The Netherlands can be divided in four regions at which blood is collected and processed, see Fig. 8. For Application 1, region North-East is considered, for Application 2 region South-East is selected.

Fig. 8 The Dutch blood banks divide The Netherlands in four regions

3.4 Application 1: Spill Reduction at Dutch Blood Bank North-East

3.4.1 Main Results

Applying this combined approach to data from the Dutch regional Blood Bank North-East, the following conclusions could be drawn

1. A simple order-up-to rule could reduce the spill from roughly 15–20 %, as a figure that also seems rather standard worldwide, to $<\mathbf{1}$ % (while also shortages were reduced and nearly vanished).
2. The combined SDP-Simulation approach led to an accuracy within **1** % of the exact optimal value for the downsized problem.

Detailed data and results are discussed below.

3.4.2 Problem Data Dutch Blood Bank North-East

The maximal shelf life of a platelet pools is five days counted from the first morning that platelet pools are released to the stock located at the blood bank. The demand for platelet pools is Poisson distributed with means as reported in Table 5.

The demand for young prefers products of at most 3-days old. The any-age demand can be met by any pools of at most 5-days old. Falling short one pool is considered to be five times as severe as wasting one platelet pool. This is a managerial trade-off

Table 5 Means of Poisson demands per weekday and per type of demand

Demand	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Weekly
'Young'	20	15	26	15	20	0	0	96
'Any-age'	6	6	6	6	6	8	10	48
Total	26	21	32	21	26	8	10	144

that is reflected in a penalty costs of 150 € for spilling one pool, and 750 € for falling short one pool. Inventory costs are estimated to be only 0.1 € per day per pool. Meeting the demand for young by products with a residual shelf life of only 2 or less days is penalized by a cost of 200 per pool. The objective is to minimize the sum of these costs.

3.4.3 Results Dutch Blood Bank North-East

An optimal stockage-dependent policy can be obtained by SDP but only after scaling the demand figures as if demand happens in multiples of 4 pools (=Step 2). The resulting policy is simulated to investigate its structure (=Step 3). The result of Steps 3 and 4 are five more rules, which neglect the age of the products in stock while setting an order quantity. The performance of these rules are compared in Table 6, using the scaled or downsized demand distributions. Clearly stockage-dependent ordering (SDP/MDP) gives lowest annual costs. A *fixed-order quantity* orders every weekday a fixed weekday dependent quantity and is clearly far from optimal. The *Order-up-to S*, which is a base stock policy, provides annual costs that are 9.9 % above the optimal cost level. For the scaled problem, the weekday dependent order-up-to levels are multiple of 4; for Monday to Friday we get $S = (64, 72, 72, 64, 80)$. The new policy *Bounded Order-up-to S*, adds a minimum and a maximum order quantity to order-up-to S policy. The effect of these bounds are that are quantities are more smooth, which results in lower annual costs, primarily due to generating less mismatches, i.e., it happens less frequently that old pools are used to meet the

Table 6 Performance of optimal policy and derived rules

Rule	Outdating[a]		Shortage[a]		Mismatch[b]		Annual costs	
MDP-optimal	37	1.95 %	4.9	0.26 %	0.09	0.01 %	36,605	–
Fixed-order-quantity	157	8.41 %	1.4	0.07 %	0.00	0.00 %	98,459	+168.9 %[c]
Order-up-to S rule	36	1.95 %	5.7	0.30 %	2.17	0.17 %	40,236	+9.9 %[c]
Bounded order-up-to S	36	1.95 %	5.6	0.30 %	0.59	0.05 %	39,077	+6.8 %[c]
2D-order-up-to rule	36	1.92 %	5.2	0.28 %	2.16	0.17 %	38,779	+5.9 %[c]
Final stock rule	36	1.91 %	5.1	0.27 %	1.98	0.16 %	38,389	+4.9 %[c]

[a] In batches of 4 pools per year; % of total (1872 batches = 7488 pools)
[b] In batches of 4 pools per year; % of young-demand (1248 batches = 4992 pools)
[c] Percentage above optimal cost level (MDP)

demand for young pools. The *2D-rule*, which has both an order-up-to level for young and for total stock, and the *Final stock rule* show further cost reductions. The best policy is still about 5 % above the minimal costs level achieved by MDP-optimal.

3.4.4 Re-optimization and Validation

For the validation, we restrict ourselves to the order-up-to S policy as it is commonly used. The parameters are reoptimized by local search and simulation with the nonscaled demand distributions resulting in $S = (65, 74, 80, 64, 82)$. Validation happens in a more detailed simulation program that takes into account the blood type of both patients and donors. Donors are selected mainly form the category O and A for being the most compatible donor, see Fig. 9. The percentages indicate that the two blood types cover 89 % of the population. Most donor provide full blood donations from which three types of blood products are made: Red blood cell concentrates, plasma products, and platelet concentrates. As for the production of platelet pools only a third to a half of the donations is needed, one usually has enough platelets available of the most compatible blood types O and A. In total, we consider eight blood types by combining O, A, B, and AB with the Rhesus-D factor.

Table 7 report estimates of annual figures by two simulation models: The multi-group model simulates patients and donors of eight different blood types; the Universal-group model simulates as if all donors and patients have identical or fully compatible blood types. The result in annual performance is virtually the same, as blood of the universal blood group O is plenty available.

If instead of 33 % of the available blood is sued for platelet production, 50 % or even 67 % is used more of other blood types is used for production. This is demonstrated in Table 8. The annual production stays the same but if blood is more scarce, more of the less compatible blood type B is produced which is no problem if demand is met choosing pools of the least favorite but compatible blood type first.

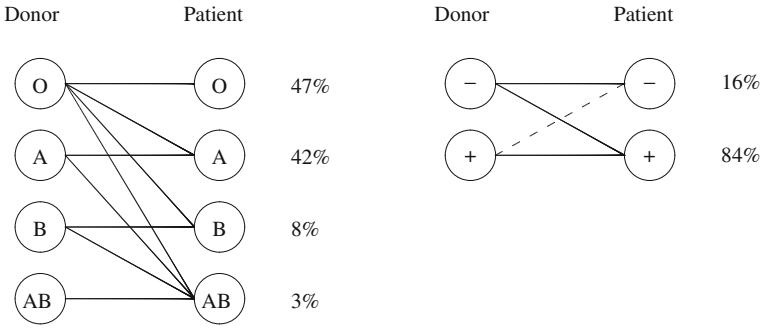


Fig. 9 Compatibility of blood types and Rhesus-D factor

Table 7 Impact of blood types on performance of order-up-to S (simulation for 100 million weeks)

Criterion	Multigroup model		Universal-group model	
	In pools	Relative (%)	In pools	Relative (%)
Production	7597		7597	
Outdating	143	1.9	142	1.9
Shortage	33	0.4	33	0.4
Quality mismatch	9	0.2	9	0.2
Annual costs	48,168		47,726	

Table 8 Production per blood group under the rescaled order-up-to S rule (over 100 simulation runs of 1 million weeks each) when on average a third, a half or two third of the whole blood donations (WBD) is used for platelet production

Scenario	Total	O^-	O^+	A^-	A^+	B^-	B^+	AB^-	AB^+
33 % of WBD used	7597								
Annual production		1582	5944	62	10	0	0	0	0
% of total production		20.8 %	78.2 %	0.8 %	0.1 %	0 %	0 %	0 %	0 %
50 % of WBD used	7597								
Annual production		1018	5268	525	786	0	1	0	0
% of total production		13.4 %	69.3 %	6.9 %	10.3 %	0 %	0 %	0 %	0 %
67 % of WBD used	7597								
Annual production		739	4201	560	1991	8	86	0	14
% of total production		9.7 %	55.3 %	7.4 %	26.2 %	0.1 %	1.1 %	0.0 %	0.2 %

3.5 Application 2: Age Reduction at Dutch Blood Bank South-East

As donated, platelets tend to clutter and the number of effective platelets within a pool decreases as time evolves; the quality of a platelet pool is directly related to its age when its transfusion takes place. Therefore, besides shortages and outdating, there is a third quality factor.

The issuing age of the platelets.

This quality factor is most important for treatment of special patients (oncology and hematology) which constitute roughly 40 % of all demand. Clearly, the SDP by itself does not take the age into account. By simulation, in contrast, issuing ages can easily be kept track of. In Kortbeek et al. [19] and Haijema et al. [16], therefore, the SDP-simulation approach was extended so that also the quality aspect of the issuing age is addressed. The extension was applied to a new Dutch Blood Bank study, the Dutch Blood Bank South-East, with a (meanwhile) extended maximal shelf life of 6 days. Below, several strategies are presented that improve the age of the platelets issued, such as by slightly relaxing the shortage performance, by introducing penalty costs for older issues or by a special "weekend" production. The results were obtained by successfully exploiting the strengths of

Table 9 Performance of base case South-East

Performance indicator	Base case
Shortage	0.04 % (in days 0.13 %)
Outdating	0.25 %
Average age	3.75
Age distribution	(2.5; 17.2; 19.1; 30.8; 25.1; 5.3) %

- SDP for optimization, and
- Simulation for evaluation.

In the first study only two cost elements are used, outdating and shortage cost. Shortage costs are taken to be five times as high as the outdating costs. In this base variant, there are no penalty costs with respect to age. The issuing policy is FIFO, that is, the oldest platelets are issued first. At Saturday there is a limited production capacity of 20 pools, and at Sunday there is no production at all.

Table 9 shows that the shortage and outdating figures are excellent. The issuing age of the platelets, however, is fairly high, with more than 30 % of the platelets being issued at shelf ages 5 and 6 days and with an average age of 3.75 days.

At this point, the SDP-Simulation approach is exploited in order to explore scenarios and strategies which can improve the issuing age. An obvious first attempt is to use the LIFO (Last In First Out) issuing policy instead of FIFO, so to always issue the youngest platelets in stock. Although the average issuing age improves to 2.26, the price with respect to outdating (11.0 %) and shortages (1.87 %) is very high. Therefore it was concluded that LIFO is not the solution. A second alternative is to allow the number of days with shortages to be relaxed to about 1 % (recall this percentage was 0.13 for the base case). 1 % amounts to 3 to 4 days per year and is considered to be acceptable by the Blood Bank. By allowing more shortages one will keep fewer inventories, which might result in issuing younger platelets. In order to find a nearly optimal order-up-to strategy giving 1 % shortage days, the ratio between shortages and outdating costs is decreased. The results are displayed in Table 10.

There is a considerable improvement of the age distribution, while outdating has become virtually zero. Only 14 % of the issued platelets is of shelf age 5 and 6 days, compared to 30 % for the base case and the average age has been decreased from 3.75 to 3.18. In the base case, the maximal shelf life is 6 days. But what happens if one decides not to use platelets of age 6 days, so that platelets become outdated at the end of day 5 or even at the end of day 4? It can be expected to lead to lower order-up-to levels, so more shortages but issuing younger platelets. With respect to the SDP this results in changes in the state space, the expected costs and the transition probabilities. Using the same cost structure as in the base case one obtains the results in Table 11. The results for a shelf life of 5 days for shortages and outdating are quite good. The percentage of the platelets issued at shelf age 5 is about 16 % and the average age of the platelets issued is 3.2 days. Although for 4 days the age distribution improves considerable, the increase in outdating makes this solution unacceptable.

Table 10 Performance South-East, when shortages are tuned to 1 % of days

Performance indicator	Shortage are tuned to 1 %
Shortage	0.35 % (in days 0.95 %)
Outdating	0.02 %
Average age	3.18
Age distribution	(8.4; 25.9; 20.3; 31.2; 13.1; 1.1) %

Table 11 Performance South-East, when shelf life is reduced

Performance indicator	5 days	4 days
Shortage	0.24 % (in days 0.66 %)	0.73 % (in days 1.88 %)
Outdating	1.22 %	5.38 %
Average age	3.23	2.70
Age distribution	(6.8; 25.8; 20.2; 31.5; 15.6, −) %	(15.9; 29.5; 23.0; 31.6, −, −) %

Another possibility is to discourage the issuing of older platelets by penalization in the cost function of the SDP. This penalty is taken to be half the outdating costs. (It is important to note that the more cost parameters are used, the more difficult it is to quantify them in such a way that the effect one is aiming for is indeed achieved.) Compared to the base case, the only change in the SDP is in the costs. Two cases are considered: In the first case day 4, 5, and 6 are discouraged, and in the second case day 5 and 6. The results are displayed in Table 12.

Both cases show almost equal results. As expected, shortages have increased, but the average age went down to 3.12, and issues of day 5 and 6 halved compared to the base case. The final scenario studies a combination of successful scenarios: a shelf life of 5 days, a penalization for issuing platelets of age 5, and shortages about 1 % in days. The results are displayed in Table 13.

The proposed combination appears to be a very satisfactory improvement, with shortages in the order of 1 % in days, outdating just below 1 %, the average age reduced from 3.75 to 3.06 and only 11.0 % issued at age 5.

Table 12 Performance South-East, when discouraging issuance of older pools

Performance indicator	Penalize day 4, 5 and 6	Penalize day 5 and 6
Shortage	0.30 % (in days 0.95 %)	0.33 % (in days 1.11 %)
Outdating	0.04 %	0.08 %
Average age	3.12	3.10
Age distribution	(9.5; 26.0; 23.1; 27.6; 12.4, 1.5) %	(9.9; 26.7; 23.5; 25.1, 12.7, 2.0) %

Table 13 Performance South-East, when combining scenarios

Performance indicator	Combination
Shortage	0.36 % (in days 1.04 %)
Outdating	0.92 %
Average age	3.06
Age distribution	(9.0; 27.0; 24.0; 29.0; 11.0; −) %

3.6 Summary Blood Inventory Management

To summarize this section it can thus be concluded that the (perishable inventory) problem is so complex that it is impossible to obtain practical results by only SDP or only by Simulation. However, substantial and practical improvements could be obtained (and have real life been implemented (see Kort et al. [11], Van Dijk et al. [32]) by their combination.

4 Rail-Track Scheduling

Railroad scheduling is highly complex as punctual and detailed scheduling at minute basis is confronted with stochastic disruptions on the other. Simple and practical rules are then required. These in turn are highly situation dependent, including time-tabling, frequencies up to (country) infrastructures. No general stochastic appears to be available approach other than simulation. In this section, again an integrated OR-simulation approach is suggested, as based upon Stochastic (Semi-Markovian) Dynamic Programming (SDP).

4.1 Motivation

An example of yet another class of stochastic decision problems for which a combined SDP-simulation approach seems most fruitful is found in rail-track scheduling. In the Netherlands, tracks are heavily used. This implies that these tracks have to be used in an intelligent way. As an example, consider the junction as depicted in Fig. 10. If two or more trains enter the junction more or less simultaneously it has to be decided which train is admitted first. To a certain extent the basis of this decision problem is deterministic but in practice it is also highly stochastic due to stochastic arrival times, delays and speed differences. Accordingly, the problem of online dynamic conflict resolution has the flavor of both a scheduling and a queueing problem.

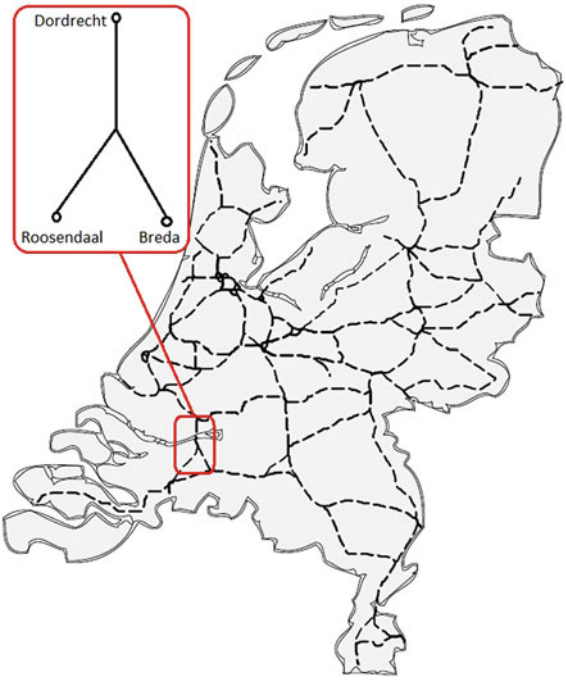


Fig. 10 The railway infrastructure of The Netherlands with selected junction for the test case

4.2 Literature

Railway scheduling problems have been extensively studied in the literature and are known to be NP-hard (Garey and Johnson [15]). Excellent overviews are given in Assad [4], Cordeau et al. [9], Törnquist [28] and D'Ariano [10]. In the overview paper Törnquist [28], the relevant literature is classified into three main categories: Tactical scheduling, operational scheduling, and rescheduling. While the Tactical and operational scheduling involved constructing the timetable from scratch, rescheduling is done when train conflicts arise due to perturbations. The online dynamic conflict resolution falls in the category of Rescheduling.

Very little literature exists on Rescheduling. The issue has been addressed only recently due to the complex nature of the problem and the very limited available computational time. The different approaches that are described in the literature minimize delay propagation by setting the train order at crossing points. Amongst these approaches is the model proposed in Adenso-Dıaz et al. [1]. The authors describe the online conflict resolution problem as a mixed integer programming model and state that solving this problem by means of the Branch-and-Bound technique is very time consuming. Instead, the authors propose a heuristic approach that intelligently reduces the search space by elimination of certain branches that are considered to be inferior. The approach is implemented at the Spanish national railway company where the tool preselects the best resolution rules and presents them to a train dispatcher.

Tornquist and Persson [29] propose a two level procedure to resolve train conflicts. The authors suggest an approximation strategy, which in most cases does well with respect to computational time and solution quality. Araya et al. [3] formulate the online scheduling problem as a 0–1 mixed integer programming problem which is solved in two steps. First, a suboptimal solution is obtained by a heuristic approach. The branch-and-bound approach is then used to find the optimal solution. A number of experiments show the efficiency of the approach in terms of computational time. Another approach is to formulate the train conflict problem as a Job-Shop problem. Here, the trains are jobs and the tracks are machines. The problem is then to find the best assignment of the trains to the tracks so that the overall delay (or some other optimization function) is minimized. Mascis and Pacciarelli [21] introduce blocking and no-wait constraints to the Job-Shop scheduling problem and use an 'Alternative graph' to solve it.

These heuristics, however, do not guarantee the optimality of the solution. Moreover they do not account for future uncertainties, such as stochastic train arrivals. In the next section, a stochastic approach is discussed. These approaches attempt to model uncertainties which are found in the real world (think of the running times, dwell times and other operations which are often stochastic).

4.3 Combined Simulation and Optimization Approach

4.3.1 OR-Approach: Optimization

This track conflict problem can partially be regarded as a 'standard' OR-scheduling problem, more precisely, as a job-shop problem with blocking. By considering trains as jobs and tracks as machines, an 'optimal train order' for a track can be found by branch-and-bound techniques. It is a job-shop problem with blocking because an occupied track section blocks a successive train to enter that section. Trains at the preceding section can thus be delayed. The usual job-shop formulation, however, uses fixed handling times without delays and variability's.

4.3.2 Simulation Approach

As delay aspects and the variability of travel times are crucial for the track conflict, a stochastic approach might be able to cover more aspects of the problem. Simulation would thus be in place, despite the fact that it does not optimize at all. Indeed, in earlier literature (see references in Al-Ibrahim [2]) simulation is used to analyze a junction. In those studies, train are assigned by dynamic priorities. The dynamic priority can be a function of the train type, its experienced delay, the delay caused by acceleration and possible other conflicts. However, optimization is not involved.

4.3.3 Combined Approach

In Al-Ibrahim [2], therefore, a more extended combination of OR and simulation is suggested. To include both queueing (time) and scheduling (optimization) aspects a Semi-Markov Decision Process (SMDP) is formulated.

4.3.4 SMDP-Formulation

The Semi-Markov Dynamic Programming (SMDP) formulation for the stochastic junction-track scheduling problem essentially takes into account the stochastic nature and different durations of transitions. It has the form:

$$V_{n+1}(A,v,d) = \min_k \left\{ \begin{array}{l} c(A,v,d) + \\ (\tau/\tau^k(A,v,d)) \sum_{(A',v',d')} P^k\left[(A,v,d);(A',v',d')\right] V_n(A',v',d') \\ + \left[1 - \tau/\tau^k(A,v,d)\right] V_{n+1}(A,v,d) \end{array} \right\} \tag{2}$$

where a state (A, v, d) represents a state of the form:

$(A, v, d) = (A_1, v_1; A_2, v_2; d_1, d_2, \ldots, d_N)$
with
A_l denoting the trains in queue $l \in \{1, 2\}$
v_l indicating whether the trains are moving ($v_l = 0$) or not ($v_l = 1$)
d_j the train type which is occupying the jth position past the junction.

The costs $c(A, v, d)$ cover the time that all trains together are spending in the subnetwork up to a next transition. Further

$P^k\left[(A, v, d); (A', v', d')\right]$ represents the transition probability from a state (A, v, d) into (A', v', d')

$\tau^k(A, v, d)$ is the average duration of a transition in state (A, v, d), when decision k is taken.

4.3.5 Simulation-SMDP Approach

A combined approach can now be suggested, which combines simulation with the SMDP optimization algorithm in a number of steps, as briefly outlined below.

Step 1: (SMDP optimization) For the junction under consideration, a semi-Markovian decision process is formulated and solved. (As shown above and argued in more detail in Al-Ibrahim [2]). For every possible, state an optimal decision is registered.

Step 2: (Simulation) Trains are generated for the junction subnetwork according to a global train schedule but with a number of stochastic elements to include initial randomness and speed differences. The trains are simulated until a conflict is detected. The simulation run is interrupted and the conflict is registered.

Step 3: (Finding the optimal SMDP decision) The train conflict situation is mapped on to a state of the SMDP model. Then the corresponding optimal decision is read and communicated to the simulation. The simulation implements this decision and the simulation continues until the next conflict occurs.

Step 4: The delays, as they occurred in the simulation with the optimal SMDP-decisions, are registered.

Step 5: Comparison. Simulation is used to also obtain the performance of a number of other heuristical rules to settle the conflicts.

Step 6: Results. The SMDP results as well as the results for the heuristics are reported.

In short, simulation is used to capture queueing, to generate conflicts and to evaluate decisions made while SMDP is used to determine the (within the model) optimal train order.

4.4 Application Results

4.4.1 Application 1: Junction Case

In cooperation with "ProRail" (the Dutch Railway operator) the approach has first been applied to a small but complicating and generic junction within The Netherlands. The junction has 12 arriving trains per hour, 6 fast passenger intercity trains (IC) and 6 slow freight trains (FR); half of the trains on each one of the arriving tracks. After the junction there are 5 positions (which reflect a distance of more than 13 km). The FR trains need 170 s to accelerate from speed 0 to speed 80 km/h, while the IC trains only need 30 s to reach the speed of 120 km/h.

To verify that the combined SMDP-simulation approach outperforms simple practical rules like the FCFS (First Come First Served) rule or a strict priority rule for passenger or for freight trains, the approach is compared with these rules by simulation. Table 14 shows the results. The values are average delays per train type over 12 days at 15 h a day. The results show that the SMDP-simulation approach almost captures the quality for passenger trains as by strictly prioritizing passenger trains and for freight trains as by strictly prioritizing freight trains.

4.4.2 Application 2: A Network Case

Next, in close cooperation with the department of "Traffic Control" of the Dutch Railway operator ProRail the approach has been applied to a more complicated network structure as shown in Fig. 11, called the corridor "Utrecht–Gouda." This is a heavily utilized corridor with frequent train conflicts. Presently these conflicts are resolved by ProRail according to so-called TAD rules. (TAD is the Dutch acronym

Table 14 Results by simulation and the SMDP-simulation rule for the FCFS, IC-FR (priority to passenger trains), and FR-IC (priority to freight trains)

12 trains per hour				
Discipline	Delay IC (s)	Delay FR (s)	Avr delay (s)	Number conflicts (per hour)
FCFS	182	86	134	2.6
IC-FR	164	109	137	2.6
FR-IC	175	48	111	2.3
SMDP	162	51	106	2.2

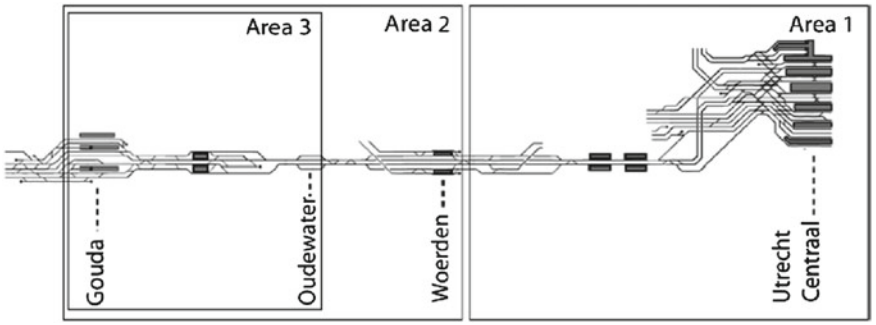


Fig. 11 Corridor Utrecht–Gouda (We here confine ourselves to the trains running from Utrecht to Gouda and do not consider the opposite direction)

for train order document.) These rules are computed offline and prescribe the train order in case a conflict arises. Table 15 shows an example of such a rule.

It is stated that train service 4000 is scheduled to be the first one to run toward the Gouda station (Gd) followed by train services 2000, 2800, and a freight train, if its delay is less than 6 min. If the train service 4000 has a delay between 6 and 10 min, the TAD rule prescribes that the train should let train services 2000, 2800 and the freight train go first.

For the corridor Utrecht–Gouda the TAD rules give unsatisfactory results. ProRail, therefore, was searching for alternative rules that improve the train punctuality for this corridor. Figure 11 shows the corridor in more detail and indicates the three areas, which in our approach will be considered separately.

Table 15 Example of a TAD rule

Train	To	Arrival time	Minimum delay	Maximum delay	Train order
4000	Gd	−0.03/−0.33	0	6	4000 − 2000 − 2800 − FR
4000	Gd	−0.03/−0.33	6	10	2000 − 2800 − FR − 4000

After inspecting the corridor and the specific elements that play a role at each conflict location, we concluded that there are three different areas where conflicts occur and where a resolution rule is needed. In Area 1, the trains leave the Utrecht station toward Gouda. When a conflict arises, one needs to establish an optimal departure order based on some optimization criterion. In Area 2, at Woerden station, there is a double track over a distance of 8 km which makes it possible for fast trains to overtake slower trains without delaying them much. Here, one needs to know if, and when, a fast train may overtake a slower one. Finally in Area 3, at the place called Oudewater, it is possible to stop a freight train so that a passenger train can overtake it. The rule here should prescribe when it is optimal to stop a freight train in favor of a passenger train. Solving the SMD model, described in the previous section, yields the so-called SMD strategy which decides about the order of the trains for each area separately. Just like the TAD strategy the SMD strategy is local and computed offline. However, while the TAD strategy assumes that only one train is delayed at a time, the SMD strategy prescribes conflict resolution rules for all possible situations. In its computation, it does not only consider trains in the direct proximity of a conflict area but it also includes information about (random) future arrivals.

The performance of the SMD strategy is compared to the performance of the TAD rules and some other simple heuristics. For this comparison simulation is necessarily required.

Within the simulation, we have applied the timetable of the year 2007 and used disturbances which are comparable to the ones recorded in 2007. By means of the "common random number" technique the different strategies are confronted one by one to the same set of events so that the differences in performance are solely related to the strategies themselves and not to the random nature of the simulation process.

Table 16 shows the punctuality in percentages at the three stations Utrecht (Ut), Woerden (Wd) and Gouda (Gd) for the different strategies. Here, a train is called "punctual" if the delay is less than 3 min. Each value represents the punctuality averaged over all trains and different train services that cross that station. Upon departure still 92 % of the trains is "punctual". As one sees, due to conflicts within the corridor,

Table 16 Punctuality (in %) of Utrecht–Gouda trajectory

Discipline	Ut	Wd	Gd
TAD	92	86	72
SMD	92	88	82
FCFS	92	85	70
IC-IR-RE-FR	92	89	83
IC-FR-IR-RE	92	89	82
FR-IC-IR-RE	92	89	80
FR-RE-IR-IC	92	86	74
RE-IR-IC-FR	92	87	78
LeastDelayedFirst	92	86	81
MostDelayedFirst	92	87	73

the punctuality decreases toward the end of the corridor. Some strategies resolve the conflicts in a more beneficial way, which translates into a higher punctuality at the end of the line. The FCFS strategy turns out to be the worst strategy. The TAD strategy improves FCFS strategy, but only a bit. The strategy MostDelayedFirst tries to minimize the delay for the most delayed trains by giving them priority over other trains, which however leads to poor overall results. Giving priority to the least delayed train turns out to be a better solution. From the train type priority rules IC-IR-RE-FR turns out to perform very well. This rule gives Intercity trains (IC) priority over all other train types. The Inter Regional (IR) trains have the second highest priority then come Regional trains (RE) and Freight trains (FR) have no priority at all. The SMD strategy improves the punctuality of the TAD strategy by 10 % points and is among the best performing strategies. When considering different scenarios, changing the percentage of freight trains or the total amount of trains, we found that the performance of the simple priority rules was quite sensitive to the number and the mix of the trains. The strategies that performed well in one case were not performing well at all in other cases. The SMD strategy performed very well in all cases, which encourages us to apply this approach to other corridors.

4.5 Summary of Rail-Track Scheduling

Summarizing the results of this section, first of all we note that there is no other way to evaluate the different train scheduling rules than by simulation. We also note that even for experienced and intelligent train schedulers, it is impossible to generate and compare all strategies. The SMDP-algorithm though, in principle computes presumably optimal decisions. In practice, these decisions can be overruled in the light of other information and expertise of schedulers, which cannot be included in the SDMP-simulation model. Nevertheless, the combined SMDP-simulation approach appeared to provide a valuable tool to support practical train scheduling.

5 Evaluation

OR (Operations Research) is well known for its value of mathematical optimization. Most famous applications considered in OR are the shortest routing problems (in route planning systems), and standard inventory optimization (e.g., by deterministic EOQ formulas). Generally, however, stochastics is involved to model uncertainty. Here the value of OR seems less famous, although simple insights and formulas from queueing theory (Q) and stochastic inventory theory (on replenishment policies) are available, next to techniques for stochastic optimization such as Markov decision theory (MDP).

In contrast, in practice OR techniques are often perceived as being too complex to apply and OR models are too simple by relying on strong underlying assumptions such as exponential distributed process times. Simulation, in particular discrete event simulation, then naturally comes as a manageable and practical tool for evaluation and

search-based optimization with virtually no restriction on either practical complexity or stochastic assumptions. The use of OR results, particularly of stochastic nature, generally seems to be skipped.

This chapter aimed to promote that even in that mathematically unsolvable practice, results from OR could still be most useful in combination with simulation, in either of two ways

(i) To provide insights so as to assist the simulation steps in search for optimization.
(ii) To provide an optimization formulation and a technique for its computational feasibility as well as its evaluation by using simulation.

The three applications discussed in this chapter, are real-life applications but their description is far from complete. These examples simply show that each practical problem description requires a tailor-made solution, for which both an OR formulation and the way it is to be integrated with simulation are to made specific and practical.

Simulation engineers might regard OR as too restricted for real-life scale and complexity. OR practitioners in contrast might regard simulation as insufficiently formally supported and specific. This chapter aimed to illustrate the opposite. That one could well benefit from the other. At practical call center scale by queueing insights and by simulation for practical improvements. For blood banks and railways, a theoretical OR solution technique for solvable systems used by simulation for expansion to simple practical rules. Accordingly, a combination appears to be highly mutually beneficial. Beyond these and other applications by themselves, this combination also seems of future research interest such as to integrate simulation for the nearly open problem of transient queueing applications on the one hand and to support simulation search approaches by OR on the other.

References

1. B. Adenso-Dıaz, M. Oliva González, and P. González-Torre. On-line timetable re-scheduling in regional train services. *Transportation Research Part B: Methodological*, 33(6):387–398, 1999.
2. A. Al-Ibrahim. *Dynamic Traffic Management for Railway Networks: A Semi Markovian Decision approach for train conflict resolution*. PhD thesis, University of Amsterdam, the Netherlands (Supervisor J.van der Wal), 2010.
3. S. Araya, K. Abe, and K. Fukumori. An optimal rescheduling for online train traffic control in disturbed situations. In *Decision and Control*, 1983. *The 22nd IEEE Conference on*, volume 22, pages 489–494. IEEE, 1983.
4. A.A. Assad. Models for rail transportation. *Transportation Research Part A: General*, 14(3):205–220, 1980.
5. S.L. Bell and R.J. Williams. Dynamic scheduling of a system with two parallel servers in heavy traffic with resource pooling: asymptotic optimality of a threshold policy. *The Annals of Applied Probability*, 11(3):608–649, 2001.
6. J.T. Blake, S. Thompson, S. Smith, D. Anderson, R. Arellana, and D. Bernard. Optimizing the platelet supply chain in nova scotia. In *Proceedings of the 29th meeting of the European Working*

Group on Operational Research Applied to Health Services (ORAHS). Prague: European Working Group on Operational Research Applied to Health Services, pages 47–66, 2003.
7. S. Borst, A. Mandelbaum, and M.I. Reiman. Dimensioning large call centers. *Operations research*, 52(1):17–34, 2004.
8. K. Cattani and G.M. Schmidt. The pooling principle. *INFORMS Transactions on Education*, 5(2):17–24, 2005.
9. J.-F. Cordeau, P. Toth, and D. Vigo. A survey of optimization models for train routing and scheduling. *Transportation science*, 32(4):380–404, 1998.
10. A. D'Ariano. *Improving real-time train dispatching: models, algorithms and applications*. Number T2008/6. Netherlands TRAIL Research School, 2008.
11. W. de Kort, M. Janssen, N. Kortbeek, N. Jansen, J. van der Wal, and N. van Dijk. Platelet pool inventory management: theory meets practice. *Transfusion*, 51(11):2295–2303, 2011.
12. Q. Duan and T.W. Liao. Optimization of blood supply chain with shortened shelf lives and abo compatibility. *International Journal of Production Economics*, 153:113–129, 2014.
13. B.E. Fries. Optimal ordering policy for a perishable commodity with fixed lifetime. *Operations Research*, 23(1):46–61, 1975.
14. N. Gans and Y.-P. Zhou. Call-routing schemes for call-center outsourcing. *Manufacturing & Service Operations Management*, 9(1):33–50, 2007.
15. M.R. Garey and D.S. Johnson. Computers and intractability: a guide to the theory of np-hardness, 1979.
16. R. Haijema, J. van der Wal, and N.M. van Dijk. Blood platelet production: Optimization by dynamic programming and simulation. *Computers & Operations Research*, 34(3):760–779, 2007.
17. R. Haijema, N.M. van Dijk, J. van der Wal, and C.Th. Smit Sibinga. Blood platelet production with breaks: optimization by sdp and simulation. *International Journal of Production Economics*, 121(2):464–473, 2009.
18. K. Katsaliaki and S.C. Brailsford. Using simulation to improve the blood supply chain. *Journal of the Operational Research Society*, 58(2):219–227, 2007.
19. N. Kortbeek, J. van der Wal, N.M. van Dijk, R. Haijema, and W. de Kort. Blood production and issuing optimization: Strategies for younger platelets. *UvA - research report*, 2008.
20. W. Krug. *Modelling, Simulation and Optimisation: For Manufacturing, Organisational and Logistical Processes*. SCS-European Publishing House, 2002.
21. A. Mascis and D. Pacciarelli. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, 143(3):498–517, 2002.
22. S. Nahmias. Perishable inventory theory: A review. *Operations Research*, 30(4):680–708, 1982.
23. T. Osogami, M. Harchol-Balter, A. Scheller-Wolf, and L. Zhang. Exploring threshold-based policies for load sharing. 2004.
24. G.P. Prastacos. Blood inventory management: an overview of theory and practice. *Management Science*, 30(7):777–800, 1984.
25. V. Sirelson and E. Brodheim. A computer planning model for blood platelet production and distribution. *Computer methods and programs in biomedicine*, 35(4):279–291, 1991.
26. D.R. Smith and W. Whitt. Resource sharing for efficiency in traffic systems. *Bell System Technical Journal*, 60(1):39–55, 1981.
27. M.S. Squillante, C.H. Xia, D.D. Yao, and L. Zhang. Threshold-based priority policies for parallel-server systems with affinity scheduling. In *American Control Conference, 2001. Proceedings of the 2001*, volume 4, pages 2992–2999. IEEE, 2001.
28. J. Törnquist. Computer-based decision support for railway traffic scheduling and dispatching: A review of models and algorithms. In *OASIcs-OpenAccess Series in Informatics*, volume 2. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2006.
29. J. Tornquist and J.A. Persson. Train traffic deviation handling using tabu search and simulated annealing. In *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*, pages 73a–73a. IEEE, 2005.
30. N.M. van Dijk and E. van der Sluis. To pool or not to pool in call centers. *Production and Operations Management*, 17(3):296–305, 2008a.

31. N.M. van Dijk and E. van der Sluis. Practical optimization by OR and simulation. *Simulation Modelling Practice and Theory*, 16(8):1113–1122, 2008b.
32. N.M. Van Dijk, R. Haijema, J. Van Der Wal, and C.Th. Smit-Sibinga. Blood platelet production: a novel approach for practical optimization. *Transfusion*, 49(3):411–420, 2009.
33. R.B. Wallace and W. Whitt. A staffing algorithm for call centers with skill-based routing. *Manufacturing & Service Operations Management*, 7(4):276–294, 2005.
34. R.W. Wolff. Stochastic modelling and the theory of queues. *Englewood Cliffs, NJ*, 1989.

Tree Search and Simulation

João Pedro Pedroso and Rui Rei

Abstract This chapter presents a general methodology for embodying simulation as part of a tree search procedure, as a technique for solving practical problems in combinatorial optimization. Target problems are either difficult to express as mixed integer optimization models, or have models which provide rather loose bounds; in both cases, traditional, exact methods typically fail. The idea then is to have tree search instantiating part of the variables in a systematic way, and for each particular instantiation—i.e., a node in the search tree—resort to a simulation for assigning values to the remaining variables; then, use the outcome of the simulation for evaluating that node in the tree. This method has been used with considerable success in gameplaying, but has received very limited attention as a tool for optimization. Nevertheless, it has great potential, either as a way for improving known heuristics or as an alternative to metaheuristics. We depart from repeated, randomized simulation based on problem-specific heuristics for applications in scheduling, logistics, and packing, and show how the systematic search in a tree improves the results that can be obtained.

1 Introduction

Tree search and branch-and-bound variants are among the most powerful search methods in combinatorial optimization. How to direct the search through the tree, in terms of the selection of a node, the selection of a variable within a node, and the selection of a value to assign to that variable, are key factors for performance. Whereas in some applications these choices are relatively straightforward, in other cases it is very difficult, mostly due to the absence of good bounds. In this work we will focus on applications for which the outcome of a decision is very difficult to

J.P. Pedroso · R. Rei (✉)
INESC TEC and DCC-FCUP, Porto, Portugal
e-mail: rui.rei@dcc.fc.up.pt

J.P. Pedroso
e-mail: jpp@fc.up.pt

M. Mujica Mota et al. (eds.), *Applied Simulation and Optimization*,
DOI 10.1007/978-3-319-15033-8_4

assess before having a complete solution, therefore requiring the simulation of the whole construction process to probe into the quality of the decision.

Designing effective methods under these circumstances involves decisions that overcome the main weaknesses of tree search: not reaching a leaf node in a limited amount of time; unbounded growth of the queue of unexplored nodes; and most importantly, getting trapped in a particular, limited part of the tree. We will develop on methods for this, based on good heuristics for constructing a solution; tree search may be seen as an enhancement of these heuristics which, in the limit case, may completely explore the search space.

The applications covered in this work are the following:

1. Number partitioning—an easy problem to formulate, which will allow us to clearly state the important aspects of tree search and simulation in this context (Sect. 3).
2. Stacking—a hard problem involving the choice of a stack to place an item in such a way that the number of item relocations is minimal (Sect. 4).
3. Recursive circle packing—a variant of circle packing where annular items can either be placed inside a rectangular container or inside other items (Sect. 5).

2 Background

2.1 Tree Search

Tree search is a method for systematically exploring the search space, with the aim of finding the best solution appearing therein. In its simplest form, all solutions are enumerated and verified, hence taking exponential time with respect to the number of variables. In optimization problems for which there is an appropriate mathematical programming formulation, it is usually easy to find bounds on the solutions that can be obtained from a node, which allows the search tree to be pruned without loss of optimality. For a minimization problem, if the lower bound in a given node is greater than the value of a known solution (i.e., greater than an upper bound on the optimum), then the tree can be safely pruned at this node. This is the standard procedure in branch-and-bound (BB) [1, 2]. Literature on BB is ample, as the method is fully general and can be applied in widely diverse areas. For a sample of recent applications of BB methods, see, e.g., [3–6].

For the problems we are dealing with here, formulations are very loose, rendering the provided bounds very ineffective and leading to very little or no pruning at all. This implies that, except for toy instances, exploration of the whole tree is unreasonable, either due to time limitations or because the size of the tree would grow unacceptably large. Consequently, the best solution found may not be optimal because the search space has not been fully explored. In this context, tree search may be used as an *approximative* algorithm.

There are several ways for exploring the nodes of a search tree. In *uninformed search* there is no use of information concerning the value of a node during tree exploration. For example, in *breadth-first search* all nodes in a level of the tree (i.e., nodes that are equally distant from the root) are explored before proceeding to the next level. In *depth-first search* (DFS) each node is expanded down to the deepest level of the tree, where a node with no expansion—i.e., a *leaf*—is found; then search backtracks until a node with unexplored children is found, which is again expanded until reaching a leaf, and so on, until the whole tree is explored. As only one path from the root to a leaf has to be stored at any given time, DFS has modest memory requirements. Problem-specific heuristics may be used in conjunction with DFS for deciding the order of exploration of each node's children; as this information is only considered locally at each node, this is usually called *partially informed search*. On *informed search*, a set of open nodes is; the most common variant is *best-first search*, which selects the next node to expand based on problem-specific knowledge. To this end, an *evaluation function* is used which conveys information about the worth of each open node, and the one with the highest rating from all open nodes is selected at each iteration.

When good guiding heuristics exist, DFS is usually very effective. When compared to greedy construction based on such heuristics, DFS allows for substantial improvements; furthermore, these improvements can be obtained very quickly due to the simplicity of DFS, which imposes an almost nonexistent overhead on the greedy construction algorithm. For situations where the exploration of the full tree is expected to be possible in a reasonable time, DFS is usually an appropriate choice. However, this is not the case for most practical problems. For sufficiently large trees, DFS suffers the problem of being unable to recover from poor decisions taken at the beginning of the search.

For trees with high branching factor, *iterative broadening*, proposed in [7], attempts to overcome the deficiencies of DFS by running a sequence of restricted depth-first searches. Each restricted DFS considers a larger subset of the tree than the previous: the first iteration examines the heuristically preferred node, the second iteration examines the two top-ranked children of each node, and so on; in a tree of depth d, at iteration k, iterative broadening visits k^{d-1} leaves.

Best-first search tries to overcome the deficiencies of DFS by considering, at each iteration, nodes from different levels in the tree. A seminal example is the A^* algorithm for the shortest path problem [8], where heuristic information that never overestimates the cost of the best solution reachable from a node is used to evaluate it. Best-first search suffers the problem of requiring exponential space, thus becoming impractical in many situations.

A related algorithm taking only linear space is *iterative deepening A** (IDA*): each iteration is a DFS modified to use heuristic evaluation as in A*, and a limit on the heuristic value to interrupt the search; the interruption threshold is increased at each iteration [9]. In practice, for many combinatorial optimization problems IDA* visits too many internal nodes before reaching its first leaf; this is mainly due to the underestimation provided by the heuristics being substantially different from the best value that can be reached from each node. Besides, IDA* does not cope well with the

fact that in many optimization problems every node has a different heuristic value, only the best of which is expanded at each iteration. In combinatorial optimization problems the depth of the search tree is bounded, and reaching a leaf is usually inexpensive; this is not exploited in IDA*.

The most widely used tools for solving combinatorial problems that can be formulated as mathematical programming models are mixed integer linear programming (MIP or MILP) solvers. These usually incorporate state-of-the-art tools in terms of pruning the search tree and adding cuts to the model, in a black-box solver targeted at obtaining the best performance in the widest range of problems possible. Still, deficiencies similar to those of DFS are often observed in practice, as the solvers may be stuck in the search for very long periods. Methods for overcoming this are a current trend, following the observation of heavy-tailed phenomena in satisfiability and constraint satisfaction [10]. The basic idea is to execute the algorithm for a short time limit, possibly restarting it from with some changes, usually with an increased time limit until a feasible solution is found. This has been recently addressed in [11], where variability of the solutions on which the MIP solver is trapped, for different initial random start conditions, is exploited. The proposed method consists in making a number of short runs with randomized initial conditions, bet on the most promising run, and bring it to completion, in an approach named *bet-and-run*. Diversified runs are produced in several fronts: exchanging rows and columns in the input instance, perturbing parameters of the MIP solver, and changing coefficients in the objective function. The choice of the candidate to bring to completion is based on 11 criteria, the most important of which are the number of open nodes, the lower bound improvement with respect to the root node, and number of integer-infeasible variables among all open nodes. Results are reported for a set of benchmark instances, showing that bet-and-run can lead to significant speedups.

We propose to exploit variability in the search in a rather different manner, by making a *dive* from each open node until reaching a leaf, and using its outcome in the evaluation of the dive's starting node. Dive results are also backpropagated up to the root node. Each of these dives corresponds to a descent in the tree, possibly in a randomized way, which in the context of Monte Carlo Tree Search—the subject of the next section—is called a *simulation*. The two terms are used interchangeably in the remainder of this chapter. In the problems dealt with in this work, simulations always lead to feasible solutions; hence, our method provides both an *anytime algorithm* and, for small instances, complete search.

2.2 Monte Carlo Tree Search: State of the Art

Monte Carlo Tree Search (MCTS) is a method for exploring the search tree and exploiting its most promising regions. Although the idea of combining Monte Carlo evaluation with tree search had been studied before (see, e.g., [12, 13]), it was not until recently—with the appearance of MCTS—that it received greater scientific attention. Coulom [14] proposed the initial version of MCTS and applied it with considerable

success to the game of Go (9×9 board). Gameplaying is still the area where the algorithm and its many variants are most commonly used [15]. In this context, MCTS has the aim of finding the most favorable decision at each step, by taking random samples from the decision space and valuating nodes of the tree according to the results of those samples. MCTS has had a particularly strong impact on games, where its application has led to the best gameplaying software, most notably for the game of Go [16, 17]; but it has also been applied on artificial intelligence approaches for other domains that can be represented as trees of sequential decisions and for planning problems (see [18] for a comprehensive survey).

There are, however, very few publications on combinatorial optimization; some results have been provided for general mixed integer optimization [19], but to the best of our knowledge, there are no reports of its application for solving specific optimization problems. This possibility will be illustrated, with a detailed implementation and results, in the following sections. In the remainder of this section we describe the basic algorithm that will be used as the foundation for the three applications presented.

MCTS is an iterative procedure in which a search tree is asymmetrically constructed, attempting to expand the tree toward its most promising parts while balancing exploitation of known good branches with exploration of seemingly unrewarding branches. The algorithm is based on the idea of Monte Carlo evaluation, i.e., the reward associated with a particular node can be estimated from the results of random simulations started from that node. Each node keeps track of the number of simulations started from its state as well as their outcomes, and these data are used to produce an estimate value for the node when deciding how to expand the tree. Each iteration of MCTS can be divided into the following four steps:

1. *Selection*: starting from the root node, select the child node which currently looks more "promising". This is done recursively until a node n which has not yet been fully expanded (i.e., some of its children have not yet been generated) is reached. The definition of promising is one of the key aspects determining the performance of MCTS. In plain MCTS, average win rate is used for node selection. The UCT algorithm [20] provides an enhancement to this simple rule, by posing the selection problem at each node as a multiarmed bandit problem, and then using the UCB1 policy [21] to achieve an optimal bound on regret. In UCT, the score or utility $U(n)$ of node n is defined as

$$U(n) = X(n) + E(n), \tag{1}$$

   where $X(n)$ is the exploitation utility associated with n, and $E(n)$ is the exploration utility of n. At each node, the child with maximum $U(n)$ is selected, until an unexpanded node is reached.
   In gameplaying, $X(n)$ is typically taken to be the average reward of simulations run from n. Later in this section, we propose an alternative expression for $X(n)$ that is more suitable for optimization.

The exploration component is normally defined as

$$E(n) = c\sqrt{\frac{\ln s_{p(n)}}{s_n}}, \tag{2}$$

where c is an exploration parameter (theoretically equal to $\sqrt{2}$), $s_{p(n)}$ is the number of simulations done under the parent node $p(n)$, and s_n is the number of simulations done under the child node n (i.e., simulations started from n or any node in the subtree under n). This expression is designed such that exploration progressively gives way to exploitation, although all siblings are eventually selected if enough iterations are allowed. This means that the search cannot become permanently trapped in any region of the search space.

To summarize, selection starts at the root node and proceeds down the tree, selecting at each node the child with highest utility. Upon reaching an unexpanded node, selection stops and the current node is chosen for expansion.

2. *Expansion*: one or more children of the selected node n are created by applying possible decisions in n to copies of n. We present two strategies for node expansion:

   Single expansion: a single child node is created using a randomly chosen unexplored decision in n. Other unexplored decisions are kept for a later time when node n is again selected for expansion.

   Full expansion: all children of n are immediately created by generating all possible decisions in the node.

3. *Simulation*: from each node created in the expansion step, perform a simulation until a terminal state (i.e., a solution) is reached, and record the value obtained. Various approaches can be taken in simulation, ranging from uniform random decisions—requiring nothing more than a generative model of the problem—to heuristic construction algorithms incorporating domain-specific knowledge. The latter approach typically allows for faster convergence at the expense of simplicity and generality. In the applications presented in this work, we will use problem-specific construction heuristics for simulation; for each problem, the heuristic used is detailed in the corresponding section.
4. *Backpropagation*: propagate the outcome of each simulation up the tree until the root node is reached; this updates statistics (simulation scores and counts) on all nodes between the selected node and the root.

Because one simulation is done per new child of the selected node n, the two expansion strategies described above will exhibit different behavior with respect to the variation of $U(n)$; namely, the number of children generated is proportional to the decrease in the exploration term $E(n)$ for the parent node n. This difference ultimately leads to different search paths in the tree, which may in turn have an impact on the performance of the algorithm. In trees with high branching factors, if

single expansion is used, a node may be confirmed (to a certain degree) as a poor choice before generating all its children, thereby saving both time and space which can be used to explore other areas of the tree. This effect is dampened in trees with lower branching factors.

Applying Monte Carlo tree search to solve optimization problems has many similarities, but also significant differences to its application in gameplaying. First, the size of the search trees in both domains is commonly large enough to prevent complete search within reasonable computational time. Hence, MCTS should direct the search, leveraging all the information gathered up to the moment in the most suitable way.

A significant difference concerns the evaluation of nodes and their associated statistics. Whereas in gameplaying a branch with a high average win rate is suggestive of a strong line of play, in optimization—since we are interested in finding extrema—the average solution under a node is not a good estimator of the optimal solution to the node's underlying subproblem. Additionally, rewards in gameplaying often take 0–1 values for loss and win, respectively; objective functions, on the other hand, may take arbitrary values. Since UCB1/UCT were designed with rewards in the [0, 1] interval in mind, we must map objective function values into that interval, in order to maintain the proper balance between the two components of (1). To address these two issues, we propose the following expression for $X(n)$:

$$X(n) = \frac{e^a - 1}{e - 1}, \text{ with } a = \frac{\hat{w}^* - \hat{z}_n^*}{\hat{w}^* - \hat{z}^*}, \tag{3}$$

where $\hat{z}^*$ and $\hat{w}^*$ are, respectively, the best and the worst simulation results found so far in the whole tree, and $\hat{z}_n^*$ is the best simulation result under node n. A plot of the proposed reward function can be seen in Fig. 1.

Although our main guiding criterion should be based on the best simulation outcome $\hat{z}_n^*$, the average outcome of simulations under a node—denoted as $\overline{z}_n$—may

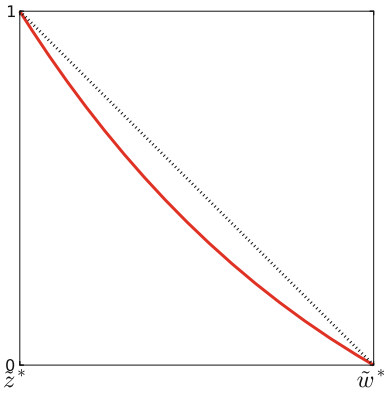


Fig. 1 The proposed reward function $X(n)$ (*solid line*), in terms of the best simulation outcome under node n, $\hat{z}_n^*$. The linear reward function (*dotted line*) is shown for reference

still provide a useful hint on the interest in exploring the node. We propose to use a function similar to $X(n)$ representing the average reward under node n, and define it as

$$\overline{X}(n) = \frac{e^b - 1}{e - 1}, \text{ with } b = \frac{\hat{w}^* - \overline{z}_n}{\hat{w}^* - \hat{z}^*}. \tag{4}$$

Instead of incorporating this information directly into the exploitation term, we integrate $\overline{X}(n)$ as a factor in the exploration term; we call this *average-weighted exploration*, and define it as

$$E'(n) = \overline{X}(n)E(n). \tag{5}$$

The above ideas, applied to the problem of combinatorial optimization, are summarized in the pseudocode in Algorithm 1.

Algorithm 1: MCTS for (minimization) combinatorial optimization.

```
Data: problem instance I
Result: upper bound on optimal value (minimization problems)
r ← create root node with initial state from I
z* ← ∞
repeat
    n ← starting from r, recursively select child node with maximum U(n)
    C ← set of child nodes obtained from expanding n
    foreach c ∈ C do
        run a simulation from c
        z ← result of the simulation
        propagate z up until reaching r
        if z < z* then
            z* ← z
until computational budget is depleted
return z*
```

3 Number Partitioning

The number partitioning problem (NPP) is a classical combinatorial optimization problem, with applications in public key cryptography and task scheduling. Given a set (or possibly a multiset) of N positive integers $\mathscr{A} = \{a_1, a_2, \ldots, a_N\}$, find a partition $\mathscr{P} \subseteq \{1, \ldots, N\}$ that minimizes the discrepancy

$$E(\mathscr{P}) = \left| \sum_{i \in \mathscr{P}} a_i - \sum_{i \notin \mathscr{P}} a_i \right|.$$

Partitions such that $E = 0$ or $E = 1$ are called perfect partitions.

A pseudo-polynomial time algorithm for the NPP is presented in [22] for the case where all a_j are positive integers bounded by a constant A; for the general case of exponentially large input numbers (or exponentially high precision, if the a_j's are real numbers) the problem is NP-hard. If the numbers a_j are independently and identically distributed random numbers bounded by $A = 2^{\kappa N}$, the solution time abruptly raises at a particular value $\kappa = \kappa_c$; this is due to the high probability of having perfect partitions for $\kappa < \kappa_c$, and this probability is close to 0 for $\kappa > \kappa_c$ (see [23, 24] for more details).

A direct application of the NPP occurs in load balancing of two identical machines. The common two-way NPP, as well as a generalization to an arbitrary number of subsets (equivalently, machines) are tackled in [25] by recasting the problem as an unconstrained quadratic binary program (UQP); the UQP is then solved using a tabu search algorithm. Another application arises in high-performance multidisk database systems. To promote parallelization of I/O and minimize query response times in such systems, data that are likely to be accessed by same queries are distributed across K disks—a process called *declustering*. This problem, equivalent to a multiway NPP, is tackled in [26] using a two-phase approach: the first phase consists in recursive bipartitioning to obtain an initial K-way partition; in the second phase, the initial partition is improved through a refinement heuristic.

The best polynomial time heuristic known for the NPP is the differencing method of Karmarkar and Karp [27] (the KK heuristic). It consists of successively replacing the two largest numbers by the absolute value of their difference and placing those items in separate subsets, but without actually fixing the subset into which either number will go (see Algorithm 2).

Algorithm 2: The Karmarkar–Karp heuristic.

Data: ordered set of positive integers $\mathscr{A}$
Result: discrepancy obtained
$\forall a_i \in \mathscr{A}$, create a vertex i with label $l_i \leftarrow a_i$
$\mathscr{E} \leftarrow \{\}$
while there is more than one labeled vertex **do**
  $u, v \leftarrow$ vertices with the two largest labels
  $\mathscr{E} \leftarrow \mathscr{E} \cup \{\{u, v\}\}$
  set label $l_u \leftarrow l_u - l_v$
  remove label l_v from vertex v
return discrepancy (i.e., the last label)

Extensions of the KK heuristic for a complete search have been proposed in [28, 29]. In each step of the KK heuristic the two largest numbers are replaced by their difference; for a complete search, the alternative of replacing them by their sum must also be considered. For the previous example, the complete search tree is represented in Fig. 2.

When applied to the set $\mathscr{A} = \{8, 7, 6, 5, 4\}$, the KK heuristic leads to the partitions $\{8, 6\}$ and $\{7, 5, 4\}$ with discrepancy 2 (Fig. 3). Figure 4 illustrates the graph corresponding to the optimal solution, as obtained by complete search. Straight edges connect differencing vertices, that will be in different partitions; curly edges connect

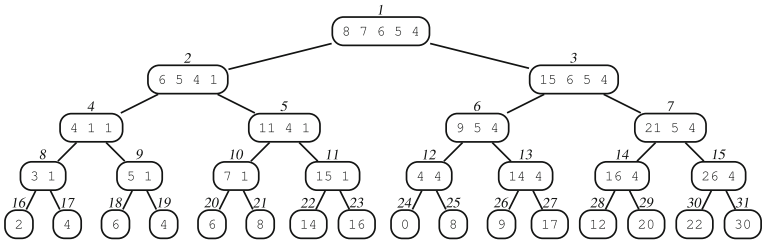


Fig. 2 Search tree for the complete differencing method with the set $\mathscr{A} = \{8, 7, 6, 5, 4\}$

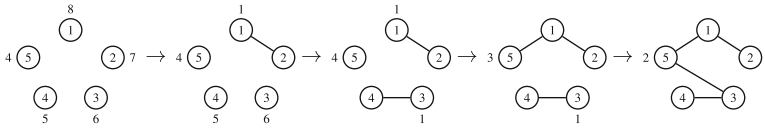


Fig. 3 Graph created with the KK heuristics, corresponding to the path $1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \rightarrow 16$ in Fig. 2

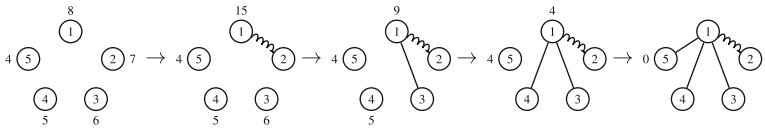


Fig. 4 Graph created while applying complete search: steps followed for creating the optimal partition, i.e., the path $1 \rightarrow 3 \rightarrow 6 \rightarrow 12 \rightarrow 24$ in Fig. 2

addition vertices, that will be in the same partition. The optimal solution is the partition {8, 7} and {6, 5, 4}, which is a perfect partition.

In the worst case, the complete differencing method has exponential complexity. Parts of the search tree may be pruned by observing that:

- the KK heuristic is exact for partitioning 4 or less numbers;
- the algorithm can be stopped when a perfect partition is found;
- when the difference between the largest remaining number and the sum of other remaining numbers is >1, the best possible solution is to place the largest number in one set and all the other numbers in the other set.

Algorithm 3 introduces depth-first search for the complete KK.

We propose the application of MCTS to this problem, using the KK heuristic as the construction method in the simulation step. Since this heuristic is deterministic, running it from the differencing child (which coincides with the heuristic's choice) would lead to the exact same solution as that obtained for the parent node; thus, we can safely reuse the parent's solution, and therefore only one new construction is required for the two children of each node. Another advantage of using this heuristic is that

Algorithm 3: dfs($\mathscr{A}$)—depth-first search.

Data: ordered set of positive integers $\mathscr{A}$
Result: optimum discrepancy
if $|\mathscr{A}| \leq 4$ **then return** *KK* ($\mathscr{A}$)
$u, v \leftarrow$ largest and second-largest values in $\mathscr{A}$
$d \leftarrow u - \sum_{a \in \mathscr{A} \setminus \{u\}} a$
if $|d| \leq 1$ **then return** $|d|$
if $d > 0$ **then return** d
$\mathscr{A} \leftarrow \mathscr{A} \setminus \{u, v\}$
$\ell \leftarrow$`dfs` $(\mathscr{A} \cup \{u - v\})$
if $\ell \leq 1$ **then return** ℓ
$r \leftarrow$`dfs` $(\mathscr{A} \cup \{u + v\})$
return $\min(\ell, r)$

repeated solutions are avoided altogether; such would not be the case with random or semi-greedy simulations. Finally, the KK heuristic allows the above pruning rules to be used, reducing the size of the search space without sacrificing completeness.

Due to the very low branching factor, and from empirical observation, we use a full expansion strategy in the expansion step of MCTS, meaning that both children of each node are immediately generated once the node is selected. Furthermore, for this problem, we choose to use the classical exploration term $E(n)$ (see (2)) in the evaluation of nodes, instead of the average-weighted exploration term $E'(n)$ (see (5)). This is motivated by the fact that average node performance can be deceptive in the NPP; varying just one or two decisions during construction has very deep and unpredictable consequences on the quality of the final solutions; this lack of correlation between particular decisions and solution quality makes average performance a poor guiding principle in this problem.

Table 1 presents results obtained with the KK heuristic and with time-limited DFS and MCTS, run on 10 hard instances from [29].

As expected, both tree search variants provide very considerable improvements over KK. It is well known that for difficult instances the optimum for NPP is very hard to find; DFS, being able to explore a much larger portion of the tree—hundreds to thousands of millions of nodes per hour, for these instances—is very difficult to beat. Also, as previously mentioned, the performance of DFS is highly dependent on the construction heuristic used, which in this case provides solutions of very good quality to begin with.

Although there is no clear indication of a winner from the results obtained, the observation of mixed results between DFS and MCTS is nonetheless impressive: MCTS was highly competitive despite exploring less than a thousandth of the nodes explored by DFS. This suggests that MCTS can effectively guide the search toward promising regions. In our view, these are very encouraging results.

We conclude with the hypothesis that MCTS's rate of convergence should improve over time, not only for this problem but also in general. While DFS gains practically nothing as the search progresses, MCTS constantly accumulates knowledge of the

Table 1 Results obtained for number partitioning with the KK heuristic, DFS, and MCTS on hard, large instances from [29]

Instance	KK	DFS (60 s)	MCTS (60 s)	DFS (600 s)	MCTS (600 s)
Hard0100	73.37	56.12	54.56	53.36	51.20
Hard0200	171.81	151.91	150.88	151.35	149.79
Hard0300	265.77	247.07	248.82	247.07	247.18
Hard0400	366.18	343.36	347.55	339.07	344.84
Hard0500	461.60	443.54	441.42	438.79	437.22
Hard0600	557.09	540.22	542.59	539.13	540.66
Hard0700	659.50	640.48	641.78	637.70	633.84
Hard0800	751.27	737.65	738.49	731.98	735.67
Hard0900	853.36	834.95	837.93	834.95	833.87
Hard1000	952.47	932.55	938.05	932.55	930.23

DFS and MCTS results are reported at 60 s (center columns) and 600 s (right-hand side columns). For MCTS, the average best solution of 10 independent runs is shown. Values reported are $\log_2(n+1)$, where n is the discrepancy obtained (which for these instances is a very large integer)

search landscape, which should theoretically help convergence toward the optimum. Having admittedly few results to support this hypothesis, we leave its confirmation as future work.

4 Stacking

The stacking problem (SP) consists of a series of placement decisions for a set of items with known dates for entrance and exit from a warehouse (see Fig. 5), denoted by *release* and *due* dates, respectively. Items are placed in vacant positions, or on top of other items forming stacks, i.e., last-in first-out queues. At any given time, only the top item of each stack can be taken, so in order to take an item that is not at the top of a stack, it is necessary to first relocate all items above it to other stacks. The objective is to store and then deliver all items, while respecting their release and due

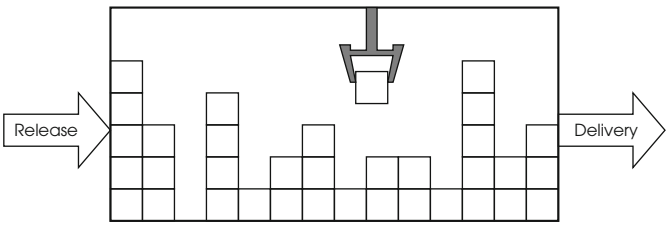


Fig. 5 A warehouse where items are stored in stacks, using a stacking crane that can only handle one item at a time

dates, with a minimum number of relocations. In the variant tackled in this work, we assume that there is no height limit on stacks (uncapacitated SP), movements occur instantaneously, and releases or deliveries cannot be anticipated or delayed.

Stacking problems have evident practical importance in container port operations [30, 31] and ship stowage planning [32, 33]. Stacking is also important in the steel industry as a means of storage for finished products [34].

As demonstrated in [34], the zero-relocation SP (decision problem) is NP-complete for any fixed number of stacks $W \geq 4$, by a polynomial reduction to the problem of coloring circle graphs. From this it follows that the general r-relocation SP is also NP-complete, and the optimization version of the SP is NP-hard.

Although the SP is first described in [34], the paper mentions the existence of similar problems and presents an overview of the literature. One such problem is the Block Relocation Problem (BRP), which is actually a subset of the SP. In the BRP, the initial state of the stacks is given as input data along with a (partial) order of retrieval of the items. The objective is to find the shortest sequence of movements such that items are retrieved in the given order. In [35], a branch-and-bound algorithm for the BRP, as well as a simple heuristic for real-time applications are proposed. The heuristic rule is based on an estimate of the number of additional relocations for each stack. In [36], an algorithm for the BRP based on the corridor method is presented. The corridor method combines mathematical programming techniques with heuristics. The main idea is to exactly solve subproblems where some variables are fixed, creating a "corridor-like" region where an item is allowed to go.

For tackling the SP itself, in [34], a discrete-event simulation model of the warehouse is used, and construction of solutions is based on a semi-greedy heuristic. The heuristic is invoked when deciding the placement of an item during a release or *reshuffle*. Reshuffling is defined as the relocation of items that is necessary to reach an item lower in the stack. For simplicity, the voluntary relocation of items in-between releases or deliveries—called *remarshalling*—is not considered during a simulation. Note, however, that this may potentially leave the optimal solution(s) out of the search space, therefore losing the guarantee of optimality even for a complete search.

A simulation consists in traversing a schedule of events (i.e., releases and deliveries) in chronological order and processing each event appropriately. When multiple events have the same date, deliveries are processed first, in increasing order of *item depth*, where the depth of an item is defined as the number of items above it in the same stack. Then, any releases are processed in inverse order of due date, that is, items with greater due date are released first. Ties are resolved randomly.

The probabilistic component of the construction heuristic is exploited by repetition of the process using different seeds for the pseudorandom number generator, in a simple method called Multiple Simulation (MS). This simple yet effective idea can also be exploited in Monte Carlo tree search, taking advantage of the tree structure to implicitly force different simulations to be executed.

In order to accelerate MS, a simulation is interrupted as soon as it is known that the number of relocations of the incumbent solution cannot be improved. This is actually a weak form of pruning, as seen in branch-and-bound algorithms. Whenever a better

solution is found, the cutoff value is updated, tightening the upper bound for future simulations. This optimization is not used in MCTS because, even after a simulation is known to be poor, the algorithm can still benefit from knowing how poor the simulation is, and therefore it is run to completion anyway.

We now describe the construction heuristic used in multiple simulation, called Flexibility Optimization (FO). FO will be used in the MS method, as well as in the simulation step of MCTS. For details on the remaining steps of the MCTS algorithm, please refer to Sect. 2.2.

First, we need to define the concept of *stack movement date*: the earliest movement date of stack s is represented as

$$m_s = \min_{i \in I_s} D_i,$$

where I_s represents the set of items currently in stack s, and D_i is the due date of item i. When stack s is empty, then $m_s = \infty$ by definition. Another important concept is that of *due date inversion*: an inversion occurs when an item i is placed (directly of indirectly) above an item j with $D_i > D_j$. In order to deliver j, item i will have to be relocated.

One can view m_s as an indicator of the *flexibility* of stack s for receiving new items without creating inversions. To illustrate this idea, consider an empty stack s, with $m_s = \infty$; this stack is considered as having infinite flexibility, since any item can be added to it without creating a new inversion. On the other hand, if $I_s \neq \{\}$, then m_s is finite and only items with due date up to m_s can be added to s without creating an inversion.

When placing an item, FO will prefer stacks where loss of flexibility is minimized, whenever this is possible without creating new inversions. If a new inversion is unavoidable, the heuristic places the item in the stack with the largest movement date, in order to postpone the forced relocation as much as possible. The heuristic makes use of a function associating to each placement decision $i \rightarrow s$ a score $f(i, s)$; this function embeds the above rules, and is defined as

$$f(i,s) = \begin{cases} \frac{1}{1+D_i-m_s} - 1 & \text{if } D_i > m_s, \\ \frac{1}{1-D_i+m_s} & \text{otherwise.} \end{cases} \tag{6}$$

The graph for this function is shown in Fig. 6. Note that the top branch in (6) represents the creation of a new inversion (since $D_i > m_s$), therefore lower scores ($f(i, s) \in [-1, 0]$) are assigned to it. The bottom branch represents the placement of an item without creating an inversion, being given a higher score than any inversion-inducing decision. Given an item i and a set of possible destination stacks T, the FO heuristic constructs a *restricted candidate list* of stacks $\text{RCL} = \{s \in T : f(i, s) \geq f(i, s'), \forall s' \in T\}$, and randomly selects a stack from the RCL as the destination of item i.

A computational experiment was conducted on the 24 benchmark instances of [34], comparing MCTS with the MS method. We use the generic MCTS

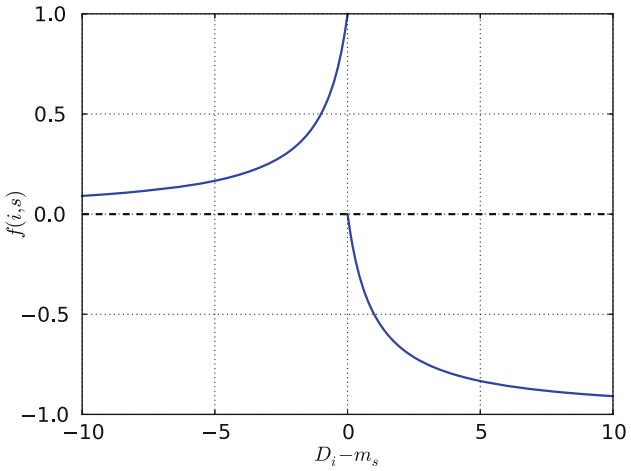


Fig. 6 Score function $f(i, s)$ used by the FO heuristic

algorithm presented in Algorithm 1, running simulations with the FO heuristic in the algorithm's simulation step. In this case, since the heuristic used is nondeterministic, a new simulation must be run for each node created in MCTS. As for the expansion strategy used, in this problem we choose single expansion due to the potentially high branching factor. Additionally, we use the average-weighted exploration term defined in (5).

The two methods were run ten times on each instance, for 600 s, using different seeds for the pseudorandom number generator. Table 2 presents the average number of relocations of the best solution found after 60 s (left-hand side columns), and at the end of the 600 s period (right-hand side columns).

The tree search is naturally expected to perform better, but the reduced computational budget (especially the 60 s limit) presents some difficulties for MCTS. As MCTS uses the results of simulations from each node to estimate their worth, it usually has a warmup period during which its decisions may be poor due to the lack of available information. As more time is allowed, node evaluation estimates improve and results are expected to be more consistent. The reduced time is an advantage for MS also because of its inexistent overhead, as opposed to MCTS which must traverse the tree from the root to a nonexpanded node at each iteration, must create and maintain the tree structure in memory, and must propagate simulation results upward.

The results indicate that even with 60 s, MCTS performs better than MS in most instances; this shows that the search is able to quickly focus on the more promising branches. As expected, performance is further improved with the increased time limit of 600 s.

Table 2 Results for the stacking problem: average number of relocations over 10 runs with the MS and MCTS algorithms, for large instances from [34], with a time limit of 60 s (left-hand side columns) and 600 s (right-hand side columns)

Instance	MS (60 s)	MCTS (60 s)	MS (600 s)	MCTS (600 s)
2-A	52872.6	52358.1	52794.3	51854.5
2-B	50820.6	50760.2	50799.1	50727.3
2-C	49450.7	48880.0	49249.2	47514.3
2-D	50227.2	50230.6	50058.9	49090.4
3-A	23148.2	23124.3	22568.4	22502.2
3-B	24608.7	24665.6	24137.5	24201.0
3-C	24627.9	24738.3	23915.3	24036.8
3-D	24130.7	24065.4	23349.3	23510.7
4-A	14439.4	14441.0	14159.9	14132.4
4-B	13355.9	13429.9	13204.6	13150.5
4-C	13981.8	14050.8	13720.9	13854.0
4-D	14796.0	14865.8	14501.4	13975.5
10-A	3524.9	3484.1	3494.1	3289.5
10-B	4031.3	3790.9	4013.4	3751.4
10-C	3644.3	3497.1	3606.1	3421.7
10-D	3416.3	3437.9	3390.3	3410.4
20-A	883.9	887.5	882.6	885.3
20-B	1029.4	950.6	1022.8	842.2
20-C	1098.0	977.1	1098.0	975.7
20-D	1178.0	1159.2	1178.0	1158.9
40-A	79.0	53.7	79.0	53.4
40-B	59.0	30.5	59.0	9.0
40-C	41.0	32.1	41.0	29.3
40-D	154.0	99.7	154.0	94.3

5 Recursive Circle Packing

The recursive circle packing problem (RCPP) originates from the tube industry, where shipping costs represent an important fraction of the total cost of product delivery [37]. Tubes are produced in a continuous extraction machine and cut to the length of the container inside of which they will be shipped. Before being placed in the container they may be inserted into other, wider tubes, so that usage of container space is maximized—a process called *telescoping*. As all the tubes occupy the full length of the container, maximizing container load is equivalent to maximizing the area filled with circles (or, more precisely, rings/annuli) in a section of the container.

This problem is evidently more general than circle packing, which is known to be NP-complete (see, e.g., [38]). We propose a heuristic method for tackling it, which has proven to be able to produce very good solutions for practical purposes.

A nontechnical, general overview of circle packing is presented in [39]; for a bibliographic review article see [40], which surveys the most relevant literature on efficient models and methods for packing circular objects/items into regions in the Euclidean plane; objects/items and regions considered are either two- or three-dimensional. A survey of industrial applications of circle packing and of methods for their solution, both exact and heuristic, is presented in [41].

In the base RCPP, a number A of tubes are available for packing in a container of width W and height H, in such a way that the value of the packing is maximum. Let $\mathscr{A} = \{1, \ldots, A\}$ be the index set of the tubes; each tube $i \in \mathscr{A}$ is characterized by an external radius r_i^{ext} and an internal radius r_i^{int}, and may be placed in the container or not. A formulation in mixed integer nonlinear programming, provided in [37], considers:

- binary variables w_i, for all $i \in \mathscr{A}$, where $w_i = 1$ if tube i is placed directly inside the container, $w_i = 0$ otherwise;
- binary variables u_{ki} for $k, i \in \mathscr{A}$ such that $r_k^{\text{int}} \geq r_i^{\text{ext}}$, where $u_{ki} = 1$ if tube i is placed directly inside tube k, $u_{ki} = 0$ otherwise (only required if $r_k^{\text{int}} \geq r_i^{\text{ext}}$; other pairs (k, i) are not excluded for facilitating notation);
- position variables (x_i, y_i) of the center of tube i, for all $i \in \mathscr{A}$ (only relevant if i is packed).

Each loaded tube is placed within the bounds of a container, which we assume to be a rectangle with vertices $(0, 0)$, $(W, 0)$, $(0, H)$, and (W, H). The constraints are the following:

- inserted tubes must be completely inside the container;
- loaded tubes may be placed either directly in the container or inside other tubes;
- for each pair of tubes (i, j) directly placed in the container, the distance between them must be greater than or equal to the sum of their external radii;
- the above constraint is likewise applied for each pair of tubes (i, j) directly placed inside the same tube k;
- if tube i is placed directly in tube k, their centers must be close enough for i to remain completely inside k.

The objective of this problem is to maximize the value of the packing, i.e., the sum of a user-defined value v_i for loaded tubes:

$$\text{maximize } V = \sum_{i \in \mathscr{A}} v_i \left(w_i + \sum_{k \in \mathscr{A}} u_{ki} \right). \tag{7}$$

We now describe a heuristic method for quickly constructing a solution to the RCPP. The method begins with an empty container and iteratively inserts new tubes either directly into the container or into a previously packed tube. An auxiliary set $\mathscr{O}$ of *open objects* is used, which initially has the container as its only element. An object (a tube or the container) is said to be open while it is possible to insert at least one of the remaining tubes into it. Whenever a new tube is packed, it is added to $\mathscr{O}$;

and when it is found that no tubes can be inserted into an object, it is removed from $\mathcal{O}$. The algorithm packs a new tube per iteration until either all available tubes have been packed or $\mathcal{O}$ becomes empty.

In order to prioritize telescoping—which seems intuitively advantageous because value is gained without further occupying the container—we choose, if possible, to insert the next tube into the open object $o \in \mathcal{O}$ with minimum free space/area. It is then checked if at least one tube can be inserted into o: if it is possible, we move on to the next step in the algorithm; otherwise, o is removed from $\mathcal{O}$ and the object having the next minimum free space is checked. If during this selection $\mathcal{O}$ becomes empty, the algorithm stops and the current solution is returned.

After selecting the object o into which a tube will be inserted, the actual tube to be inserted is selected. In this step, the algorithm greedily chooses the tube t which has the maximum estimated *value-to-area ratio*. This ratio is an estimate of the total value of a tube and all tubes that can potentially be telescoped into it, divided by its area. It is approximated in a manner similar to the greedy heuristic for the knapsack problem, which is based on the value-to-weight ratio of items.

Finally, from the set of positions of tube t in the open object o, a position p with minimum ordinate is chosen; for tiebreaking, the position with smallest abscissa is selected. An iteration ends by inserting tube t into object o at position p, and updating $\mathcal{O}$ to include t.

Since the position variables (x_i and y_i) in the mathematical model are real variables, the set of positions for a tube is often infinite. In the computation of candidate positions for tube t, we reduce this possibly infinite set to a finite set through a number of simple rules. When inserting a new tube t directly into the container (Fig. 7), the candidate positions considered are:

- the two positions placing t at the bottom corners of the container;
- for each tube u already packed directly into the container, include all positions where t is tangent to u and to any wall of the container;
- for each pair of tubes (v, w) already packed directly into the container, include all positions where t is tangent to both v and w.

Similarly, when t is being inserted into a wider, previously packed tube t' (Fig. 8), the set of candidate positions includes:

- the position placing t at the bottom center of t';
- for each tube u packed directly into t', include all positions that are tangent to both t' (from the inside) and u (from the outside);

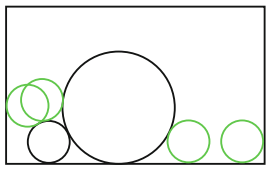


Fig. 7 Circle packing inside a rectangle: positioning possibilities given previously placed, fixed circles (in *black*)

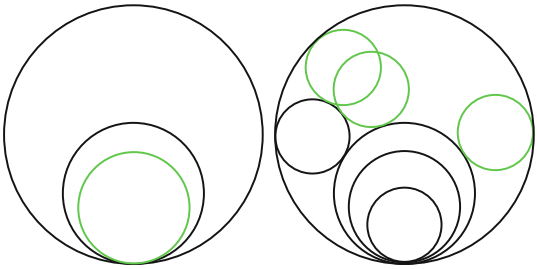


Fig. 8 Circle packing inside another circle (telescoping): positioning possibilities given previously placed, fixed circles (in *black*)

- for each pair of tubes (v, w) already packed directly into t', include all positions that are tangent to both v and w;

After the set of candidate positions is computed, positions violating any constraint (e.g., positions where t overlaps with a packed tube) are discarded. It must be noted that with this simplification we are excluding the majority of the original problem's search space, so the property of proven optimality is lost even when this restricted search space is fully explored. Figure 9 shows an example of an optimal solution which cannot be generated by the described method; whichever the first tube is, it is not at a corner of the container.

The above construction method is purely greedy, leading to a single solution if the same data is provided multiple times. The method is converted into a semi-greedy algorithm by introducing a probabilistic component into one of the choices; in our implementation, the position into which a new tube is inserted is randomly selected, giving higher probability to positions closer to the greedy decision. The full construction method is summarized in Algorithm 4.

This semi-greedy variant is used in both algorithms compared in the computational experiment. The semi-greedy (SG) algorithm consists in repeating constructions with different pseudorandom number generator seeds, whereas MCTS uses it for the simulation step. As in the stacking problem, MCTS uses a single-expansion strategy and scores nodes using the average-weighted exploration term.

The computational experiment included six instances of the RCPP, adapted from [37]. The two algorithms were run 10 times on each instance, with a time limit of 600 s. Table 3 reports the average total value of the best solution found after 60 s, and at the end of the full 600 s period.

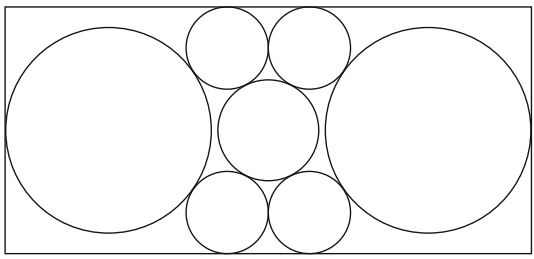


Fig. 9 An optimal solution which is not contained in the restricted search space

Table 3 Average value packed in a container, for 10 independent observations, for the RCPP with semi-greedy construction and with Monte Carlo tree search, with a time limit of 60 s (left-hand side columns) and 600 s (right-hand side columns)

Instance	SG (60 s)	MCTS (60 s)	SG (600 s)	MCTS (600 s)
Large03	3660023.3	3660021.8	3660024.2	3660023.5
Large05	4140042.1	4140044.0	4140043.0	4140047.1
Large16	30311008.0	30625928.6	30311016.6	31192827.3
Small03	938000.0	950000.0	940000.0	957000.0
Small05	1090000.0	1120000.0	1090000.0	1120000.0
Small16	10438035.5	10388039.2	10450036.6	10534032.5

Instances adapted from [37]

Although the number of instances is small, the results indicate a superior performance of MCTS after 60 s, further widening the gap when the full 600 s are allowed. This is an expected consequence of allowing extra time, as more information is gathered and MCTS's estimates of node values are refined.

Algorithm 4: Construction heuristic for recursive tube packing.

Data: container C and set of available tubes $\mathscr{A}$
Result: set $\mathscr{S}$ of tubes packed and their respective positions

$\mathscr{S} \leftarrow \{\}$
$\mathscr{O} \leftarrow \{C\}$
while $\mathscr{O} \neq \{\}$ *and* $\mathscr{A} \neq \{\}$ **do**
  $o \leftarrow$ element of $\mathscr{O}$ with minimum unused area
  foreach $t \in \mathscr{A}$ (from largest to smallest value-to-area ratio) **do**
    $\mathscr{P} \leftarrow$ positions for t inside o
    if $\mathscr{P} \neq \{\}$ **then**
      choose position $p \in \mathscr{P}$ (*semi-greedy choice*)
      $\mathscr{S} \leftarrow \mathscr{S} \cup \{(t, p)\}$
      $\mathscr{O} \leftarrow \mathscr{O} \cup \{t\}$
      $\mathscr{A} \leftarrow \mathscr{A} \setminus \{t\}$
      break
  if could not place any tube inside o **then**
    $\mathscr{O} \leftarrow \mathscr{O} \setminus \{o\}$
return $\mathscr{S}$

6 Conclusion

Tree search is at the heart of the solution of combinatorial optimization problems. The use of simulation to estimate node values is twofold advantageous: it provides quick solutions to the problem; and it prevents misleading evaluations given by poor bounds.

In this work, we propose the use of tree search in combination with problem-specific heuristics for three relevant problems in logistics, to provide better estimates of node values and speed up convergence toward good solutions. In the first application—the number partitioning problem, arising e.g., in load balancing—tree search exploits the quality of the well-known Karmarkar–Karp construction heuristic. The second application is the stacking problem, common in container port operations and in handling warehouse storage, for which full simulation is necessary to evaluate even the earliest placement decisions. Finally, tree search is applied to the recursive circle packing problem—a generalization of circle packing in rectangles—where, once more, early decisions during construction may not be accurately assessed before the solution is complete. In the two last applications, a random component is introduced in the construction heuristic for the sake of diversification. A simulation based on such construction heuristics builds a feasible solution, whose value is used in the evaluation of all nodes in the path between the root of the search tree and the node from which the simulation originated.

In logistics, many situations are studied using simulation models, due to the difficulty in fully characterizing them as formal mathematical models. Simulation-based optimization is a general framework for improving solutions for these models; tree search provides a systematic way for using the simulation models already available in an optimization context. One promising application area with relevance in logistics is resource-constrained scheduling; construction rules can be used for simulation, and decisions complementary to these rules can be implicitly explored under the tree search.

The integration of the paradigms of simulation and, potentially, exact search, is a promising step toward the solution of hard problems, whose formulation in mathematical optimization is too loose for mixed integer solvers to provide acceptable solutions. It also enlarges the scope in which Operations Research exact methods can be applied, in the sense that the simulation may deal with problems which are not linear or even well-defined. The only requirements in this regard would be to contain all relevant solutions in the search tree, and to have consistent evaluations provided by simulation. Although complete enumeration is usually a remote possibility, solutions found on the searched path may provide excellent, realistic approximations, as has been illustrated with the study cases of number partitioning, stacking, and recursive circle packing.

Acknowledgments This work was partially funded by PhD grant SFRH/BD/66075/2009 from the Portuguese Foundation for Science and Technology (FCT).

References

1. Land, A.H., Doig, A.G.: An automatic method of solving discrete programming problems. Econometrica **28** (1960) 497–520
2. Lawler, E.L., Wood, D.E.: Branch-and-bound methods: A survey. Operations Research **14** (1966) 699–719

3. Buchheim, C., Caprara, A., Lodi, A.: An effective branch-and-bound algorithm for convex quadratic integer programming. Mathematical Programming **135** (2012) 369–395
4. Ng, C., Wang, J.B., Cheng, T.E., Liu, L.: A branch-and-bound algorithm for solving a two-machine flow shop problem with deteriorating jobs. Computers & Operations Research **37** (2010) 83–90
5. Bazin, J., Li, H., Kweon, I.S., Demonceaux, C., Vasseur, P., Ikeuchi, K.: A branch-and-bound approach to correspondence and grouping problems. IEEE Transactions on Pattern Analysis and Machine Intelligence **35** (2013) 1565–1576
6. Delling, D., Goldberg, A.V., Razenshteyn, I., Werneck, R.F.: Exact combinatorial branch-and-bound for graph bisection. In: Proceedings of the 14th Meeting on Algorithm Engineering and Experiments (ALENEX'12), Society for Industrial and Applied Mathematics (2012) 30–44
7. Ginsberg, M.L., Harvey, W.D.: Iterative broadening. Artificial Intelligence **55** (1992) 367–383
8. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics **4** (1968) 100–107
9. Korf, R.: Depth-first iterative-deepening: An optimal admissible tree search. Artificial Intelligence **27** (1985) 97–109
10. Gomes, C.P., Selman, B., Crato, N., Kautz, H.: Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. Journal of Automated Reasoning **24** (2000) 67–100
11. Fischetti, M., Monaci, M.: Exploiting erraticism in search. Operations Research **62** (2014) 114–122
12. Bouzy, B.: Associating shallow and selective global tree search with Monte Carlo for 9×9 Go. In: Computers and Games. Springer (2006) 67–80
13. Juille, H.R.: Methods for Statistical Inference: Extending the Evolutionary Computation Paradigm. PhD thesis, Waltham, MA, USA (1999)
14. Coulom, R.: Efficient selectivity and backup operators in Monte-Carlo tree search. In: Proceedings of the 5th International Conference on Computers and Games. CG'06, Berlin, Heidelberg, Springer-Verlag (2007) 72–83
15. Winands, M.H., Björnsson, Y., Saito, J.T.: Monte-Carlo tree search solver. In: Computers and Games. Springer (2008) 25–36
16. Takeuchi, S., Kaneko, T., Yamaguchi, K.: Evaluation of Monte Carlo tree search and the application to Go. In: 2008 IEEE Symposium On Computational Intelligence and Games (CIG), IEEE (2008) 191–198
17. Gelly, S., Kocsis, L., Schoenauer, M., Sebag, M., Silver, D., Szepesvári, C., Teytaud, O.: The grand challenge of computer Go: Monte Carlo tree search and extensions. Communications of the ACM **55** (2012) 106–113
18. Browne, C.B., Powley, E., Whitehouse, D., Lucas, S.M., Cowling, P.I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S.: A survey of Monte Carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in Games **4** (2012) 1–43
19. Sabharwal, A., Samulowitz, H., Reddy, C.: Guiding combinatorial optimization with UCT. In: Proceedings of the 9th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2012), Springer (2012) 356–361
20. Kocsis, L., Szepesvári, C.: Bandit based Monte-Carlo planning. In Fürnkranz, J., Scheffer, T., Spiliopoulou, M., eds.: Machine Learning: ECML 2006. Volume 4212 of Lecture Notes in Computer Science. Springer, Berlin Heidelberg (2006) 282–293
21. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning **47** (2002) 235–256
22. Garey, M.R., Johnson, D.S.: Computers and Intractability. W. H. Freeman, New York (1979)
23. Mertens, S.: The easiest hard problem: Number partitioning. In Percus, A., Istrate, G., Moore, C., eds.: Computational Complexity and Statistical Physics, New York, Oxford University Press (2006) 125–139
24. Mertens, S.: Phase transition in the number partitioning problem. Physical Review Letters **81** (1998) 4281–4284

25. Alidaee, B., Glover, F., Kochenberger, G.A., Rego, C.: A new modeling and solution approach for the number partitioning problem. Journal of Applied Mathematics and Decision Sciences **9** (2005) 113–121
26. Koyutürk, M., Aykanat, C.: Iterative-improvement-based declustering heuristics for multi-disk databases. Information Systems **30** (2005) 47–70
27. Karmarkar, N., Karp, R.: The differencing method of set partitioning. Technical Report UCB/CSD 82/113, University of California - Berkeley, Computer Science Division (1982)
28. Korf, R.E.: A complete anytime algorithm for number partitioning. Artificial Intelligence **106** (1998) 181–203
29. Pedroso, J.P., Kubo, M.: Heuristics and exact methods for number partitioning. European Journal of Operational Research **202** (2010) 73–81
30. Dekker, R., Voogd, P., Asperen, E.: Advanced methods for container stacking. In Kim, K.H., Günther, H.O., eds.: Container Terminals and Cargo Systems. Springer, Berlin Heidelberg (2007) 131–154
31. Hartmann, S.: A general framework for scheduling equipment and manpower at container terminals. OR Spectrum **26** (2004) 51–74
32. Avriel, M., Penn, M., Shpirer, N.: Container ship stowage problem: Complexity and connection to the coloring of circle graphs. Discrete Applied Mathematics **103** (2000) 271–279
33. Avriel, M., Penn, M., Shpirer, N., Witteboon, S.: Stowage planning for container ships to reduce the number of shifts. Annals of Operations Research **76** (1998) 55–71
34. Rei, R.J., Pedroso, J.P.: Tree search for the stacking problem. Annals of Operations Research **203** (2013) 371–388
35. Kim, K.H., Hong, G.P.: A heuristic rule for relocating blocks. Computers and Operations Research **33** (2006) 940–954
36. Caserta, M., Voß, S., Sniedovich, M.: Applying the corridor method to a blocks relocation problem. OR Spectrum **33** (2011) 915–929
37. Pedroso, J.P., Cunha, S., Tavares, J.N.: Recursive circle packing problems. International Transactions in Operational Research (2014). doi:10.1111/itor.12107
38. Lenstra, J., Rinnooy Kan, A.: Complexity of packing, covering, and partitioning problems. In Schrijver, A., ed.: Packing and Covering in Combinatorics. Mathematisch Centrum, Amsterdam (1979) 275–291
39. Stephenson, K.: Circle packing: A mathematical tale. Notices of the American Mathematical Society **50** (2003) 1376–1388
40. Hifi, M., M'Hallah, R.: A literature review on circle and sphere packing problems: Models and methodologies. Advances in Operations Research **2009** (2009) 1–22
41. Castillo, I., Kampas, F.J., Pintér, J.D.: Solving circle packing problems by global optimization: Numerical results and industrial applications. European Journal of Operational Research **191** (2008) 786–802

Part II
Scheduling Problems

Integrated Solutions for Delivery Planning and Scheduling in Distribution Centres

Galina Merkuryeva and Vitaly Bolshakov

Abstract This chapter presents integrated solutions for product delivery planning and scheduling in distribution centres. Integration of cluster analysis, forecasting, computer simulation, metaheuristic optimisation and fitness landscape analysis allows improving planning decisions at both tactical and operational levels. Scheme for integrated solutions is provided. Integrated solutions are described and illustrated by a demonstration case for a regional distribution centre and a large network of retail stores. Cluster analysis and classification methods are applied to determine typical demand patterns and corresponding tactical product delivery plans. A multi-objective optimisation approach is introduced for grouping stores by their geographical location and demand data. Different simulation optimisation scenarios to define the optimal delivery routes and schedules at the operational planning level are considered and compared. Potential applications of a fitness landscape analysis for adjusting an optimisation algorithm are described and applied for product delivery scheduling. Two-stage vehicle routing supplement with vehicle scheduling is described and shown in application experiments. Optimisation techniques described in this chapter are applicable to solve specific routing and scheduling tasks in logistics.

1 Introduction

To ensure business competitiveness, modern management practices require application of different methods in the fields of information technology and operations research. To find the best solution to the problem, these methods must be integrated to complement each other for mutual benefit. Cluster analysis, computer simulation and metaheuristic optimisation techniques may be applied to provide an integrated planning and scheduling of deliveries from a distribution centre (DC) to a net of regional stores.

G. Merkuryeva (✉) · V. Bolshakov
Riga Technical University, Riga, Latvia
e-mail: Galina.Merkurjeva@rtu.lv

V. Bolshakov
e-mail: Vitalijs.Bolsakovs@rtu.lv

M. Mujica Mota et al. (eds.), *Applied Simulation and Optimization*,
DOI 10.1007/978-3-319-15033-8_5

1.1 What is Problem Complexity?

Product delivery planning and scheduling is a high commercial priority task in transport logistics. In real-life applications the problem has different stochastic performance criteria and conditions. Optimisation of transportation schedules itself is computationally time-consuming task which is based on the data from tactical planning of weekly deliveries. This chapter also focuses on the methodology that allows reducing the effect of the demand variation on the product delivery planning and avoid numerous time-consuming planning adjustments and high computational costs.

In distribution centres, this problem is related to deliveries of various types of goods to a net of stores in predefined time windows, taking into account transportation costs and product demand variability. The problem has a high number of decision variables, which complicates the problem solution process. In practice, product demand from stores is variable and non-deterministic. As a result, the product delivery tactical plan that is further used for vehicle routing and scheduling has to be adjusted to real demand data, and product delivery re-planning supervised by a planner is often required. This task is very time consuming and requires specific knowledge and experience of planning staff in this domain.

1.2 What is Motivation for Integrated Solutions?

An acceptable vehicle schedule can be created with the help of commercial scheduling software usually based on heuristic optimisation. However, in practice a schedule created with heuristic algorithms cannot satisfy all constraints of the real-life problem. In this case, an analyst needs to modify or correct a schedule generated by a standard software in order to satisfy the problem specific constraints and to adapt this schedule to a new information received by a planner. Simulation modelling can be applied in this case to numerically evaluate the efficiency of a new schedule candidate.

Integration of different techniques such as cluster analysis, forecasting, computer simulation and optimisation to product delivery planning and scheduling allows improving tactical and operational decisions in logistics distribution centres. A cluster analysis of product demand data of stores allows identifying typical dynamic demand patterns and associated product delivery tactical plans. Customer regional clustering based on multiple criteria is performed through multi-objective optimisation. Metaheuristic optimisation provides effective techniques to define optimal parameters of product transportation and delivery schedules. Vehicle scheduling may be performed also for the routed solution. Fitness landscape analysis allows enhancing the optimisation process and tuning of parameters of optimisation algorithms. An attention should be given to simulation optimisation of the vehicle schedule, which allows incorporating stochastic data. Integration of these technologies advances traditional optimisation techniques known in operations management and lead to coordinated decisions in the area of delivery planning and scheduling at different management (strategic, tactical and operational) levels.

1.3 What are the Main Objective and the Problem Solution?

The main objective is to prepare an effective tactical plan for product deliveries from DC to a net of stores for the upcoming week. In practice, the real demand can be very different from the expected or average demand. Hence, significant changes should be made in the delivery plan for each new week. The reasons for the demand variance can be product demand seasonal effects or marketing events. In this case, it is very important to reduce the effect of demand variation on the delivery planning process and avoid numerous time-consuming adjustments of the base delivery plan.

The problem solution is a detailed delivery plan, in which schedules, routes and amounts of goods to be delivered are defined as the best ones for the input data defined. Input data contain information on the historical demand and location of the stores, available vehicles for the product transportation and existing rural delivery routes. *Additional constraints* such as time windows for product deliveries to specific stores need to be taken into account. An optimal delivery plan should satisfy the following criteria:

1. An amount of goods delivered to the stores should be equal to the demand of these stores for a particular day.
2. Product delivery costs have to be minimised. This implies sub-criteria such as the number of vehicles used to deliver all goods should be decreased, and transportation costs should be minimised by optimising delivery routes and schedules.

2 Integrated Solution Scheme

This solution scheme provides selecting an appropriate product delivery tactical plan for the upcoming week and optimising product transportation routes and delivery schedules. This is achieved by integration of a cluster analysis to define typical product dynamic demand patterns and identify an appropriate demand cluster and tactical weekly delivery plan, and using simulation and optimisation techniques to model and optimise vehicle routes and delivery schedules for product deliveries.

Vehicle routing and schedule optimisation is based on the data from tactical planning for a week delivery. At the same time, a weekly delivery plan is dependent on the data about a number of goods to be delivered to stores in a particular day of a specific week and geographical allocation of stores. In practice, historical data of store demands can be very different from expected or average one, which is determined in a predefined or base plan. Thus, significant changes should be made in the base delivery plan to be adjusted for each new week. So, it is reasonable to specify typical patterns of dynamic daily demand for different planning weeks and introduce several base plans each representing an appropriate product delivery timetable for a specific demand pattern. This will reduce the work of adjusting a typical or base delivery plan to the current situation. Since there are now more typical delivery plans that are based on typical demand patterns, the work will be reduced to making a decision, which delivery plan should be used for the next week and small adjustments of it still

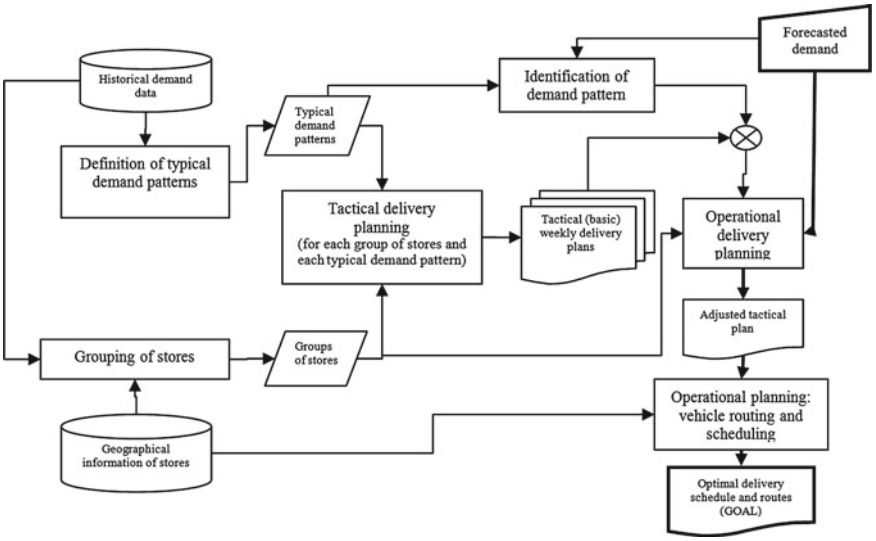


Fig. 1 Scheme of integrated solution

may be required. In addition, selecting the most suitable delivery plan may ensure better scheduling solutions and reduce their computational costs.

The integration scheme for the problem solution (Fig. 1) includes the following main tasks [20]:

1. Definition of typical dynamic demand patterns by clustering historical daily demand data available for different planning weeks.
2. Grouping of stores based on their geographical locations to leverage the total product demand over regions.
3. Tactical weekly delivery planning performed for each group of stores and each demand pattern.
4. Identification of a specific demand pattern based on the classification model created for typical dynamic demand patterns and selection of an appropriate tactical delivery plan for the new week.
5. Adjustment of a selected tactical weekly delivery plan to a new or forecasted demand.
6. Vehicle routing and scheduling, e.g. by using metaheuristic optimisation methods [21, 22].

3 Cluster Analysis of Dynamic Demand Data

This section describes methods for determination and recognition of weekly patterns of dynamic demand data. Determination of patterns of similar dynamic demand data allows introducing base delivery plans each representing an appropriate product

delivery timetable for a specific demand pattern. Described methods allow determining a number of typical demand patterns based on historical demand data, as well as defining characteristic features of these patterns and identifying which weekly delivery plan would be the most suitable for the forthcoming week. An example of cluster analysis of dynamic demand data is provided.

3.1 Motivation

It is assumed that it is possible to specify *typical patterns* of dynamic daily demand for different planning weeks and introduce several *base plans* each representing an appropriate product delivery timetable for a specific demand pattern [17].

The required weekly *tactical delivery plan* is found based on information on the weekly demands during one year. The objective is to find certain dynamic demand patterns, which combines weeks into groups in the way that demand data are similar for all weeks within a specific group but different from those weeks that belong to other groups.

In this case, a cluster analysis of historical demand data [27] provides an opportunity to divide a variety of planning weeks into clusters and to find a number of clusters that represent weeks with a specific demand pattern. It also gives information for the construction of the classification model to identify which weekly delivery plan would be the most suitable for the forthcoming week [17].

Thus a cluster analysis of dynamic demand data is used to:

1. Find a number of typical dynamic demand patterns and corresponding clusters of planning weeks;
2. Construct a classification model that for any week allows determining an appropriate demand pattern, allocating a specific week to one of previously defined clusters and determining correspondent product delivery plan.

3.2 Determination of Typical Dynamic Demand Patterns

To identify typical dynamic demand patterns based on historical demand data, or its observations, the k-means clustering algorithm [15] is applied. It divides n observations into a user-specified number k of clusters, in which each observation belongs to a cluster with the nearest mean value representing a cluster centroid. The result is a set of k clusters that are as compact and well-separated as possible.

K-means clustering is a partitioning method that operates on actual observations and creates a single level of clusters. Thus, for large amounts of data, k-means clustering is often more suitable than hierarchical clustering.

K-means clustering uses an iterative algorithm that minimises the sum of distances from each object to its cluster centroid, over all clusters. This algorithm moves objects

between clusters until this sum cannot be further decreased. The result is a set of clusters that are as compact and well separated as possible. Like many other types of numerical minimisations, the solution that k-means clustering reaches often depends on the starting points. It is possible for an algorithm to reach a local minimum, where reassigning any one point to a new cluster would increase the total sum of point-to-centroid distances, but where a better solution does exist. However, it is possible to overcome that problem by performing a cluster analysis multiple times and selecting the best result.

While implementing k-means clustering algorithm, its parameter k that defines a number of the resulted clusters needs to be specified. In case of typical demand patterns' determination, this number corresponds to an approximate number of such patterns and can be calculated using the following methods.

3.3 Definition of an Appropriate Number of Demand Patterns

There exist a number of approaches to find the best number of clusters by completing cluster validity or measuring goodness of the clustering results compared with ones created by other clustering algorithms. Here, an appropriate number of k clusters, or typical demand patterns is defined by using silhouette plots [14]. In this method, a numerical measure of how close each point is to other points in its own cluster compared to points in the neighbouring cluster is defined as follows:

$$s_i = \frac{b_i - a_i}{\min(a_i, b_i)}, \tag{1}$$

where s_i is a silhouette value for point i, a_i is an average dissimilarity of point i with the other points in its cluster, and b_i is the lowest average dissimilarity between point i and other points in another cluster. Higher mean values of silhouettes show better clustering results that determine better clusters giving the best choice for a number of clusters.

Another method uses the Davies–Bouldin (DB) index [9], which defines the average similarity between each cluster and its most similar one and is calculated by a formula:

$$DB_n = \frac{1}{n}\sum_{i=1}^{n} R_i, \tag{2}$$

where n is a number of clusters and R_i is defined as follows:

$$R_i = \max_{i=1,\ldots,n, i\neq j} R_{ij}, \text{ and } R_{ij} = \left(s_i + s_j\right)/d_{ij}, \tag{3}$$

where index R_{ij} is a similarity measure between two clusters and is based on a measure of dispersion s of a cluster i and a dissimilarity measure d_{ij} between two clusters.

3.4 NBTree for Dynamic Demand Pattern Recognition

A classification model that assigns an appropriate demand cluster is presented by an NBTree, which induces a hybrid of decision tree and Naive Bayes classifiers. This algorithm is similar to classical recursive partitioning schemes, except that leaf nodes created are Naive Bayes categorizers instead of nodes predicting a single class [27].

For a specific week and demand time series, a cluster is identified by determining a proper leaf number C according to the decision tree. When the leaf number is known, a cluster is estimated by a formula:

$$C = \arg\max_{c_j=C} P\left(c_j\right) \prod_{i=1}^{m} P\left(a_i|c_j\right), \tag{4}$$

where $P(c_j)$ defines the probability that weekly demand belongs to cluster c_j, and $P(a_i/c_j)$ defines a conditional probability that demand on day a_i belongs to cluster c_j. Probabilities $P(c_j)$ are calculated from clustering results, while $P(a_i/c_j)$ are defined from the classifier according to the above-determined leaf number.

For a specific week, an NBTree allows identifying an appropriate cluster and then choosing weekly tactical delivery base plan corresponding to this cluster. The selected weekly delivery plan is then used for the optimisation of parameters of vehicle schedules.

3.5 Example Applications

Let us assume that historical demand data for 52 weeks are available and specified by weekly demand time series each representing a sequence of points—daily numbers of the product deliveries for a specific week (see Fig. 2).

K-means clustering experiments are performed on the historical demand data for a number of clusters from 2 to 8. Then for each clustering experiment, silhouette plots are built and mean values of silhouettes per cluster are calculated (Fig. 3).

Analysis of silhouettes mean values leads to the conclusion that the best cluster separation could be done at $k = 4$ that gives the highest silhouette mean value equal to 0.558. Clusters 1–3 seem to be appropriately clustered (see Fig. 4).

However, silhouettes values for the cluster 4 are negative. Theoretically, weekly dynamic demands assigned to this cluster could be better allocated to another cluster. These weeks are different due to demand dynamics and specific days, where demand peaks are observed.

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
2885	3390	3891	4115	4612	4687	3371
2831	3553	3859	3785	4432	4899	3527
2763	3548	4067	4631	4838	5057	3511
2951	3820	[illegible]	4360	5075	[illegible]	3345
2595	2731	3101	2988	[illegible]	3524	2643
3150	3459	4339	4377	5187	4956	3545
2394	3229	3643	3693	4018	4411	3583

Fig. 2 Sample demand data

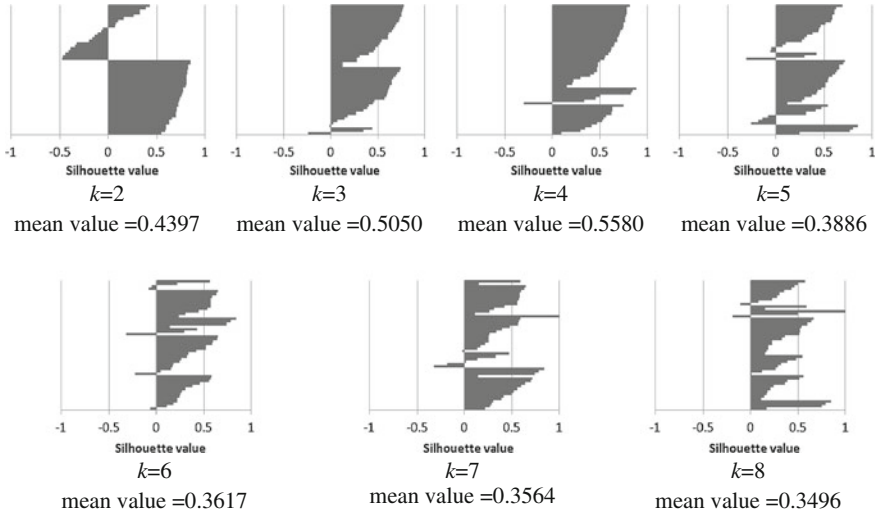


Fig. 3 Silhouette plots for the number of clusters $k = 2$ to $k = 8$

Reallocation of 'unlike' weeks avoids receiving negative silhouette values (see Fig. 5). However, this does not provide an increase of the silhouette mean value as might be expected. In this case, 'unlike' weekly demands behave as a 'noise' in their 'native' clusters, decreasing silhouette values. Then, clustering experiments have been performed with 49 weeks, where three 'unlike' weeks have been excluded from a cluster analysis. This has allowed increasing the silhouette mean value up to 0.5822, while getting the same groups of data clusters 1–3.

The DB index is calculated based on the output data from k-means clustering experiments for k in the range 2–8 clusters (Fig. 6). The lower value of DB index which is equal to 0.899 is received for the number of clusters equal to 4. Thus, the best number of clusters in this case confirms the results of the silhouette plot analysis.

As a result, a number of clusters is set to $k = 4$. It is worth noting that a tactical weekly delivery base plan is defined for a cluster with a silhouette mean value greater than 0.5. In this case, a tactical product delivery base plan is selected, adjusted or

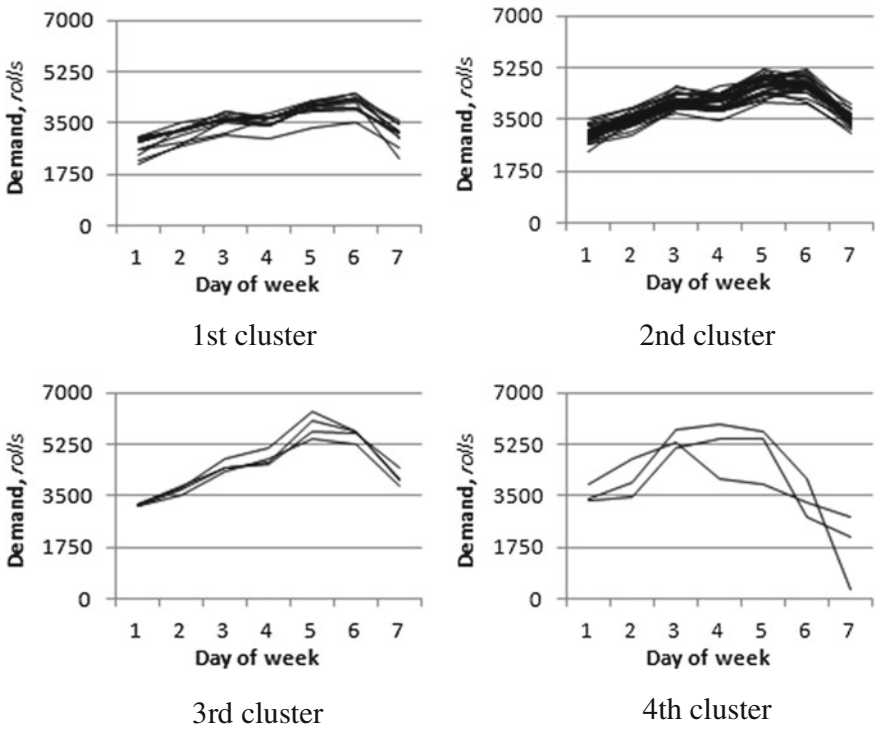


Fig. 4 Sample demand patterns found by k-means analysis for a number of clusters $k = 4$

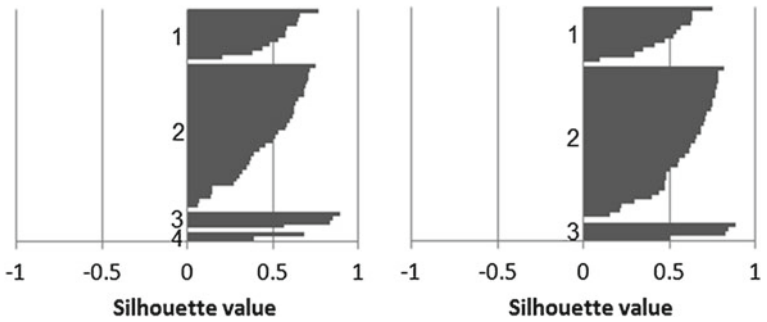


Fig. 5 Silhouette plots for the number of clusters $k = 4$ with reallocation of 'unlike' weeks and for the number of clusters $k = 3$ and 49 sample weeks

built for the first three clusters and not analysed for the last one. Dynamic patterns received for clusters from 1 to 3 are presented in Fig. 4.

To identify an appropriate demand cluster for a specific planning week, NBTree classification model is built (see Fig. 7). This will allow choosing a weekly delivery plan which corresponds to this demand cluster.

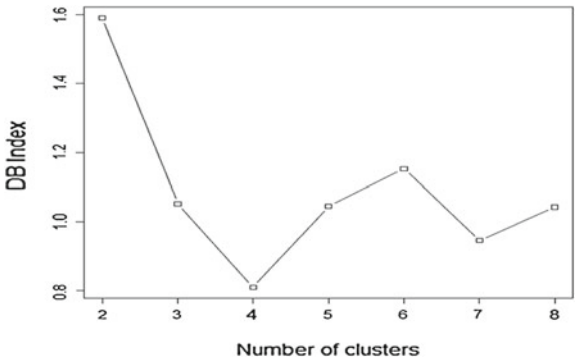
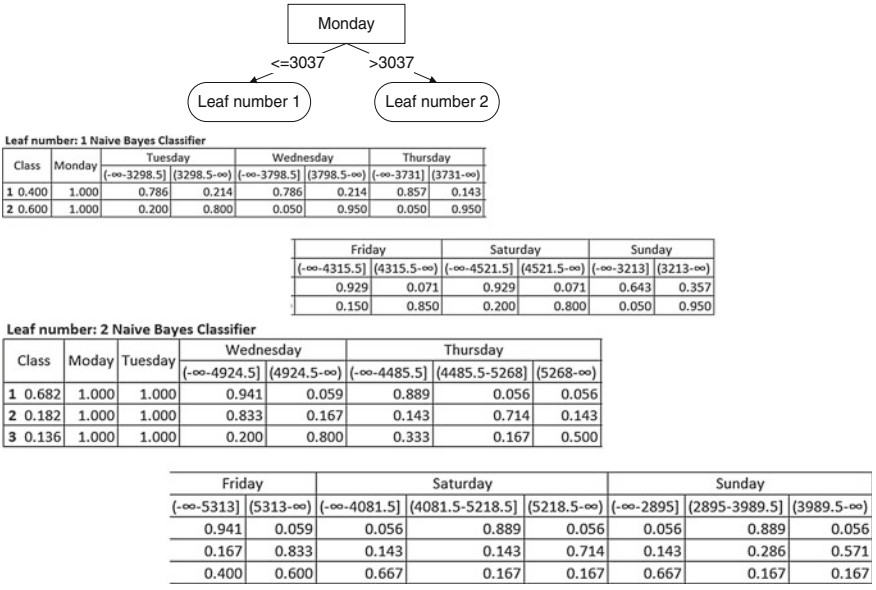


Fig. 6 Davies–Bouldin index for different numbers of clusters



Leaf number: 1 Naive Bayes Classifier

Class	Monday	Tuesday		Wednesday		Thursday	
		(-∞-3298.5]	(3298.5-∞)	(-∞-3798.5]	(3798.5-∞)	(-∞-3731]	(3731-∞)
1 0.400	1.000	0.786	0.214	0.786	0.214	0.857	0.143
2 0.600	1.000	0.200	0.800	0.050	0.950	0.050	0.950

Friday		Saturday		Sunday	
(-∞-4315.5]	(4315.5-∞)	(-∞-4521.5]	(4521.5-∞)	(-∞-3213]	(3213-∞)
0.929	0.071	0.929	0.071	0.643	0.357
0.150	0.850	0.200	0.800	0.050	0.950

Leaf number: 2 Naive Bayes Classifier

Class	Moday	Tuesday	Wednesday		Thursday		
			(-∞-4924.5]	(4924.5-∞)	(-∞-4485.5]	(4485.5-5268]	(5268-∞)
1 0.682	1.000	1.000	0.941	0.059	0.889	0.056	0.056
2 0.182	1.000	1.000	0.833	0.167	0.143	0.714	0.143
3 0.136	1.000	1.000	0.200	0.800	0.333	0.167	0.500

Friday		Saturday			Sunday		
(-∞-5313]	(5313-∞)	(-∞-4081.5]	(4081.5-5218.5]	(5218.5-∞)	(-∞-2895]	(2895-3989.5]	(3989.5-∞)
0.941	0.059	0.056	0.889	0.056	0.056	0.889	0.056
0.167	0.833	0.143	0.143	0.714	0.143	0.286	0.571
0.400	0.600	0.667	0.167	0.167	0.667	0.167	0.167

Fig. 7 NBTree-based classification model

To improve the performance of the classification model, demand data sample size has been increased up to 156 weeks. Two demand time series were generated for each planning week by daily demand uniform change within ±5 %. In a similar way, input data for another 52 weeks have been generated to validate a classification model itself. Built on this data the NBTree-based classification model with an example of the leaf Naive Bayes classifier is given in Fig. 8. In this case, tenfold cross-validation showed that only eight weeks have not been classified correctly, which produced an error value of about 5 %.

```
FRI <= 4294.5
|   TUE <= 3380.5: 1 cluster
|   TUE > 3380.5: Leaf Number 1
FRI > 4294.5
|   TUE <= 3700.5
|   |   MON <= 3062: 2 cluster
|   |   MON > 3062
|   |   |   THU <= 4368.5: Leaf Number 2
|   |   |   THU > 4368.5: Leaf Number 3
|   TUE > 3700.5: Leaf Number 4
```

Leaf Number: 4 Naive Bayes Classifier

Class	Monday	Tuesday	Wednesday			Thursday	
			(-∞-4406]	(4406-5152.5]	(5152.5-∞)	(-∞-5467.5]	(5467.5-∞)
2 0.348	1.000	1.000	0.727	0.182	0.091	0.900	0.100
3 0.522	1.000	1.000	0.067	0.867	0.067	0.929	0.071
4 0.130	1.000	1.000	0.167	0.167	0.667	0.200	0.800

Friday		Saturday			Sunday		
(-∞-5204]	(5204-∞)	(-∞-4299.5]	(4299.5-4903]	(4903-∞)	(-∞-1745.5]	(1745.5-3770]	(3770-∞)
0.900	0.100	0.091	0.818	0.091	0.091	0.727	0.182
0.357	0.643	0.067	0.067	0.867	0.067	0.067	0.867
0.200	0.800	0.667	0.167	0.167	0.667	0.167	0.167

Fig. 8 Detailed NBTree classification model

Table 1 Weekly demand sample data (in roll containers)

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
3231	3462	3842	4437	5021	4868	3391

Let us assume that demand for Friday and Tuesday is 5021 and 3462 roll containers, correspondingly (see Table 1). From the classification model in Fig. 8, the leaf number 4 is selected.

Then a likelihood for cluster 2 is calculated by multiplication of probabilities received from the Leaf Number 4 Classifier for specific week days, i.e. 0.348, 0.727, 0.9, 0.9, 0.818 and 0.727. This probability is equal to 0.122. Likehoods for cluster 3 and 4 are calculated and are equal to $5.2 \cdot 10^{-5}$ and $2.4 \cdot 10^{-5}$, correspondingly.

The probability that weekly dynamic demand belongs to cluster 2 is defined as follows:

$$p(C_2) = \frac{0.122}{0.122 + 5.2 \cdot 10^{-5} + 2.4 \cdot 10^{-5}} \approx 0{,}9994. \tag{5}$$

Similar, probabilities that weekly dynamic demand belongs to cluster 3 or 4 are calculated: $p(C_3) = 0.0004$; $p(C_4) = 0.0002$. Finally, a cluster with the highest probability is selected. As a result, the considered week belongs to cluster 2 with the probability which is very close to 100 %.

For a specific week, an NBTree allows identifying an appropriate demand cluster and then choosing weekly tactical delivery base plan corresponding to this demand cluster. Then, selected weekly delivery plan is used for optimisation of parameters of vehicle schedules.

4 Grouping of Stores Based on Geographical Locations

In case of a large number of stores in a delivery network, grouping of stores in relatively small groups based on their geographical locations allows to simplify the product delivery optimisation task by decreasing its dimension. In this section, the problem statement of store grouping that also provides the uniform distribution of the total demand over geographic regions is given, and problem-solving techniques are described and provided with illustrative examples.

4.1 Optimisation Problem Statement

In practice, weekly delivery planning is done based on the demand and data about store allocations to geographical regions. Grouping of stores based on their geographical locations allows leveraging the total product demand over regions. Let us suppose that all stores are grouped manually, and rearranging regions in case a new store is added may require. Also, it would be desirable to have separation of regions with a similar weekly total demand, or the total demand uniformly distributed over regions.

The use of a cluster analysis for dividing stores into regions according to their locations does not allow getting the total product demand equally distributed between these regions. But the region clustering task may be formulated as a multi-objective optimisation problem.

Input data contains the number of stores n, the number of regions k, two geographical coordinates x_i and y_i for each store i, $i = 1, \ldots, n$ defined in the Cartesian coordinate system and the total weekly demand d_i for each store i.

Decision variables define a region (or cluster) a_i to which a store i is assigned:

$$a_i \in \{1, 2, \ldots, k\}; \; i = 1 \ldots n. \tag{6}$$

Additional auxiliary variables are introduced as:

$$A_j = \{b | a_b = j\}, \tag{7}$$

where A_j is a set of stores assigned to each region, and

$$r(i, j) = \sqrt{\left(x_i - \dot{x}_j\right)^2 + \left(y_i - \dot{y}_j\right)^2}, \tag{8}$$

where $r\,(i, j)$ defines an Euclidean distance from store i to the centroid of cluster j, and $\dot{x}_j$ and $\dot{y}_j$ are mean values of coordinates for all stores in cluster j:

$$\dot{x}_j = \frac{\sum\limits_{i \in A_j} x_i}{|A_j|}, \quad \dot{y}_j = \frac{\sum\limits_{i \in A_j} y_i}{|A_j|}. \tag{9}$$

Two objective functions are introduced in the problem. The first objective function determines how good regions generated from the geographical location point of view are, and the second one defines if the total demand is equally distributed among these regions. Both objective functions are minimised, i.e.:

$$f_1 = \sum_{j=1}^{k} \sum_{i \in A_j} r\,(i, j) \rightarrow \min, \tag{10}$$

$$f_2 = \sum_{j=1}^{k} \left| \sum_{i \in A_j} d_i - \frac{\sum\limits_{i=1}^{n} d_i}{k} \right| \rightarrow \min, \tag{11}$$

where f_1 defines the sum of distances between centroids of the regions and stores assigned to them, and f_2 is the sum of differences of the total demand for each region and the average demand per region. No additional constraints are defined in the optimisation problem.

4.2 Multi-objective Optimisation Algorithm

As the problem has two objective functions, a multi-objective optimisation algorithm should be applied for grouping of stores into geographical regions. Thus for the grouping task, an application of widely used Nondominated Sorting Genetic Algorithm II (NSGA-II) [10] is proposed. Further discussion on application of NSGA-II implemented in the HeuristicLab optimisation framework [32] is given.

The optimisation problem itself is implemented as a multi-objective optimisation problem plug-in of HeuristicLab with integer encoding of solutions and their evaluation by two mathematical functions (10) and (11). Correspondingly, a chromosome representing a solution is defined as a string of n integer numbers and formalised as a vector $(a_1, a_2, \ldots, a_n)$ of decision variables, where $a_i \in [1, k]$.

In experiments with NSGA-II, the following GA operators are applied: (1) discrete crossover operator for integer vectors [13], (2) uniform One Position Manipulator [16] and (3) crowded tournament selector [10].

Parameters of the algorithm may differ for different sizes and complexity of the problem and should be tuned experimentally. For the case discussed in this chapter, with a number of stores equal to 88, the following parameters were determined in experiments: a termination criterion in a number of generations equal to 5000; the population size equal to 200; the crossover rate of 90 %; the mutation rate of 5 %; and 400 selected parents in a new generation.

4.3 Examples of Optimisation Experiments

The results of optimisation experiments for grouping of stores into geographical regions are performed for different numbers of generations with a population size of 200 solutions (see Fig. 9). Here, *demand quality* is equal to the sum of variances of the total demand for each region, and *geographic quality* corresponds to the sum of distances between stores in groups and centroids of these groups. An increase in the number of generations improves the Pareto front of non-dominating solutions by minimising both objective functions. However, when the number of generations exceeds 2000, these improvements are small. Finally, at 5000th generation the grouping of stores into geographical is reached that lead to a uniform distribution of demand between regions. Further improvements become minor for the discussed case.

However, graphical representation of the obtained results shows that solutions with high demand quality give worse results for the second objective function and vice versa. Thus, a solution in the middle of the Pareto front (see Fig. 10, where different regions with the stores assigned are shown) is selected which provides compact clusters of stores in regions. A large number of regions in the central part

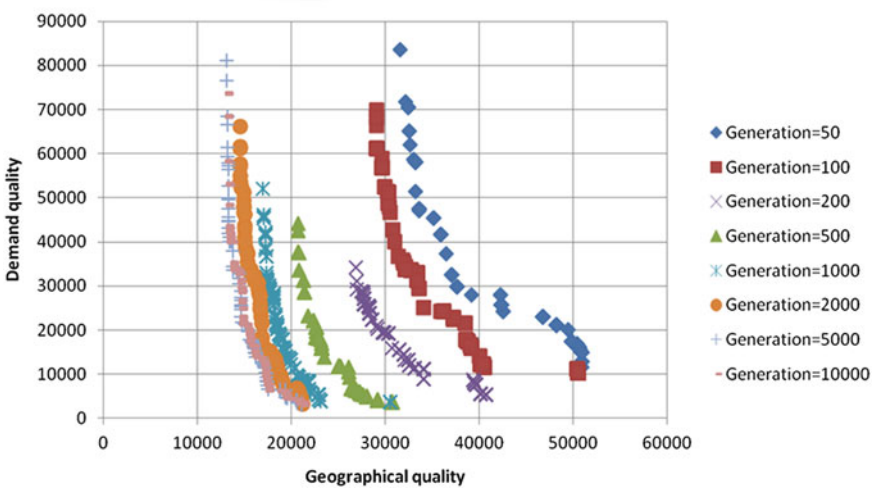


Fig. 9 Pareto fronts for different numbers of generations

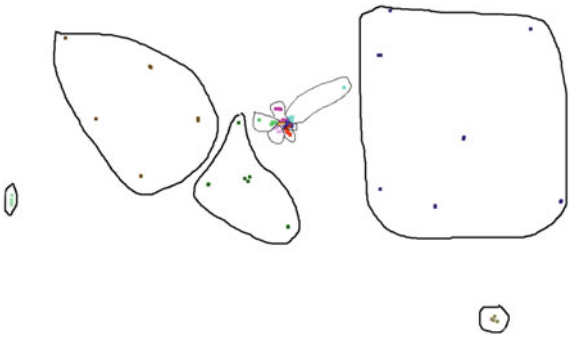


Fig. 10 Solution in the middle of Pareto front

of Fig. 10 may be explained by high density of stores with high product demands in this geographical part, which corresponds to a large city or urban area. Moreover, there are only two regions with demands that are lower than others. Further gradual leverage of the regional demand may be experimentally tested. In the example that is considered, this may worsen the geographical location of high-priority regions (see Table 2).

Further on, for each group of stores and each demand pattern a *weekly delivery base plan* that defines an amount of products to be delivered to stores on specific days of the calendar week (Table 3) is built. It is based on the average demand for each store from a specific group in the planning weeks that belong to a certain demand cluster. Here, a weekly delivery base plan is developed once using knowledge-based heuristics and updated relatively rare.

Table 2 The distribution of the total weekly demand

Region	Total demand	Region	Total demand
1	28,443	7	25,328
2	21,429	8	21,552
3	23,440	9	21,787
4	21,687	10	12,152
5	23,583	11	14,722
6	23,101	12	21,860

Table 3 Sample delivery plan for one week

Day	MON	TUE	WED	THU	FRI	SAT	SUN
Store A	30	35	30	25	45	50	30
Store B	–	25	–	–	35	–	–
Store C	20	–	15	–	10	–	25
Store D	–	25	–	40	–	35	20
Σ	50	85	45	65	90	85	75

Variation in daily deliveries for specific days of the week may produce extra costs on peak delivery rates. Thus, variation in daily deliveries needs to be minimised for each store and each group of stores. Additionally, customers with low demand would not be served each day as it will produce extra transportation costs to deliver small order quantities too often. This could be achieved by simple division of the total weekly demand into a number of delivery dates. For stores with low demand, a number of delivery days is reduced to minimise transportation costs. Afterwards, the weekly delivery base plan still needs to be further customised to correspond with demand dynamics during the week (see Table 3).

To ensure maximal operational effectiveness, the base plan selected is adjusted in order to match the actual customer demand for the forthcoming week. Then, for each day of the week, vehicle routes and/or delivery schedules are defined to minimise their transportation costs.

5 Simulation Optimisation of Vehicle Schedules

When the base delivery plan for the upcoming week is selected and adjusted, product delivery (vehicle) routes and schedules need to be optimised in order to reduce the transportation costs. Two types of a vehicle routing and scheduling problem are discussed. In this section, simulation-based vehicle scheduling when delivery routes are fixed and known is considered. Later on in this chapter a more complex case is introduced, when first the vehicle routes have to be determined and then these routes have to be scheduled.

Here, a vehicle scheduling problem with time windows is discussed, with a description of a vehicle schedule simulation model and scheduling optimisation scenarios.

5.1 Problem Express Analysis

A *vehicle schedule* defines a schedule of deliveries of various types of goods from DC to a network of stores. Distribution *routes or trips* for *vehicles* are fixed. For each route, the following parameters are defined: a sequence of stores (route points), average time intervals for vehicle moving between these points, loading and unloading average times and types of goods to be carried on this route. Goods are delivered to stores in the predefined time windows. For each store, an average demand of goods of each type is defined. Vehicle capacities are limited and known.

Vehicles are assigned to routes and schedules for routes are generated that minimise the total costs of a schedule. Samples of predefined vehicle routes and schedules that define at which time each route has to be started and which vehicle will perform it are shown in Fig. 11. The vehicle idle time is defined as a sum of time periods, when a vehicle is waiting for the next trip in the DC depot.

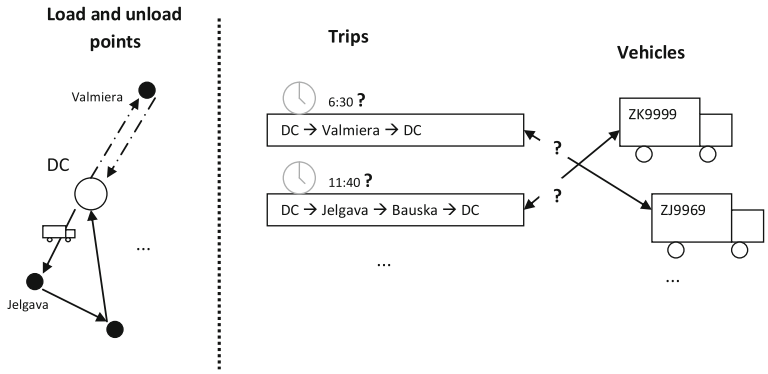


Fig. 11 Examples of vehicle routes

Vehicle Scheduling Problems (VSP) present a class of optimisation problems that are aimed at assigning a set of scheduled trips to a set of vehicles, in such a way, that each trip is associated with one vehicle, and a cost function for all trips is minimised [11, 24]. This problem is often modified with additional constraints, like time windows, different vehicle capacity, etc. Correspondingly, a VSP with Time Windows is denoted as VSPTW. A number of methods to solve VSP problems are proposed in the literature, e.g. integer programming, combinatorial methods, heuristics [11]. Such problems do not have efficient traditional optimisation methods and can be solved by application of evolutionary algorithms. Furthermore, an analysis of a fitness landscape can be used to evaluate the optimisation problem complexity and to select the most appropriate algorithm.

In practice, the VSP also can be complicated by stochastic processes existing in the problem, e.g. when the duration of a trip is a random variable. In this case, evaluation of potential solutions can be made through simulation, and simulation optimisation could be used to solve such problems. Simulation technology provides a flexible tool to determine the optimality of each solution. Therefore, the simulation-based fitness landscape analysis that supposes fitness evaluation of the solution with use of simulation becomes an important task.

5.2 Problem Statement

Decision variables are introduced to assign vehicles v_i to routes and define a start time t_i for each route [21, 22], where i is a route number, v_i is a vehicle assigned to trip i and t_i is start time of the trip i. The *objective function* f is aimed to minimise the total idle time for all vehicles:

$$f = \sum_{i=1}^{N} T_{idle}^{i} \rightarrow \min, \tag{12}$$

where T^i_{idle} is the total idle time for vehicle i; and N is a number of vehicles.

The *problem constraints* are divided into three groups presenting vehicle capacity constraints, delivery time constraints, and gate capacity constraints, correspondingly. In the last case, a number of vehicles that can be loaded in a warehouse simultaneously cannot exceed a number of gates.

Express analysis shows that the problem could have many solutions that are not feasible within defined constraints. This makes a solution search process non-efficient in terms of computational time. To increase optimisation efficiency, all constraints are converted into soft constraints [21], and the objective function f in (12) is modified by introducing penalties taking into account the total number of times when the constraints were not satisfied by a potential solution:

$$f^* = \sum T_{idle} + k_1 T_c + k_2 T_m + k_3 T_0 + k_4 N_{ol} + k_5 N_{ot}, \tag{13}$$

where f^* is the modified objective function; T_c defines the total duration of overlapping trips for one vehicle; T_m defines the total time of window mismatches; T_o and N_{ol} determine the total time and a number of vehicles that exceeded the total working time; N_{ot} is a number of vehicles overloaded. In (13), all indexes for unsatisfied constraints are multiplied with penalty coefficients $k_i > 1$; $i = 1...5$ that artificially increase the value of the objective function and make the fitness of a potential solution worse.

5.3 Simulation of Vehicle Schedules

To estimate fitness of potential schedule solutions, the vehicle schedule simulation model is introduced. It is built as a discrete-event simulation model, for example using AnyLogic simulation software [3]. It is based on the object-oriented conception and presents a simulation model as a set of active objects that are functioning simultaneously and interact with each other.

In the vehicle schedule model [22], each vehicle is modelled as an active object, and its behaviour is described by a state chart that defines vehicle states (e.g. parking, loading, moving and unloading) and transitions between them (see Fig. 12). The objects of each vehicle are aggregated by the model main object. Three classes are defined for store, trip and job objects to specify input data. During the model initialisation it is connected to a database of input data and variable collections of the main active object are set up with data from the database. Processes related to DC operations are simulated. During simulation the constraint violations are monitored and a number of violations are stored in the model.

In the model animation screenshot (Fig. 13), utilisation graphs (timelines) of all vehicles are combined in a Gantt chart for a vehicle schedule where different states of the vehicle are shown with different colours, e.g. the grey colour for parking in DC, green one for loading and blue for moving between route points times.

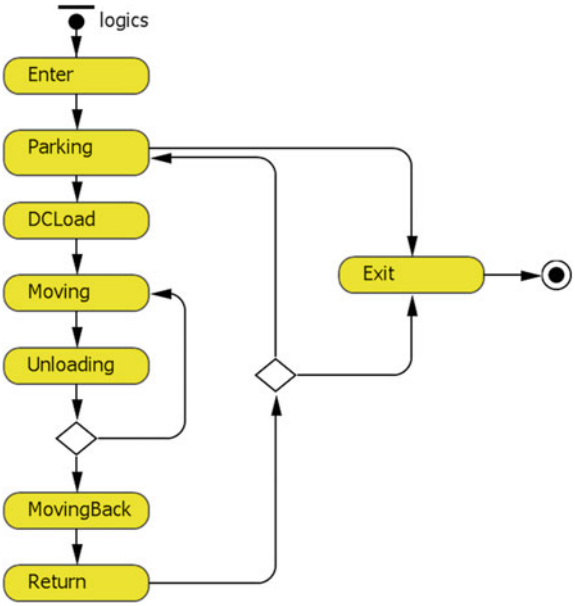


Fig. 12 State chart of active object "Vehicle"

Fig. 13 Screenshot of the simulation model

The utilisation chart for DC gates during the daytime is displayed below the vehicle utilisation timelines. It shows how many loading gates are busy each time. Online statistics about the idle time and completed jobs is provided along the timeline for the corresponding vehicle. A list of performed trips including visited shops is generated for each vehicle.

5.4 Vehicle Schedule Optimisation Scenarios

Example input data of the VSP contains 37 routes, 17 vehicles and 36 stores. Specific parameters of vehicles, stores and routes are defined in the simulation model. The input data for vehicle moving times are interpreted as deterministic and then as stochastic (depends on the optimisation scenario). As simulation output the total idle time for all vehicles is calculated. The number of decision variables that define a vehicle schedule is equal to 74. Thus, exploration of the problem search space requires evaluation of a huge number of possible solutions [23]. Finally, function (13) is used for fitness evaluation of the solutions simulated. Two optimisation scenarios based on using genetic algorithms and a fitness landscape analysis are discussed in this section, and the third scenario based on scheduling of routed solutions is considered in Sects. 6 and 7.

5.4.1 Simulation-Based Optimisation with GA

In this scenario, a genetic algorithm (GA) is used to search for the best combination of the schedule parameters, while simulation model is applied to estimate quality of a schedule generated. The optimisation tool is implemented as a Java class, which interacts with the simulation model via 'Parameter variation' experiment in AnyLogic [18].

In GA, solution candidates of the scheduling problem are encoded as integer vector chromosomes, which length is twice a number of trips (routes). In chromosome, genes with even sequence numbers represent start times of corresponding trips, and ones with odd sequence numbers define a vehicle assigned for this trip. For example trip 1 will be performed by vehicle 2 starting at 12:20 a.m. (Fig. 14).

A mutation operator is introduced that changes one randomly selected trip in the solution candidate. For a selected trip, a new randomly chosen vehicle is assigned, and the start time is shifted by a certain constant value. One-point crossover with rate

2	20	1	600	2	690	1	120	3	490
Trip 1		Trip 2		Trip 3		Trip 4		Trip 5	

Fig. 14 A sample chromosome of the vehicle schedule

of 75 %, above-mentioned mutation operator with rate of 1 %, one elite individual and tournament selection with tournament size of two individuals are involved in algorithm. Termination condition of GA is set to occur when there is no significant improvement of the best solution in population after a large number of generations.

Optimisation results show that a feasible schedule which satisfies all defined constraints can be found. Acceptable results are obtained with a population size of 1000 chromosomes, but larger populations significantly increase computational time. The solution allows decreasing the total idle time from 1140 to 700 min.

5.4.2 Fitness Landscape Analysis and Optimisation Performance

While a genetic algorithm provides very good solutions in VSP optimisation, nonetheless it needs tuning of parameters and operator selection by skilled personal. To make tuning and adjusting of an optimisation algorithm easier *Fitness Landscape Analysis* (FLA) is proposed in the literature [26, 29]. FLA provides methods and techniques for a mathematical analysis of a search space of optimisation problems, and can be applied as a support tool to enhance optimisation of complex systems. It was proposed that the structures of a fitness landscape affect the way, in which a search space is examined by a metaheuristic optimisation algorithm. The fitness landscape analysis would allow getting more information on the problem's properties dependent on a specific optimisation method, which will guide the optimisation process.

FLA techniques apply different strategies for data collection based on simple moves, which generate a trajectory through the landscape (e.g. *random walk*). The *information analysis* interprets a fitness landscape as an ensemble of objects, which are characterised by their form, size and distribution and is based on the information theory. Four information measures are proposed by [30], e.g. the *information content* is a measure of entropy in the system and *partial information content* characterises the modality of the performed walk. Higher information content and partial information content values indicate higher hardness of the analysed problem. The *statistical analysis* proposed in [33], calculates the autocorrelation function in the random walk to measure the ruggedness of the landscape. In case of a high correlation between fitness values the landscape is considered less rugged and thus the problem should be easier. In this section, to analyse the problem fitness landscape the autocorrelation function is successfully used.

To apply FLA in the vehicle schedule optimisation the methodology described in [5] is applied. The optimisation problem is experimentally analysed in a comprehensive way to find how parameters of the problem and selection of optimisation operators and representations of solutions influence both FLA measures and optimisation performance. If the relationships between the fitness landscape measures and optimisation performance are found, afterwards, for the same class of the problem an optimisation algorithm can be adjusted based on the FLA results to a new problem instance.

To perform comprehensive optimisation and analysis of the VSPTW, the above described simulation model [22] was re-implemented as a plug-in of HeuristicLab framework [32] maintaining all model's logic and functionality.

Hereinafter two types of mutation operators are defined for the representation described in this chapter. The single position replacement manipulator (*VSPManipulator*) changes the start time of the trip to a new uniformly distributed random number, but the single position shift manipulator (*VSPShiftManipulator*) shifts the start time with a uniformly distributed random number.

To enhance the quality of optimisation results, permutation encoding for the VSP solutions is introduced. The encoding is based on the Alba encoding [2] for a vehicle routing problem. A chromosome is of the permutation type and contains $m+n$ genes, where n is a number of vehicles and m is equal to a number of trips in the problem. Genes that have values less or equal to m encode a trip number and values greater than m encode delimiters or vehicle designators and define a vehicle number for the next sequence of trips.

The logic of the simulation model is that if no time window constraints are defined for the first trip, it starts at midnight; otherwise it starts at time to match the first customer's window. The next trip starts immediately after the previous one, unless its start time should be delayed to satisfy time windows of customers in the route of this trip. No times are encoded, and hence, the potential solution has no directly encoded idle time, and its trips cannot overlap. Due to the high universality of this encoding, different permutation manipulation operators can be applied in the search of the optimal solution.

A grid of landscape analysis experiments is created to compare values between different landscapes. First, comparison of different mutation operators is performed. Second, comparison between existing and proposed encodings is done. Results of comprehensive analysis experiments are described in [6]. Particularly found, that in a random walk, values of autocorrelation function are slightly lower for the replacement operator. In the up-down walk the situation is the opposite: replacement mutation has higher correlation than shift mutation, but the three artificial problems are different to the others (see Fig. 15; black dots are for *replacement* and green for *shift* mutator). Moreover, the value of the autocorrelation function in random and up-down walks is lower for the permutation encoding, which means that landscapes of this encoding should be more rugged.

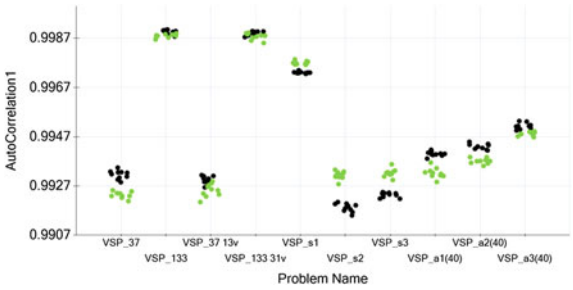


Fig. 15 Autocorrelation in up-down walks

5.4.3 Optimisation with HeuristicLab

A number of VSP optimisation experiments with the algorithms implemented in the HeuristicLab framework are performed. For the integer encoding both Evolution Strategy (ES) and Simulated Annealing (SA) algorithms are experimentally found to be fast and highly successful, but the ES is able to find solutions with better quality [6]. A genetic algorithm is able to find even better solutions, but with larger numbers of evaluations than evolution strategy. Permutation encoding is more effective in optimisation of the VSP, than integer encoding, as algorithms are able to find good solutions in less time. Moreover, almost all idle times are eliminated, and trip overlapping is also avoided. The evolution strategy is more effective than a genetic algorithm also for the permutation encoding. Even though the search space for this type of encoding is more complex and rugged, nevertheless, due to its smaller size the search of the globally optimal solution becomes more effective.

The best results found by different algorithms and for different encodings are presented in Table 4. Each algorithm for a specific problem has been run 10 times and the mean fitness of the best solution in each run has been calculated. A GA involved the population size of 100 chromosomes, tournament selection, 5 % mutation rate and termination criterion defined by 500 generations. ES (20 + 100) strategy has been applied with maximum 1000 generations. Simulated annealing has been run with 3000 iterations and 10 evaluations in each iteration. Each run of GA or ES on a computer with 4-core CPU required from 8 to 10 s, and about 3 s for simulated annealing.

Additional detailed optimisation experiments with similar instances and parameters show that there are relations between values of fitness landscape analysis measures and an ability of the optimisation algorithm to find the best solution. The genetic algorithm with population size of 100 individuals and termination condition of 500 generations is applied in one series, and the evolution strategy (20 + 100) with

Table 4 Mean fitness values of the best solutions found by different algorithms

Problem	Genetic algorithm		Evolution strategy		Simulated annealing
	Integer	Permutation	Integer	Perm.	Integer
VSP_37	13 144.6	2 481	39.6	0	10 414.4
VSP_133	111 196.1	132 475	18 279.2	0	112 539.1
VSP_37(13)	47 313.1	26 367	15 860.1	180	67 904.6
VSP_133(31)	199 750.8	313 750	75 233.6	2 887	230 752.4
VSP_s1	33 759.3	3 600	10 346.1	0	12 741.2
VSP_s2	4 082.1	0	757.5	0	2708.5
VSP_s3	5 666.2	5 382	984.6	0	14 274.8
VSP_a1	12 446.2	6 715	347.7	0	26 530.7
VSP_a2	30 575.8	31 880	6377	0	52 755.6
VSP_a3	10 927.2	11 156	1 257.6	0	36 462.7

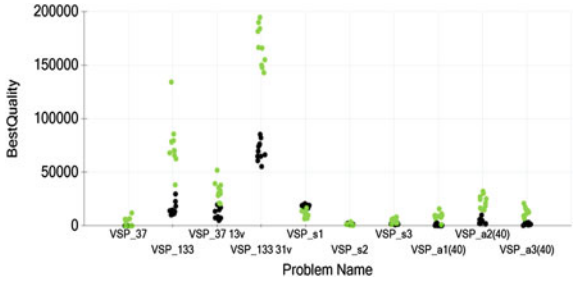


Fig. 16 Quality of best found solutions with ES

crossover and 1000 generations in the second series of experiments. For the problem instances, which are better solved with the shift operator by GA, the autocorrelation for this operator also is higher (Figs. 15 and 16; black dots are for *replacement* and green for *shift* mutator). The same dependency is found for the ES algorithm.

Finally, it is concluded, that optimisation using the ES is the best choice for the solution of a vehicle scheduling problem with time windows. In case of the integer vector encoding is applied, selection of an appropriate mutation operator is based on the measures of the FLA, i.e. an operator, which has the highest autocorrelation value in the up-down walk should be selected.

6 Vehicle Routing

Often the delivery planning at the operational level also requires optimisation of vehicle routes. The section describes optimisation approach for the vehicle routing with time windows. First, the problem statement is given, and then an optimisation algorithm and its experimental adjustment are discussed.

6.1 Problem Statement

The classical statement of the vehicle routing problem with time windows (VRPTW) [7] is used further. Input data contains a set V of vehicles, a set C of customers and data about their geographical locations in the form of a directed graph G. The graph consists of $|C| + 2$ vertices, whereby the customers are denoted as $1, 2, \ldots, n$ and the depot (in this case DC) is represented by vertices 0 and $n + 1$. A set of vertices of G is denoted as N, while a set A of arcs represents connections between customers and between the depot and customers. For each arc (i, j), where $i \neq j$, a distance c_{ij} and a travel time t_{ij} are defined. Each customer i has demand d_i and should be served by one vehicle k with capacity q_k and only once within a planning horizon. For each customer, time window $[a_i, b_i]$ when it has to be served is defined. Vehicles routes start and end in DC.

The VRPTW model contains two sets of *decision variables*, namely: x and s, which are defined as flow and time variables, correspondingly. Variable x_{ijk} for each arc (i, j), where $i \neq j$, $i \neq n+1$, $j \neq 0$, and each vehicle k is defined as follows: $x_{ijk} = 1$ if vehicle k drives from vertex i to vertex j, and $x_{ijk} = 0$, otherwise. The decision variable s_{ik} for each vertex i and each vehicle k denotes the time, when vehicle k starts to service customer i.

Shortest routes for a fleet of homogenous vehicles with a limited capacity have to be found, i.e.:

$$\sum_{k \in V} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ijk} \rightarrow \min. \tag{14}$$

The main problem constraints are defined as follows [7]:

1. Each customer is visited only once;
2. No vehicle is overloaded:

$$\sum_{k \in V} d_i \sum_{j \in N} x_{ijk} \leq q; \ \forall k \in V; \tag{15}$$

3. Each vehicle leaves depot 0, leaves a customer after its serving and finally arrives at the depot $n+1$;
4. Vehicle k cannot arrive at customer j before time $s_{ik} + t_{ij}$ if it is travelling from customer i to customer j; and
5. Time windows are defined by:

$$a_i \leq s_{ik} \leq b_i; \forall i \in N, \forall k \in V. \tag{16}$$

6.2 Optimisation Algorithm

For vehicle routing, an *island genetic algorithm with offspring selection* (IOSGA) described in [31] is used. It presents a coarse-grained parallel genetic algorithm where population is divided into several islands in which GA works independently. Periodically, after a certain number of generations best solutions migrate between islands. The IOSGA is enhanced with an *offspring selection* to prevent a premature GA convergence. Offspring selection forces the algorithm to produce offspring solutions with better fitness than their parents [1].

For the considered problem instances, operators and parameters of the IOSGA are determined experimentally as follows: a proportional selector; 5 islands; 200 individuals in population; ring migration each 20 generations with 15 % rate: random individuals are replaced with the best ones from the neighbour island. The maximal selection pressure was set equal to 200 and the mutation rate equal to 5 %. Mutation operators provided in HeuristicLab framework were involved, and a GVR crossover [25] is selected in an experimental analysis below.

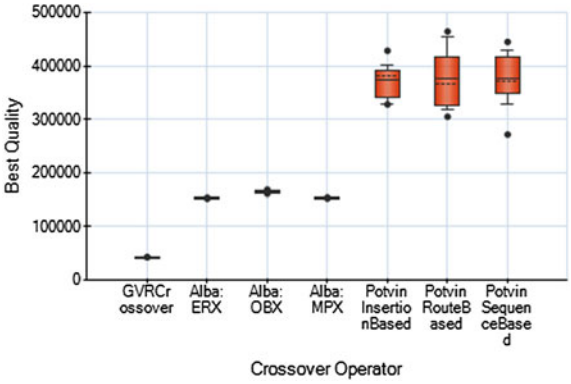


Fig. 17 Performance of crossover operators in VRP

It is worth mentioning that GA is not considered as the strongest optimisation method for the VRP [4, 12], and more often the Tabu Search (TS) algorithm with constraint relaxation [8] is recommended as more efficient approach. However, genetic algorithms show good performance for routing problems and are highly robust and adjustable. Also, for the considered example instance, the IOSGA shows better results than TS as described below.

6.3 Route Optimisation Experiments

After an optimisation algorithm is selected, its components such as the crossover operator need to be justified. To select a crossover operator for the IOSGA a set of *route optimisation experiments* are performed (see Fig. 17). Here, the GVR crossover [25], edge recombination (ERX) and maximal preservative (MPX) crossovers for solutions using Alba encoding [2] are analysed. As the VRP is minimisation problem, better operators have lower best found quality values in Fig. 17. ERX application provided better results in terms of the total length of vehicles' routes, while preserving an available number of vehicles. However, the results obtained for Alba encoded solutions lost in terms of capacity constraints. In turn, application of GVR crossover provided solutions with an overflow of available vehicles, nevertheless the capacity constraints were satisfied in most cases. Finally, the GVR crossover operator which works with an unlimited number of vehicles, but provided the best feasible results in terms of keeping routes not overloaded was selected.

To determine, if selected optimisation algorithm has the best performance for the reviewed problem, additional route optimisations experiments are performed with the tabu search algorithm [8]. In these experiments TS was set up with following parameters: for the move generation and evaluation, the Potvin Shift Exhaustive Move operator; and the tabu tenure equal to 15 for problems with 109 customers and 13 for problems with 53 and 56 customers. The initial solution in each run was created using Push Forward Insertion heuristics with $\alpha = 0.7$, $\beta = 0.1$ and $\gamma = 0.2$. The termination criterion was defined by 1000 generations.

Table 5 Routing results obtained with different algorithms

Problem	Stores	IOSGA		Tabu search	
		Distance	Vehicles	Distance	Vehicles
1	109	6331	44	6396	43
2	53	2561	18	2599	18
3	56	4346	26	4410	25

Results of IOSGA and TS optimisation experiments for the same problem instances are compared in Table 5. For each problem instance, the number of stores is defined. For each algorithm and problem instance, the shortest distance in km and a minimum required number of vehicles received from 10 optimisation experiments are given. A sample VRP instance of problem 2 solved by the IOSGA can be seen in Fig. 20.

The IOSGA provides better results in terms of distances and found solutions with less number of evaluations. The Tabu Search outperformed in minimisation of a number of required vehicles, which is supposed to be reduced in the next step of integrated methodology. Thus, the Tabu Search does not provide the significant improvement of the routing solution in the considered case.

7 Vehicle Scheduling for Routed Solution

The vehicle scheduling for routed solution is discussed. The approach is similar to one described in Sect. 5, but the difference is in input data and that scheduling in this case also aims at minimising a number of required vehicles for the same number of routes. The section presents the problem statement for routed solution is given and an optimisation algorithm to solve the problem. Then sequential application of vehicle routing and route scheduling algorithms is described.

7.1 Problem Statement

Obtained in Sect. 6 the vehicle routing solution provides minimal costs on vehicle driving distances. At the same time this solution can be improved further with a proper scheduling of routes between available vehicles to minimise also a number of required vehicles and related costs. In the classical VRPTW statement vehicle may perform only one route in the planning horizon. In practice, all routes often are shortened due to a limited capacity of vehicles. This may lead to ineffective solutions when a vehicle performs only one short route of a few hours long, while most of the day it may be idle. Thus, the *vehicle scheduling problem* is formulated *for the routed solution*. Here, routes are assumed to be independent from vehicles, while vehicles may perform a fair number of routes during the day. To solve the problem, methods developed in [19] are considered.

As far as the VRPTW solution is feasible for capacity and time window constraints, it can be further improved by combining and compacting routes. As a result, each vehicle can perform a sequence of predefined routes during the day and as a result, utilisation of vehicles may be significantly increased. A vehicle capacity is not involved in the statement as in the capacitated VRP all vehicles have the same capacity and no route of a feasible solution may exceed this value. Finally, *input data* is defined as follows: (1) ready time a_i for customer i; (2) due time b_i for customer i; (3) service time z_i for customer i; (4) a list of routes R obtained in VRP solution, where each route defines a sequence of visited customers; (5) a set of transportation times (or vehicle moving times t_{ij} between route sequential points i and j; (6) an estimated number of vehicles $|V|$.

Decision variables are similar to ones introduced in the VRPTW model, except that $x_{ijk} = 1$ means that route j is the next route after i for vehicle k. Two soft constraints are introduced for each vehicle:

1. a number of times N_{ad} when a vehicle leaves a customer after due time:

$$N_{ad} = |\{i | s_{ik} + z_i > b_i; \forall i \in N; \forall k \in V\}|; \tag{17}$$

2. a number of times when vehicle busy time may exceed 24 h in a day:

$$N_{ot} = \left|\left\{k | (s_{(n+1)k} - s_{0k}) > 24 \cdot 60; \forall k \in V\right\}\right|. \tag{18}$$

Additional constraints are introduced to assure the model integrity and provisions of schedule simulations. Finally, a fitness function f defines a sum of vehicles idle times due to fitting deliveries to the time windows and a number of constraint violations multiplied by penalty values:

$$f = \sum_{k \in V} l_k + p_{ad} N_{ad} + p_{ot} N_{ot} \rightarrow \min, \tag{19}$$

$$l_k = \sum_{i \in R} \sum_{j \in R} \max\left(a_i - s_{ik}, 0\right) x_{ijk}; \forall k \in V, \tag{20}$$

where l_k is the total idle time of vehicle k; V defines a set of available vehicles; p_{ad} and p_{ot} are penalty values or coefficients for late deliveries and vehicle overtimes, correspondingly, and p_{ad}, p_{ot} are assumed to be significantly greater than 1.

7.2 Optimisation Algorithm

For vehicle scheduling, a problem plug-in in HeuristicLab optimisation framework is implemented by maintaining the logic of the above described vehicle schedule simulation model. Here, a fitness evaluator simulates a vehicle schedule candidate

6	4	10	3	8	7	12	1	11	5	2	9

Fig. 18 A sample chromosome of permutation encoding for vehicle scheduling

and identifies time windows mismatches as well as evaluates vehicle idle and busy times. As a result, plug-in is used to calculate a fitness of a candidate solution by formula (19).

The chromosome which represents the schedule candidate is encoded as a permutation of integer numbers. Integers larger than the number of routes encode gaps in the chromosome when a new sequence of vehicle routes starts. Other integers define corresponding routes in sequences. The encoding used is similar to that described in Sect. 5.4. A sample chromosome for an instance with a number of vehicles $n = 3$ and a number of routes $m = 9$ is shown in Fig. 18. Here, the first vehicle marked as '10' performs the sequence of routes: 3, 8 and 7; the second one the sequence of five routes: 5-2-9-6-4 and the third vehicle will perform only one route, marked as '1'.

The application of permutation-based encoding allows easy use of different recombination and mutation operators. For schedule optimisation, an Evolution Strategies (ES) algorithm implemented in HeuristicLab [32] is selected as it has shown high efficiency in vehicle schedule optimisation [6].

7.3 Schedule Optimisation Experiments

Various series of *schedule optimisation experiments* are performed to compare scheduling results obtained by different metaheuristic optimisation algorithms. The following algorithms are examined: ES (λ, μ) and $(\lambda + \mu)$ algorithms, GA, island genetic algorithm with 5 islands (IGA), and offspring selection genetic algorithm (OSGA) [1]. Maximal preservative crossover and insertion manipulator are applied in all algorithms. Numbers of solution evaluations performed to obtain candidates with equal fitness values are compared on hard instances, with a low number of vehicles and short time windows. The results of optimisation experiments for a single instance are shown in Fig. 19. As considered problem is a minimisation problem, best solutions have lower fitness and are in the bottom of the chart.

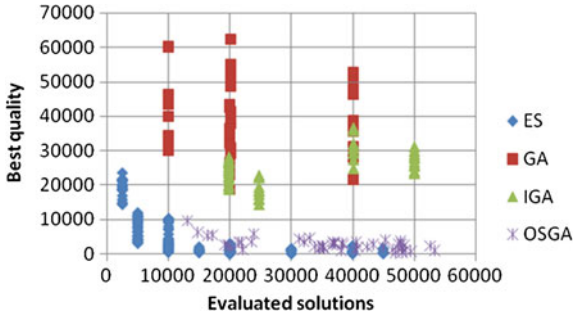


Fig. 19 Productivity of optimisation algorithms for VSPTW

The experimental results show that the same instance is solved with ES and OSGA algorithms in shorter time, while GA without modifications demonstrated the worst result. The ES algorithm is selected as the most suitable having an ability to search for global solutions with fewer evaluations.

7.4 Example

The following is an example of vehicle routes and schedule optimisation for a daily plan and specific demand data defined for 53 stores. Time windows for most stores are fixed from 6:00 a.m. to 9:00 p.m. But some stores can be served in any time. Application of the IOSGA in the VRP has given 34 routes (see Fig. 20) in the best solution. Due to a limited capacity of vehicles, most routes in the solution are very short, including one or two customers. As most of stores are located close to the DC (see inset in Fig. 20), due to small vehicle moving times, the trip length for many routes is relatively short. But, in case of large time windows and a long planning horizon, these routes can be combined.

Finally, evolution strategies (20 + 100) are applied for the routed solution. In schedule optimisation experiments, the maximal preservative crossover and insertion mutation operators were applied, and the termination condition was defined by 1000 generations. In experiments it is obtained that the problem instance that required 34 vehicles for deliveries (Fig. 20) has globally optimal solutions with all constraints satisfied when a number of vehicles is equal to 6.

The corresponding schedule Gantt chart for a planning horizon of 24 h is demonstrated in Fig. 21. Here, green lines define loading times in DC and the beginning of routes from the VRP solution, while blue and yellow lines depict transportation times and unloading times at stores, correspondingly. For example, the 4th vehicle in

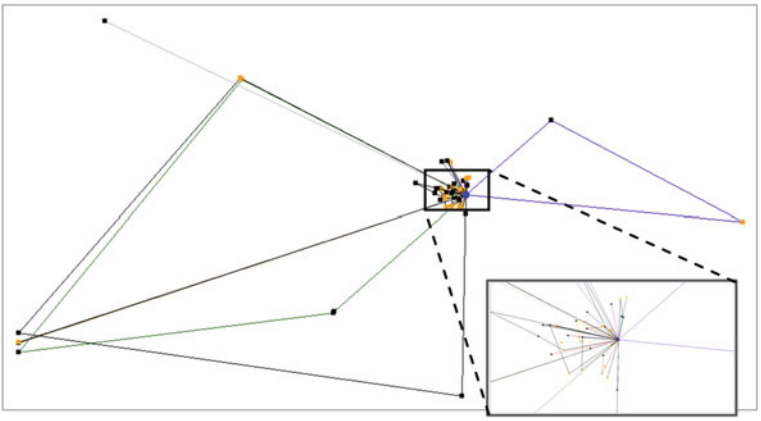


Fig. 20 VRP solution of case instance

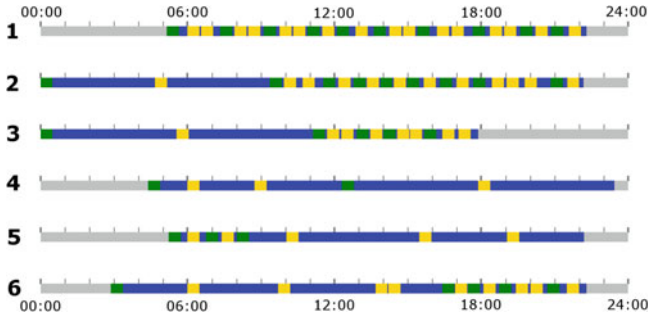


Fig. 21 VSP solution of case instance

Fig. 21 has to perform two very long trips that go to the stores in bottom left corner of Fig. 20. At same time the vehicle on first timeline is scheduled to perform 10 very short trips to the stores located near DC and having large time windows.

Similar experiments performed for multiple problem instances of the business case shows that vehicle scheduling applied for the routing solution allows reducing the number of vehicles required for daily deliveries.

To assess the quality of this approach the benchmark instances [28] are used. First, vehicle routes are found using the IOSGA, and then vehicles are scheduled with the ES. The experimental results show that these benchmark problems are designed to fit each route for each vehicle, so that subsequent scheduling does not provide any significant enhancement. When capacities of vehicles in benchmark instances are decreased twice to shorten potential vehicle routes and making them similar to those specified in the business case, the results of experiments on the modified benchmark problems show the benefit of the proposed approach for a number of instances. For example instance C102 requires only 16 vehicles instead of 21 to fulfil the product delivery plan from DC to a network of stores.

8 Conclusions

Modern delivery planning in large distribution networks with various constraining factors requires application of a number of methods to minimise delivery costs and cope with stochastic demand. Methods described in this chapter, such as cluster analysis, simulation, optimisation and fitness landscape analysis—are combined together into an integrated methodology to increase their application efficiencies and to reduce the computational requirements. Most of these methods are heuristic and metaheuristic and thus do not ensure obtaining globally optimal solutions, nonetheless they provide very good solutions, which are enough in most business cases, in less computational time comparing with traditional optimisation techniques.

The proposed integrated approach to product delivery tactical planning and scheduling allows identifying typical dynamic demand patterns and corresponding

product delivery tactical plans as well as finding the optimal parameters of product delivery schedules. Application of cluster analysis and classification algorithms to historic dynamic demand data allows identifying typical weekly demand patterns and developing base tactical delivery plans for groups of weeks with similar demand dynamics. Both k-means and NBTree methods are simple in implementation and implemented in a number of existing data mining tools. Store grouping allows determination of groups of stores, which have nearby location, considering the total demand of group. Application of multi-objective optimisation algorithm allows getting a set of non-dominating solutions and provides an opportunity to choose solution balancing between geographical and demand objectives. Both weekly demand pattern recognition and store grouping allow developing the base tactical delivery plan.

Two types of schedule optimisation solutions are provided. The first solution is designed, which has predefined vehicle routes and is simulation-based that allows dealing with stochastic factors of the deliveries, e.g. stochastic moving times. Fitness landscape analysis methods are presented and shortly described for tuning and adjustment of an optimisation algorithm. These methods if necessary could be also used for the second solution which allows optimising both vehicle routes and schedules when routes are not predefined. Application of advanced genetic algorithms or Tabu search methods allows obtaining in a fairly short time vehicle routes, which minimise transportation costs under constraints such as delivery time windows. Supplementing vehicle scheduling methods allows improving this routed solution by minimising a number of required vehicles and related costs. Both routing and scheduling are performed in same open source optimisation framework. Experimental results show that proposed two-stage routing and scheduling methodology provides good solutions in cases when vehicle routes are constrained with a small capacity. Joint sequential application of all presented methods allows obtaining cost-effective solutions in large store network delivery planning. In both solutions the parameters of the optimisation algorithms should be tuned for a specific case.

The proposed integrated approach to product delivery tactical planning and scheduling allows reducing the effect of product demand variation on the delivery planning process and avoids numerous time-consuming adjustments of the delivery tactical plans. Also, identifying demand pattern and an appropriate delivery plan ensure more qualitative solutions of the schedule optimisation task and cut down its computational costs.

References

1. Affenzeller, M., Winkler, S., Wagner, S., Beham, A. (2009) Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications. Chapman & Hall/CRC.
2. Alba, E. and Dorronsoro, B. (2004) ‘Solving the vehicle routing problem by using cellular genetic algorithms’ in Proceedings of EvoCOP 2004, April 5–7, Coimbra, Portugal, pp. 11–20.

3. AnyLogic (2014). Multimethod Simulation Software and Solutions. [ONLINE] Available at: http://www.anylogic.com/. (Accessed 01 August 2014).
4. Baker, B.M. and Ayechew, M. A. (2003) 'A genetic algorithm for the vehicle routing problem' in Computers and Operations Research, Vol. 30, pp. 787–800.
5. Bolshakov, V. (2013) Simulation-based Fitness Landscape Analysis and Optimisation of Complex Systems. PhD Thesis, Riga Technical University, Latvia.
6. Bolshakov, V., Pitzer, E., and Affenzeller, M. (2011) 'Fitness Landscape Analysis of Simulation Optimisation Problems with HeuristicLab' in Proc. of 2011 UKSim 5th European Symposium on Computer Modeling and Simulation, November 16–18, 2011, Madrid, Spain, pp. 107–112.
7. Cordeau, F., Desaulniers, G., Desrosiers, J., Solomon, M.M. and Soumis, F. (2001a) 'VRP with Time Windows' in Toth P. and Vigo D. (Eds.), The vehicle routing problem, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 157–193.
8. Cordeau, J.F., Laporte, G., Mercier, A. (2001b) 'A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows' in The Journal of the Operational Research Society, Vol. 52, No. 8, pp. 928–936.
9. Davies, D.L., Bouldin, D.W. (1979) 'A cluster separation measure' in IEEE Transactions on Pattern Analysis and Machine Intelligence, 1(2), pp. 224–227.
10. Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002) 'A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II', IEEE Transactions on Evolutionary Computation, 6(2), pp. 182–197.
11. Eliiyi D.T., Ornek A., Karakutuk S.S. (2008) 'A vehicle scheduling problem with fixed trips and time limitations' in International Journal of Production Economics, Vol. 117, No.1., pp. 150–161.
12. Gendreau, M., Laporte, G., and Potvin, J.Y. (2001) 'Metaheuristics for the capacitated VRP' in Toth P. and Vigo D. (Eds.), The vehicle routing problem, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 129–154.
13. Gwiazda, T.D. (2006) Genetic algorithms reference Vol. I. Crossover for single-objective numerical optimization problems [online]. TOMASZGWIAZDA E-BOOKS. http://www.tomaszgwiazda.com/ebook_a.htm. (Accessed 10 January 2013).
14. Kaufman, L. and Rousseeuw, P. J. (1990) Finding Groups in Data: An Introduction to Cluster Analysis. Hoboken, NJ: John Wiley & Sons, Inc.
15. MacQueen, J. B. (1967) 'Some Methods for Classification and Analysis of MultiVariate Observations' in Proc. of the 5th Berkeley Symposium on Math. Statistics and Probability, Vol. 1, pp. 281–297.
16. Michalewicz, Z. (1999) Genetic Algorithms + Data Structures = Evolution Programs. Third, Revised and Extended Edition. Springer-Verlag Berlin Heidelberg.
17. Merkuryeva, G. (2012) 'Integrated Delivery Planning and Scheduling Built on Cluster Analysis and Simulation Optimisation' in Proceedings of the 26th European Conference on Modelling and Simulation, pp. 164–168.
18. Merkuryeva, G. and Bolshakov, V. (2012a) 'Simulation-based Fitness Landscape Analysis and Optimisation for Vehicle Scheduling Problem' in Moreno D.R., et al., (Eds.), Computer Aided Systems Theory – EUROCAST 2011, Part I, LNCS 6927, pp. 280–286.
19. Merkuryeva, G. and Bolshakov, V. (2012b) 'Simulation optimisation and monitoring in tactical and operational planning of deliveries' in Proceedings of the 24th European Modeling and Simulation Symposium, EMSS 2012, pp. 226–231.
20. Merkuryeva, G., Bolshakov, V. and Kornevs, M. (2011) 'An Integrated Approach to Product Delivery Planning and Scheduling', Scientific Journal of Riga Technical University, Computer Science, Information Technology and Management Science Ser. 5, Vol. 49, pp. 97–103.
21. Merkuryeva G. and Bolshakovs V. (2010a) 'Simulation-based vehicle scheduling with time windows' in Proc. of First International Conference on Computer Modelling and Simulation, pp. 134–139.
22. Merkuryeva, G. and Bolshakovs, V. (2010b) 'Vehicle schedule simulation with AnyLogic' in Proc. of 12th Intl. Conf. on Computer Modelling and Simulation, pp. 169–174.

23. Merkuryeva, G., Merkuryev, Y. and Bolshakov, V. (2010) 'Simulation-based fitness landscape analysis for vehicle scheduling problem' in Proc. of the 7th EUROSIM Congress on Modelling and Simulation, September 6–10, 2010, Prague, Czech Republic.
24. Nagamochi, H., Ohinishi, T. (2008) 'Approximating a vehicle scheduling problem with time windows and handling times' in Theoretical Computer Science, Vol. 393, Is. 1–3, pp. 133–146.
25. Pereira, F.B., Tavares, J., Machado, P. and Costa E. (2002) 'GVR: A New Genetic Representation for the Vehicle Routing Problem' in Proceedings of the 13th Irish International Conference on Artificial Intelligence and Cognitive Science (AICS '02), September 12–13, Limerick, Ireland, pp. 95–102.
26. Pitzer, E., Affenzeller, M. (2012) 'A Comprehensive Survey on Fitness Landscape Analysis' in Recent Advances in Intelligent Engineering Systems: Springer, pp. 161–191.
27. Seber, G.A.F. (1984) Multivariate Observations, John Wiley & Sons, Inc., Hoboken, NJ.
28. Solomon, M.M. (2005) VRPTW Benchmark Problems. [online] http://w.cba.neu.edu/~msolomon/problems.htm (Accessed 26 September 2013).
29. Stadler, P.F. (2002) 'Fitness Landscapes' in Lassig M. and Valleriani A. (Eds.), Biological Evolution and Statistical Physics, Springer, pp. 183–204.
30. Vassilev, V.K., Fogarty, T.C., Miller, J.F. (2000). 'Information Characteristics and the Structure of Landscapes' in Evolutionary Computation, 8(1), pp. 31–60.
31. Vonolfen, S., Affenzeller, M., Beham, A., Wagner, S. (2011) 'Solving large-scale vehicle routing problem instances using an island-model offspring selection genetic algorithm', in Proc. of 3rd IEEE International Symposium on Logistics and Industrial Informatics (LINDI), 2011, August 25–27, 2011, Budapest, Hungary, pp. 27–31.
32. Wagner, S. (2009) Heuristic Optimization Software Systems - Modeling of Heuristic Optimization Algorithms in the HeuristicLab Software Environment. PhD Thesis, Institute for Formal Models and Verification, Johannes Kepler University Linz, Austria.
33. Weinberger, E. (1990) 'Correlated and Uncorrelated Fitness Landscapes and How to Tell the Difference' in Biological Cybernetics, 63(5), pp. 325–336.

Large Neighbourhood Search and Simulation for Disruption Management in the Airline Industry

Daniel Guimarans, Pol Arias and Miguel Mujica Mota

Abstract The airline industry is one of the most affected by operational disruptions, defined as deviations from originally planned operations. Due to airlines network configuration, delays are rapidly propagated to connecting flights, substantially increasing unexpected costs for the airlines. The goal in these situations is therefore to minimise the impact of the disruption, reducing delays and the number of affected flights, crews and passengers. In this chapter, we describe a methodology that tackles the Stochastic Aircraft Recovery Problem, which considers the stochastic nature of air transportation systems. We define an optimisation approach based on the Large Neighbourhood Search metaheuristic, combined with simulation at different stages in order to ensure solutions' robustness. We test our approach on a set of instances with different characteristics, including some instances originating from real data provided by a Spanish airline. In all cases, our approach performs better than a deterministic approach when system's variability is considered.

1 Introduction

Operational disruptions are alterations of originally planned operations due to external events. The airline industry is notably one of the most affected industries regarding operational disruptions. The costs associated to them have gained more and more importance with the increase of fuel costs and the punctuality policies that

D. Guimarans (✉)
Optimisation Research Group, National ICT Australia (NICTA),
Sydney, Australia
e-mail: daniel.guimarans@nicta.com.au

P. Arias
Smart Logistics and Production Group, Internet Interdisciplinary Institute (IN3-UOC),
Barcelona, Spain
e-mail: pol.arias5@gmail.com

M. Mujica Mota
Aviation Academy, Amsterdam University of Applied Sciences,
Amsterdam, Netherlands
e-mail: m.mujica.mota@hva.nl

M. Mujica Mota et al. (eds.), *Applied Simulation and Optimization*,
DOI 10.1007/978-3-319-15033-8_6

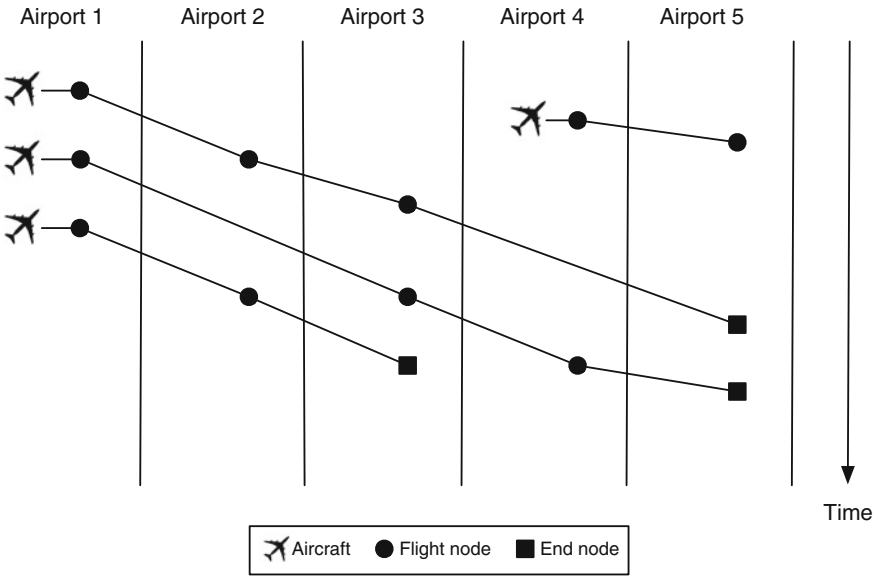


Fig. 1 Flight legs are, in general, a component of different flight schedules. Hence, a perturbation in one flight leg may propagate to other elements in the network, affecting aircraft, crew and passenger connections

airlines have been forced to implement in order to maintain competitiveness [61]. Due to these and other emerging regulations that the aeronautical industry is facing nowadays [21], the optimisation of resources has become an important issue in the aeronautical agenda.

Flight plans are usually made several months prior to the actual day of operation. As a consequence, changes often occur in the period from the development of the schedule to the day of operation. Those changes may include unforeseen delays due to weather phenomena, air traffic control delays, reactionary delays, ground operations, etc. The fact that a single flight leg is, in general, a component of different flight schedules implies that a single perturbation may propagate to other elements in the network (Fig. 1). Thus, a delay usually leads to reactionary delays on several other flights. This effect is in many cases exacerbated by airlines network characteristics, which for most of the largest carriers relies on a hub-and-spoke configuration. Hence, a perturbation on the schedule in the hub may easily lead to many parts of the network being affected by the disruption.

According to the Association of European Airlines [1], 22.4 % of departures of intra-European flights were delayed more than 15 min in the first quarter of 2008, an increase from 20.5 % in 2007. This figure is consistent with the observed average delay (22.74 %) during the ten-year period 1998–2007. Breaking down the figures, the same study found that 7.3 % of delayed departures were due to pre-flight preparation, i.e. the aircraft is not ready to leave on time because of late loading, crew availability,

completion of paperwork, etc. Besides, 13.1 % of delays were attributable to air traffic management (ATM) or airport management-related issues. This figure increased from 11.5 % in 2007 to 13.1 % in 2008, suggesting that the European airspace is getting more congested. Considering 27 major European airports, we observe an average of 23.1 % of European departures delayed 15 min or more, with an average delay of 41.8 min. Arrival figures are similar, with an average of 25 % of flights delayed 15 min or more, with an average delay of 41.3 min. Among major airports, London Heathrow is clearly the most affected, with 44.1 % of European departures subject to delays of 15 min or more. Other studies [10, 47] suggest that USA figures are similar.

Estimating the cost airlines incur due to operational disruptions is difficult because of the many factors involved and some unquantifiable effects, e.g. passenger inconvenience. Shavell [47] reported that, for the year 1998, the total estimated costs of irregular operations incurred by the ten USA airlines that report performance data to the Department of Transportation was $1.826 billion. Analysing this figure, $858M were attributable to cancelled flights, $909M to delays and $59M to diversions, i.e. a flight that is directed to a different airport than the originally scheduled. From a case study of a heavy snowstorm in Boston, Shavell [47] concludes that the estimated cost per minute of delay over 15 min is $13.35. In a more recent study [20], EUROCONTROL's Central Office for Delay Analysis (CODA) estimates the cost per minute of delay at €82 for delays in excess of 15 min. According to their statistics and extrapolating the percentage of delayed flights to all flights in Europe, CODA estimated the total cost of delays from all causes at more than €7 billion in 2008.

One of the major problems airlines face on a daily basis is that specific flight legs are not entirely predictable. It is normal that flights are subject to a certain level of variability in daily operations, due to the stochastic nature of air transport and the number of stakeholders involved. Moreover, due to network configurations, a late arrival of an aircraft often causes delays in the next departure of this aircraft and connecting flights. Traditionally, airlines cope with this problem in their schedules by introducing additional *buffer* time in between flights, aiming to absorb potential delays in case they occur. However, buffer time increases operations cost significantly. EUROCONTROL estimates the cost of one minute of buffer time for an Airbus A320 at €49 per flight [19]. Hence, in the last years, airlines and researchers are addressing more efforts towards solving operational disruptions more efficiently (see Sect. 2). This way, airlines may configure tighter schedules and reduce operational costs, while still being able to respond to unforeseen events and disruptions. These problems make evident the need of decision support tools that help decision makers to cope with operative problems under urgent circumstances.

Among the different elements involved in a disrupted scenario (aircraft, crews and passengers), aircraft have received most attention from the research community (see Sect. 2), since it is normally considered the scarce resource and problem dimensions are relatively small. In general, the main objective is to restore the flight schedule as much as possible using the existing aircraft, i.e. minimise the number of cancellations and the total delay, in order to minimise the impact of the disruption. Given an original flight schedule and one or more disruptions (i.e. flight delays), the optimisation

approach generates a solution by means of delaying flights, swapping aircraft-to-flight allocation, or even cancelling flights. Such plan considers all flights scheduled within a certain period of time by a given fleet, including the original departure, expected flight durations and connections between airports. This challenging problem is known as the Aircraft Recovery Problem (ARP) and regarded to be NP-Hard [9]. Introducing variability in the values associated to the problem, i.e. flights duration, turnaround times or delays, the Stochastic Aircraft Recovery Problem (SARP) arises. The SARP accounts for all characteristics inherited from the ARP, adding additional considerations present in realistic problems.

In this work, we propose an optimisation approach for the SARP that integrates simulation at different stages. On the one hand, optimisation-based methods have proved their efficiency to deal with operative problems in complex fields, e.g. logistics or manufacturing problems. However, many optimisation approaches lack the flexibility needed for tackling real operational problems. In most cases, the stochasticity inherent to these systems is not included in the developed models, thus reducing their applicability to real scenarios. On the other hand, simulation approaches have great flexibility and allow the modeller to face the problem under a different scope, including not only stochastic elements but also the dynamics of interest of the system. Nevertheless, in most simulation-based approaches the level of optimisation achieved depends on the number of scenarios evaluated, which in general is just a small fraction of the whole available configurations. Hence, optimality may not be guaranteed with the standalone simulation approach, likewise suitability is not ensured with the standalone optimisation approach. In the present approach, we combine both methodologies in order to obtain pseudo-optimal solutions which are robust enough to cope with system's stochasticity.

We present a methodology based on the *Large Neighbourhood Search* (LNS) metaheuristic (see Sect. 4.1). In our approach, we developed a *Constraint Programming* (CP) formulation to tackle the deterministic ARP (see Sect. 3.2). This model was combined with simulation and tested in previous works [7, 8]. In this case, we use the same model as a *repair* method in the LNS approach. It should be noticed that we do not consider cancellations in our approach. In these situations, connecting flights are generally cancelled so the aircraft may be assigned to a later flight from the same airport. In other cases, aircraft may be *ferried*, i.e. flying without passengers, to the destination airport in order to restore the original schedule. In both situations, we can represent a cancelled flight or aircraft shortage, e.g. due to temporary mechanical problems, as a long delay at flight's origin. We include simulation at different stages of the optimisation process, defining the SimLNS methodology (see Sect. 4). With this approach, we are able to increase solutions' robustness, as solutions are only accepted if they perform better than the incumbent in a variety of simulated scenarios. The proposed methodology has been assessed on a set of instances with different characteristics, some of them obtained from real data provided by a Spanish airline (see Sect. 5).

2 Literature Review

Traditionally, disruption management research is divided into aircraft recovery, crew recovery and passenger recovery. In this section we review previous academic research on aircraft recovery. For a more thorough review of academic research on different aspects of disruption management, including aircraft recovery, we refer the reader to the comprehensive reviews by Kohl et al. [32], Clausen et al. [15] and Le et al. [35].

The ARP has received much attention among operational recovery problems as aircrafts are considered the scarce resource. In addition, rules applied for aircraft reallocation are often less complex than those governing crew or passenger recovery. However, in most cases crew recovery permits more flexibility by using standby crews at airline's bases. In the case of passenger recovery, it is not uncommon to reallocate heavily disrupted passengers in flights operated by other airlines. This solving flexibility is more difficult to achieve in aircraft recovery, since the number of spare aircraft is normally quite limited or inexistent. Furthermore, research on aircraft recovery to this date only deals with a single airline and does not support cooperation between different carriers.

Teodorović and Guberinić [49] are among the pioneers of the ARP. In their work, given one or more unavailable aircraft, their objective is to minimise the total passenger delay by allowing flight retiming and aircraft swaps. The model is based on a network flow with side constraints, which is solved using a branch and bound method [34]. In a later work, Teodorović and Stojković [50] consider aircraft shortage and propose an improved approach. The authors solve the problem by a heuristic algorithm based on dynamic programming using an algorithm based on a lexicographic ordering of the flights. The constructed model allows cancellations, retiming and swaps. The main objective is to minimise the number of cancellations. If there are several solutions with the same number of cancelled flights, they use the total passenger delay as secondary objective.

The literature contains several works on different aspects of the ARP. Many of them are based on a multi-commodity flow problem solved on a time-band network. Jarrah et al. [29] develop two network flow models to cope with aircraft shortage. Both models—a delay model and a cancellation model—permit using standby aircraft. In both cases, the goal is to minimise the cost of the recovery, including not only cancellation and delay costs, but those associated to swapping or ferrying aircraft. The main drawback is that models are dissociated: one model handles retiming only, while the other handles cancellations only. Their work was deployed at United Airlines as part of their decision support system. Cao and Kanafani [13, 14] extended the delay model of Jarrah et al. [29] to include cancellations and multiple airports. However, Løve and Sørensen [36] proved that these models have serious deficiencies.

In a later work, Løve et al. [37] presented a heuristic for the ARP based on local search. The schedule is represented by the lines of work for each available aircraft. In order to solve the model, consider the cancellations, delays and aircraft reallocation, both within a single fleet or between fleets. The objective is to minimise the recovery costs related to delays, cancellations and swaps. It is even possible to assign costs on

individual flights to weigh the relative importance of different flights. The proposed approach was tested on the short-haul operation of British Airways (79 aircraft, 44 airports and 339 flights).

Granberg and Värbrand [24] proposed a mixed integer multi-commodity flow formulation with side constraints for the ARP, although they name the problem as *Flight Perturbation Problem*. They further reformulate the problem into a set packing model using the Dantzig–Wolfe decomposition [16]. Cancellations, delays and aircraft swaps are allowed in order to solve the perturbation. The authors propose two-column generation strategies and test them with data from a Swedish domestic airline. Results show that the methods are capable of obtaining high-quality solution, but running times increase drastically with instance size, e.g. 1139 s for one of the largest instances containing 215 flights.

Argüello et al. [5, 6] presented a method based on the Greedy Randomised Adaptive Search Procedure (GRASP) metaheuristic [42] to reschedule the aircraft routings if one or more aircrafts are unavailable. The heuristic is capable of cancelling and retiming flights. As in the approach of Løve et al. [37], it also allows swaps between different fleet types. The goal is to produce a recovery plan so the original schedule is restored by the following day. The cost to be minimised includes measures of passenger inconvenience and lost flight revenue. Their approach was tested in a fleet of 16 aircraft, 42 flights and 13 airports, reporting results within 10 % of optimality, according to the authors.

Yan and Yang [59] and Yan and Tu [58] developed four models to cope with temporary aircraft shortage. The models were specifically developed for small airlines. In the first model, it is possible to cancel flights to repair the disrupted schedule. In the second model, ferrying of spare aircraft is also considered together with flight cancellations. The third model considers cancelling and retiming flights. The last model incorporates all previous possibilities. In all models, swaps are allowed within a fleet. The objective in all models is to minimise the cost of repairing the schedule, including passenger revenue. The first two models are built as network flow models and can be solved to optimality very fast. The other two models contain side constraints and are solved using Lagrangian Relaxation and subgradient optimisation. In Yan and Tu [58], a multi-fleet version of these models is presented. In this case, a larger aircraft can be assigned to a flight that originally was serviced by a smaller aircraft. Yan and Young [60] also consider multiple aircraft types, but aircraft swaps between fleets are not allowed. The developed methods were tested on 534 different scenarios based on China Airlines data, solving all instances to optimality or at most 1 % from optimality within 5.5 min. Yan and Lin [57] extend these models to solve the special situation when an airport is temporarily closed. The model presented allows swapping flights, retimings and cancellations, but not diverting flights. Therefore, flights arriving to or departing from the closed airport have to be cancelled or delayed.

Thengvall et al. [51] extend the models presented in [58, 59] by penalising deviations from the original schedule in the objective function. They use Linear Programming (LP) relaxation to solve the proposed network flow model with side constraints. In case an integer solution is not reached with this approach, the authors provide a rounding heuristic that finds feasible solutions within a small fraction of the optimum.

Their approach is tested on data provided by Continental Airlines. In [52], the same authors extend their study by developing three multi-commodity network models for determining a recovery schedule following a hub closure.

Eggenberg et al. [17, 18] proposed a column generation scheme for the ARP with heterogeneous fleet, made of regular and reserve aircraft, and planned maintenance. The authors model the problem as a commodity flow problem on a dedicated network, one for each plane of the fleet. They report results on instances from Thomas Cook Airlines, ranging from 40 to 760 flights serviced with a fleet of 16 aircraft [18].

Wu and Le [54] also consider maintenance and regulations in their work. They base their model on flight strings, instead of individual flights, and transform it into a time-space network. The authors develop a heuristic, called *Iterative Tree Growing with Node Combination*, to solve this network model. Results are reported over a set of instances from China Airlines data consisting of 170 flights, 5 fleets, 35 aircraft and 51 airports.

Rosenberger et al. [44] were the first to use simulation when studying the ARP. The authors propose a problem decomposition, where the master problem is defined as a set partitioning problem (i.e. each flight is either cancelled or flown by an aircraft), and each subproblem is a route generation problem. The objective is to minimise the cost of cancellations and delays. In order to make the approach more computationally efficient, they define a heuristic to select only a subset of aircraft to be included in the set partitioning problem. The authors assess their results using the simulation environment *SimAir* [45]. They simulate 500 days of operations for 3 fleets, with datasets ranging from 32 to 96 aircraft and 139 to 407 flights. Nevertheless, Rosenberger et al. [44] do not use simulation in the optimisation phase in an integrated manner. Recovery procedures are invoked from *SimAir* any time a disruption is preventing the system to execute the flying schedule as planned. The disrupted scenario is then solved deterministically using data provided by the simulator, assessing the provided solution by resuming simulation with the recovery schedule.

In a more integrated approach, Wu [56] used simulation to calculate random ground operational delays and airborne delays in an airline network, instead of estimating delay propagation through the system. In a previous work [55], Wu showed that delays are inherent in airline operations due to stochastic delay causes. In [56], the author applies simulation to assess the robustness of airline schedules. This approach resembles the one proposed in our work. In a similar fashion, we use simulation to account for stochasticity in airborne and ground delays in order to obtain a more robust recovery plan (see Sect. 4).

3 Problem Formulation

The proposed formulation for the ARP is based on the *Constraint Programming* (CP) formalism. CP is a powerful paradigm for representing and solving combinatorial problems, whose nature provides easily adaptable problem representations. Moreover, constraints can be added or modified, even dynamically, without altering

search procedures. A brief introduction to CP is provided in Sect. 3.1, whereas the proposed ARP formulation is described in Sect. 3.2.

3.1 Constraint Programming

Constraint Programming (CP) is a powerful paradigm for representing and solving a wide range of combinatorial problems [46]. In the last few decades, it has attracted much attention among researchers due to its flexibility and its potential for solving hard combinatorial problems in areas such as scheduling, planning, timetabling and routing. CP combines strong theoretical foundations (e.g. techniques originated in different areas such as Mathematics, Artificial Intelligence, and Operations Research) with a wide range of applications in the areas of modelling heterogeneous optimisation and satisfaction problems. Moreover, CP nature provides other important advantages such as fast programme development, economic programme maintenance and efficient runtime performance.

Problems are expressed in terms of three entities: *variables*, their corresponding *domains* and *constraints* relating them. Constraints can be considered as the heart of CP. They are treated as logical relations among several *unknowns* (or *variables*), each taking a value from a set of accepted values called *domain*, which can be a range with lower and upper bounds or a discrete list of numbers. The representation of the problem, in terms of constraints, results in short and simple programmes easily adaptable to future changing requirements.

Since CP is the study of computational systems based on constraints, its idea is to solve problems by stating constraints (requirements) about the problem area and, consequently, finding a solution satisfying all the constraints. This class of problems is usually termed *Constraint Satisfaction Problems* (CSP) and the core mechanism used in solving them is *constraint propagation*. Constraint propagation embeds any reasoning which consists in explicitly forbidding values, or combinations of values, for some variables of a problem because a given subset of its constraints cannot be satisfied otherwise [12]. In other words, constraint propagation is a way to produce the consequences of a decision. In general, when a variable belonging to a constraint is labelled, that value is propagated to the rest of variables involved in that constraint.

An important contribution of CP is to allow the end user to control the search. The topic of search comes from the heart of Artificial Intelligence, which has developed several algorithms to perform the search in a solution space. End user's search control is achieved by combining generic techniques, when the generation of the whole search tree is unfeasible, and problem-specific techniques, when there is an extra knowledge about special features of the problem. Thus, while mathematical programming is mainly based in the application of certain algorithms to a model, CP allows the user to take some decisions on the search stage like the order of instantiation of the variables and the order of selection of values from domains. This point represents one of the most important differences with Linear Programming (LP): when using LP, once the problem is modelled, the rest of the work is done by the solver. In the CP methodology, the order of variable labelling and value selection is essential to

drive the search. However, it is important to notice that, although a search improved by these techniques can be useful to find a faster solution for a problem, it can significantly slow down the solution of a different problem. Depending on these choices, the way decisions are made is totally different and the performance of the search algorithm can be highly affected.

Solutions to a CSP can be found by searching (systematically) through the possible assignments of values to variables, i.e. generating the whole search tree. From the theoretical point of view, solving a CSP is trivial using systematic exploration of the solution space. But that is not true from the practical point of view, where efficiency takes place. Search methods can be divided into two broad classes: those that traverse the space of partial solutions (or partial value assignments), and those which explore the space of complete value assignments (to all variables) stochastically.

The simplest algorithm that searches the space of complete labellings, is called *Generate-and-Test* (GT) [33]. The idea of GT is very simple: firstly, a complete labelling of variables is randomly generated and, consequently, if this labelling satisfies all the constraints then the solution is already found; otherwise, another labelling is tried. Its search space corresponds to the Cartesian product of all variables' domains. The GT algorithm is clearly a weak generic algorithm with poor efficiency for two reasons: it has a non-informed generator and there is a late discovery of inconsistence.

Backtracking (BT) [3] is a method used for solving CSPs by incrementally extending a partial solution that specifies consistent values for some of the variables, towards a complete solution, by repeatedly choosing a value for another variable consistent with the values in the current partial solution. BT is a merge of the generating and testing phases of GT. The variables are labelled sequentially and as soon as all the variables relevant to a constraint are instantiated, the validity of the constraint is checked. If a partial solution violates any of the constraints, backtracking is performed to the most recently instantiated variable that still has available alternatives. Clearly, whenever a partial instantiation violates a constraint, BT is able to eliminate a subspace from the Cartesian product of all variables' domains. Hence, BT is strictly better than GT. However, its running complexity for most non-trivial problems is still exponential.

BT still has as a major drawback the late detection of conflicts. *Consistency techniques* [12] are used to detect inconsistencies in partial solutions sooner in the search, and they are at the core of constraint propagation. These techniques are based on the idea of removing inconsistent values from variables' domains until a solution is found. It is very important to note that consistency techniques are deterministic. There exist several consistency techniques, but most of them are not complete [3]. For this reason, these techniques are rarely used alone to solve a CSP completely and normally are combined with search algorithms such as BT. Attention should be paid to the use of these consistency techniques. They provide a good mechanism to remove inconsistent values from variables' domains during search, but they often penalise with respect to efficiency terms. For this reason, they are often reduced to the most basic forms, i.e. *node consistency* and *arc consistency* [22].

To cope with *Constraint Optimisation Problems* (COP), one should take into account the cost function. The appropriate modification of the BT search schema is called *Branch and Bound* (BB) [34]. During the search, BB maintains the current best value of the cost function (*bound*) and, each time a solution with a smaller cost is found, its value is updated. There are many variations on the BB algorithm. One consideration is what to do after a solution with a new best cost is found. The simplest approach is to restart the computation with the *bound* variable initialised to this new best cost. A less naive approach is to continue the search for better solutions without restarting. In this case, the cost function upper bound is constrained to the *bound* variable value. Each time a solution with a new best cost is found, this cost is dynamically imposed through this constraint. The constraint propagation triggered by this constraint leads to a pruning of the search tree by identifying the nodes under which no solution with a smaller cost can be present.

Generic techniques for local search, such as *Genetic Algorithms* (GA) [41], *Simulated Annealing* (SA) [38] or *Tabu Search* (TS) [23], can also be used to aid CP to find quasi-optimal solutions when it is not feasible to generate the whole search tree (due to memory or CPU time problems). These methods are used when the size of the problem is huge and it is not possible to find the optimal solution. Usually, CP is used to find fast poor solutions which will be used as initial values for these techniques. A good solution is sought from these input values. If the best solution found by these techniques is not good enough, a new initial solution is generated by CP. To avoid the same values than in previous searches, either new constraints are added or some of the existing constraints are removed. Alternatively, CP may be embedded at different stages of the local search, either to quickly check feasibility [28], reduce neighbourhood size by using consistency techniques [27], or to repair partial solutions in *Large Neighbourhood Search* approaches (see Sect. 4).

3.2 Aircraft Recovery Problem Formulation

The proposed CP formulation for the ARP intends to enhance the use of constraint propagation, modelling the problem with two sets of variables: predecessors (P) and successors (S). These variables allow us modelling the same search space from two different perspectives, while redundant constraints propagate decisions made in any of the two sets to the other one. Thus, search is carried out simultaneously in both variable sets, increasing overall efficiency and speeding up problem solving. This formulation is inspired on the Vehicle Routing Problem formulation by Kilby and Shaw [31].

We consider a set of n flights and a fleet of m aircraft. Then, the variables used in this formulation are:

- $\Psi = \psi_1 ... \psi_n$ are the flights to be attended;
- $A = a_1 ... a_m$ are the available aircraft;
- $G = g_1 ... g_{n+2m}$ is the assignment set, with domain G: : $[1..m]$.

It should be noticed that there is one assignation for each flight and two special assignations per aircraft: the starting and ending airports for the aircraft. Thus, two subsets of G, F and L, are defined as the aircraft departure and arrival airports to ensure the closure of the cycle:

- $F = n + 1...n + m$ is the set of first assignments;
- $L = n + m + 1...n + 2m$ is the set of last assignments.

Then, the predecessor and successor sets are defined as:

- $P = p_1...p_{n+m}$ is the predecessors set, with domain P: : $[1..n + m]$: : $(G - L)$;
- $S = s_1...s_{n+m}$ is the successors set, with domain S: : $[1..n, n + m + 1..n + 2m]$: : $(G - F)$.

A set of constraints is imposed to relate all the variables and define the problem. The predecessor and successor variables form a permutation of G and are therefore subject to the *difference constraints*.

$$p_i \neq p_j \quad \forall i, j \in G \wedge i < j \qquad s_i \neq s_j \quad \forall i, j \in G \wedge i < j \tag{1}$$

Equations (1) force predecessor and successor sets to contain no repetitions. Thus, one flight can have one and only one predecessor and successor. In practice, these constraints are implemented using the CP global constraint *alldifferent* [53] to enhance constraint propagation efficiency.

The successor variables are kept consistent with the predecessor variables via the following *coherence constraints*:

$$s_{p_i} = i \quad \forall i \in G - F \qquad p_{s_i} = i \quad \forall i \in G - L \tag{2}$$

Equations (2) connect the concepts successor and predecessor as follows: the former shows that i is the successor of its predecessor, and the latter indicates that i is the predecessor of its successor.

Along a set of connected flights, all assignations are performed by the same aircraft. This is maintained by the following *leg constraints*:

$$g_i = g_{p_i} \quad \forall i \in G - F \qquad g_i = g_{s_i} \quad \forall i \in G - L \tag{3}$$

Equations (3) are used to ensure that the aircraft assigned to i is the same as that assigned to its predecessor and successor.

Other sets of variables are defined to ensure the connections between origin and destination airports, as well as the times that aircraft are assigned to their flights:

- $O = o_1...o_n$ is the origin airport set;
- $D = d_1...d_n$ is the destination airport set;
- $\Delta = \delta_1...\delta_n$ is the flight duration list, including minimum turnaround times;
- $T = t_1...t_n$ is the departing times list, indicating the time when a flight departs;
- $\tau = \tau_1...\tau_n$ is the scheduled times list, indicating the time when a flight is originally scheduled to depart;

- $\Gamma = \gamma_1 ... \gamma_n$ is the list containing the initial delays that have disrupted the system;
- $\Lambda = \lambda_1 ... \lambda_n$ is the delays list, indicating the accumulated delays for each flight.

The actual departure time is calculated given the *departure time constraints*:

$$t_i \geq t_{p_i} + \delta_{p_i} \quad \forall i \in G - F \qquad t_i \leq t_{s_i} - \delta_i \quad \forall i \in G - L \tag{4}$$

Equations (4) bound the departure time of flight i. This time is, at least, the departure time plus duration time of its predecessor (δ_{p_i}). Equally, this time must be, at most, the departure time of its successor, minus the duration time of flight i (δ_i).

The connection between origin and destination airports is done by using the *connectivity constraints*:

$$o_i = d_{p_i} \quad \forall i \in G - F \qquad d_i = o_{s_i} \quad \forall i \in G - L \tag{5}$$

Equations (5) are used to narrow down the combinations of flights. The origin of flight i must be the destination of its predecessor. In the same way, the destination of flight i is the origin of its successor.

Equation (6) ensures that the departing time of flight i is greater than the scheduled time plus the initial delay.

$$t_i \geq \tau_i + \gamma_i \quad \forall i \in G - F - L \tag{6}$$

Equation (7) allows to calculate the total accumulated delay by obtaining the difference between the actual time of departure (t_i) and the scheduled time of departure (τ_i).

$$\lambda_i = t_i - \tau_i \quad \forall i \in G - F - L \tag{7}$$

Finally, the objective function (8) to be minimised is defined as the sum of accumulated delays for all flights.

$$\min \sum_{i=1}^{n} \lambda_i \tag{8}$$

4 SimLNS: Large Neighbourhood Search and Simulation

Aiming to solve the SARP, we propose an optimisation approach based on the *Large Neighbourhood Search* metaheuristic, which is described in Sect. 4.1. This local search method has proven to be specially successful when combined with CP. Therefore, we integrate our CP model for the ARP as part of the methodology, embedded in the so-called *operators* (see Sect. 4.2). In order to deal with the stochasticity present in the problem, we apply simulation to the obtained solutions in different phases of

our approach (see Sect. 4.3). This way, we improve final solutions' robustness by only accepting those solutions which, on average, perform better than previous ones. Thus, each solution is evaluated in a set of simulated scenarios according to system's variability before being accepted or rejected. Rather than just testing the final solution, as most traditional approaches, we moved the evaluation to previous stages of the search aiming to detect earlier in the process undesired solutions' characteristics, e.g. extremely sensitive solutions due to connection tightness.

4.1 Large Neighbourhood Search

In *Large Neighbourhood Search* (LNS), proposed by Shaw [48], an initial solution is gradually improved by alternately destroying and repairing the solution. Over the years, LNS has proved to be competitive with other local search techniques, especially when combined with CP. It complements the CP framework as LNS benefits from improved propagation while CP benefits from this efficient, while simple, search framework [39]. A complete introduction to the subject can be found in [40].

The LNS metaheuristic belongs to the class of heuristics known as *Very Large Scale Neighbourhood search* (VLSN) algorithms [2]. A neighbourhood search algorithm is considered as belonging to the class of VLSN algorithms if the neighbourhood it searches grows exponentially with the instance size or if the neighbourhood is simply too large to be searched explicitly in practice. Although the concept of VLSN was not formalised until recently, algorithms based on similar principles have been used for decades [2].

All VLSN algorithms are based on the observation that searching a large neighbourhood results in finding local optima of high quality, and hence a VLSN algorithm may return better solutions. However, searching a large neighbourhood is time-consuming, therefore various filtering techniques are used to limit the search. In VLSN algorithms, the neighbourhood is typically restricted to a subset of the solutions that can be searched efficiently.

Intuitively, searching a very large neighbourhood should lead to higher quality solutions than searching a small neighbourhood. Nevertheless, in practice, small neighbourhoods can provide similar or superior solution quality if embedded in a metaheuristic framework, because they typically can be searched more quickly. Large neighbourhoods generally lead to local solutions of better quality, but the search is more time-consuming. Thus, a natural idea is to gradually extend the size of the neighbourhood each time the search gets trapped in a local minimum (Fig. 2).

In the LNS metaheuristic, the neighbourhoods are implicitly defined by methods (often heuristics) which are used to destroy and repair an incumbent solution. A *destroy* method destructs part of the current solution while a *repair* method rebuilds the destroyed solution. The destroy method typically contains an element of stochasticity such that different parts of the solution are destroyed in every invocation of the method. The neighbourhood $N(x)$ of a solution x is then defined as the set of solutions that can be reached by first applying the destroy method and then the

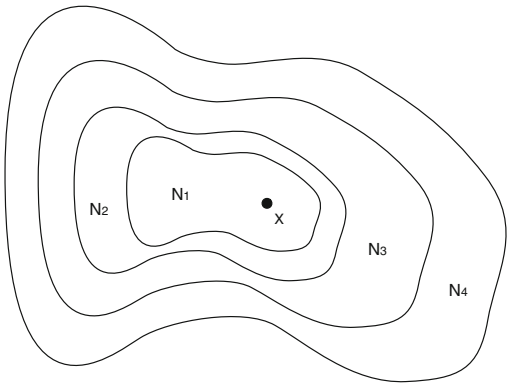


Fig. 2 Neighbourhood structure usually explored with Very Large Scale Neighbourhood (VLSN) search algorithms

repair method. Since the destroy method can destruct a large part of the solution, the neighbourhood contains a large amount of solutions, which explains the name of the heuristic. It should be noticed that the LNS metaheuristic does not search the entire neighbourhood of a solution, but merely samples it.

The steps of the LNS method are detailed in Algorithm 1 and depicted in Fig. 3. Three variables are maintained by the algorithm: the variable x^b is the best solution observed so far during the search, x is the current solution, and x' is a temporary solution that can be discarded or promoted to the status of current solution. The function $d(\cdot)$ is the destroy method while $r(\cdot)$ is the repair method. More specifically, $d(x)$ returns a copy of x that is partially destroyed. Applying $r(\cdot)$ to a partly destroyed solution repairs it, i.e. it returns a feasible solution built from the destroyed one. Both destroy and repair methods can be implemented in different ways obeying different criteria. In step 4 the new solution is evaluated, and then the heuristic determines whether this solution should become the new current solution or whether it should be rejected. The *accept* function can be implemented in different ways. The simplest choice is to only accept improving solutions, as shown in Fig. 3. In this case, x^b corresponds to the current solution x at any time and steps 4 and 6 in Algorithm 1 are merged. However, some works propose an acceptance criteria borrowed from SA [43], that is, accepting solutions that may be worse than the incumbent aiming to diversify the search.

The destroy method is an important part of the LNS heuristic. The most important choice when implementing the destroy method is the degree of destruction: if only a small part of the solution is destroyed then the heuristic may have troubles exploring the search space as the effect of a large neighbourhood is lost. If a very large part of the solution is destroyed, then the LNS heuristic almost degrades into repeated re-optimisation or a multi-start process. This can be time-consuming or yield poor quality solutions dependent on how the partial solution is repaired. Shaw [48] proposed to gradually increase the degree of destruction, while Ropke and Pisinger [43] choose the degree of destruction randomly at each iteration from a specific range dependent on the instance size. The destroy method must also be chosen such that

Algorithm 1: Large Neighbourhood Search (LNS)

```
1 x^b ← find an initial solution x
2 repeat
3 |   x' = r(d(x))
4 |   if accept(x', x) then
5 |   |   x ← x'
6 |   if f(x') < f(x^b) then
7 |   |   x^b ← x'
8 until stopping condition is met
9 return x^b
```

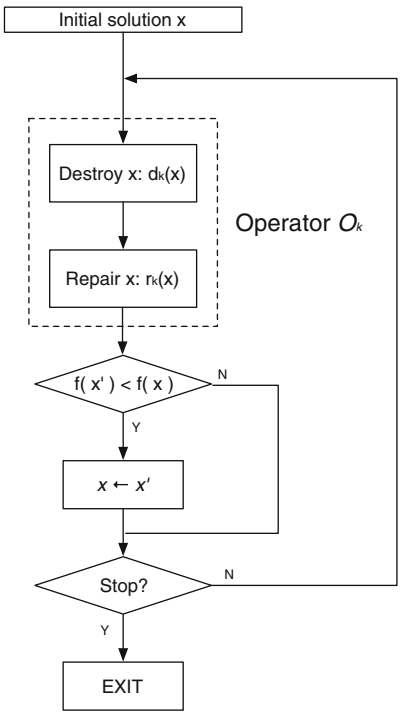


Fig. 3 Large Neighbourhood Search (LNS)

the entire search space can be reached, or at least the interesting part of the search space where the global optimum is expected to be found. Therefore, it cannot focus on always destroying a particular component of the solution but must be possible to destroy every part of the solution.

Diversification and intensification for the destroy methods can be accomplished as follows: to diversify the search, one may randomly select the parts of the solution that should be destroyed (*random destroy* method). To intensify the search, one may try to remove a number of "critical" variables, i.e. variables having a large cost or variables spoiling the current structure of the solution (*worst destroy* or *critical*

destroy, respectively). One may also choose a number of related variables that are easy to interchange while maintaining feasibility of the solution (*related destroy* method). Finally, one may use *history based destroy*, where a number of variables are chosen according to some historical information.

Choosing the repair method permits much more freedom when implementing a LNS heuristic. A first decision is whether the repair method should be optimal, in the sense that the best possible full solution is constructed from the partial solution, or whether it should be a heuristic, assuming that one is satisfied with a good solution constructed from the partial solution. An optimal repair operation will be slower than a heuristic one, but may potentially lead to high-quality solutions in a few iterations. However, from a diversification point of view, an optimal repair operation may not be attractive: only improving or identical-cost solutions will be produced. Therefore, it can be difficult to leave valleys in the search space unless a significant part of the solution is destroyed at each iteration.

Finally, several destroy and repair methods may be combined to explore multiple neighbourhoods within the same search. Neighbourhood structures may be nested or cover different regions of the search space. In general, these neighbourhoods are explored in a systematic fashion, i.e. switching to the next neighbourhood whenever the current solution is not improved, or using different strategies to enhance the search, such as *Variable Neighbourhood Search* [26]. A more sophisticated LNS variant is the *Adaptive Large Neighbourhood Search* (ALNS) heuristic proposed by Ropke and Pisinger [43]. In this case, each destroy/repair method is assigned a weight that controls how often the particular method is attempted during the search. The weights are adjusted dynamically as the search progresses depending on the performance of each neighbourhood, so that the heuristic adapts to the instance at hand and to the state of the search.

4.2 LNS Operators for the ARP

In LNS, neighbourhoods are implicitly defined by the *destroy* and *repair* operators. The destroy method typically contains an element of stochasticity such that different parts of the solution are destroyed in every invocation of the method. Nevertheless, deterministic destroy methods can also be implemented.

As introduced in Sect. 3.1, CP search methods are mainly based on assigning values to variables, in such a way that constraints are satisfied and other variables' domains are reduced to their compatible values through constraint propagation. Therefore, CP-based destroy and repair methods will *unassign* and *assign* values to variables, respectively, at different stages of the search. It can be inferred then that a solution is a complete assignment (or complete *labelling*) to the variables of the problem, in such a way that all constraints are satisfied at once.

A CP-based destroy method unassigns some values from a solution, destroying it partially. For the ARP, we have defined two destroy methods, based on the idea of extending the size of neighbourhoods to be explored. In the *1-airport destroy method*,

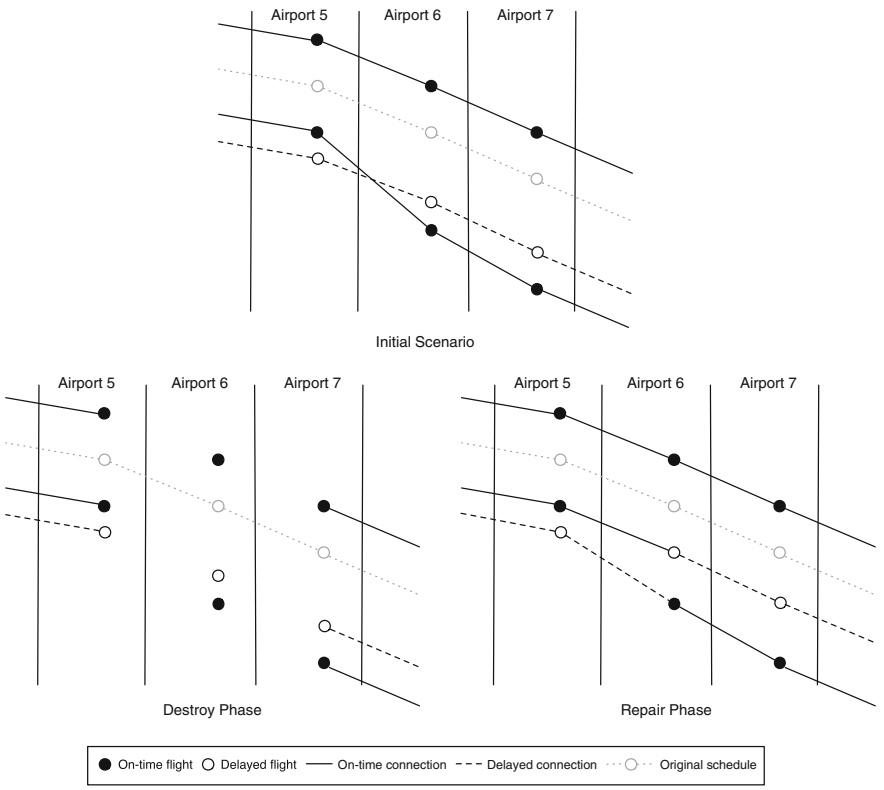


Fig. 4 Destroy/repair phases of the 1-airport operator

all aircraft-to-flight allocations are unassigned for all flights departing an airport (Fig. 4). As we are using the formulation introduced in Sect. 3.2, where redundant sets for *predecessor* and *successor* variables are used in order to enhance constraint propagation, aircraft allocations for previous and next flights should also be removed. Otherwise, the only feasible solution contained in the neighbourhood would be the preceding solution. The exploration of this neighbourhood permits swapping aircraft and delaying flights in one airport. We systematically use this operator to explore the corresponding neighbourhood for every airport present in the instance at hand. We only switch to the next operator with a higher degree of destruction when all airports have been explored and no improvement is found.

The second destroy method is based on the same principle of removing aircraft-to-flight allocations for particular airports. In this case, we unassign all variables corresponding to flights departing from three consecutive airports (*3-airport destroy method*). Therefore, we extend the size of the previously described neighbourhood by considering allocations of connected flight legs, rather than individual flights. In this case, the repair phase of the operator has more freedom to adjust flight departures (i.e. delaying flights) and swapping aircraft in particular legs.

It should be remarked that other destroy methods were considered. In particular, we explored neighbourhoods where the solution was partially destroyed for two consecutive airports. However, we found that all improvements obtained by means of this operator were also achieved and often exceeded by the 3-airport destroy method. All improvements obtained with the 1-airport operator can also be attained with the 3-airport operator, but the time required to explore the latter is higher. Therefore, we use 1-airport neighbourhoods to lead the algorithm to better solutions quicker. Starting closer to a local optimum reduces the search space for 3-airport operators, since previously explored feasible non-improving solutions are discarded. This way, these neighbourhoods can still be explored efficiently while reducing the required time. In terms of computational burden, we determined that larger neighbourhoods increase the required time excessively, due to the exact repair method we use.

We have chosen a CP-based repair method according to the formulation presented in Sect. 3.2. We use the same repair method for both 1-airport and 3-airport operators. Concretely, we apply a BB method (see Sect. 3.1) to repair the partially destroyed solution. Although slower than heuristic methods, we benefit from high-quality solutions, while not being penalised with an excessive computational time by considering reasonably sized neighbourhoods. During search, the upper bound is set to the total delay of the best solution found so far. Aircraft-to-flight allocations may be forbidden if they take the lower bound on the total delay over the defined upper bound. We form the lower bound as the current total delay of the partial solution constructed during search, i.e. the lower bound is not computed separately by any other method. This makes the repair method faster, but the search tree is larger than it would be if we calculate an accurate lower bound. In its simplest form, the BB search explores the whole tree for the reallocation of all flights to aircraft and rescheduling all flights within their feasible time ranges in order to minimise the total delay.

4.3 Using Simulation: SimLNS

Large Neighbourhood Search has proved to be an efficient metaheuristic to deal with complex combinatorial optimisation problems [40]. However, LNS is designed to provide high-quality solutions under deterministic scenarios. In some real-life problems, like in the case of the ARP and many air transportation applications, uncertainty is present. In these cases, a deterministic approach may not be accurate enough, since it does not reflect the real stochastic nature of the system. Therefore, it is necessary to extend the deterministic framework to account for the variability of the system. A natural way is to integrate simulation within the optimisation process to cope with such stochastic combinatorial optimisation problems, e.g. the SARP.

Traditional simulation approaches allow the user to model the stochastic behaviour of the system, as well as all interactions between different elements. Nevertheless, in most simulation-based approaches the level of optimisation achieved depends on the number of evaluated scenarios. In general, this number is a small fraction of available configurations, not guaranteeing optimality and providing poor feedback regarding

solution's quality. On the other hand, deterministic optimisation approaches do not take into account the uncertainty present in the system. Albeit using simulation to check the behaviour of different solutions is an extended practice, the decision on how to use the information provided by simulation is normally left to the final user. An example of this traditional approach applied to the ARP can be found in [44].

We propose two extensions of the LNS metaheuristic that integrates simulation at different stages of the search. The proposed approaches are similar to the *Iterated Local Search* extensions introduced by Grasas et al. [25]. They all fall within the *SimHeuristics* category [30].

In our first approach, which we call *SimLoop*, we include simulation as the final step to assess solution's robustness (Fig. 5). Up to this stage, the algorithm only accepts solutions that reduce the total delay with respect to the previous solution. Hence, either the initial solution or a solution with minor total delay is returned by the LNS framework previous to the simulation stage. However, this solution may not fulfil other desired characteristics, e.g. number of swaps, or may present an unacceptable degree of variance. We use simulation to test these attributes and provide feedback to the optimisation schema, extending the application of traditional simulation approaches.

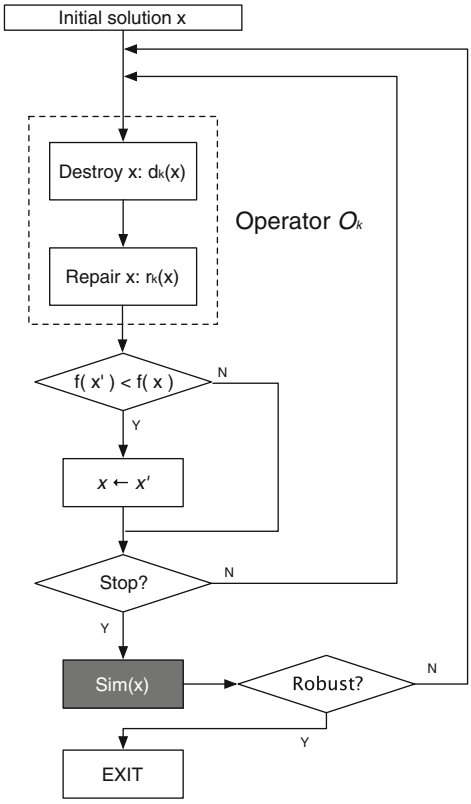


Fig. 5 SimLoop: Large Neighbourhood Search schema including simulation as the final step to assess solution's feasibility. Final solution's robustness is tested in this final step. In case the solution does not meet robustness criteria, results obtained from this process are used as feedback for further iterations of the local search process

Different criteria can be considered to determine whether a solution is robust or not. First, a solution may be considered to be robust if the standard deviation of the simulated solutions is proportional to the variation of the used probabilistic distributions and its expected propagation due to problems size. Second, a solution may be considered robust if the gap between the average of the simulated solutions and the deterministic solution falls within a tolerance interval. Third, we may define the criterion as the number of solutions whose gap to the deterministic solution is smaller than a given value. Finally, operational considerations such as the number of swapped flights/aircraft assignments may be introduced. In the practical case presented in this work, we use the first criterion to determine the robustness of the obtained solutions.

The proposed methodology for the SARP, which integrates SimLoop as the optimisation method, is structured in the following steps:

1. The stochastic problem is simplified to a deterministic instance by using the average values of the adjusted probability distributions of the different processes.
2. As the original flight schedule is known, we compute the total delay (the objective function) associated to this solution. This provides an initial value for the total delay, which is used as the initial upper bound for the objective function in the local search process. As the original flight schedule is known to be feasible, we use it as the initial solution for our method. Intuitively, starting from the original schedule and applying local changes leads to solutions with a smaller number of swaps, a desired characteristic for most airlines.
3. A deterministic LNS framework is then used as a local search process to improve the initial solution, allowing flights to be delayed and enabling swaps. An improved flight schedule reducing the total delay is found as the result of this step.
4. The optimised solution is then checked using simulation to verify its robustness: a set of stochastic instances is generated using the probability distributions for the processes, namely flight and turnaround times. Maintaining the improved flight schedule returned by the LNS, we compute the total delay for each instance. This way, a single solution is evaluated in different scenarios. The results are analysed in order to determine the level of robustness of the obtained solution. If the solution is not robust, its objective function value is used as a lower bound and the optimisation process is repeated, i.e. the LNS is re-launched with a new lower bound. This way a worst solution may be found but with better robustness characteristics. Otherwise, the solution is accepted and the algorithm ends. Thus, the methodology returns either the optimal deterministic solution, if it is robust enough, or a quasi-optimal one whose properties are more suitable for the stochastic problem at hand.

In our second approach, which we call *SimLNS*, we integrate simulation at an earlier stage in the process, as depicted in Fig. 6. In this case, simulation is embedded in the acceptance criterion instead of being used to evaluate the final result. The solution obtained after applying destroy and repair methods is then tested in a set of scenarios before deciding whether it is accepted or discarded. Simulation results may be used

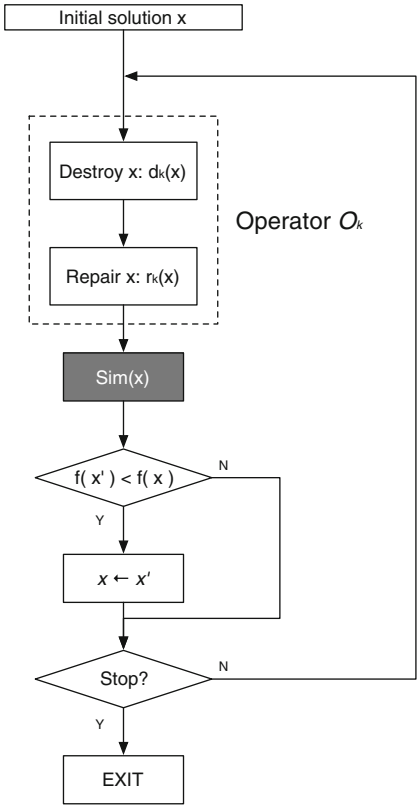


Fig. 6 SimLNS: Large Neighbourhood Search schema including simulation as part of the acceptance criterion. The solution obtained by operator O_k is accepted or discarded according to results obtained from its simulation

to evaluate the solution in different ways. The most common approach consists of accepting solutions which, on average, are better than the incumbent. Hence, we are only accepting solutions whose average behaviour is better than previous solutions. A different criterion is borrowed from robust optimisation approaches [11], where we only accept solutions whose simulated worst case improves previous solution's worst case. Thus, we aim at accepting solutions able to perform reasonably well under challenging conditions.

By moving the simulation step to the acceptance criterion, our goal is to detect earlier in the process solutions with undesired attributes, e.g. extremely variable solutions due to tight connection times. If a solution has a lower deterministic total delay, but on average presents a worse behaviour due to solution's variability, it may be rejected and the local search process may proceed to evaluate the next neighbourhood. Using a more traditional approach or the SimLoop method, this solution would be accepted and would not be tested until the end of the local search process, therefore detecting late its unwanted aspects. On the other hand, the approach adopted in the SimLNS increases substantially the number of solutions to be evaluated. Although simulating a higher number of scenarios may increase the reliability of the final

solution, so does the required computational time. Therefore, we need to find a trade-off between algorithm's execution time and the number of scenarios per solution to be tested.

5 Application

We have tested the methodologies described in the previous section on a set of instances with different characteristics. In all cases, tests have been done in a personal computer with an Intel Core i7 processor at 2.9 GHz and 8 Gb of RAM, running OS X 10.9. The different SimLNS variants have been implemented in Java, whereas the CP model (see Sect. 3.2) has been coded using the ECLiPSe 6.0 platform [4]. Simulation processes have also been implemented in Java.

Since most articles devoted to the ARP deal with specific instances (see Sect. 2), there are no accepted benchmarks for the ARP. Thus, we had to define a set of instances to assess our proposed methodologies. We have generated two separate sets with different attributes.

The first set contains scenarios with dense networks, i.e. networks with a higher density of flights and a larger connectivity degree for airports. Since we consider a dense network, the number of feasible swaps at each airport increases and so does the size of the neighbourhoods to be explored. These instances are purely theoretical and their goal is to push our methodology in more challenging and difficult scenarios. Details of the generated theoretical instances are as follow:

- Scenario 50 (denoted *50_*): consists of 49 flights, 3 airports, and 8 aircraft. In this scenario, all the airports have approximately the same number of flights.
- Scenario 100 (denoted *100_*): consists of 98 flights, 6 airports, and 16 aircraft. Again, airports have approximately the same number of flights. In both the scenarios 50 and 100, there is no visible hub.
- Scenario 200 (denoted *200_*): consists of 196 flights, 11 airports, and 32 aircraft. In this scenario, airport 1 has nearly twice the number of flights than other airports, therefore behaving like a hub. It also gives more opportunities for swapping aircraft.
- Scenario 300 (denoted *300_*): consists of 294 flights, 17 airports, and 48 aircraft. Again, we use the first airport as a hub to get more swapping opportunities.

The second set contains instances derived from real data provided by a Spanish airline (due to confidentiality agreements we cannot disclose the name of the airline). This airline relies on a hub-and-spoke network configuration. This kind of network provides fewer opportunities to swap aircraft between flights outside hub airports and generally propagates delays much faster to different parts of the network. We have generated several disrupted scenarios of different sizes from the original flight data provided by the airline. To keep consistency among instances, all of them have similar flights per airport and flights per aircraft ratios. Further details of real scenarios characteristics are described next:

- Real scenario 50 (denoted *real_50_*): consists of 49 flights, 17 airports, and 9 aircraft. In this scenario, Madrid's airport works as a hub.
- Real scenario 100 (denoted *real_100_*): consists of 110 flights, 23 airports, and 23 aircraft. Madrid's airport works as a hub.
- Real scenario 150 (denoted *real_150_*): consists of 163 flights, 35 airports, and 40 aircraft. This scenario is a whole day of operation for the airline. Again, Madrid's airport works as a hub.

In the real scenarios, Madrid's airport is the main hub. Two more airports constitute secondary hubs for the airline: Palma de Mallorca and Barcelona. As it can be seen, the density of the real scenarios is substantially smaller than the theoretical scenarios, i.e. for a similar number of flights, the number of airports and aircraft is significantly higher for real scenarios.

For each described scenario, we generate four instances with different degrees of disruption. To do so, we introduce delays at selected flights and airports. These delays range from 30 to 120 min, increased in 30-minute intervals. The size of disruptions is denoted as a suffix in scenario's name, e.g. *50_30* corresponds to an instance of scenario 50 with 30-minute delays in some flights. In total, we generated 28 instances corresponding to disrupted scenarios: 16 with a dense network and 12 based on real data with a hub-and-spoke configuration.

Results for all instances are presented in Table 1. It shows the obtained total delay and computational times for all described methods: (i) deterministic LNS approach (*Det.*), simulating the best solution in the final step but not providing feedback to the optimisation process; (ii) SimLoop approach (*SimLoop*), using simulation to get feedback on solution's robustness; (iii) SimLNS approach (*SimLNS*), which uses simulation in the acceptance criterion optimising the average total delay; (iv) SimLNS approach optimising the worst solution (*SimLNS-W*), instead of the average total delay. Table 1 also contains the total delay of the original flight schedule (*Orig.*).

The original schedule is used as the initial solution for all approaches. Starting from the original schedule and performing local moves may lead to solutions with a minor total delay and few swaps. If a construction heuristic is used to obtain the initial solution, the algorithm may start at a region of the search space far from the original schedule. Although the obtained final solution may be better in terms of total delay, it may imply a large number of swaps regarding the original schedule, an unacceptable characteristic for most airlines.

In all approaches, we run 20 tests per solution whenever the simulation process is called. This means that the total number of run simulations may differ depending on the applied methodology. Probability distributions for flight durations and turnaround times are adjusted to reflect a similar behaviour to reported observations [19]. For the SimLoop approach, a 5 % limit on the standard deviation has been imposed to consider a solution as robust.

As for operators, instances *100*, *200* and *300* are solved using only the 1-airport operator. Utilising the 3-airport operator for these instances increases the computational time dramatically. On the other hand, instances *50* and all real scenarios are solved using both 1- and 3-airport operators.

Table 1 Results for 28 tested instances

Instance	Orig.	Det.	CPU (s)	SimLoop	CPU (s)	SimLNS	CPU (s)	SimLNS-W	CPU (s)
50_30	228.2	213.2	0.395	213.4	0.398	210.3	0.435	213.5	0.817
50_60	718.6	652.1	0.572	653.0	0.570	635.9	1.692	642.6	1.178
50_90	1458.4	1346.4	0.640	1395.1	0.624	1360.4	0.659	1369.9	1.270
50_120	2450.9	2346.1	0.623	2335.5	0.618	2290.4	0.871	2306.9	0.652
100_30	264.0	255.9	11.656	253.5	11.520	244.3	42.764	250.2	11.931
100_60	738.4	703.6	13.175	715.0	12.710	687.0	12.977	685.8	12.856
100_90	1577.3	1503.4	14.656	1513.5	13.752	1460.2	14.321	1460.1	14.312
100_120	2577.0	2476.9	14.216	2451.7	13.844	2430.5	27.908	2452.6	27.818
200_30	359.2	338.3	56.234	334.3	58.060	319.6	123.367	336.1	59.226
200_90	1692.5	1484.1	136.920	1470.4	145.809	1465.8	262.170	1478.2	130.145
200_60	959.9	813.6	68.088	815.0	70.195	792.1	69.624	809.1	87.114
200_120	2401.8	2023.1	204.585	2004.6	191.781	1993.5	340.803	2005.3	178.114
300_30	421.6	405.3	97.388	402.3	99.915	396.2	96.324	399.5	97.419
300_60	1010.6	950.1	101.819	935.0	104.345	922.1	282.811	931.7	100.527
300_90	1816.3	1649.4	97.697	1648.5	97.772	1609.1	102.590	1658.5	103.085
300_120	2605.4	2425.0	94.475	2417.6	96.248	2391.2	205.514	2427.6	100.611
real_50_30	120.6	121.0	6.618	121.5	6.518	120.1	6.748	120.0	6.610
real_50_60	241.2	241.2	6.411	241.9	6.616	240.1	6.364	240.3	6.828
real_50_90	385.4	387.0	7.231	385.9	8.510	383.5	7.318	384.2	6.971
real_50_120	535.8	536.6	6.642	536.2	6.716	533.5	13.347	533.9	7.017
real_100_30	450.6	451.3	8.386	451.5	8.356	450.1	8.198	450.4	8.086
real_100_60	906.0	907.3	8.416	905.4	8.446	904.0	8.433	903.9	8.133

(continued)

Table 1 (continued)

Instance	Orig.	Det.	CPU (s)	SimLoop	CPU (s)	SimLNS	CPU (s)	SimLNS-W	CPU (s)
real_100_90	1505.4	1494.7	10.722	1499.8	10.159	1490.1	30.471	1495.1	10.293
real_100_120	2322.2	2292.0	10.806	2290.2	10.931	2285.1	24.084	2285.7	21.691
real_150_30	98.4	97.9	36.160	102.5	38.536	96.3	81.135	98.9	37.036
real_150_60	196.4	194.2	37.341	193.1	36.820	188.4	42.760	188.4	46.805
real_150_90	336.2	309.5	55.739	304.8	43.308	303.6	85.841	304.0	87.155
real_150_120	513.5	425.2	38.368	427.2	40.655	424.0	226.505	425.4	38.444

Total delay (in minutes) and computational time are reported for the original schedule and solutions obtained by means of the different described methodologies

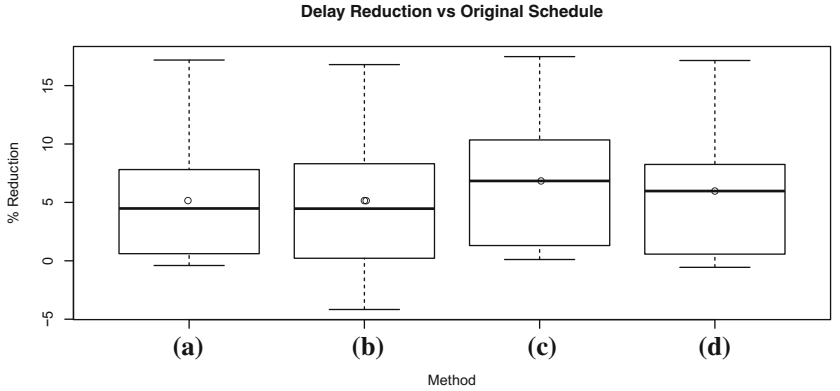


Fig. 7 Relative delay reduction with respect to the original schedule for all tested instances using different methods: Deterministic approach using simulation at the end of the process (**a**); SimLoop, using simulation to provide feedback on solution's robustness (**b**); SimLNS using simulation integrated in the acceptance criterion (**c**); SimLNS-W using simulation integrated in the acceptance criterion, but optimising for the worst obtained solution (**d**)

In general, we observe that SimLNS performs better than other approaches. For most instances, the largest reduction on total delay is achieved by means of this methodology, although computational time is slightly increased due to a higher number of simulations is required. These results are corroborated in Fig. 7. In this figure, the relative reduction of the total delay regarding the original schedule is presented for all approaches over all instances. We observe clearly that major reductions are obtained by means of the SimLNS approach. This means that applying simulation during the search process may lead to better results than performing it as a final step.

It can be noticed that total delay reductions are lower for instances based on real data than for those based on a dense network. This can be easily explained by the network topology of real instances, since hub-and-spoke configurations provide less opportunities to perform swaps in order to reduce delays. In addition, initial delays account for a bigger proportion of total delay in instances based on real data. As a general practice to avoid consequences from unforeseen events, airlines introduce additional buffer time between flights. Since we are using real data for these instances, these oversized buffers are present and can absorb part of the initial delays, reducing their propagation through the network. We have defined tighter flight connections in the theoretical instances.

It is important to notice that total delay is not necessarily higher in large instances. This is due to the fact that for most instances initial delays are introduced in the first flights departing the hub airport, i.e. all flights up to one hour from the first departing flight are delayed. Some large scenarios have less flights departing from the hub during the first hour, therefore having a smaller initial delay.

As for efficiency terms, we observe that instances based on real data require more time to be solved. The cause may be attributed to two combined facts: we use the

Table 2 Results for 28 tested instances when only delays greater than 15 min are considered

Instance	Orig.	Det.	SimLoop	SimLNS	SimLNS-W
50_30	182.1	168.6	167.4	167.2	172.9
50_60	672.0	607.7	600.1	589.0	592.2
50_90	1421.7	1303.5	1348.1	1309.6	1323.0
50_120	2442.3	2332.1	2325.2	2275.6	2292.0
100_30	187.8	178.1	179.5	171.9	178.1
100_60	672.2	642.4	649.2	620.0	619.8
100_90	1520.5	1445.6	1454.5	1400.4	1393.0
100_120	2555.4	2453.3	2426.8	2406.2	2427.3
200_30	295.7	248.5	245.9	239.5	248.7
200_90	1664.9	1395.7	1394.9	1392.2	1396.0
200_60	901.7	703.3	701.8	691.2	703.5
200_120	2372.0	1964.9	1946.9	1928.5	1953.0
300_30	282.1	259.9	254.5	246.0	253.3
300_60	879.9	810.4	793.3	774.6	787.7
300_90	1716.3	1523.9	1515.3	1477.2	1531.8
300_120	2510.7	2312.3	2310.5	2277.2	2312.7
real_50_30	120.0	120.0	120.0	120.0	120.0
real_50_60	240.0	240.0	240.0	240.0	240.0
real_50_90	384.6	385.4	384.4	383.1	383.8
real_50_120	534.9	536.2	534.9	533.2	533.0
real_100_30	450.0	450.0	450.0	450.0	450.0
real_100_60	900.0	900.0	900.0	900.0	900.0
real_100_90	1471.0	1459.9	1463.1	1461.4	1462.3
real_100_120	2319.3	2290.4	2287.5	2281.6	2283.9
real_150_30	90.8	90.0	92.9	90.0	90.0
real_150_60	186.1	182.2	184.0	181.0	180.9
real_150_90	317.0	301.9	296.8	296.6	296.2
real_150_120	504.0	417.5	421.6	418.2	417.5

Total delay (in minutes) is reported for the original schedule and solutions obtained by means of the different described methodologies

two defined airport-based operators and there is major number of airports present in these instances.

Although we account for total delay, airlines only consider delays larger than 15 min. Table 2 shows our results under this consideration. Equally, Fig. 8 presents a graphical interpretation on the relative reduction of delays in excess of 15 min. As expected from previous results, embedding simulation in the search yields better results than using simulation at the end of the process. This is clearly appreciated in Fig. 8, where we observe that SimLNS provides approximately a 10 % average delay reduction and over 20 % for some instances. We also see in this figure how both SimLNS methods outperform more traditional approaches.

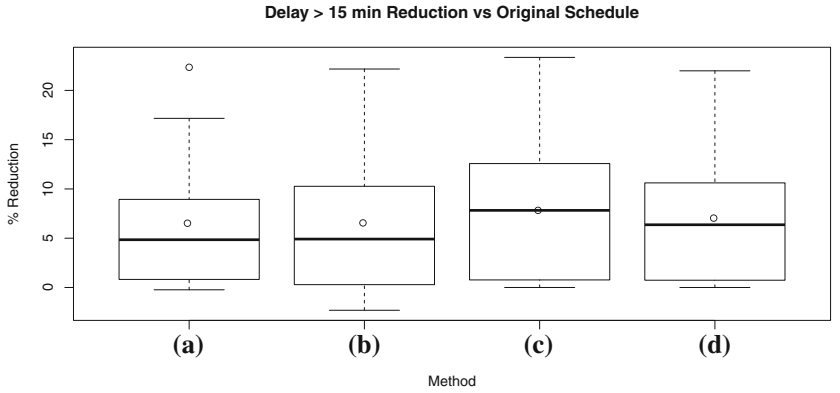


Fig. 8 Relative delay reduction for delays greater than 15 min with respect to the original schedule for all tested instances using different methods: Deterministic approach using simulation at the end of the process (**a**); SimLoop, using simulation to provide feedback on solution's robustness (**b**); SimLNS using simulation integrated in the acceptance criterion (**c**); SimLNS-W using simulation integrated in the acceptance criterion, but optimising for the worst obtained solution (**d**)

Table 3 Estimated cost of disruptions (in euros) for instances generated from real data

Instance	Orig.	Det.	SimLoop	SimLNS	SimLNS-W
real_50_30	9,840.00	9,840.00	9,840.00	9,840.00	9,840.00
real_50_60	19,680.00	19,680.00	19,680.00	19,680.00	19,680.00
real_50_90	31,533.10	31,598.70	31,520.80	31,414.20	31,471.60
real_50_120	43,861.80	43,964.30	43,861.80	43,718.30	43,706.00
real_100_30	36,900.00	36,900.00	36,900.00	36,900.00	36,900.00
real_100_60	73,800.00	73,800.00	73,800.00	73,800.00	73,800.00
real_100_90	120,617.90	119,707.70	119,970.10	119,830.70	119,904.50
real_100_120	190,178.50	187,812.80	187,570.90	187,087.10	187,279.80
real_150_30	7,445.60	7,380.00	7,617.80	7,380.00	7,380.00
real_150_60	15,256.10	14,936.30	15,088.00	14,837.90	14,829.70
real_150_90	25,994.00	24,755.80	24,333.50	24,321.20	24,284.30
real_150_120	41,328.00	34,235.00	34,571.20	34,292.40	34,235.00

Results are reported for all described methodologies

As mentioned in Sect. 1, airlines face significant monetary costs when disruptions occur. If we consider the reported cost of delays beyond the first 15 min, €82 per minute according to CODA [20], we can estimate the associated cost of each solution. We report these results for instances based on real data in Table 3. It can be seen that consequences of a disruption affecting a reduced number of flights may escalate quickly to costs over €190,000 (e.g., instance *real_100_120*). This observation reinforces the fact that decision support tools can be extremely useful in disrupted scenarios in order to attain feasible solutions able to reduce incurred costs in reasonable times.

6 Conclusions

Operational disruptions are deviations from originally planned operations. Airlines are among the most affected industries by this kind of events and have to face significant associated costs. For this reason, designing methods to deal with operational disruptions efficiently are getting an increasing attention from airlines and the research community.

Cancelled, delayed and diverted flights impact different elements involved in airlines' operations: aircraft, crews and passengers. Traditionally, aircraft are seen as the scarce resource, as well as a more tractable problem due to its size. In this work, we have tackled the Aircraft Recovery Problem, whose goal is to restore the original flight schedule as much as possible in a disrupted scenario by means of swapping aircraft-to-flight allocations and delaying flights when necessary. As real operations are subject to variability, we define an ARP variant which includes this inherent uncertainty: the Stochastic Aircraft Recovery Problem.

As a first step, we have developed a Constraint Programming model to solve the deterministic ARP. To the best of our knowledge, it is the first CP formulation presented for this problem. We integrate this model within a Large Neighbourhood Search framework, a metaheuristic which has proved to be very efficient when combined with CP to cope with a variety of combinatorial optimisation problems. In our approach, we embed the CP model in operators and take advantage of constraint propagation efficiency in the destroy and repair phases of the LNS. Concretely, we unassign a set of variables in the destroy phase and use an exact branch and bound method to re-optimise the partially destroyed solution. This method has shown to be efficient for solving the ARP.

In order to deal with the stochasticity present in the SARP, we have modified our LNS approach and included simulation at different stages of the search. In particular, we define two LNS-based frameworks that make extensive use of simulation. In our first schema, the SimLoop, simulation is performed at the end of the LNS method. This way, we simulate the solution obtained in the optimisation process to check its robustness. In this test, different criteria for robustness or solution's characteristics may be used. If the solution is rejected, a new lower bound is imposed and the LNS process is re-launched. In our second approach, the SimLNS, simulation is integrated at an earlier stage of the search. After the destroy/repair phase of the LNS, we test the obtained solution in several simulated scenarios and utilise these results in the acceptance criterion. Therefore, we check solutions' behaviour before accepting or rejecting them, allowing to detect undesirable attributes earlier in the process.

Results show that the proposed LNS variants constitute an efficient approach to tackle the SARP. Indeed, as they are defined as general methodologies, the presented LNS-based methods can be used to solve any combinatorial problem where stochasticity is an inherent characteristic of the system. In general, the SimLNS methodology provides the best solutions for most instances, although computational times are slightly higher because of a major number of simulated scenarios. Among defined instances, we have observed that scenarios based on real data provide few

margin for improvement due to network configuration and oversized buffers. We believe that further developments on efficient tools to support decision making in disrupted scenarios may lead to schedules with reduced buffer times, as airlines may be able to respond more effectively to unforeseen events.

The work presented in this chapter leaves several lines open for future research. On the one hand, we aim at improving search efficiency and being able to increase neighbourhood's size. With this purpose, a more effective CP representation of the ARP is to be developed in order to enhance constraint propagation. This upgraded formulation may include other elements present in a disrupted scenario, i.e. crews and passengers. On the other hand, we contemplate integrating simulation within the CP search tree. This way, simulation is used to propagate most likely values in the domain of stochastic variables during search, instead of simulating complete scenarios after labelling variables in a deterministic fashion.

Acknowledgments NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

References

1. Association of European Airlines: European airline punctuality in 1st quarter 2008. Tech. rep., Association of European Airlines (2008)
2. Ahuja, R., Ergun, O., Orlin, J., Punnen, A.: A survey of very large-scale neighborhood search techniques. Discrete Applied Mathematics **123**, 75–102 (2002)
3. Apt, K.: Principles of Constraint Programming. Cambridge University Press, Amsterdam (2003)
4. Apt, K., Wallace, M.: Constraint Logic Programming using ECLiPSe. Cambridge University Press, New York, USA (2007)
5. Argüello, M.F., Bard, J.F., Yu, G.: A GRASP for aircraft routing in response to grounding and delays. Journal of Combinatorial Optimization **5**, 211–228 (1997)
6. Argüello, M.F., Bard, J.F., Yu, G.: Models and methods for managing airline irregular operations. In: G. Yu (ed.) Operations Research in the Airline Industry, *International Series in Operations Research & Management Science*, vol. 9, pp. 1–45. Springer-Verlag (1998)
7. Arias, P., Guimarans, D., Mújica, M.A.: A new methodology to solve the stochastic aircraft recovery problem using optimization and simulation. In: International Conference on Interdisciplinary Science for Innovative Air Traffic Management (ISIATM). Toulouse, France (2013)
8. Arias, P., Guimarans, D., Mújica, M.A., Boosten, G.: A methodology combining optimization and simulation for real applications of the stochastic aircraft recovery problem. In: 8th EUROSIM Congress on Modelling and Simulation, pp. 265–270. Cardiff, UK (2013)
9. Arora, S., Barak, B.: Computational complexity: A modern approach. Cambridge University Press, New York, USA (2009)
10. Ball, M., Barnhart, C., Nemhauser, G., Odoni, A.: Air transportation: irregular operations and control, *Handbooks in Operations Research & Management Science*, vol. 14, chap. 1, pp. 1–67. Elsevier (2007)
11. Bertsimas, D., Nohadani, O., Teo, K.M.: Robust optimization for unconstrained simulation-based problems. Operations Research **58**(1), 161–178 (2010)
12. Bessiere, C.: Constraint Propagation, vol. Handbook of Constraint Programming, chap. 3, pp. 29–83. Elsevier, Amsterdam (2006)

13. Cao, J., Kanafani, A.: Real-time decision support for integration of airline flight cancellations and delays. Part I: Mathematical formulation. Transportation Planning and Technology **20**(3), 183–199 (1997)
14. Cao, J., Kanafani, A.: Real-time decision support for integration of airline flight cancellations and delays. Part II: Algorithm and computational experiments. Transportation Planning and Technology **20**(3), 201–217 (1997)
15. Clausen, J., Larsen, A., Larsen, J., Rezanova, N.J.: Disruption management in the airline industry – Concepts, models and methods. Computers & Operations Research **37**(5), 809–821 (2010)
16. Dantzig, G.B., Wolfe, P.: Decomposition principles for linear programs. Operations Research **8**(1), 101–111 (1960)
17. Eggenberg, N., Salani, M., Bierlaire, M.: A column generation algorithm for disrupted airline schedules. Tech. Rep. TRANSP-OR 071203, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland (2007)
18. Eggenberg, N., Salani, M., Bierlaire, M.: Constraint-specific recovery network for solving airline recovery problems. Computers & Operations Research **37**, 1014–1026 (2010)
19. EUROCONTROL: Report on punctuality drivers at major European airports. Tech. rep., EUROCONTROL (2005)
20. EUROCONTROL: Planning for delay: influence of flight scheduling on airline punctuality. Tech. Rep. Trends in Air Traffic - Vol. 7, EUROCONTROL (2010)
21. European Commission: Guide to European Community legislation in the field of civil aviation. Directorate-General for Energy and Transport (2007)
22. Freuder, E.C., Mackworth, A.K.: Constraint Satisfaction: an Emerging Paradigm, vol. Handbook of Constraint Programming, chap. 2, pp. 13–27. Elsevier, Amsterdam (2006)
23. Gendreau, M., Potvin, J.Y.: Tabu Search, vol. Handbook of Metaheuristics, chap. 2, pp. 41–60. Kluwer Academic, Boston, MA (2010)
24. Granberg, T.A., Värbrand, P.: The flight perturbation problem. Transportation Planning and Technology **27**(2), 91–117 (2004)
25. Grasas, A., Juan, A.A., Lourenço, H.R.: SimILS: a simulation-based extension of the Iterated Local Search metaheuristic for stochastic combinatorial optimization. Journal of Simulation (2014). DOI 10.1057/jos.2014.25
26. Guimarans, D.: Hybrid algorithms for solving routing problems. Ph.D. thesis, Universitat Autònoma de Barcelona, Barcelona (2012)
27. Guimarans, D., Herrero, R., Ramos, J.J., Padrón, S.: Solving vehicle routing problems using constraint programming and lagrangian relaxation in a metaheuristics framework. International Journal of Information Systems and Supply Chain Management **4**(2), 61–81 (2011)
28. Guimarans, D., Herrero, R., Riera, D., Juan, A.A., Ramos, J.J.: Combining probabilistic algorithms, constraint programming and lagrangian relaxation to solve the vehicle routing problem. Annals of Mathematics and Artificial Intelligence **62**(3–4), 299–315 (2011)
29. Jarrah, A.I.Z., Yu, G., Krishnamurthy, N., Rakshit, A.: A decision support framework for airline flight cancellations and delays. Transportation Science **27**(3), 266–280 (1993)
30. Juan, A.A., Grasman, S.E., Caceres-Cruz, J., Bektaş, T.: A simheuristic algorithm for the single-period stochastic inventory routing problem with stock-outs. Simulation Modelling Practice and Theory **46**, 40–52 (2014)
31. Kilby, P., Shaw, P.: Vehicle Routing, vol. Handbook of Constraint Programming, chap. 23, pp. 801–836. Elsevier, Amsterdam (2006)
32. Kohl, N., Larsen, A., Larsen, J., Ross, A., Tiourine, S.: Airline disruption management – Perspectives, experiences and outlook. Journal of Air Transportation Management **13**(3), 149–162 (2007)
33. Kumar, V.: Algorithms for constraint-satisfaction problems: a survey. AI Magazine **13**(1), 32–44 (1992)
34. Lawler, E.L., Wood, D.E.: Branch-and-bound methods: A survey. Operations Research **14**(4), 699–719 (1966)

35. Le, M., Wu, C., Zhan, C., Sun, L.: Airline recovery optimization research: 30 years' march of mathematical programming – A classification and literature review. In: 2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE), pp. 113–117. IEEE, Changchun, China (2011)
36. Løve, M., Sørensen, K.R.: Disruption management in the airline industry. Master's thesis, Technical University of Denmark, Lyngby, Denmark (2001)
37. Løve, M., Sørensen, K.R., Larsen, J., Clausen, J.: Disruption management for an airline – Rescheduling of aircraft. In: S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, G.R. Raidl (eds.) Applications of Evolutionary Computing, *Lecture Notes in Computer Science*, vol. 2279, pp. 315–324. Springer-Verlag, Berlin Heidelberg (2002)
38. Nikolaev, A.G., Jacobson, S.H.: Simulated Annealing, vol. Handbook of Metaheuristics, chap. 1, pp. 1–40. Kluwer Academic, Boston, MA (2010)
39. Perron, L., Shaw, P., Furnon, V.: Propagation guided large neighborhood search. In: M. Wallace (ed.) 10th International Conference on Principles and Practice of Constraint Programming (CP-04), *Lecture Notes in Computer Science*, vol. 3258, pp. 468–481. Springer-Verlag, Berlin Heidelberg (2004)
40. Pisinger, D., Ropke, S.: Large Neighborhood Search, vol. Handbook of Metaheuristics, chap. 13, pp. 399–420. Kluwer Academic, Boston, MA (2010)
41. Reeves, C.R.: Genetic Algorithms, vol. Handbook of Metaheuristics, chap. 5, pp. 109–140. Kluwer Academic, Boston, MA (2010)
42. Resende, M.G.C., Ribeiro, C.C.: Greedy randomized adaptive search procedures: advances, hybridizations, and applications, vol. Handbook of Metaheuristics, chap. 10, pp. 283–320. Kluwer Academic, Boston, MA (2010)
43. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science **40**(4), 455–472 (2006)
44. Rosenberger, J.M., Johnson, E.L., Nemhauser, G.L.: Rerouting aircraft for airline recovery. Transportation Science **37**(4), 408–421 (2003)
45. Rosenberger, J.M., Schaefer, A.J., Goldsman, D., Johnson, E.L., Kleywegt, A.J., Nemhauser, G.L.: A stochastic model of airline operations. Transportation Science **36**(4), 357–377 (2002)
46. Rossi, F., van Beek, P., Walsh, T. (eds.): Handbook of Constraint Programming. Elsevier, Amsterdam (2006)
47. Shavell, Z.A.: The effects of schedule disruptions on the economics of airline operations. In: 3rd USA/Europe Air Traffic Management R&D Seminar. Napoli, Italy (2000)
48. Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. In: 4th International Conference on Principles and Practice of Constraint Programming (CP-98), *Lecture Notes in Computer Science*, vol. 1520, pp. 417–431. Springer-Verlag, Berlin Heidelberg (1998)
49. Teodorović, D., Guberinić, S.: Optimal dispatching strategy on an airline network after a schedule perturbation. European Journal of Operational Research **15**(2), 178–182 (1984)
50. Teodorović, D., Stojković, G.: Model for operational daily airline scheduling. Transportation Planning and Technology **14**(4), 273–285 (1990)
51. Thengvall, B.G., Bard, J.F., Yu, G.: Balancing user preferences for aircraft schedule recovery during irregular operations. IIIE Transactions **32**, 181–193 (2000)
52. Thengvall, B.G., Yu, G., Bard, J.F.: Multiple fleet aircraft schedule recovery following hub closures. Transportation Research A **35**, 289–308 (2001)
53. van Hoeve, W.J., Katriel, I.: Global constraints, vol. Handbook of Constraint Programming, chap. 6, pp. 169–208. Elsevier, Amsterdam (2006)
54. Wu, C., Le, M.: A new approach to solve aircraft recovery problem. In: INFOCOMP 2012: The Second International Conference on Advanced Communications and Computation, pp. 148–154. Venice, Italy (2012)
55. Wu, C.L.: Inherent delays and operational reliability of airline schedules. Journal of Air Transportation Management **11**(4), 273–282 (2005)
56. Wu, C.L.: Improving airline network robustness and operational reliability by sequential optimisation algorithms. Networks and Spatial Economics **6**(3–4), 235–251 (2006)

57. Yan, S., Lin, C.: Airline scheduling for the temporary closure of airports. Transportation Science **31**(1), 72–82 (1997)
58. Yan, S., Tu, Y.: Multifleet routing and multistop flight scheduling for schedule perturbation. European Journal of Operational Research **103**(1), 155–169 (1997)
59. Yan, S., Yang, D.: A decision support framework for handling schedule perturbation. Transportation Research B **30**(6), 405–419 (1996)
60. Yan, S., Young, H.: A decision support framework for multi-fleet routing and multi-stop flight scheduling. Transportation Research A **30**(5), 379–398 (1996)
61. Zhang, X., Zhao, M., Kuang, S.M., Du, Q.: Research on airline company fuel-saving model based on petri network. Advanced Materials Research **616–618**, 1107–1110 (2013)

Allocation of Airport Check-in Counters Using a Simulation-Optimization Approach

Miguel Mujica Mota and Catya Zuniga Alcaraz

Abstract The aviation industry is expected to grow at a pace of 4 % per annum in the coming years, therefore it is necessary to have techniques that support the management of the resources at hand in the best possible way so that facility expansion is delayed as much as possible with the corresponding capital savings. This chapter presents a methodology that combines evolutionary algorithms and simulation for performing the allocation of the check-in desks in such a way that the different stochastic and deterministic variables are taken into account for a more robust solution. The evolutionary algorithm is developed to satisfy the different mandatory restrictions for the allocation problem such as minimum and maximum number of check-in desks per flight, load balance at the counters, opening times of check-in desks, and other restrictions imposed by the level of service agreement. Once the solutions are obtained, a second evaluation is performed using a simulation model of the terminal that takes into account the stochastic aspects of the problem such as passenger arrival profiles, passenger profile, layout of the facility, among others, with the purpose of determining an airport terminal's check-in area which allocation is the most efficient in real situations to keep the quality indicators at the desired level. The example presented is for an airport terminal's check-in area, but the methodology can be used for similar allocation problems in the aviation industry and in other industries such as logistics or manufacturing.

1 Introduction

The aeronautical industry is still under expansion, in spite of the rise in oil prices, limited capacity, and new regulations. Different scenarios have been forecasted to explore the future of the Air Transportation System, the most likely scenario predicts

M. Mujica Mota (✉)
Aviation Academy, Amsterdam University of Applied Sciences, 1097, DZ, Amsterdam, The Netherlands
e-mail: m.mujica.mota@hva.nl

C.Z. Alcaraz (✉)
Logistics & Supply Chain Management Department, Universidad Popular Autonoma del Estado de Puebla, 17 Sur 901, Barrio de Santiago, 72410 Puebla, Puebla, Mexico
e-mail: catyaatziry.zuniga@upaep.mx

M. Mujica Mota et al. (eds.), *Applied Simulation and Optimization*,
DOI 10.1007/978-3-319-15033-8_7

that air traffic will double in the next 15 years. In the forecast of IFR flight movements in Europe up to 2035, the most likely scenario predicts 14.4 million flights, 50 % more than in 2012. Even under relatively conservative assumptions a steady 4–5 % annual growth will lead to a near doubling of total air travel during this period [1–3].

Increased air traffic makes the efficient management of available resources on both the airside and the landside of an airport even more complex. On the airside, it is even more evident on the runways and in the airspace surrounding airports, where the arrivals and departures serve a large number of aircraft that are subject to many logistical problems, which must continuously be solved to make sure every flight and passenger travels safely and efficiently. Besides the increased number of flights, there has also been an increase in the size of aircraft which in turn augments the number of passengers. These conditions could generate potential bottlenecks or congestion in the terminal buildings if the available resources are not efficiently managed. Inside the terminal buildings, they manifest as huge queues at the security filters, baggage handling systems, and check-in counters, to name but a few. In addition, they also cause excessive waiting times that the customers see as bad service levels.

For the sustainability of the Air Transportation System (ATS) all over the world, various ideas have been proposed to alleviate traffic growth and its implications such as the construction of additional runways or terminal buildings, or improved sequencing of operations in both air and land side, etc. However, more fundamental and innovative changes are required to improve the use of available air capacity.

In the case of airport terminals, analysts need to take into account not only the typical operational restrictions inherent in the system but also other measures that will make it possible to evaluate the perception of passengers, who are the main customers of the terminal and will drive the economic and social development of the system.

The International Air Transport Association (IATA) has published some guidelines for what is called as Level of Service (LOS) indicators. These measures evaluate characteristics associated with perceived comfort inside the terminal, such as available area per passenger, the speed at which the passengers can move inside the terminal, waiting times, queue lengths, etc., in different situations. Those metrics are of particular interest to airport planners which is why scientific community has focused on determining the factors that influence perception [4]. Table 1 illustrates some of these typical values suggested by IATA [5].

Optimization techniques are able to give optimal or close-to-optimal solutions to problems that are deterministic in nature; on the other hand, simulation approaches can consider the stochastic nature of the processes that participate in the system under study, while at the same time being able to describe the studied systems at different levels of abstraction. This chapter presents a methodology that combines the two approaches to generate better solutions than the ones that could be achieved by applying each technique independently. The methodology presented is applied to the check-in allocation problem for illustrative purposes but the methodology itself can be used in a wide range of problems from diverse fields to generate more robust solutions.

Table 1 Level of service indicators

m^2/Pax	A	B	C	D	E
1. Few trolleys and passengers with check-in baggage (row width 1.2 m)	1.7	1.4	1.2	1.1	0.9
2. Few trolleys and 1 or 2 pieces of baggage per passenger (row width 1.2 m)	1.8	1.5	1.3	1.2	1.1
3. High percentage of passengers using trolleys (row width 1.4 m)	2.3	1.9	1.7	1.6	1.5
4. Heavy flights with 2 or more items per passenger and high percentage of passengers using trolleys (row width 1.4 m)	2.6	2.3	2.0	1.9	1.8

The problem is tackled in the following way. First a brute-force approach is implemented to obtain initial feasible solutions taking into account quantitative restrictions such as minimum and maximum number of check-in desks per flight, load balance in the check-in islands, opening times of check-in desks, and other restrictions imposed by the LOS.

The initial solutions are then encoded as chromosome-like data structures and operations are performed in order to obtain the most promising solutions under a particular cost function. Once the initial solutions are obtained, they are in turn evaluated using a simulation model of the particular terminal under study including in the model stochastic variables that count for the passenger arrival profiles, opening times, layout of the facility, interactions between passengers, efficiencies of processes, etc. With these elements it is possible to determine which allocation is the most quality-efficient in a close-to-real scenario in order to maintain the LOS indicators at the desired level. The proposed methodology has been put into practice using information from a real terminal but it can be easily adapted to a different one with different restrictions imposed by the corresponding LOS agreements between the airport operator and the corresponding airlines.

The reminder of the chapter continues in the following way: Sect. 2 is a brief review of the principles of both approaches, simulation, and evolutionary algorithms are presented. Section 3 introduces the proposed methodology, Sect. 4 presents and discusses the check-in problem in detail, and the different steps of the methodology are described. Finally, Sect. 5 wraps up the discussion of the chapter.

2 Evolutionary Algorithms and Simulation

Evolutionary algorithms are a group of so-called metaheuristics. The authors selected evolutionary algorithms for tackling these problems because they have been widely used by scientific community and the implementation is rather easy. In the

optimization process, the cost function can be designed using different metrics which makes the model flexible. However, the solutions obtained do not represent real-life systems where random factors play an essential role to get optimal results. The drawbacks of evolutionary algorithms are overcome with the integration of simulation in the methodology. This integration leads to a more robust approach that can be easily adapted for problems in other fields by the reader.

There have been several theoretical and practical contributions to evolutionary algorithms field, as evidenced by the books, papers, and workshops proceedings published in the last years such as [6–8] or [9] among others. The use of metaheuristics such as evolutionary algorithms to solve problems of this nature has been motivated mainly because the population-based nature allows the generation of several elements of the Pareto optimal set in a single run. Evolutionary algorithms can be very useful for the selection of parameters to optimize the performance of a system. Furthermore, The choice of any decision parameters can cause the system to perform better or worse, which can be measured by some relevant objective or fitness function, as in real systems, where the interactions between the parameters are not generally amenable for analytical treatment.

In this section, a review of both approaches, Evolutionary Algorithms and Simulation are presented so that the reader can have a clear understanding of the differences and advantages between them.

2.1 Evolutionary Algorithms

The Evolutionary Algorithms (EAs) fall within the so-called population-based metaheuristics [10]. These techniques, which are considered as a general class of stochastic optimization algorithms, are employed to find optimal (or as optimal as possible) solutions to hard problems in a very wide range of areas.

Evolutionary algorithms are a group of methods inspired by the evolutionary processes found in nature, they borrow some concepts from population biology, genetics, and evolution such as inheritance, mutation, natural selection, and recombination (or crossover) to guide the search within the solution space. Detailed information can be found in [7, 10–14] to mention just some of the literatures.

The general idea behind an evolutionary algorithm is the representation of a solution to the problem in the form of a vector of decision variables. Using biological terms, the *genotype* is the encoded representation of the variables, and the *phenotype, chromosome* or *genome*, the set of variables themselves. In other words, a genotype or individual, represent a solution to the problem to be solved, and is represented by a list of parameters, also called chromosomes.

In most cases the transformation or modeling task is not simple but rather a complex one and depends on the perception of the modeler. Thus the transformation of the decision variables into a vector-like representation is per se an interesting and challenging problem. Once the decision variables have been represented in the form of a vector, the optimization problem can be specified.

Let us assume that we have a discrete search space and a function that assigns a value to each of the elements in the search space.

$$f : X \to \mathbb{R} \tag{1}$$

The general problem is to find:

$$minf, x \in X \tag{2}$$

Here x is a vector of decision variables, and represents the objective function. Such a problem is commonly called discrete or combinatorial optimization problem [7].

Basic evolutionary algorithms follow the next steps. First, an initial population is constructed where several individuals are randomly generated to form the first initial population $POP(k)$. Then each individual is evaluated, and a value of fitness is returned by a fitness function. The initial population undergo a selection, mutation, and recombination process to identify the best adapted individual. Figure 1 illustrates this transformation process.

There are different ways of selecting individuals and a very popular one is the deterministic (λ, μ)-tournament selection. This selection begins by randomly selecting λ individuals from the current population $POP(k)$. Together with the selection, a fitness measure f is performed to evaluate each individual to keep the μ best ones. Using this evaluation, solutions that have a higher value of the so-called fitness function are identified and better opportunities for further evolution are given to those solutions. These steps are repeated until a new intermediate population (POP_i) is completed. Following selection, the evolutionary operations, mutation, and recombination (crossover) can be applied to improve the original problem.

The chromosomes of the parents are mixed during crossover, hence crossover results in two new individual child, which are added to the next generation population $POP(k + 1)$. Parents and children are joined in some fashion to form a new

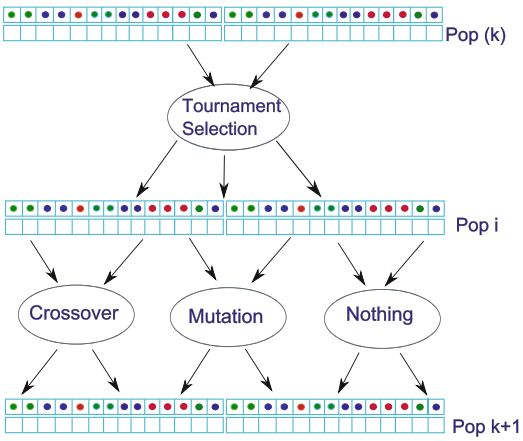


Fig. 1 Evolutionary algorithm process

next-generation population that is different from the initial generation, and the cycle continues. This generational process is repeated until a termination condition, imposed by the developer, has been reached.

2.2 Simulation

Simulation is the imitation of the operation of a real system or process over time. It is used to generate artificial history and data of a system, and for the observation and analysis of that artificial history to draw inferences concerning the operating characteristics of the real system [11].

The model usually represents a set of parameters and assumptions concerning the operation of the system. These assumptions are expressed in mathematical, logical, and symbolic relationships between the entities, or objects of interest of the system. Once developed, verified and validated, a model can be used to investigate a wide variety of "what-if" scenarios about the real-world system [15]. Potential changes to the system can then be simulated in order to predict their impact on the system's performance. Furthermore simulation can also be used to study systems in the design stage, before such systems are built based on relationships taken from other fields. Thus, simulation techniques can be used both, as an analysis tool for predicting the effect of changes to existing systems, and as a design tool to predict the performance of new systems under varying sets of circumstances. Nowadays, with the evolution of computer capacities, computer simulation is also able to develop very accurate and graphically appealing models that can represent a system at different levels of abstraction, depending on the objective of the study.

A simulation model can be developed using different tools, for example, several studies have been performed using modeling formalisms such as Coloured Petri Nets [16] or using commercial software such as SIMIO [17] or ARENA [18] in which the modeler makes use of a library of objects and just has to put them together.

Simulation alone has been proven to give good solutions in different fields such as the transport industry [19], manufacturing [20], airport operations [21], etc. However when it is used as a decision support tool it cannot ensure the best outcome since the experiments only explore a subset of the whole different configurations of the system under study and depending on the size of the model and the characteristics of the computer the number of experiments is limited to the time window for the decision to be taken.

Simulation recently has been used in combination with other techniques in order to overcome the aforementioned drawbacks. It has been used to explore scenarios more efficiently in combination with Petri nets [20] or for the evaluation of disturbances with the use of constraint programming techniques [15], just to mention a couple of examples. Thus the decision-making process supported by simulation experiments always has a certain level of uncertainty that can be minimized as more experiments are performed, however, this activity is time consuming and penalizes its potential for timely decisions over the real system.

3 Methodological Approach OPT-SIM

The methodology presented has been applied to the case study of an airport terminal with good results. Nevertheless, the methodology can be implemented in a different set of problems from the one presented in this chapter. Some typical problems that can be tackled with this approach are the analysis of counter areas such as those of banks, service areas, ports, security filters and, in general, in situations where allocation of resources must be performed and the stochasticity presented in the system cannot be included in the analytical model, which would otherwise be quite easily implemented using some mathematical programming techniques. Allocation problems in particular can benefit from a mathematical programming formulation, especially when operational or sample size constraints lead away from straightforward or closed-form solutions.

Figure 2 gives the diagram that illustrates the different steps in the methodology. At the beginning a brute-force approach or constraint satisfaction problem generates feasible solutions. The feasible solutions are encoded and are improved by the evolutionary algorithm. Then some solutions are selected and evaluated in the simulated environment, where the stochasticity is integrated to come up with a more robust solution.

The Evolutionary Approach starts by representing the problem variables in a chromosome-like structure to produce some initial feasible set of solutions for the allocation problem to feed the evolutionary algorithm. As briefly described in Sect. 2, an initial population is constructed, where several solutions are randomly generated to conform the initial population $POP(k)$. Then, the evaluation process is performed, and a value of the fitness function is returned to measure the quality of each solution to end up with n best solutions. The fitness function is designed in such a way that it

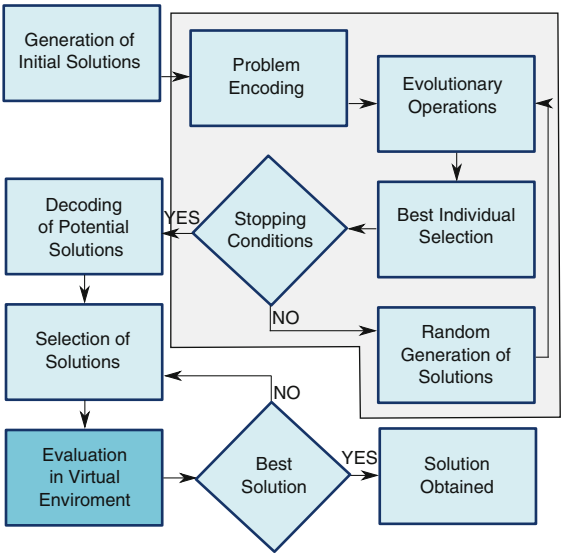


Fig. 2 OPT-SIM methodology

measures the solutions depending on the objectives of the study. The new population undergoes a mutation and recombination process. After k iterations, a pool of efficient solutions is obtained. Figure 1 illustrates this transformation process.

The next phase of the methodology integrates the stochastic factors using a simulation model, where the solutions obtained in the previous phase are further improved through experiments with the simulation model. With this approach the simulation model starts in an improved configuration, as the solutions obtained using the evolutionary algorithm are cost efficient, therefore the improvement with the simulation approach is less time consuming.

With the previous implementation, the problem is approached by taking as many characteristics as possible into account instead of restricting only to either the ones that are limited to the perception of the modeler with an analytical technique or a time-consuming analysis performed with a simulation approach.

The best solution or solutions are selected depending on how the dynamic model has performed, thus ending up with a more robust solution than the one that could have been obtained through the sole use of one technique or the other.

4 Case Study: The Check-In Allocation Problem

An airport terminal is a facility where passengers start on their journey through the air transport service. In order to board the aircraft, the passenger must undergo different processes involving management resource. First the passengers must arrive to the terminal by any mean of transport that, depending on the location, could be public transport (bus, metro, train) or private transport (private car, taxi, shuttle). Once they enter the terminal, a registration process starts.

The check-in service consists of passenger registration, commonly known as the check-in, and handing over their baggage; moreover, this is the moment when passengers get a first impression of the airport and the airline. Although this first impression is very important, there are other issues involved in the management of resources. Baggage management is also a vital issue that it must be considered to successfully monitor the proper operation of the airport and the airline. The baggage has to be transported to the right airplane using conveyor belts and trolleys.

Passengers have to go to a check-in desk or a common check-in island to get their boarding passes and to drop off their baggage. The check-in process can be either manual or automatic. Even though automatic check-in (self check-in) is growing in popularity, there are still a lot of passengers who prefer the manual process, which is normally performed by personnel provided by the airline. In some cases, personnel provided by the airport may perform this process, depending on the agreement between the parties involved. The facility and resources needed to perform the check-in activity are normally provided by the airport through an agreement that sets up the conditions and also the performance indicators (PI) that will measure the quality of service provided.

At this point, there are two opposing objectives: on the one hand, the airlines want to provide the best possible service to their customers at the least cost while, on the other hand the airport needs to provide this service with limited resources that, in the case of a check-in area, means the available check-in desks and personnel. Furthermore, due to increased traffic and the consequent large inflow of passengers and baggage heading for many different flights, the check-in allocation problem has been gaining importance in the relevant literature.

An inefficient management of resources such as ground services, personnel, desks, filters, etc., are appreciated as congestion in terminals. The congestion in turn can be appreciated in several points throughout the passenger boarding process; for example, in the check-in desks, the security filters, passport control, and sometimes at the boarding gate. These problems have been traditionally faced by the aviation industry through the increase of physical resources (e.g., increase in the check-in desks, increase in the number of security filters, etc.). Furthermore the increase in competition between airlines and airports has forced both actors to optimize their resources at hand in order to reduce their costs and keep competitive. On the other hand, the increase of passenger traffic in airports makes necessary the development of novel decision support tools that take into consideration all the different elements that are involved in the system and influence the correct allocation of resources.

Furthermore the increase in traffic caused mainly by the competition between airlines, market liberalization, and the increasing number of low-cost airlines will force the need of efficient strategies and procedures for allocating the resources inside terminals if the LOS are to be maintained.

The general check-in allocation problem consists of allocating the available desks of a terminal in such a way that the allocation satisfies a series of restrictions imposed by the airport and the companies through a service contract. These restrictions may change depending on the airport, the airlines, the region it serves, and the type of terminal [22].

The check-in allocation problem is a well-known problem in airport terminals that has been studied by some authors using evolutionary approaches or mathematical formulations [23–25]. These techniques have the drawbacks that do not take into account all the different elements in the terminal's check-in area or the interactions of passengers inside the terminal with each other or with other elements of the facility.

The methodology presented in this chapter deals with the problem of performing the allocation of check-in desks in a terminal, considering not only the internal policies and quality indicators but also the interactions between passengers and the physical facilities of a terminal. The various rules and information data were provided by a terminal in the Middle East under a confidentiality agreement. We shall refer to this airport, when applicable, as "the airport."

4.1 Literature Review

The check-in allocation problem has received little attention in the literature over the last few decades, but in recent years , because of the increasing traffic demand, special attention has been placed on this process due to its economic and time importance. It has been defined and studied using different modeling techniques and methods for its resolution.

Parlar and Sharafali [26] use a dynamic programming approach for the check-in allocation problem assuming that it is possible to close or open counters depending on the demand. This practice is efficient from the passengers' point of view as the perceived quality of service is high in comparison to a static way of managing the counters but it is difficult for some companies and the airport to count on having extra counters when the demand is at its peak.

For these kinds of resolution, the parties have different objectives; the airlines want to minimize their use, while the airports want to maximize their use. On the other hand, although the trend in airports now is to use paperless tickets and self check-in kiosks, this is not the case in many of the airports around the world, especially in the growing Asia-Pacific region where they are used to more personal treatment. Besides, some developing regions, such as Latin America or Africa, still depend strongly on the manual check-in process.

Another similar paper by Littler and Whitaker [27], provides a procedure for estimating staffing requirements to meet a preset processing time target. It uses a stochastic simulation of passenger arrivals at the terminal. This paper mainly focuses on the design phase of a managerial schema for the use of check-in counters.

Another paper on this problem in the Hong Kong Airport was presented by Chun and Mak [28]. Their work combines simulation with an allocation system, taking restrictions and desires of the companies at the airport into account. They use a simulation-based optimization approach to determine the best check-in allocation, considering the stochasticity of some of the processes involved, such as service and arrival rates, thus evaluating the fitness of the solutions by analyzing the efficiency of processing passenger by predicting queue lengths.

More recently, Park and Ahn [29], revisited the problem of passenger arrivals at Gimpo airport to determine the most appropriate number of check-in counters, again this work focuses on the sizing of the resources with a view to a particular objective.

With a different scope, Yan et al. [30], provide an integer programming approach to the assignment of common-use check-in counters for a model of Taipeis International Airport. However, due to some size limitations they had to come up with a heuristic method to solve the model. This work aims at assigning the allocation of common-use counters in a close-to-optimal way.

van Dijk and van der Sluis [31] present a paper that is connected with the work presented here; the authors use simulation to determine the minimum numbers of desks and then an integer programming approach to optimize, as much as possible, the resources spent on allocating passengers. The work presented by these authors analyzes the problem from the passengers' point of view, with an eye to shortening

service times. It only considers some of the restrictions that are presented in this chapter.

On the other hand, the problem presented by Hsu et al. [24] deals with the dynamic allocation of the check-in services required by passengers aiming at reducing the total time the passengers spend on the check-in procedure. The assumptions made by the authors are from the standpoint of the passengers, so the model could be used for assigning the steps the passengers need to follow in order to minimize their processing time; some assumptions, such as an average processing time, are made, thus discarding the inherent stochasticity of this operation. Their approach uses th average values and clustering the passengers.

Very recently Castillo-Manzano and Lopez-Valpuesta [32] analyze the check-in problem from the sociodemographic factors that influence passengers the decision of using one type of check-in facility over another (e.g., check-in counters or common-use self-service desks).

The work presented in this chapter differs from the aforementioned review in the sense that it performs the allocation of check-in counters through an approach that combines an analytical approach using evolutionary algorithms with simulation that allows modeling the dynamic and stochastic characteristics of the system under study. The combination of a deterministic solution with the stochastic elements stressed in the simulation model provides a more robust and reliable solution than the one that could be achieved with the sole use of the evolutionary or the simulated approach. The applicability of this approach is validated by using information from a real airport terminal provided under a confidential agreement between the authors and the airport.

4.2 Technical Approach

The problem consists of performing the check-in desks allocation satisfying a series of rules provided through a contract between the airlines that use the counters and the airport. These rules are classified as hard or soft.

The *hard rules*, i.e., rules that cannot be broken under any circumstances must be satisfied when the allocation is performed. On the other hand, it is desirable for *soft rules* to be satisfied but they can be broken when there is no other available option. It is important to mention that the violation of soft rules would impact on the perception of quality by the passengers inside the terminal, so it is an important factor to be taken into account by the company that performs the allocation.

4.2.1 Hard Rules

The following are the mandatory rules:

1. Overlap Verification: The current allocation must always be aligned with the allocation of the previous month in order to avoid having allocated the same check-in desks or nearby for the last flights of the previous month and for the first flights of the current month.
2. Balanced Loads: Allocating flights to check-in counters will consider an aspirational usage of 20 % for each of areas A, B, C, D, and E. The acceptable deviance is 1 % on a daily basis and 5 % on each 2-hours window. This restriction means that the terminal is used in a balanced way, therefore the usage of the areas is maximized with an increase of the passengers perception of quality. Figure 3 illustrates the different zones of the terminal area under study.
3. Number of Desks: Allocating flights to check-in counters will consider a minimum standard of one check-in counter per 45 passengers. This restriction has been defined to provide a good quality of service to the passengers at the check-in process, but a higher limit of 5 counters per airline will also be established.
4. Sorting Hall: Since there are two different sorting areas, the airport authority has established that the allocation of check-in desks should be done in such a way that the allocation of check-in desks should be done in such a way that no desks for the same flight use different sorting lines, as there is a high risk of lowering the efficiency of the baggage transport process and also of ending on a different flight. Figure 4 is a diagram of the two baggage conveyors systems. In this system

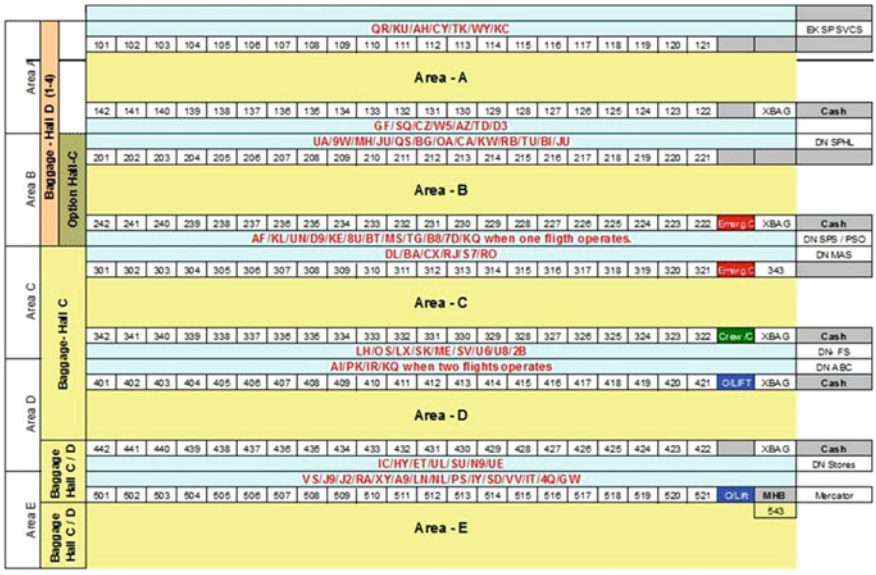


Fig. 3 The check-in zones in the terminal under study

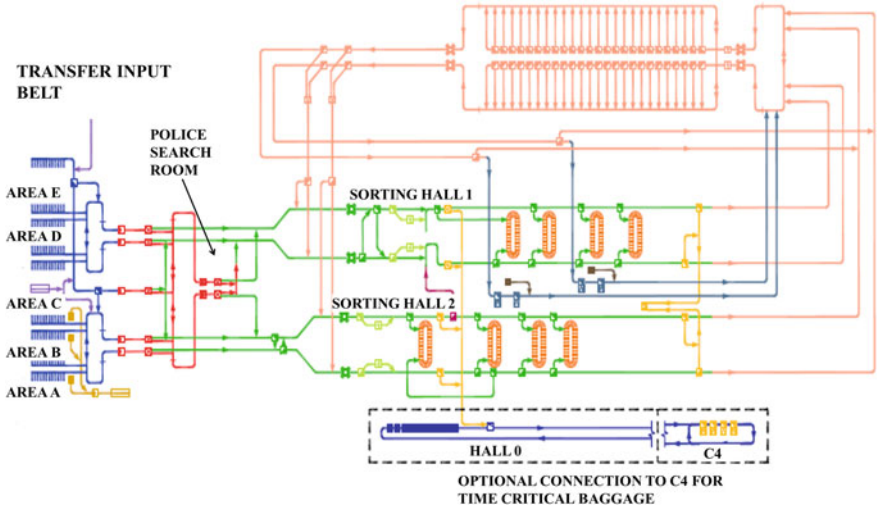


Fig. 4 Layout of the baggage handling system inside the terminal

areas A and B are processed in a different hall from areas C, D and E, therefore allocation must consider the boundary between these areas in order to avoid the allocation of a flight that could use both baggage systems.

4.2.2 Soft Rules

These rules are directly associated with the quality perceived by passengers; therefore in order to get a good evaluation of the quality of service, these rules should be largely satisfied.

1. Optimized Queuing/Circulation Areas for heavy flights: The allocation will avoid placing more than 3 heavy flights on the same row island during any given one-hour window. If this restriction is not satisfied, there is a high risk of congestion inside the island or row with the corresponding perception of poor quality on the part of the passengers.
2. Optimized Queuing/Circulation Areas for any flights: The allocation will avoid placing 2 flights on consecutive counters in any given one-hour window. The recommended practice is to leave at least one counter free between two flights for reasons of redundancy and flexibility. This practice is common in the terminal but the airport authority claims that better ways of allocation should be explored in order to make more efficient use of all the available resources, as this practice reduces the capacity of the check-in resources.
3. Airline Preferences: The preferences of the airline for their flights to be allocated to specific rows or fixed desks are officially taken note of during meetings and

the solution will consider their requests, but only after complying with all the allocation rules. It is desirable and common practice for companies from the same alliance to be allocated in the same area most of the time, therefore an efficient allocation will give priority of use to those airlines belonging to alliances inside the terminal islands.

4.3 Constraint Satisfaction

The first step of the methodology is the constraint satisfaction problem. This problem performs a static allocation based on the flight plan provided by the airport. The allocation algorithm performs an allocation of the planned flights taking into account the following constraints:

1. There is no overlap between flights
2. Counters are opened 3 h in advance
3. It calculates the number of counters needed in a base of 45 pax/counter
4. It leaves one check-in desk in between flights
5. The flights are allocated in the corresponding sections of the check-in area, so that the baggage does not end up in a different baggage hall
6. It randomly allocates check-in desks, trying to distribute the flights uniformly (load balance).

The allocation for the flights is performed sequentially in time slots, taking into account the aforementioned constraints as it is done in common practice. It takes every flight at a time and looks for the corresponding available time slot that satisfies the restrictions and, once allocated, it continues with the next flight on the allocation list. After all the flights are allocated an initial solution is obtained.

In order to get a variety of solutions for the evolutionary algorithm, a random selection of flights is performed every time the allocation algorithm is run. Using this approach a population of initial solutions is generated.

Once the initial solutions are generated, the next challenging task is the transformation of the solutions into vectors with the information that will be used by the evolutionary algorithm for improving the initial solutions.

4.4 Chromosome Representation

One of the key tasks in this approach is the proper representation of the solutions in the form of a vector of information. The representation will significantly influence the performance of the evolutionary algorithm. Every field of information holds key information that is useful for the performance of the operations of the evolutionary algorithm. In this chapter, the vectors have been defined as follows:

1. ID: Flight Identifier (string)
2. CI_OT: check-in Desk Opening time (min)
3. CI_CT: check-in Desk Closing time (min)
4. I_C: Initial check-in Counter (integer)
5. F_C: Final check-in Counter (integer)
6. Soft: check-in Allocation Soft Rules (Integer)
7. Hard: check-in Allocation Hard Rules (Integer)

Field 1 holds the Identifier of the corresponding flight. This field is used for keeping track of the corresponding flight. Fields 2 and 3 refer to the time the check-in counters are open and closed, respectively. Fields 4 and 5 provide the information about to which desk numbers are used. The last two fields (Fields 6 and 7) hold the information of the number of check-in desks needed to satisfy the hard and soft rules. They are necessary to identify if a counter is left or not in between flights.

4.5 Crossover Operations

Crossover is the main operation used for improving the current solutions. Crossover is performed in such a way that the feasibility of the new generated solution is maintained.

The crossing will be performed between chromosomes or elements of two current solutions (SolA and SolB) and it will perform the crossing between pairs. Figure 5 illustrates the crossing process.

The crossing is performed in the locus associated with the check-in desks being used, *I_C* and *F_C*, see Fig. 5. The reason is that the timeslots where the check-in is performed cannot vary, therefore only the counters will be the ones that will give variability to the generated solutions. The light blue color elements of the solutions (offspring) are the ones that have been changed by the crossover operators.

In order to maintain consistency in the generated solutions, the algorithm will verify three aspects of the new solution:

1. The crossing is performed between pairs that must use the same timeslot:

$$CI\_OT_A = CI\_OT_B \tag{3}$$

$$CI\_CT_A = CI\_CT_B \tag{4}$$

In order to choose a proper candidate for the crossing, the algorithm will take one element of the *SolA* and randomly choose another one from *SolB*. After the selection of the element of *SolB* it will verify that its timeslot corresponds to the same one as the element from *SolA*. If that is not the case it will take another one until a feasible one is found.

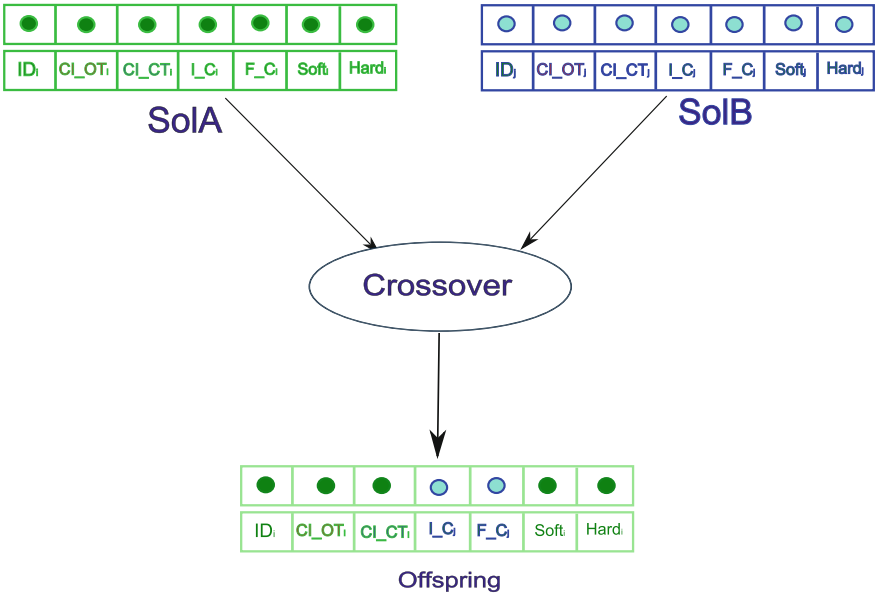


Fig. 5 Crossover between solutions

2. The crossing does not hinder previous or future flights.
   It may be the case that the previous restriction is satisfied but the time and desks used for the allocation overlap previous or future allocations. So in order to avoid this situation the algorithm checks that this situation is avoided. Figure 6 illustrates the potential conflicts that may be encountered when the crossover is performed.

This example is schematized in Fig. 6. The counters of *Flight1* from *SolA* are swapped with the ones from *Flight4* of *SolB* as both flights have the same check-in time window. If we focus on the new *SolB* generated (right-hand side of Fig. 6), we can see that *Flight4* is in conflict with *Flight2*, while in the case of *SolA* there are no conflicts.

In order to avoid these types of conflicts, a procedure has been coded for the crossover operation. It compares the allocation performed against all the different elements of the current solution that fall within a time window of [open_time − 180, open_time + 180]. This comparison ensures that the previous allocations do not conflict with the current allocation, and at the same time that the timeslot of the present allocation does not conflict with a future one. If the allocation is conflict free, then the allocation is allowed and performed. However, if the time windows overlaps (a potential conflict) with some allocation, then the used check-in desks are verified for overlapping, if there is no conflict then the allocation is allowed. On the other hand when both conditions occur (time conflict and desks used overlapped), the crossing is not performed.

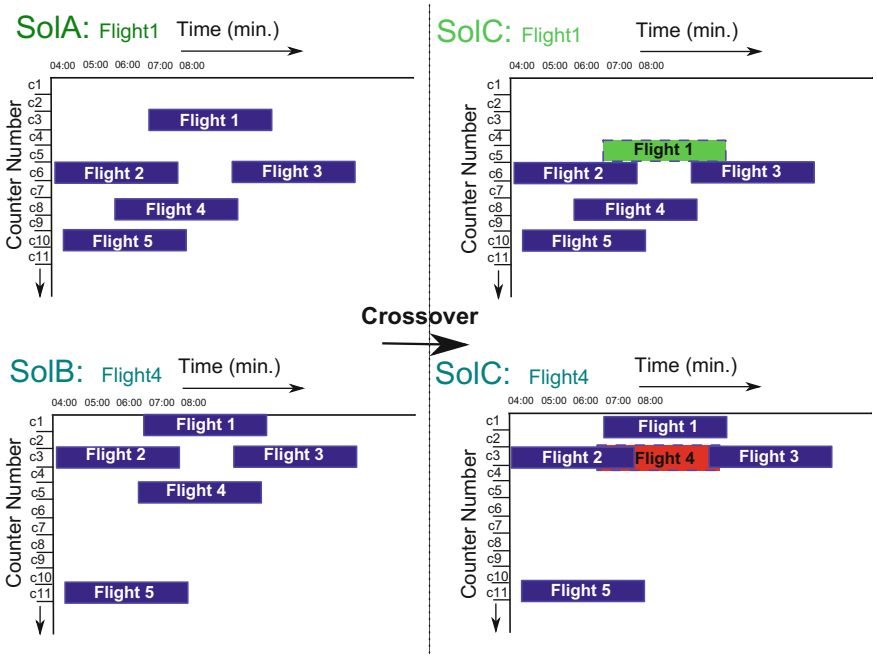


Fig. 6 Potential conflicts

3. Hard rules consistency
   Finally, the crossover will evaluate if the allocation does not violate the hard rules concerning the number of desks needed (*Field7*). If the new allocation does not violate the minimum number of desks needed it will be kept as a feasible allocation. If the solution does not leave a desk in between flights, it will be kept as a feasible solution and later on this situation will be evaluated to see whether or not it affects the fitness of the new solution.

4.6 Objective Function Evaluation

During the performance of the evolutionary algorithm, once a feasible solution is generated, it is evaluated on a static basis using an objective function. This function evaluates the fitness of the solution by calculating several factors that make up the final value assigned by the function. These factors must have a direct impact on the LOS indicators.

The function used by this approach uses four parameters but it is not only restricted to those values. The analyst could extend the formula to include more parameters depending on the particular case of the airport in question. For the case presented

in this chapter, the function $F(v1, v2, v3, v4)$ is made up by a linear form of the 4 parameters:

$$F = a_1 v1 + a_2 v2 + a_3 v3 + a_4 v4 \tag{5}$$

where:
v1: is a factor that measures the number of flights in the solution that does not respect the 1-check-in desk in between flights.
v2: is a factor that evaluates the balance loads for the solution in accordance with the policies imposed by the airport.
v3: evaluates the number of heavy flights in the same island during 1-hour window.
v4: evaluates the distribution of flights in the islands of the check-in area.
a_n: is the weight of the corresponding factor.

In this case study of the different weights were kept at the value of 1 for illustrative purposes but the parameters can be changed to assign a different priority to one or several parameters over other ones. These priorities would drive the selection of the different feasible solutions, depending on the airport's requirements.

The evolutionary process is performed using the values of the objective function to calculate the goodness-of-fit of the different allocations and the selection process is carried out based on those values. Using this approach, the solutions are incrementally improved until a stop condition is satisfied. This condition is determined arbitrarily.

4.7 Simulation-Based Improvement

A pool of potential solutions for the allocation problem is obtained from the evolutionary algorithm to be later tested using a simulation model of the facility under study. This evaluation will provide a better estimation of the quality levels that can be achieved in the real system.

It is important to mention that certain requirements are desirable for the simulator in order to have the best evaluation of the quality indicators, i.e.:

- Agent-based so that the interaction between entities is more approximated to reality.
- High-description level; the more accurate the better is the evaluation.
- The model must allow interaction between agent–agent and agent–objects, so that it is possible to determine differences between relative positions among agents and objects.

There are some simulators in the market that satisfy these requirements [17], thus the methodology can be implemented making use of the one that suits best the objective of the study.

The use of the simulated scenario makes it possible to test the potential best solutions in a close-to-real environment. Sometimes it happens that solutions do

Table 2 Departure flight schedule

Airline	Passenger	Departure flight	Departure time
AIC	90	AI 0975	7:10
UAL	220	UA 0807	8:00
CHH	100	CH 9999	8:20
RNA	110	RN 0604	8:00
PIA	120	PI 0414	7:30
AFR	130	AF 8866	10:00
CSN	140	CS 0582	7:00
DLH	90	DL 0853	11:00
KLM	90	KL 0815	10:00
ABQ	90	AB 7777	12:00
SAI	90	SA 0570	14:00
NAX	90	NA 0835	14:00
SWR	285	SW 0847	10:20
ROT	306	RO 0705	18:00
BBC	120	BB 8888	22:00
AUA	116	UA 0221	21:10
KQA	314	KQ 0432	18:30
BAW	206	BA 0530	15:40
AFL	337	AF 0650	21:20
KLM	120	KL 0814	15:20

not perform well in the real system once they are implemented. The latter could be caused by some obstacles present in the facilities (e.g., big columns, trolley stations, etc.) that cause a potential good solution not to be such in reality because of congestion generated by pax–pax or pax–object interaction. Other causes are the emergent dynamics due to interactions of the entities and these can be easily observed in a terminal during congestion or during a disruption.

This methodology has been used to develop an initial solver for the check-in desk allocation for the airport. The initial approach will be used to evaluate the feasibility of the approach and, once it has been validated as a decision support tool, it will be extended to an operational level.

An initial flight plan has been used for testing the approach and its implementation in the simulated environment. Table 2 presents the flight plan used for the example presented here.

The titles of the columns are self-explanatory. Although Table 2 was not the actual flight plan, it is sufficient to test and validate the approach presented in the work.

4.8 Initial Solution

The set of initial solutions is generated for hundred desks as it has been explained in Sects. 3 and 2.1. A graphic representation of the solution on one airplane is depicted

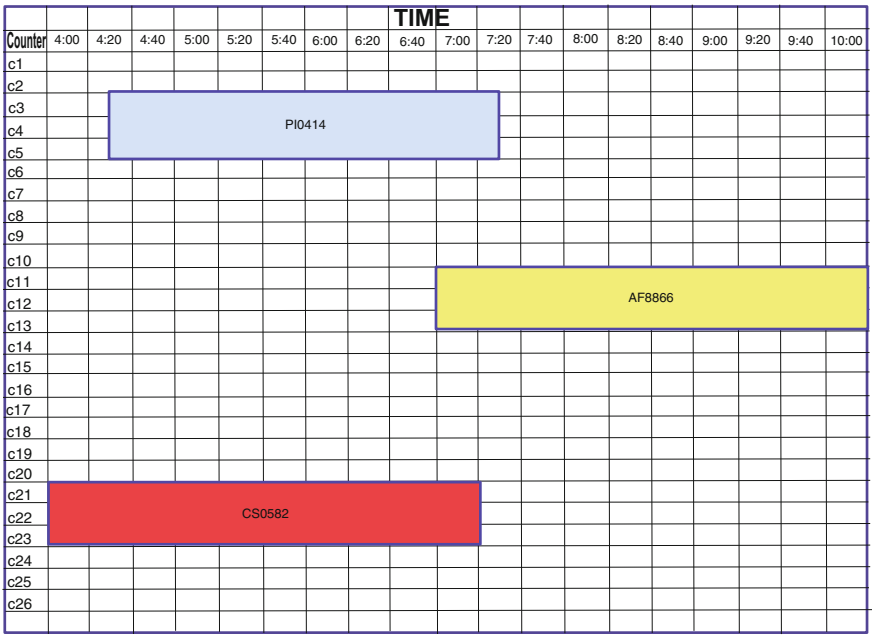


Fig. 7 Partial representation of an initial solution

in Fig. 7. The horizontal axis represents time and the vertical one represents the check-in desks used. Figure 7 exemplifies the initial solutions for 3 flights. The main outcome from the initial phase is a population of feasible solutions that are generated considering the different requirements for the allocation problem in question.

4.9 Chromosome Encoding and Evolution

The solutions generated in the initial phase are encoded as explained in Sect. 4.9 to start the evolutionary algorithm.

Table 3 gives an example of one encoded solution. The first column entitled *Flight* gives information about the Flight number to be allocated. The *PAX* column provides the information about the number of passengers registered for the flight. The following two columns, called *DeskIN* and *DeskEND*, store the information about desks that have to be allocated for the corresponding flight. The last two columns, *OpenTime* and *CloseTime*, give the information for the opening and closing time of the corresponding flight. For example, the first row means that flight *CS0582*, which has 140 passengers, will use the desks from 21 to 24 and they will be open from 4:00 am until 7:00 am.

Table 3 Encoded solution

Flight	PAX	DeskIN	DeskEND	OpenTime	CloseTime
CS 0582	140	21	24	4.00	7.00
AI 0975	90	61	63	4.17	7.17
PI 0415	120	1	3	4.50	7.50
UA 0807	220	71	75	5.00	8.00
RN 0604	110	91	93	5.00	8.00
CH 9999	100	81	83	5.33	8.33
AF 8866	130	11	13	7.00	10.00
KL 0815	90	41	43	7.00	10.00
SW 0847	285	84	88	7.33	10.33
DL 0853	90	31	33	8.00	11.00
AB 7777	90	51	53	9.00	12.00
SA 0570	90	61	63	11.00	14.00
NA 0835	90	71	73	11.00	14.00
KL 0814	120	51	53	12.33	15.33
BA 0530	206	31	35	12.67	15.67
RO 0705	306	91	95	15.00	18.00
KQ 0432	314	21	25	15.50	18.50
UA 0221	116	11	13	18.17	21.17
AF 0650	337	41	45	18.33	21.33
BB 8888	120	1	3	19.00	22.00

For each flight, whether or not they respect the check-in desk in between flights is evaluated. The number of check-in desks open is calculated based on the rule that one check-in desk must be open for every 45 passengers. As an example, see the initial flight on Table 3, for flight *CS0582* with 140 passengers, for this rule to be respected three check-in desks should be opened.

The balance load in each area A, B, C, D, and E is also calculated for the solutions. This balance load depends on the layout of the airport and the internal policy of the airport authority. The computation divides the check-in desks into zones and then calculates how many flights have been allocated to which zones.

The third parameter needed to compute the objective function is the one for heavy flights. In this case the zoning is performed based on the time of the allocation and the correspondent zone, this way it is possible to penalize the allocation of heavy flights in the same island during a time period.

The fourth value is calculated taking into account the number of flights in each island, so that the flights are evenly distributed. Thus, the space in between flights is optimized and the level of service improved.

Using the objective function, the selection among the siblings solutions is made in such a way that the value of the objective function is progressively improved.

Table 4 Values of the cost function

Iterations	Avg. cost	Value
1	1.36	
10	1.012	25.58823529
20	0.6714	50.63235294
30	0.468	65.58823529
40	0.4228	68.91176471
45	0.3542	73.95588235
60	0.15068	88.92058824
624	0.145276	89.31794118
891	0.13397	90.14926471
3858	0.126434	90.70338235
6605	0.125325	90.78492647
6914	0.112363	91.73801471
31,600	0.11074	91.85735294

Table 4 illustrates how the objective function is improved as the algorithm evolves. Meanwhile, Fig. 8 illustrates the evolution of the cost function versus the number of iterations.

4.10 Performance Evaluation in a Virtual Environment

The model of the terminal area has been developed using a general-purpose simulation software called SIMIO [17]. The simulator has been selected for these studies as it possesses most of the necessary characteristics previously mentioned. There are

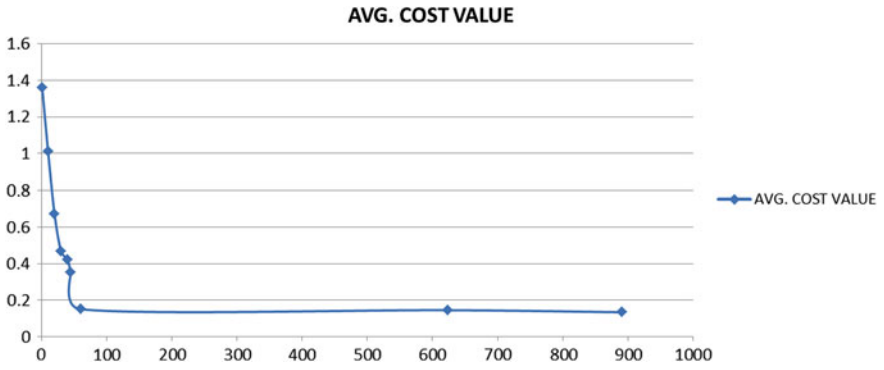


Fig. 8 Convergence of the cost function

other tools that possess better characteristics of agent interaction but for the sake of illustration the authors consider that SIMIO is good enough for evaluating different interactions. However, if the reader wants to emphasize in the interactions, other tools could be more suitable.

The simulation model represents the layout of the area under study, which comprises 100 check-in desks in an area of 170 × 70 square meters (see Fig. 9). With the help of the virtual environment, the LOS can be evaluated alongside other performance indicators that are important for assessing the correct management of the area under study.

Figure 9 presents the simulated layout of the terminal area together with some data and notation below the model. These notations illustrate how the different sections of the terminal have been identified in order to dynamically calculate the LOS over time. In the terminal area there are 5 sections, namely areas A, B, C, D and E; with 5 subsections each. Therefore, a total of 25 areas were monitored in the study.

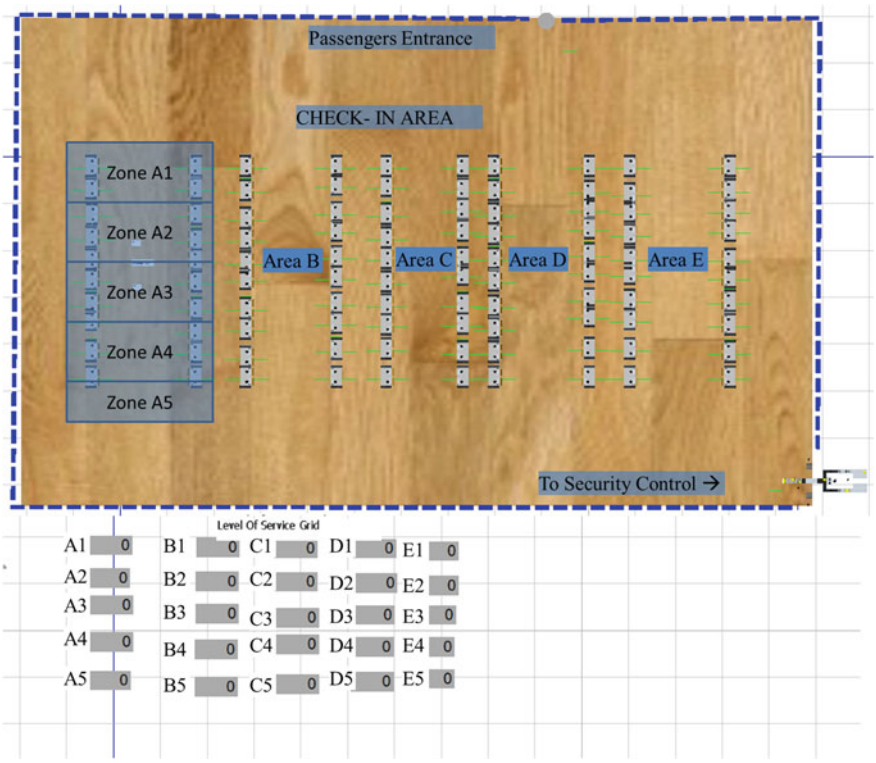


Fig. 9 The simulated environment

4.10.1 Stochastic Parameters

The advantage of using a simulation model for assessing the different solution, is that it is possible not only to test the proposed allocations in a close-to-real scenario but also different characteristics of the passengers can be added to the model, thus making the evaluation more reliable than the evolutionary algorithm by itself. Some examples of the dynamic parameters that have been added to the model are presented in Table 5. These parameters are just an example of common values, however, the reader should perform a data acquisition exercise to come up with the right values for its particular study.

Different configurations provided by the allocation algorithm have been evaluated using the simulation model. Table 6 is useful to illustrate the differences between the initial allocation and the final one.

It was considered that zones with congestion problems where those areas with values under 3 m^2/pax to be considered as critical. As presented in Table 6, initially, there were several areas where congestion could be perceived as the LOS indicators illustrate; namely areas E1, C1, B1, and E2. This configuration presents the worst LOS value in area E1, with a value of 1.1113 m^2/pax. When the allocation provided by the evolutionary algorithm is evaluated in the simulation environment, the LOS

Table 5 Parameters of the simulation model

Concept	Type
Check-in desks processing time	Lognormal (0.709, 0.154) with a minimum of 1
Arrival profile	Triangular (−180, −90, −40) min
Passenger speed	Uniform (0.4, 0.9) m/s
Check-in area	$170 \times 70\,m^2$
Passengers desk	1 person/desk
Passengers do not show earlier than 3 h prior to departure	

Table 6 Level of service indicators

	Critical zones	Minimum value m^2/pax
Initial		
	Zone E1	1.1113
	Zone C1	1.6585
	Zone B1	1.8214
	Zone E2	1.9767
Final		
	Zone E1	2.3415
	Zone D1	2.1223
	Zone C1	2.5617

indicators show a significant improvement, see Table 6. For the new allocation, there were only three critical zones and the minimum value was perceived in zone D1, with a value of 2.1223 m^2/pax. In conclusion, with the use of the evolutionary approach an aoptimized allocation for desks was found. Due to this, the congested areas in the terminal could be reduced, thus providing a better allocation than the one that can be achieved by manual allocation.

5 Discussion

The present work introduces a new methodology that combines an evolutionary approach with simulation to perform the check-in desk allocation for optimizing the LOS indicators in an airport terminal. The strength of the methodology lies in tackling the problem in such a way that it possible to take into account deterministic and stochastic characteristics resulting in a more robust and reliable solution. The algorithm uses an evolutionary approach to improve the initial allocation of check-in desks taking into account the policy restrictions imposed by the airport. Once good solutions from the mathematical standpoint are obtained, they are further improved using a simulated environment that takes into account other elements of the problem such as physical locations, queue policies, passenger arrival profiles, efficiency of the personnel, etc.

The results show that the methodology is robust enough to provide good solutions with few iterations and the reliability of the solutions is improved with the simulated model. In addition the methodology is flexible enough to include more constraints either in the evolutionary algorithm or the simulation model in order to provide solutions that are in line with the objectives of the airport. The methodology that has been presented can be easily implemented in other terminals or in other industries following the guidelines and suggestions presented here but the simulation model must be developed for the corresponding terminal. In future implementations a metamodel can be integrated into the evolutionary algorithm in order to develop a stand-alone tool for decision-making. This methodology would be recommendable for the planning phases of new passenger terminals or for assessing the current performance and to evaluate future implementations that involve policy restrictions.

Acknowledgments The authors would like to thank the Aviation Academy of the HvA, the Mexican Council for Science and Research (CONACYT) and the Popular Autonomous University of Puebla for their support for this work.

References

1. BOEING. Current market outlook 2013 2032. commercial airplanes market analysis, 2013.
2. EUROCONTROL. Challenges of growth: summary report, 2013.
3. AIRBUS S.A.S. Global market forecast. future journeys 2013 2032, 2013.

4. A.R. Correia, S.C. Wirasinghe, and A.G. Barros. Overall level of service measures for airport passenger terminals. *Transportation Research Part A: Policy and Practice*, 42(2):330–346, 2008.
5. R. Neufville, A.R. Odoni, Belobaba P.P., and T.G. Reynolds, editors. *Airport Systems: Planning, Design, and Management*. McGraw-Hill Education: New York, Chicago, San Francisco, Lisbon, London, Madrid, Mexico City, Milan, New Delhi, San Juan, Seoul, Singapore, Sydney, Toronto, 2013.
6. S. Delahaye, D., Puechmorel. *Modeling and Optimization of Air Traffic*. Wiley-ISTE, 2013.
7. F. Glover and G.A. Kochenberger. *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2003.
8. KingLoong Shiu and K.Y. Szeto. *Self-adaptive Mutation Only Genetic Algorithm: An Application on the Optimization of Airport Capacity Utilization*, volume 5326 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008.
9. Thomas Black. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Oxford, UK, 1996.
10. D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.
11. M. Affenzeller, S. Winkler, A. Beham, and S. Wagner. On the influence of selection schemes on the genetic diversity in genetic algorithms. In Roberto Moreno-Daz, Franz Pichler, and Alexis Quesada-Arencibia, editors, *Computer Aided Systems Theory - EUROCAST 2009*, volume 5717 of *Lecture Notes in Computer Science*, pages 777–784. Springer Berlin Heidelberg, 2009.
12. J.R Koza. *Genetic Programming*. MIT press, 1992.
13. Z. Michalewicz. *Genetic algorithms + Data Structures = Evolution Programs*. Springer-verlag, 1992.
14. H.P. Schwefel. *Evolution and Optimum Seeking*. Wiley, New York, 1995.
15. J. Banks, Carson J.S., and B.L. Nelson. *Discrete Event System Simulation*. Prentice Hall, Upper Saddle River, NJ, 1996.
16. K. Jensen and L.M. Kristensen. *Coloured Petri Nets: Modeling and Validation of Concurrent Systems*. Kluwer Academic Publishers, 2009.
17. SIMIO. Simio homepage, 2002.
18. ARENA. Arena homepage, 2002.
19. F. Longo. Advances of modeling and simulation in supply chain and industry. *Simulation*, 87(8):651–656, aug 2011.
20. Juan Ignacio Latorre, Emilio Jiménez, and Mercedes Pérez. The optimization problem based on alternatives aggregation petri nets as models for industrial discrete event systems. *Simulation*, 89(3):346–361, March 2013.
21. S. Yan, C.Y. Shieh, and M. Chen. A simulation framework for evaluating airport gate assignments. *Transportation Research Part A: Policy and Practice*, 36(10):885–898, 2002.
22. R.E. Kazda, A.; Caves. *Airport Design and Operation*. Emerald, Inc., 2007.
23. B. Giuseppe and A. Genovese. A mathematical model for the optimization of the airport check-in service problem. *Electronic Notes in Discrete Mathematics*, 36(0):703–710, 2010. {ISCO} 2010 - International Symposium on Combinatorial Optimization.
24. C.I. Hsu, C.C. Chao, and K.Y. Shih. Dynamic allocation of check-in facilities and dynamic assignment of passengers at air terminals. *Computers & Industrial Engineering*, 63(2):410–417, 2012.
25. M. Parlar, B. Rodrigues, and M. Sharafali. On the allocation of exclusive-use counters for airport check-in queues: static vs. dynamic policies. *OPSEARCH*, 50(3):433–453, 2013.
26. M. Parlar and M. Sharafali. Dynamic allocation of airline check-in counters: A queueing optimization approach. *Management Science*, 54(8):1410–1424, 2008.
27. R.A. Littler and D. Whitaker. Estimating staffing requirements at an airport terminal. *Journal of the Operational Research Society*, 48(2):124–131, 1997.
28. H.W. Chun and R. W.T. Mak. Intelligent resource simulation for an airport check-in counter allocation system. *Trans. Sys. Man Cyber Part C*, 29(3):325–335, August 1999.

29. Y. Park and S.B. Ahn. Optimal assignment for check-in counters based on passenger arrival behaviour at an airport. *Transportation Planning and Technology*, 26(5):397–416, 2003.
30. S. Yan, C.H. Tang, and M. Chen. A model and a solution algorithm for airport common use check-in counter assignments. *Transportation Research Part A: Policy and Practice*, 38(2): 101–125, 2004.
31. N.M. van Dijk and E. van der Sluis. Check-in computation and optimization by simulation and ip in combination. *European Journal of Operational Research*, 171(3):1152–1168, 2006.
32. J.I. Castillo-Manzano and L. Lopez.-Valpuesta. Check–in services and passenger behaviour: Self service technologies in airport systems. *Computers in Human Behavior*, 29(6):2431–2437, 2013.

Part III
Transportation Case-Studies

Simulation and Optimization of the Pre-hospital Care System of the National University of Mexico

Idalia Flores De La Mota, Alexander Vindel Garduño and Esther Segura Pérez

Abstract This chapter presents two operational research techniques, simulation and integer programming, that we used to find a better ambulance location solution and shorten ambulance response time in the main campus, (Ciudad Universitaria) of the Universidad Nacional Autónoma de México, México City. Toregas' integer programming model, known as the maximal covering model, is best suited for an approach to the needs of the problem; despite its age, it has proven to be a simple and efficient model. For this job, we not only employed the location model, but also linked it to a simulation model whose function was to identify the stochastic demand and analyze the results of the model so as to find the best possible solution, within the limits set when creating different scenarios; and, at the same time shorten the response time.

1 Introduction

Records of the treatment of injured or sick patients go back to biblical times. During the eighteenth and nineteenth centuries, different methods were used with this purpose, though it was Jean Dominique Larrey who started the first pre-hospital care system.

The National Autonomous University of Mexico (Universidad Nacional Autónoma de México UNAM) has been offering this service since 1982, using vehicles with enough capacity for a multidisciplinary team made up by both professionals and technicians to provide basic and advanced life support. Despite the fact that some studies have been carried out to improve the efficiency in patient's transport times in

I.F. De La Mota (✉) · A. Vindel Garduño
Facultad de Ingeniería, Universidad Nacional Autónoma de México, Mexico City, Mexico
e-mail: idalia@unam.mx

A. Vindel Garduño
e-mail: alexander.vindel@gmail.com

E. Segura Pérez
Instituto de Ingeniería, Universidad Nacional Autónoma de México, Mexico City, Mexico
e-mail: ESeguraP@iingen.unam.mx

M. Mujica Mota et al. (eds.), *Applied Simulation and Optimization*,
DOI 10.1007/978-3-319-15033-8_8

the APH (Pre-hospital care), there is as yet little evidence of any careful study having been made on this subject either in Mexico or elsewhere. The efficiency of the APH system is measured by the system's average response times, which is the time taken by the emergency medical technicians to arrive at the scene of the accident, attend to the patient and transfer him or her to hospital if necessary.

In order to meet the Mexican official standards for quality and improve its services, a group of professionals from the Department of Operations Research were asked to make a diagnosis on the performance of this center. On the basis of this diagnosis, the operations research group developed and applied quantitative planning techniques to measure the quality of the infrastructure available, pre-hospital service, and emergency response capacity, among other performance criteria. This chapter is organized as follows: Section 2 is about concepts and a classification of location problems. In Sect. 3 a state of the art is presented with some historic data about the problem and recent published papers. Section 4 is about simulation, how it works, and why we need it. Section 5 is the description of the problem, and followed by Sect. 6 with the methodology that was used and the proposal that was drawn up to achieve the solution. Finally in Sect. 7 we discuss the proposed solution and some experiments are shown and explained, ending with some discussion and conclusions aimed at identifying this chapter's contribution and; in particular, the advantages of using these two operational research tools.

2 Location Problem

This section covers the problem of finding the best location for ambulances in order to optimize the service time. In this vein, in the state of the art, the literature on the ambulance location problem is approached using optimization, simulation, or both techniques. Some articles are not exactly about ambulance location as there has not been much work done on this subject in the past. However, it is important to mention the progress that has been made in this application for medical services.

Location problems arise from the need of finding the most convenient place to locate facilities such as: Distribution centers, production plants, garbage dumps, and fire, police and ambulance stations, among many others.

In general terms, the problem can, according to Daskin, be expressed as: Given the location of each user, demand, and costs (time, distance, etc.) of transport in the region in question, the number of services, the geographical location and capacity of each must be determined in order to optimize the costs of transport, operation, etc. [13].

2.1 Components of the Location Models

The basic parameters in these models are [13]:

- The customers that are to be found at points or on routes, depending on the case.
- Facilities that must be situated.

- Place where the customers and facilities should be located.
- A metric that indicates the distances, costs, or times between the customers and the facilities.

To put it another way, the main components of a location model are

(a) Demand, defined as the interaction between services and points of demand.
(b) Number of services, which represent the amount of desired or required services to be located.
(c) Measurement of distance, which is a measurement of the shape of the journey between the points of demand and the location of the services in the area being studied.
(d) Feasible solution space, this indicates the different sites where a service can be located.
(e) Objective function, this lets us assess alternative solutions and generally represents the total cost of locating the services [2].

2.2 *Classification of Models for the Location Problem*

Figure 1 shows the classification of location models based on the space in which the problems are modeled proposed by [14]. Analytic models are the simplest of location models and it is assumed that demands are distributed continuously across a service region and that facilities can be located anywhere within the region. Continuous Models typically assume that demands arise only at discrete points. The classical Weber problem is typical of this class [45]. Network models assume that demands arise, and facilities can be located only on a network composed of nodes and links. Often demands occur only in the nodes while facilities can be located anywhere on the network. In the discrete models, there may or may not be an underlying distance metric. Demands generally arise on the nodes and facilities are restricted to a finite set of candidate locations [14].

Table 1 gives a classification of the solution models for the discrete location problem. By this way, one can find the model proposed for the solution of the optimization problem.

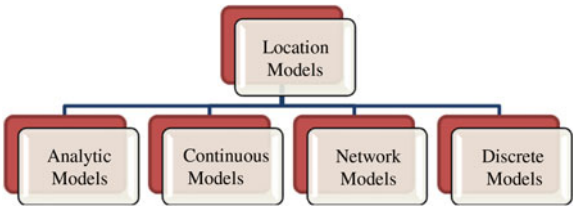


Fig. 1 Taxonomy of location models. *Source* Daskin [14]

Table 1 Classification of the main location models source: [14]

Discrete location models		
Set covering models	Median-based models	Other models
Its main objective is to cover a demand in full or in part. In many cases, the distance or response time between the customers and the points of service is decisive for customer satisfaction	Median-based models minimize the demand-weighted average distance between a demand node and the facility to which it is assigned. Such models are typically used in distribution planning contexts in which minimizing the total outbound or inbound transport cost is essential	*P-dispersion*: The model seeks to maximize the minimum of the distances between any pair of facilities
Set covering: The minimum number of facilities required to cover all the demand is obtained, guaranteeing that each node is covered by at least one feasible facility	*P-median models*: The main goal is to find the location of facilities in such a way as to minimize the cost of the service (cost associated with the distances between the demand nodes and the facilities)	One example of this is when a location is sought for a garbage dump, as it should not be close to inhabited areas
P-center models: It has the number of facilities wanted to be installed as an input parameter, and the model objective is to minimize the longest distance between a facility and the farthest demand node assigned to it	*Fix charge*: This problem is much related with p-median models and include a fix facility cost of locating at any site candidate	
Maximum covering: Constrains the number of facilities that can be located and turns the objective function to coverage, maximizing with a fixed number of facilities. The solutions can only be constrained to the nodes; this procedure is known as p-center problem vertex, and may be located not only on the nodes but also on the arcs that join them, known as p center complete		

3 State of the Art

The optimization of various aspects of emergency medical service (EMS) vehicle systems has been, since at least the mid 1960s, a very active area of research for applied mathematics and operations research. There have been hundreds of journal articles dealing with the development of models to support important decisions such as those used by e.g., [36].

1. The locations, capacities and staffing of bases;
2. The scheduling of crews;
3. The number and type of vehicles to deploy at each base;
4. The choice of which vehicle to dispatch to an emergency; and
5. The redeployment of vehicles as a function of the system state.

According to [21], these decisions can be classified into operational decisions, such as choice of dispatching policy; strategic decisions, for example, where ambulances should be stationed and what times they should operate, and tactical decisions, as station location selection. Any solution to this problem requires careful balancing of political, economic, and medical objectives. This decision process has a direct influence on the system's response time for arriving at the place where the patient is to be found [38].

There are several reasons why the design and operation of EMS systems has attracted so much attention from the operations research community. On the one hand, these issues are very important to society. Considering the large costs associated with obtaining and maintaining EMS equipment and the highly qualified staff needed, it is of prime importance to make sure that available resources get the best possible use. On the other hand, the problems are rich and interesting from the mathematical point of view; so it is important to keep up with the subtleties and complexities inherent in them as well as to come up with approaches that can be implemented in practice, given the limitations in data availability and computational resources.

Emergency medical service providers face the problem of allocating a fixed number of ambulances among a set of bases. The ultimate goal is to ensure the best possible medical outcomes for patients.

This literature review is based on the modeling approach used, and presented in chronological order. To complement the analytical works, another application-oriented articles and case studies developed so far are reviewed in this article.

Mathematical models are designed in operational research whose purpose is to optimize resources and, in line with this idea; an EMS must guarantee the provision of optimum care, considering the number of incidents and demand, in the shortest possible time. According to the type of objective function under which they operate, according to [32]: "The location and relocation models can be directed toward

- Minimizing the (average or total) time taken to attend to the incidents per time period.
- Minimizing the maximum time taken to attend to any incident.

- Maximizing the area covered in a shorter time, in the time determined as standard (Ts).
- Maximizing the number of incidents attended within a shorter time than the time that is established as the standard.

Sometimes some of these criteria are combined to get the configuration that is most appropriate for the EMS.[1]"

There can be variants in the objective function and the constraints such as: Objective function:

- Maximization of balance in coverage
- Minimizing costs
- Multicriterion: maximizing coverage at the lowest cost.

Constraints:

- Number of vehicles per located station
- Number and availability of vehicles.
- Availability of the vehicles over time.

Depending on the availability of vehicles over time, Parra [32] classifies the models as follows:

(a) Deterministic models are the ones where the resource called vehicle is modeled as a parameter with 100 % availability the moment an incident is reported. These models do not take into account the fact that the coverage of an area is partially lost when a vehicle is dispatched to attend to an incident.
(b) Probabilistic static models are those where the vehicles may or may not be available, being modeled as servers in a queue system. Calls or incident reports enter the system and, if the default server (vehicle to be dispatched) is busy, are assigned to the other servers available in the system.
(c) Dynamic models are the latest ones and focus on solving the vehicle relocation problem when some zones are left without coverage because the vehicle is busy attending to an incident in progress.

Table 2 is a summary of static-deterministic models and dynamic models, giving the names of the authors, the name of the model, the characteristic of the objective function, the characteristics of the coverage constraints and location constraints, and finally the characteristics of the ambulances [8].

Table 3 is a summary of probabilistic models, where the same items are observed as in Table 2, except that a column specifying the time the ambulance is busy has been added [8].

And finally, Tables 4, 5, and 6 present ambulance location models based on simulation, optimization, and a combination of both techniques.

[1] Baker et al. [5].

Table 2 Summary of deterministic static and dynamic models

Title	Authors/reference	Model	Description
The location of emergency service facilities. Operations research	Toregas et al. [44]	LSCM	This paper shows the location of emergency facilities as a set covering problem with equal costs in the objective. The sets are composed of the potential facility points within a specified time or distance of each demand point. One constraint is written for each demand point requiring "cover," and linear programming is applied to solve the covering problem, a single-cut constraint being added as necessary to resolve fractional solutions. The authors reports one type with an unlimited number
The maximal covering location problem	Church and ReVelle [10]	MCLP	In this paper, authors propose a model to maximize coverage (population covered) within a desired service distance S by locating a fixed number of facilities. Authors report one type and one ambulance per site
The TEAM/FLEET models for simultaneous facility and equipment sitting	Shilling et al. [39]	TEAM/FLEET	This paper proposes the TEAM and FLEET models. These models are based on the MCLM model except that they consider two different ambulances and assign different variables to a group of ambulances. Thus a demand point is not considered to be covered if a vehicle type is not available in the standard time allotted for each category. The FLEET model is a derivation of TEAM with the addition of the constraint of selecting plants for the location of the stations. Authors report 2 types of ambulances
A hierarchical objective set covering model for emergency medical service vehicle deployment	Daskin and Stern [11]	HOSC	In this paper, the authors formulate a hierarchical objective covering problem for locating EMS vehicles. The approach attempts to explicitly account for the importance of interdistrict responses. Authors modify the CSC (Conventional Set Covering) problem and propose the Hierarchical Objective Set Covering (HOSC) problem in which they find the minimum number of vehicles needed to cover all zones while simultaneously maximizing the extent of multiple coverage zones. At most one ambulance per site, one type of ambulance

(continued)

Table 2 (continued)

Title	Authors/reference	Model	Description
Concepts and applications of backup coverage	Hogan and ReVelle [22]	Modified MCLP (BACOP1 and BACOP2)	In this paper a backup coverage, or the second coverage of a demand node, is suggested as a decision criterion in modeling the location of emergency services on a network. The efficient handling of stochastic demand by vehicles which can respond to only one call at a time may require backup coverage in areas of high demand as a means to maintain a more uniform level of service. This new criterion is applied in the context of the classic covering models, the location set covering problem, and the maximal covering location problem. First coverage as defined in these models is traded off against backup coverage in the present work. Other efforts which incorporate additional levels of coverage are reviewed. The authors also show how to extend these models to third or more subsequent coverage. At most one ambulance per site, one type of ambulance
Solving an ambulance location model by tabu search	Gendreau et al. [17]	DSM	This paper considers a double coverage ambulance location problem. A model is proposed and solved with a tabu search heuristic. One type of ambulance
A dynamic model and parallel tabu search heuristic for real-time ambulance relocation	Gendreau et al. [18]	DDSM	This paper considers the redeployment problem for a fleet of ambulances. A dynamic model is proposed and a dynamic ambulance management system is described. One type of ambulance

Table 3 Summary of probabilistic models

Title	Authors/reference	Model	Description
A maximum expected covering location model formulation, properties and heuristic solution	Daskin [12]	MEXCLP	In this paper, the author proposed a variant of the maximum covering location problem that accounts for the possibility that facilities may be unable to respond to demands. The model is called the maximum expected covering location problem (MEXCLP). Several properties of this model were proven. One type of ambulance
The maximum availability location problem	ReVelle and Hogan [37]	MALP	In this paper the authors introduced a probabilistic version of the maximal covering location problem. The maximum available location problem (MALP) positions p servers in such a way as to maximize the population which will find a server available within a time standard with α reliability. The maximum availability problem is structured here as a zero—one linear programming problem and solved on a medium-sized transportation network that represents Baltimore City. One type of ambulance
The maximal expected covering location problem revisited	Batta et al. [7]	Adjusted MEXCLP (AMEXCLP)	In this paper, the Maximal Expected Coverage Location Problem is addressed (MEXCLP) by locating servers optimally in order to maximize the expected coverage of demand while at the same time taking into account the possibility of servers being unavailable when a call enters the service system. One type of ambulance
Validating and applying a model for locating emergency medical vehicles in Tucson, AZ	Goldberg et al. [19]	Adjusted MEXCLP	This paper deals with the problem of locating emergency medical vehicles in Tuczon, AZ. The model is based on a general service time approximation model for spatially distributed queuing systems. Provisions for unequal vehicle utilizations, stochastic travel times, and multiple call classes are included. The model is tailored for emergency medical systems that experience low vehicle utilizations. One type of ambulance

(continued)

Table 3 (continued)

Title	Authors/reference	Model	Description
A reliability model applied to emergency vehicle location	Ball and Lin [6]	Modified LSCM (Rel-P)	This article proposes a reliability model for emergency service vehicle location. Emergency services planners must solve the strategic problem of where to locate emergency services stations, and the tactical problem of the number of vehicles to place in each station. Authors study the problem from a system reliability perspective, where system failure is interpreted as the inability of a vehicle to respond to a demand call within an acceptable amount of time. Based on a reliability bound on the probability of system failure, they proposed a 0–1 integer programming (IP) optimization model. One type of ambulance
Developing and validating a decision support system for locating emergency medical vehicles in Louisville, Kentucky	Repede and Bernardo [35]	Time-dependent MEXCLP (TIMEXCLP)	In this paper, a maximal expected coverage location model with time variation (TIMEXCLP) is developed and integrated into a decision support system (DSS) to aid EMS planners to allocate vehicles within their service area. One type of ambulance
The queuing probabilistic location set covering problem and some extensions	Marinov and ReVelle [26]	QPLSCP	In this paper, queuing theory is applied to the development of the availability constraints. This new generation of probabilistic location model thus corrects the prior assumption of independence of server availability. One type of ambulance
Covering models for two-tiered emergency medical services systems	Mandell [25]	TTM	Emergency medical services (EMS) systems commonly consist of two types of providers, basic life support (BLS) units, and advanced life support (ALS) units, with different capabilities. In contrast to other systems in which multiple vehicle responses are necessary, in EMS systems, the first-arriving unit can begin service prior to the arrival of the second-arriving unit. In this paper, a covering-type model for two-tiered EMS systems is considered. The model, which maximizes the expected number of calls for service that are served adequately, takes server availability into account through a two-dimensional queuing model

Table 4 Ambulance location problem solved with simulation

Title	Authors/reference	Model	Description
Cooperative strategies to reduce ambulance diversion	Hagtvedt et al. [20]	Birth-death process, discrete event simulations, agent-based simulation model, and some game theory	Authors use different methods to reduce diversion, including contracts and pressure from outside regulators. The tools include a birth-death process, discrete event simulations, agent-based simulation model, and some game theory to examine the potential for cooperative strategies. The authors do not mention number of ambulances
Hypercube simulation analysis for a large-scale ambulance service system	Morohosi and Furuta [29]	Hypercube simulation analysis	A powerful simulation model for an ambulance service system is devised and applied to a large-scale ambulance system in the Tokyo metropolis. The authors report 162 ambulances
Reducing ambulance response time using simulation: the case of ValdeMarne department emergency medical service	Aboueljinane et al. [1]	Discrete simulation techniques	In this paper, discrete simulation techniques are used to model the SAMU of the Val-de-Marne department (France) in order to investigate several alternative configurations for potential improvements to reduce ambulance response time. The authors report 8 ambulances
Comparison of ambulance diversion policies via simulation	Ramirez-Nafarrate et al. [34]	Markov Decision Process (MDP)	To compare ambulance diversion (AD) policies via simulation. The AD policies analyzed include (i) a policy that initiates diversion when all the beds are occupied; (ii) a policy obtained by using a Markov Decision Process (MDP) formulation; and (iii) a policy that does not allow diverting at all. The authors do not mention number of ambulances
An efficient simulation-based approach to ambulance fleet allocation and dynamic redeployment	Yue et al. [47]	Simulation, greedy allocation algorithm	To position an entire fleet of ambulances to base locations to maximize the service level (or utility) of the Emergency Medical Services (EMS) system. The authors report 58 ambulances

Table 5 Ambulance location problem solved with optimization

Title	Authors/ reference	Model	Description
Solving a Maximal Covering Model of Emergency Ambulance Location Problem in Urban Areas by Dynamic Programming Technique	Limpattanasiri and Taniguchi [24]	MCLP and Dynamic Programming models	This paper considers the maximal covering ambulance location problem for regular traffic situation and heavy traffic congestion situation in urban areas. A two hierarchical objectives model for planning level based on MCLP model is proposed and exact searching algorithm based on Dynamic Programming technique is developed for its solution. The authors report 26 fire stations with ambulances
Location of ambulance emergency medical service in the Kumasi metropolis, Ghana	Amponsah et al. [3]	MEXCLP	In this paper a case study of the ambulance location problem in an urban setting as the Kumasi metropolis in Gahana is solved, The nonlinear Maximum Expected Covering Location Problem (MEXCLP) implemented by Satdam and Aytug was used. To solve the problem, Genetic Algorithms (GA) that uses random key coding was implemented. Real route distances were used for computation and statistical deviation was introduced in the selection of the optimal route. The authors report seven ambulances
Ambulance location and relocation problems with time-dependent travel times	Schmid and Doerner [40]	Variable neighborhood search	In this paper the authors developed a multi-period version, taking into account time-varying coverage areas, where vehicles are allowed to be repositioned in order to maintain certain coverage standard throughout the planning horizon. A mixed integer program was formulated for the problem at hand, which tries to optimize coverage at various points in time simultaneously. The problem is solved metaheuristically using variable neighborhood search. The authors report 14 ambulances
Using genetic algorithms to optimize current and future health planning the example of ambulance locations	Sasaki et al. [38]	Modified genetic algoritms	The authors predict cases for 5-year intervals from 2020 to 2050 by correlating current EMS cases with demographic factors at the level of the census area and predicted population changes. They applied a modified grouping genetic algorithm to compare current and future optimal locations and numbers of ambulances. The authors report 27 ambulances

(continued)

Table 5 (continued)

Title	Authors/year	Model	Description
Framework of TAZ_OPT Model for Ambulance Location and Allocation Problem	Shuib and Zaharudin [41]	Goal programming	This paper presents the framework of a study about an Emergency Medical Services (EMS) ambulance location and allocation model called the Time-based Ambulance Zoning Optimization Model (TAZ_OPT). The model is formulated using the goal programming (GP), where the goals are to determine the satellite locations of ambulances and the number of ambulances to be allocated at these locations. The authors do not mention number of ambulances
A case study of optimal ambulance location problems	Hozumi Morohosi [30]	Coverage and median models	A coverage model is presented in this paper; this model can be thought of reliability oriented model. On the other hand, in median model, the objective is to minimize the total traveling distance of the ambulances from the station to the scene of call. This model gives more weight to the efficiency of ambulance operation. The authors report 145 ambulances
Ambulance location and relocation models	Brotcorne et al. [8]	Deterministic and probabilistic models	This article traces the evolution of ambulance location and relocation models proposed over the past 30 years. The models are classified in two main categories. Deterministic models are used at the planning stage and ignore stochastic considerations regarding the availability of ambulances. Probabilistic models reflect the fact that ambulances operate as servers in a queuing system and cannot always answer a call. In addition, dynamic models have been developed to repeatedly relocate ambulances throughout the day. The authors do not mention number of ambulances
Solving an ambulance location model by tabu search	Gendreau et al. [17]	Coverage and Tabu Search	This paper considers a double coverage ambulance location problem. A model is proposed and a tabu search heuristic is developed for its solution. The authors mention a number of ambulances from 30 to 45

Table 6 Ambulance location problem solved with simulation and optimization

Title	Author/ reference	Model	Description
Optimization model and simulation for improving ambulance service system	Morohosi and Furuta [28]	Facility location and simulation models	This paper presents a brief survey of operations research works for ambulance service design, focusing on two mainstreams, i.e., facility location and simulation methods. Authors show some application examples of those methods using actual ambulance dispatch data. The authors do not mention number of ambulances
A simulation-based iterative method for a trauma center—air ambulance location problem	Lee et al. [23]	Integer programming and simulation	Timely transport of a patient to a capable medical facility is a key factor in providing quality care for trauma patients. This paper presents a mathematical model and a related solution method to search for optimal locations of trauma centers and air ambulances. The authors present a method that uses integer programming and simulation to iteratively update busy fraction parameters in the model, based in real data of Korean trauma cases. The authors report 15 helicopter ambulances
Design of centralized ambulance diversion policies using simulationoptimization	Ramirez-Nafarrate et al. [33]	Simulation-optimization approach	Ambulance Diversion (AD) has been an issue of concern for the medical community because of the potential harmful effects of long transportation; however, AD can be used to reduce the waiting time in Emergency Departments (EDs) by redirecting patients to less crowded facilities. This paper proposes a Simulation-Optimization approach to find the appropriate parameters of diversion policies for all the facilities in a geographical area to minimize the expected time that patients spend in nonvalue added activities, such as transporting, waiting, and boarding. The authors do not mention number of ambulances

(continued)

Table 6 (continued)

Title	Author/ refrernce	Model	Description
Simulation and optimization of the pre-hospital care system of the national university of mexico using travelling-salesman problem algorithms	Segura Pérez et al. [42]	Simulation and optimization models	A hybrid methodology was developed in this paper using optimization and simulation techniques to analyze efficiency in a pre-hospital healthcare system offered by Emergency Medical Technicians (EMTs) or paramedics. This healthcare service is offered in Mexico City while students are sitting their exams for admission to the National Autonomous University of Mexico. This study presents an optimization of the routes of an ambulance in charge of serving 26 modules installed at schools where students were attending their admission exams. This optimization is based on algorithms used to solve the travelling-salesman problem (TSP) and simulation is used to generate different system states including the calls for the ambulance. The authors report 5 ambulances
Ambulance redeployment: An approximate dynamic programming approach	Maxwell et al. [27]	Simulation and Dynamic Programming	Emergency medical service (EMS) providers are charged with the task of managing ambulances so that the time required to respond to emergency calls is minimized. One approach that may assists in reducing response times is ambulance redeployment, i.e., repositioning idle ambulances in real time. The authors formulate a simulation model of EMS operations to evaluate the performance of a given allocation policy and use this model in an approximate dynamic programming (ADP) context to compute high-quality redeployment policies. The authors report 16 ambulances
Ambulance location through optimization and simulation: The case of Milano urban area	Aringhieri et al. [4]	Integer programming and simulation models	In this paper, the problem of locating ambulance posts over an urban area is considered. A three steps approach is presented; first the real life data on the considered system behavior are analyzed. Then, integer linear programming models are considered with the aim of finding new post locations. As such models represent a simplification and an abstraction with respect to the real life situation, the behavior of the proposed solutions is tested with a simulation framework, tailored on the considered problem features. The authors report 29 ambulances

4 Simulation Paradigm

This technique provides the flexibility for modeling processes and events at different levels of complexity, dynamism, and stochastic situations. It facilitates the essential levels of real situation required to properly model the demand of events, including services such as, in this case, the emergency services.

Simulation has, over time, been used to model events and consider future events. For the service to be not only timely but efficient, the use of simulation and healthcare can be an invaluable tool where it is possible to eliminate inefficiencies, thus facilitating an optimum reordering. Simulation has mainly been used in the field of public health to analyze resource and scheduling requirements, to paraphrase [46]. When these alternatives are analyzed, the standard performance measurements are reported considering a variety of factors, including times and length of queues.

One of the goals for ambulance service in the main campus is for it to be timely and efficient, so we need to make decisions that take this service to more competitive levels, which is why simulation is an important tool for tactical decision-making. Furthermore, if we consider that the optimization model to be used is integer-based, sensitivity analysis cannot be used as it is very complex, but we can instead create scenarios that let us know alternative solutions for the model.

Simulation is an important analytical technique that has been used in management for some aspects of healthcare, mainly to maximize the system's efficiency in areas of direct patient care. Given that the power of simulation lies in its ability to model alternative systems for comparison studies and estimate the number of varied performance measurements, it lets us make changes to the system that would otherwise be impossible.

There are tried and tested different techniques for stochastic decision-making problems, as the model is closer to the real problem it becomes more complex. Owing to the stochastic nature of the variables involved, the only practical option is to use simulation. In particular, the digital computer can be used to analyze more complex problems when, owing to the quantity of data to be processed and the type of variables, they become intractable. The optimization models are in these cases expensive and time consuming. In this sense, simulation was used for modeling the system in particular the stochastic variables as the occurrence of events in the case of the main campus, where, initially, the current situation of the ambulance service is presented and the area where the work is to be done defined.

Due to its flexibility as well as its ability to find solutions that can be implemented in reality and that allow us to consider all the factors involved in the reality of a system, simulation is proposed as a suitable technique for the analysis and assessment of this system.

Discrete event simulation refers to the computer modeling of systems that evolve over time through instant changes to the variables of state. The changes occur at separate points of time.

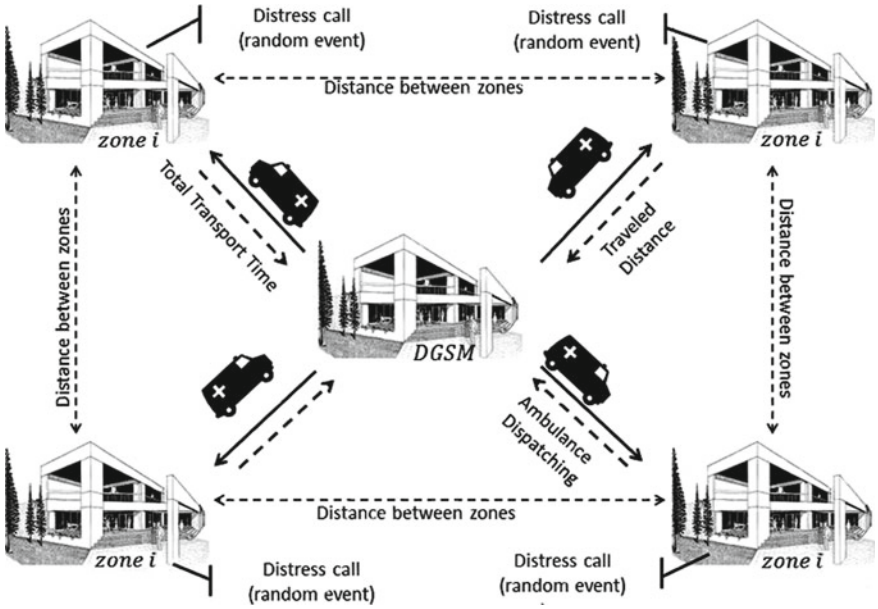


Fig. 2 Elements to consider for the simulation

In more mathematical terms, the changes in the system occur in a countable set of points in time, so we can, in this case, mention that we have a discrete system, where the changes are predominantly discontinuous. Particularly in this chapter, the events that require the ambulance service in the main campus, UNAM can be discretely simulated.

Designing a good simulation project is one of the hardest aspects of the job when the aim is to solve a real problem. However we can consider a series of characteristics that a good simulation project should have to give us the opportunity of using creativity when planning a model and finding the solution. The following model (Fig. 2) was conceived for this case, considering that each zone corresponds to each one of the faculties in the main campus:

5 The Problem in the Campus

Before the problem is explained a short history about the pre-hospital care system is presented in order to show how long has been taken to be in this situation nowadays.

5.1 The Pre-hospital Care System: A Brief History

Pre-hospital care is defined as an operational and coordination service for urgent medical problems. This has been in existence for thousands of years. It started when man first decided to go hunting and to war, when the need for early assistance was immediately recognized. The timeline (Fig. 3) illustrates part of the evolution of pre-hospital care.

As can be seen, the precursor of pre-hospital was first recorded in Ancient Greece, with the Greeks being the first to designate specialists, called arrow pullers, to look after the wounded in battles. Their method of treatment, using rudimentary techniques that could even cause more harm than good, was to continue for many years.

History followed its course until the middle ages and the arrival of the Arabs, who brought with them new methods for the treatment of patients. These were translated into Latin by the clerics of the time who, applying the knowledge they had acquired from the Arabs, performed the duties of paramedics during the crusades. But it was not until 1797, in France, that Dominique–Jean Larrey, a surgeon Napoleon's army, introduced a new way of treating wounds on the battlefield that significantly lowered the mortality rate. This innovative system, known as field care, consisted of a wagon bearing a doctor and the necessary equipment to take care of and stabilize certain wounds, and then the vehicle collected the wounded from the battlefield, taking the most serious cases to hospital. This is how the first pre-hospital transport was born, but it was not until 1865 that a hospital offered a service comparable to the service we are familiar with nowadays.

In Mexico, as in many other Latin American countries, we follow the Anglo-Saxon model where the patient is transferred to hospital. The main characteristic of this model is that pre-hospital care is carried out by emergency medical technicians. Proper emergency care cuts the number of deaths by 11 % and disabilities by 12 %, provided this care is swift, specialized and efficacious during what is called the golden hour. During this hour, the treatments and techniques revolve around resuscitation and keeping the patient stable, especially as regards keeping their airways open, and dealing with potentially treatable lifethreatening injuries in the case of patients

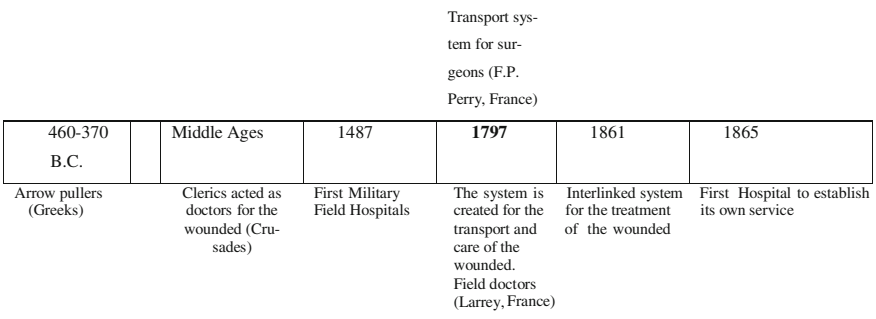


Fig. 3 History of pre-hospital treatment

with "time-dependent" clinical profiles where the response time is key. Every minute counts for patients who have suffered a trauma or heart attack, when each minute of delay in care lowers the probability of survival by between 10 and 12 %. Therefore it is essential to look for strategies aimed at promoting the development of health systems with the emphasis on the need to improve pre-hospital care in the transportation system. This task is ongoing in any pre-hospital system.

5.2 The Medical Service at the UNAM

The purpose of the UNAM's Medical Services is: "To promote, protect and restore the health of university students as part of their overall development as well as promoting healthy living among the university community and the general public" Medical Services Bureau (D.G.S.M.) of the UNAM.

In this context, medical care does not just mean the possibility of a cure but is also the means whereby young people learn to act preventively. This is basically an educational process, aimed at replacing risky behavior with habits that favor a better quality of life. This is on top of the emergency services it offers.

5.2.1 The Emergency Service

The emergency service of the University Medical Center has resources to deal with mild to moderate cases, while only serious cases and ones requiring hospitalization are sent to other health institutions.

One of the achievements of the Bureau has been the introduction of pre-hospital care programs as one of the pivotal activities of the department. The purpose of pre-hospital care is to provide immediate first aid and emergency care in situ during the first 60 min, "the golden hour"; responding to the needs of the case, by stabilizing, immobilizing, and moving the patient to the specialist service they require. Timely care offers the patient a better chance of life and lowers the incidence of invalidity or its sequels.

The medical emergency service operates 24 h a day in the University Medical Center, with the support of the Emergency Care Headquarters. It has: Four ambulances, two of which have advanced life support equipment and two have just basic support equipment. At the present time the response time of the ambulances is, on average, 5 to 6 min to the perimeter of the main campus. They operate through guards who use the phone or radio. However, in view of the rise in the population of students and faculty members, as well as administrative and service staff over the last few years, this service is not enough anymore, which is the reason why we have had to carry out this analysis to find a way to make it more efficient.

It is worth mentioning that the flow of users does not cease at any time of the year and the service is still provided during the inter-year or inter-semester periods, with a 2012 population density of 214,364 students, and 45,253 faculty members (estimated

Table 7 Number of students and faculty members in the main campus during the 2008–2012 periods

Year	University degree	Postgraduate	Faculty
2008	167,891	22,527	43,151
2009	172,444	23,875	43,252
2010	179,052	25,036	44,348
2011	180,763	25,167	44,869
2012	187,195	26,169	45,253

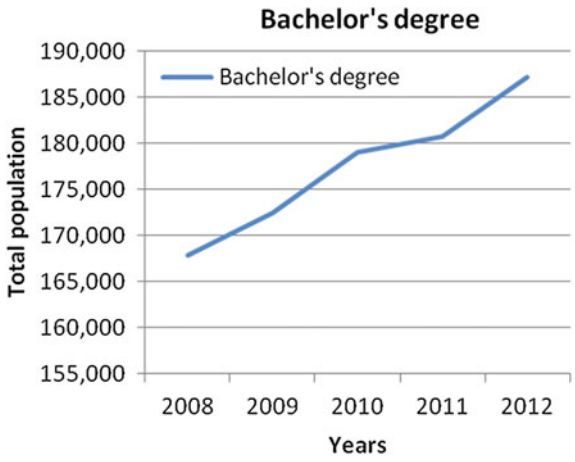


Fig. 4 Growth of undergraduate population

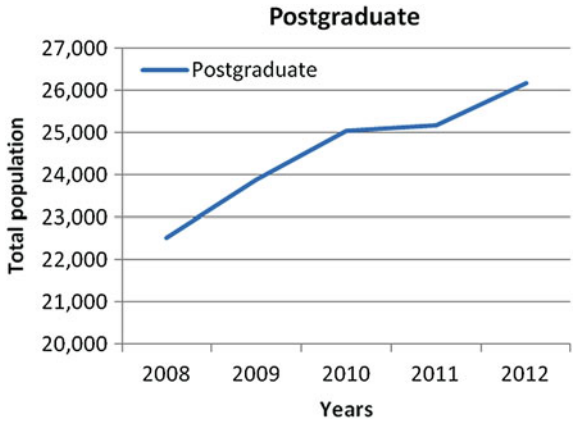


Fig. 5 Growth of the postgraduate population

values taken from the UNAM, 2012 University Statistics Webpage) (Table 7), as shown in the following figures. Figures 4, 5, and 6 show the population growth for bachelor, postgraduate, and faculty.

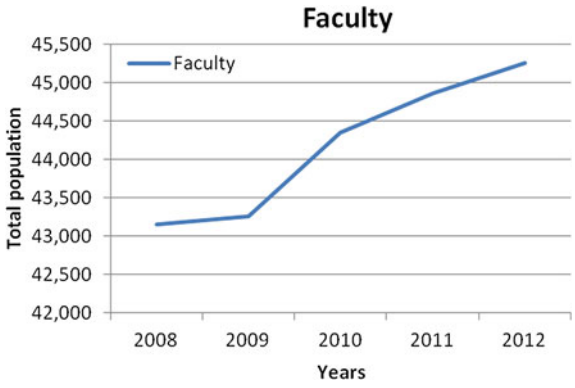


Fig. 6 Growth of the faculty's population

As can be observed in the above graphs the growth in the total population of the UNAM rises year by year, which underlines the need for more effective pre-hospital systems to look after the university community.

6 Methodology

As has already been mentioned, the methodology will be developed as follows: In the first place, all the information is collected and employed to fit these data into probability distributions that will be used in the simulation model which will help to define the stochastic demand. Then we will work with these data in the maximum covering model, experimenting with the parameters. Finally we will use the results to carry out another simulation of scenarios (Fig. 7).

6.1 Information Gathering

A research study for the location of the required ambulance services in the main campus will be valid when verifiable information is used to estimate future values and scenarios. This is why it was essential to collect 1919 items of data, such as place, type of population requiring attention and dates, using, as a sample, the control records of APH (Pre-Hospital Care) for the period from 2008 to 2011. Said data are concentrated (Table 8) and analyzed.

In order to simplify our data treatment, we only covered working days while, considering that each one of the faculties on the main campus, owing to the proximity of some of them the decision was taken to consider them by areas as per the following Table 9.

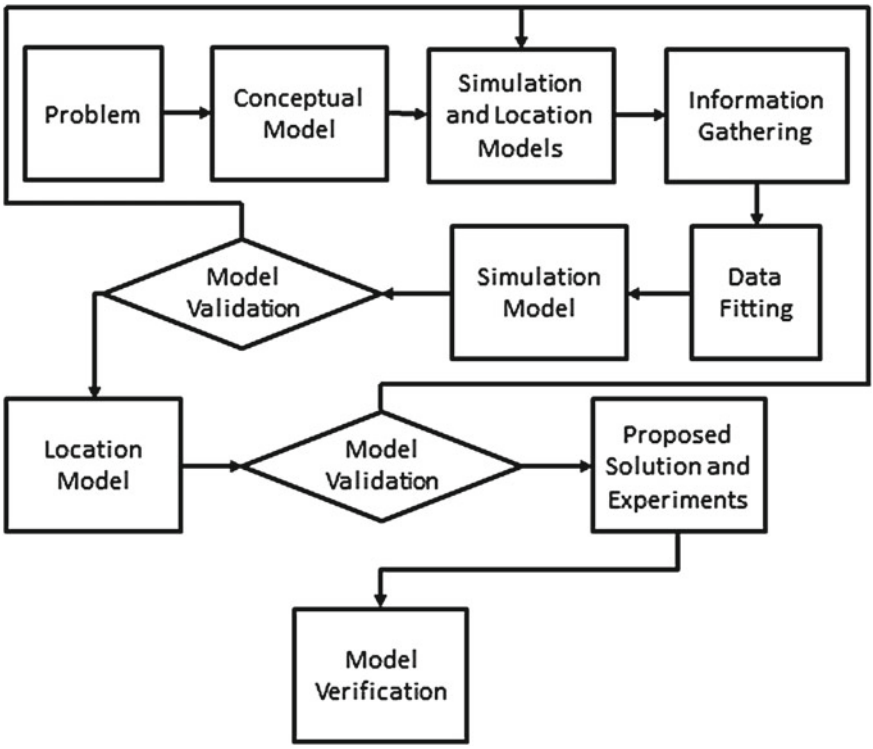


Fig. 7 Methodology developed

In the case of events that occurred during the period that was analyzed, we found a distribution of type of person to whom the service was offered, as shown following Fig. 8.

As can be observed, the population that required the most care was the students, being the majority. These data were useful for both the fit and the simulation model.

6.2 Data Fitting

The collected data need to be adjusted to some probabilistic distributions since simulation needs to have this information in order to work with. There is some commercial software for doing it, and the one that in our experience gives good results is the EasyFit© software that allows to fit data to probability distributions, giving as a result the parameters that should be considered in the simulation model, they are presented in the following Table 10.

According to the goodness-of-fit tests (Anderson-Darling and Kolmogórov-Smirnov that are defined below) provided by this program, only the values given

Table 8 Concentrate of events according to the day of the week, for the 2008–2011 periods

Concentrate	1	2	3	4	5	6	7	Total
Architecture	12	16	10	13	9	6	3	69
Political Sc	16	23	8	19	27	10	3	106
Lang. Center	8	17	10	11	4	1	1	52
Sciences	35	43	62	49	39	9	2	239
Accountancy	29	29	34	37	34	9	3	175
Law	32	46	35	36	54	12	6	221
Economics	12	9	13	13	15	3	2	67
Philosophy	31	39	30	43	38	15	11	207
Engineering	17	19	17	23	21	2	3	102
Medicine	26	25	29	29	31	9	7	156
Dentistry	26	19	20	22	20	6	3	116
Psychology	12	10	17	14	14	2	1	70
Chemistry	18	23	24	29	32	4	2	132
Social work	4	7	11	11	10	0	1	44
Vet. and Zoo.	10	22	19	12	11	4	3	81
Eng. Annex	11	11	18	12	10	3	0	65
Total	299	358	357	373	369	95	51	

Table 9 List of zones and faculties

Ref.	Zone	Faculty	Faculty	Faculty
ZB	1	Law	Economics	
ZC	2	Medicine	Dentistry	
ZD	3	Engineering	Lang center	Architecture
ZE	4	Philosophy	Central library	
ZF	5	Political sciences		
ZG	6	Sciences		
ZH	7	Accountancy	Administration	
ZI	8	Psychology		
ZJ	9	Chemistry		
ZK	10	Social work		
ZL	11	Veterinary science		
ZM	12	Engineering annex		

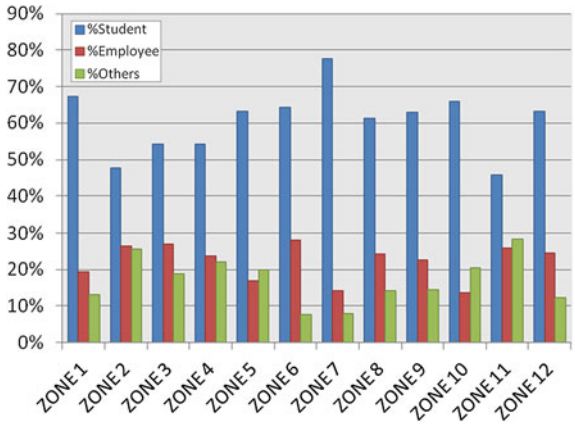


Fig. 8 Percentage of service users

for each zone are considered. These values make it possible to estimate the demands for service. The following figures represent the fitted data from the different zones considered in the model (Figs. 9, 10, 11 and 12).

Definition 6.1 Anderson-Darling Test: "The Anderson-Darling test [43] is used to test if a sample of data came from a population with a specific distribution. It is a modification of the Kolmogorov-Smirnov (K-S) test and gives more weight to the tails than does the K-S test. The K-S test is distribution free in the sense that the critical values do not depend on the specific distribution being tested (note that this is true only for a fully specified distribution, i.e., the parameters are known). The Anderson-Darling test makes use of the specific distribution in calculating critical

Table 10 Probability density function fitting parameters

	Poisson λ
Zone 1	3.2462
Zone 2	3.303
Zone 3	3.1667
Zone 4	3.451
Zone 5	3.5926
Zone 6	3.1667
Zone 7	3.2439
Zone 8	3.3333
Zone 9	3.3125
Zone 10	3.7692
Zone 11	3
Zone 12	3.214

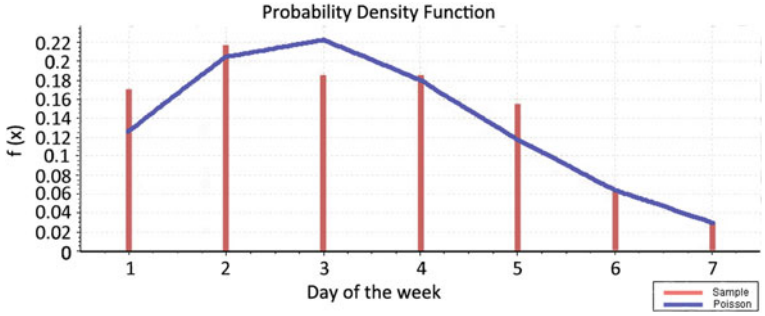


Fig. 9 Fitted Data Zone 1

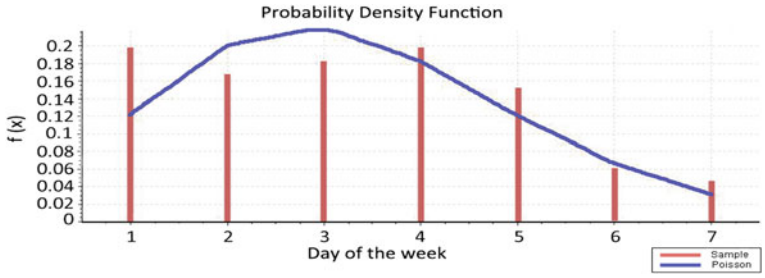


Fig. 10 Fitted Data Zone 2

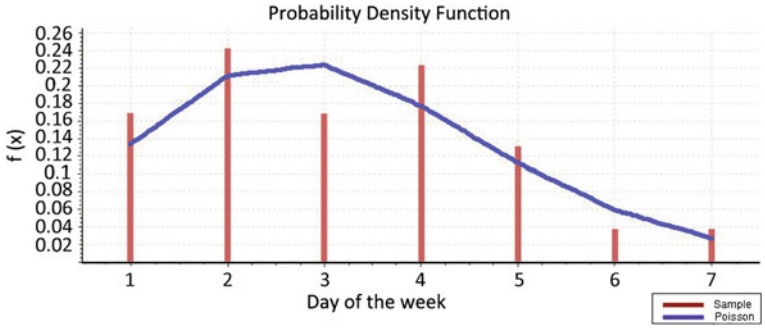


Fig. 11 Fitted Data Zone 3

values. This has the advantage of allowing a more sensitive test and the disadvantage that critical values must be calculated for each distribution".[2]

Definition 6.2 The Kolmogorov-Smirnov test [9] "is used to decide if a sample comes from a population with a specific distribution.

[2] http://www.itl.nist.gov/div898/handbook/eda/section3/eda35e.htm.

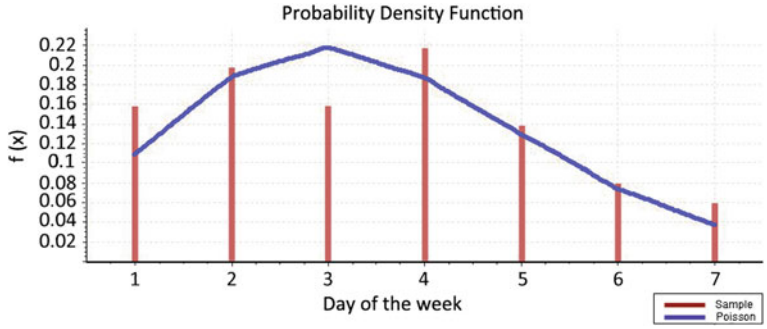


Fig. 12 Fitted Data Zone 4

The Kolmogorov-Smirnov (K-S) test is based on the empirical distribution function (ECDF). Given N ordered data points $Y_1, Y_2, \ldots, Y_N$, the ECDF is defined as

$$EN = n(i)/N$$

where $n(i)$ is the number of points less than Yi and the Yi are ordered from smallest to largest value. This is a step function that increases by $1/N$ at the value of each ordered data point."[3]

6.3 Simulation Model

This section shows the simulation model that was used to determine the behavior of the stochastic demand for which the data was fitted as shown in Sect. 6.2. Considering a Poisson distribution for each of the zones, we proceeded to construct the simulation model where each zone is considered a source that generates demand for service or as in simulation terms, a number of events. These events in turn are attended by servers which are units of transport that also have their own operating parameters. As soon as the sources generate an event, it is attended by the servers and taken to a general sink, in this case the DGSM. As the model is large, considering 12 sources, each one with its own time between arrivals, 4 ambulances (servers) and a single sink, the calculation by pencil of a sixmonth simulation would be a cumbersome task indeed. By this time there were two options, the first one was to simulate the model constructed by using a general programming language as C+, the advantage is that the simulation built would be more precise but there are two main disadvantages: One is that simulation would work just for this problem, and second animation and presentation of the solution would be less attractive. The second one was to

[3] http://www.itl.nist.gov/div898/handbook/eda/section3/eda35g.htm.

choose commercial simulation software which is why we opted to employ the SIMIO software.

SIMIO[4] uses an object-oriented modeling approach, where the models are constructed by combining objects that represent the physical component of the system. It also provides the user with a simple interface with an attractive display for the models developed in this software (Figs. 13 and 14).

In this model, the values of events per zone were estimated considering a semester with ten repeated samples, whose results are shown below.

The results given in the graphs show the quantity of events requiring the service over six months, Figs. 15, 16, 17 and 18. These results will be used in the location model as the demands for each zone. The averages of each zone will serve to facilitate the calculation. Figure 19, also shows the transporting time for the ambulances, and will be used as a comparison between the current model and the model proposed by this paper.

Once the demands of each node (zone) are estimated we shall proceed to solve a location model that allowed us to shorten the response time. Then, when several proposed solutions have been reached, the results are compared with each other (experimentation) using simulation, in order to finally provide the best solution that can be achieved within the parameters.

6.4 Location Model

The main campus has an area of approximately 9.1 km^2 in which the various university activities are carried out, such as academia, sports, and recreation, to name but a few. The needs within the grounds of the main campus necessarily correlate with the number of accidents that occur within those same grounds in a particular timeframe.

In order to systematize the optimal location of the ambulances for them to be able to provide a proper and efficient service is necessary to define: Modality, Place, and Time.

Modality is the protocol itself, which must be respected by the emergency medical technicians, while place refers to the best location inside the main campus in order to have a greater area of influence and the above items shall be assessed in such a way that, comparatively speaking, the response is optimum in terms of time.

This is where location models have an important function in planning. Some of these models have been used for medical service zones because of their mathematical similarity to tool selection problems for flexible manufacturing. In ambulance location cases, the aim is to locate them as close as possible to the demand sites [13] in order to shorten the response time, when demands can be served by any available resource. One of these models is the set covering model, where the question to be answered is: When there are a given number of people who require the service, how many resources should be located? So that everyone who is to be served is within

[4] http://www.simio.com/index.php.

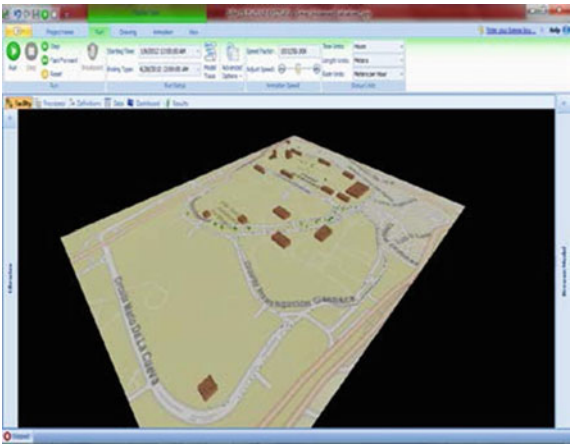


Fig. 13 3D View of the model in SIMIO

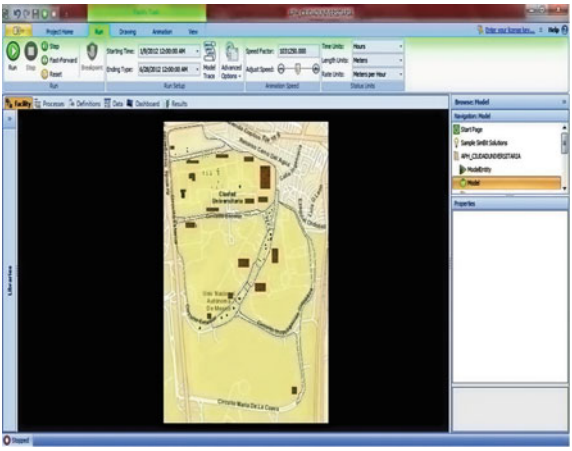


Fig. 14 Top view of the model in SIMIO

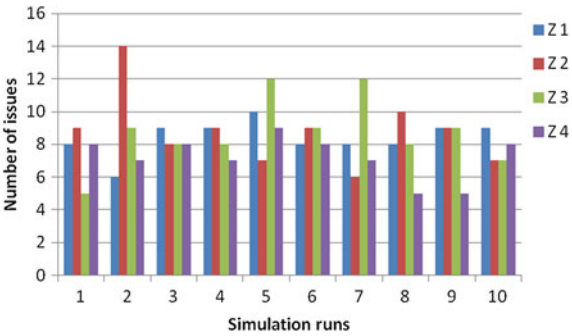


Fig. 15 Total numbers of events per zone, zones 1–4

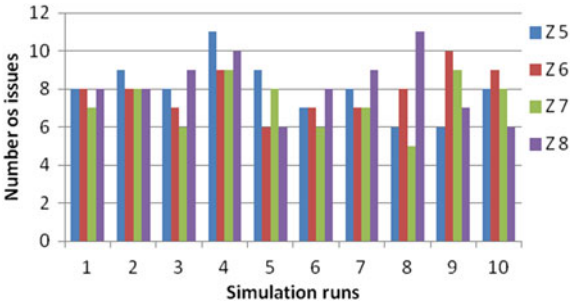


Fig. 16 Total numbers of events per zone, zones 5–8

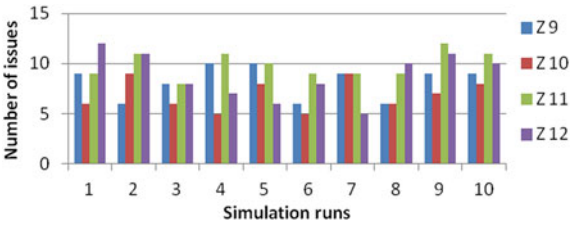


Fig. 17 Total numbers of events per zone, zones 9–12

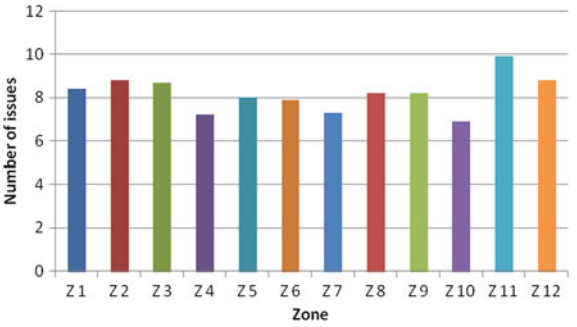


Fig. 18 Concentrate of total number of events per zone zones 1–12

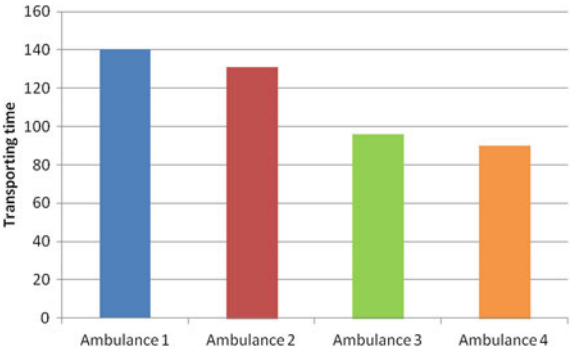


Fig. 19 Transporting time (minutes) per ambulance

Table 11 Distances in km between zones, shaded cells fulfill <= 2 kms

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*
Zone	**DGSM***	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
DGSM*	0	1.2	1.5	0.7	1	2.9	1.7	1.6	0.9	0.8	2	0.9	1.2
1	1.85	0	0.5	1.6	1	3.2	2.6	1.9	0.8	1.2	2.3	1.1	1.5
2	1.39	1.4	0	1.1	2	2.8	1.6	1.4	1.8	0.7	1.8	0.7	1.1
3	0.27	1.4	1.8	0	1.1	3.2	2	1.8	1	1	2.1	1.2	1.5
4	2.23	0.4	0.9	2	0	3.6	2.4	2.3	0.6	1.5	2.6	1.5	1.9
5	2.82	3.2	3.1	2.9	3.7	0	1.8	2.1	3.2	2.4	2.1	2.4	2.9
6	2	2	2	1.7	2.1	2	0	0.3	2.1	1.3	0.7	0.7	1.1
7	1.43	2.3	2.7	1.8	2	2	1	0	2	2	0.4	1.4	1.8
8	2.67	1.2	1.3	2.4	1.7	4	2.8	2.7	0	2	3.1	1.9	2.4
9	0.75	1.9	2.3	0.5	1.5	2.5	1.1	1	1.4	0	1.4	0.4	0.6
10	1	1.9	2.3	1.5	1.5	2	1.1	1.4	1.4	1.9	0	1.7	2
11	1.3	2.4	2.9	1.1	2.1	2.1	1.4	1.3	2	0.6	1.7	0	0.9
12	1.85	2.2	1.8	1.5	2.6	2.1	0.6	0.4	2.2	1.1	0.8	0.5	0

a given distance from the nearest resource. One common problem we have when using this model is that the solution provided is often that more resources should be located than actually existed, which would increase the costs. As it is, we already have a fixed number of ambulances and it is not very probable that more resources are going to be acquired, apart from the fact that these resources might not mean a significant reduction [13] in the response time. Therefore we employed a variation in the set covering model which is the maximum covering model, where the objective is to maximize the coverage provided by the resources, given that the resources are limited to a certain number, this being an important objective of this research work.

6.4.1 Maximum Covering Model for the Ambulances on the Main Campus

For the construction of this model, we need to know the number of ambulances to be located, the demands of the zones (nodes) and the distances between them. These data are given in Table 11.

These data will be useful for the constraints of the model, where the ambulances to be located must cover nodes within 2 km (distance between nodes). This is to guarantee their proximity to the node, and thus shorten the response time. This table can easily be replaced by a table of run times, however for the purposes of this research, we will use distances. It is worth mentioning that this is a binary integer programming model as shown below, and the DGSM row and column are excluded.

Where in equation **1** we have that h_i is the demand in place i, demands provided by the simulation model, and Z_i is equal to one if node i is covered by a resource, otherwise zero, this function maximizes the demands covered by the ambulances.

From constraints **2** to **11** we have that each one ensures that every node is covered, based on the constraint of a distance of two km or less for at least one resource.

From equations **2** to **14** we have that X_i takes values of one, if a resource is located at node i, and zero in every other case. These kinds of constraints are similar to the ones of an assignment model.

In equation **14** there is a guarantee that only the maximum number of available resources is located, which, in the case of this research, is a value of four ambulances. If the number of ambulances increases, then we need to change the right hand side of this equation.

It is worth mentioning that these constraints are in keeping with the distances in Table 11 and may vary, depending on the constraint for proximity of nodes, as shall be seen in the experimentation part, where modifications will be made for said limitation.

Objective function

Equation 1

$$max\, h_B Z_B + h_C Z_C + h_D Z_D + h_E Z_E + h_F Z_F + h_G Z_G + h_H Z_H + h_I Z_I + h_K Z_K + h_L Z_L + h_M Z_M$$

Subject to:

Equation 2

$$X_C + X_D + X_E + X_H + X_I + X_J + X_L + X_M \geq X_B$$

Equation 3

$$X_B + X_D + X_E + X_G + X_H + X_I + X_J + X_K + X_L + X_M \geq X_C$$

Equation 4

$$X_B + X_C + X_E + X_G + X_H + X_I + X_J + X_L + X_M \geq Z_D$$

Equation 5

$$X_B + X_C + X_D + X_I + X_J + X_L + X_M \geq X_E$$

Equation 6

$$X_G \geq Z_F$$

Equation 7

$$X_C + X_D + X_H + X_J + X_K + X_L + X_M \geq Z_G$$

Equation 8

$$X_D + X_E + X_F + X_G + X_I + X_J + X_K + X_L + X_M \geq Z_H$$

Equation 9

$$X_B + X_C + X_E + X_J + X_L \geq Z_I$$

Equation 10

$$X_B + X_D + X_E + X_G + X_H + X_I + X_K + X_L + X_M \geq Z_J$$

Equation 11

$$X_B + X_D + X_E + X_F + X_G + X_H + X_I + X_J + X_L + X_M \geq Z_K$$

Equation 12

$$X_D + X_G + X_H + X_I + X_J + X_K + X_M \geq Z_L$$

Equation 13

$$X_C + X_D + X_G + X_H + X_J + X_K + X_L \geq Z_M$$

Equation 14

$$X_B + X_C + X_D + X_E + X_F + X_G + X_H + X_I + X_K + X_L + X_M \leq 4$$

$$Z_i = 1, 0 \qquad i = B, C, D, \ldots, M$$

$$X_i = 1, 0 \qquad i = B, C, D, \ldots, M$$

7 Proposed Solution and Experiments

LINDO™[5] (linear, nonlinear, integer, stochastic, and global programming solvers have been used by thousands of companies worldwide to maximize profit and minimize cost on decisions involving production planning, transportation, finance, portfolio allocation, capital budgeting, blending, scheduling, inventory, resource allocation and more); was used to solve the location model and, as the model is linear and integer, it can be easily solved using this software, without having to resort to a more complex one. The model can be written in this software without requiring any special commands. This makes it possible to have a simple interface with the user where it is enough to have sufficient knowledge about how problems are set out to be able to start to solve them with this tool, as shown in Fig. 20.

Solving this model we observe that a maximum demand of $Z = 98$ events is covered when the ambulances are located as in Table 12, at nodes X_H, X_L, X_G, X_F, in respect of the zones.

In other words, an ambulance located at node X_H will cover nodes: B, C, D, G, J, K, L M; and ambulance X_L will respond to demands in nodes: B, C, D, E, G, H, I, J, K, M; ambulances located at X_G and X_F are covering each other. We can notice that, in combination, all selected nodes offer a full coverage through the zones concerned.

[5] http://www.lindo.com/.

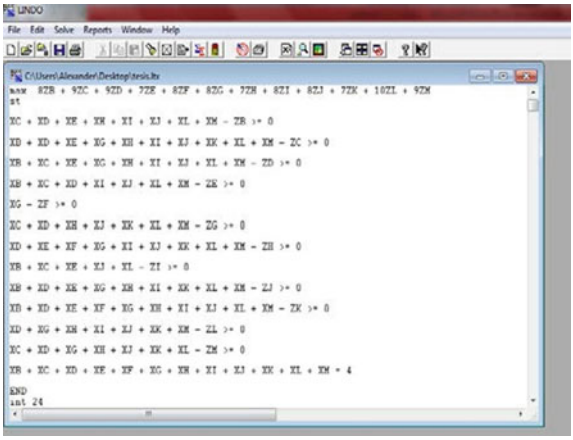


Fig. 20 LINDO Screenshot with the written model

Table 12 Location of ambulances

Node	Zone	School
H	7	Accountancy and administration
L	11	Veterinary science and zootechnics
G	6	Sciences
F	5	Political sciences

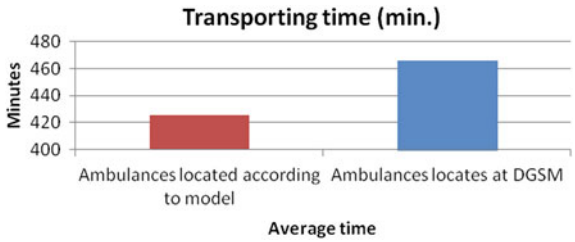


Fig. 21 Comparison of average transporting times

Once we have these data, we shall compare the results for transporting times of the SIMIO model when the ambulances are located at DGSM and when they are located as per the result given by the model.

The results of Fig. 21 show a reduction in the response time, and we can see the behavior of the model in Fig. 22 and how the transporting times decrease. Thanks to the simplicity of the models, we can experiment to try to improve the difference between the two times, as we will do below.

Experiments

The purpose of the experiments presented below is to further improve the response time (reduction of transporting time), through adjustments to the criterion for the selection of the constraints (equations 2–4) used to solve the model, bearing in mind that the first criterion used for a first approximation to the result was a distance between zones of less than or equal to 2 km (model "b"), 3 more parameters are proposed below:

1. Distance between zones less than or equal to 1.7 km apart (model "c")
2. Distance between zones less than or equal to 1.5 km apart (model "d")
3. Distance between zones less than or equal to 1.1 km apart (model "e")

The experimentation shall start with model "c" from the above list to the end, and Table 13 is employed to facilitate the writing of the constraints.

Based on Table 11 and comparing it with Table 13, we note a decrease in the number of nodes involved for each equation, as the objective function of the location model has not changed, only the rest of the model is modified, in other words from equation 2 to equation 14, giving us the following equations

$$X_C + X_D + X_E + X_I + X_J + X_L + X_M \geq Z_B$$
$$X_B + X_D + X_G + X_H + X_J + X_L + X_M \geq Z_C$$
$$X_B + X_E + X_I + X_J + X_L + X_M \geq Z_D$$
$$X_B + X_C + X_I + X_J + X_L \geq Z_E$$
$$X_H + X_J + X_K + X_L + X_M \geq Z_G$$

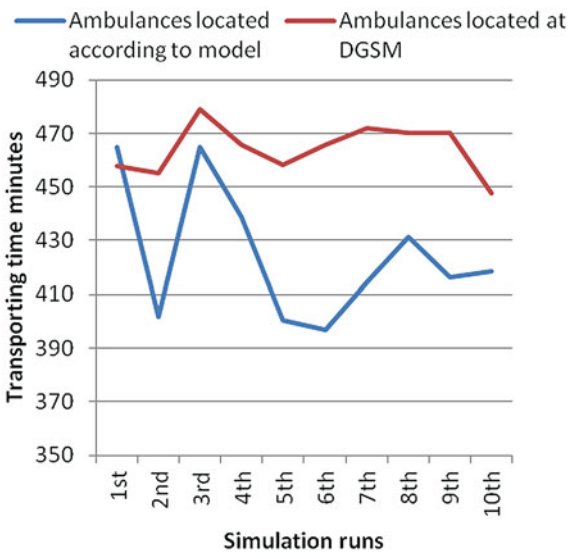


Fig. 22 Model behavior as it moves throughout simulation runs

Table 13 Distances between zones <= 1.7 km

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*
Zone	**DGSM**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
DGSM*	0	1.2	1.5	0.7	1	2.9	1.7	1.6	0.9	0.8	2	0.9	1.2
1	1.85	0	0.5	1.6	1	3.2	2.6	1.9	0.8	1.2	2.3	1.1	1.5
2	1.39	1.4	0	1.1	2	2.8	1.6	1.4	1.8	0.7	1.8	0.7	1.1
3	0.27	1.4	1.8	0	1.1	3.2	2	1.8	1	1	2.1	1.2	1.5
4	2.23	0.4	0.9	2	0	3.6	2.4	2.3	0.6	1.5	2.6	1.5	1.9
5	2.82	3.2	3.1	2.9	3.7	0	1.8	2.1	3.2	2.4	2.1	2.4	2.9
6	2	2	2	1.7	2.1	2	0	0.3	2.1	1.3	0.7	0.7	1.1
7	1.43	2.3	2.7	1.8	2	2	1	0	2	2	0.4	1.4	1.8
8	2.67	1.2	1.3	2.4	1.7	4	2.8	2.7	0	2	3.1	1.9	2.4
9	0.75	1.9	2.3	0.5	1.5	2.5	1.1	1	1.4	0	1.4	0.4	0.6
10	1	1.9	2.3	1.5	1.5	2	1.1	1.4	1.4	1.9	0	1.7	2
11	1.3	2.4	2.9	1.1	2.1	2.1	1.4	1.3	2	0.6	1.7	0	0.9
12	1.85	2.2	1.8	1.5	2.6	2.1	0.6	0.4	2.2	1.1	0.8	0.5	0

Table 14 Location of ambulances according to model "c"

Node	Zone	School
D	3	Engineering, languages, and architecture
L	11	Veterinary science and zootechnics
B	1	Law and economics
F	5	Political sciences

$$X_G + X_K + X_L \geq Z_H$$
$$X_B + X_C \geq Z_I$$
$$X_D + X_E + X_G + X_H + X_I + X_K + X_L + X_M \geq Z_J$$
$$X_D + X_E + X_G + X_H + X_I \geq Z_K$$
$$X_D + X_G + X_H + X_I + X_K + X_M \geq Z_L$$
$$X_D + X_G + X_H + X_I + X_K + X_M \geq Z_M$$

In the new solution, a maximum demand of 98 events is satisfied, locating each ambulance at the following nodes: X_D, X_L, X_B and X_F, being interpreted in accordance with Table 14.

For the new criterion we get the behavior of the model shown in Fig. 23.

This new behavior shows a reduction in the transporting time (response time) when a different model is used. We observe that model "c" offers a better solution than model "b" and we can assume that we will continue to see a reduction in time for the next models that offer alternative ambulance location strategies.

Just as a table was used to facilitate the writing of constraints in model "c", we resorted to Table 15 for model "d".

It would be a waste of time to list once again the different variations that are shown, as they tend to be so minor that it is not necessary to describe them, but we do have to mention the results we got from model "d", as shown in Table 16.

The results we obtained from the analysis of the selection of ambulance locations, an analysis that was done using the simulation model, are presented in Fig. 24.

We observe that the transporting time between models "c" and "d" has not generated any major variation but is still shorter than that of models "a" and "b", which shows that models "c" and "d" have better strategies.

Table 17 which is the base for model "e" and Table 18, the solution provided by said model, are presented below.

Based on the strategy proposed by model "e", we have the following behavior shown and discussed (Fig. 25).

The transporting times calculated for model "e" are notably longer than those provided by models "c" and "d" but less than the values in models "a" and "b", in other words times e> times c and d, times e< times a and b. Hence we can assume that shortening the distances between faculties to less than 1.5 km is better, but given the limited number of ambulances (4) we can conclude that the best option is for the distances to be between 1.5 and 1.7 km as shown in Fig. 26.

As clearly shown by the trend line, the argument holds that the best range of criterion for a shorter time is located in the 1.5 km $\leq$ *distance between zones* $\leq$

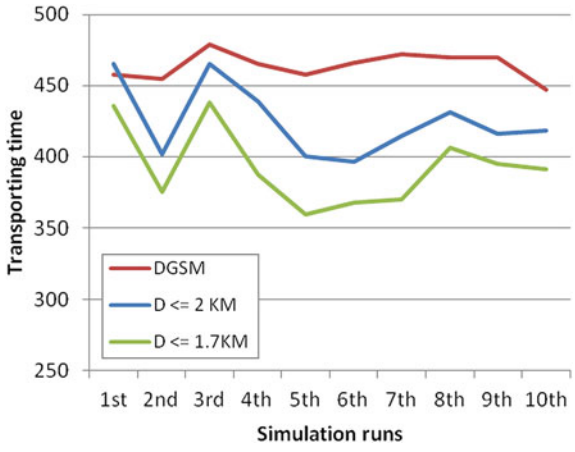


Fig. 23 Comparison of the behavior of models a, b, and c

Table 15 Distances between zones <= 1.5 km

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*
Zone	**DGSM**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
DGSM	0	1.2	1.5	0.7	1	2.9	1.7	1.6	0.9	0.8	2	0.9	1.2
1	1.85	0	0.5	1.6	1	3.2	2.6	1.9	0.8	1.2	2.3	1.1	1.5
2	1.39	1.4	0	1.1	2	2.8	1.6	1.4	1.8	0.7	1.8	0.7	1.1
3	0.27	1.4	1.8	0	1.1	3.2	2	1.8	1	1	2.1	1.2	1.5
4	2.23	0.4	0.9	2	0	3.6	2.4	2.3	0.6	1.5	2.6	1.5	1.9
5	2.82	3.2	3.1	2.9	3.7	0	1.8	2.1	3.2	2.4	2.1	2.4	2.9
6	2	2	2	1.7	2.1	2	0	0.3	2.1	1.3	0.7	0.7	1.1
7	1.43	2.3	2.7	1.8	2	2	1	0	2	2	0.4	1.4	1.8
8	2.67	1.2	1.3	2.4	1.7	4	2.8	2.7	0	2	3.1	1.9	2.4
9	0.75	1.9	2.3	0.5	1.5	2.5	1.1	1	1.4	0	1.4	0.4	0.6
10	1	1.9	2.3	1.5	1.5	2	1.1	1.4	1.4	1.9	0	1.7	2
11	1.3	2.4	2.9	1.1	2.1	2.1	1.4	1.3	2	0.6	1.7	0	0.9
12	1.85	2.2	1.8	1.5	2.6	2.1	0.6	0.4	2.2	1.1	0.8	0.5	0

Table 16 Location of ambulances with model "d"

Node	Zone	School
L	11	Veterinary science and zootechnics
B	1	Law and economics
H	7	Accountancy and administration
F	5	Political sciences

1.7 km range, where the ambulances will perform better. This point can be referred to as a local minimum because:

"Let x^* be a feasible point for the general case of an optimization problem and a set of feasible points that are, at most, a distance of x^* away from δ. Let us say that point x^* is a local minimum if $\delta > 0$, so that

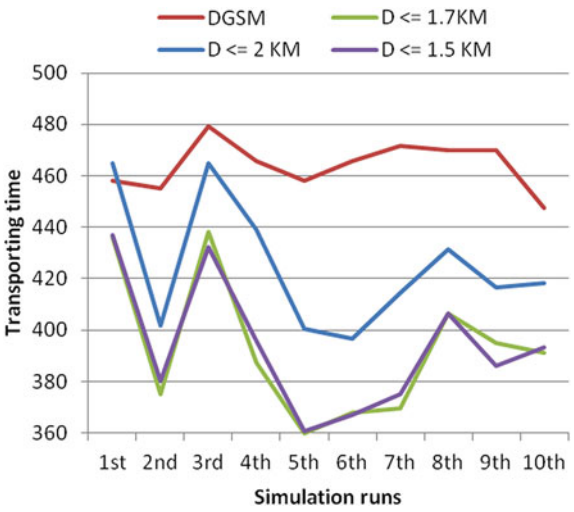


Fig. 24 Comparison of the behavior of models a, b, c, and d

Table 17 Distances between zones <= 1.1 km

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*
Zone	**DGSM**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
DGSM	0	1.2	1.5	0.7	1	2.9	1.7	1.6	0.9	0.8	2	0.9	1.2
1	1.85	0	0.5	1.6	1	3.2	2.6	1.9	0.8	1.2	2.3	1.1	1.5
2	1.39	1.4	0	1.1	2	2.8	1.6	1.4	1.8	0.7	1.8	0.7	1.1
3	0.27	1.4	1.8	0	1.1	3.2	2	1.8	1	1	2.1	1.2	1.5
4	2.23	0.4	0.9	2	0	3.6	2.4	2.3	0.6	1.5	2.6	1.5	1.9
5	2.82	3.2	3.1	2.9	3.7	0	1.8	2.1	3.2	2.4	2.1	2.4	2.9
6	2	2	2	1.7	2.1	2	0	0.3	2.1	1.3	0.7	0.7	1.1
7	1.43	2.3	2.7	1.8	2	2	1	0	2	2	0.4	1.4	1.8
8	2.67	1.2	1.3	2.4	1.7	4	2.8	2.7	0	2	3.1	1.9	2.4
9	0.75	1.9	2.3	0.5	1.5	2.5	1.1	1	1.4	0	1.4	0.4	0.6
10	1	1.9	2.3	1.5	1.5	2	1.1	1.4	1.4	1.9	0	1.7	2
11	1.3	2.4	2.9	1.1	2.1	2.1	1.4	1.3	2	0.6	1.7	0	0.9
12	1.85	2.2	1.8	1.5	2.6	2.1	0.6	0.4	2.2	1.1	0.8	0.5	0

$$F(x) \text{ is defined in } N(x^*, \delta) \text{ and}$$
$$F(x) < F(y), \forall y \in N(x^*, \delta), y \neq x^{*''}$$

Table 18 Location of ambulances according to model "e"

Node	Zone	School
I	8	Psychology
M	12	Engineering annex
G	6	Sciences
F	5	Political sciences

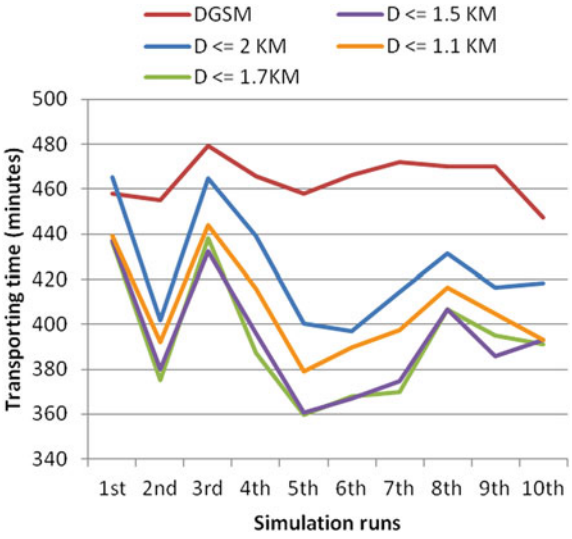


Fig. 25 Comparison of the behavior of models a, b, c, d, and e

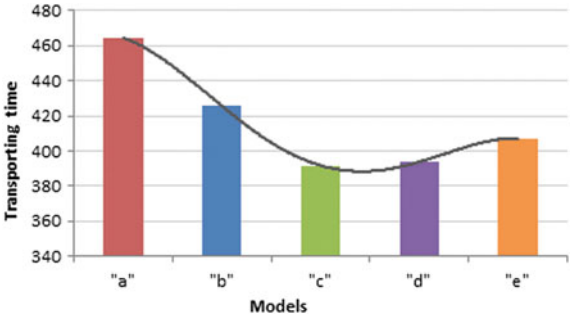


Fig. 26 Results of the comparison

Another way of knowing that we have reached a local minimum is by comparing neighboring gradients with the point that is being assessed, if the gradient that is on the left is negative and the one on the right is positive we can say that a minimum has effectively been reached. This can also be shown as ($m_i < 0$) and ($m_d > 0$). Of course, this is only an approximation.

Fig. 27 Aerial view of the zones that would include an ambulance

The best solution found under the parameters reached by the experimentation we are facilitated by simulation is by means of model "c", shown in Fig. 27.[6]

Here the zones that are marked with an oval are the ones that maximize the covering of demand and, at the same time, shorten the response time by approximately 16 % compared with the current time.

8 Discussion

The methodology described in this chapter is a combination of two models, the first being simulation and the second location, to find solutions for the problem of shortening ambulance response time on the main campus of the Universidad Nacional Autónoma de México. The simulation model is used to approach the stochastic nature of the problem, there being uncertain demands, as well as for the experimentation of results that allowed us to get a clearer and more precise picture of the nature of the problem involved with satisfying the demand in the shortest possible time, taking the constraints of the installed infrastructure into account.

[6] Pardines Lence [31].

Once we had the simulation model, it was validated against historical data that the paramedical personnel gave us, and thus we were able to go on to the second phase, which was the location model.

Thanks to the location model, we were able to consider different scenarios to set bounds for the possible solutions to the problem; and thus, having future changes in resource and demand management, we will have a more effective analytical and optimization tool.

The methodology employed enables us to include a higher number of variables and pose more complex problems within the same university, such as the transfer of patients to hospitals or including other university facilities. Likewise bigger problems such as cities can be considered where there is a larger population and more resources to be located; it is evident that the simulation model would have to be adapted as would the location model.

This methodology is recommendable for decision-making sectors where we have interference in the location of resources.

9 Conclusions

After having experimented, it is worth noting how the use of two Operational Research techniques can improve our analysis of a system until better results, such as the reduction of the response time, are achieved. If it were not for simulation and only the location model had been used, a solution that minimizes the time would, in effect, have been found but it is necessary to consider two problems: First, simulation gave the chance to build scenarios in the case the optimal solution cannot be implemented; and second, when the time to look for an optimal solution is short, at least simulation gives good solutions in short time.

There is a high level of randomness in the ambulance services on the main campus of Mexico City, with the time taken for the jobs forming part of the routine of the system. Moreover, there are no clear rules for many of the jobs, so they depend on the people doing the job. However, discrete event simulation is a useful tool for the analysis of these systems, and it is possible to develop models that are capable of satisfactorily representing all the phenomena that form part of all the activities of the system. These models facilitate the analysis of scenarios with a level of reliability that is statistically acceptable, assessing scenarios with an increased demand and others in order to shorten the response time. Therefore the combined use of optimization techniques, such as the location optimization technique, and simulation improves the search of optimum values for the system, making the simulation and analysis of a large number of alternatives possible.

Acknowledgments This research was supported by UNAM-PAPIIT grant IN116012.

References

1. Aboueljinane, Lina, Jemai, Zied, Sahin, Evren (2012) Reducing ambulance response time using simulation: the case of Valde-Marne department emergency medical service. Proceedings of the 2012 Winter Simulation Conference, pp. 943–954.
2. Ambrosino, D. et al., (2009). A heuristic based on multi-exchange techniques for a regional fleet assignment location-routing problem. Computers & Operations Research, 36, 442–460.
3. Amponsah, S.K., Amoako, Gordon, Darkwah, K.F., Agyeman, E. (2011). Location of ambulance emergency medical service in the Kumasi metropolis, Ghana. African Journal of Mathematics and Computer Science Research Vol. 4(1), pp. 18–26, January, 2011
4. Aringhieri, Roberto, Carello, Giuliana, Morale, Daniela (2007). Ambulance location through optimization and simulation: the case of Milano urban area. http://air.unimi.it/handle/2434/40782. Accessed 15-02-14.
5. Baker J. R., Clayton E. R., Taylor B. W. (1989). A Non-Linear Multi-Criteria Programming Approach for Determining County Emergency Medical Service Ambulance Allocations. The Journal of the Operational Research Society, 40(5), 423–432.
6. Ball, M. O., Lin, F. L. (1993). A reliability model applied to emergency vehicle location. Operations Research, 41(1), 18–36.
7. Batta, R., Dolan, J., Krishnamurthy, N. (1989). The Maximal Expected Covering Location Problem Revisited. Transportation Science, 23(4), 277–287.
8. Brotcorne, L., Laporte, G., Semet, F. (2003). Ambulance location and relocation models. European Journal of Operational Research, 147, 451–463.
9. Chakravari, Laha, Roy. (1967). "Handbook of Methods of Applied Statistics, Volume I", John Wiley and Sons, pp. 392–394.
10. Church R. L., ReVelle C. (1974). The maximal covering location problem. Papers in Regional Science, 32(1), 101–118.
11. Daskin M. S., Stern E. H. (1981). A hierarchical objective set covering model for emergency medical service vehicle deployment. Transportation Science, 15(2), 137–152.
12. Daskin, M. S. (1983). A maximum expected covering location model Formulation, properties and heuristic solution. Transportation Science, 17 (1), 48–70.
13. Daskin Mark S. (1995). Network and Discrete Location, models, algorithms and applications, John Wiley & Sons, pp. 92–125, 198, 208.
14. Daskin, M.S. (2008). What should know about location modeling. Naval research logistics. Wiley InterScience. 283–294. doi:10.1002/nav.
15. Flores de la Mota, Idalia, Mayra Elizondo Cortés. (2006) "Apuntes de Simulación". México, Universidad Nacional Autónoma de México, Facultad de Ingeniería.
16. Fraga-Sastrías, Juan Manuel, (2010). "Sistemas médicos de emergencia en México, una perspectiva pre hospitalaria" Archivos de Medicina de urgencia en México, Enero - April 2010.
17. Gendreau, M., Laporte, G., Semet, F. (1997). Solving an ambulance location model by tabu search. Location Science, 5 (2), 75–88.
18. Gendreau, M., Laporte, G., Semet, F. (2001). A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. Parallel Computing, 27, 1641–1653.
19. Goldberg, J., Dietrich, R., Chen, J. M., Mitwasi, G., Valenzuela, T., Criss, E. (1990). Validating and applying a model for locating emergency medical vehicles in Tucson, AZ. European Journal Of Operational Research, 49 (3), 308–324.
20. Reidar Hagtvedt, Mark Ferguson, Paul Griffin, Gregory Todd Jones, Pinar Keskinocak. (2009). Cooperative strategies to reduce ambulance diversion. Proceedings of the 2009 Winter Simulation Conference. pp. 1861–1874.
21. Henderson S.G. and Mason A.J. (2004). Ambulance Service Planning: Simulation and Data Visualisation. Operations Research and Health Care International Series in Operations Research & Management Science. 70, 70–102.
22. Hogan, K., ReVelle, C. (1986). Concepts and applications of backup coverage. Management Science, 32 (11), 1434–1444.

23. Lee, Taesik, Jang, Hoon, Cho, Soo-Haeng, Turner, John G. (2012). A simulation-based iterative method for a trauma center – air ambulance location problem. Proceedings of the 2012 Winter Simulation Conference. pp. 955–966.
24. Limpattanasiri, W., Taniguchi, E. (2013). Solving a Maximal Covering Model of Emergency Ambulance Location Problem in Urban Areas by Dynamic Programming Technique. Proceedings of the Eastern Asia Society for Transportation Studies, Vol. 9, 2013.
25. Mandell, M. (1998). Covering models for two-tiered emergency medical services systems. Location Science, 6(1–4), 355–368.
26. Marianov, V., ReVelle, C. (1994). The Queuing Probabilistic Location Set Covering Problem and some Extensions. Socio-Economic Planning Sciences, 28(3), 167–178.
27. Matthew S. Maxwell Shane G. Henderson Huseyin Topaloglu. (2009). Ambulance redeployment: an approximate dynamic programming approach. Proceedings of the 2009 Winter Simulation Conference. pp. 1850–1860.
28. Morohosi, Hosumi, Furuta, Takehiro (2013). Optimization model and simulation for improving ambulance service system. Operations Research and its Applications in Engineering, Technology and Management 2013 (ISORA 2013), 23–25 August, 2013.
29. Morohosi, Hosumi, Furuta, Takehiro (2012). Hypercube simulation analysis for a large-scale ambulance service system. Proceedings of the 2012 Winter Simulation Conference. pp. 1211–1218.
30. Morohosi, Hozumi (2008), A case study of optimal ambulance location problems. The 7th International Symposium on Operations Research and Its Applications (ISORA'08) Lijiang, China, October 31–Novemver 3, pp. 125–130.
31. Pardines Lence, Inmaculada; (2007). Técnicas paralelas aplicadas a optimización no lineal en sistemas de memoria distribuida. España 2007, p. 8.
32. Parra O. O.J. (2011). Revisión del estado del arte en modelos de localización y relocalización de vehículos para atención de emergencias. Revista Elementos 1.
33. Ramirez-Nafarrate, Adrian, Fowler, John W., Wu, Teresa. (2011). Design of centralized ambulance diversion policies using simulation-optimization. Proceedings of the 2011 Winter Simulation Conference, pp. 1251–1262.
34. Ramirez-Nafarrate, A. Baykal Hafizoglu, Esma S. Gel, John W. Fowler (2012). Comparison of ambulance diversion policies via simulation. Proceedings of the 2012 Winter Simulation Conference. pp. 967–978, 2012.
35. Repede, J., Bernardo, J. (1994). Developing and validating a decision support system for locating emergency medical vehicles in Louisville, Kentucky. European Journal of Operational Research, 75(3), 567–581.
36. Restrepo M. (2008). Computational methods for static allocation and real-time redeployment of ambulances. Dissertation Faculty of the Graduate School of Cornell University.
37. ReVelle, C., Hogan, K. (1989). The Maximum Availability Location Problem. Transportation Science, 23(3), 192–200.
38. Sasaki S., Comber A., Suzuki H., Brunsdon C. (2010). Using genetic algorithms to optimise current and future health planning - the example of ambulance locations. International Journal of Health Geographics, 9(4), 1–10.
39. Schilling David, et al. (1979). The team / fleet models for simultaneous facility and equipment siting, Operations Research Society of America, pp. 163–175.
40. Schimd, Verena, Doerner, Karl F. (2010). Ambulance location and relocation problems with time-dependent travel times, European Journal of Operational Re-search 207 (2010) 1293–1303.
41. Shuib, Adibah, Zaharudin, Zati Aqmar. (2010). Framework of TAZ_OPT Model for Ambulance Location and Allocation Problem. (2010) World Academy of Science, Engineering and Technology. Vol: 48 2010–12-22.
42. Segura, Esther, Altamirano, Luis, Flores, Idalia. (2010). Simulation and Optimization of The Pre-Hospital Care System of the National University of Mexico Using Travelling Salesman Problem Algorithms. Proceedings of: SummerSim '10-2010 Summer Simulation Multiconference, Ottawa, ON, Canada, July 11–14.

43. Stephens, M. A. (1974). EDF Statistics for Goodness of Fit and Some Comparisons, Journal of the American Statistical Association, 69, pp. 730–737.
44. Toregas, C., Swain, R., ReVelle, C., Bergman, L. (1971). The Location of Emergency Service Facilities. Operations Research, 19(6), 1363–1373.
45. Weber, A. (1909) Uber den standort der industrien, tubingen english translation, by J.C. Frieerich. Translated as Alfred Weber's Theory of the location of industries. University of Chicago Press, 1929.
46. Weng, Mark L. and Houshmand, Ali A. (1999). Healthcare simulation: a case study at a local clinic, Proceedings of the 1999 Winter Simulation Conference, pp. 1577–1584.
47. Yisong Yue, Lavanya Marla, Ramayya, Krishnan. (2012). An Efficient Simula-tion-based Approach to Ambulance Fleet Allocation and Dynamic Redeployment. http://www.select.cs.cmu.edu/publications/scripts/papers.cgi?Yue-Marla-Krishnan:aaai2012. Accessed 28-02-14.

Simulation-Based Optimization Using Greedy Techniques and Simulated Annealing for Optimal Equipment Selection Within Print Production Environments

Sudhendu Rai, Eric Gross and Ranjit Kumar Ettam

Abstract Xerox has invented, tested, and implemented a novel class of operations-research-based productivity improvement offerings, marketed as Lean Document Production (LDP), for the $100 billion printing industry in the United States. The software toolkit that enables the optimization of print shops is data-driven and simulation-based. It enables quick modeling of complex print production environments under the cellular production framework. The software toolkit automates several steps of the modeling process by taking declarative inputs from the end user and then automatically generating complex simulation models that are used to determine improved design and operating policies. This chapter describes the addition of another layer of automation consisting of simulation-based optimization using simulated annealing and greedy search techniques that enable the search of a large number of design alternatives in the presence of operational and cost constraints. The greedy search procedure quickly determines an acceptable solution in a web-based online application environment. The simulated annealing technique is more time consuming and is performed offline. The results of the application of this approach to real-world problems are described.

1 Introduction

Xerox is the world's leading enterprise for business process and document management solutions. Xerox produces and sells a range of color and black-and-white printers, multifunction systems, photocopiers, digital production printing presses, and related consulting services and supplies. Xerox participates in the printing industry by providing services, via Xerox Managed Services (XMS), to manage print

S. Rai (✉) · E. Gross · R.K. Ettam
Xerox Corporation, 800 Phillips Road, Webster, NY 14450, USA
e-mail: Sudhendu.Rai@xerox.com

E. Gross
e-mail: Eric.Gross@xerox.com

R.K. Ettam
e-mail: Ranjit.Kumar2@xerox.com

M. Mujica Mota et al. (eds.), *Applied Simulation and Optimization*,
DOI 10.1007/978-3-319-15033-8_9

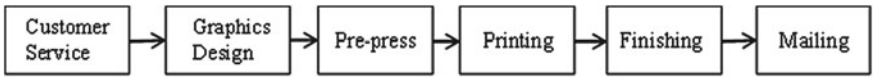


Fig. 1 A print production workflow showing the various production operations

operations for clients who choose to outsource their in-plant print operations. Xerox has invented, tested, and implemented a novel class of operations-research-based productivity improvement offerings for the printing industry that has been extensively described in [17]. This work was a finalist in the 2008 Franz Edelman competition.

Print service centers are document manufacturing systems which take raw material and information as input and through a series of processing steps create final finished document products such as books, brochures, checks, invoices, and the like. They are designed to manufacture highly customized documents that are often embedded in their workflows. The document production steps associated with print jobs are indicated in Fig. 1. Typically print service centers have departments that support individual steps in this workflow. Each department supports many different types of internal workflows resulting from the use of different types of software tools, printing machines (e.g., offset, digital), and finishing equipments such as cutting, binding, laminating, and shrink wrapping. For further description of the steps we refer the reader to [17].

The LDP software toolkit automates several steps of the print production modeling process by taking declarative inputs from the end user and then automatically generating complex simulation models that are used to determine improved design and operating points for print shops. In this chapter, we describe the addition of another layer of automation to the LDP toolkit consisting of simulation-based optimization using greedy search techniques and simulated annealing that enables the automated search of a large number of design alternatives in the presence of operational constraints to determine a cost-effective solution for the print production environment.

The printing industry is highly fragmented with thousands of print shops that are geographically distributed. This approach lends itself to being utilized for optimizing print shops across multiple geographies by users less skilled in the art of modeling, simulation and optimization, thereby allowing unprecedented scalability of a simulation-based optimization approach to a wide user base. This is important since users are able to utilize the simulation-based optimization toolkit to make complex design and operational decisions and develop optimized designs without the arduous task of building the simulation models and the associated optimization framework.

This chapter is organized as follows. Section 2 provides a literature review on simulation-based optimization approaches. Section 3 describes the specifics of the problem being addressed in this chapter. Section 4 provides an overview of the Lean Document Production toolkit. Section 5 describes the existing procedure of selecting the optimal printing equipment. Section 6 describes the simulation-based optimization techniques using the LDP toolkit. Section 7 describes some applications and case studies using real-world examples. Lastly in Sect. 8 we present our conclusions and future scope of work.

2 Literature Review

The problem of constrained simulation optimization over a finite discrete set of decision variables is common and has received significant attention. A two-phase statistically valid procedure that detects feasibility of systems in the presence of one constraint with a prespecified probability of correctness was presented in [3]. This procedure was extended to the case of multiple constraints in [4]. An algorithm for optimal sampling allocations using large deviation theory was provided under stochastic settings [19]. Iterative heuristic algorithms [11], optimal computing budget allocation framework [15] was proposed for selecting the best design from a discrete number of alternatives in the presence of a stochastic constraint via simulation experiments with limitations on simulation budget or probability of correctness. A novel method [13] that converts constrained optimization into unconstrained optimization by using the Lagrangian function was proposed for the problem over discrete sets with noisy constraints.

The approaches discussed above either visit all the designs or convert the problem into a single objective function to find the best system. Suppose we conduct n simulation replications for each of θ designs, we need $n\theta$ total simulation replications. If the precision requirement is high, and if the total number of designs in a problem is large, then $n\theta$ can be very large, making the system evaluation computationally expensive using the existing methods. In such cases, stochastic search algorithms such as simulated annealing, tabu search, and genetic algorithms prove to be the best choice. Simulated annealing [12] has shown successful applications in a wide range of combinatorial optimization problems, and this fact has motivated researchers to use simulated annealing in many simulation optimization problems. But these search techniques need to be adapted for the stochastic environment associated with discrete-event systems optimization.

Haddock and Mittenthal have investigated the feasibility of using a simulated annealing algorithm in conjunction with a simulation model [7]. A variant of the simulated annealing algorithm was developed for solving discrete unconstrained stochastic optimization problems by using a constant temperature and convergence criteria as the number of visits made by the different states in the first m-iterations to estimate the optimal solution [2]. Two variants of the simulated annealing algorithm with a decreasing cooling schedule that are designed for solving unconstrained discrete simulation optimization problems was presented in [14]. For solving stochastically constrained simulation optimization systems, an integrated approach using the simulated annealing algorithm for parameter selection followed by Monte Carlo simulation for performance evaluation was presented in [1].

Unlike ranking and selection procedures, the application of metaheuristics techniques to simulation optimization problems in stochastic settings may not guarantee that an acceptable solution, if one exists, will be found. But in most cases we observe that they converge to acceptable solutions in a reasonable amount of time which is most desirable in many real-world applications. In this paper we have present the modified simulated annealing approach that can handle uncertainty in simulation

output and stochastic constraint(s). The algorithm starts with an initial feasible solution and utilizes a decreasing cooling schedule. We perform the *student's t* hypothesis test for determining the feasibility of a solution at the current iteration [1]. Our algorithm is distinct to the procedure in [1] by not restricting the neighborhood search to feasible moves only.

In a web-based simulation optimization applications, approaches that result in optimal/near optimal solutions in a reasonable time are desirable. A greedy approach is frequently a good alternative which makes locally optimal choices at each stage with the hope of finding a global optimum. An efficient greedy approach to allocate ambulance fleet in emergency medical services system was presented [20]. To determine the optimal configuration of a conveyor-based automatic material handling systems in wafer fabs, a greedy heuristic was proposed [10]. Discussion on greedy approximation for dock allocation in a food distribution center can be found in [6]. In this chapter we present the greedy approach for optimal allocation of equipment in print production environment in a web-based online application. The greedy algorithm initially starts with sufficient number of production equipment and systematically reduces the number. The algorithm removes one or more devices such that the customer's performance criteria are not violated. This process is repeated until no more cost reduction is possible subject to the constraint. Alternatively, the algorithm can begin with no or a minimal set of equipment and systematically increases that number.

3 Problem Description

Print service centers experience many sources of variability that make them hard to analyze and optimize. They exhibit high levels of job size variations, routing complexity, and demand fluctuations as shown in Fig. 2. These service centers are primarily make-to-order service systems that cater to specific requests of each incoming customer. The incoming service requests have random arrival and due date requirements that vary from job to job and often exhibit variability within the same job type. The job sizes are often characterized by highly non-normal distributions and sometimes heavy-tailed [16]. In addition to the above challenges, print shops also experience long bid times, variability in labor and equipment characteristics, etc.

The LDP toolkit automates the workflow modeling and analysis of the print service center. In order to optimize the cost and performance of a print service center, the user manually evaluates a limited number of designs and selects the best design among them. For example, to select the optimal equipment that minimizes the cost of equipment, while simultaneously meeting the performance of a print service center such as turnaround time, number of late jobs, operator or equipment utilization, process cycle efficiency, etc., the user have to simulate multiple equipment configuration scenarios manually and select the cost-effective solution among them. This process is labor intensive, time consuming, and often ad hoc. In this chapter we have described an automated method to assist in selecting cost-effective solutions for a print service center by integrating the optimization algorithm with the LDP solution.

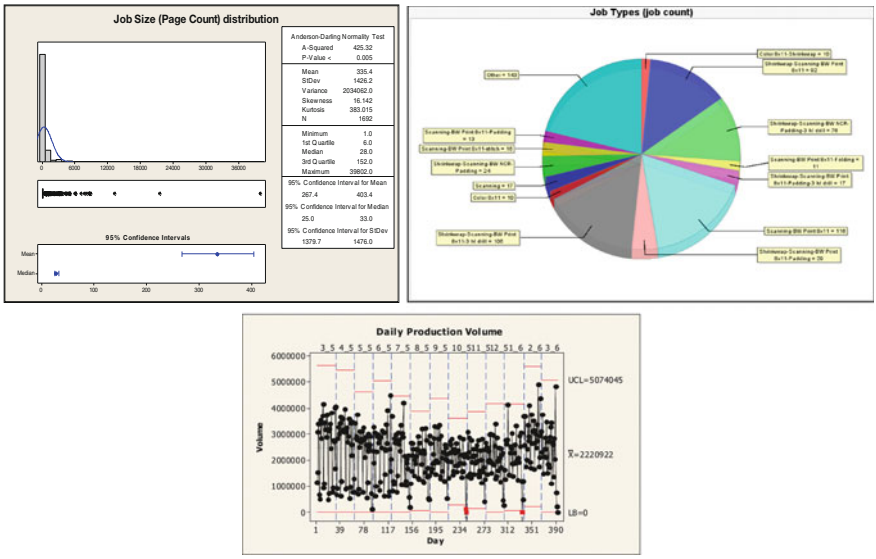


Fig. 2 Multiple sources of variability in a print production environment such as power law job size distributions, multiple coexistent job types, and high demand fluctuation

4 The LDP Solution for Print Service Center Environment

To address the complexity of operations associated with the print production processes, the service center resources are organized into autonomous cells [17]. As a result, the most common jobs can be finished autonomously inside (at least) one of these cells. Figure 3 shows how traditional print service centers are organized based on a departmental structure operated by specialized workers and compare it to the redesigned operational framework based on autonomous cells where diverse pieces of equipment are collocated and operated by cross-trained workers.

To orchestrate the flow and control of jobs through the parallel hierarchical cell structure, the Lean Document Production Controller (LDPC) uses 2-level architecture for production management. The LDPC has:

- A service center controller module (Service center CM)—high-level controller, in charge of global service center management.
- Several cell controller modules (Cell CMs)—low-level controllers, in charge of local management inside cells.

4.1 Simulation

Simulation is performed to assess the results of improvements resulting from changes in workflow grouping, operator cross-training, grouping diverse equipment into

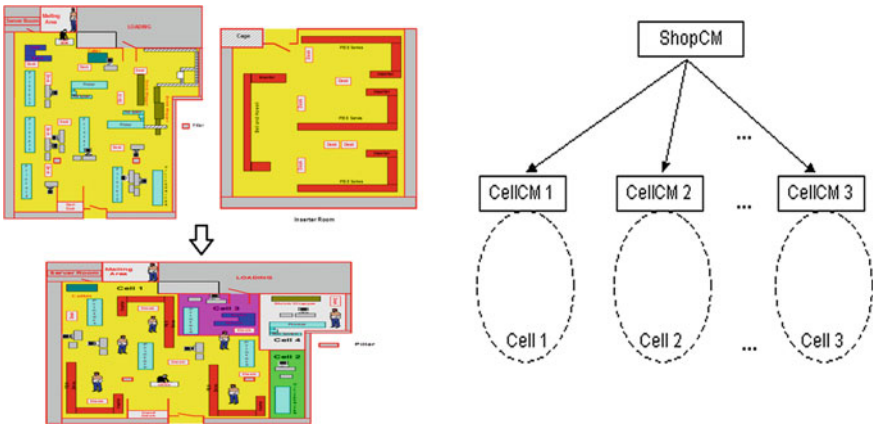


Fig. 3 Figure showing how a departmental configuration of a print service center is transformed into a cellular structure utilizing autonomous cells and the corresponding two-level architecture for the Lean Document Production Controller

autonomous cells and scheduling policies. Building discrete-event simulation models is often a time-intensive effort especially when various scenarios have to be investigated to determine improved solutions. To facilitate the model building process, the LDP tool is employed to build the simulation models from a declarative user interface (Fig. 4). This allows for fast and efficient evaluation of a large number of what-if scenarios and greatly aids in determining an improved solution out of a large search space.

The user specifies the equipment characteristics, elements of the cell, scheduling policies, number of operators and their skill level, and workflow/job characteristics

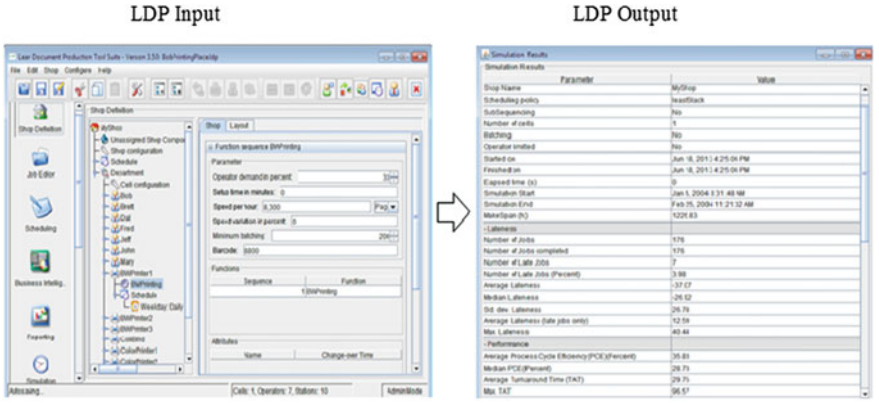


Fig. 4 Illustrates the user interface for defining the printing equipment, operators and shop policies, and simulation results for a sample print service center

as inputs to the simulation model using the LDP user interface (Fig. 4). Before the shop is simulated, the user schedules the jobs automatically using the scheduling architecture as described above. Next the tool simulates the operation of the print service center and outputs various performance metrics such as average turnaround time, number of late jobs, operator and equipment utilization, maximum turnaround time, and process cycle efficiency, etc., as shown in Fig. 4.

5 Existing Procedure for Selecting Optimal Equipment Design in a Print Service Center

The selection of optimal printing equipment in the print service center is currently carried out manually. The user first defines the necessary equipment type, cost, and other characteristics (speed, setup time, failure, and repair rates, etc.) in each cell. The job workflow characteristics and other shop operating policies (job sequencing policy, batching, and work in progress, etc.) are collected from the shop and uploaded to the LDP tool. An equipment design is defined as a combination of different numbers of equipment types in each cell. The user has to create different equipment designs that he is interested in by varying the quantity of each equipment type in each cell. Each of these equipment designs is simulated N times in order to create performance metric distributions (in the case where the simulation is subject to random events such as machine failures and job variation). Then, the mean performance measure of interest and total cost of the equipment is computed. Finally, the user selects the equipment design that has the least cost and meets the desired print service center performance goal as specified by the user. This process of evaluating multiple design configurations is labor intensive, time consuming, and can lead to solutions far from optimal. Figure 5 illustrates the detailed process flow diagram of the existing procedure.

6 Simulation-Based Optimization Using the LDP Toolkit

The main idea presented here is the integration of the optimization routine and simulation module within the LDP toolkit that embodies many elements of shop specification and modeling automation. This enables the automatic search of an optimal solution for the print production service center. For more detailed discussion, applications and benefits of integrating optimization with simulation can be found in [5, 8, 18, 21].

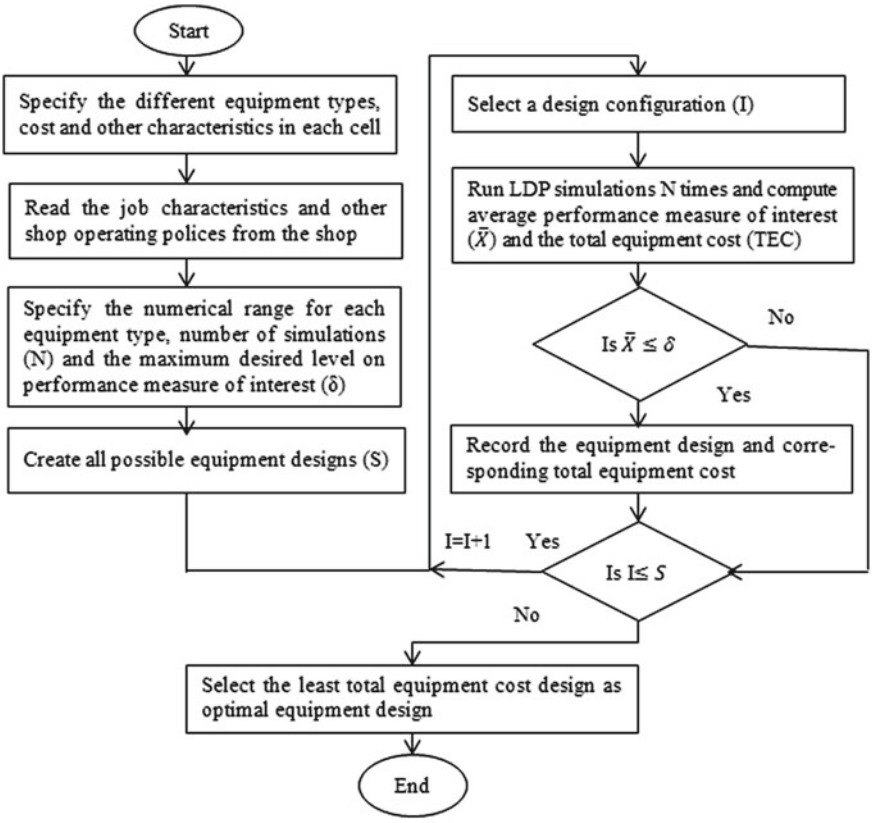


Fig. 5 The existing procedure for selecting optimal equipment configuration using LDP tool in print service center

6.1 Problem Formulation

The problem of selecting the cost-optimal equipment solution for the print production environment in the presence of stochastic operational constraints such as average turnaround time, number of late jobs, maximum turnaround time, etc., over a large number of design alternatives can be formulated as below.

$$\begin{aligned} \text{Objective}: \quad & \min_{\boldsymbol{X_k} \in \boldsymbol{S}} \; f_0(\boldsymbol{X_k}) \\ \text{Subject to}: \quad & f_1(\boldsymbol{X_k}) \leq \delta \\ & lb_{ij} \leq x_{ij} \leq ub_{ij}, i = 1..n_j, j = 1..m \\ & \boldsymbol{X_k} = [x_{ij}] \end{aligned} \tag{1}$$

where $\boldsymbol{S}$, the search space, is a finite and discrete set of equipment design configurations; X_k is the kth equipment design configuration, which is the vector combination in the number of each type of equipment in each cell; k is the index of equipment design configuration; x_{ij} is the number of the ith type of equipment in the jth cell; n_j is the number of unique equipment types in cell j; m represents the total number of cells in the print service center; lb_{ij} and ub_{ij} are the lower and upper bounds on the number of the ith type of equipment in the jth cell; $f_0(X_k)$ is the total equipment cost defined as $C_{ij} \times x_{ij}$, where C_{ij} is the cost of ith equipment in jth cell; $f_1(X_k)$ is the print service center performance measure such as average turnaround time, number of late jobs, maximum turnaround time, etc., which cannot be evaluated exactly, but needs to be estimated via the LDP simulation. Let A_{kl} be the print service center performance observation observed from simulation replication l of system k, then $f_1(X_k) = E[A_{kl}]$; and δ is the maximum desirable level of the print service center performance measure.

6.2 Modified Simulated Annealing Algorithm

Here, we present the modified simulated annealing algorithm used for solving Eq. 1. The algorithm consists of two phases: initial feasible solution phase and optimal solution phase. In the initial feasible solution phase, the algorithm starts with searching for an initial feasible solution by randomly selecting a solution from the design search space until the stopping criteria is met. If an initial feasible solution was found, the algorithm starts with this solution and identifies the optimal solution by utilizing a decreasing cooling schedule in the optimal solution phase. In the case of initial unfeasible solution, the algorithm is terminated.

Moreover, the constraints in Eq. 1 are stochastic and the general-purpose simulated annealing approach has to be adapted to consider the feasibility of a solution when it moves from one solution to another. A solution is feasible if it meets the print service center performance goal as specified by the user. To test the feasibility of a solution, we use the following procedure [1].

Let us consider, an arbitrary stochastic constraint $g(\boldsymbol{\theta}) \leq \delta$, where $g(\boldsymbol{\theta})$ is the stochastic simulation output for design $\boldsymbol{\theta}$ and δ being the maximum desirable level specified by the user. Letting $g_i(\boldsymbol{\theta})$ denote the ith simulation replication and running simulation n times, the mean and variance estimate for $g(\boldsymbol{\theta})$ could be determined over n replications as:

$$\hat{g}(\boldsymbol{\theta}) = \sum_{i=1}^{n} g_i(\boldsymbol{\theta}) \big/ n$$

$$\hat{\sigma}_{\hat{g}(\boldsymbol{\theta})} = \sum_{i=1}^{n} (g_i(\boldsymbol{\theta}) - \hat{g}(\boldsymbol{\theta}))^2 \big/ n - 1$$

The hypothesis statements for feasibility conditions are as follows:

$$\textit{Null Hypothesis } H_0 : \hat{g}(\boldsymbol{\theta}) <= \delta$$
$$\textit{Alternate Hypothesis } H_1 : \hat{g}(\boldsymbol{\theta}) > \delta$$

Accept the null hypothesis H_0, if $\omega = \hat{g}(\boldsymbol{\theta}) + t_{n-1,1-\alpha} \times \hat{\sigma}_{\hat{g}(\boldsymbol{\theta})}/\sqrt{n} \leq \delta$ where,

$n-1$ is the degrees of freedom
$1-\alpha$ is the upper critical point for the t distribution
$\hat{g}(\boldsymbol{\theta})$ is the mean value of n simulation observations
$\hat{\sigma}_{\hat{g}(\boldsymbol{\theta})}$ is the standard deviation of $\hat{g}(\boldsymbol{\theta})$.

Unlike to the approach [1], our algorithm does not restrict the neighborhood search to feasible moves only. In their approach the temperature length (M) parameter is not incremented until a neighboring feasible solution is found, resulting in unknown/more number of evaluations. When the probability of finding a feasible neighborhood solution is very low, this may result in indefinite looping. In the modified simulated annealing algorithm, a move to a neighborhood solution is irrespective of the feasibility of the solution, providing more control on the total number of evaluations by the algorithm. Let T_0 be the initial temperature, T_f be the final temperature (T_0/T_{depth}) and r the temperature decay rate. This results in the following series of annealing temperatures:

$$T_0, T_0 \times r, T_0 \times r^2, T_0 \times r^3 \ldots \ldots + T_0 \times r^n,$$

$$T_f = \frac{T_0}{T_{depth}} = T_0 \times r^n,$$

$$n = \frac{\log^1/T_{depth}}{r}.$$

If the number of times to search a neighborhood solution at a given temperature is L, then the number of evaluations is $n \times L$. The value of L is fixed throughout the algorithm and is determined using trial-and-error approach. To estimate the performance measure the algorithm makes use of all the historical observations obtained at that solution. Next, we define the following:

Definition 1 The search space $\boldsymbol{S}$ is a set of equipment design configurations whose cardinality or |S| is $\boldsymbol{\Pi_{j=1}^{m} \Pi_{i=1}^{n_j} (ub_{ij} - lb_{ij} + 1)}$.

Definition 2 For each $\boldsymbol{X_k} \in \boldsymbol{S}$, there exists a subset $\boldsymbol{N(\theta)}$ of $\boldsymbol{S} - \{\boldsymbol{X_k}\}$ which is called the set of neighbors of $\boldsymbol{X_k}$, such that each point in $N(\boldsymbol{\theta})$ can be reached from $\boldsymbol{X_k}$ in a single transition. For example in Fig. 6, the search space S = {A, B, C, D, E, F, G} and the set of neighbors of X_k = B is $N(\boldsymbol{\theta})$ = {A, D, G}.

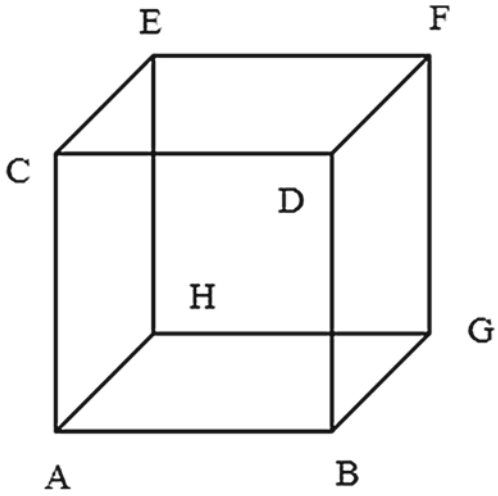


Fig. 6 Illustrates the discrete search space of a cube

6.2.1 Algorithm

Parameters

Number of times to run the simulations at design $(\mathbf{X}_k)$: $\boldsymbol{n}$
Temperature depth: T_{depth}
Temperature decay rate: r
Maximum desirable level of secondary performance measure: δ
No. of times to search a neighborhood solution at a given temperature: L
Fraction of the total search space $\boldsymbol{S}$ for obtaining initial feasible solution in %: β
Significance value for *t-test*: α

Phase I: Finding initial feasible solution

1. *feasibility* = *false*
2. $max = |S|^{*}\beta$
3. $i = 0$
4. Repeat:
   - 4.1. Randomly select the design configuration: $\mathrm{X_i} \in \mathrm{S}$
   - 4.2. Generate n simulation observations for performance measures: $\{\mathrm{f_0(X_i)}\}_{\mathrm{j=1}}^{\mathrm{n}}$, $\{\mathrm{f_1(X_i)}\}_{\mathrm{j=1}}^{\mathrm{n}}$
   - 4.3. Evaluate: $\hat{\mathrm{f}}_0(\mathrm{X_i})$, $\hat{\mathrm{f}}_1(\mathrm{X_i})$, $\hat{\sigma}_{\hat{\mathrm{f}}_0(\mathrm{X_i})}\hat{\sigma}_{\hat{\mathrm{f}}_1(\mathrm{X_i})}$ and $\mathrm{t_{n-1,1-\alpha}}$
   - 4.4. If $\hat{\mathrm{f}}_1(\mathrm{X_i}) + \mathrm{t_{n-1,1-\alpha}} \times \hat{\sigma}_{\hat{\mathrm{f}}_1(\mathrm{X_i})}/\sqrt{\mathrm{n}} \leq \delta$ then
   - 4.5. *feasibility* = *true*
   - 4.6. End if
   - 4.7. $i = i + 1$
5. Until *feasibility* = *true* or $i > max$

6. If $feasibility = true$ then
7. Return X_i as initial feasible design configuration
8. Else
9. Return initial design configuration cannot be found in *max* iterations
10. End If

Phase II: Finding optimal design solution

1. $value = \hat{f}_0(X_i)$, $T_{initial} = value/2$, $T_{final} = T_{initial}/T_{depth}$
2. Repeat:
   - 2.1. For $j = 1.....L$
   - 2.2. Randomly select the neighborhood design X_j, where $X_j \in N(X_i)$ and $N(X_i)$ is the set of neighborhood of X_i
   - 2.3. Generate n simulation observations for performance measures: $\{f_0(X_j)\}_{p=1}^{n}$, $\{f_1(X_j)\}_{p=1}^{n}$
   - 2.4. Evaluate: $\hat{f}_0(X_j)$, $\hat{f}_1(X_j)$, $\hat{\sigma}_{\hat{f}_0(X_j)}$, $\hat{\sigma}_{\hat{f}_1(X_j)}$ and $t_{n-1,1-\alpha}$
   - 2.5. If $\hat{f}_1(X_j) + t_{n-1,1-\alpha} \times \hat{\sigma}_{\hat{f}_1(X_j)}/\sqrt{n} \leq \delta$
     - 2.5.1. $newvalue = \hat{f}_0(X_j)$
     - 2.5.2. $delta = newvalue - value$
     - 2.5.3. Generate uniform random number $U_k \sim U[0, 1]$
     - 2.5.4. If $delta < 0$ or $e^{-delta}/T \geq U_k$ then
     - 2.5.5. $value = newvalue$
     - 2.5.6. $X_i = X_j$
     - 2.5.7. End If
   - 2.6. End If
   - 2.7. Next j
3. Reduce the temperature: $T = r \times T$
4. Until $T \geq T_{final}$
5. Return X_i as the optimum equipment design configuration and *value* as optimum total equipment cost value.

6.3 A Greedy Algorithmic Approach for Equipment Allocation

In this section we consider another approach in a different class from that of simulated annealing. That is an approach formulated from a greedy perspective. The greedy methodology to optimization applies a heuristic that makes the locally optimal choice at each step. Often the globally optimal solution will not be found but the greedy heuristic may yield an adequate solution in reasonable time.

Consider the case of assigning a subset of N devices to a production shop. For N devices, there are 2^N possible assignments. Even for moderate values of N this can be prohibitively large. Also each assignment may require the completion of a time-consuming simulation run since there is no analytic model capable of expressing

the production process characteristics of interest except in the simplest cases. Each run itself may need to be repeated to obtain distributions on performance metrics. Also if it is desirable to provide timely production design services via, for example, a web-based tool, then there may be additional constraints on the timely completion of a solution. The greedy algorithm initially starts with a sufficient number of devices. Next, the algorithm removes one or more devices such that the customer's performance criteria are not violated. This process is repeated until no more cost reduction is possible. Alternatively, the algorithm can start with none, or a minimal set of devices (such that each required function can be performed) and from this configuration devices can be added one or more at a time until the constrained performance criteria is achieved. The device chosen to be added or removed at each iteration is the device with the best (as detailed below) cost to benefit trade-off. The method is analogous to forward selection, backward selection, and mixed selection methods applied in the area of parsimonious model selection discussed in [9].

To illustrate the approach, we consider a shop model in the form of a discrete-event simulation that must be exercised over some duration and some job list condition. We define best performance as that with least cost with a job turnaround time metric below a specified upper constraint. The discrete-event model provides as output the turnaround metric and the costs incurred in processing the set of jobs. We will assume for the example below that the performance metric is the average turnaround time (TAT). Other performance metrics can be selected. The approach proceeds in the following steps:

1. Complete a simulation with all N machines, $\{M_1, M_2 \ldots M_N\}$, assigned to the shop. This will produce as output a turnaround time metric (*TAT*) and cost (or if runs are stochastic then the output will be in the form of distributions). Check that the *TAT* metric is below the performance constraint(s). If not then stop since there is no solution possible that would satisfy the constraint. If no solution exists then one must start with more than N machines. If a solution exists then proceed to step 2.
2. Run N more simulations. Each of the N simulations will consist of N-1 machines. For the first simulation remove M_1 and retain $\{M_2 \ldots M_N\}$, for the second simulation replace M_1 and remove M_2, so that we now retain $\{M_1, M_3, M_4 \ldots M_N\}$. Repeat until each machine has been removed in turn. From these N simulations determine the set of *TAT* metrics and cost that is output from the simulation.
3. Consider the average *TAT* metric and cost output from step 1 above, and the N from step 2. This is shown in Fig. 7 (For clarity only 4 of the N results, labeled points A, B, C, and D, are shown from step 2). Here a decision is made in which one of the machines is removed so as to reduce the set from N machines to N-1. Any point, such as point D that results in the constraint being violated is not a candidate and is to be removed. If all points lie above the constraint then no reduction in machines is permissible, and so the machine removal portion of this method is stopped. A number of sensible rules can be applied to define which machine to remove. Example performance heuristics are:

- Remove the machine which resulted in the largest cost reduction without violating the *TAT* constraint. This would correspond to point *A* in Fig. 7.
- Remove the machine which resulted in the smallest increase in *TAT* metric, this would be point *B*. Or,
- Remove the machine that resulted in the greatest cost reduction per unit *TAT* increase which would be point *C*. This would also capture the case in which a *TAT* "increase" is actually negative—a very favorable condition.

All three of the above rules can be applied in three completely independent applications of the method and the best result chosen. This will require approximately 3 times the number of runs.

4. Steps 2 and 3 above are repeated until the costs can no longer be achieved subject to the constraint. This would result in an absolute maximum $(N + 1)^* N/2$ simulations. So for a pool of 30 machines that would be 465 simulations maximum. The maximum is unlikely to be run and certain policies can be adopted to reduce further the number of simulations. For example,

   - Step 2 is likely to stop because of *TAT* constraint violations with *N* substantially larger than 1 in all except the most trivial of problems.
   - Also the assumption that the machines are unique is highly unlikely. For example, if machines M_1, M_2, and M_3 are identical then fewer simulations would be required since removing M_1 is equivalent to removing M_2 or M_3. In this way the existence of equivalent machines may reduce the required number of simulations.
   - A final way in which the number of simulations can be reduced is to use as a guide the utilization results. So, for example, after the completion of step 1, we have as output the utilization rates of the set of *N* machines. These should be ordered from low utilization to high utilization. Step 2 can begin by removing the machines in the order of lowest to highest utilization. One can then choose not to run simulations for the machines that are highly utilized.
   - Lastly, a modification that may be particularly effective would be to look at larger groups of machines to be removed and replaced. For example, with the case of 10 unique machines, instead of removing 1 machine at a time we may consider removing groups of 2 (or 3, etc…) machines at a time. Suppose we consider the removal of 2 machines at a time. Then there are $10 * 9/2 = 45$ unique 2-tuples ($10 * 9 * 8/6 = 120$ unique 3-tuples). And so 45 evaluations would be run after which 2 machines are removed that yielded the best trade-off in cost and performance. There are now 8 machines left in the pool. The number of simulations increased from 19 (10 + 9) to 45, but the method explores a larger set of possibilities and is therefore more likely to produce a solution closer to optimal. If the group size is increased from 1 to 2, then from 2 to 3, and then from 3 to 4, etc… a more extensive area of the search space is evaluated. As the group size approaches *N* we approach the state of exhaustive search. If a time constraint is set on the simulation time then one can proceed logically through configurations as outlined in this chapter until the time limit is reached.

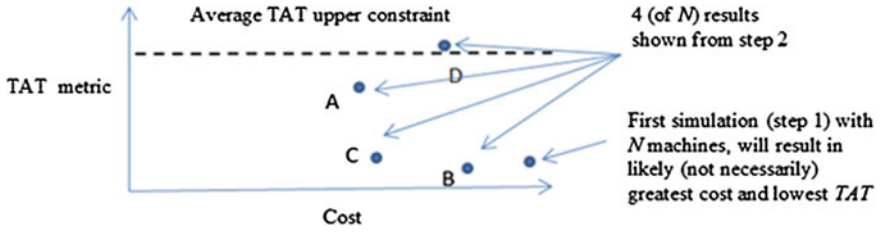


Fig. 7 Plot of *TAT* metric versus *cost*

- All the above stated approaches aim at reducing the number of required simulations. Since the operations in step 2 and step 3 can be performed independently, the method easily lends itself to parallel processing which may further reduce the computation time.

The final phase of this method can be to introduce machine failures. This final step is a more conventional approach and so it is only outlined here for completion (unlike steps 1–4 above, the final step does not constitute a core idea of this chapter). Directionally to cope with machine failures more machines may need to be added (not subtracted as in the steps above). A number of simulations are to be run and the *TAT* metric distribution estimated. Machine(s) that are bottleneck devices and/or highly utilized and therefore vulnerable failure points are identified and if their reliability levels are sufficiently low, backup machines are added until the distribution of the *TAT* metric is adequate.

7 Application and Case Study

Print service centers can be classified into three categories based on the activity that they perform: transaction printing, on-demand publishing, or a combination of both. A transaction printing environment produces documents such as checks and invoices. Each document set is different. Mail metering and delivery are part of the workflow. On-demand publishing environments focus on producing several copies of identical documents with more finishing options such as cutting, punching, and binding. Examples of such products include books, sales brochures, and manuals. Other environments perform both types of document production simultaneously with varying emphasis on each one.

In this section we illustrate the selection of the equipment configuration in three print service centers using existing, simulated annealing, and greedy algorithmic approaches for different performance criteria's. The total equipment cost is deterministic and defined $C_{ij} \times x_{ij}$ as where, C_{ij} is the fixed cost of ith equipment in the jth cell and x_{ij} is the number of ith type of equipment in the jth cell. The print service center performance measure $f_1(\boldsymbol{X_k})$ is problem specific and can only be estimated by running simulations using the LDP toolkit.

7.1 Print Service Center 1

This print service center has two cells and six stations and can process printing and inserting job workflows. Table 1 shows the equipment in each cell and their fixed cost.

Job data over a period of 10 days is collected from the print service center with a total of 2692 jobs during that period. The number of equipment of each function/station type in a cell is varied between 1 and 3 and so the total number of possible equipment configuration is 729. Table 2 illustrates a sample of all the possible equipment configurations.

Next, we illustrate the selection of optimal or near-optimal equipment configurations for the print service center using the existing approach with N equal to 30 (N is the number of simulations replications for each design configuration), simulated annealing approach with the parameters $n = 5$, $L = 5$, $T_{depth} = 100$, $r = 0.9$, $\beta = 5\,\%$ and $\alpha = 0.01$, and greedy algorithm starting initially with a solution having 3 number of equipment of each type in each cell for two test cases.

7.1.1 Test Case 1

In this problem, we have consider the print service center performance measure $f_1(\boldsymbol{X_k})$ as the average turnaround time less than or equal to 5 h. The average turnaround time is defined as the arithmetic average of turnaround times (difference between the completion time and arrival time of job) of all the jobs. Table 3 illustrates the results summary.

Table 1 The printing equipment in each cell and their fixed cost

Cell	Station	Fixed cost ($)
Cell one	Printer A	2,448,874
Cell one	Inserter A	423,366
Cell one	Inserter B	1,443,304
Cell one	Printer B	2,448,874
Cell two	Printer C	3,000,000
Cell two	Inserter B	1,443,304

Table 2 A sample of equipment configuration in print service center 1

Design no	No. of printer A's in cell one	No. of inserter A's in cell one	No. of inserter B's in cell one	No. of printer B's in cell one	No. of printer C's in cell two	No. of inserter B's in cell two
1	1	1	1	1	1	1
2	1	1	1	1	1	2
.	.	.	.	.	.	.
729	3	3	3	3	3	3

Table 3 Test case 1 results summary

	Existing approach	Simulated annealing		Greedy algorithm	
		Run1	Run2	Run1	Run2
Printer A's in cell one	1	1	1	1	1
Inserter A's in cell one	1	1	1	1	1
Inserter B's in cell one	3	3	3	3	3
Printer B's in cell one	1	1	1	1	1
Printer C's in cell two	1	1	1	1	1
Inserter B's in cell two	2	2	2	2	2
Optimal total station cost ($)	15,537,634	15,537,634	15,537,634	15,537,634	15,537,634
Average turnaround time (h)	4.8	4.75	4.82	4.78	4.78
Number of simulations	21870	1120	1115	72	72
Time in hours	29.94	1.44	1.66	0.106	0.11

7.1.2 Test Case 2

In this problem, we have considered the print service center performance measure $f_1(\boldsymbol{X_k})$ as number of late jobs less than or equal to 0. A print job is late if the completion date exceeds the due date. Table 4 illustrates the results summary.

7.2 Print Service Center 2

This print service center has 4 cells and 70 stations and can process job workflows having printing, cutting, binding, punching, and other finishing and mailing services. The search for the optimal equipment configuration is performed only for the printing equipment in the print service center. Only two cells in the print service have printing equipment. Table 5 shows the printing equipment in each cell and their monthly fixed costs.

Job data for a period of 20 days is collected from the print service center with 2593 jobs in the period. The number of equipment of each type in a cell is varied between 1 and 3 and the total number of possible equipment configuration is 2187.

Table 4 Test case 2 results summary

	Existing approach			Simulated annealing		Greedy algorithm	
				Run1	Run2	Run1	Run2
Printer A's in cell one	1	1	1	1	1	1	1
Inserter A's in cell one	2	3	1	1	3	3	3
Inserter B's in cell one	2	2	2	2	2	2	2
Printer B's in cell one	1	1	1	1	1	1	1
Printer C's in cell two	2	2	2	2	2	2	2
Inserter B's in cell two	2	2	3	3	2	2	2
Optimal total	17,517,696	17,941,062	18,537,634	18,537,634	17,941,062	17,941,062	17,941,062
Station cost ($)							
No of late jobs	0	0	0	0	0	0	0
No of simulations		21,870		1105	1120	64	64
Time in hours		29.94		1.77	1.50	0.094	0.095

Table 5 The printing equipment in each cell and their fixed cost

Cell	Station	Monthly fixed cost ($)
Cell one	Printer A	1601
Cell one	Printer B	6771
Cell one	Printer C	3907
Cell two	Printer D	6771
Cell two	Printer E	1544
Cell two	Printer F	2472
Cell two	Printer G	2120

Next, we illustrate the selection of optimal or near-optimal equipment configurations for the print service center using the existing approach with N equal to 5, the simulated annealing approach with parameters $n = 5$, $L = 5$, $T_{depth} = 100$, $r = 0.9$, $\beta = 5\,\%$ and $\alpha = 0.01$, and the greedy algorithm starting initially with a solution having 3 number of equipment of each types in each cell for two test cases.

7.2.1 Test Case 3

In this problem, we have considered the print service center performance measure $f_1(\boldsymbol{X_k})$ as the average turnaround time less than or equal to 2 h. Table 6 illustrates the results summary.

7.2.2 Test Case 4

In this case, we have considered the print service center performance measure $f_1(\boldsymbol{X_k})$ as the maximum turnaround time less than or equal to 48 h. The maximum turnaround time is defined as the maximum value of turnaround times over all the jobs. Table 7 illustrates the results summary.

7.3 Print Service Center 3

This print service center has two cells and four stations and can process job workflows having printing, and inserting. Table 8 shows the printing equipment in each cell and their monthly fixed costs.

Job data over a period of 30 days is collected from the print service center with a total of 2833 jobs in the period. The number of equipment of each type in a cell is varied between 1 and 8 and the total number of possible equipment configuration is 4096.

Table 6 Test case 3 results summary

	Existing approach		Simulated annealing		Greedy algorithm	
			Run1	Run2	Run1	Run2
Printer A's in cell one	1	2	1	1	2	2
Printer B's in cell one	1	1	1	1	1	1
Printer C's in cell one	1	1	1	1	1	1
Printer D's in cell two	1	1	1	1	1	1
Printer E's in cell two	2	1	2	2	1	1
Printer F's in cell two	1	1	1	1	1	1
Printer G's in cell two	1	1	1	1	1	1
Optimal total station cost (20 days)	$17,822	$17,860	$17,822	$17,822	$17,860	$17,860
Avg turnaround time (h)	2.0	1.94	1.98	2.0	1.91	1.93
Number of simulations	10,935		1105	1110	61	61
Time in hours	58.76		5.55	5.53	0.316	0.305

Next, we illustrate the selection of optimal or near-optimal equipment configurations for the print service center using the existing approach with N equal to 5, simulated annealing approach with the parameters $n = 5$, $L = 5$, $T_{depth} = 100$, $r = 0.9$, $\beta = 5\,\%$ and $\alpha = 0.01$, and greedy algorithm starting initially with a solution having 8 number of each equipment type in each cell for two test cases.

7.3.1 Test Case 5

In this problem, we have consider the print service center performance measure $f_1(\boldsymbol{X_k})$ as the average turnaround time less than or equal to 5 h. Table 9 illustrates the results summary.

Table 7 Test Case 4 results summary

	Existing approach		Simulated annealing		Greedy algorithm	
			Run1	Run2	Run1	Run2
Printer A's in cell one	1	1	1	1	1	1
Printer B's in cell one	1	1	1	1	1	1
Printer C's in cell one	1	1	1	1	1	1
Printer D's in cell two	1	1	1	1	1	1
Printer E's in cell two	2	1	2	2	2	2
Printer F's in cell two	1	1	1	1	1	1
Printer G's in cell two	1	2	1	1	1	1
Optimal total station cost (20 days)	$17,822	$18,206	$17,822	$17,822	$17,822	$17,822
Max turnaround time (h)	41.52	44.30	40.80	41.22	40.97	41.75
Number of simulations	10,935		1110	1110	67	67
Time in hours	58.76		5.58	5.67	0.33	0.35

Table 8 The printing equipment in each cell and their fixed cost

Cell	Station	Monthly fixed cost ($)
Cell one	Printer A	19,156
Cell one	Printer B	3907
Cell two	Inserter A	21,267
Cell two	Inserter B	11,485

7.3.2 Test Case 6

In this case, we have considered the print service center performance measure $f_1(\boldsymbol{X_k})$ as the maximum turnaround time less than or equal to 48 h. Table 10 illustrates the results summary.

Table 9 Test case 5 results summary

	Existing approach		Simulated annealing		Greedy algorithm	
			Run1	Run2	Run1	Run2
Printer A's in cell one	8	8	8	8	8	8
Printer B's in cell one	1	2	2	2	1	1
Inserter A's in cell two	1	1	1	1	1	1
Inserter B's in cell two	5	5	5	5	5	5
Optimal total station cost ($)	235,847	239,754	239,754	239,754	235,847	235,847
Average turnaround time (h)	4.95	4.88	4.9	4.92	4.97	5
No of simulations	20,480		1155	1165	67	67
Time in hours	57.41		1.58	1.83	0.103	0.092

Table 10 Test Case 6 results summary

	Existing approach		Simulated annealing		Greedy algorithm	
			Run1	Run2	Run1	Run2
Printer A's in cell one	5	5	5	5	5	5
Printer B's in cell one	4	2	4	2	4	4
Inserter A's in cell two	1	2	1	2	1	1
Inserter B's in cell two	3	2	3	2	3	3
Optimal total station cost ($)	167,130	169,098	167,130	169,098	167,130	167,130
Max turnaround time (h)	47.82	47.38	47.79	47.29	47.8	47.76
No of simulations	20,480		1145	1110	99	99
Time in hours	57.41		1.44	1.30	0.135	0.133

7.4 Results and Discussion

We have demonstrated the selection of optimal equipment in print service environments using modified simulated annealing and greedy algorithm techniques for different test cases. These test cases differ either in the performance measures or the problem size. In test cases 1 and 4, the simulated annealing and greedy algorithm finds the optimal solutions for both the experimental runs. Whereas, in test case 3 the simulated annealing outperforms the greedy algorithm solutions and in test case 5 the greedy algorithm outperforms simulated annealing solutions. In test case 2 and 6, the simulated annealing and greedy algorithm performs equally in one experimental run, but the greedy technique outperforms annealing algorithm in the second experimental run.

The results show that the greedy algorithm and simulated annealing perform adequately for a set of tasks typical in the improvement of print operations irrespective of the size of the problem. The simulated annealing technique is more time consuming and is performed offline and used during preliminary print service center cost evaluations. The simulated annealing algorithm is wrapped around the stand-alone LDP modeling framework, enabling the users to determine the optimal equipment configuration by evaluating a very large number of possible configurations automatically. In addition, by enabling automated simulation-based optimization, we can enable less skilled users to utilize the power of the LDP toolkit in making informed and optimized decisions offline. The business value of this automated simulation optimization solution can be enhanced further by incorporating this into an online web-based LDP optimization framework. As the greedy algorithm is much faster than simulated annealing, it is used in a web-based online application as shown in Fig. 8.

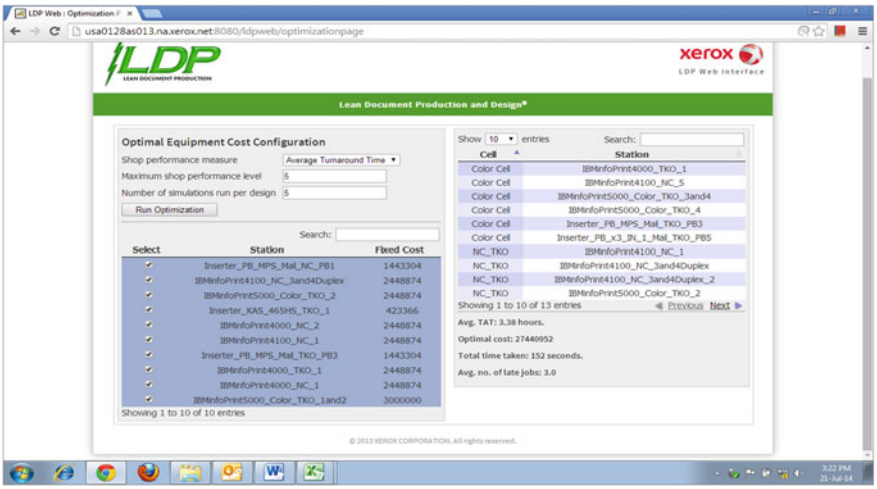


Fig. 8 Online web-based LDP tool kit

8 Conclusions and Future Work

This chapter presents a simulation-based optimization solution using simulated annealing as an offline approach and a greedy methodology as an offline or online approach for optimal print shop equipment selection. It describes how suitable abstractions and automation of the simulation tool can enable deployment of the Lean Document Production solution for cost-optimal equipment selection within a highly fragmented printing industry, while optimizing key performance objectives such as average turnaround time, number of late jobs, operator or equipment utilization, process cycle efficiency, etc. Though the techniques described here are applied within printing industry, they can also be utilized in other service-based operations with similar workflow characteristics.

Here we have used simulated annealing as an optimization approach, other evolutionary approaches such as ant colony and genetic algorithms can also be utilized for this purpose. But, these techniques need to be adapted to suit to the stochastic environments. The computational speed of these algorithms can be improved further by parallelizing, running on cloud-based platforms. We carried the above study by considering a single performance measure; further study can be made to extend the algorithm for multiple shop performance measures such as labor cost and operational cost.

References

1. Ahmed MA, Alkhamis TM, Hasan M (1997) Optimizing discrete stochastic systems using simulated annealing and simulation. Computers & Industrial Engineering 32:823–836
2. Alkhamis TM, Ahmed MA (2004) Simulation based optimization using simulated annealing with confidence interval. In: Ingalls RG, Rossetti MD, Smith JS, Peters BA (eds) Proceedings of the 2004 winter simulation conference, Washington DC, 2004
3. Andradóttir S, Goldsman D, Kim SH (2005) Finding the best in the presence of a stochastic constraint. In: Kuhl ME, Steiger NM, Armstrong FB, Joines JA (eds) Proceedings of the 2005 winter simulation conference, Florida, 2005
4. Batur D, Kim SH (2005) Procedures for feasibility detection in the presence of multiple constraints. In: Kuhl ME, Steiger NM, Armstrong FB, Joines JA (eds) Proceedings of the 2005 winter simulation conference, Florida, 2005
5. Fu MC, Andradóttir S, Carson JS, Glover F, Harrell CR, Ho YC, Kelly JP, Robinson SM (2000) Integrating optimization and simulation: research and practice. In: Joines JA, Barton RR, Kang K, Fishwick PA (eds) Proceedings of the 2000 winter simulation conference, Florida, 2000
6. Gopakumar B, Sundaram S, Wang S, Koli S, Srihari K (2008) A simulation based approach for dock allocation in a food distribution center. In: Mason SJ, Hill RR, Moench L, Rose O, Jefferson T, Flower JW (eds) Proceedings of the 2008 winter simulation conference, Florida, 2008
7. Haddock J, Mittenthal J (1992) Simulation optimization using simulated annealing. Computers & Industrial Engineering 22:387–395
8. Harkan IA, Hariga M (2007) A simulation optimization solution to the inventory continuous review problem with lot size dependent lead time. The Arabian Journal for Science and Engineering 2:327–338

9. James G, Witten D, Hastie T, Tibshirani R (2013) An introduction to statistical learning with applications in R. Springer Heidelberg, New York
10. Johnson A, Carlo HJ, Jimenez JA, Nazzal D, Lasrado V (2009) A greedy heuristic for locating crossovers in conveyor based ahms in wafer fabs. In: Rossetti MD, Hill RR, Hohansson B, Dunkin A, Ingalls RG (eds) Proceedings of the 2009 winter simulation conference, Texas, 2009
11. Kabirian A, Olafsson S (2009) Selection of the best with stochastic constraints. In: Rossetti MD, Hill RR, Hohansson B, Dunkin A, Ingalls RG (eds) Proceedings of the 2009 winter simulation conference, Texas, 2009
12. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220: 671–680
13. Luo Y, Lim E (2011) Simulation based optimization over discrete sets with noisy constraints. In: Jain S, Creasey RR, Himmerspach J, White KP, Fu M (eds) Proceedings of the 2011 winter simulation conference, Arizona, 2011
14. Prudius AA, Andradóttir S (2005) Two simulated annealing algorithms for noisy objective functions. In: Kuhl ME, Steiger NM, Armstrong FB, Joines JA (eds) Proceedings of the 2005 winter simulation conference, Florida, 2005
15. Pujowidianto NA, Lee LH, Chen CH, Yap CM (2009) Optimal computing budget allocation for constraint optimization. In: Rossetti MD, Hill RR, Hohansson B, Dunkin A, Ingalls RG (eds) Proceedings of the 2009 winter simulation conference, Texas, 2009
16. Rai S (2008) Fat tail inputs in manufacturing systems. In: Flower J, Mason S (eds) Proceedings 2008 industrial engineering research conference, Norcross, GA
17. Rai S, Duke CB, Lowe V, Trotter CQ, Scheermesser T (2009) LDP lean document production -O. R. - enhanced productivity improvements for the printing industry. Interfaces 39: 69–90
18. Sandeman T, Stanford C, Fricke C, Bodon P (2010) Integrating optimization and simulation a comparison of two case studies in mine planning". In: Johansson B, Jain S, Montoya - Torres J, Hugan J, Yücesan E (eds) Proceedings of the 2010 winter simulation conference, Maryland, 2010
19. Szechtman R, Yücesan E (2008) A new perspective on feasibility determination. In: Mason SJ, Hill RR, Moench L, Rose O, Jefferson T, Flower JW (eds) Proceedings of the 2008 winter simulation conference, Florida, 2008
20. Yue Y, Marla L, Krishnan R (2012) An efficient simulation based approach to ambulance fleet allocation and dynamic redeployment. In: proceedings of the 26th AAAI conference on artificial intelligence, Toronto, Ontario, Canada, 2014
21. Zeng Q, Yang Z (2009) Integrating simulation and optimization to schedule loading operations in container terminals. Computers and Operations Research 36: 1935–1944

Linear Bus Holding Model for Real-Time Traffic Network Control

Leonardo G. Hernández-Landa, Miguel L. Morales-Marroquín, Romeo Sánchez Nigenda and Yasmín Á. Ríos-Solís

Abstract One of the most annoying problems in urban bus operations is *bus bunching*, which happens when two or more buses arrive at a stop nose to tail. Bus bunching reflects an unreliable service that affects transit operations by increasing passenger-waiting times. This work proposes a linear mathematical programming model that establishes bus holding times at certain stops along a transit corridor to avoid bus bunching. Our approach needs real-time input, so we simulate a transit corridor and apply our mathematical model to the data generated. Thus, the inherent variability of a transit system is considered by the simulation, while the optimization model takes into account the key variables and constraints of the bus operation. Our methodology reduces overall passenger-waiting times efficiently given our linear programming model, with the characteristic of applying control intervals just every 5 min.

1 Introduction and Problem Description

The study of complex bus operating systems is usually divided into two main areas, *line planning* and *real-time control* [3, 8]. The *line planning* process involves strategic, tactical, and operational decisions. Strategic problems relate to long-term network design decisions. Tactical and operational decisions ultimately define the service offered to the public; for example, frequency of buses, definition of stops, bus

L.G. Hernández-Landa · M.L. Morales-Marroquín · R.S. Nigenda · Y.Á. Ríos-Solís (✉)
Graduate Program in Systems Engineering, Universidad Autónoma de Nuevo León (UANL), San Nicolás, Mexico
e-mail: yasmin.riossls@uanl.edu.mx

M.L. Morales-Marroquín
e-mail: morales.mike@gmail.com

R.S. Nigenda
e-mail: romeo.sanchezng@uanl.edu.mx

L.G. Hernández-Landa
e-mail: leogabrielhdz@gmail.com

M. Mujica Mota et al. (eds.), *Applied Simulation and Optimization*,
DOI 10.1007/978-3-319-15033-8_10

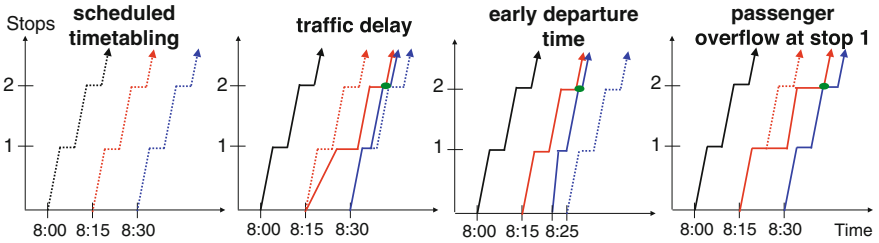


Fig. 1 Causes of bus bunching (modified from Ceder [3])

timetabling, vehicle scheduling, driver scheduling, maintenance scheduling, among other problems.

Real-time control tries to maintain the bus system operational along the day in order to minimize passenger inconvenience caused by the inherent stochastic dynamics of the network or traffic situations [8]. Although bus frequency is planned for each stop in the network, changes in the passenger flow, traffic, or even in the timetabling, produce perturbations that give rise to one of the most annoying problems in urban transportation operations, the *bus bunching problem* (BBP) that happens when two or more buses arrive at a stop nose to tail. BBP is one of the most common customer complaints in today's networks since it reflects an unreliable service that affects transit operations by increasing passenger-waiting times.

In Fig. 1, we show the causes of bus bunching for a single bus line with three trips, which have the following timetable: 8:00, 8:15, and 8:30. For the four graphs, time is represented by the x-axis, while the first two stops are represented by the y-axis. The first graph shows how the planning should look like if everything were deterministic. We can see that the lines of the three trips are *parallel*, so the time differences between them (called headways) are of exactly 15 min. The second graph shows the perturbations that arise when a traffic delay hits the second trip between the depot and the first stop. The dotted lines are the planned schedules, while the plain lines are the real executed delayed plans. Since the 8:15 bus takes longer to arrive at stop 1, there are more passengers waiting to board it. When the bus that departed at 8:30 arrives at stop 1, many of the passengers that should have boarded it have already boarded the 8:15 bus. Then, these two buses will bunch close to stop 2. Graph three represents bunching situations when the departure time of a trip is moved earlier. Similarly to the second case, there will be less passengers at stop 1 so the bus will go faster and catch the 8:15 bus around stop 2. Finally, the fourth graph considers the case of passenger overflow. This graph shows that since there are extra passengers at stop 1 the dwell time of the second bus at that stop will be longer. In other words, the second bus is taking passengers who would be normally assigned to the third bus. By the time the second and third buses arrive at stop 2, they are generating a bus bunching situation.

In this work, we provide solutions to the *bus bunching problem* by maintaining *congruent headways*. Furthermore, we show that maintaining congruent headways

implicitly reduces passenger-waiting times. As mentioned, the headway is a quality measure given to the time difference between two consecutive buses. A bus line could have equally distant headways or different ones for each pair of buses Ceder [2], Ibarra-Rojas and Rios-Solis [14], Ibarra-Rojas et al. [15]. We say that *headways are congruent* if the real-time differences between buses are nearly identical to the originally planned. Headway congruence does not necessarily comply with planned timetables. Indeed, the time when a bus arrives at a stop may not be the planned one, but if the distance to its predecessor is almost the planned headway, then it will be a congruent headway. Congruent headways reflect a reliable service, especially for cases when timetables are not intended for the public so the users only know estimated headways for the lines as in Monterrey, Mexico, and many Latino-American cities.

Our methodology interleaves optimization and real-time data retrieving to maintain congruent headways and solve BBP along the day. During the optimization phase, a linear programming model is built and solved to exactly determine the holding times of the buses at the stops in order to maintain congruent headways. The real-time data retrieving phase indicates, at every interval of time, the positions of the buses along a single corridor where only one line operates at a given frequency. In Fig. 2, we can observe how optimization and real-time data retrieving interleave. Real-time (or simulated) data are acquired from the bus corridor to obtain the distance between each bus and its last visited stop, together with the number of passengers waiting at each stop. Then, these data are used to populate our linear programming model, which yields the optimal holding times for each bus in the corridor.

Most of the works in the literature base their quality measure on the waiting times of the passengers, or the variance between the departure times of the buses at the stops, which are generally modeled with quadratic functions that are harder to solve and therefore difficult to operate by real-time systems. By using a linear objective function that minimizes the penalties arising when headways are not congruent, our methodology returns optimal solutions in a short time. One of the main contributions of this work is that by maintaining congruent headways, we implicitly reduce the overall passenger waiting and travel times, as our experimental results will demonstrate.

The remainder of this chapter is structured as follows. A brief revision of the state of the art is presented in Sect. 2. In Sect. 3, we present our new linear programming

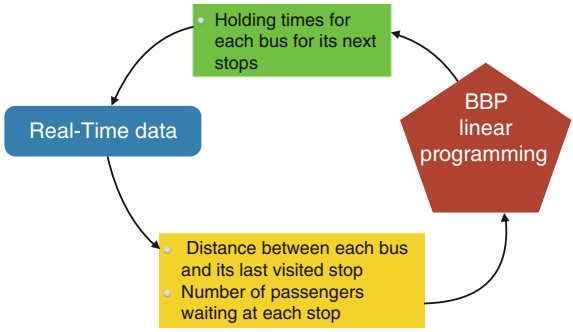


Fig. 2 Framework for interleaving optimization (BBP LP modeling) and real-time data retrieving (or simulation)

model inspired in earliness and tardiness penalties of just-in-time scheduling problems, which determines the optimal holding times of the buses at the stops. Then, Sect. 4 shows the efficiency of our model on a discrete event simulation of a single corridor. Finally, Sect. 5 presents our conclusions, and discusses open research questions that arise from this work.

2 State of the Art Research in Real-Time Bus Operations

Most of the literature related to real-time bus operations deals with models that have nonlinear objective functions. Therefore, the holding times that each bus must be held at the stations are approximations. Work by Zhao et al. [24] minimizes the average waiting cost of passengers, including both off-bus and on-bus costs that are nonlinear, when there is no capacity imposed on the buses. Eberlein et al. [11] minimize the variance between the departure times, which is a quadratic function, and therefore propose heuristic solutions. Sun and Hickman [23] propose a convex quadratic programming problem to minimize the variance between the departure times. A closer work to ours is proposed by Ding and Chien [10], since they consider the minimization of the total variance of headways between buses at all stops.

Daganzo [4] and Daganzo and Pilachowski [5] propose adaptive control schemes aiming to provide quasi-regular headways, while maintaining as high commercial speed as possible. In Daganzo and Pilachowski [5] the authors continuously adjust bus cruising speed based on a cooperative two-way-based approach that considers the headways of the previous and later buses. Bartholdi III and Eisenstein [1] abandon the idea of any *a priori* target headway, allowing headways to dynamically self-equalize by implementing a simple holding rule at a control point. It is worth noting that the aim of the previously mentioned studies is to maintain headways equally, so they do not consider timetables where the headways may be different for each pair of buses and they are not apt for situations when the buses reach their capacities.

Our work deals with the capacity of the vehicles as Zolfaghari et al. [25] do, where the authors minimize the waiting time of passengers at every stop by taking into account the variance between the departure times. These authors propose heuristics to circumvent the complexity of the proposed model. Puong and Wilson [18] propose a nonlinear mixed-integer linear programming for a real-time disruption response model with emphasis on the train holding strategy. In Delgado et al. [6, 7] the aim is to minimize the total waiting times experienced by passengers in the system using a quadratic model.

Our work aims at maintaining congruent headways considering capacity of the vehicles, and in doing so, we expect to reduce passenger-waiting times in the bus corridors. We improve the work of Delgado et al. [7] by reducing the number of variables in the model and the number of times the model is used in real-time scenarios, obtaining exact solutions for the holding times. Moreover, in order to reduce the waiting times of the passengers we bound the holding times of the buses. Another

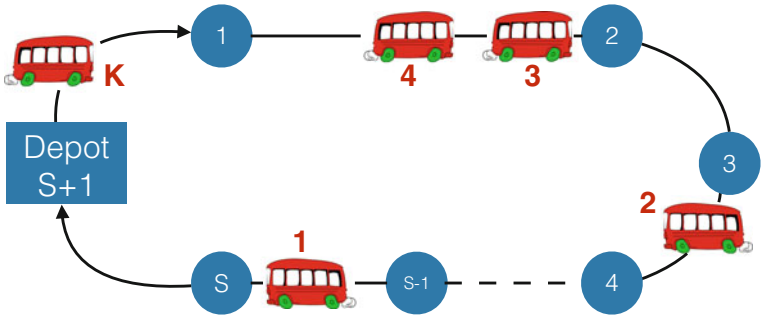


Fig. 3 Transit bus line model: each bus k leaves the depot according to an established timetable, serving stops 1 to S before coming back to the depot where all the remaining passengers must alight

advantage of our proposal is that it adapts easily to cases where the headways are equal or different during different planning horizons along the day.

3 Methodology and Approach

As mentioned earlier, the core of our methodology consists of interleaving optimization and real-time data retrieving of the bus lines in a rolling horizon planning. The optimization phase of our approach builds and solves efficiently a linear model to maintain congruent headways along the bus line. Our model is used at every given time interval[1] to decide how long the buses should be held at the bus stops. Our model requires a real-time data estimation of the state of the system to operate. Such data are provided by the real-time retrieving phase, which in our case of study is supported via simulation. The simulation of the system provides data related to the position of the buses, number of passengers aboard each bus, and the number of passengers waiting at the stops to build our model.

More precisely, the Bus Bunching Problem, BBP, consists of K buses, each with its own capacity and speed that serve all S stops of a single bus corridor. We can see in Fig. 3 that each bus k leaves the depot according to an established timetable, serving stops 1–S before coming back to the depot where all the remaining passengers must alight. Notice that overtaking is not permitted. For the optimization phase, we consider that travel times between stops, and λ_s (passengers arrival rate per minute) are deterministic during the period of interest. Moreover, each stop has a dwell time function depending linearly on the number of passengers that board ($boardT$ minutes per passenger).

The characteristics of the line are as follows. Parameter cap_k corresponds to the capacity of bus k, $dist_s$ is the distance in meters between stops s and $s-1$, $speed_{ks}$

[1] The time interval is a parameter in our model that could be specified by the control unit of the bus company.

is the operating speed in meters per minute of bus k between stops s and $s-1$ while the bus is moving, and $OD_{kss'}$ is the fraction of passengers boarding bus k at stop s whose destination is stop s' (for all $s < s'$). The headway between buses k and $k-1$ in this line must be between the interval $[minHead_k, maxHead_k]$ to be considered congruent, which is specified as an input parameter for our model.

At time t^0, instant when the holding decisions are needed, we assume that we have the following state of the transit corridor:

- d_k^0 distance between bus k and its last visited stop at time t^0. If the bus is still at a stop, then $d_k^0 = 0$.
- $s(k)$ indicates the last stop that bus k has visited at time t_0. If bus k is at stop s', then $s(k) = s' - 1$. In Fig. 3, $s(2) = 3$ and $s(3) = 1$, and to simplify the notation, $s(K) = 0$, but $s(1) + 2 = S + 1$.
- c_s^0 is the number of passengers waiting at stop s at time t^0.

Decision variables of our model are the holding times for each bus k at control point s, denoted by h_{ks}. There are auxiliary variables that depend on h_{ks}, like the departure times of bus k at stop s that is denoted as td_{ks}. If the departure times at stop s of buses k and $k-1$ are between $[minHead_k, maxHead_k]$, then we consider that they are complying with the established headways. Nevertheless, if this difference in departure times is outside this interval, we use the concepts of *earliness* and *tardiness* which is frequent in just-in-time scheduling theory [19–22]. The *earliness of the headway* between buses k and $k-1$ at stop s is defined as $E_{ks} = \max(minHead_k - (td_{ks} - td_{k-1s}), 0)$ which can be linearized as follows:

$$E_{ks} \geq minHead_k - (td_{ks} - td_{k-1s}),\ k = 2, \ldots, K, s = s(k)+1, \ldots, S \quad (1)$$

$$E_{ks} \geq 0,\ k = 2, \ldots, K, s = s(k)+1, \ldots, S. \quad (2)$$

While the *tardiness of the headway* is $T_{ks} = \max((td_{ks} - td_{k-1s}) - maxHead_k, 0)$ which is equivalent to

$$T_{ks} \geq (td_{ks} - td_{k-1s}) - maxHead_k,\ k = 2, \ldots, K, s = s(k)+1, \ldots, S \quad (3)$$

$$T_{ks} \geq 0,\ k = 2, \ldots, K, s = s(k)+1, \ldots, S. \quad (4)$$

Then, the objective function of BBH is the minimization of the sum of all early and tardy headways:

$$\min \sum_{k=2}^{K} \sum_{s=s(k)+1}^{S} \psi E_{ks} + \epsilon T_{ks}, \quad (5)$$

where ψ and ϵ are linear penalization for the earliness and the tardiness, respectively, subject to constraints (1)–(4). Additionally, the departure times of each bus k at each stop s are defined with two different sets of restrictions. The first one is the case where the bus k at time t^0 is between stops $s(k)$ and $s(k)+1$ (in Fig. 3 this case would apply for bus 2 that is between stops 3 and 4). Here, the departure time of k

at $s(k)+1$ is the time that needs the bus to arrive at the stop, plus the dwelling time $dwell_{ks(k)+1}$ (that will be computed later) plus the time the model decides that this bus will hold. This situation is reflected by constraints (6). The second case is similar but considers that the bus has not yet reached stop $s-1$ (constraints (7)). Restrictions (8) impose a limit of $maxHold$ to each holding time to guarantee a certain traveling time quality of the passengers.

$$td_{ks(k)+1} = t_0 + \frac{dist_{s(k)} - d_k^0}{speed_{ks(k)}} + dwell_{ks(k)+1} + h_{ks(k)+1}, \quad k \in K \tag{6}$$

$$td_{ks} = td_{ks-1} + \frac{dist_{s-1}}{speed_{ks-1}} + dwell_{ks} + h_{ks}, \quad k \in K, s = s(k)+2, \ldots, S-1. \tag{7}$$

$$h_{ks} \leq maxHold, \quad k \in K \backslash \{1\}, \; s = s(k)+1, \ldots, s(k-1). \tag{8}$$

From the state variables of the system, we can compute the total number of passengers that will be at stop s when bus k will reach this stop, denoted as $pass_{ks}$ in (9) and (10), as the number of passengers who are actually in the stop plus the ones that will arrive. The number of passengers who will be in bus k at stop s is equal to the passengers who want to board bus k, $pass_{ks}$, minus the proportion of the passengers that left the bus before stop s (restrictions (11)). In this manner, we can compute the dwell times of bus k at s (restrictions (12)). Notice that alighting and friction between the passengers who stay inside the bus could be easily included in the last restriction set.

$$pass_{ks} = c_s^0 + \lambda_s(td_{ks} - t_0), \quad k \in K, \; s = s(k)+1, \ldots, s(k-1) \tag{9}$$

$$pass_{1s} = c_s^0 + \lambda_s(td_{1s} - td_{Ks}), \; s = s(K)+1, \ldots, S \tag{10}$$

$$passBus_{ks} = \min\left(\sum_{i=1}^{s-1} pass_{ki}\left(1 - \sum_{j=i+1}^{s-1} OD_{kij}\right), cap_k\right), \quad k \in K, \; s = s(k)+1, \ldots, S \tag{11}$$

$$dwell_{ks} = passBus_{ks} boardT, \quad k \in K, s = s(k)+1, \ldots, S. \tag{12}$$

The following restrictions are the different cases that need to be considered in order to avoid bus overtaking:

$$td_{ks} - td_{k-1s} \geq 0, \quad k \in K \backslash \{1\}, \; s = s(k-1)+1, \ldots, S \tag{13}$$

$$td_{1s} - td_{Ks} \geq 0, \quad s = s(k)+1, \ldots, s(1) \tag{14}$$

$$td_{k-1s} - td_{ks} \geq 0, \; k \in K \backslash \{1\}, \; s = s(k)+1, \ldots, s(k-1). \tag{15}$$

The LP for BBP is then

$$\begin{aligned} \min \quad & \sum_{k=2}^{K} \sum_{s=s(k)+1}^{S} \psi E_{ks} + \epsilon T_{ks} \\ \text{s.t.} \quad & \\ & E_{ks}, T_{ks}, h_{ks} \geq 0, \quad k \in K, s \in S. \end{aligned}$$

Notice that all variables are required to be positive but not integer, so LP can be solved by the simplex method or by a polynomial barrier algorithm. Indeed, the main variables h_{ks} represent a time interval so we can consider them as continuous variables. One of the main advantage of LP, besides the fast computational time, is that we could use linear programming sensitivity analysis. Nevertheless, the holding times that are going to be transmitted to the drivers at the bus stations should be integer. Then, variables h_{ks} should be in seconds or in minutes and therefore integer variables. Preliminary results showed no drastic increment in the computing times when bus holding variables h_{ks} are integer [1, 4, 6, 15].

Our model improves and differs from the model of Delgado et al. [7] in the following aspects.

- Our objective function is linear so we can obtain optimal solutions for our model.
- The departure times of the buses are according to their established headway or timetable. Only perturbations that arise along the trip are taken into account.
- We only take into account the possible holding times of a bus from its actual position up to the depot instead of considering the holding times for all stops. This reduces the number of variables and makes the problem more realistic.
- We bound the amount of time that a bus can be held at a stop.
- We may have different headways for every pair of buses. In this way, recent synchronization timetables can be benefited by our approach and dealing with different planning periods (e.g., rush hour, night time) is natural.
- We do not need to call the model every time a bus arrives at a stop, we can do it at each fixed interval of time. This fact is more realistic for a bus company. In our case of study, the company retrieves data of the buses every 2 min.

4 Experimental Results

The BBP LP model described in the previous section needs data to be populated. Data can be retrieved through the use of monitoring technologies like Global Positioning Systems (GPS) and Automatic Vehicle Location systems (AVL) in real-time during the execution of the bus corridor. However, to study the impact of our model under different scenarios in the traffic corridor we consider a discrete event simulation.

The single corridor is simulated using the discrete event and stochastic simulator ExtendSim AT version 9.0 [9, 17]. The simulator triggers an event at every fixed amount of time, in which the positions of the buses and their loads, and the passengers waiting at the stops, together with their traveling destinations, are updated.

Our BBP LP model uses deterministic functions to forecast demands and travel times. Nevertheless, we use stochastic processes in the simulation to reflect a real system. We use a single corridor of 10 km with 30 stops and one depot uniformly distributed, like in Delgado et al. [7]. There are only 30 stretches, since the last stop is merged with the depot. Travel times of the buses between each pair of stops are distributed as Lognormal with a mean of 0.77 min and variance of 0.4 [13, 24]. At each stop, passengers arrive randomly using a Poisson distribution with rate equal to one [16]. The mean of the distributions are the parameters used by our model.

When passengers arrive at a bus stop, a destination is assigned to them. Passengers wait in line to board the bus in a first-in/first-out manner. Boarding and alighting times of passengers are set to 2.5 and 1.5 s respectively, since all buses have two doors, one for boarding and another for alighting. If passengers cannot enter a bus because it reached its capacity, they will wait in the stop until the next bus with free space arrives. This waiting time is denoted as W_{first}. The headway time windows are set to $[minHead_k, maxHead_k] = [0.3, 0.46]$ minutes for all the buses. Note that these time windows are easily adjustable for cases where there are different periods along the day, and for the synchronization timetables that favor transfers. We can measure the waiting and travel times of the passengers and the buses in the simulation since we have modeled these structures as individual agents.

We use a fleet of 60 buses with a maximum capacity of 100 passengers per bus. At every fixed amount of time *interval*, we determine the actions that should be followed by creating the BBP LP model in Java, and solving it with the linear package of Gurobi 5.6. The solution generated contains the holding times for all the buses for all the future stops up to the depot. If after a time *interval* a new solution is generated, then the holding times are updated using a rolling horizon scheme.

Even if we base our scenarios on the ones generated by Delgado et al. [7], there is no fair comparison since our methodologies consider different assumptions. Nevertheless, we can observe that our approach indeed improves the overall waiting and travel times of the passengers.

The scenarios for the simulation are divided into two parts: *time interval* scenarios and the *parameters setting* scenarios; and they are described in the following subsections.

4.1 Time Interval Scenarios

The aim of the time interval scenarios is to determine the optimal policy for controlling when new holding times must be computed and given to the system.

In our case study for the city of Monterrey, México, the bus company updates at every 2 min the positions and all the related data of the buses in the transit corridor. Following this policy, Table 1 shows the time interval scenarios in which we test our approach. The first column in Table 1 identifies the scenarios while the second column sets the time intervals (in minutes) in which our BBP LP model is constructed and solved to introduce the resulting holding times to the system. We vary these control

Table 1 *Time interval* scenarios with earliness and tardiness penalties $\psi = \epsilon = 1$

Scen	Control (min)	$maxHold$ (min)	W_{first} (min)	Travel (min)	Pass	W_{first}/ pass	Travel/ pass
TI_0	✗	✗	1798.0	12035.8	1713.3	1.0	7.0
TI_1	2	✗	1115.88	17045.40	1703.1	0.66	10.01
TI_2	5	✗	1136.92	18256.20	1746.2	0.65	10.45
TI_3	7	✗	1222.66	18907.13	1705.8	0.72	11.08
TI_4	10	✗	1362.68	18652.68	1708.5	0.80	10.92
TI_5	2	0.38	1219.52	13112.15	1721.4	0.71	7.62
TI_6	5	0.38	1330.89	13171.48	1737.4	0.77	7.58
TI_7	7	0.38	1463.25	12851.01	1725.6	0.85	7.45
TI_8	10	0.38	1424.52	12450.34	1697.7	0.84	7.33

values from 2 to 10 min. Scenario TI_0 does not have any control, and we use it as a baseline to compare the performance of our BBP LP model. The third column is an indicator if restriction (8) is applied; that is, if the holding times are bounded. For these scenarios, we set the earliness and the tardiness penalties $\psi = \epsilon = 1$. The fourth column, W_{first}, corresponds to the total average waiting time (in minutes) of a passenger to board a bus. The fifth column (Travel) represents the total average travel time of passengers in minutes, while the column Pass indicates the average number of passengers in the system during the simulation time. The last two columns indicate the normalized waiting and travel times of each passenger.

Ten simulation runs were executed for every scenario, each of them corresponding to one hour of bus operations. Each run has the same initial conditions initialized with random numbers. At the beginning of the simulation the buses are placed evenly spaced along the corridor. For each simulation run, we let the system evolve freely for 5 min before making any holding. Indeed, 5 min is enough to observe several bus bunching situations to arise.

We observe an increase in the passenger riding time, and potentially operation costs because of the introduction of holding times in the corridor. This behavior is expected, and in concordance with other works [12]. Nevertheless, the passenger-waiting times for the first bus are always reduced, which in fact is what we wanted to show in the first place. Indeed, by controlling the headway we can also control the passenger-waiting times, without the need for using a quadratic objective function in the model.

We can also observe that the best passenger-waiting times are for cases where the holding controls are applied every 2–5 min, and without the bounds on the holding times. However, the bounds on the holding times induce a reduction in the travel times, which is an important asset. Figure 4 shows the differences in performance when the control (8) ($maxHold$) is applied. It shows the percentage of increase in the passenger-waiting times when bounds are applied and the percentage of increase in the travel times when they are not applied. As mentioned, we observe that even if there

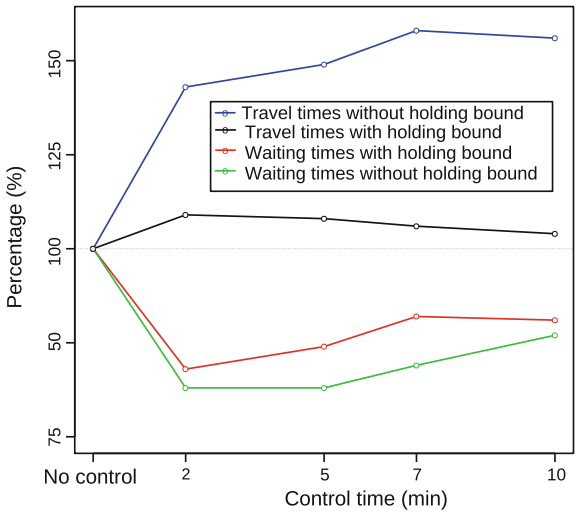


Fig. 4 Decrease in the waiting times and increase in the travel times for the *time interval* scenarios with earliness and tardiness penalties $\psi = \epsilon = 1$ and applying bound to holding time

is an increase in the passenger waiting times when the holding times are bounded, the benefit on the passenger travel times is considerable. Then, maintaining congruent headways reduces the overall travel time of passenger along the whole network.

For a bus company, the less the traffic controller has to give holding orders to the system (i.e., to the bus drivers), the better. Therefore, from Table 1 and Fig. 4, we conclude that the best policy is to consider bounds on the holding times, and apply the controls to the system at every 5 min, like in the TI_6 scenarios.

In Fig. 5, we show two histograms of the length of the holding times (x-axis in minutes) for the time interval scenarios with earliness and tardiness penalties

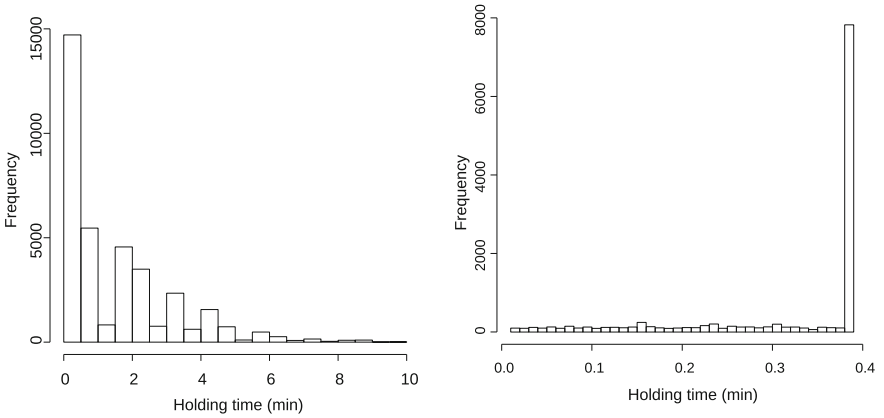


Fig. 5 Holding times histogram without bounds (*left* histogram) and with bounds (*right* histogram) for the *time interval* scenarios with earliness and tardiness penalties $\psi = \epsilon = 1$, and a control of 5 min

$\psi = \epsilon = 1$, and a control of 5 min with and without bounds on the holding times. On the y-axis, we have the frequency the BBP LP model is called for all the simulations of class TI_6. Notice that not all of the holding times are applied, since the rolling horizon may modify several of them. The case when there are limits on the holding times shows that the model either chooses to apply the holding times close to these limits, or not to apply them at all. This is an implicit benefit for the users, and for the traffic controller.

The aim of the BBP model is to reduce bus bunching by maintaining congruent headways. To graphically show that this behavior is being improved by our model, we present Figs. 6, 7, 8 and 9 for the scenarios with bounds on the holding times. The x-axes in these graphs correspond to time (in minutes), while the y-axes represent stops. Each line in these graphs represents a bus that departs from the depot and cruises all the bus stops. Recall from Sect. 1 (see Fig. 1) that in the ideal case, we

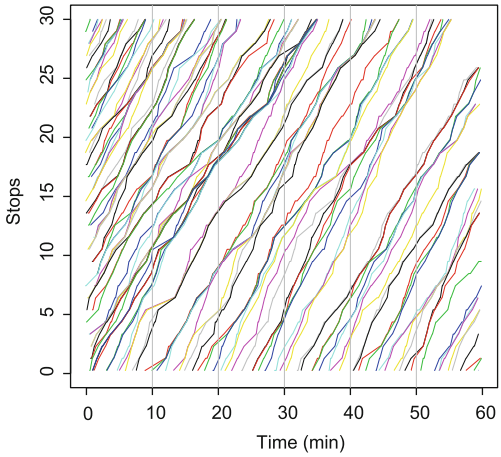


Fig. 6 Bus transit behavior without control

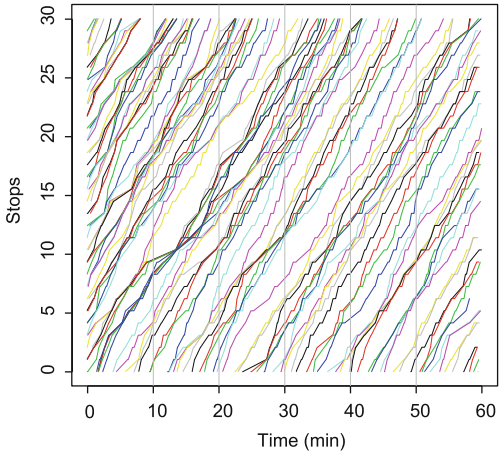


Fig. 7 Transit with control every 2 min

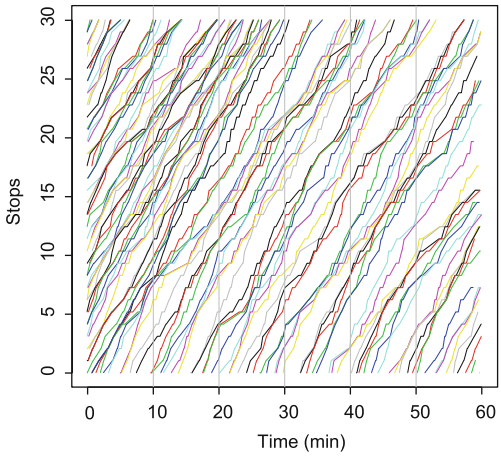


Fig. 8 Transit with control every 5 min

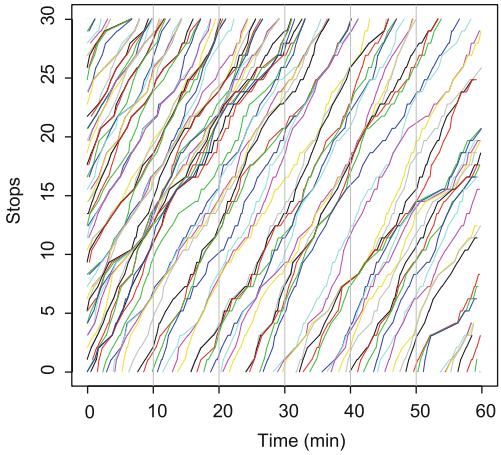


Fig. 9 Transit with control every 7 min

would have *parallel* lines. Figure 6 displays the case without control and shows that the simulation makes a stochastic scenario. Here the bus bunching problem is notorious, since there are white gaps between the lines. Figures 7, 8 and 9, have time interval controls of 2, 5, and 7 min, respectively. We can observe that with 2 and 5 min controls the BBP is reduced, while for control intervals of 7 min the BBP appears again.

Figure 10 shows two histograms that have in their x-axes the round time of a bus trip. An aspect that we noticed from Table 1 is that the travel times increase with the BBP model. This is obvious because the BBP model introduces holding times for the buses in the corridor. Nevertheless, Fig. 10 shows that the standard deviation when BBP is applied every 5 min (right histogram) is reduced with respect to the case where no controls are used (left-hand side histogram).

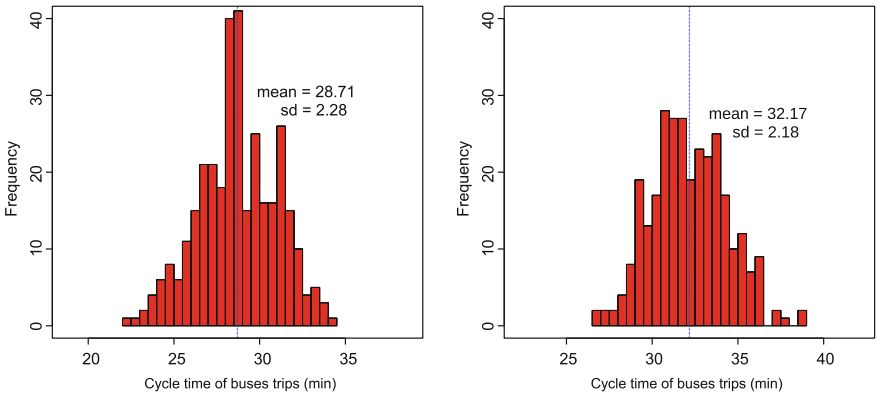


Fig. 10 Histogram of travel cycle without control (*left*) and with control interval of 5 min (*right*)

4.2 Parameter Setting Scenarios

Our next set of experiments modify the earliness ψ and tardiness ϵ parameters of the BBP LP objective function to observe the impact they have in the passenger-waiting times and travel time. We can see this set of experiments in Table 2. The first column in the table identifies the scenarios. Ten simulation runs were considered per scenario. The second column represents the values of the earliness parameter, while the third corresponds to the tardiness one. The column "Board" denotes the average time (in seconds) a passenger takes to board a bus, while $maxHold$ stands for the time (in minutes) that the holding times are bounded. This table shows the percentage of reduction in passenger-waiting times (W_{first}), and the percentage of increase in the travel times (Travel). Finally, the last column represents the addition of the last two values. Indeed, if there is a reduction in this last column, the percentage would be negative.

An interesting observation from these results is that if we reduce the earliness parameter, we obtain the best results with respect to the passenger-waiting and travel times. Moreover, the BBP LP model yields better results when the holding times are limited by 0.19 min, which is also a quality asset for the user.

A statistical analysis confirms the observations from Table 2. The most influential parameters are the earliness penalty and the $maxHold$ limit. In Table 3, we show a linear regression of the parameters studied in this section. The first column is the parameter, the second corresponds to the "Estimate", the third is the standard error, the fourth stands for the t value, and the fifth one is the significance.

Table 2 Improvement in the behavior of waiting time and travel time managing parameters

Scen	ψ	ϵ	Board (sec)	$maxHold$ (min)	W_{first} % reduction (%)	Travel % increase (%)	W_{first} + Travel % increase (%)
P_1	0	1	1.25	0.19	19	−2	−4
P_2	0	1	1.25	0.38	22	1	−2
P_3	0	1	2.5	0.19	22	−4	−6
P_4	0	1	2.5	0.38	25	−1	−4
P_5	0.5	1	1.25	0.19	34	10	4
P_6	0.5	1	1.25	0.38	55	39	27
P_7	0.5	1	2.5	0.19	37	11	5
P_8	0.5	1	2.5	0.38	56	41	28
P_9	1	0	1.25	0.19	40	11	5
P_{10}	1	0	1.25	0.38	59	42	29
P_{11}	1	0	2.5	0.19	44	12	5
P_{12}	1	0	2.5	0.38	63	48	34
P_{13}	1	0.5	1.25	0.19	36	10	4
P_{14}	1	0.5	1.25	0.38	54	41	28
P_{15}	1	0.5	2.5	0.19	39	11	5
P_{16}	1	0.5	2.5	0.38	57	40	27
P_{17}	1	1	1.25	0.19	39	11	5
P_{18}	1	1	1.25	0.38	48	27	17
P_{19}	1	1	2.5	0.19	39	56	43
P_{20}	1	1	2.5	0.38	57	50	36

Table 3 Linear regression on the main parameters of the BBP model

	Estimate	Std. error	t value	Pr(> |t|)
(Intercept)	1.0314	0.0928	11.11	0.0000
ψ	−0.2234	0.0489	−4.57	0.0004
ϵ	0.0273	0.0489	0.56	0.5848
Board	−0.0845	0.0646	−1.31	0.2106
$maxHold$	−0.2956	0.0646	−4.57	0.0004

5 Concluding Remarks

In this paper, we presented a methodology based on interleaving optimization and real-time retrieving data to maintain congruent headways in a bus corridor with the aim of solving one of the most annoying problems in public transit networks, the Bus Bunching Problem (BBP).

During the optimization phase of our approach, a linear programming model is built and solved to determine the optimal holding times of the buses at the stops to avoid bus bunching. Our model requires real-time data of the state of the system to operate. Such data is provided by the real-time retrieving phase of our approach, which in our case is supported via simulation. The simulation phase of the system provides data related to positions of the buses, number of passengers in the buses, current bus capacities, and number of passengers waiting at the stops to build our model.

One of the main advantages of considering simulation in our methodology is the evaluation of multiple parameters to assess their impact in our BBP linear programming model. Therefore, we presented a comprehensive evaluation of such parameters, and found that applying holding controls just every 5 min, and bounds on the holding times reduce not only bus bunching frequency but also passenger-waiting times.

We also discussed that most of the works in the literature minimize passenger waiting times, or the variance in the departure times of the buses using quadratic optimization functions, which are more complex to solve. Instead, the linear programming model of our approach makes it suitable for returning optimal solutions efficiently and for interleaving the optimization and real-time retrieving data phases in real-time scenarios.

Although we observe an increase in the travel time of passengers given the introduction of holding times for the buses in the corridor, our approach performs better (i.e., less passenger-waiting time and acceptable travel time) than not introducing any control into the system. A part of our future work will consider the introduction of other actions into our models to reduce the travel time of the passengers in the corridor and lower operational costs. Particularly, we believe that the introduction of bus overtaking actions (i.e., skipping stops) will balance the total time a passenger spends in the system.

Acknowledgments L.G. Hernández-Landa and M.L. Morales-Marroquín wish to thank the Mexican National Council of Science and Technology (CONACyT) for graduate scholarship support.

References

1. Bartholdi III JJ, Eisenstein DD (2012) A self-coördinating bus route to resist bus bunching. Transportation Research Part B: Methodological 46(4):481–491
2. Ceder A (2001) Bus timetables with even passenger loads as opposed to even headways. Transportation Research Record: Journal of the Transportation Research Board 1760(1):3–9
3. Ceder A (2007) Public Transit Planning and Operation: Theory, Modeling and Practice. Elsevier, Butterworth-Heinemann
4. Daganzo CF (2009) A headway-based approach to eliminate bus bunching: Systematic analysis and comparisons. Transportation Research Part B: Methodological 43(10):913–921
5. Daganzo CF, Pilachowski J (2011) Reducing bunching with bus-to-bus cooperation. Transportation Research Part B: Methodological 45(1):267–277

6. Delgado F, Muñoz JC, Giesen R, Cipriano A (2009) Real-time control of buses in a transit corridor based on vehicle holding and boarding limits. Transportation Research Record: Journal of the Transportation Research Board 2090(1):59–67
7. Delgado F, Munoz JC, Giesen R (2012) How much can holding and/or limiting boarding improve transit performance? Transportation Research Part B: Methodological 46(9):1202–1217
8. Desaulniers G, Hickman M (2007) Public transit. Transportation, Handbooks in Operations Research and Management Science pp 69–127
9. Diamond B, Krahl D, Nastasi A, Tag P (2010) Extendsim advanced technology: integrated simulation database. In: Proceedings of the Winter Simulation Conference, Winter Simulation Conference, pp 32–39
10. Ding Y, Chien SI (2001) Improving transit service quality and headway regularity with real-time control. Transportation Research Record: Journal of the Transportation Research Board 1760(1):161–170
11. Eberlein XJ, Wilson NH, Bernstein D (2001) The holding problem with real-time information available. Transportation science 35(1):1–18
12. Furth PG, Muller TH (2007) Service reliability and optimal running time schedules. Transportation Research Record: Journal of the Transportation Research Board 2034(1):55–61
13. Hickman MD (2001) An analytic stochastic model for the transit vehicle holding problem. Transportation Science 35(3):215–237
14. Ibarra-Rojas OJ, Rios-Solis YA (2012) Synchronization of bus timetabling. Transportation Research Part B: Methodological 46(5):599–614
15. Ibarra-Rojas OJ, López-Irarragorri F, Rios-Solis YA (2015) Multiperiod synchronization bus timetabling. Transportation Science
16. Jolliffe J, Hutchinson T (1975) A behavioural explanation of the association between bus and passenger arrivals at a bus stop. Transportation Science 9(3):248–282
17. Krahl D (2009) Extendsim advanced technology: discrete rate simulation. In: Winter Simulation Conference, Winter Simulation Conference, pp 333–338
18. Puong A, Wilson NH (2008) A train holding model for urban rail transit systems. In: Computer-aided Systems in Public Transport, Springer, pp 319–337
19. Ríos-Mercado RZ, Ríos-Solís YA (2012) Just-in-time Systems, vol 60. Springer
20. Rios-Solis YA (2008) Scheduling with earliness-tardiness penalties and parallel machines. 4OR 6(2):191–194
21. Rios-Solis YA, Sourd F (2008) Exponential neighborhood search for a parallel machine scheduling problem. Computers & Operations Research 35(5):1697–1712
22. Sourd F, Kedad-Sidhoum S (2003) The one-machine problem with earliness and tardiness penalties. Journal of Scheduling 6(6):533–549
23. Sun A, Hickman M (2008) The holding problem at multiple holding stations. In: Computer-aided systems in public transport, Springer, pp 339–359
24. Zhao J, Bukkapatnam S, Dessouky MM (2003) Distributed architecture for real-time coordination of bus holding in transit networks. Intelligent Transportation Systems, IEEE Transactions on 4(1):43–51
25. Zolfaghari S, Azizi N, Jaber MY (2004) A model for holding strategy in public transit systems with real-time information. International Journal of Transport Management 2(2):99–110