Modeling and solution of the joint quay crane and truck scheduling problem

Lixin Tang a,⇑, Jiao Zhao a, Jiyin Liu b

a Liaoning Key Laboratory of Manufacturing System and Logistics, The Logistics Institute, Northeastern University, Shenyang 110819, China
b School of Business and Economics, Loughborough University, Leicestershire LE11 3TU, UK

A R T I C L E I N F O

Article history:
Available online 6 September 2013

Keywords:
Container terminal
Quay cranes
Internal trucks
Joint scheduling problem
Particle swarm optimization

A B S T R A C T

This paper addresses the joint quay crane and truck scheduling problem at a container terminal, considering the coordination of the two types of equipment to reduce their idle time between performing two successive tasks. For the unidirectional flow problem with only inbound containers, in which trucks go back to quayside without carrying outbound containers, a mixed-integer linear programming model is formulated to minimize the makespan. Several valid inequalities and a property of the optimal solutions for the problem are derived, and two lower bounds are obtained. An improved Particle Swarm Optimization (PSO) algorithm is then developed to solve this problem, in which a new velocity updating strategy is incorporated to improve the solution quality. For small sized problems, we have compared the solutions of the proposed PSO with the optimal solutions obtained by solving the model using the CPLEX software. The solutions of the proposed PSO for large sized problems are compared to the two lower bounds because CPLEX could not solve the problem optimally in reasonable time. For the more general situation considering both inbound and outbound containers, trucks may go back to quayside with outbound containers. The model is extended to handle this problem with bidirectional flow. Experiment shows that the improved PSO proposed in this paper is efficient to solve the joint quay crane and truck scheduling problem.

1. Introduction

Container terminals are crucial interfaces between land and sea transportation modes. Container ships carry inbound containers to a terminal and carry outbound containers away. At container terminals, containers are transferred from one mode to another (Vis & De Koster, 2003). Inbound containers are unloaded from container ships by quay cranes and then transported by internal trucks to storage yard where they are stacked by yard cranes to their allocated positions waiting for the consignees to pick up. Outbound containers are handled in the opposite direction. Shippers send the containers into the terminal and yard cranes store them in their allocated positions. Later they are retrieved by yard cranes and transported by trucks to quayside where they are loaded onto ships by quay cranes. Fig. 1 illustrates the handling processes for inbound and outbound containers.

As described above, the whole terminal operation is very complex and involves different types of equipments. The operations of the equipments need to be planned appropriately in order to ensure an efficient operation, especially for busy terminals with increasing throughputs.

Operations planning and scheduling at container terminals include berth allocation to incoming ships, quay crane scheduling, ground transport equipment scheduling, yard crane scheduling, storage space allocation, and so on. From the above brief description of terminal operations, we can see that quay cranes are the equipment directly unloading containers from and loading containers to ships. Trucks provide ground transportation of containers between the quay cranes and the storage yard. Effective quay crane scheduling and truck scheduling are both important in terminal management. The operations of quay cranes and trucks are closely linked and need good coordination to avoid efficiency loss due to waiting for each other. Therefore in this paper we study the problem of jointly scheduling quay cranes and trucks.

The paper is organized as follows. In the next section, we first briefly review related research in the existing literature. Then a detailed problem description is given in Section 3. A mixed-integer linear programming model is formulated in Section 4 for the problem with unidirectional flow. Section 5 provides one optimal property, four valid inequalities and two lower bounds for the unidirectional flow problem. A Particle Swarm Optimization (PSO) algorithm is proposed in Section 6 to solve the unidirectional flow problem. Section 7 considers the more general situation with ships
Involving both unloading and loading of containers. The model and the PSO algorithm are extended to solve this general problem with bidirectional flow. The lower bounds are also modified to suit the general situation. Experiment results are reported in Section 8 showing that the proposed PSO algorithm is effective and efficient in solving the joint quay crane and truck scheduling problem with unidirectional or bidirectional flow. Section 9 provides conclusions.

2. Literature review

There has been little research on truck scheduling at container terminals. Nishimura, Imai, and Papadimitriou (2005) addressed a trailer routing problem at a container terminal. They formulated both the single-trailer and multi-trailer problems for pickup and delivery of containers in the terminal, and then employed genetic algorithm to solve them. Most previous work on this topic considered the problem of scheduling other types of ground transport equipment, such as transfer cranes, straddle carriers, and automated guided vehicles (AGV). For example, Kim and Kim (1997) investigated the routing problem for a single transfer crane and focused on outbound containers to be loaded onto a ship. A straddle carrier routing problem was studied by Kim and Kim (1999), and a beam search algorithm was used for solutions. Vis, De Koster, Roodbergen, and Peeters (2001) determined the minimum number of AGVs at a semi-automated container terminal by a minimum flow algorithm. However, none of the previous ground transport equipment scheduling research considered the link with the quay crane scheduling.

Due to the practical importance of crane scheduling problem, it has received much research attention. Zhang, Wan, Liu, and Linn (2002) studied the dynamic Rubber Tired Gantry Cranes (RTGCs) deployment problem, and a mixed integer programming model has been formulated. In their paper, the objective is to minimize the total workload overflow. The multiple yard cranes scheduling problem with inter-crane interference was examined by Ng (2005), and a dynamic programming-based heuristic was proposed to solve the problem. Research on the quay crane (QC) scheduling problem started twenty years ago (Daganzo, 1989), and the problem received more attention recently. Lim, Rodrigues, Xiao, and Zhu (2004) discussed the quay crane scheduling problem with three particularly common constraints, which were the noncrossing constraint, the neighborhood constraint and the job-separation constraint, and three algorithms were proposed for obtaining solutions. In their study, a job was defined as a collection of cargo from a given area on a ship. Kim and Park (2004) studied the quay crane scheduling problem, in which a cluster of containers to load or unload was called as a task. A branch and bound (B & B) method and a greedy randomized adaptive search procedure (GRASP) were used to obtain solutions. Liu, Wan, and Wang (2006) modeled the quay crane assignment and scheduling problem considering noncrossing requirement and safety distance. The problem was then decomposed to two levels and solved efficiently using smaller models. The sequence for quay cranes to handle holds in one container vessel were decided by Lee, Wang, and Miao (2008), and a genetic algorithm was presented for solving the problem.

There have been also some papers discussing the integration of different decision problems in container terminals. For example, Bish, Leong, Li, Ng, and Simchi-Levi (2000) studied a problem of allocating storage locations for unloading containers as well as dispatching trucks to transport the containers. A heuristic algorithm was proposed to get solutions. A multiple-crane-constrained scheduling problem was also considered by Bish (2003), in which the unloading and loading containers were both contained. A formulation was presented by Imai, Chen, Nishimura, and Papadimitriou (2008) for the simultaneous berth and quay crane allocation problem, and a heuristic was proposed employing genetic algorithm to obtain near solutions. The problem of scheduling both trucks and quay cranes has rarely been studied. Li and Vairaktarakis (2004) proposed an optimal algorithm and some heuristic algorithms to solve the loading and unloading problem with one quay crane and several vehicles. Chen, Bostel, Dejax, Cai, and Xi (2007) considered the integrated scheduling problem of container handling systems including quay cranes, yard cranes and yard vehicles. They viewed the problem as a Hybrid Flow Shop Scheduling problem with precedence and blocking constraints (HFSS-B). Instead of detailed allocation and scheduling of quay cranes to perform the loading/unloading tasks, they assumed that each quay crane covers a given range of bays and that the ranges for different quay cranes are not overlapping to avoid potential collision. Though such a restriction simplifies the problem, it may make the optimal solution infeasible. In this paper we address the joint quay crane and truck scheduling problem and consider conflicts between quay cranes explicitly. We decide not only the assignment of containers to the quay cranes and trucks, but also the sequence of tasks to be performed by each quay crane and each truck.

3. The joint quay crane and truck scheduling problem

When container ships berth at the terminal, the loading and unloading operations are carried out by quay cranes. The quay crane scheduling problem needs to decide the assignment of quay cranes to perform the tasks of handling individual containers and to determine the handling sequence of the containers assigned to each quay crane. The containers unloaded from the ships are to be transported by trucks to the yard for storage; while the containers to be loaded onto the ships need to be transported by trucks from the yard to the quay cranes for loading. The truck scheduling problem assigns the transport tasks to the trucks and decides the handling sequence of containers assigned to each truck. In this paper, we consider the integrated quay crane and truck scheduling problem and make all the above decisions jointly.

In the traditional terminal operation, quay cranes are often scheduled first to process coming ships, and then trucks are assigned to serve specific quay cranes. In this method, each working quay crane is teamed up with a fixed set of trucks. The quay crane and the trucks may often need to wait for each other. Take the unloading process as an example. If a container is unloaded by a quay crane, but none of the trucks has arrived, the quay crane has to hold the container and wait for a truck. On the other hand, if the quay crane has not completed the unloading of a container when a truck comes, the truck has to wait. Such waiting causes productivity loss and shows a need for better coordination between the operations of quay cranes and trucks. As shown in Fig. 2, there are two ships berthed at the terminal. Quay crane 1 and Quay crane 2 are allocated to Ship 1 and Ship 2, respectively.
Truck 1 and Truck 2 are both allocated to Quay crane 1 for transporting containers to the yard; Truck 3 is allocated to Quay crane 2. When there is no available truck to transport the container unloaded by Quay crane 2, Quay crane 2 has to hold the container and wait for a truck. At the same time, both Truck 1 and Truck 2 which have been allocated to Quay crane 1 have to wait in a queue for transporting containers unloaded by Quay crane 1. To avoid such a case, in this paper, we determine the quay crane scheduling and the truck scheduling jointly so as to minimize the turnaround time of the ships, which is shown in Fig. 3. In Fig. 3, we can see that Truck 1 turns to serve Quay crane 2. As a result, Quay crane 2 does not have to wait, and thus the total productivity of the container terminal can be improved.

The decisions in our problem include the allocation of quay cranes and trucks for handling containers and the sequence for handling tasks allocated to each quay crane and each truck. A quay crane or a truck is assumed to handle at most one container at a time. Some quay cranes are able to unload two twenty-foot containers simultaneously and put them together on one truck. In this case two containers can be considered as one. In our integrated scheduling problem we consider the following factors:

1. Quay cranes are heavy equipment and their movements are slow. Hence once a quay crane starts the handling process in a bay, it will complete all the work in this bay before it can go to another bay. This requirement is practical in container terminals.

2. There are precedence relationships between containers. For example, during the unloading process in one bay, a container must be unloaded before the unloading of containers below it. These precedence relations are given and must be satisfied.

3. The quay cranes move along the same track and so cannot cross each other. In this paper, we do not assume any fixed range of bays for any quay crane. A quay crane may be allocated to any bay that it can reach so long as the quay cranes do not cross each other and keep a safety distance from each other. This will allow the quay cranes to realize their full potentials.

4. The two types of equipment may wait for each other when the coordination between them is poor, which will cause efficiency loss for the terminal. Therefore, the waiting time of quay cranes and trucks should be minimized.

4. MILP model for the unidirectional flow problem

In this section we present a mathematical model for integrated quay crane and truck scheduling to unload containers from ships. We assume that the yard cranes have sufficient handling capacity so that the trucks do not have to wait in the yard. Note that the berthing positions of the ships are known from the berth allocation plan. Each container to be unloaded has a known location on the ship and a known destination location in the yard. The quay crane for unloading this container must be at the bay where the
container located. Therefore, the time for a truck to transport a container \(j\) from the quayside to the yard and return to the quayside is known in advance. We denote the time as \(P_{j2}\).

**Notations:**
- \(P_{j1}\): The quay crane time needed to unload container \(j\).
- \(P_{j2}\): The time for a truck to transport container \(j\) from the quay crane to the container’s destination in the yard and to return to the quay crane.
- \(Q\): The set of quay cranes; \(Q = \{1, 2, \ldots, |Q|\}\), the quay cranes are numbered sequentially along the quay from one end to the other.
- \(T\): The set of trucks.
- \(M\): A very large number.
- \(\Omega\): The set of containers to be unloaded from the ships.
- \(\Omega_0\): \(\Omega_0 = \Omega \cup \{0\}\), \(0\) represents the initial state of any equipment.
- \(\Omega_T\): \(\Omega_T = \Omega \cup \{F\}\), \(F\) represents the final state of any equipment.
- \(\Psi\): The set of container pairs \((i, j)\) such that \(i\) must precede \(j\) in unloading.
- \(E\): The set of bays in the ships to be unloaded; \(E = \{1, 2, \ldots, |E|\}\), the bays are numbered sequentially along the quay in the same direction as for the quay cranes.
- \(B_h\): The set of containers in bay \(h\).
- \(L_i\): The bay which container \(i\) is in.

**Decision variables:**
- \(C_{i1}\): The completing time of unloading container \(i\).
- \(C_{i2}\): The completing time of the round trip for transporting container \(i\).
- \(A_h\): The starting time of bay \(h\).
- \(D_h\): The completing time of bay \(h\).
- \(Z_{iq}\): A binary variable. It equals 1 if container \(i\) is unloaded immediately before container \(j\) by quay crane \(q\), 0 otherwise.
- \(Z_{iq}^u\): A binary variable. It equals 1 if container \(i\) is allocated to quay crane \(q\) for unloading, 0 otherwise.
- \(Y_{hq}\): A binary variable. It equals 1 if bay \(h\) is finished no later than bay \(h’\) starts, 0 otherwise.
- \(X_{iu}^v\): A binary variable. It equals 1 if bay \(i\) is handled by quay crane \(q\), 0 otherwise.
- \(X_{ju}\): A binary variable. It equals 1 if container \(i\) is allocated to truck \(u\) for transporting, 0 otherwise.

Based on the above assumptions and notations, we formulated the problem as following mathematical model:

\[
\begin{align*}
\text{min} & \quad \max_{i \in \Omega} C_{i2} \\
C_{i1} + P_{i2} & = C_{i2} \quad i \in \Omega \\
C_{i1} + P_{j1} & \leq C_{j1} \quad (i, j) \in \Psi \\
A_h + P_{i1} & \leq C_{i1} \quad i \in B_h, h \in E \\
D_h & \geq C_{i1} \quad i \in B_h, h \in E \\
D_h - A_{h'} + Y_{h}M & \geq 0 \quad h, h' \in E \\
D_h - A_{h'} - (1 - Y_{h'})M & \leq 0 \quad h, h' \in E
\end{align*}
\]

The objective (1) of the model is to minimize the makespan, i.e., the time to complete the unloading and transporting operations of all required containers. Constraints (2) ensure that there must be a required transport time between the completion of unloading a container and the completion of transporting it. Constraints (3) ensure that the unloading of containers must satisfy the precedence constraints. Constraints (4) state that the unloading of a bay starts once the unloading of any container in the bay has completed. Constraints (5) guarantee that the unloading of a bay completes only when the unloading of every required container in the bay has completed. Constraints (6) define the variable \(Y_{hq}\), i.e., set \(Y_{hq} = 1\) if bay \(h\) starts after bay \(h'\) finishes. Similarly, constraints (7) set \(Y_{hq} = 0\) when \(h\) finishes later than bay \(h'\) starts. Constraints (8) avoid conflicts in the relative positions of different quay cranes. Note that quay cranes are numbered sequentially.
along the quay from one end to the other, and the bays are also numbered sequentially along the quay in the same direction as for the quay cranes. When \( h < h' \) and if they are performed simultaneously, i.e., \( Y_{h0} + Y_{h'0} = 0 \), constraints (8) become \( \sum_{i=0}^{Q} Q_{h0} - \sum_{i=0}^{Q} Q_{h'0} = 0 \), meaning that the index of the crane performing \( h < \) the index of the crane performing \( h' \). Constraints (9) require that each bay is assigned to one and only one quay crane. Constraints (10) and (11) ensure that each quay crane starts at its initial state 0, and ends at its final state \( F \), respectively. Constraints (12) ensure the flow balance for each quay crane before and after performing a container. Constraints (13) determine the completion time of unloading the first container by each quay crane \( q \). Constraints (14) require that each container is assigned to one and only one quay crane. Constraints (15) mean that if quay crane \( q \) unloads container \( i \), it must be in the initial state or have unloaded another container immediately before that. For each quay crane, constraints (16) relate the order of unloading the containers assigned to it and the completion times of unloading these containers. These constraints also effectively eliminate sub-tours in the unloading sequence. Constraints (17) and (18) ensure that every truck starts at its initial state 0, and ends at its final state \( F \), respectively. Constraints (19) ensure the flow balance of each truck before and after transporting a container. Constraints (20) determine the completion time of transporting the first container by truck \( u \). Constraints (21) require each container is assigned to one and only one truck. Constraints (22) and (23) have the same functions for trucks as constraints (15) and (16) for quay cranes, respectively. Note that constraints (23) should be understood considering the relations in constraints (2). Constraints (24) ensure that all the containers in a bay must be unloaded by the quay crane assigned to the bay. Constraints (25) and (26) specify the ranges of variable values.

5. Valid inequalities, property, and lower bounds for the unidirectional flow problem

5.1. Valid inequalities

In this section we describe several valid inequalities for the joint quay crane and truck scheduling problem. All the proposed valid inequalities are redundant for the model in Section 4, but the running time of CPLEX software can be reduced after added these valid inequalities to the model.

**Bay completion time inequality.** \( D_h \geq \sum_{i=0}^{Q} P_{i1} \) \( \quad (27) \)

Inequality (27) ensures that each bay must be finished no earlier than the total processing time of all the containers in the bay. It is valid because each bay can only be processed by one quay crane, and the quay crane needs to unload containers one by one.

**The average workload of the quay cranes inequality.** \( \sum_{i=Q}^{1} P_{i1} / |Q| \leq \max_{i=Q} C_{i2} \) \( \quad (28) \)

Inequality (28) ensures that the average workload of the quay cranes must not be greater than the makespan.

**Variable relationship inequality.** \( Z_{jiq} \leq Z_{jq} \quad i, j \in \Omega, \quad q \in Q \) \( \quad (29) \)

Inequality (29) ensures that if container \( i \) is not unloaded by quay crane \( q \), then \( Z_{jiq} = 0 \). When container \( i \) is unloaded by quay crane \( q \) immediately before container \( j \), then \( Z_{jq} = 1 \).

**Variable relationship inequality.** \( X_{jiu} \leq X_{iu} \quad i, j \in \Omega, \quad u \in T \) \( \quad (30) \)

Inequality (30) ensures that if container \( i \) is not transported by truck \( u \), then \( X_{jiu} = 0 \). When container \( i \) is transported by truck \( u \) immediately before container \( j \), then \( X_{iu} = 1 \).

5.2. A property of the optimal solutions

For the above joint quay crane and truck scheduling problem, we observe a property for the optimal solutions, and the details are as follows.

**Property 1.** Consider the problem of unloading containers in one bay. If \( (i, j) \notin \Psi \) for any container pair \( (i, j) \) in this bay and if there are sufficient trucks so that the quay crane never need to wait for trucks, then it is optimal to unload the containers in the bay in descending order of their transport times.

**Proof.** Given any schedule of unloading this bay, if the containers in it are not sequenced in the above order, there must be two containers \( i \) and \( j \) in the schedule such that \( i \) is immediately before \( j \) and \( P_{i2} < P_{j2} \). Let \( Q \) denote the time at which the crane completes the unloading of the container immediately before \( i \). The maximum completion time for transporting \( i \) and \( j \) in this schedule can be calculated as \( Q = \max((Q + P_{i1} + P_{j2}), (Q + P_{j1} + P_{j2})) \).

We can construct a new schedule by exchanging the unloading order of \( i \) and \( j \). This will not affect the completion times of any container before them. The maximum completion time for transporting \( i \) and \( j \) in this new schedule will be \( Q = \max((Q + P_{i1} + P_{j2}), (Q + P_{j1} + P_{j2})) \). Clearly, neither of the two terms in \( Q \) is larger than the second term in \( Q \), and so \( Q_2 < Q_1 \). It is easy to see that the maximum completion time for unloading \( i \) and \( j \) keeps unchanged, and thus the completion times of the containers after them will not change. Therefore, the new schedule is not worse than the original one.

Continue such exchanges we can obtain a schedule in which containers are unloaded in descending order of their transport times and which is not worse than the original one. Since this can be done for any given schedule, this proves that it is optimal to unload the containers in the bay in descending order of their transport times.

5.3. Lower bounds

Since each container needs to be unloaded and then transported, the optimal makespan of the unloading stage plus the minimum transporting time will be a lower bound of the original problem. We give two lower bounds below by minimizing the makespan of the first stage.

**Lower bound 1.** The makespan of the optimal solution must not be less than the average workload of the quay cranes plus the minimum transporting time.

This is obviously a lower bound, because it considers the minimal possible completion time of the first stage under the perfect balance of crane workloads and the minimum transporting time of the last unloaded container. However, it does not consider the crane position restrictions and so it will not be tight. Lower bound 2 below considers such restrictions and the actual workload of each crane.
Lower bound 2. We define the following additional notation and use a small model to calculate lower bound 2.

\[ S_h = \sum_{i \in \{0,1\}} P_{i,h} \]  
\[ w_{h|q} \]  
\[ F_1 \]  

The total unloading time of all the containers in bay \( h \) 
If bay \( h' \) is unloaded (or loaded) immediately after bay \( h \) on quay crane \( q \), it equals to 1, otherwise, it equals to 0 
The earliest time at which the unloading of all bays completes

The lower bound model is as follows.

Minimize \( F_1 + \min_i \{ P_{i,h} \} \) 
Subject to constraints (6), (7), (8), (9), (25), (26) and

\[ F_1 \geq D_h \quad h \in E \]  
\[ D_h \geq A_h + S_h \quad h \in E \]  
\[ \sum_{h \in E} w_{hq} = 1 \quad q \in Q \]  
\[ \sum_{h \in E} w_{hq} = 1 \quad q \in Q \]  
\[ \sum_{h \in E} w_{h|q} - \sum_{h' \in E} w_{h'|q} = 0 \quad h \in E, \quad q \in Q \]  
\[ D_h - A_h = S_h \quad h \in E \]  
\[ D_h = S_h + M(1 - w_{hq}) \quad h \in E \]  
\[ w_{hq} - w_{h|q} = Y_{hq} \quad h \in E, \quad q \in Q \]  
\[ w_{h|q} \in \{0,1\} \quad q \in Q, \quad h \in E \]

Constraints (32) define the decision variables \( F_1 \). Constraints (33) ensure that a bay completes only when it has been processed for the required time. Because \( F_1 \) is the earliest time at which the unloading stage can finish and the constant \( \{ P_{i,h} \} \) is the minimum transporting time needed for a container, the optimal objective of this model is a lower bound for our problem. Since the model treats each bay as one job and considers only the allocation of cranes, it is small in size and can be easily solved. Constraints (34) and (35) ensure that every quay crane starts at its initial state 0, and ends at its final state \( F \), respectively. Constraints (36) ensure the flow balance for each quay crane before and after performing a bay. Constraints (37) determine the completion time of performing the first bay by each quay crane \( q \). For each quay crane, constraints (38) relate the order of performing the bays assigned to it and the completion times of performing these bays. These constraints also effectively eliminate sub-tours in the unloading sequence. Constraints (39) mean that if quay crane \( q \) performs bay \( h \), it must be in the initial state or have performed another bay immediately before that. Constraints (40) specify the ranges of variable values.

6. PSO algorithm

In practice the number of containers to be unloaded from a ship is often very large. For problems of practical size, the model in Section 4 cannot be solved using standard software within reasonable time. Therefore we have to consider heuristic solutions.

Particle Swarm Optimization (PSO) was first presented by Eberhart and Kennedy (1995), and they used it to optimize the continuous nonlinear functions. Liao, Tseng, and Luarn (2007) successfully applied a PSO algorithm to solve a discrete optimization problem, and Marinakis and Marinaki (2010) developed a PSO algorithm for traveling salesman problem. In this paper we propose an improved PSO algorithm to solve the joint quay crane and truck scheduling problem.

PSO is a population-based optimization algorithm in which each particle representing a solution has a position and a velocity at each iteration. While the position reflects the quality of the solution, the velocity determines where it will move in the next iteration. The ordinary updating formulae of velocity and position are as follows:

\[ v_{ik}^{t+1} = v_{ik}^t + c_1 r_1 (p_{bestik} - x_{ik}^t) + c_2 r_2 (g_{best} - x_{ik}^t) \]  
\[ x_{ik}^{t+1} = x_{ik}^t + v_{ik}^{t+1} \]

where \( x_{ik}^{t+1} \) and \( x_{ik}^t \) represent the current and previous position of particle \( i \) on dimension \( k \), respectively. \( v_{ik}^{t+1} \) and \( v_{ik}^t \) represent the current and previous velocity of particle \( i \) on dimension \( k \), respectively. The acceleration weights \( c_1 \) and \( c_2 \) are set to determine whether a particle flies toward the best position it has ever reached or the global best position of the whole swarm. \( r_1 \) and \( r_2 \) are random numbers generated in \([0,1]\). \( p_{bestik} \) and \( g_{best} \) are the best position of particle \( i \) on dimension \( k \) up to iteration \( t \), and the best position of the whole swarm on dimension \( k \) until iteration \( t \)

The velocity in the basic PSO may grow unlimitedly and this affects the convergence around the optimal solution. To overcome this problem, Shi and Eberhart (1998), Shi and Eberhart (1999) modified the velocity updating formula as follows:

\[ v_{ik}^{t+1} = \omega v_{ik}^t + c_1 r_1 (p_{bestik} - x_{ik}^t) + c_2 r_2 (g_{best} - x_{ik}^t) \]  
\[ x_{ik}^{t+1} = x_{ik}^t + v_{ik}^{t+1} \]

where \( \omega = \frac{\omega_{max} - \omega_{min}}{\omega_{max} - \omega_{min}} + \omega_{min} \), which is the inertia weight. \( \omega_{max} \) and \( \omega_{min} \) represent the maximum and the minimum values of the inertia weight, respectively.

6.1. Solution representation

The scheduling problem in this paper has to decide quay crane and truck allocations and the sequence of containers handled by each quay crane and each truck. Therefore we need to encode the particles in terms of the problem, and decode them to get the solutions. Because the problem has two stages consisting of the unloading operations by quay cranes and the transporting operations by trucks, we encode quay crane allocation and the container sequence for each quay crane and then allocate the trucks to containers in terms of the quay crane scheduling sequence.

Let \( N = |Q| \). We use a 2N-dimensional sequence to represent a solution. The first \( N \) dimensions indicate the quay crane allocation. The value in each of the dimensions is a real number between 0 and \( |Q| \). The value in dimension \( i \), \( X_i \), represents that container \( i \) is allocated to crane \( X_i \). The second \( N \) dimensions indicate the sequence on each crane. The values are real numbers between 0 and \( N \). The value in dimension \( i + \) is for container \( i \). The unloading sequence for a quay crane is determined by the ascending order of the values in those dimensions corresponding to the containers allocated to this crane. For example, Table 1 shows the encoding of a solution for a problem with 5 \((N = 5)\) containers and 2 quay cranes. The corresponding solution after decoding is in Table 2. In this solution crane 1 unloads containers 4, 2 and 5 in this order, and crane 2 unloads container 1 and then container 3.
the direction a particle flies is determined not only by the best position of the global best particle and the best position this particle has ever reached, but also by the positions of two randomly selected particles. Therefore, the particle will be more likely to get out of local optima to search for better solutions.

6.4. The PSO procedure

Using the above components, we can present the complete PSO procedure for solving the multiple crane and truck scheduling problem.

**Step 1.** Initialize \( l \) particles as a swarm to obtain the quay crane allocation and the sequence of containers on each quay crane (as demonstrated in Tables 1 and 2). Set iteration number \( \tau = 1 \).

**Step 2.** For \( l = 1, 2, \ldots, l \), make sure that the containers in the same bay must be serviced by the same quay crane in the initial solutions (as demonstrated in Table 3), and the precedence between containers must be satisfied (as demonstrated in Table 4).

**Step 3.** For \( l = 1, 2, \ldots, l \), if conflict occurs between quay crane \( i \) and quay crane \( j (i < j) \), quay crane \( i \) waits until quay crane \( j \) completes the unloading task in current bay. Continue this process until there is no conflict between any two quay cranes.

**Step 4.** For \( l = 1, 2, \ldots, L \), allocate the trucks to containers in terms of the quay crane scheduling sequence.

**Step 4.1.** Arrange all the containers based on the ascending order of their completion time on the allocated quay crane.

**Step 4.2.** Consider the containers one-by-one in the above order. Allocate the container to the truck with the earliest available time, and calculate the completion time of the container on this truck. Continue this process until all the containers have been allocated to trucks.

**Step 5.** For \( l = 1, 2, \ldots, l \), calculate the fitness value, which is equal to the objective value.

**Step 6.** Update the best position of every particle, pbest.

**Step 7.** Update the best position of the swarm, best.

**Step 8.** Update the velocity using formula (44) and update the position using formula (42) of each particle.

**Step 9.** If \( \tau \) reaches the preset maximum iteration, stop; otherwise set \( \tau = \tau + 1 \) and go to step 2.

7. The bidirectional flow problem

7.1. The extended model for the bidirectional flow problem

The model for the unidirectional flow problem is only suitable for the situation where there are only containers to be unloaded from ships. The trucks in this case only carry containers from quay crane to storage yard and they are empty when returning to the quay cranes. In practice, during a planning period, there are both containers to be unloaded from the ships and containers to be loaded onto the ships. The unloaded containers are transported to the yard after unloading, while the loaded containers are transported from the yard to the quay cranes before being loaded onto

---

**Table 1**
The encoding of the solutions.

<table>
<thead>
<tr>
<th>Container No.</th>
<th>( X_i )</th>
<th>( X_{i\omega} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.6</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>2.5</td>
</tr>
<tr>
<td>3</td>
<td>1.9</td>
<td>0.6</td>
</tr>
<tr>
<td>4</td>
<td>0.2</td>
<td>2.1</td>
</tr>
<tr>
<td>5</td>
<td>0.4</td>
<td>3.5</td>
</tr>
</tbody>
</table>

**Table 2**
The decoding of the solutions.

<table>
<thead>
<tr>
<th>Container No.</th>
<th>Quay crane No.</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

**Table 3**
An initial generated solution (1).

<table>
<thead>
<tr>
<th>Container No.</th>
<th>Bay No.</th>
<th>Original ( X_{i\omega} )</th>
<th>Modified ( X_{i\omega} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.6</td>
<td>0.6</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>1.2</td>
<td>1.2</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>1.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

**Table 4**
An initial generated solution (2).

<table>
<thead>
<tr>
<th>Container No.</th>
<th>Bay No.</th>
<th>Original ( X_{i\omega} )</th>
<th>Modified ( X_{i\omega} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2.8</td>
<td>2.8</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0.6</td>
<td>0.6</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>3.2</td>
<td>3.2</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>0.4</td>
<td>0.5</td>
</tr>
</tbody>
</table>
the ships. The problem becomes bidirectional and the trucks may carry containers in both directions reducing the empty trips.

The MILP model needs to be modified for applying to the bidirectional flow case. To present the modified model, we first re-define some parameters. Because each unloaded or loaded container has a fixed original location and a known destination location, so the time for a truck to transport one container from the quayside (yard) to the yard (quayside) is known in advance, and we call the time \( P_{ij}^l \). The time for a truck to serve another container is determined by the destination location of the container it just served and the original location of the next container it is going to transport.

\[
\begin{align*}
\Omega_1 & \text{ The set of containers to be unloaded from the ships} \\
\Omega_2 & \text{ The set of containers to be loaded onto the ships} \\
\Omega & \text{ The set of all containers to be unloaded or loaded from or onto the ships, } \Omega = \Omega_1 \cup \Omega_2 \\
P_{ij}^l & \text{ The time for a truck to transport container } j \text{ from its bay to container } i \\
s_{ij} & \text{ The travel time of a quay crane from bay to container } i \\
t_{ij} & \text{ The transporting time of a truck from the destination location of container } i \text{ to the original location of container } j
\end{align*}
\]

Many constraints of the original model are still valid to the bidirectional flow situation. The modifications to the model are presented below.

\[
\min \ \max_{i,j} C_{ij} \tag{45}
\]

Subject to constraints (3)–(12), (14), (15), (17)–(19), (21), (22), (24)–(26) and

\[
\begin{align*}
C_{ij} &= C_{ij}^l - P_{ij}^l & i \in \Omega_2 \\
C_{ij}^l + P_{ij}^l &= C_{ij}^r & i \in \Omega_1 \\
C_{ij}^l - P_{ij}^l - s_{ij} + M(1 - Z_{qij}) & \geq 0 & j \in \Omega, \ q \in Q \\
C_{ij}^l + M(1 - Z_{qij}) & \geq C_{ij}^l + s_{ij} + P_{ij}^l & i, \ j \in \Omega, \ q \in Q \\
C_{ij}^r - P_{ij}^r - t_{ij} + M(1 - X_{qij}) & \geq 0 & j \in \Omega, \ u \in T \\
C_{ij}^r + M(1 - X_{qij}) & \geq C_{ij}^r + t_{ij} & i \in \Omega, j \in \Omega_1, \ u \in T \\
C_{ij}^r + M(1 - X_{qij}) & \geq C_{ij}^r + t_{ij} + P_{ij}^r & i \in \Omega, j \in \Omega_2, \ u \in T
\end{align*}
\]

The lower bound model is as follows.

\[
\begin{align*}
\text{Minimize } & \ F1 & \tag{53} \\
\text{Subject to constraints (6)–(9), (25)–(26), (32)–(36), (39)–(40) and} & \\
D_h - D_{h'} + S_{hi'} - S_{hi} & \leq M(1 - w_{hi'}) \quad h, \ h' \in E, \ q \in Q \\
D_h - S_h - S_{hi} + M(1 - w_{hi}) & \geq 0 \quad h \in E, \ q \in Q \tag{54}
\end{align*}
\]

7.3. The PSO algorithm for the unidirectional flow problem

For the bidirectional flow problem, two methods are proposed to improve the PSO algorithm. In our PSO algorithm, a disturbance method with the formula (56). This is the second improvement of our PSO algorithm, which is called VPSO.

\[
t^{t+1} = \alpha t^t + c_1 r_1 (x^t_{\text{pbest}_i} - x^t_i) + c_2 r_2 (x^t_{\text{gbest}} - x^t_i) \tag{56}
\]

We call the PSO algorithm with the above two improvements IPSO, and the PSO algorithm with no improvement SPSO.

The complete PSO procedure for solving the bidirectional flow problem is as follows.

\textbf{Step} 1. Initialize \( I \) particles as a swarm to obtain a processing sequence of all the containers, and each particle has \( N \) dimensions, in which \( N \) denotes the number of containers. The encoding of the solutions is the same as \( X \), as demonstrated in Table 1. Set iteration number \( \tau = 1, I = 1 \).

\textbf{Step} 2. Make sure that the containers in the same bay must be served by the same quay crane in the initial solutions (as demonstrated in Table 3). In addition, the precedence between containers must be satisfied (as demonstrated in Table 4). Set \( i = 1 \).

\textbf{Step} 3. If container \( i \) is to be unloaded, go to step 4, else go to step 5.

\textbf{Step} 4. If container \( i \) is the first one in its bay to be unloaded, allocate the quay crane with the earliest available time to container \( i \), else allocate the quay crane that has been allocated to serve this bay to container \( i \). Allocate the truck with the earliest available time to container \( i \). Go to step 6.

\textbf{Step} 5. Allocate the truck with the earliest available time to container \( i \). If container \( i \) is the first one in its bay to be loaded, allocate the quay crane with the earliest available time to container \( i \), else allocate the quay crane that has been allocated to serve this bay to container \( i \). Go to step 6.

\textbf{Step} 6. Calculate the completion time of the container on the allocated quay crane and truck, and make sure there are no conflicts between different quay cranes. Then update the earliest available times of the allocated container and truck.
obtain good solutions within acceptable time.

and the solutions generated by the GA, because CPLEX cannot

are compared with the optimal solutions obtained by CPLEX, the

formance of the model. The results of the proposed PSO algorithm

test the contribution of the proposed valid inequalities to the per-

unidirectional flow problem the bidirectional flow problem,

rithms are coded in C++, and the models are solved using CPLEX.

2.83 gigahertz Intel Core 2 CPU and 3.25 gigabyte RAM. The algo-

computational experiments have been carried out on both unidirectional

problems and bidirectional flow problems. In the PSO algo-

rithm, the maximum number of iterations and the population size

are set as 500 and 30, respectively, for all the experiments. Based

on the results of test runs, we set \( \mu = 3, c_1 = 2, c_2 = 2 \). Similar to

the original PSO algorithm, the value of \( c_3 \) needs to be set by the

user. We tested the PSO with \( c_3 = 0 \) (the same as in the original

PSO algorithm), 0.5, 1, 1.5 and 2 to solve some test problems.

The results show that \( c_3 = 1 \) performs the best. So we choose

\( c_3 = 1 \) for all the experiments. The results of the PSO algorithm

are compared with the lower bounds and for small sized problems

also with the optimal solutions obtained by solving the model. In

addition, we have compared the PSO algorithm with another nature

inspired algorithm as well. To the best of our knowledge, there

has not been any nature inspired algorithm previously used to

solve the problem studied in this paper. We choose the genetic

algorithm (GA) presented by Lee et al. (2008) for comparison. A

chromosome of the GA represents a sequence of containers which
can be encoded in the same way as in the proposed PSO algorithm,

i.e., a feasible solution is generated by scheduling the trucks based

on the container sequence. We also use the adaptive crossover

operator and the adaptive mutation operator in literature (Yan,

2010) to improve the GA. In the GA used in all the computational

experiments, the maximum and minimum probability of cross-

over, the maximum and minimum probability of mutation, and

the population size of the GA are set as 0.3, 0.2, 0.12, 0.08, and

200, respectively. The GA stops when the computation time

reaches the same time spent by the proposed PSO for the problem

instance.

All the experiments are performed on a computer with

2.83 gigahertz Intel Core 2 CPU and 3.25 gigabyte RAM. The algo-
rithms are coded in C++, and the models are solved using CPLEX.

In the two subsections below, we report the experiments for the

unidirectional flow problem the bidirectional flow problem,

respectively.

8.1. Computational experiments for the unidirectional flow problem

In this section, we first give the problem generation, and then
test the contribution of the proposed valid inequalities to the per-
formance of the model. The results of the proposed PSO algorithm
are compared with the optimal solutions obtained by CPLEX, the
proposed lower bounds, and the solutions obtained by the GA for
all the small sized problems. For large sized problems, the results
of the proposed PSO algorithm are compared with lower bounds
and the solutions generated by the GA, because CPLEX cannot
obtain good solutions within acceptable time.

8.1.1. Problem generation

For the experiments on the unidirectional flow problem, we
randomly generated 2 types of problems - nine small sized prob-
lem instances and nine large sized problem instances. We consider
the cases of 10, 15 and 20 containers for small sized problems; and
100, 200 and 300 containers for large sized problems. The problem
data are randomly generated as follows.

1. The time for unloading a container follows the uniform dis-
tribution, ranging from 140 seconds to 190 seconds (including
the picking up time, dropping off time and traveling time of quay
cranes).

2. The traveling speed of the trucks is 5 meters per seconds.

3. In the unidirectional flow problem, the time for a truck to
transport an unloaded container is determined by the dis-
tance from the original location (quayside) of the container

to its destination (blocks in the yard), the distance from the
destination of the container to the quayside, as well as
the speed of the truck. We take a bay number of the ship
as the original location of an unloaded container, and
the destination of each unloaded container is expressed by
a block number of the yard. In the experiments the destina-
tions of all the unloaded containers are generated based on

the layout of a container terminal.

8.1.2. Computational results

The small sized unidirectional flow problems are first solved
using the proposed model, and model with the valid inequalities
(27) to (30) and the proposed PSO algorithm. The objective value
results obtained by these methods and the computation time used
are presented in Table 5. In the table, \( N, c \) and \( t \) are the numbers
of containers, cranes and trucks, respectively. \( M \) denotes the original
model and \( M \) denotes the model with valid inequalities.

From Table 5, we can observe that models with and without
the valid inequalities yield the same solution. However, the valid
inequalities help to reduce computation time, from over 2.5 min-
utes less than 2 minutes on average. The proposed PSO solve
each of the small problems in only about half second on average.

Table 6 shows the two lower bounds calculated for the small
sized unidirectional problems and the relative deviations of the
PSO solution and the lower bounds from the optimal model solu-
tion (Opt). From the results we can see that the proposed PSO algo-
rithm generates solutions very close to optimum with an average
relative deviation of 1.89% from the optimum. Lower bound (LB)
2 is much tighter than LB 1. LB2 is only 2.5% below the optimum
on average while LB1 is 7.17% below optimum. Therefore we will
compare the PSO solution with LB2 in subsequent experiments.

For the small unidirectional problems, the relative deviation of
PSO solution from LB2 is 4.53% on average. This is larger than the

<table>
<thead>
<tr>
<th>Table 5</th>
<th>Results of the model and the proposed PSO for small sized unidirectional flow problems.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td>( N &lt; c \times t )</td>
</tr>
<tr>
<td>( N \times M \times M )</td>
<td>( M \times M \times PSO )</td>
</tr>
<tr>
<td>1</td>
<td>10 \times 2 \times 6 &amp; 1095 &amp; 1095 &amp; 1123</td>
</tr>
<tr>
<td>2</td>
<td>10 \times 2 \times 8 &amp; 1064 &amp; 1064 &amp; 1064</td>
</tr>
<tr>
<td>3</td>
<td>10 \times 3 \times 8 &amp; 892 &amp; 892 &amp; 921</td>
</tr>
<tr>
<td>4</td>
<td>15 \times 2 \times 6 &amp; 1481 &amp; 1481 &amp; 1501</td>
</tr>
<tr>
<td>5</td>
<td>15 \times 2 \times 8 &amp; 1470 &amp; 1470 &amp; 1487</td>
</tr>
<tr>
<td>6</td>
<td>15 \times 3 \times 8 &amp; 1155 &amp; 1155 &amp; 1185</td>
</tr>
<tr>
<td>7</td>
<td>20 \times 2 \times 6 &amp; 1864 &amp; 1864 &amp; 1900</td>
</tr>
<tr>
<td>8</td>
<td>20 \times 2 \times 8 &amp; 1864 &amp; 1864 &amp; 1895</td>
</tr>
<tr>
<td>9</td>
<td>20 \times 3 \times 8 &amp; 1363 &amp; 1363 &amp; 1397</td>
</tr>
<tr>
<td>Average</td>
<td></td>
</tr>
</tbody>
</table>
problems except two. Furthermore, the average relative deviation is better than the GA solution on average and for all the tested large problems.

Comparing the PSO solution with optimal solution and lower bounds for small sized unidirectional flow problems.

The results indicate that the proposed PSO algorithm is effective and efficient to solve this kind of scheduling problems. The results show that the PSO solution could be much closer to the optimum than 5.18%, as demonstrated for case of the small problems in Table 6. Table 7 also shows the computation time of the proposed PSO algorithm. While the model for the large problems cannot be solved in reasonable time, the PSO solve the problems in 74.56 seconds on average. The results indicate that the proposed PSO algorithm is effective and capable time, the PSO solve the problems in 74.56 seconds on average.

8.2. Computational experiments for the bidirectional flow problem

This section reports the experiment results on the bidirectional flow problems. Different versions of the PSO algorithm are tested. They are the standard PSO algorithm (SPSO), the PSO algorithm with disturbance method (DPSPSO), the PSO algorithm with the new velocity updating method (VP-SPO), and the PSO algorithm with both of the above improvements (IPSO). The results of proposed PSO algorithm are compared with the optimal solutions obtained by solving the model section 7 using CPLEX for small sized problems, and with the proposed lower bound for the large sized problems.

### Table 6
Comparing the PSO solution with optimal solution and lower bounds for small sized unidirectional flow problems.

<table>
<thead>
<tr>
<th>n</th>
<th>Size (×××)</th>
<th>Results (seconds)</th>
<th>Relative deviation (%)</th>
<th>CPU time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10 × 2 × 6</td>
<td>1005</td>
<td>1039</td>
<td>2.56</td>
</tr>
<tr>
<td>2</td>
<td>10 × 2 × 8</td>
<td>992</td>
<td>1035</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>10 × 3 × 8</td>
<td>735</td>
<td>868</td>
<td>3.25</td>
</tr>
<tr>
<td>4</td>
<td>15 × 2 × 6</td>
<td>1418</td>
<td>1461</td>
<td>1.35</td>
</tr>
<tr>
<td>5</td>
<td>15 × 2 × 8</td>
<td>1409</td>
<td>1443</td>
<td>1.16</td>
</tr>
<tr>
<td>6</td>
<td>15 × 3 × 8</td>
<td>1026</td>
<td>1115</td>
<td>2.60</td>
</tr>
<tr>
<td>7</td>
<td>20 × 2 × 6</td>
<td>1812</td>
<td>1835</td>
<td>1.93</td>
</tr>
<tr>
<td>8</td>
<td>20 × 2 × 8</td>
<td>1820</td>
<td>1838</td>
<td>1.66</td>
</tr>
<tr>
<td>9</td>
<td>20 × 3 × 8</td>
<td>1264</td>
<td>1330</td>
<td>2.49</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>/</td>
<td>/</td>
<td></td>
</tr>
</tbody>
</table>

### Table 7
Comparison between the PSO solution and LB2 for large sized unidirectional flow problems.

<table>
<thead>
<tr>
<th>n</th>
<th>Size (×××)</th>
<th>Results (seconds)</th>
<th>Relative deviation (%)</th>
<th>CPU time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PPO</td>
<td>LB2</td>
<td>(PSO – LB2)/LB2</td>
</tr>
<tr>
<td>10</td>
<td>100 × 3 × 10</td>
<td>6187</td>
<td>5880</td>
<td>5.22</td>
</tr>
<tr>
<td>11</td>
<td>100 × 3 × 12</td>
<td>6016</td>
<td>5770</td>
<td>4.26</td>
</tr>
<tr>
<td>12</td>
<td>100 × 4 × 16</td>
<td>5018</td>
<td>4549</td>
<td>10.31</td>
</tr>
<tr>
<td>13</td>
<td>200 × 3 × 10</td>
<td>11,927</td>
<td>11,599</td>
<td>2.83</td>
</tr>
<tr>
<td>14</td>
<td>200 × 3 × 12</td>
<td>12,258</td>
<td>11,748</td>
<td>4.34</td>
</tr>
<tr>
<td>15</td>
<td>200 × 4 × 16</td>
<td>9782</td>
<td>9201</td>
<td>6.31</td>
</tr>
<tr>
<td>16</td>
<td>300 × 3 × 10</td>
<td>18,541</td>
<td>17,761</td>
<td>4.39</td>
</tr>
<tr>
<td>17</td>
<td>300 × 3 × 12</td>
<td>18,339</td>
<td>17,553</td>
<td>4.48</td>
</tr>
<tr>
<td>18</td>
<td>300 × 4 × 16</td>
<td>14,512</td>
<td>13,887</td>
<td>4.50</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>/</td>
<td>/</td>
<td></td>
</tr>
</tbody>
</table>

8.2.1. Problem generation

To demonstrate the effectiveness of the proposed PSO algorithm for the bidirectional flow problems, we test the algorithm on 2 types of randomly generated problems – ten small sized problems, and eight large sized problems. We consider the cases of container number from 7 to 11 for small sized problems; and 100, 200, 300 and 500 containers for large sized problems.

In this section the data in the experiments are generated as follows:

1. The time of unloading or loading a container is generated follows the uniform distribution, ranging from 105 seconds to 161 seconds (including the picking up time, dropping off time of quay cranes, but including the traveling time of quay cranes).
2. The traveling speed of the trucks is 5 meters per seconds.
3. For the bidirectional flow problems, the time for a truck to transport an inbound container is determined by the distance from the original location (bays) of the container to its destination (blocks in the yard) and the traveling speed of the truck. In practical operations at container terminals, the original locations of all the inbound containers are informed by the ships before their arrival and the destination locations of these containers are planned in storage allocation planning and so are also known.
4. The time for transporting an outbound container by a truck equals the distance from the original location (blocks in the yard) of the container to its destination location (bays) divided by the traveling speed of the truck. The original locations (block numbers in the yard) of all the outbound
From the results, we observe that the solution by the PSO respectively. The best solution for each problem is shown in bold. Compared to the standard PSO algorithm, the solutions obtained by IPSO are better than those obtained by other algorithms. For large sized problems, five in eight solutions obtained by IPSO are better than those of others for any of the small sized problems. The relative deviation of IPSO solution is only 0.22% from optimum. The computation time for CPLEX to solve the model optimally increases quickly as the problem size increases. The average computation time of CPLEX is about 5 minutes for the small problems while IPSO only takes 0.17 seconds on average.

The average relative deviation of LB from the optimal solution is 1.02% for the small sized problems. The relative deviation of IPSO solution from LB is 1.26% on average for the small problems. For large sized problems, five in eight solutions obtained by IPSO are better than those of others. With almost the same computation time, IPSO generates better solution than the PSO solution. In the subsequent comparisons we use IPSO to represent the proposed PSO algorithm.

Table 9
Comparison between the proposed PSO and the GA for large sized unidirectional flow problems.

<table>
<thead>
<tr>
<th>n</th>
<th>Size</th>
<th>GA</th>
<th>PSO</th>
<th>Relative deviation (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(N \times C \times t)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>100 \times 3 \times 10</td>
<td>6161</td>
<td>6187</td>
<td>0.42</td>
</tr>
<tr>
<td>11</td>
<td>100 \times 3 \times 12</td>
<td>6092</td>
<td>6016</td>
<td>1.26</td>
</tr>
<tr>
<td>12</td>
<td>100 \times 4 \times 16</td>
<td>5122</td>
<td>5018</td>
<td>2.07</td>
</tr>
<tr>
<td>13</td>
<td>200 \times 3 \times 10</td>
<td>11,905</td>
<td>11,927</td>
<td>0.18</td>
</tr>
<tr>
<td>14</td>
<td>200 \times 3 \times 12</td>
<td>12,474</td>
<td>12,258</td>
<td>1.76</td>
</tr>
<tr>
<td>15</td>
<td>200 \times 4 \times 16</td>
<td>10,131</td>
<td>9782</td>
<td>3.57</td>
</tr>
<tr>
<td>16</td>
<td>300 \times 3 \times 10</td>
<td>18,766</td>
<td>18,541</td>
<td>1.21</td>
</tr>
<tr>
<td>17</td>
<td>300 \times 3 \times 12</td>
<td>18,427</td>
<td>18,339</td>
<td>0.48</td>
</tr>
<tr>
<td>18</td>
<td>300 \times 4 \times 16</td>
<td>14,057</td>
<td>14,512</td>
<td>3.07</td>
</tr>
<tr>
<td>Average</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>1.42</td>
</tr>
</tbody>
</table>

Containers are known. The destination locations of the outbound containers are also known from the stowage plan of the ships.

In our experiment, the original (destination) locations of the inbound (outbound) containers are generated randomly considering the configuration of the ships, and the destinations (original) locations of the inbound (outbound) containers are generated based on the practical layout of a container terminal.

8.2.2. Computational results

In order to understand the performance of the improvement strategies in the proposed PSO algorithm on the bidirectional flow problem, we first compare the standard PSO algorithm and the strategies in the proposed PSO algorithm on the bidirectional flow problem. The results for small and large sized problems are reported in Table 10 and Table 11, respectively. The best solution for each problem is shown in bold. From the results, we observe that the solution by the PSO algorithm with both improvement strategies (IPSO) is not worse than those of others for any of the small sized problems. For large sized problems, five in eight solutions obtained by IPSO are better than those by others. With almost the same computation time, IPSO generates better solution than the PSO solution. In the subsequent comparisons we use IPSO to represent the proposed PSO algorithm.

The IPSO solution is compared with optimal solution (Opt) and lower bound (LB) for small sized problems in Table 12. From the results, we can observe that the average relative deviation of the IPSO solution is only 0.22% from optimum. The computation time for CPLEX to solve the model optimally increases quickly as the problem size increases. The average computation time of CPLEX is about 5 minutes for the small problems while IPSO only takes 0.17 seconds on average.

The average relative deviation of LB from the optimal solution is 1.02% for the small sized problems. The relative deviation of IPSO solution from LB is 1.26% on average for the small problems. For

Table 10
Comparison of different PSO algorithms for small sized bidirectional flow problems.

<table>
<thead>
<tr>
<th>n</th>
<th>Size</th>
<th>Objective values</th>
<th>Computation time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(N \times C \times t)</td>
<td>SPSO</td>
<td>DPSO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1230.7</td>
<td>1230.7</td>
</tr>
<tr>
<td>19</td>
<td>7 \times 1 \times 3</td>
<td>1230.7</td>
<td>1230.7</td>
</tr>
<tr>
<td>20</td>
<td>7 \times 2 \times 4</td>
<td>680.9</td>
<td>677.3</td>
</tr>
<tr>
<td>21</td>
<td>8 \times 1 \times 3</td>
<td>1337.3</td>
<td>1333.3</td>
</tr>
<tr>
<td>22</td>
<td>8 \times 2 \times 4</td>
<td>717.3</td>
<td>713.6</td>
</tr>
<tr>
<td>23</td>
<td>9 \times 1 \times 3</td>
<td>1520.7</td>
<td>1520.7</td>
</tr>
<tr>
<td>24</td>
<td>9 \times 2 \times 4</td>
<td>814.6</td>
<td>810.9</td>
</tr>
<tr>
<td>25</td>
<td>10 \times 1 \times 3</td>
<td>1662.4</td>
<td>1661.6</td>
</tr>
<tr>
<td>26</td>
<td>10 \times 2 \times 4</td>
<td>880</td>
<td>853</td>
</tr>
<tr>
<td>27</td>
<td>11 \times 1 \times 3</td>
<td>1807.5</td>
<td>1780</td>
</tr>
<tr>
<td>28</td>
<td>11 \times 2 \times 4</td>
<td>954.4</td>
<td>951.6</td>
</tr>
<tr>
<td>Average</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
</tbody>
</table>

Table 11
Comparison of different PSO algorithms for large sized bidirectional flow problems.

<table>
<thead>
<tr>
<th>n</th>
<th>Size</th>
<th>Objective values</th>
<th>Computation time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(m \times C \times t)</td>
<td>SPSO</td>
<td>DPSO</td>
</tr>
<tr>
<td>29</td>
<td>100 \times 2 \times 10</td>
<td>6917.4</td>
<td>6856</td>
</tr>
<tr>
<td>30</td>
<td>100 \times 4 \times 14</td>
<td>4010.6</td>
<td>3982.6</td>
</tr>
<tr>
<td>31</td>
<td>200 \times 2 \times 10</td>
<td>13793.1</td>
<td>13777.9</td>
</tr>
<tr>
<td>32</td>
<td>200 \times 4 \times 14</td>
<td>7770.8</td>
<td>8020.6</td>
</tr>
<tr>
<td>33</td>
<td>300 \times 2 \times 10</td>
<td>20425.6</td>
<td>20349.8</td>
</tr>
<tr>
<td>34</td>
<td>300 \times 4 \times 14</td>
<td>11800</td>
<td>12186.4</td>
</tr>
<tr>
<td>35</td>
<td>500 \times 2 \times 10</td>
<td>33682.1</td>
<td>33750</td>
</tr>
<tr>
<td>36</td>
<td>500 \times 4 \times 14</td>
<td>20264.2</td>
<td>19961.8</td>
</tr>
<tr>
<td>Average</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
</tbody>
</table>

large sized problems, the average relative deviation of IPSO solution from LB is 2.48% as shown in Table 13. This demonstrates that our IPSO algorithm is effective for the joint quay crane and truck scheduling problem with bidirectional flow.

Tables 14 and 15 compare the IPSO and the GA for bidirectional problems with small and large sizes, respectively. IPSO shows better performance than the GA for all of the problems. The average relative differences between the results of IPSO and GA are 2.26% and 6.05% respectively for small and large sized problems.

In the unidirectional flow problem, the truck traveling time for each container is set as the time for a truck to transport the container from the quay crane to the container’s destination in the yard and to return to the quay crane. In cases where each truck serves a dedicated quay crane, the traveling time is accurate. Otherwise, if a truck can go to another quay crane after transporting a container to the yard, such a traveling time is only an estimation. In this case the model and algorithm for the bidirectional flow problem can be applied to solve the unidirectional problem, as the unidirectional problem is a special case of the bidirectional problem and different parts of the traveling time in the bidirectional flow model are more precisely calculated in detail.

9. Conclusions

In this paper, a mixed integer programming model was first formulated for the joint quay crane and truck scheduling problem with only inbound containers. A PSO algorithm with a modified update strategy was proposed for this unidirectional flow problem. The results of the proposed PSO algorithm were compared with the optimal solutions and two lower bounds. The model was then extended to the general situation with both inbound and outbound containers. For this bidirectional flow problem, a new velocity estimation strategy was proposed for this unidirectional flow problem.

Acknowledgements

This research is partly supported by State Key Program of National Natural Science Foundation of China (Grant No. 71032004) and the Foundation for Innovative Research Groups of the National Natural Science Foundation of China (Grant No. 71321001).

References


