OR Spectrum (2010) 32:543–567
DOI 10.1007/s00291-010-0207-2

REGULAR ARTICLE

Simulation-based optimization for discharge/loading operations at a maritime container terminal

Pasquale Legato · Rina Mary Mazza · Roberto Trunfio

Published online: 7 April 2010

Abstract The discharge/loading process of a single container ship by multiple quay cranes and shuttle vehicles moving back and forth from the quay to the yard and vice versa is focused in this paper. The core problem of this major operational issue reduces to finding the optimal assignment and optimal sequencing (schedule) of bays (jobs) processed by a fixed number of available cranes (machines). Under the classical assumption that machines have no release time and that their processing occurs with continuity, at a constant rate, in literature it has been tackled as a deterministic machine scheduling problem and formulated by integer programming as the *quay crane scheduling problem* (QCSP). Here, instead, the QCSP is viewed as a decisional step within an uncertain and dynamic logistic process where the quay cranes are the resources to be managed at the best, i.e., by minimizing the time spent waiting for each other due to conflicts, as well as the time wasted for blocking and starvation phenomena due to congestion occurring along the path from the quay area and to the stacking yard and vice versa. We present a simulation-based optimization (SO) model for this wider modeling problem with the objective of finding the schedule which optimizes a classical objective function. The search process for the optimal schedule is accomplished by a simulated annealing (SA) algorithm, while performance estimation of the overall container discharge/loading process is provided by the simulation framework as a whole. Numerical experiments on a real instance are presented for tuning purposes of the SA procedure implemented within the simulator.

Keywords Container terminal logistics · Scheduling · Simulation-based optimization · Simulated annealing

P. Legato (✉) · R. M. Mazza · R. Trunfio
Dipartimento di Elettronica Informatica e Sistemistica, Università della Calabria,
Via P. Bucci cubo 41C, 87036 Arcavacata di Rende (CS), Italy
e-mail: legato@deis.unical.it
URL: http://www.deis.unical.it/legato

1 Introduction

The forecasted growth rate of containerized trade for the future years has driven competing container terminals to enhance their individual ability in fulfilling customer demand with high standard quality service, while keeping operations lean. A natural corollary to this level of commitment is found in the daily pursuit of terminal managers to obtain a nearly seamless global system and maintain its operational efficiency. A similar goal calls for the use of systematic design and verification methodologies that can cope with the major sources of system complexity and return "reliable" measures for terminal performance such as container throughput, vessel/vehicle turn-around time and/or unproductive times.

In modern container terminals the use of Operations Research methods and models as a response to this quest is becoming a rather "large" issue. Indeed, container terminal logistics have received great interest in the scientific literature from both the theoretical and practical standpoint (an exhaustive discussion can be found in Kim and Günther 2007; Stahlbock and Voß 2008; Steenken et al. 2004). A satisfactory contribution to these complex logistic problems—arising in dynamic and stochastic real environments—often requires the combination of multiple stand-alone OR techniques to deliver overall system performance measures. This practical need may be accomplished by resorting to the methodology known as *simulation-based optimization* (SO) (Fu and Nelson 2003). SO may provide an adequate setting for supporting logistic decisions and their evaluation in a dynamic and stochastic real framework, starting from mathematical programming-based formulations. To this purpose, a modern SO framework for logistic systems must provide an easy-to-understand language for modeling policies and system dynamics, as well as a tool for describing constraints and objectives of the underlying optimization problem.

In this paper we focus on SO in order to face the *determination and evaluation of quay crane schedules* not as the isolated pursuit resulting from the pure application of an MIP formulation (QCSP); rather, scheduling decisions are embedded within the wider container discharge/loading process whose overall functionality is governed by the relationship between equipment speed and container flow in both the quay and yard stacking areas. *Crane scheduling* (i.e. which and when to assign each *group of containers* [task] from a vessel-bay to a specific *quay crane* [machine]) is envisioned as the "triggering activity" of the entire logistic process, in order to minimize a given performance measure provided that:

- a minimum security distance must be left between adjacent quay cranes (*non-simultaneity constraints*);
- some vessel holds require operation before others (*precedence constraints*);
- crane deployment occurs according to actual resource availability (*release constraints* or *time windows*);
- quay cranes moving on a common track undergo spatial restrictions during operations (*non-crossing constraints*).

Most of the work in literature is based on integer programming and focuses on branch and bound based algorithms or heuristics for minimizing a single objective function [such as the makespan in Lee et al. (2008), Ng and Mak (2006), Sammarra et al. (2007),

Zhu and Lim (2006) or crane throughput in Lim et al. (2004) or the maximum relative tardiness of vessel departures in Liu et al. (2006)] or proposes the minimization of a multi-objective function [such as the weighted sum of the makespan and the total completion time of all quay cranes in Kim and Park (2004) or the service and delay times of vessels and the standard deviation of the quay crane working times in Liang and Mi (2007)]. Clearly, whatever the objective function, it is evaluated by its closed form expression as a deterministic value. Our work is based on *discrete-event simulation* to estimate the weighted sum of the makespan and other quay crane times by sample paths returned from simulation runs and it focuses on *simulated annealing* to search for the crane schedule which minimizes the above weighted objective function. The motivating assumption of our methodological choice is that both the makespan and the crane completion times may be affected by some congestion phenomena (e.g. crane blocking and starvation, queuing and so on) and the random duration of some operations (e.g. container handling from vessel to quay and vice versa and container transfer from quay to yard and vice versa).

Moreover, the implementation of the optimum-seeking methodologies in a simulation based operational model, which is more flexible than the one achievable by integer programming formulations, is expected to enhance their usage in real applications.

The remainder of this paper is organized as follows. The first section describes the overall discharge/loading process in a maritime container terminal and the major role played by the QCSP is provided by means of a conceptual model. In the second section attention is drawn to the visual modeling paradigm used to represent the operational model according to which the simulation-based optimization will be carried-out. The following section is then devoted to describing the simulation-based optimization approach used to reproduce the overall system dynamics, with a particular focus on a multi-neighbor simulated annealing procedure used as search algorithm for the QCSP. Numerical experiments are in the last section followed by the conclusions.

2 The conceptual model

Container discharge/loading plays a primary role in the *core business* of a maritime container terminal. In this integrated process, ship-to-shore cranes, such as *rail-mounted gantry cranes* (RMGCs) or *rubber-tired gantry cranes* (RTGCs), operate in synchronicity with selected shuttle vehicles to provide container transfer from the vessel to the yard and vice versa. As in most European and North American container terminals, in the following a *direct transfer system* (DTS) based on the use of *straddle carriers* (SCs)—special vehicles designed to pick-up/set-down and transfer one or more units per time—is considered for both container transfer and handling [for a complete classification on terminal equipment see Steenken et al. (2004)]. For sake of simplicity, attention is drawn to container discharge and the corresponding *work cycle* is represented by the model in Fig. 1 (clearly, when considering container loading the order of the resource request and acquisition is reversed).

The berth *weekly plan* usually provides a number of valuable information for operation initiation among which the expected workload for each vessel and, most importantly, the number of quay cranes assigned. This stated, once a vessel is berthed and

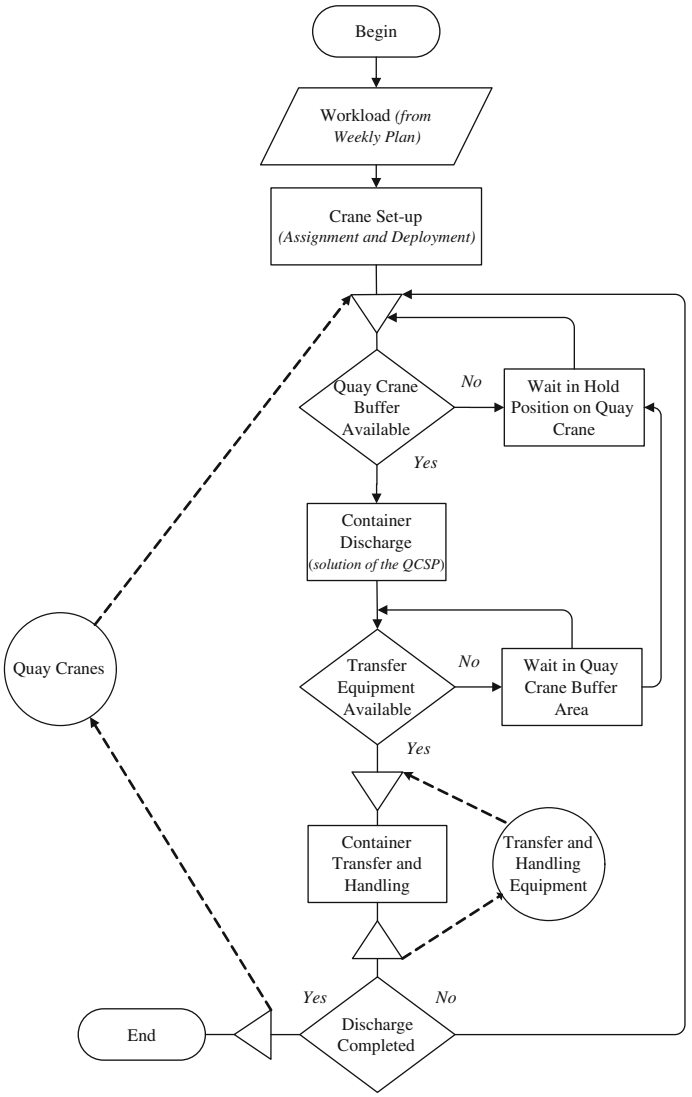

Fig. 1 Work cycle in the discharge/loading process for container stacking in the yard

properly equipped with human and mechanical resources, container discharge (loading) can be carried-out according to the schedule returned by the solution of the QCSP.

Practically, a container is discharged (loaded) by a quay crane and placed in the limited buffer area located at the feet of the crane. If this area is already at its full capacity (e.g. six slots), the container is held by the crane until a space becomes available (*wait in hold position on quay crane*). Once discharged, a container is picked up by a straddle carrier that provides for its transportation to the yard area. If a transfer vehicle is not immediately available then the container must attend in the buffer until

this occurs (*wait in quay crane buffer area*). After the container reaches the yard, it is set-down in the designated position by the straddle carrier which, as previously stated, also performs the handling movements for stacking (retrieval) purposes. The cycle continues looping until all containers have been stacked (retrieved) in the yard (loaded on the vessel).

The entire *work cycle* in the discharge/loading process described above is based on a sequence of dynamic and random-based seize-delay-release actions stemming from container flow between bordering terminal areas. In this operational and organizational context many factors may cause a non-continued coordination and interaction between equipment speed and container flow. On the quay side, a disruption or delay may be triggered by any of the constraints on the quay cranes listed in the previous crane scheduling description (i.e. *non-simultaneity constraints*, *precedence constraints*, *machine release constraints*, *time windows* and/or *non-crossing constraints*) or by the unproductive times generated by quay crane movement from one bay to another (i.e. *overhead*). On the yard side, obviously, container transfer between this sub-system and the quay area affects overall terminal performance in terms of idleness as well. For example, if the transfer activity carried out by straddle carriers from the quay to the yard is too slow, then the quay crane activity is prone to be affected by *blocking* during discharge operations due to container space that is likely unavailable in the buffer area (by *starvation* during loading operations in which case the buffer area is likely empty).

As a general result, the previous non-deterministic and non-static scenarios all lead to possible system congestion and consequential performance deterioration.

Whatever the case, operational consequences fall on the wider problem of minimizing the quay crane idle times, as requested by the managers at the real container terminal of interest.

On the strength of these considerations, we believe that an integrated methodology such as simulation-based optimization, where a heuristic search of the best crane schedule is integrated with a simulation based evaluation of the objective function, can both embody the crane scheduling issues and model the dynamics of the entire discharge/loading process across the different sub-systems of the terminal (i.e. quay and yard).

3 The operational model

The issue concerning which modeling paradigm could be the most suitable for representing operational processes in port logistics was faced by these authors when first choosing to apply an *event graph* methodology (Schruben 1983) to the sole vessel loading process (Canonaco et al. 2008). The limitations of the same modeling paradigm in terms of capability to represent constraints and decision rules (Chan and Schruben 2004) were still encountered when building an integrated simulation model for channel contention and berth management at a maritime container terminal (Canonaco et al. 2007).

Henceforth, the modeling capability of *event graphs* against alternative paradigms/languages (i.e., *Petri nets* and *hierarchical control flow graphs*) was investigated in

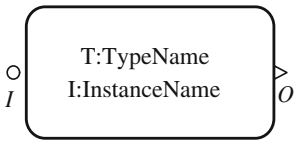

Fig. 2 The outer view of a model object in an HMP simulation model

Legato et al. (2008b), under the special purpose of constructing simulation-based optimization models for port logistics. To make this paper self-contained, the *holism-based* (Pidd and Castro 1998) *modeling paradigm* (HMP) proposed in Legato et al. (2008b) for the entire discharge/loading process of a vessel is briefly resumed here. It is aimed to guarantee the terminal manager with three major features for the resulting SO models: readability, reusability and customizability.

Model readability is achieved by describing the components' behavior within a simulation model by a sort of flow-chart. As for the reusage property, our experience at the Gioia Tauro Container Terminal confirms the requirement that a specialized simulation tool has to be reused in some of its forms (model reuse, component reuse, function reuse and code scavenging). Model reuse, under calibration and repeated tuning, occurs as soon as traffic conditions change over time. Furthermore, component and function reuse are both required to give the operations manager the possibility of quickly implementing a first order model of some emerging situations, before the structured intervention of external expertise. In this sense, a hierarchical, modular definition and redefinition of simulation parameters should be well appreciated. Finally, model customizability relies on a user-definition of process properties that allow describing uncommon situations, as it is the case when the modeler is asked to represent local, best practices in logistics organization and management.

A brief summary of our operational model is given here, but the reader may refer to Trunfio (2008) for details. Basically, it is developed around a set of model objects, or objects for short. For each model object an inner and outer view is defined. The outer view is depicted as a box equipped with input and output ports (see Fig. 2).

An HMP simulation model results as a network of model objects. Hierarchical modeling is pursued by coupling different processes and grouping the resulting network of model objects in a sub-model acting as an inner model or an outer one.

The conceptual model of the overall discharge/loading process already described within the flowchart of Fig. 1 is now translated into the HMP operational model shown in Fig. 3.

It is based on coupling six types of model objects which are: (i) a *Crane* to represent the quay RMGC resource, (ii) a *CraneManager* to manage the interactions among RMGCs working on the same vessel and assign the sequence of tasks of this vessel, (iii) a *ContainerLoader* to perform container pick-up operations for empty SCs, (iv) a *Queue* to represent the waiting line of empty SCs that wait for discharged containers and (v) a *FiniteQueue* to represent the buffer area under a crane (for 6 TEUs at the most).

As far as the inner view is concerned, a model object is defined as a process (i.e. a sequence of activities and events that define the model object behavior). A process is represented as a particular hierarchical flow-chart, called *event-activity diagram*

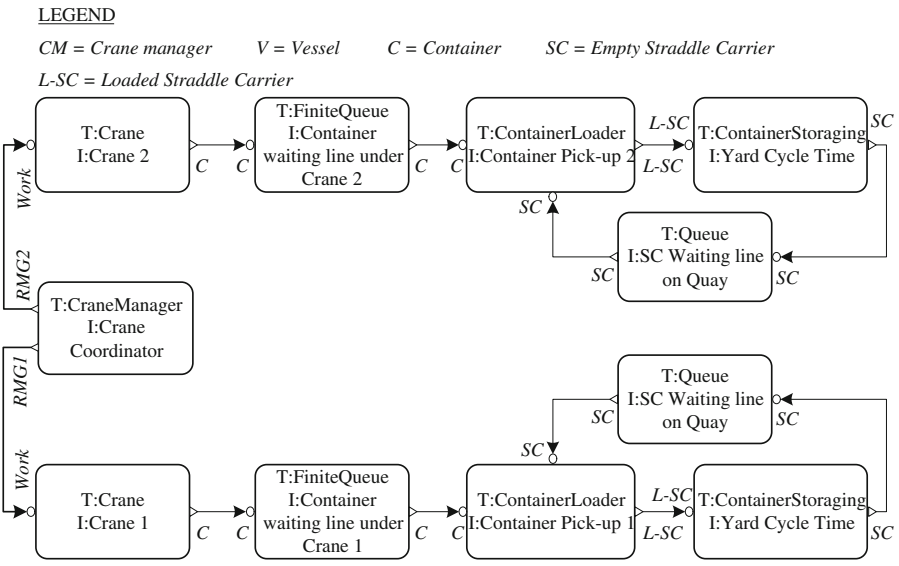

Fig. 3 The HMP model for container discharge in the overall discharge/loading process

(EAD), which allows to achieve, at a good extent, the readability objective especially during the design of model objects in conjunction with terminal experts.

3.1 Resources as model objects

Two basic classes of objects, *resources* and *resource managers*, are crucial in an SO framework. Resources are *active* or *passive* depending on their role in the simulation model. Passive resources are not depicted explicitly and are not able to execute action/events or process entities. Nevertheless, a passive resource is able to execute incoming requests and actions of other model objects. Passive resources are generally managed by an active resource. An active resource can possess passive resources and it can offer a service to one or more entities per time. It can also make queries to other objects to which it is linked by message passing via input/output ports. The (active) resource manager is a special high-level object which is able of interacting with a set of model objects, including other resource managers (coordination action) in order to produce decisions (e.g., solve a scheduling or an assignment problem, negotiate the use of a sub-system, etc.) according to rules, policies and constraints. In particular, a resource manager plays its role when resource conflicts arise during simulation (e.g., multiple quay cranes discharging/loading different bays on the same vessel).

3.2 Processes and EADs for discharge/loading operations

In our EAD flow-charting methodology for process description of the inner view of a model object, activities and events are nodes, while edges fix the logical and temporal

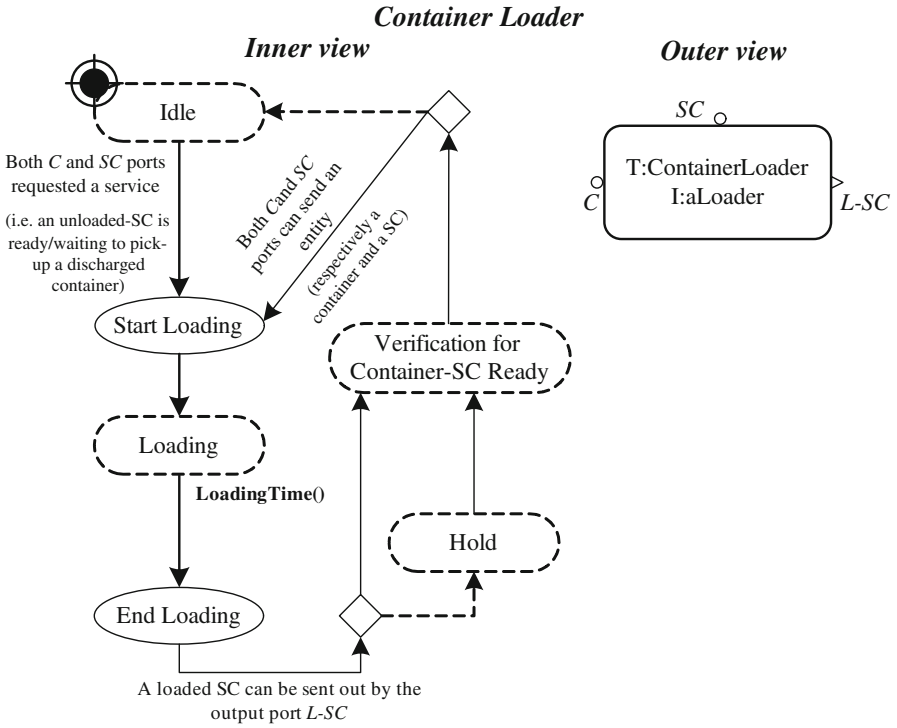

Fig. 4 The EAD for the ContainerLoader model object

sequence between nodes (i.e. other node types, such as fork and join operations). Activities and events are not intended to perform actions, while directed edges perform actions and introduce time delays between activities and/or events. An in-depth discussion about these components may be found in Legato et al. (2008b). With reference to the HMP model in Fig. 3, here we show the EAD for the container loader model object in Fig. 4.

4 The methodology

In the previous sections both the conceptual and operational models proposed to represent the overall container discharge/loading stress the random nature and variety of the underlying activities governed by uncertainty. Their solution leads to the definition of a class of problems in which whatever expected performance measure cannot be determined analytically, but must be estimated by sample paths generated via discrete-event simulation. With respect to these issues, in companion papers the authors already experimented manual-driven approaches (Canonaco et al. 2008) and joint simulation and optimization approaches (Legato et al. 2008a). As a continuation, in this paper performance estimation of the overall container discharge/loading process is provided by a simulation-based optimization framework.

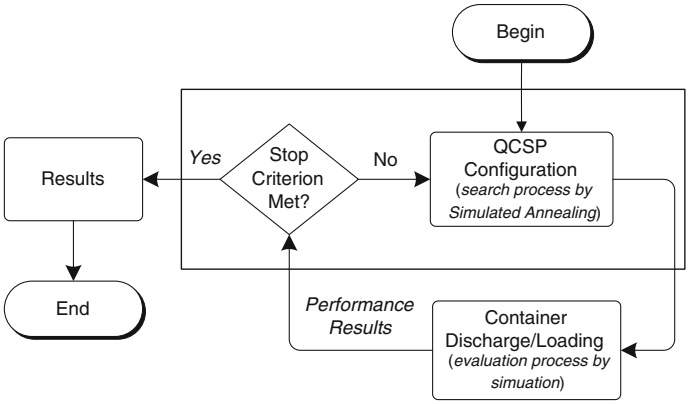

Fig. 5 Logic of the simulation-based optimization framework for the discharge/loading process

4.1 The logic of the simulation based optimization framework

Simulation-based optimization (SO) means searching for the settings of controllable decision variables that yield the minimum (maximum) expected cost (performance) of a stochastic system that is represented by a simulation model. Formally,

$$\min_{\theta \in \Theta} E\left[f(\theta)\right] \tag{1}$$

where Θ is the solution space; θ is the set of controllable decision variables; $f(\theta)$ is a random variable representing the cost/performance measure of interest; $E[f(\theta)]$ is the mathematical expectation of the cost/performance measure of interest. Practically, in the SO procedure, a structured iterative approach calls an algorithm which explores the solution space to decide how to change the values for the set of decision variables θ (i.e. the quay crane-container group assignment in the QCSP for the problem at hand) and then uses the sample mean estimate of the responses $E[f(\theta)]$ (i.e. the weighted sum of the makespan and other quay crane times) generated by the simulation runs to guide the selection of the next θ. The logic of this approach is shown in Fig. 5.

The computational expense of a single replication of the simulation model of interest is likely to be cumbersome. Consequently, the trade-off between the amount of computational time needed to find improved configurations on the optimization side (*search process*) versus the effort in estimating via simulation the performance at a particular configuration (*evaluation process*) becomes a key issue and some practical "compromises" need to be made (Banks et al. 2001). With respect to this, simulated annealing has been recently applied with success in scheduling problems related to port logistics (Kim and Moon 2003). In particular, it has been observed that schedules obtained from slower decreasing temperature schemas lead to better solutions but at the expense of longer computation times. So our feeling is that it could be worthwhile to take the chance given by the QCSP to exploit the practical possibility of designing an SA algorithm also based on a non-decreasing temperature and, as a consequence, requiring a non-classical stopping criterion.

4.2 Search process by simulated annealing

For the search process in the SO framework of Fig. 5, we specialize the variant proposed in Alrefaei and Andradóttir (1999) of the classical simulated annealing algorithm (Kirkpatrick et al. 1983).

This variant discards the basic assumption common to previous studies in which the positive control parameter or *temperature* $T_k \to 0$ as soon as k, the iteration index of the SA algorithm, goes to infinity. Thus, it works with a constant temperature $T_k = T > 0$ for all $k \in N$. By this way, the classical cooling schema—upon which the stopping criterion is guaranteed in a short time, but at the expense of the optimality of the returned solution—is lost. Under constant temperature, the stopping criterion relies on the concept of the most visited solution (Alrefaei and Andradóttir 1999) but its effectiveness is not yet consolidated for complex and large practical applications. This issue is discussed in the following. For more details the reader is referred to Mazza (2008).

By keeping a fixed value of T, the configurations that are consecutively visited by the SA algorithm can be seen as the (positive and recurrent) states of a time-homogeneous, irreducible Markov chain with transition matrix:

$$p_{ij}(T) = \begin{cases} g_{ij} \cdot a_{ij}(T) & \text{if } j \neq i \\ 1 - \sum_{k=1}^{|S|} g_{ik} \cdot a_{ik}(T) & \text{if } j = i \end{cases} \tag{2}$$

where

$$g_{ij} = \begin{cases} |N(i)|^{-1} & \text{if } j \in N(i) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

and

$$a_{ij}(T) = \min\left\{1, \exp\left(\frac{-(f(j) - f(i))}{T}\right)\right\} \tag{4}$$

are respectively the probability of generating state j from state i if j is a neighbor of i (i.e. $j \in N(i)$) and the probability of accepting j as the next state to visit.

Recalling that the stationary distribution of the underlying Markov chain is

$$\pi_i = \frac{1}{\sum_{k \in S} |N(k)| \cdot \exp^{-f_k/T}} \cdot |N(i)| \cdot \exp^{-f_i/T} \tag{5}$$

then, in terms of visit ratios,

$$\pi_j = V_i \left[\sum_{j \in S} V_j\right]^{-1} \qquad \forall\, j \in S \tag{6}$$

where V_i, the average long-run number of visits to state i, is defined as

$$V_i \hat{=} \lim_{k\to\infty} \frac{E\left[V_i(k)\right]}{k}. \tag{7}$$

Practically, given a specific sample path ω of the Markov chain, we resort to the ergodic property to estimate $V_i\ \forall i$ as follows

$$V_i = \lim_{k\to\infty} \frac{v_i(k,\omega)}{k} \tag{8}$$

and, thus, we can select the most visited configuration b as

$$b = \arg\max_{i\in S} \left\{ \frac{v_i(k,\omega)}{k} \right\}, \quad k >> 0. \tag{9}$$

Similar approaches are also pursued by other authors in Ahmed and Alkhamis (2002, 2004).

Each of the these SA variants based on a constant temperature is proved to converge almost surely to the set of global solutions as k, the number of iterations required by the SA algorithm, goes to infinity. On the strength of these methodological results, a major focus of this paper consists in analyzing the SA behavior with constant temperature, rather than the more consolidated decreasing temperature version when dealing with a complex and real problem such as the QCSP. In particular, the constant temperature approach needs to be carefully examined when it is cumbersome to carry out a large number of visits to every state of the system and estimate the best performing configuration. As a matter of fact, if n is the number of fixed tasks to be operated for discharge/loading operations and $n_i \geq 0$ is the number of tasks assigned to crane $i, i = 1, \ldots, m$ ($\sum_{i=1}^{m} n_i = n$ is a necessary imposition), then the state space of this particular QCSP is very large and equal to the number of unordered partitions of n tasks among m cranes (Liu 1968):

$$\text{number of states} = \binom{n+(m-1)}{(m-1)} \cdot n! = \frac{(n+(m-1))!}{(m-1)!}. \tag{10}$$

As one can observe, even a very limited number of cranes and tasks may generate difficult-to-solve combinatorial problem instances. For example, for a medium-size vessel with 10 tasks waiting to be operated by 3 cranes, the total number of possible combinations is 239,500,800. Therefore, unlike the simple $M/M/1$ queuing system proposed in Alrefaei and Andradóttir (1999) or the system with at the most 20 configurations presented in Ahmed and Alkhamis (2002), in the QCSP a very large number of visits (see (9)) to every alternative task-crane schedule may go beyond practical possibilities. Thus, a proper stopping criterion must be devised in order to obtain a reasonable amount of visits to "good" states without the burden of greater computation times.

In response to this performance-centered issue, a guided-search refinement in the constant-T SA algorithm based on a different choice of the candidate solution is investigated in the following. In particular, at iteration k, let i be the current solution in a minimization problem and m, with $m > 1$, the number of candidate neighboring configurations to be generated from the current solution. At this point, the refinement consists in defining j as $j = \arg\min_{\substack{l=1..m \\ j_l \in N(i)}} f(j_l)$, meaning that, among the m neighboring solutions $j_1, j_2, \ldots, j_m$ of configuration i, j becomes the candidate solution and $f(j)$ is the corresponding estimated value of the objective function. This refinement does not affect the solution acceptance mechanism (4) of the SA procedure. If the generation probabilities (3) of the candidate neighboring configurations are not affected as well, then the time-homogeneous property of the Markov chain is preserved and, with it, the convergence results.

Clearly, on one hand, the approach is meant to reach a globally optimum solution within a finite number of iterations by evaluating a sufficient, but not exhaustive, number of configurations. On the other, the approach requires accurate estimates for $f(j_1), \ldots, f(j_m)$ and $f(i)$, while also guaranteeing the selection of the best configuration or a configuration within δ units of the best according to a pre-defined level of confidence β. Both of these prerequisites can be met by using an n-stage Indifference-zone Ranking and Selection procedure (Goldsman et al. 2002) within the SA algorithm, as illustrated by the following pseudo-code in which $n = 2$ (Rinott 1978), or more recent procedures (Chen et al. 1997; Lee et al. 2006). The following algorithm (Mazza 2008) is reported here to make the paper self-contained.

Observe that the implementation of an SA algorithm with constant temperature must incorporate a data structure that is able of collecting all the distinct solutions visited during the search stage with the purpose of identifying the most visited solution. In our case, both the schedule and value of the corresponding objective function are memorized for each solution visited during the search process. Unlike the past, when similar pioneering approaches where marked as "non-realistic" (Roenko 1990), nowadays the increasing availability in computer memory and power enables us to compute and memorize data structures adequate for real applications.

This stated, from now we will concentrate on if and how the most visited configuration returned from the above schema can represent the "best" solution in relation to (i) a given time-budget determined by a stopping criterion that sets the number of iterations to be carried-out by the algorithm, (ii) the chosen temperature setting; (iii) the classical and more commonly used decreasing temperature-based SA procedure, (iv) a single neighbor generation; (v) a multi-neighbor generation mechanism. As a reply, in the following the performance of each option will be assessed in terms of tradeoff between quality of the solution returned and computation time required.

5 Numerical experiments

The purpose of the present section is to direct attention on the numerical evidence returned by the integrated framework previously proposed for simulating the overall discharge/loading (D/L) process in a container terminal. Our simulation-based

optimization framework has been implemented in Java 6.0 and experiments have been run on an Intel Core Duo T2400 1.83 GHz computer with 1GB of memory with the prospect of exploring the practical usefulness of the framework's theoretical foundations (see Sect. 4.2) in real-life situations. To fix ideas, let us consider an ordinary, yet complex operational scenario for the process under investigation which is commonly

---

Algorithm: Modified Simulated Annealing with T = constant

G, N, m, T, β, δ, n_0, $h(\beta, n_0) \leftarrow$ set input parameters (i.e. generation probabilities, neighborhood structure, number of candidate neighbors, temperature, confidence level, indifference-zone, number of first-stage simulation runs and Rinott's constant)

$k = 0 \leftarrow$ initialize iteration

$i_0 \in S \leftarrow$ select initial solution from feasible state space S

$V(i_0) = 1,\ V(i) = 0\ \forall i \in S \leftarrow$ mark visit

while stopping condition = false **do**

  $i = i_k \leftarrow$ set current solution

  for l = 1 **to** m **do**

    $j_l \in N(i) \leftarrow$ generate neighbor of i with probability $G(i, j_l)$

    for r = 1 **to** n_0 **do**

      $X_{j_l}(r) \leftarrow$ generate n_0 i.i.d. unbiased observations for candidate solution j_l

    end for

    $$f(j_l(n_0)) = \frac{1}{n_0}\sum_{r=1}^{n_0} f\left(X_{j_l}(r)\right),$$

    $$S^2_{j_l}(j_l(n_0)) = \frac{1}{n_0 - 1}\sum_{r=1}^{n_0}\left(f\left(X_{j_l}(r)\right) - f(j_l(n_0))\right)^2 \leftarrow$$ compute first-stage sample mean and variance of candidate solution j_l

    $$N_{j_l} = \max\left(n_0, \frac{h \cdot S_{j_l}}{\delta}\right)^2 \quad \forall\ j_l \in N(i)$$

  end for

  if $n_0 < N_{j_l}\ \forall\ j_l \in N(i)$ **then**

    for $r = n_0 + 1$ **to** N_{j_l} **do**

      $X_{j_l}(r) \leftarrow$ generate $N_{j_l} - n_0$ i.i.d. unbiased observations $\forall\ j_l \in N(i)$

18: **end for**

19: $f_k(X_{j_l}) \leftarrow$ generate $N_{j_l} - n_0$ i.i.d. unbiased observations for candidate solution j_l

20: $f(j_l(N_{j_l})) = \frac{1}{N_{j_l}} \sum_{r=1}^{N_{j_l}} f(X_{j_l}(r)) \leftarrow$ compute second-stage sample mean of candidate solution j_l

21: $j = \arg\min_{\substack{l=1..m \\ j_l \in N(i)}} f(j_l) \leftarrow$ set j as i's best neighboring candidate solution

22: $U_k \sim U[0,1] \leftarrow$ generate of a random number

23: $a_{ij}(k) = \exp\left[\frac{-[f(j(N_j)) - f(i(N_i))]^+}{T_k}\right] \leftarrow$ compute acceptance probability

24: **if** $U_k \le a_{ij}(k)$ **then**

25: $i_{k+1} = j \leftarrow$ accept the candidate solution

26: **Else**

27: $i_{k+1} = i \leftarrow$ keep the current solution

28: **end if**

29: $k = k + 1 \leftarrow$ increase iteration

30: $V(i_k) = V(i_k) + 1 \leftarrow$ mark visit

31: **end while**

32: $i^* = \arg\max_{i \in S} V(i) \leftarrow$ return the most visited solution

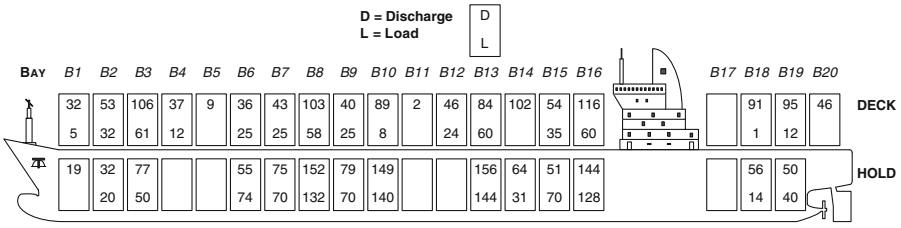

Fig. 6 The real instance evaluated within the overall discharge/loading process

triggered by the arrival of a so-called *mother* or *oceanic* vessel as the one given in Fig. 6.

The vessel map and data depicted herein (i.e. number of containers—cntrs for short—to discharge/load for every bay, whether in hold or on deck), along with more general operational specifications for both the vessel and terminal equipment as the

Table 1 Vessel characteristics and other features of terminal equipment

Length (m)	Capacity (cntrs)	Moves (cntrs)	Bays (no.)	Tasks (no.)	Prec. Constr. (no.)
Vessel characteristics					
332.4	8402	3769	20	61	42
Width (m)	Security distance (m)	Speed on rail (m/min)	Average D/L time per cntrs (min)	D/L distribution law & shape	Buffer space (no. of cntrs)
Crane characteristics					
25	12.5	15	2.45	Erlang, 32	6
Crane 1	Crane 2	Crane 3	Crane 4		
Crane release times (starting time 0:00)					
24:00	18:00	0:00	0:00		
Assigned per crane (no.)	Average handling time (min)	Handling distribution law & shape	Average cycling time (min)	Cycling distribution law & shape	
SC characteristics					
4	1.67	Erlang, 16	10.2	Weibull, 1.28	

ones listed in Table 1, are courtesy of Medcenter Container Terminal S.p.A. (MCT) in Gioia Tauro, Italy. They define the set-up profile for process initiation. It is worth observing that only the 42 precedence constraints specified in Table 1 for the QCSP set-up can be labeled as "static" and, thus, verified a priori by the framework. The remaining constraints (i.e. non-simultaneity and non-crossing) as well as other process characteristics such as crane *blocking* or *starvation* are evaluated at run-time by the simulation platform.

From the methodological side, additional specifications are required for the problem at hand. In particular, as far as solution generation via simulated annealing is concerned, we need to define the proper set-up for generating feasible schedules, meaning the list of tasks assigned to each single crane. Recalling that in the QCSP a task is defined as the discharge or loading of a group of containers related to the deck or the hold of a specific vessel bay, when generating candidate schedules, one must consider the precedence relationships among tasks related to the same bay. In this case:

– during discharge operations, tasks stowed on the deck must be performed before tasks in the hold of the same ship-bay; moreover, discharge operations must precede loading operations on the same bay;
– loading operations in a hold must precede loading operations on the deck of the same bay.

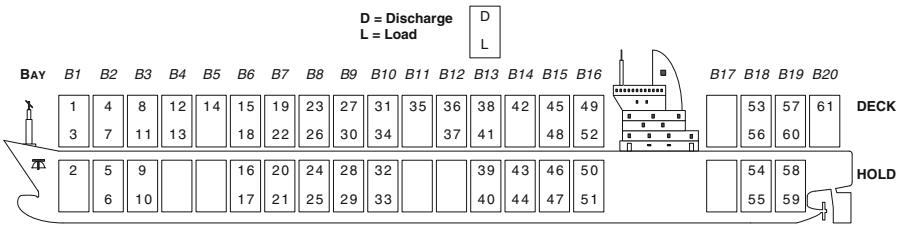

Fig. 7 Task numbering on the real instance

Table 2 Initial task assignment

Crane	Task assignment
Crane 1	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
Crane 2	19 20 21 22 23 24 25 26 28 29 30
Crane 3	27 31 32 33 34 35 36 37 38 39 40 41 42 43 44
Crane 4	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61

This stated, for convenience, tasks are numbered according to an increasing order starting from the first bay of the vessel. Figure 7 shows the task numbering for our instance. For completeness, let us add that, according to a common company practice, cranes usually operate increasing/decreasing ordered bays moving along a single direction rather than back and forward.

The schedule reported in Table 2 is the one provided by the terminal planner, i.e. the pre-stow plan resulting from MCT's current practice on scheduling cranes under a one-way transfer direction. Thus, we consider their schedule a "natural" starting point for the SA-based search process embedded in the SO framework defined upon the operational model of Fig. 3. Indeed, it will be shown by an IP based lower bound model introduced later that MCT's schedule is very close to optimality, thus acting as a very difficult term for numerical comparisons. Nevertheless, from this initial schedule, other candidate schedules will be searched according to the following move: choose a task assigned to crane i in a random way and reassign it according to a feasible order to a randomly selected crane j, where $i \neq j$.

Recalling the objective function in Kim and Park's MIP formulation (Kim and Park 2004):

$$\alpha_1 \cdot W \quad + \quad \alpha_2 \cdot \sum_{k \in Q} C_k \tag{11}$$

where W is the makespan, C_k is the completion time of crane k (out of set Q) and α_1, α_2 are properly defined weights, the simulation model depicted in Fig. 3 is used to estimate the following three components of the completion time:

– *idleness* (i.e. quay crane activity disruption)
– *overhead* (i.e. quay crane movement from one bay to another)
– *working time* (i.e. actual quay crane discharge/loading time).

It is worth to remark that the first term measures a cumulative unproductive time forced by the need to satisfy both precedence and non-simultaneity constraints in the presence of a lack of synchronicity between equipment speed and container flow. Moreover, the term *idleness* captures the effects of both *blocking* during discharge operations (due to container space that is likely unavailable in the buffer area) and *starvation* during loading operations (in which case the buffer area is likely to be empty).

On specific request of the terminal manager, we also estimate via simulation the so-called *quality of service* (QoS) defined as:

$$\text{QoS} = \frac{\sum_{k \in Q} \left[(\text{completion}\,[k] - \text{release}\,[k]) - \text{idleness}\,[k] - \text{overhead}\,[k] \right]}{\sum_{k \in Q} (\text{completion}\,[k] - \text{release}\,[k])}. \tag{12}$$

This positive performance index of a specific crane schedule is a relative quantity which expresses both an expectation and demand in customer care; thus, the greater the better.

5.1 Parameter tuning

Here the overall simulation framework is used as a tuning environment for the parameters of the simulated annealing. According to Ingber (1993), "correct" values for SA settings are usually suggested by empirical knowledge and with respect to specific disciplines. Since greater knowledge has been acquired on the SA behavior based on a decreasing temperature schema, this variant is applied straightforward by setting the initial temperature equal to 10 and the final one to 1. As for the constant temperature approach which is less consolidated, we need to further investigate the proper settings for the logistic problem at hand, while exploiting the stopping criterion behind solution generation for our QCSP.

This stated, we first search for a temperature value within the range [1; 10] that will provide us with a minimum level of stability during which the system is prone to make frequent visits to "good" states and, at the same time, enables eventual jumps out of local minima. Figure 8 is useful in searching for a fair compromise between the above features. Each trend line of the objective function (11) (measured in hours) for five different temperatures is averaged on 30 experiments. As one may observe, high temperatures (i.e. $T = \{5,\ 10\}$) are affected by high-frequency fluctuations that may likely bring the search process to continuously jump in-and-out of each state and render the most visited criteria unsuccessful. From now on, we set T equal to 1.

The second tuning option consists in selecting M, the number of candidate neighboring configurations to be generated from the current solution (see Sect. 4.2).

With respect to this issue a trade-off is required in terms of computation time and improvement in solution quality during the search process. As shown in Fig. 9, the algorithm exhibits a strong improvement of the objective function during the earlier search stage when a multiple, rather than a classical single-neighbor configuration, is adopted. This initial improvement is substantially comparable among the cases

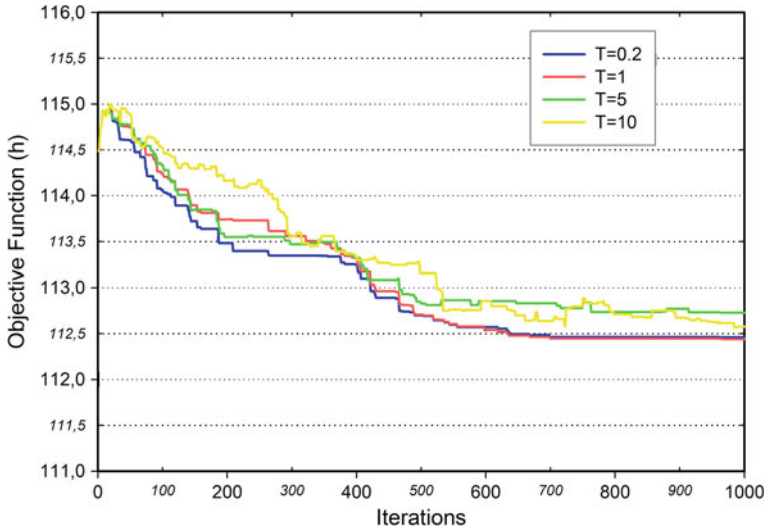

Fig. 8 The behavior of a constant temperature based SA algorithm

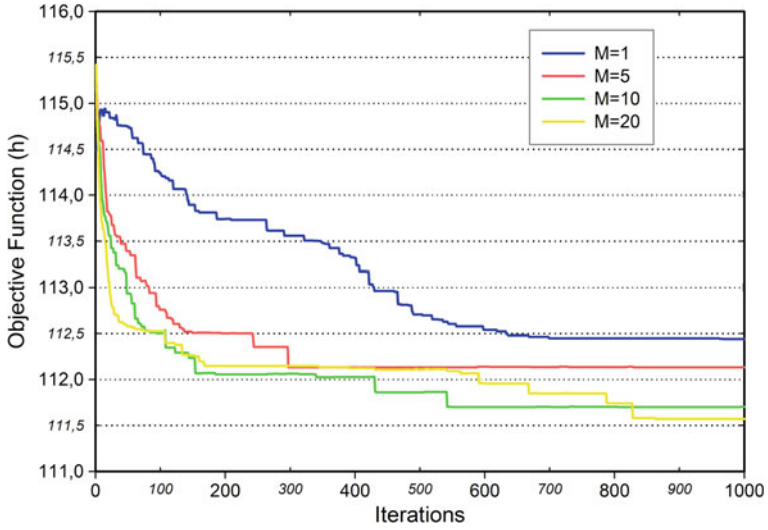

Fig. 9 The behavior of an SA algorithm featuring $T = 1$ and the generation of M neighbors

$M = 5, 10, 20$. On the other hand, computation time among these three neighborhood configurations doubles from $M = 5$ to $M = 10$ and quadruples from $M = 5$ to $M = 20$. For this reason, we set $M = 5$.

Our final considerations on parameter tuning for constant temperature based SA are devoted to verifying how "large in practice" the infinite number of iterations must be in order to guarantee both convergence to a (sub)optimal solution within a reasonable amount of time and solution quality returned under such circumstances. In the specific case at hand, we set "infinity" equal to 5000 iterations (corresponding to a time budget

Table 3 Three-way comparison between intermediate most visited solutions, most visited solutions at infinity and overall best solutions (with $M = 5$ and $T = 1$)

Intermediate algorithm run no. (N)	% most visited solution at run N is =most visited at ∞ (%)	Δ% average value of f.o. at iteration N and f.o. of best (%)	Δ% average value of f.o. at iteration N and f.o. at ∞ (%)
250	52.63	0.36	0.40
500	76.32	0.31	0.32
1000	76.32	0.16	0.25
2000	100.00	0.11	0.18
5000	–	0.01	–

of nearly 67 m) and we expect to obtain the same value for the objective function of the most visited solution at this iteration and the objective function of the best solution over all iterations (best, for short). Once this hypothesis is verified, we then investigate at what point of the simulation run the algorithm has already identified an intermediate most visited solution that is the same as the most visited at infinity. The last step consists in making a three-way comparison on the differences (in percentage over 100 different simulated search trajectories) between the average values of the objective function (11) of the three types of solutions under evaluation (i.e. intermediate most visited, the most visited at "infinity" and the "best").

As one may observe in Table 3, the search process already at an early stage (e.g. at run number $N = 500$, corresponding to about 400 s) and, thus, with a limited computational effort, returns the intermediate most visited solution which, in a significant manner (e.g. 76.32%) is the same as the most visited at "infinity". In addition, whatever the most visited solution at iteration $N = 500$, the average difference between the value of its objective function and those of the other reference solutions is practically of no consequence (e.g. 0.31% for the best and 0.32% for "infinity").

5.2 Comparisons

To better demonstrate the usefulness of the SO framework for the QCSP, here we perform some numerical comparisons regarding the schedule returned by the SA algorithm, whether with constant or decreasing temperature. In the former case we leave the constant temperature $T = 1$; in the latter, we set the initial $T = 10$, along with a $cooling\_rate = 0.999$ in the corresponding cooling schema $T_{i+1} = T_i \cdot cooling\_rate$, in order to reach $T = 1$ after 100 iterations. Accordingly, numerical results shown in Tables 4 and 5 are carried-out under 1000 SA iterations per SO experiment.

The two SA algorithms are embedded in the overall simulated, stochastic discharge/loading process (SS-D/L) described in Fig. 3. At first, both the makespan component and the total value of the objective function are compared against those obtained by implementing the SA algorithms in a simulated, but deterministic discharge/loading process (SD-D/L) whose steps are still the same ones found in the conceptual model of Fig. 3, but instanced by average values. Since the underlying, respective, schedules are expected to be different (as the variance of the process grows) we have also eval-

Table 4 Deterministic and stochastic evaluation (in h) of the D/L operational model in Fig. 3

M	SD-D/L		SS after SD-D/L		SS-D/L	
	f^*	Makespan	f^*	Makespan	f^*	Makespan
SA with decreasing T						
1	104.89	53.29	115.48	58.91	112.54	57.19
5	104.27	52.75	114.98	58.54	112.14	56.83
10	104.04	52.34	114.81	58.24	111.73	56.43
20	103.98	52.23	114.22	58.03	111.65	56.40
SA with constant T ($T = 1$)						
1	104.43	52.98	114.84	58.94	112.44	57.23
5	104.22	52.56	114.57	58.65	112.13	57.14
10	103.96	52.28	114.23	58.12	111.70	56.90
20	103.81	52.19	114.01	57.98	111.57	56.74

Table 5 Deterministic and stochastic evaluation (in h) of the simplified D/L operational model

	f^*	Idleness	Makespan	Overhead	Working time	QoS
Simplified SD-D/L						
SA with decreasing T	99.33	0.47	50.16	0.96	153.27	99.08
SA with constant T ($T = 1$)	98.87	0.10	40.82	0.86	153.27	99.38
Simplified SS after SD-D/L						
SA with decreasing T	119.97	35.90	61.94	0.97	153.24	81.42
SA with constant T ($T = 1$)	119.93	34.75	62.21	0.86	153.27	81.76

uated by stochastic simulation both the makespan and the total objective function of the schedule resulting from the SD-D/L (SS after SD-D/L)

Observe that MCT's schedule resumed in Table 2 is adopted as initial feasible schedule in these numerical experiments. Their schedule provides a f^* value of 103.01 h with a makespan of 54.04 h by SD-D/L and a f^* value of 115.1 h with a makespan of 59.85 h by SS after SD-D/L

Bearing in mind that numerical results in Table 4 are averaged over 10 experiments for each of the three variants (SS-D/L, SS after SD-D/L and SS-D/L), one may first catch the differences resulting between column SD-D/L and column SS-D/L. In particular, one may see that up to 5 h of difference, on average, are registered as induced on the makespan value by the schedule underlying the SS-D/L against the SD-D/L (constant $T = 1$ and $M = 5$). Whatever the numerical result returned by the real instance at hand, the general capability of stochastic simulation to capture the above difference is the real benefit of using the SA based search process to determine the best schedule.

For sake of completeness, the results in column SS after SD-D/L show the effect of Monte Carlo generation of the D/L times (Erlang-32 distributed), SC handling times (Erlang-16 distributed) and SC cycling times (Weibull (1.28, 10.2)) on the best

schedule returned in the case of SD-D/L. The comparison between results in the SS after SD-D/L column against the SS-D/L column leads to the conclusion that just a 1–2 h saving may be achieved on both the makespan as well as the complete objective function by the schedule returned by the algorithm which uses stochastic simulation at each step within the search process for the best schedule. In other words, for the real instance at hand, we could even save computation time by delaying the stochastic evaluation of the current schedule to the last step of the SO procedure. Nevertheless, we recommend using stochastic simulation upon each intermediate schedule whenever a higher variance of the discharge/loading process is revealed by statistical analysis on the real case of study.

At this point we wish to stress the role of the stochastic simulation of the final schedule even when a very simplified version of the HMP model in Fig. 3 is adopted as operational model of the complete discharge/loading process. To this end, suppose we relax the representation of the blocking and starvation phenomena in such a way that, for each loading task, all the related containers are assumed to be already present in an infinite-length buffer near the crane and, for each discharge task, an infinite number of SCs are assumed to be immediately available to pick-up the containers just placed in the buffer. Nevertheless, both the makespan and completion times in formula (11) are still affected by *overhead* due to crane transfer under non-crossing constraints and, more importantly, forced idle time (*idleness*) due to precedence and non-simultaneously constraints. Observe that, under these assumptions, the modeling capability of the operational model of Fig. 3 has become similar to that achieved by mathematical programming formulations (Kim and Park 2004). In our objective function (11), α_1 and $\alpha_2$have been set equal to 1 and 0.25, respectively. In Table 5, one may appreciate the contribution of stochastic simulation for a finer evaluation of the final schedule returned by the SA procedure under deterministic input values. In particular, one may observe that the contribution of idleness to the objective function is significant and can only be estimated by stochastic simulation.

Finally, let us give-up the representational capability of the HMP driven methodology in favor of a stand-alone IP formulation of a machine scheduling based version of the discharge/loading process. To get an exact deterministic solution (i.e. crane schedule) we tried to run the well-known formulation proposed by Kim and Park [op. cit.] under the following, straightforward, variables setting $X^k_{ji} = 0, \forall i,\ j \in \Omega, i < j, \forall k \in K$ to account for the one-way transfer for all the cranes. Unfortunately, computation time prevents obtaining even a feasible solution after running several hours under ILOG's CPLEX. Hence, we resorted to a lower bound model proposed in Lee et al. (2008) by adding release times on the cranes. The model is resumed in Appendix 1. The resulting (infeasible) schedule produces an objective function value of 97.62 h with a makespan of 48.82 h and it is reported here (Table 6).

By comparing this lower bound value of the objective function with the average values returned by the SO framework for the simplified version of the HMP model of Fig. 3, under deterministic instancing (i.e., expected value of discharge/loading times), as shown in column $f*$ of Table 5 one may recognize that the SO framework works very well. In particular, we remark that out of the two SA variants the best schedule (f.o. value of 98.15 h with a makespan of 49.13 h) is returned by the constant tempera-

Table 6 Best crane schedule returned by the lower bound model

Crane	Task assignment
Crane 1	5 16 23 24 27 36 46 48 56 57
Crane 2	1 8 14 20 31 35 42 43 47 52 53 60 61
Crane 3	3 4 7 9 10 17 18 22 26 28 29 30 33 39 41 50 51
Crane 4	2 6 11 12 13 15 19 21 25 32 34 37 38 40 44 45 49 54 55 58 59

Table 7 Best crane schedule returned by the SO framework for the simplified HMP model

Crane	Task assignment
Crane 1	1 2 3 4 5 6 7 8 9 10 11 17 18 22
Crane 2	12 13 16 19 20 21 24 25 26 29 30 37
Crane 3	14 15 23 27 28 31 32 33 34 39 40 41 43 44 48 52
Crane 4	35 36 38 42 45 46 47 49 50 51 53 54 55 56 57 58 59 60 61

Table 8 Best crane schedule returned by the SO framework for the complete HMP model

Crane	Task assignment
Crane 1	1 2 3 4 5 6 7 8 9 10 11 12 13 14 17 18
Crane 2	16 20 21 22 23 24 25 26 29 30 34
Crane 3	15 27 28 31 32 33 35 36 37 39 40 41 42 43 44
Crane 4	19 38 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61

ture SA variant ($T = 1,\ M = 1$) in less than 5 s against the unsuccessful several-hour effort required by CPLEX. The best schedule is reported here (Table 7).

As a further conclusive remark on numerical experiments, we highlight the contribution of the stochastic simulation of the complete operational model of Fig. 3, especially for evaluating the idleness within both the completion time and makespan, and we report in Table 8 the best schedule obtained by the SS-D/L, under SA with $T = 1$ and $M = 1$. The objective function grows to 111.41 and, in particular, the makespan to 56.12 (7 h longer than the corresponding value obtained by the deterministic approach).

In conclusion, on the strength of the above considerations, one could be confident that the proposed SO framework returns solutions that are a more than reasonable compromise between computational effort required and quality of results. Solution quality is measured in terms of both hour savings on the makespan and complete evaluation of all crane idle time components affecting vessel completion time. In the specific case at hand, one may appreciate the time improvement achieved by the SO schedule in Table 8 on the vessel schedule initially provided by the terminal planner in Table 2: a 3.7 h gain on the makespan value (56.12 h against 59.85 h). In general, if one could have gained 2–3 h on each of the 680 mother vessels served by the terminal in 2009 for which contractual agreements require the assignment of an average

number of 3 gangs (a.k.a. *crane intensity target*) at the cost of € 150.00 per hour of gang service, then potential annual savings would have ranged from €500,000.00 to € 1,000,000.00

$$\left(\cong \text{h savings} * 3\text{gang/vessel} * 150€/\text{h} \cdot \text{gang} * 680\text{vessel/year}\right)$$

6 Conclusions

We have addressed the design and implementation of a simulation optimization framework built around a well known core problem in port logistics referred as the quay crane scheduling problem. The modeling benefit of the framework is twofold. On one hand, it allows to embed a mathematical programming based approach for resource allocation and scheduling in a wider context where dynamic, random based events and duration of logistics operations occur. On the other, through simulation experiments it gives the possibility to set and tune the parameters of the metaheuristic algorithm used for problem solution of real size instances. In particular, we have investigated a simulated annealing procedure based on the less consolidated constant temperature variant and a multiple neighborhood generation mechanism and provided numerical evidence of its effectiveness in testing and improving a common company practice for assigning cranes to vessel bays and sequencing discharge/loading operations. Results encourage research efforts towards enriching the simulation optimization framework by embracing other logistic processes, among which container stacking/retrieval on the terminal yard.

Appendix 1

The lower bound model proposed in Lee et al. (2008) was properly modified to consider release times for the quay cranes. Let us consider:

Ω as the set of tasks;
Q as the set of quay cranes;
p_i as the processing time of task $i \in \Omega$;
r_k as the release time for every crane $k \in Q$;
W as the vessel completion time (makespan);
C_k as the completion time for operations performed by crane $k \in Q$;
A_i^k as the binary variable set to 1 if task $i \in \Omega$ is assigned to crane k, 0 otherwise.

A lower bound to our D/L operational model is obtained by solving the following:

$$\begin{aligned}
&\text{minimize} \quad \alpha_1 \cdot W + \alpha_2 \sum_{k \in Q} C_k \\
&C_k \leq W \quad \forall k \in Q \\
&\sum_{k \in |Q} A_i^k = 1 \quad \forall i \in \Omega \\
&C_k - r_k \geq \sum_{i \in \Omega} p_i \cdot A_i^k \quad \forall k \in Q
\end{aligned}$$

$$W \geq 0$$
$$C_k \geq 0 \quad \forall k \in Q$$
$$A_i^k \in \{0, 1\} \quad \forall i \in \Omega, \forall k \in Q$$

References

Ahmed MA, Alkhamis TM (2002) Simulation-based optimization using simulated annealing with ranking and selection. Comput Oper Res 29:387–402

Ahmed MA, Alkhamis TM (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, pp 514–519

Alrefaei MH, Andradóttir S (1999) A simulated annealing algorithm with constant temperature for discrete stochastic optimization. Manag Sci 45(5):748–764

Banks J, Carson JS, Nelson BL, Nicol DM (2001) Discrete-event system simulation, 3rd edn. Prentice-Hall, Upper Saddle River

Canonaco P, Legato P, Mazza RM (2007) An integrated simulation model for channel contention and berth management at a maritime container terminal. In: Zelinka I, Oplatková Z, Orsoni A (eds) Proceedings 21st European conference on modelling and simulation, pp 353–362

Canonaco P, Legato P, Mazza RM, Musmanno R (2008) A queuing network model for the management of berth crane operations. Comput Oper Res 35:2432–2446

Chan WK, Schruben LW (2004) Generating scheduling constraints for discrete event dynamics systems. In: Ingalls RG, Rossetti MD, Smith JS, Peters BA (eds) Proceedings of the 2004 winter simulation conference, pp 568–576

Chen HC, Chen CH, Dai L, Yücesan E (1997) New development of optimal computing budget allocation for discrete event simulation. In: Andradóttir S, Healy KJ, Withers DH, Nelson BL (eds) Proceedings of the 1997 winter simulation conference, pp 334–341

Fu M, Nelson B (2003) Guest editorial. ACM Trans Model Comput Simul 13(2):105–107

Goldsman DM, Kim S-H, Marshall WS, Nelson BL (2002) Ranking and selection for steady-state simulation: procedures and perspectives. INFORMS J Comput 14(1):2–19

Hong LJ, Nelson BL (2007) Selecting the best system when systems are revealed sequentially. IIE Trans 39:723–734

Ingber L (1993) Simulated annealing: practice versus theory. Math Comput Model 18(11):29–57

Kim KH, Günther H-O (2007) Container terminal and terminal operations. In: Container terminals and cargo systems. Springer, Berlin, pp 3–12

Kim KH, Moon KC (2003) Berth scheduling by simulated annealing. Transp Res B 37:541–560

Kim KH, Park Y-M (2004) A crane scheduling method for port container terminals. Eur J Oper Res 156:752–768

Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 221:671–680

Lee LH, Chew EP, Manikam P (2006) A general framework on the simulation-based optimization under fixed computing budget. Eur J Oper Res 174(3):1828–1841

Lee D-H, Wang HQ, Miao L (2008) Quay crane scheduling with non-interference constraints in port container terminals. Transp Res E 44(1):124–135

Legato P, Mazza RM, Trunfio R (2008a) Simulation-based optimization for the quay crane scheduling problem. In: Mason SJ, Hill R, Moench L, Rose O (eds) Proceedings of the 2008 winter simulation conference, pp 2717–2725

Legato P, Gullì D, Trunfio R (2008b) Modeling, simulation and optimization of logistics processes. In: Bruzzone AG, Longo F, Piera MA, Aguilar RM, Frydman C (eds) Proceedings of 20th European modeling and simulation symposium (Simulation in Industry), pp 569–578

Liang C, Mi W (2007) A quay crane scheduling problem by hybrid evolutionary algorithm for berth allocation planning. Comput Ind Eng 56(3):1021–1028

Lim A, Rodrigues B, Xiao F, Zhu Y (2004) Crane scheduling with spatial constraints. Naval Res Logist 51:386–406

Liu CL (1968) Introduction to combinatorial mathematics. McGraw-Hill, New York

Liu J, Wan Y-W, Wang L (2006) Quay crane scheduling at container terminals to minimize the maximum relative tardiness of vessel departures. Naval Res Logist 53:60–74

Mazza RM (2008) Simulation-based optimization in port logistics. Ph.D. Dissertation, Università della Calabria

Ng WC, Mak KL (2006) Quay crane scheduling in container terminals. Eng Optim 38:723–737

Pidd M, Castro RB (1998) Hierarchical modular modelling in discrete simulation. In: Medeiros DJ, Watson EF, Carson JS, Manivannan MS (eds) Proceedings of the 30th conference on winter simulation, pp 383–389

Rinott Y (1978) On two-stage selection procedures and related probability-inequalities. Commun Stat Theory Methods A 7(8):799–811

Roenko N (1990) Simulated annealing under uncertainty. Technical report, Institute for Operations Research, University of Zurich

Sammarra M, Cordeau J-F, Laporte H, Monaco MF (2007) A tabu search heuristic for the quay crane scheduling problem. J Sched 10:327–336

Schruben LW (1983) Simulation modelling with event graphs. Commun ACM 26(11):957–963

Stahlbock R, Voß S (2008) Operations research at container terminals: a literature update. OR Spectrum 30(1):1–52

Steenken D, Voß S, Stahlbock R (2004) Container terminal operation and operations research—a classification and literature review. OR Spectrum 26:3–49

Trunfio R (2008) Modeling, simulation and optimization in logistics. Ph.D. dissertation, Università della Calabria

Zhu Y, Lim A (2006) Crane scheduling with non-crossing constraint. J Oper Res Soc 57:1464–1471