Computers & Operations Research 52 (2014) 68–83

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Approaches for solving the container stacking problem with route distance minimization and stack rearrangement considerations

Niraj Ramesh Dayama [a,b,d,*], Mohan Krishnamoorthy [a,d], Andreas Ernst [c], Vishnu Narayanan [b], Narayan Rangaraj [b]

[a] IITB-Monash Research Academy, IIT Bombay, Powai, Mumbai 400076, India
[b] Industrial Engineering and Operations Research, IIT Bombay, Powai, Mumbai 400076, India
[c] CSIRO Mathematical and Information Sciences, Private Bag 10, Clayton South MDC, Victoria 3169, Australia
[d] Department of Mechanical and Aerospace Engineering, Monash University, Melbourne, Victoria 3168, Australia

ARTICLE INFO

Available online 24 July 2014

Keywords:
Combinatorial optimization
Crane scheduling
Container stacking
Mixed integer program
Stacker crane problem

ABSTRACT

We consider an optimization problem of sequencing the operations of cranes that are used for internal movement of containers in maritime ports. Some features of this problem have been studied in the literature as the *stacker crane problem* (SCP). However, the scope of most literature (including SCP) is restricted to minimizing the route or distance traveled by cranes and the resulting movement-related costs. In practice, cargo containers are generally stacked or piled up in multiple separate *columns, heaps* or *stacks* at ports. So, the cranes need to often rearrange or shuffle such container stacks, in order to pick up any required container. If substantial re-stacking is involved, cranes expend considerable effort in container stack rearrangement operations. The problem of minimizing the total efforts/time of the crane must therefore account for both – the stack rearrangement costs and also the movement-related (route distance) costs. The consolidated problem differs from standard route distance minimization situations if stack rearrangement activities are considered. We formally define the consolidated problem, identify its characteristic features and hence devise suitable models for it. We formulate several alternative MIP approaches to solve the problem. We compare the performance of our MIP formulations and analyze their suitability for various possible situations.

1. Introduction

The assignment and scheduling of cranes for container movement operations have been well studied in the context of intermodal freight container transportation at cargo container terminals (see [42]). Optimization models have often been applied to improve overall performance and efficiency in terms of turnaround time or throughput [8,45]. Specifically, the *stacker crane problem* (SCP) addresses the problem of minimizing the cost/time incurred when vehicles of unit load carrying capacity are deployed to pickup and deliver containers between specified locations [39].

Major cargo terminals handle a large number of containers, so space constraints often compel that containers be stacked up or piled on top of each other in *stacks*, *columns* or *heaps*. Stacking occurs at storage yards (where containers are stored, often for long periods of time) or at the berth areas (the quay where cranes load or unload containers from ships). Stacking eventually leads to additional, non-trivial costs whenever container stacks are subsequently rearranged to fetch a container that was piled under other containers. However, discussions of crane scheduling and SCP in the existing literature neglect the impact of stacking on operational efficiency and schedules [23,30].

Stack rearrangement efforts are distinct from the horizontal movement activities performed by the cranes while physically moving the containers along pathways in the terminal. The total cost incurred in container handling operations is the sum of the (vertical) stack rearrangement costs and the (horizontal) movement cost. We deal with the sequential ordering of containers, so as to minimize the overall handling costs with a focus on the unified vertical-horizontal cost minimization. The underlying problem can also be extended to more general cases (like industrial warehouses) and other examples in which stacks of objects need to be efficiently rearranged by forklifts. We do not address side constraints like time window restrictions.

The internal movement of containers within cargo terminals involves a variety of operations that need to be executed. We illustrate this using Fig. 1. This figure shows containers stacked at locations ($\zeta_1 \ldots \zeta_6$) in a cargo terminal. The figure shows the initial

* Corresponding author at: IITB-Monash Research Academy, IIT Bombay, Powai, Mumbai 400076, India.
E-mail address: niraj.ramesh@iitb.ac.in (N.R. Dayama).

http://dx.doi.org/10.1016/j.cor.2014.06.018
0305-0548/

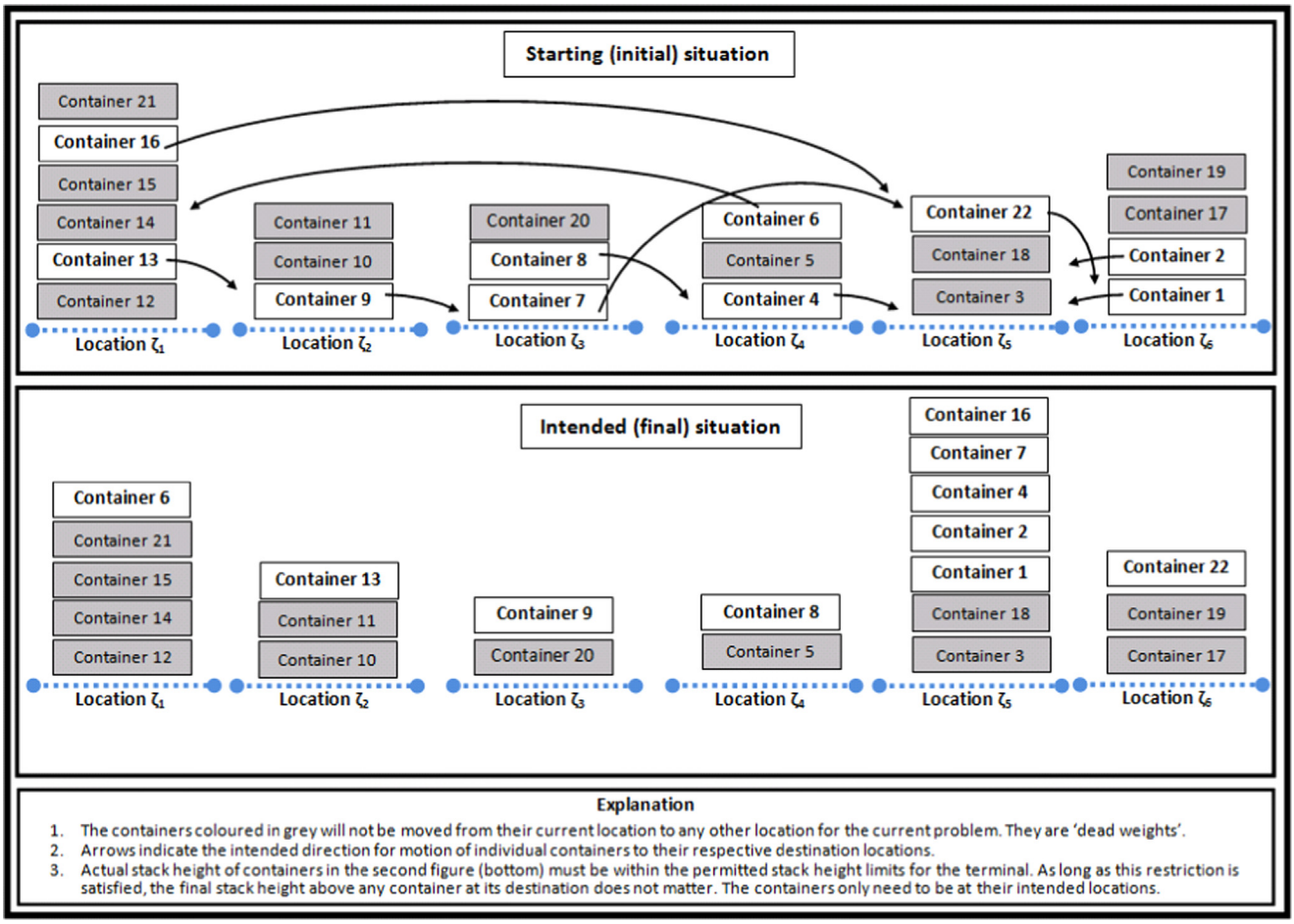

Fig. 1. Sketch showing required rearrangement of containers for a typical SpRP data instance.

stack arrangement and the desired final arrangement of containers at these six locations. A single crane is allocated to execute all the horizontal and vertical rearrangement tasks that are required. Consider container 1, which is placed in a stack under containers 2, 17 and 19 at location ζ_6. Container 1 needs to be moved to a new location ζ_5 within the terminal. The crane deployed for this activity must do the following:

1. The empty crane must move from its current location to the pickup point ζ_6 of container 1. We call this effort as *no-load horizontal motion effort* (NLHM). NLHM involves a sequence-dependent cost of horizontal movement because it depends on the *immediate* precedence sequence of containers handled by this crane.
2. The crane must now rearrange the stack at location ζ_1 to remove the containers 19, 17, 2, which are stacked or piled above the required container 1. We call this effort as the *vertical stack rearrangement effort* (VSR). VSR also depends on the sequence of containers handled by this crane. But it is radically different from NLHM, in the sense that, VSR for any container j depends on the *cumulative* effect of *all* containers handled before this specific container j.
3. The crane carries container 1 to its destination location ζ_5. We call this as *full-load horizontal motion effort* (FLHM). FLHM involves a fixed cost of horizontal movement.
4. The crane must drop container 1 at the top of the stack at destination location ζ_5. We neglect the cost and efforts of doing this activity. The final stack position of 1 at ζ_5 is not an issue for the purpose of the current discussion.

The total cost of container relocation is the summation of the immediate-precedence dependent costs (NLHM), the cumulative-sequence dependent costs (VSR) and the fixed costs for all given containers (FLHM). We denote the problem of minimizing this summation over all containers as the container *stacking plus routing problem (SpRP)*. For cargo terminals handling 100+ containers daily (see [31,33]) and employing container stacks around 8–10 high (see [32]), substantial savings can be realized using crane operational schedule that do account for VSR in conjunction with NLHM. This is the motivation for our study of SpRP, in which we extend the SCP, that has been studied up until now, to also include the non-negligible VSR cost too.

NLHM is modelled by transforming the problem (see [30,39]) into an asymmetric traveling salesman problem as follows: we model the containers as nodes to be visited on a graph, with asymmetric arc costs between any pair of nodes $i,j \in N$ equal to the distance (say E_{ij}) between the corresponding locations of those containers. The horizontal motion is modelled via a directed complete graph $G_h(\hat{N},\hat{E})$. The $n+1$ nodes $\hat{N}:=\{0,1,2,\ldots,n\}$ of this graph denote the location 0 and all n containers in N. The edges $\hat{E}$ include all directed edges needed to connect any pair i,j of nodes from $\hat{N}$. The cost of any directed edge i,j in $\hat{E}$ is the distance E_{ij}. A Hamiltonian cycle over $\hat{N}$ construes a feasible operating sequence for handling all containers. To minimize NLHM cost, we search for the Hamiltonian path which minimizes the cost of all edges traversed. For a given SpRP instance $\mathbb{X}$, we define the graph $G_h^{\mathbb{X}}$ as its *horizontal graph*. Such a modeling of NLHM is a standard practice in SCP literature (see [30,39]).

To discuss VSR costs from Fig. 1, consider container 7. It needs to be moved from its pickup location ζ_3 to delivery location ζ_5. We neglect dead weight container 20 for our discussion. But container 8 is stacked above 7 at ζ_3. Also, container 9 is supposed to be moved into location ζ_3. Consider that the crane has already handled container 9 some time before handling 7, but container 8 has not yet been handled. Now, if container 7 is fetched, there exist containers 8 and 9 above container 7. These two containers need to be temporarily removed from the stack of ζ_3 before fetching container 7. After container 7 is taken out of the stack, these containers must be placed back on location ζ_3 in their original order. Thus, the additional stacking cost of two containers is incurred while fetching container 7 from ζ_3.

On the contrary, consider that the crane handled container 8, then 7 and finally 9. In this case, neither 8 nor 9 appear above 7 when 7 is to be fetched. So the VSR cost for 7 is zero (neglecting the dead weight of container 20). So, the sequence of container handling for 8 and 9 decides the VSR cost for container 7. Generalizing this to other locations, the placement or removal of some container j impacts the VSR cost of other containers at a later time.

1.1. Problem definition

Consider n unrelated identical containers to be moved (without pre-emption) within a known time horizon. These n containers are piled up in columns or stacks. There may be some more containers present in the stacks (other than the n containers that need to be moved). These additional containers act as dead weight in the stack handling. For any container i, which is one of the n containers handled, the initial pickup location P_i and the final destination location D_i are known. The physical distance between locations P_i and D_i is denoted as C_i. For a pair of containers i,j, the distance from delivery location D_i to pickup location P_j is denoted as E_{ij}. The initial number of containers in the stack at P_i above container i is denoted as H_i. However, the actual number of containers above i changes whenever more containers may be dropped onto or taken out from above i.

A single crane is available to move the n containers. The crane is parked at a special location 0 (*the depot*) at the beginning of the time horizon and must be returned to this depot location 0 by the end of the time horizon. This crane is the only resource capable of executing all actions needed for SpRP. Any action involved in NLHM, FLHM or VSR expends an effort that translates into a proportional cost. We define these costs as follows:

1. The cost incurred by crane to travel horizontal distance of one unit is λ_h (irrespective of NLHM or FLHM). If the crane has placed container i at location D_i. Then it moves to pickup location P_j of container j. For this, it travels distance E_{ij} as NLHM effort and incurs cost $\lambda_h \times E_{ij}$. Then, the crane will carry container j from location P_j to location D_j traversing distance C_j as FLHM. This incurs cost $\lambda_h \times C_j$.
2. The total cost incurred by the crane in removing and then replacing one container from a stack (while rearranging containers) is λ_v. Consider that container j currently has h_j containers above it in the stack. If j is fetched, VSR cost of $\lambda_v \times h_j$ must be incurred. This VSR cost is independent of the actual location P_j and any containers below j in the stack.

Factors λ_h, λ_v are fixed for the port/terminal and are assumed as initial parameters for SpRP. Then, any SpRP data instance involves the following parameters:

1. $P \longrightarrow (n \times 1)$ vector of those locations that are identified as the initial source or pickup locations for the n containers.
2. $D \longrightarrow (n \times 1)$ vector of those locations that are identified as the delivery destination (or drop locations) for the n containers.
3. $H \longrightarrow (n \times 1)$ vector of initial stack heights of containers piled up in the stack at pickup point P_i above container i (defined for all containers i among the original n containers). *The actual number of containers stacked above i may change later on, whenever other containers are moved.*
4. $C \longrightarrow (n \times 1)$ vector of horizontal distances between pickup point P_i and delivery point D_i for any containers i.
5. $E \longrightarrow (n \times n)$ vector of horizontal distances between delivery point D_i of ith container and pickup point P_j of jth containers, for all pairs of containers i,j among the original n containers. E need not be symmetric.

We define term E_{0j} as the vector of horizontal distances between location 0 and the pickup location P_j of jth container. Similarly, we define term $E_{i\,0}$ as the vector of horizontal distances between the delivery location D_i of ith container and the location 0. Finally, for convenience of discussion, we define set $N = \{1, 2, \ldots, n\}$ as the unordered set of all n containers to be handled.

1.2. Assumptions

While studying the SpRP, we make the following simplifying assumptions:

1. *Time aspect*: We neglect any time windows restrictions or deadlines for container handling operations. We will use an aspect of time in several MIP formulations. We assume that the time expended in executing a particular activity is numerically equal to the cost incurred or effort invested in doing that activity. For example, if the VSR effort in handling container i is $\lambda_v \times h_i$, we assert that $\lambda_v \times h_i$ units of time were needed by the crane to fetch container i from its stack. So, minimization the total time needed in completing all activities for any SpRP instance leads to minimization of total cost.
2. *Staging areas*: There exist some small temporary staging areas near the pickup locations where the containers might be placed during stack rearrangement. Consider that a crane is tasked with fetching container i from some location ζ. But container j is stacked over i. So, the crane will first take out and place j into the staging area. Then the crane will pick and place i next to j. Thereafter, it will replace j on the stack at location ζ. Finally, it will pick i and start moving towards D_i. The staging area is to be used during the stack rearrangement activities only and must be vacated as soon as possible.
3. *Preparatory rearrangement for containers*: No crane can do any anticipatory or preparative rearrangement for any container or stack. Also, operations for a given container cannot be pre-empted or interrupted. For example, with reference to Fig. 1, suppose that a crane is scheduled to handle container 9 first, immediately followed by containers 7 and 8 will be handled much later. Then we assume that the crane will first deliver container 9 above the stack at location ζ_3 above containers 7 and 8. This completes the required actions for container 9. Thereafter, the crane will begin the operations for container 7. For fetching container 7, the crane will take off containers 9, 20 and 8 temporarily, place them in staging area, then fetch container 7, then replace containers 9, 20 and 8 on ζ_3 and finally move away with container 7. Specifically, the crane will not interrupt the delivery of container 9 to facilitate the future retrieval of container 7 from ζ_3.
4. *Final resulting stack height at delivery locations*: Suppose that container i is being delivered to its drop location D_i. During or after this delivery operation, the stack height of containers at D_i above or below i has no impact on costs of i. The exact final

sequence of i in the stack at D_i does not matter for SpRP. The terminal operators and stevedores may need to know this specification for future operations, but it does not affect the cost/time/effort for SpRP.

The second and third assumptions need further explanations: these assumptions are made based on practical concerns and limitations faced in ports. First, there may often be multiple cranes operating in close proximity to each other. So, if one crane starts using additional areas (beyond the identified staging areas) or undertakes any anticipatory rearrangement of containers, this can affect the movement of other cranes in the area. Secondly, housekeeping principles and other restrictions often demand that containers may only be placed in specific ways. So, any anticipatory prior rearrangements of stacks may not be permitted at all. Finally, dynamic situations and rapidly changing requirements at ports often allow very little time margins for any preparatory rearrangement. Further, it is not possible to use adjacent columns/stacks/locations as temporary storages for rearrangement. This is because it is very likely that the adjacent columns may often be at maximum height already. Also, in busy container yards, some other crane or device many be working on some other rearrangement activity at those locations. Thus it is an unsafe practice to assume that the adjacent container columns can be used for any re-stacking and re-arrangement processes.

1.3. Literature survey

SpRP involves cumulative impact of stack rearrangement and sequence-based aspect of precedence costs. The stack rearrangement sub-problem reduces to the feedback arc set problem, which has been shown to be *NP*−hard [11,17]. Minimization of NLHM effectively implies solution of the asymmetric travelling salesman problem on horizontal-graphs. Thus, both underlying sub-problems of SpRP are individually *NP*−hard. As their combination, SpRP is a unique problem and it has not been handled previously in the literature (to the best of our knowledge). To survey the literature relevant to the SpRP, we first briefly look at literature regarding operations at sea-ports and cargo terminals. The issue of internal rearrangement of containers in sea-ports and terminals has been discussed by Kim and Kim [27], Dekker et al. [18], Peterkofsky and Daganzo [37], Forster and Bortfeldt [20], Chen [13] and Kim and Bae [26]. Optimization of schedules for cranes in cargo container terminals has also been widely studied (see [40,42]) in the literature. A generic discussion of overall operations in terminals is presented in, Stahlbock and Voß [40], Zheng et al. [44], Gambardella et al. [21], Murty et al. [36] and Bish et al. [7]. Factors governing the efficiency of container terminals are discussed in Kozan [29].

If an individual container movement request is mapped as a "job", then NLHM maps to a job scheduling problem with sequence-dependent processing/setup time between jobs (JSSD). The logical link between crane scheduling and JSSD has been discussed in the literature [28,43]. Typically, JSSD aims to either minimize the overall makespan, total flow time or total tardiness [6]. Some of the earliest exact solutions for a single machine problem with the objective of minimizing the completion times are provided in Bianco et al. [4,5]. Specifically, Bianco et al. [5] provides a branch-and-bound algorithm where the structure for the dual variables and unimodularity of the dual (of the linear program) is used to determine the lower bounds for a branch-and-bound based formulation. Further, Bianco et al. [4] provides a dynamic programming formulation and also heuristic techniques to solve the problem. Some applications of JSSD, such as the scheduling of aircraft landings, involve the development of optimal schedules for planes landing on runways at airports (see [3,19]). In these, the separation time between consecutive landings depends on the relevant planes in the sequence that is developed. This problem is clearly similar to the JSSD. Ernst et al. [19] uses partial ordering information and presents a modified form of simplex algorithm for use in a branch and bound algorithm. Similar to the approach presented earlier in Bianco et al. [5], here too, the inherent structure of the dual is identified and exploited. Another mixed integer programming approach is presented in Beasley et al. [3].

1.3.1. Literature on the stack rearrangement aspect of SpRP

We look at the literature addressing stack rearrangement (for cases where re-positioning of one object hinders or assists the retrieval of another object later). This discussion typically involves operations needed for rearrangement of items (such as containers) within a finite space horizon. Apart from the current context of container operations at ports, it appears in several other contexts including automated material handling equipments (see [12]) at warehouses and other industrial applications [14,35,1]). The problems handled in these references slightly resemble the intention of the VSR aspect of SpRP. For each of these, the problem that is studied is variously denoted as 'crane sequencing problem', 'container relocation problem', 'container stacking problem', 'blocks relocation problem', etc. To further complicate the nomenclature, the literature also discusses several problems such as 'crane scheduling problems' which arise in the same domain but have entirely different specific details. As of today, there does not seem to be any standardization in the formal description or nomenclature of related problems. Christofides and Collo [14] are the seminal work for finding optimal rearrangement operating sequences to minimize rearrangement costs. Here, a graphical algorithm works in two stages to get results that are actually optimal in a restricted sense. This work differs from SpRP in the sense that classes of items with usages/turnover rates are defined for efficient handling. Further work in this direction has led to the 'blocks relocation problem' where the optimum relocation sequence must be determined for a given initial situation with a known sequential order of pickup (see [9]). Another direction of study is presented by Aslidis [1], where the sequence of pickup itself is to be determined. Strategies to estimate or minimize the number of re-arrangements needed are discussed in de Castillo and Daganzo [10] and Kim [25].

1.3.2. Literature on the path traversal aspect of SpRP

The horizontal movement efforts involved in NLHM naturally imbibe an aspect of path-generation. Such an approach to model the path-generation aspect of SpRP follows from Laporte [30], Srour and van de Velde [39]. If all containers are represented as nodes on the graph, the movement of the crane translates to a Hamiltonian path on this graph. This representation leads to a possible variant of the asymmetric traveling salesman problem (see [2,22,24,38]).

Integer programming formulations for a path-generation problem use specific constraints called subtour elimination constraints (abbreviated as SECs in this paper) to eliminate partial tours between subset of intermediate paths. SpRP also needs to eliminate sub-tour while building routes for cranes. The set of SECs originally suggested in Dantzig et al. [16] (hereafter called as 'Dantzig–Fulkerson–Johnson' or DFJ-SECs) impose connectivity requirements on any feasible routes by checking all possible sub-tours of cardinality less than n. The 'Miller–Tucker–Zemlin' (or MTZ) approach eliminates sub-tours using a polynomial number of constraints ([34]) by introducing $O(n^2)$ additional continuous variables indicating the order of the corresponding node in the overall sequence. Another approach by Claus [15] tracks the

traversal of path between two nodes indirectly via a third node. This approach involves $O(n^3)$ number of new variables and constraints and has not been very heavily used in the literature. Many more approaches for SECs have proposed by Kara and Bektas [41], Svestka and Huckfeldt [24].

2. Formulations and solution approaches

In this section, we present four different MIP formulations to solve the SpRP. Before that, we discuss a method to model the VSR costs and also introduce some terminology and sets, which will be used in all the MIP formulations later.

2.1. Modeling of VSR costs

For an SpRP instance $\mathbb{X}$, we assume that its representation on horizontal graph $G_h^{\mathbb{X}}$ is available. We now create a model specifically for the VSR aspect. This aims to capture the impact of handling a given container on the VSR of all other containers. We model $\mathbb{X}$ on a directed graph $G_v^{\mathbb{X}}(N,A)$. The nodes of $G_v^{\mathbb{X}}$ are the n containers to be handled. The set of arcs A is defined such that the existence of arc (i,j) between a pair of nodes $i,j \in N$ (corresponding to containers i,j) implies that exactly one of the following conditions holds:

1. The pickup point P_j of container j is the same location as the delivery point D_i of container i
2. The pickup point P_i of container i is the same location as the pickup point P_j of container j, and container i was initially below container j in the stack (meaning $H_j < H_i$)

The graph $G_v^{\mathbb{X}}$ developed for an SpRP data instance $\mathbb{X}$ is called as its *Precedence-Graph*. For the sample SpRP data instance that was shown earlier in Fig. 1, we now present the corresponding Precedence-Graph in Fig. 2 below.

We make the following observations regarding these Precedence-Graphs:

1. Any directed arc (i, j) on the Precedence-Graph represents an increment in VSR costs if container i is handled any time before container j. Conversely, existence of directed arc (i, j) on the Precedence-Graph implies that the recommended sequence of handling the containers (as far as only i,j are concerned) is j *before* i – inverse of direction pointed by the arc between them.
2. Often, there might not be a direct arc, but a directed path comprising several arcs may be found; we still conclude that recommended sequence of handling containers is inverse of the directed path sequence.
3. In absence of any such directed arc or path between a specified pair of containers, the order of execution between that pair of containers is immaterial for the overall cost of VSR.
4. Any nodes from N that do not have any arcs associated with them may even be removed from this graph $G_v^{\mathbb{X}}(N,E)$; sequencing decisions regarding these containers have absolutely no impact on the VSR costs of these or other containers.

Any proposed (feasible) solution to the SpRP data instance $\mathbb{X}$ is essentially an ordered sequence of all n nodes. For example, for instance shown in Figs. 1 and 2, consider the ordered sequence $22, 16, 6, 2, 1, 4, 8, 7, 9, 13$. This sequence easily translates to a directed Hamiltonian path on a *Horizontal graph*. Now, we imagine that the Hamiltonian path on the *horizontal graph* is being traversed. But at every node i during the traversal, the VSR cost of handling the underlying container i is being calculated by looking at the *precedence-graph*. At every node i, this VSR cost of handling container i is equal to the number of incoming arcs (j,i) from unvisited nodes $j \in N$ to that node i on the Precedence-Graph. Thus, the directed Hamiltonian path was traversed on horizontal graph, but the VSR costs were calculated using the precedence-graph. The objective is to minimize the sum of VSR costs for all containers – this is accomplished by selecting a Hamiltonian path that minimizes the sum of those incoming arcs for all nodes $i \in N$.

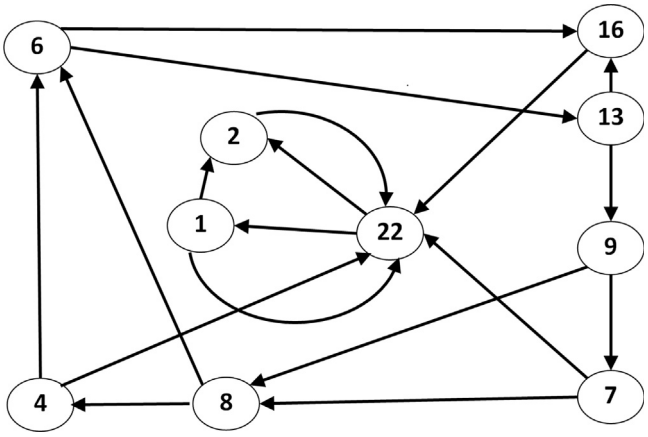

Fig. 2. Precedence-Graph of SpRP data instance presented earlier in Fig. 1.

The Horizontal graph (which is typically used in modeling route related problems) is a complete graph (all connecting arcs for every pair of nodes). On the contrary, the Precedence-Graph has only some specific arcs. Even among these, only few arcs (originating from containers already handled) contribute to the objective cost of SpRP. Thus, the *cumulative* objective of SpRP cannot be captured merely by the immediate precedence arc representations alone. SpRP needs novel representative formulations that can cater to these two different features.

2.2. General terminology

To assist in formulating VSR costs, we introduce some new terms. The terms defined below are pre-computed and act as deterministic problem parameters (not decision variables) for the MIP formulations that will be developed later in this section.

1. Set $S_i^1 := \{k \in N : D_k = P_i\} \quad \forall i \in N$
2. Set $S_i^2 := \{k \in N : P_k = P_i \text{ and } h_k < h_i\} \quad \forall i \in N$
3. Scalar term $T_{max} := \lambda_h \times \arg\max_i(C_i + E_{ij}) + \lambda_v \times \arg\max_i(H_i + |S_i^1|) \quad \forall i \in N$

For any container $i \in N$, the sets S_i^1 and S_i^2 denote the sets of containers for which the stacking cost (VSR) has some direct correlation to the relative precedence order of handling. Any container $k \in S_i^1$ moved before i increases the subsequent stack rearrangement cost of i. Any container $k \in S_i^2$ moved before i reduces the subsequent stack rearrangement cost of i. The hypothetical maximum time that the crane might take for handling any container is denoted by the term T_{max}.

Sets S_i^1 and S_i^2 form a crucial part of all further developments on modeling the VSR aspect of SpRP. Consider that a crane is assigned to move some n containers within a terminal. While executing these, a situation is reached that some k containers out of n have been handled in time t. Let these k containers be denoted by set $\tilde{N}$. The remaining $n-k$ containers are yet to be moved. Given one specific pending (un-handled) container j, we wish to find the current stack height h_j above j at pickup location P_j. We know that at the beginning of the problem horizon, H_j number of containers were stacked above j at P_j. We derive the instantaneous stack

height h_j from the initial stack height H_j as follows:

$$h_j = H_j + |S_j^1 \cap \tilde{N}| - |S_j^2 \cap \tilde{N}|,$$
where $\tilde{N}$:= Set of containers already handled by time t (1)

Eq. (1) will be used within all MIP formulations in various ways.

2.3. MIP formulations

2.3.1. Continuous time MIP formulation (CTF)

In this formulation, we rely on the notion of 'time' at which a specific container is handled. We apportion the time required for any activity and thus try to minimize the overall effort by minimizing the completion time. The continuous time variable t is defined as follows:

t_i = Time when the crane reaches the pickup location P_i for picking up ith container specifically. (2)

Further, the following decision variables defined $\forall i, j \in N$, where $i \neq j$ assist in tracking the relative sequential order of containers handled:

$$\delta_{ij} = \begin{cases} 1 & \text{If } i\text{th container is handled any time before } j\text{th container} \\ 0 & \text{If } j\text{th container is handled any time before } i\text{th container} \end{cases} \quad (3)$$

We minimize the earliest time of return of the crane to the depot after completion of all activities. We define $\tilde{T}$ as this time of return. Then the complete MIP formulations is as follows:

Minimize $\tilde{T}$, such that

$$t_i \geq \lambda_h E_{0i} \quad \forall i \in N \quad (4)$$

$$\tilde{T} \geq t_i + \lambda_h(E_{i0} + C_i) + \lambda_v \left(H_i + \sum_{k \in S_i^1} \delta_{ki} - \sum_{k \in S_i^2} \delta_{ki} \right) \quad \forall i \in N \quad (5)$$

$$\delta_{ij} + \delta_{ji} = 1 \quad \forall i,\ j \in N \quad (6)$$

$$t_j - t_i \geq \lambda_h(C_i + E_{ij}) + \lambda_v \left(H_i + \sum_{k \in S_i^1} \delta_{ki} - \sum_{k \in S_i^2} \delta_{ki} \right) + T_{max}(\delta_{ij} - 1) \quad \forall i,\ j \in N \quad (7)$$

Eq. (7) serves as a modified form of the MTZ SECs discussed earlier in Section 1.3. Strictly speaking, we should not need separate constraints for SECs or for transitivity in the δ variables. The MIP formulation induced by (4)–(7) offers a sound and complete MIP model for SpRP. We call this model as the *BasicCTF*.

We consider the following options to strengthen the CTF:

1. *Option* 1 – *additional transitivity constraints*: The following equations enforce transitivity within the δ variables:
$$\delta_{ij} \geq \delta_{ik} + \delta_{kj} - 1 \quad \forall i, j, k \in N : \ i \neq j \neq k \quad (8)$$
This leads to a slightly tighter formulation, but at the cost of adding $O(n^3)$ number of constraints. We refer to such a formulation as the *CTF with transitivity constraints*.
2. *Option* 2 – *transitivity as cuts*: The transitivity constraints generated through Eq. (8) can be enforced as (lazy) IP cuts for fractional solutions. Given a fractional solution of CTF, it is relatively easier and cheaper to detect any transitivity violation, and hence, enforce new cuts. We refer to such a formulation as the *CTF with transitivity cuts*.
3. *Option* 3 – *cycle control variables*: The SpRP solution induces a Hamiltonian path on the horizontal graph. Considering that the depot node 0 is the starting and the ending node, we may consider this as a Hamiltonian tour. The concept and usage of δ variables in defining this Hamiltonian tour makes the CTF amenable to the sub-tour elimination approach discussed in Claus [15]. In this approach, the direction of notional network flow across three distinct nodes i,j and k is captured by an additional decision variable $w_{ijk} \in [0,1]$ defined for $i,j,k \in N$ where $i < j < k$. In the current context, w_{ijk} indicates whether or not the crane moves from i to k via j. It as defined as follows:
$$w_{ijk} = \delta_{ij} + \delta_{jk} - \delta_{ik} \quad (9)$$
In the SpRP context, we term the w_{ijk} variables as cycle control variables. The MIP model involving the δ and w variables along with (4)–(7) and (9) (but not (8)) is termed as *CTF with cycle variables*. The important aspect of the cycle control variables in the SpRP context is that apart from the bounds $[0,1]$ and the defining equation (9), no separate constraint is needed on these cycle control variables.

Eq. (7) is an ill-conditioned constraint that weakens the CTF and hampers its performance for even medium sized problem data instances. We now consider formulations where we avoid such ill-conditioned equations and finally seek approaches where time is not explicitly considered.

2.3.2. Discrete sequence formulation (DSF)

We can avoid any ill-conditioned constraints if we totally replace the time aspect by a discrete sequence numbering approach. Here, we look at the sequential order of the containers as slots in the optimal numbering. Consider the following decision variables defined for $k \in \{1, 2, \ldots, n\}$ and $i \in N$:

$$\mathbb{Y}_i^k = \begin{cases} 1 & \text{If } i\text{th container is ordered so as to be handled at } k\text{th sequence in the overall ordering} \\ 0 & \text{If } i\text{th container is } \textit{not} \text{ ordered so as to be handled at the } k\text{th sequence in the overall ordering} \end{cases} \quad (10)$$

$$v^k = \text{Time spent or effort invested in horizontal motion immediately between the containers that were ordered so as to be handled at } k\text{th and } (k+1)\text{th sequence in the overall ordering} \quad (11)$$

Further, v^0 represents the time spent or effort invested for horizontal movement from depot location to the first container. Also, v^n represents the horizontal movement from last container to depot and the δ variables are as defined in (3).

NLHM is captured by sum of v^k. VSR is represented by rearranging (1). We minimize the sum of these

$$\text{Minimize}: \quad \sum_{i=0}^{n} v^k + \sum_{i \in N} \lambda_v \left(\sum_{j \in S_i^1} \delta_{ji} - \sum_{j \in S_i^2} \delta_{ji} \right) \quad (12)$$

such that

$$\sum_{i \in N} \mathbb{Y}_i^k = 1 \quad \forall k \in \{1, 2, \ldots, n\} \quad (13)$$

$$\sum_{k=1}^{n} \mathbb{Y}_i^k = 1 \quad \forall i \in N \quad (14)$$

$$\sum_{k=1}^{n} (k \mathbb{Y}_i^k) = \sum_{j \in N} \delta_{ji} + 1 \ \forall i \in N \quad (15)$$

$$\delta_{ij} + \delta_{ji} = 1 \quad \forall i,\ j \in N \quad (16)$$

$$\delta_{ij}, \mathbb{Y}_i^k \in \{0, 1\} \quad \forall i,\ j \in N \quad (17)$$

Further, we define v^k through the following equations:

$$v^0 = \lambda_h \sum_{i \in N} E_{0i} \mathbb{Y}_i^1 \tag{18}$$

$$v^n = \lambda_h \sum_{i \in N} E_{i0} \mathbb{Y}_i^n \tag{19}$$

$$v^k \geq \lambda_h E_{ij} (\mathbb{Y}_i^k + \mathbb{Y}_j^{k+1} - 1) \quad \forall k \in \{1, 2, \ldots, n-1\} \tag{20}$$

In this formulation, (13) and (14) are akin to the degree constraints enforced in route based problems like TSP. Eq. (15) ties the δ variables to the $\mathbb{Y}_i^k$ variables. We denote the MIP formulation induced by these constraints as the discrete sequence formulation (DSF).

In view of the integrality constraint on the δ_{ij} variable, we deduce some cuts that apply on the formulation. For some values of i, j, k, consider the permissible possibilities for δ_{ij} and $\mathbb{Y}_i^k$:

1. If δ_{ij} is 1 and ith container is ordered so as to be handled at or after the kth in sequence, then j cannot be ordered at or before kth sequence position.
2. Conversely, if δ_{ij} is 1 and jth container is ordered so as to be handled at or before the kth in sequence, then i cannot be ordered at or after kth sequence position.

We combine these assertions in the following equation:

$$2 \geq \delta_{ij} + \sum_{q=1}^{k} \mathbb{Y}_j^q + \sum_{q=k}^{n} \mathbb{Y}_i^q \quad \forall k \in \{1, 2, \ldots, n\} \text{ and } i,\ j \in N \tag{21}$$

These cuts can be used to strengthen the DSF formulation. We refer to this approach as *DSF with cuts*.

2.3.3. Integrated graph topology MIP formulation (IGTF)

In the preceding discussion, the stacking (VSR) aspect of SpRP had been totally decoupled from the routing (NLHM) aspect. We have modeled these on separate graph structures (Precedence-Graph for VSR and horizontal-graph for NLHM). But, even if the objective terms of both are different, eventually both visualize the solution in terms of a Hamiltonian path covering all n containers. In this section, we explore whether a cross pollination of ideas and concepts across these separate graphs leads to any hybrid technique yielding substantial improvement for SpRP. We continue to model VSR using δ_{ij} variables as defined in Eq. (3), but we will look at TSP-like approach for NLHM costs as well as for modeling of an overall route.

The TSP-based approaches have addressed Hamiltonian path problems by using an immediate precedence sequence or an arc traversal based decision variable. We begin by adapting that approach to the SpRP case:

$$u_{ij} = \begin{cases} 1 & \text{If } i\text{th container is handled immediately before } j\text{th container} \\ 0 & \text{Otherwise} \end{cases} \tag{22}$$

$$u_{i0} = \begin{cases} 1 & \text{If } i\text{th container is handled last in the sequence} \\ 0 & \text{Otherwise} \end{cases} \tag{23}$$

$$u_{0j} = \begin{cases} 1 & \text{If } j\text{th container is handled first in sequence} \\ 0 & \text{Otherwise} \end{cases} \tag{24}$$

Such decision variables u_{ij} are a standard choice for representing NLHM efforts. As discussed in Section 1.3, one major concern for these decision variables is finding efficient ways of eliminating sub-tours. If that can be managed, we can represent VSR through δ_{ij} variables and NLHM through u_{ij} variables to get efficient modeling of both costs. We temporarily postpone the discussion on sub-tour elimination and look at a minimal MIP model induced by δ_{ij} and u_{ij} variables only.

We explicitly express the overall effort in stacking and routing as the objective function to be minimized:

$$\text{Minimize}: \quad \lambda_h \sum_{i=0}^{n} \sum_{j=0}^{n} E_{ij} u_{ij} + \lambda_v \sum_{i \in N} \sum_{k \in S_i^1} \delta_{ki} - \lambda_v \sum_{i \in N} \sum_{k \in S_i^2} \delta_{ki} \tag{25}$$

The degree constraints for route are defined as

$$\sum_{i=0}^{n} u_{ij} = 1 \quad \forall j \in \{0, 1, 2, \ldots, n\} \tag{26}$$

$$\sum_{j=0}^{n} u_{ij} = 1 \quad \forall i \in \{0, 1, 2, \ldots, n\} \tag{27}$$

Anti-symmetry for VSR order is defined as

$$\delta_{ij} + \delta_{ji} = 1 \tag{28}$$

Interlink between constraints is enforced as

$$\delta_{ij} \geq u_{ij} \tag{29}$$

We term the MIP formulation induced by Eqs. (26)–(29) as an *Integrated* graph topology formulation (IGTF).

Suppose that the values of only δ variables are known from a feasible non-fractional solution of IGTF. With this limited information only, it is still possible to determine a relative sequential ordering of containers. Alternately, consider that the values of only u variables from a feasible non-fractional solution of IGTF are known. It is still possible to determine a relative sequence order of all n containers from u values alone. Thus a single (integer) solution for IGTF allows the δ variables and u variables to independently define two different routes in almost unrelated ways. The only coupling or interconnection between these two routes is through Eq. (29). This is a weak constraint due to the following reasons:

1. Even if value of u_{ij} is 1 for some i, j, it merely compels that in the route induced by δ variables, i must be handled any time before j. Feasible solutions of δ variables with some container k handled between i and j can still be built.
2. Conversely, even if the route induced by δ variables does not envisage container k between i and j, the u variables can allow such a route.
3. Both routes independently allow separate unrelated sub-tours.

This means that the two different partial solutions (or routes) induced by u and δ variables do not necessarily yield a single meaningful tour for the crane. IGTF (as discussed till now) is not yet a complete formulation to solve SpRP – some additional constraints are required.

The major flaw in terms of the δ variables is the absence of any transitivity enforcing constraint (similar to Eq. (8)). In terms of u_{ij} variables, the flaw is the absence of sub-tour elimination constraints. Since sub-tours of TSP have been well studied, we first consider the option of enforcing DFJ or MTZ family of SECs on the u variables only. Unfortunately, even if the route induced by u variables allows no sub-tours, the route induced by δ variables still permits sub-tours and meaningless routes. This happens because the interlinking equation (29) forms only a lower bound on δ variables. In the absence of an upper bound, some $\delta_{ij} = 1$ disrupts the routes induced. Apparently then, it may be necessary to enforce two separate sets of route-correcting constraints – one each on δ and u variables. So, the next option is to independently enforce SECs on both δ and u. This leads to an unwieldy MIP model (with too many constraints).

To avoid these problems, we consider the cross-over interacting effect of the structure of δ and u variables to make a noteworthy assertion about permissible routes. Suppose that in addition to the constraints from Eqs. (26)–(29), we also restrict sub-tours within δ

variables by some means. We do not put any additional constraints on any variables (especially u). Then, we argue that Eq. (29) inherently prevents sub-tours within u_{ij} variables also. Further, we argue that the transitivity of δ variables is then automatically assured by the constraints on u variables. We formally state and prove these assertions in Theorem 1.

Theorem 1. *For any feasible non-fractional solution of IGTF, if the route induced by values of δ variables strictly does not involve any sub-tours, then the following must be true:*

1. *Route induced by u variables does not involve any sub-tours.*
2. *Transitivity constraints (Eq. (8)) between the sequence of δ variables are not violated.*

Proof. We prove these statements by contradiction. We consider that a non-fractional solution of IGTF is available for an SpRP instance $\mathbb{X}$. Further, this solution involves no cycles for δ variables.

We assume that the values of u variables do involve a sub-tour of k containers out of n. Then, the lower limits on δ variables from Eq. (29) imply that a corresponding cycle must have been induced on δ variables also. This conclusion contradicts our premise that the δ variables involved no cycles, so the starting assumption must be wrong. This completes the first part of the proof.

Continuing with a non-fractional solution of IGTF for values of δ variables involving no cycles, we assume that the route induced by δ variables violates a transitivity constraint (Eq. 8). Consider the values of u variables. The degree constraints in Eqs. (26) and (27) ensure that every node $i \in N$ is visited exactly once for the route induced by u variables. The first part of the proof ensured that there cannot be any sub-tours in such a route. This means that the route induced by u variables is a valid Hamiltonian tour. We mark this Hamiltonian tour on the Precedence-Graph to compare with routes induced by δ variables. Consider three distinct containers $i,j,k \in N$, such that the transitivity constraint between these is violated. Without loss of generality, we assume that $\delta_{ij}=1, \delta_{jk}=1$, but $\delta_{ik}=0$. Then, by Eq. (28), $\delta_{ki}=1$. This means that the Precedence-Graph $G_v^{\mathbb{X}}$ obeys edges (i,j), (j,k) and (k,i) giving a sub-tour within vertices i,j,k for route of δ variables. This conclusion contradicts our initial premise, so the starting assumption of violation of transitivity constraints must be wrong. Hence by contradiction, part 2 is proved. □

Consider the converse of the second part of Theorem 1. We begin with a feasible non-fractional solution of IGTF. If this solution does not violate the transitivity constraints (Eq. (8)), we check whether the two routes induced by this solution allow any sub-tours or discrepancies.

Theorem 2. *For any feasible non-fractional solution of IGTF, if the route induced by values of δ variables does not violate transitivity constraints in Eq. (8), then the following must be true:*

1. *Route induced by δ variables does not involve any sub-tours.*
2. *Route induced by u variables does not involve any sub-tours.*

Proof. We prove these statements by contradiction. We consider that a non-fractional solution of IGTF is available for an SpRP instance $\mathbb{X}$. Further, the values of δ variables this solution do not violate any transitivity constraints from Eq. (8).

In order to prove the first part of the theorem, we assume that a subset $\mathbb{K} \subsetneq N$ of containers does form a sub-tour within the route induced by values of δ variables. For containers $i,j \in \mathbb{K}$, the transitivity constraints do not permit any non-fractional (integer) value for the δ_{ij} variable. So, no such subset $\mathbb{K} \subsetneq N$ can form a sub-tour. For the second part of the proof, we continue with the non-fractional solution of IGTF for SpRP instance $\mathbb{X}$ where the values of δ variables do not violate any transitivity constraint. This time, we assume that a subset $\mathbb{K} \subsetneq N$ of containers does form a sub-tour within the route induced by values of u variables. In this case, $\forall i,j \in \mathbb{K}$ where $u_{ij}=1$, we have $\delta_{ij}=1$ from Eq. (29). So, the sub-tour on u variables must result in a sub-tour within δ variables. But the first part of this proof has already precluded any sub-tour within routes induced by δ variable values. Again, the assumption of sub-tours on $\mathbb{K}$ is refuted. This completes the proof. □

Theorems 1 and 2 allow us to argue that satisfying the transitivity of δ variables is necessary and sufficient to eliminate sub-tours among route induced by δ_{ij} variables. Also, either of these assures the elimination of sub-tours in u variables. Identifying and eliminating sub-tours from u or δ variables is more complicated than enforcing transitivity on δ variables. We have the following options to do so.

- *Option* 1 – *direct transitivity constraints or cuts*: Transitivity equations enforced in MIP model directly as constraints will preclude sub-tours. This option has been discussed earlier in CTF because the same set of δ variables in the same sense was used there as well. The IGTF augmented with transitivity constraints is denoted as *IGTF with transitivity constraints*. But we expect even better results from enforcing transitivity as lazy cuts. This is because the IGTF constraints inherently ensure that some transitivity equations are satisfied. We only wish to identify and eliminate solutions where any transitivity constraint is indeed violated. We denote the IGTF formulation with transitivity based (lazy) cuts as *IGTF with transitivity cuts*. This approach is expected to be more useful because it involves absolutely no ill-conditioned constraints and less than $O(n^3)$ number of constraints.
- *Option* 2 – *cycle control variable*: Transitivity is implicitly enforced by the additional cycle control variables w_{ijk} as seen earlier in the CTF approach. IGTF is used in conjunction with w_{ijk} variables and related (9) is termed as *IGTF with cycle variables*. This formulation has no ill-conditioned constraints (Big-M terms); also it involves only $O(n^2)$ number of explicit inequality constraints. However, in a sense, it corresponds to IGTF with transitivity constraints.
- *Option* 3 – *MTZ sequencing constraints on δ variables*: We can eliminate sub-tours within δ by including an additional decision variable to represent the sequential order of nodes in the tours. Traditionally, the MTZ constraints were enforced on u variables, but Theorem 2 indicates that SpRP can be modeled by enforcing them on δ variables. We denote such an approach as *IGTF with MTZ constraints*. Obviously, it will involve $O(n^2)$ number of Big-M type constraints.

2.3.4. Hybrid formulation (HF)

The DSF approach (in Section 2.3.2) was designed to inherently enforce an explicit sequential ordering for the nodes through the discrete sequencing variable $\mathbb{Y}$. The unresolved lacuna of IGTF is easily addressed by inclusion of additional decision variable $\mathbb{Y}$ and Eqs. (13)–(15). Such a hybrid formulation inherently precludes any sub-tours. It has no Big-M kind of constraints and is compact because only $O(n^2)$ number of equations are involved. Also, it is possible to further tighten this formulation using cycle variables, transitivity constraints or additional cuts (from DSF or more). This leads to four sub-options within this approach – Basic HF, HF with cycle variables, HF with transitivity constraints and HF with additional cuts.

2.4. Equivalence of objective functions

The CTF (in Section 2.3.1) had the objective to minimize the time $\tilde{T}$ for the crane. The DSF (in Section 2.3.2) minimized the summation of two terms in Eq. (12). The objective for IGTF in Section 2.3.3 as represented in Eq. (25) also minimized the overall effort in stacking and routing. However, all these three (seemingly different) objective functions are functionally equivalent and will effectively yield the same optimal routes for any SpRP instance. They merely represent alternative ways to calculate the total time spent or effort expended in handling all containers.

We explain this as follows:

1. $\tilde{T}$ in CTF is recursively defined through Eq. (7). In conjunction with Eqs. (4) and (5), the total NLHM and VSR costs are counted in $\tilde{T}$. This definition of $\tilde{T}$ also holds a term in $\lambda_h C_i$. But since FLHM is constant for absolutely any feasible solution, it does not lead to any difference in optimal routes.
2. The summation of $E_{ij}u_{ij}$ terms in IGTF for Eq. (25) represents the NLHM efforts. This is equivalent to $\sum v^k$ term in DSF for Eq. (12) because v^k is constrained by Eqs. (18)–(20).
3. The VSR terms in λ_v for Eqs. (12) and (25) are exactly same.
4. We conclude that the objective terms for DSF and IGTF are physically equal to each other for any feasible route of any SpRP instance. CTF gives an objective value which differs from that of IGTF/DSF by exactly a constant term ($\lambda_h C_i$). However, all objective functions are logically equivalent to each other.

3. Evaluation of SpRP formulations

To ascertain the capability and suitability of our techniques, we will now test them for a wide range of problem data instances. This section first explains the development of the data instances and then presents the results for computations by the various techniques.

3.1. SpRP data instances

We first look at characteristic features of SpRP data instances. The intent is to identify major features of an instance that make it difficult for a specific MIP formulation to address. This information will later be used in designing the data instances, so that the impact of any such feature on the relative performance of MIP techniques is identified clearly.

We consider the following metrics for quantifying the size of any SpRP data instance:

1. *Number of containers n and number of locations l*: Both of these values are obvious primary indicators of problem instance size. An increase in n will increase the number of nodes handled on the horizontal graph as well as the priority graph. The impact of a change in l (with fixed n) on NLHM and VSR costs is not so straightforward. We expect any major increase in l, leads to lesser interaction between stacks and hence the VSR should decrease.
2. *Connectivity of precedence graph*: If the Precedence-Graph for a given SpRP instance has very few edges (with very small number of strongly connected components), then the impact of routes chosen on VSR costs is easier to compute. Most optimization efforts in such cases is directed to minimization of NLHM. The number of strongly connected components detected in Precedence-Graph is expected to be a very strong indicator for this. We attempt to make instance families that cover a diverse range in connectivity.

Table 1
Configurations explored for $\lambda_R = \lambda_h/\lambda_v$.

λ_h	100	1	1	1	1	1	1
λ_v	1	1	5	10	25	60	100
Ratio $\lambda_R = \lambda_h/\lambda_v$	100	1	0.2	0.1	0.04	0.016667	0.01

3. *Symmetry or similarity in containers*: Consider a case where multiple containers, say $1, 2, 3$, need to be picked up from a single location ζ_1 and delivered to the same location ζ_2. The E_{ij} values for NLHM costs for all these containers are same. Such cases of similarity induce some symmetry in the MIP formulation. We extend this discussion to cases where the pickup locations (or delivery locations) for a subset of containers are very close to each other. Such clustered container-handling cases often appear at ports when similar containers (same destination, similar cargo, etc.) are placed in specific areas of the yard. With increasing size of instance families, we proportionally introduce more cases of cluster symmetry.
4. *Ratio $\lambda_h : \lambda_v$*: If λ_v is set to zero or negligibly low value, the SpRP reduced to SCP. On the contrary, if λ_h is set to zero or negligibly low value, SpRP indicates a situation where only vertical stack rearrangement is to be minimized. This situation may be applicable in cases like industrial warehouses where the distances between locations are negligible compared to stack height. All practical SpRP instances are expected to lie somewhere between these extremes. To cover this entire range, we will look at several alternative configurations (of λ values) as listed in Table 1.

We expect to discuss the ratio of $\lambda_h : \lambda_v$ in several different contexts for the remainder of this paper. So, we define a new term λ_R to indicate this ratio of $\lambda_h : \lambda_v$. For any given topology of locations and given arrangement of containers, we expect to see very different solutions depending on λ_R value. Further, λ_R is expected to be governed by the configuration of the maritime port for which the SpRP is being solved. In that sense, the λ_R value captures the configuration of the port while other parameters like n, H, P or D capture the specific operational aspects. Hence, we will use the term *configuration* to refer to the scenario caused by a given λ_R value for all possible problem data instances where that value of λ_R is used. The results in Section 3 will be organized based on configurations as defined in Table 1.

Creation and organization of data instances: Individual problem data instances are developed by taking the topology of a typical maritime port and defining locations from it. The operational data for containers is randomly generated. The complete data instance families in all configurations are available from the authors on request. We intend to design and organize problem data instances such that the impact of all the quantifying metrics discussed earlier can be individually identified. Hence, we distribute the problem data instances into sets called as 'families'. All instances within a given 'family' are incremental, meaning that the instance of size $k+1$ will have all containers from the instance of size k, with the same topology and arrangement. Only some new additional topology or operational requirement will be added over the previous instance. On this basis, we develop following families of problem data instances:

1. *Small-sized assorted instances*: This family includes assorted, small sized instances.
2. *Constant l, increasing n*: This family includes instances with a fixed set of locations and successive problem data instances involve increasing number of containers. This situation closely

Table 2
Results for MIP approaches for some SpRP data instances.

Instance details			**Instances solved for configuration** $\lambda_R = 100$									**Instances solved for configuration** $\lambda_R = 1$								
			Optimum	Computational time taken (s)				B&B nodes traversed for optimal solution				Optimum	Computational time taken (s)				B&B nodes traversed for optimal solution			
Name	L	N		CTF	DSF	IGTF	HF	CTF	DSF	IGTF	HF		CTF	DSF	IGTF	HF	CTF	DSF	IGTF	HF
L4N6M1	4	6	2011	0.03	0.08	0	0.01	0	0	0	0	31	0.07	0.26	0.01	0.02	0	0	0	0
L5N10M1	5	10	21,029	0.52	11.16	0.13	0.24	256	7436	27	0	239	3.57	11.07	0.23	0.67	318	7189	77	6
L8N12M1	8	12	27,042	2.81	1353.45	0.77	1.21	1028		876	23	312	5.92	*32%*	0.67	1.47	1550		420	33
L10N14M1	10	14	52,256	4.28	*72%*	0.48	0.88	2005		0	0	578	9.18	*71%*	1.92	2.56	6809		179	13
L15N15M1	15	15	116,074	10.04	*91%*	0.08	0.99	8412		0	0	1234	22.85	*87%*	1.1	1.53	9929		0	0
L11N16M1	11	16	39,126	131.32	*67%*	1.52	2.55	24,804		5	0	417	197.31	*64%*	3.93	15.35	39,381		343	42
L15N16M1	15	16	142,276	37.81	*95%*	0.1	0.7	24,958		0	0	1497	87.9	*9%*	2.27	1.43	32,633		339	0
L15N17M1	15	17	163,479	327.01	*96%*	0.41	1.04	93,403		0	0	1712	401.73	*92%*	9.32	3.16	184,918		656	0
L12N18M1	12	18	42,243	56.03	*78%*	1.82	14.3	13,492		34	3	465	445.24	*7%*	4.66	22.58			147	206
L15N18M1	15	18	161,380	407.88	*96%*	0.14	2.51			0	0	1693	1307.81	*92%*	78.99	110.57			13,172	257
L15N19M1	15	19	165,582	*15%*	*97%*	0.61	8.51			0	0	1737	*15%*	*92%*	217.89	636.82			18,641	1516
L15N20M1	15	20	171,882	*15%*	*97%*	0.16	4.81			0	0	1799	*29%*	*93%*	141.51	148.76			9692	139
L10N25M1	10	25	74,904	*40%*	*95%*	38.99	633.86			1694	380	850	*44%*	*86%*	484.73	*0.1%*			15,732	
L11N25M1	11	25	68,105	*43%*	*94%*	58.19	862.33			1224	630	783	*42%*	*86%*	460.15	*0.1%*			10,864	
L12N25M1	12	25	154,603	*50%*	*97%*	3.98	55.13			116	12	1643	*46%*	*93%*	133.82	*0%*			7976	
L13N25M1	13	25	170,499	*45%*	*97%*	3.27	54.81			0	0	1801	*52%*	*93%*	168.18	*0%*			10,449	
L14N25M1	14	25	145,804	*41%*	*97%*	3.14	83.7			0	0	1559	*49%*	*92%*	291.92	*0%*			5671	
L15N25M1	15	25	97,594	*44%*	*94%*	11.53	551.47			125	76	1066	*46%*	*88%*	57.7	*0%*			6233	
L9N25M1	9	25	49,101	*45%*	*93%*	607.55	*0.1%*			11672		590	*47%*	*79%*	942.2	*0.1%*			30,625	
L20N25M1	20	25	350,413	*62%*	*89%*	4.84	126.36			448	47	-X-								
L15N30M1	15	30	145,620	*64%*	*97%*	530.36	*0.1%*			5398		-X-								
L20N30M1	20	30	389,244	*68%*	*91%*	6.4	1185.07			115	20	-X-								
L15N35M1	15	35	110,959	*68%*	*99%*	1328.57	*0.1%*			5023		-X-								

Instance details			**Instances solved for configuration** $\lambda_R = 100$									**Instances solved for configuration** $\lambda_R = 1$								
			Optimum	Computational time taken (s)				B&B nodes traversed for optimal solution				Optimum	Computational time taken (s)				B&B nodes traversed for optimal solution			
Name	L	N		CTF	DSF	IGTF	HF	CTF	DSF	IGTF	HF		CTF	DSF	IGTF	HF	CTF	DSF	IGTF	HF
L4N6M1	4	6	74	0.03	0.28	0.08	0.08	0	0	0	0	121	0.04	0.23	0.11	0.24	145	0	0	0
L5N10M1	5	10	355	1.69	13.3	0.26	0.73	1310	8530	0	10	500	4.9	37.18	0.39	1.34	1653	10,367	101	0
L8N12M1	8	12	479	10.96	*24%*	0.83	4.1	7708		291	52	684	31.71	*18%*	2.22	8.57	13,027		161	174
L10N14M1	10	14	802	39.85	*54%*	8.71	28.23	15,357		7654	336	1075	166.05	*41%*	7.99	27.67	59,386		2236	412
L15N15M1	15	15	1530	186.74	*72%*	9.57	12.19	86,657		3408	49	1887	1436.85	*59%*	4.78	17.5			3892	69
L11N16M1	11	16	521	271.04	*55%*	2.1	23.74	26,554		214	79	650	228.23	*47%*	1.08	27.15	43,613		598	115
L15N16M1	15	16	1795	1606.34	*76%*	14.13	8.42			4555	41	2149	*19%*	*64%*	10.93	23.03			1379	96
L15N17M1	15	17	2019	*17%*	*79%*	45.08	18.52			8252	259	2387	*28%*	*67%*	36.3	13.55			6796	43
L12N18M1	12	18	637	830.24	*58%*	4.11	58.03			673	169	852	*35%*	*52%*	14.65	176.45			1761	1161
L15N18M1	15	18	1995	*21%*	*79%*	12.13	55.69			5058	260	2363	*36%*	*67%*	34.81	72.99			4441	379
L15N19M1	15	19	2052	*28%*	*79%*	79.83	566.35			8815	1023	2435	*36%*	*67%*	56.74	1256.81			7964	2430
L15N20M1	15	20	2118	*39%*	*8%*	314.2	175.5			26868	600	2501	*41%*	*68%*	250.48	113.93			17,880	250
L11N25M1	11	25	1189	*56%*	*65%*	1042.08	*0.7%*			35867		-X-								
L12N25M1	12	25	2035	*54%*	*78%*	866.43	*0.2%*			39473		2523	*61%*	*67%*	899.77	*0.2%*			39,669	
L13N25M1	13	25	2188	*53%*	*8%*	308.82	*0.1%*			12,788		2663	*56%*	*68%*	278.09	*0.4%*			8096	
L14N25M1	14	25	1961	*51%*	*77%*	142.87	*0.1%*			7866		2461	*52%*	*65%*	610.58	*0.3%*			13,186	
L15N25M1	15	25	1427	*56%*	*72%*	126.39	*0.4%*			8475		1868	*61%*	*60%*	379.13	*0.4%*			14,971	
L9N25M1	9	25	-X-									1472	*63%*	*47%*	1609.59	*0.9%*			43,996	

Table 2 (*continued*)

Instance details			**Instances solved for configuration** $\lambda_R = 100$								**Instances solved for configuration** $\lambda_R = 1$								
			Optimum	Computational time taken (s)				B&B nodes traversed for optimal solution				Optimum	Computational time taken (s)				B&B nodes traversed for optimal solution			
Name	L	N		CTF	DSF	IGTF	HF	CTF	DSF	IGTF	HF		CTF	DSF	IGTF	HF	CTF	DSF	IGTF	HF
L4N6M1	4	6	241	0.05	0.15	0.01	0.02	0	0	0	0	521	0.05	0.3	0.02	0.02	0	0	0	0
L5N10M1	5	10	935	11.39	44.5	0.42	2.26	2728	15041	133	64	1922	12.24	75.24	1.36	4	16297	12738	565	200
L8N12M1	8	12	1296	70.21	*10%*	1.58	12.33	55,496		1354	405	2570	181.61	*0.5%*	3.52	25.57	221,131		1449	968
L10N14M1	10	14	1870	807.13	*26%*	18.16	173.39			7722	2773	3594	*16%*	*18%*	*10.22*	265.01			8129	
L15N15M1	15	15	2921	*19%*	*39%*	4.97	11.65			1729	54	5269	*27%*	*24%*	2.78	3.79			1369	0
L11N16M1	11	16	1025	1487.29	*42%*	1.72	43.22			429	242	1890	*32%*	*37%*	11.39	325.83			5097	3085
L15N16M1	15	16	3199	*39%*	*45%*	6.5	11.75			3794	74	5595	*45%*	*27%*	5.81	32.68			3107	160
L15N17M1	15	17	3482	*56%*	*48%*	24.21	34.41			8817	102	5983	*54%*	*30%*	15.21	131.54			5958	470
L12N18M1	12	18	1469	*46%*	*40%*	30.6	1500.31			4577	6738	2761	*52%*	*33%*	121.43	*0.4%*			24,110	
L15N18M1	15	18	3458	*61%*	*48%*	17.8	37.24			4270	98	5960	*56%*	*30%*	26.13	105.33			5438	283
L15N19M1	15	19	3574	*57%*	*48%*	152.05	543.8			9666	1180	6154	*58%*	*30%*	48.33	265.44			12,786	654
L15N20M1	15	20	3641	*54%*	*49%*	174.59	*0.3%*			19,402		6248	*58%*	*32%*	166.31	1527.09			12,300	4182
L11N25M1	11	25	-X-									6447	*71%*	*29%*	1371.94	*0.4%*			81,516	
L13N25M1	13	25	4075	*69%*	*51%*	*300.73*	*0.7%*			11,129		7239	*70%*	*36%*	576.48	*0.5%*			45,729	
L15N25M1	15	25	3134	*67%*	*42%*	121.56	*0.5%*			9722		5956	*67%*	*32%*	146.67	*0.4%*			14,458	

Instance details			**Instances solved for configuration** $\lambda_R = 100$								**Instances solved for configuration** $\lambda_R = 1$								
			Optimum	Computational time taken (s)				B&B nodes traversed for optimal solution				Optimum	Computational time taken (s)				B&B nodes traversed for optimal solution			
Name	L	N		CTF	DSF	IGTF	HF	CTF	DSF	IGTF	HF		CTF	DSF	IGTF	HF	CTF	DSF	IGTF	HF
L4N6M1	4	6	841	0.05	0.39	0.02	0.01	0	0	0	0		0.05	0.24	0.04	0.06				
L5N10M1	5	10	2875	19.96	17.9	0.64	3.35	18,948	2852	0	122		7.75	30.05	0.49	1.80				
L8N12M1	8	12	3930	102.23	*0.2%*	1.66	32.49	20,5351		1588	1160		57.9	1353.5	1.6	12.2				
L10N14M1	10	14	5398	*20%*	*15%*	4.49	30.62			1023	824		205.3		7.0	75.5				
L15N15M1	15	15	7949	*38%*	*16%*	2.61	12.58			315	94		414.1		3.7	8.6				
L11N16M1	11	16	2801	*37%*	*28%*	51.07	1176.58			12,747			463.0		10.4	230.6				
L15N16M1	15	16	8315	*44%*	*18%*	7.33	81.98			4452	500		577.4		6.7	22.9				
L15N17M1	15	17	8823	*45%*	*20%*	34.17	192.14			5918	882		364.4		23.5	56.3				
L12N18M1	12	18	4114	*56%*	*32%*	113.11	*0.3%*			23,492			443.8		41.5	354.3				
L15N18M1	15	18	8800	*57%*	*21%*	55	101.23			9104	158		857.8		28.3	69.4				
L15N19M1	15	19	9058	*54%*	*21%*	12.81	210.78			6283	763				81.2	498.4				
L15N20M1	15	20	9131	*60%*	*22%*	42.7	1243.44			6810	3813				155.7	535.6				
L11N25M1	11	25	10,074	*69%*	*27%*	812	*0.2%*			16,071					748.9	633.9				
L12N25M1	12	25	10,637	*75%*	*28%*	1601.91	*0.7%*			35,564					743.8	862.3				
L13N25M1	13	25	10,783	*73%*	*28%*	130.82	*0.5%*			10,902					244.3	54.8				
L14N25M1	14	25	10,918	*71%*	*25%*	942.3	*0.6%*			21,149					398.2	83.7				
L15N25M1	15	25	9065	*68%*	*29%*	22.03	*0.1%*			7131					123.6	551.5				
L20N25M1	20	25	12,547	*76%*	*32%*	340.14	*0.2%*			11,830					172.5	126.4				

(1) MIP techniques were executed with computational time limit of 1800s. If the optimal solution was not found by this time, the table displays the percentage relative MIP gap in place of the computational time. In such a case, the entry is italicized and a percentage sign (%) is appended to it. (2) For any MIP formulation, the best observed performance among all its variants is reported in this table. (3) The entry -X- indicates that no MIP formulation gave the optimal solution within the prescribed time.

mirrors the practical situation in ports. We consider families with 15–25 locations.

3. *Constant n, Increasing l*: This family contains instances with a fixed number of containers being shuffled around several locations, such that successive problem data instances involve more locations. We consider a family with 25 containers and 3–15 locations. This situation is for demonstrative and analysis purposes only.

3.2. Computational results

The solution techniques discussed in this paper were coded in Java™SE (build 1.6) language and executed on a computer running a 16-core 64-bit Intel™Xeon™processor at 2.93 Ghz with 108 GB RAM. MIP formulations were solved using IBM™ Cplex™ 12.5 solver. The cuts proposed in this paper were implemented as feasibility cuts (or lazy cuts) for Cplex solver. Transitivity constraints are additionally marked as user-cuts and heuristically introduced when suitable fractional solutions are detected.

All MIP techniques (from Section 2) were individually used to solve the instances (as listed in Section 3.1) and the performance was recorded for various metrics. We attempted to solve every data instance for every configuration of λ_R ratio (listed in Table 1). Since the data generated is quite voluminous, we restrict ourselves to tabulating the results only from a cross section of instances across all families. Any conclusions that are drawn are based on a summary of data over larger number of instances.

Table 2 presents an abridged extract of the computational results. The data in this table is arranged as follows:

- This table is organized in terms of λ_R configurations as defined in Table 1 earlier. Further, the average performance over all configurations is also tabulated at the end of this table.
- Within the data for any specific configuration, we have grouped the computational effort required (time taken) for every MIP approach. However, Section 2 had considered many alternative variants (including cuts, etc.) for every MIP approach. We postpone any discussions on internal comparison of these variants to the later parts of this section. Table 2 presents the results for the best possible variant within every MIP formulation.
- The MIP formulations are executed by restricting the solver to 1800s of computational time. If the optimal solution is obtained by this time, the computational time taken (in seconds) is recorded. Also, the number of branch-and-bound nodes traversed is recorded.
- If the optimal solution was not yet reached by 1800s, then the percentage relative MIP gap is reported. In such cases, the entry is italicized and marked with a percentage (%) sign. An MIP gap of 0% means that the optimal value was found, but optimality was not yet proved by the MIP solver (which means that some

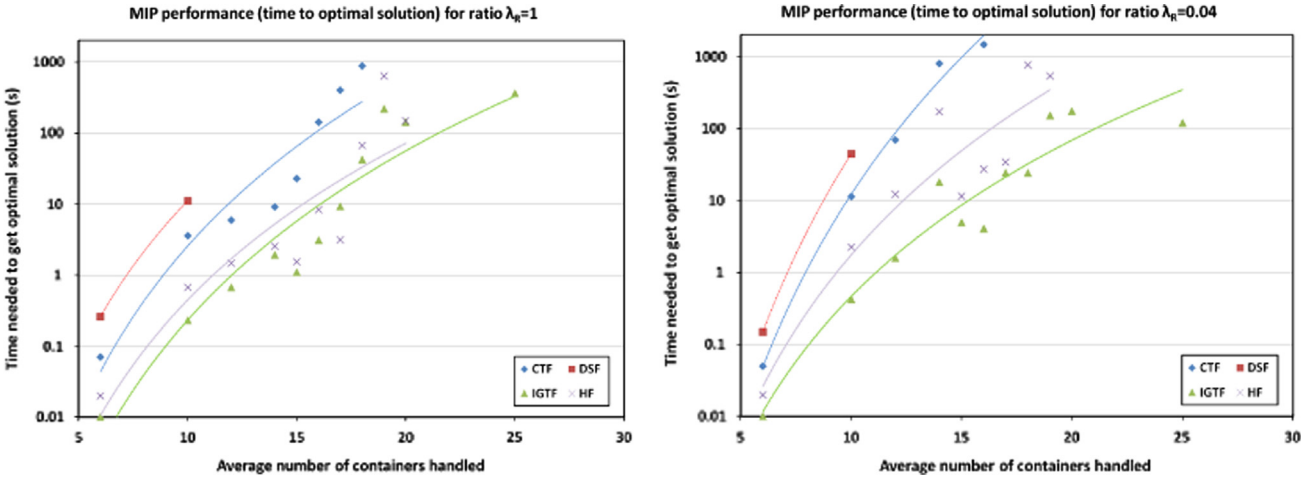

Fig. 3. Plots for computational time taken by MIP methods for configurations with ratio $\lambda_R = 1$ and 0.04.

Table 3
Tightness of MIP formulations measured using LP relaxation values.

Average number of containers	Objective value of LP relaxation expressed as % of the Optimal objective value											
	Ratio $\lambda_R = 0.2$			Ratio $\lambda_R = 0.1$			Ratio $\lambda_R = 0.04$			Ratio $\lambda_R = 0.0166$		
	CTF	DSF	IGTF/HF	CTF	DSF	IGTF/HF	CTF	DSF	IGTF/HF	CTF	DSF	IGTF/HF
6	31.1	62.2	88.2	24	66.9	88.8	21.2	77.2	94.3	20.5	82.7	95.8
10	69.6	46.8	93.2	50.4	53.2	90.4	28.7	60.5	87.2	21	65.9	86
12	50.7	42	92.4	38.6	50.6	89.9	25.1	60.3	87.2	18.2	69.9	89.6
14	55.2	34.4	95.4	43.5	44.7	93.4	29	58.6	90.5	20	70.4	90.6
15	30.1	24.4	98.9	25.5	37	98.3	18.6	57.3	97.9	13.8	74.9	98.1
16	48.8	27.8	97.7	40.3	36.5	97.2	27.4	50.9	95.7	17.9	63.9	92.9
17	22.6	19.5	99.4	19.7	30.9	99.2	14.9	50.9	98.6	12.1	70	98.3
18	41.1	28.3	97	34.5	37.3	95.4	25.8	52	94.1	20.6	66.3	93.2
19	22.2	19.6	99.3	19.3	31.1	98.8	14.5	51	98	11.1	70	98.3
20	21.5	19	99.2	18.8	30.3	98.8	14.2	50.1	98.1	9.2	59.5	84.3
25	33.9	24	98.4	28.6	36.3	97.7	21	55.1	96.7	16.6	73	95.7

more unexplored branch and bound nodes were pending for traversal). In such cases, the number of branch and bound nodes are no longer relevant.

- If none of them MIP approaches find the optimal solution for a specific problem data instance in a specific configuration, we indicate this by marking '-X-' in the column for optimal value. In such cases, the other columns are no longer relevant. For example, the instance L9N25M1 in the configuration $\lambda_R = 0.2$ was not solved by any solver. Hence, its entries in the relevant rows are empty. Similarly, instance L11N25M1 for the configuration $\lambda_R = 0.04$ has empty rows.

To visualize the results from Table 2, we plot the computational times needed by MIP methods for the configurations $\lambda_R = 1$ and $\lambda_R = 0.04$ in Fig. 3. Other configurations show a similar trend.

Both, Table 2 and Fig. 3 demonstrate that IGTF clearly outperforms all other MIP approaches. Further, the performance of DSF and CTF is comparatively too poor. This is in line with our expectation that any reliance on the time aspect will spoil the performance of the MIP solver. The computational performance of HF is slightly poorer than the IGTF. This contrasts the observation that HF needs to traverse much fewer number of branch and bound nodes (and hence, it is seemingly a tighter formulation). We postulate that HF, though a tighter MIP formulation, is hampered by usage of too many variables and constraints. This potentially makes every branch and bound node much slower to solve. It remains to be seen whether or not the additional variables and constraints tighten the LP relaxation itself in any substantial way.

To confirm that CTF and DSF are indeed weak formulations, we ascertain their relative strengths in terms of their LP relaxations. The ratio of the objective value of LP relaxation to the optimal value for some λ_R configurations is presented in Table 3. To assist in visualizing this table, we also plot the values in Fig. 4. The objective values for LP relaxation for hybrid formulation were very close to those of IGTF (often within 1%). So we combine them into one column marked as IGTF/HF.

Based on the above results, we make the following observations:

1. Fig. 4 confirms that CTF indeed has a very weak LP relaxation bound over all λ_R configurations and over all data instances. For the purposes of SpRP as discussed in this paper, CTF does not have any useful features. We do not discuss it any further.
2. The results for DSF are more interesting. For very high values of λ_R ratio, DSF is quite weak. So, DSF does not model the NLHM routing costs well. But as λ_R ratio reduces (λ_v dominates λ_h) and VSR costs become significant, the strength of DSF increases strongly. To confirm this observation, we summarize the values from Table 3 into Table 4 to give the trend of the strength of MIP formulations across the values of ratio λ_R. This is also plotted in Fig. 5.

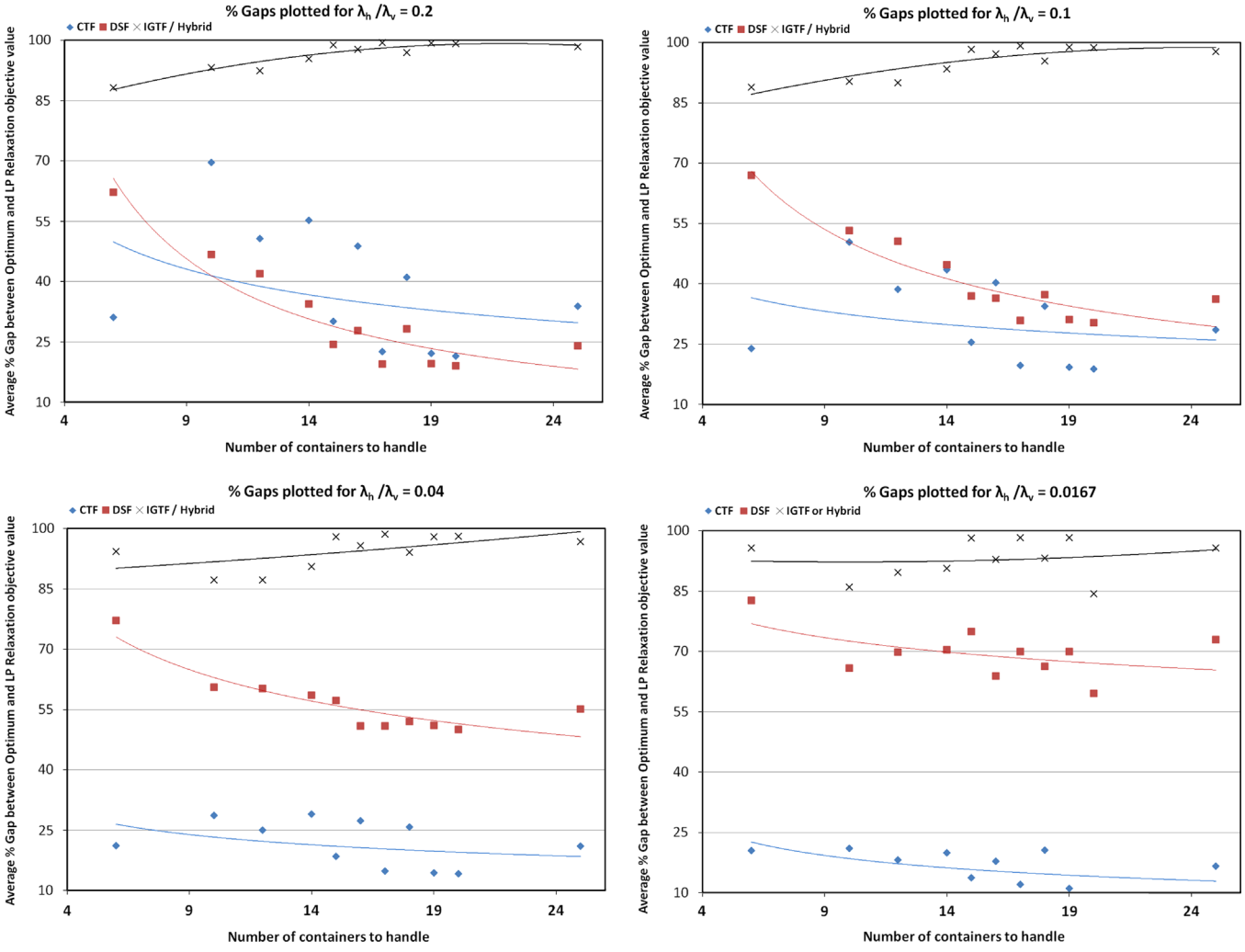

Fig. 4. Plots of objective values of LP relaxation expressed as % of optimum for various configurations of ratio $\lambda_R = \lambda_h/\lambda_v$.

Table 4
Strength of MIP formulations for various λ_R ratio values measured in terms of the objective value of LP relaxations expressed as % of optimal objective value.

Ratio λ_R	CTF	DSF	IGTF/Hybrid
0.2	38.7	29.7	96.8
0.1	31.5	39.8	95.9
0.04	22.3	55.8	94.9
0.0167	16.8	69.7	93.5

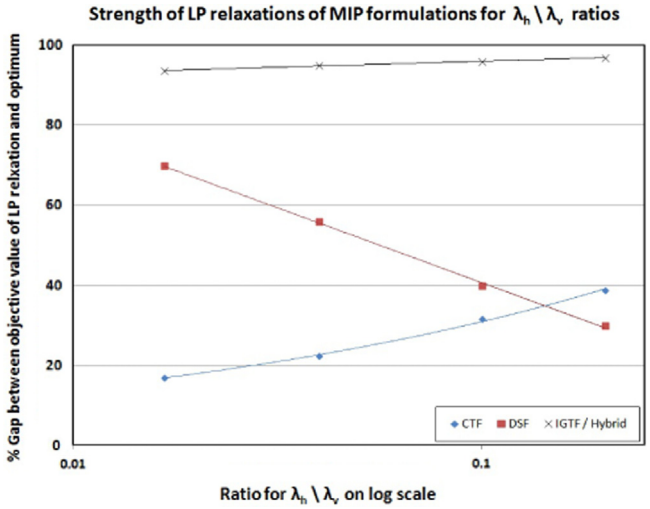

Fig. 5. Strength of MIP formulations for various λ_R ratio values measured in terms of the objective value of LP relaxations expressed as % of optimal objective value.

Table 5
Comparison of variants within Hybrid formulation in terms of time needed (in seconds) to reach optimal solution.

Number of containers	Additional MTZ SEC constraints	Additional cycle variables w_{ijk}	Enforcing transitivity as constraint	Enforcing transitivity as cuts
6	0.386	0.166	0.194	0.082
10	7.956	2.956	2.052	0.698
12	19.492	7.186	6.086	1.962
14	58	20.76	20.546	10.068
15	19.67	13.012	12.854	5.274
16	89.466	33.788	32.496	13.168
17	118.438	38.808	69.856	31.718
18	334.111	78.653	102.562	43.104
19	250.982	96.766	121.886	93.444
20	644.158	341.334	272.954	267.254
25	1121.665	548.5748	767.8614	470.2745
30	-X-	-X-	-X-	492.26

Table 4 and Fig. 5 capture the applicability of the sequencing approach ($\mathbb{Y}_i^k$ variables) over the spectrum of λ_R configurations. Lower values of λ_R correspond to VSR and higher values to NLHM. Fig. 5 indicates that the sequencing approach of $\mathbb{Y}_i^k$ variables in DSF is effective for modeling the stack rearrangement aspect of SpRP, but it is very inefficient for the routing aspect. This makes us question whether the relatively fair performance of hybrid formulation is good because of the $\mathbb{Y}_i^k$ variables or due to the structure of IGTF. We look at IGTF and HF in detail.

The discussion in Section 2 often considered several variants within IGTF and HF. This included alternative definitions involving different decision variables, additional cuts, etc. We now evaluate the relative effectiveness of these all variants from the two MIP formulation. For this, we first compare variants of any one MIP formulation internally to see how they match up with each other (not with other formulations). The results for variants of IGTF are tabulated[1] in Table 5 and plotted in Fig. 6.

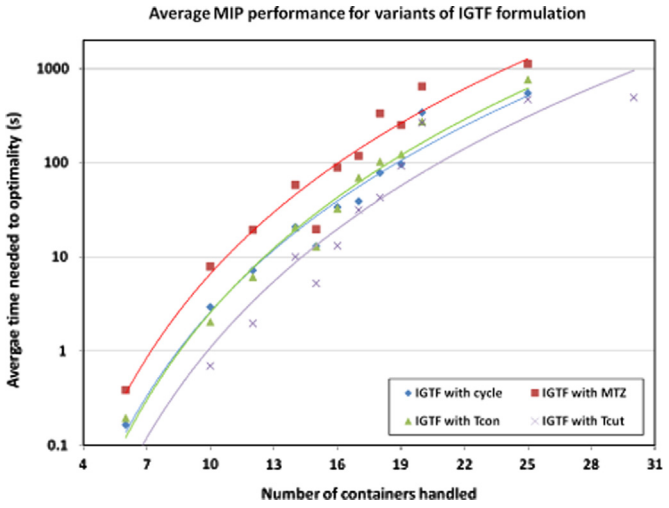

Fig. 6. Comparison of variants within hybrid formulation in terms of time needed (in seconds) to reach optimal solution.

Table 6
Comparison of variants within Hybrid formulation in terms of time needed (in seconds) to reach optimal solution.

No. of Ctrs n	Only basic MIP without any additions	Additional cycle variables w_{ijk}	Enforcing transitivity as constraint	Enforcing transitivity as cuts
6	0.264	0.148	0.25	0.132
10	7.838	2.846	5.786	5.306
12	181.46	17.74	20.606	162.554
14	610.7575	105.134	182.114	383.5033
15	165.894	11.542	47.914	225.722
16	385.885	194.8	362.531	305.12
17	240.01	107.396	161.134	376.77
18	277.2214	292.3888	285.63	168.625
19	709.5	668.354	1016.97	1737.11
20	-X-	764.99	1769.29	-X-

Performance of the IGTF with transitivity enforced as cuts (to remove the sub-tours) is observed to be the best among all IGTF variants. Next, the results for variants within 'Hybrid' formulation are tabulated[1] in Table 6 and plotted in Fig. 7.

Within the variants of *Hybrid formulation*, the best observed performance is for the case where transitivity is enforced. This may be done directly using constraints (as shown in Eq. (8) within the MIP model or even better by the use of cycle variables w_{ijk}. This is in contrast with the observation that the best IGTF performance was for the variant where transitivity was only enforced as lazy cuts. So, we compare the performance of the best observed variant of IGTF against the best variant of Hybrid formulation in Fig. 8.

This plot indicates that the performance of IGTF (with transitivity cuts) is consistently superior to the best variant within HF (where transitivity was enforced via constraints or cycle variables w_{ijk}).

We explain this observation as follows: we had devised IGTF such that it inherently involved very apt ways of modeling the stack rearrangement aspect (VSR) through δ variables and the routing aspect (NLHM) through u variables; it only missed route feasibility

[1] The entry '-X-' indicates that the corresponding value was not found within 1800s of computational time.

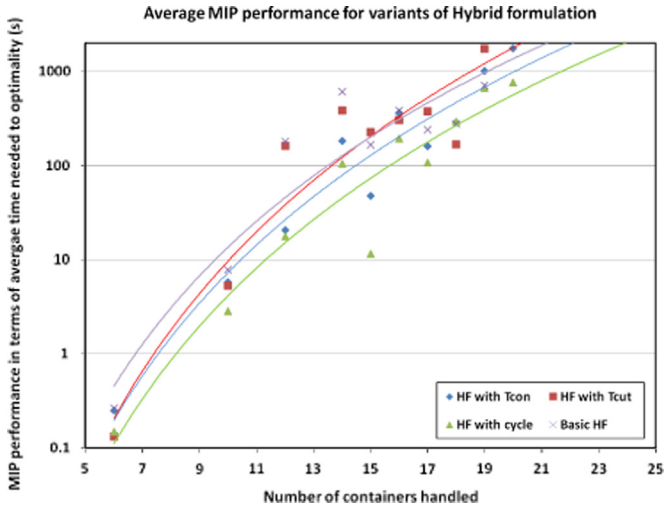

Fig. 7. Comparison of variants within hybrid formulation in terms of time needed (in seconds) to reach optimal solution.

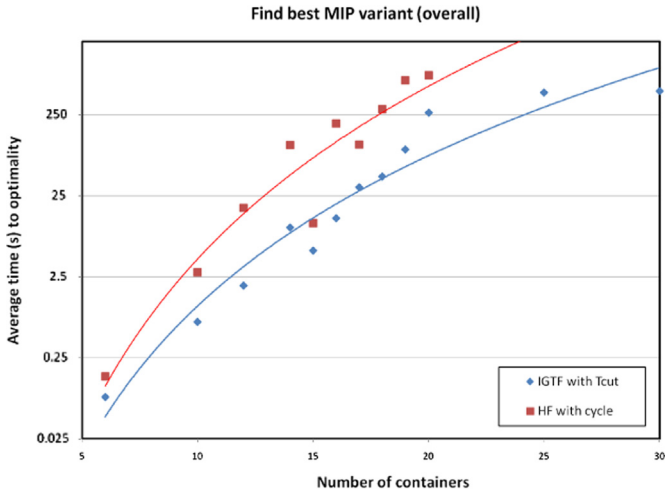

Fig. 8. Determining the overall best MIP variant.

considerations (or SECs). Such a route infeasibility or sub-tour is expected in only a few cases because Eq. (29 (however weak) does enforce an interconnection between the two routes induced by the two sets of variables. So, route feasibility restrictions are best enforced by doing remedial or curative intervention only if δ variables do induce a meaningless route (violation of transitivity). This remedial intervention in terms of transitivity cuts keeps the underlying formulation compact and easier to solve. Any additional constraints or variables (such as w_{ijk} or $\mathbb{Y}_i^k$) is superfluous and in fact detrimental. Thus, the application of Theorems 1 and 2 has led to an novel cross-over of constraints between routing and stacking, resulting in an MIP formulation that performs best among all options explored.

4. Conclusions and future work

In this paper, we have introduced SpRP a new contribution to the literature (to the best of our knowledge, SpRP has not been studied in the literature before.) We presented several traditional MIP formulations that model and solve the SpRP. As a part of the computational analysis, we also showed the strength and applicability of specific formulations for different configurations.

A vital contribution was the logical basis (Theorems 1 and 2) to combine the key concepts from different MIP approaches. This used the inter-applicability of constraints to develop a stronger MIP formulation. We demonstrated the superiority of this approach over many possible formulations/approaches for a wide gamut of problem data instances with different configurations.

Although these approaches are compelling and efficient and although the mathematical foundations in this work have enabled a strengthening of the formulations that we developed, the approaches still struggle to solve larger-sized problem instances. Further, some additional practical considerations and constraints encountered in maritime ports also need to be addressed. For example, container ports may have specific time restrictions by which certain containers must reach a departing ship, train or truck. We also need to consider dynamic/online instances in future studies. We believe that the best way to solve larger SpRP data instances (with additional constraints) is through the development of efficient heuristic approaches.

Acknowledgements

The authors wish to acknowledge the insightful comments of the anonymous reviewers, whose comments have helped improve this paper substantially.

References

[1] Aslidis A. Combinatorial algorithms for stacking problems [Ph.D. thesis]. Massachusetts Institute of Technology; 1989.
[2] Baldacci R, Hadjiconstantinou E, Mingozzi A. An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. Oper Res 2004;52:723–38.
[3] Beasley JE, Krishnamoorthy M, Sharaiha YM, Abramson D. Scheduling aircraft landings–the static case. Transp Sci 2000;34:180–97.
[4] Bianco L, Dell'Olmo P, Giordani S. Minimizing total completion time subject to release dates and sequence dependent processing times. Ann Oper Res 1999;86:393–415.
[5] Bianco L, Ricciardelli S, Rinaldi G, Sassano A. Scheduling tasks with sequence-dependent processing times. Nav Res Logist 1988;35:177–84.
[6] Bigeas L, Gamache M, Savard G. The time-dependent traveling salesman problem and single machine scheduling problems with sequence dependent setup times. Discret Optim 2008;5:685–99.
[7] Bish E, Chen F, Leong Y, Nelson B, Cheong Ng J, Simchi-Levi D. Dispatching vehicles in a mega container terminal. In: Kim K, Günther H-O, editors. Container terminals and cargo systems. Berlin, Heidelberg: Springer; 2007. p. 179–94.
[8] Bohrer P. Crane Scheduling in container terminals [Ph.D. thesis]; 2005.
[9] Caserta M, Voß S, Sniedovich M. Applying the corridor method to a blocks relocation problem. OR Spectr 2011;33:915–29.
[10] de Castillo B, Daganzo CF. Handling strategies for import containers at marine terminals. Transp Res Part B: Methodol 1993;27:151–66.
[11] Charbit P, Thomassé S, Yeo A. The minimum feedback arc set problem is np-hard for tournaments. Comb Probab Comput 2007;16:1–4.
[12] Chen L, Langevin A, Riopel D. A tabu search algorithm for the relocation problem in a warehousing system. Int J Prod Econ 2011;129:147–56.
[13] Chen T. Yard operations in the container terminal–a study in the 'unproductive moves'. Marit Policy Manag 1999;26:27–38.
[14] Christofides N, Colloff I. The rearrangement of items in a warehouse. Oper Res 1973;21:577–89.
[15] Claus A. A new formulation for the travelling salesman problem. SIAM J Algebr Discret Methods 1984;5:21–5.
[16] Dantzig G, Fulkerson R, Johnson S. Solution of a large-scale traveling-salesman problem. Oper Res 1954;2:393–410.
[17] Dayama NR, Ernst A, Krishnamoorthy M, Narayanan V, Rangaraj N. CSIRO technical report EP136768. Technical Report CSIRO Mathematics and Information Sciences; 2013.
[18] Dekker R, Voogd P, Asperen E. Advanced methods for container stacking. In: Kim K, Günther H-O, editors. Container terminals and cargo systems. Berlin, Heidelberg: Springer; 2007. p. 131–54.
[19] Ernst AT, Krishnamoorthy M, Storer RH. Heuristic and exact algorithms for scheduling aircraft landings. Networks 1999;34:229–41.
[20] Forster F, Bortfeldt A. A tree search procedure for the container relocation problem. Comput Oper Res 2012;39:299–309.
[21] Gambardella LM, Mastrolilli M, Rizzoli AE, Zaffalon M. An optimization methodology for intermodal terminal management. J Intell Manuf 2001; 12:521–34.

[22] Gutin G. Traveling salesman problem. Springer; 2009, http://dx.doi.org/10.1007/978-0-387-74759-0_687.
[23] Ileri Y, Bazaraa M, Gifford T, Nemhauser G, Sokol J, Wikum E. An optimization approach for planning daily drayage operations. Cent Eur J Oper Res 2006;14:141–56.
[24] Kara I, Bektas T. Integer linear programming formulations of multiple salesman problems and its variations. Eur J Oper Res 2006;174:1449–58.
[25] Kim K. Evaluation of the number of rehandles in container yards. Comput Ind Eng 1997;32:701–11 (cited By (since 1996) 82).
[26] Kim KH, Bae JW. Re-marshaling export containers in port container terminals. Comput Ind Eng 1998;35:655–8 (selected oapers from the 22nd ICC and IE conference).
[27] Kim KH, Kim HB. The optimal determination of the space requirement and the number of transfer cranes for import containers. Comput Ind Eng 1998;35:427–30 (selected papers from the 22nd ICC and IE conference).
[28] Kim K-H, Park Y-M. A crane scheduling method for port container terminals. Eur J Oper Res 2004;156:752–68.
[29] Kozan E. Optimising container transfers at multimodal terminals. Math Comput Model 2000;31:235–43.
[30] Laporte G. Modeling and solving several classes of arc routing problems as traveling salesman problems. Comput Oper Res 1997;24:1057–61.
[31] AAPA rankings. Traffic of containers handled at major container ports. ⟨http://en.wikipedia.org/wiki/List_of_world's_busiest_container_ports⟩; 2010 (last accessed: 19.05.13).
[32] Container HandBook from GDV. Structural and testing regulations. ⟨http://www.containerhandbuch.de/chb_e/stra/index.html?/chb_e/stra/stra_03_01_00.html⟩; 2013 (last accessed: 19.05.13).
[33] World Shipping council. Data on container ports. ⟨http://www.worldshipping.org/abouttheindustry/globaltrade/top50worldcontainerports⟩; 2013 (last accessed: 19.05.13).
[34] Miller CE, Tucker AW, Zemlin RA. Integer programming formulation of traveling salesman problems. J Assoc Comput Mach 1960, http://dx.doi.org/10.1145/321043.321046.
[35] Mittal AK. Optimal rearrangement of objects [Ph.D. thesis]. Case Western Reserve University; 1975.
[36] Murty KG. A decision support system for operations in a container terminal. Decis Support Syst 2005;39:309–32.
[37] Peterkofsky RI, Daganzo CF. A branch and bound solution method for the crane scheduling problem. Transp Res Part B: Methodol 1990;24:159–72.
[38] Ropke S, Cordeau J-F, Laporte G. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. Networks 2007;49:258–72.
[39] Srour FJ, van de Velde S. Are stacker crane problems easy? A statistical study Comput Oper Res 2013;40:674–90.
[40] Stahlbock R, Voß S. Operations research at container terminals: a literature update. OR Spectr 2007;30:1–52.
[41] Svestka JA, Huckfeldt VE. Computational experience with an M-salesman traveling salesman algorithm. Manag Sci 1973;19:790–9.
[42] Vos S, Stahlbock R, Steenken D. Container terminal operation and operations research – a classification and literature review. OR Spectr 2004;26:3–49.
[43] Wong A, Kozan E. Optimization of container process at seaport terminals. J Oper Res Soc 2009;61:658–65.
[44] Zheng H, Li L, Ren S. Study on container terminal and production resources model, vol. 387. ASCE; 2010.
[45] Zhu Y, Lim A. Crane scheduling with spatial constraints: mathematical model and solving approaches. In: AIM 30-2004, eighth international symposium on artificial intelligence and mathematics, Fort Lauderdale, vol. 6; 4 January 2004. Citeseer. p. 1–10.