Optimisation for job scheduling at automated container terminals using genetic algorithm

Bradley Skinner, Shuai Yuan, Shoudong Huang, Dikai Liu, Binghuang Cai, Gamini Dissanayake, Haye Lau, Andrew Bott, Daniel Pagac

School of Electrical, Mechanical and Mechatronic Systems, Faculty of Engineering and Information Technology (FEIT), University of Technology, PO Box 123, Sydney, NSW 2007, Australia

Patrick Technology Systems, 4b Lord Street, Botany, NSW 2019, Australia

A R T I C L E   I N F O

Article history:
Received 20 August 2011
Received in revised form 13 June 2012
Accepted 28 August 2012
Available online xxx

Keywords:
Modelling
Scheduling
Genetic algorithms
Autonomous straddle carrier
Automated seaport container terminals

A B S T R A C T

This paper presents a genetic algorithm (GA)-based optimisation approach to improve container handling operations at the Patrick AutoStrad container terminal located in Brisbane Australia. In this paper we focus on scheduling for container transfers and encode the problem using a two-part chromosome approach which is then solved using a modified genetic algorithm. In simulation experiments, the performance of the GA-based approach and a sequential job scheduling method are evaluated and compared with different scheduling scenarios. The experimental results show that the GA-based approach can find better solutions which improve the overall performance. The GA-based approach has been implemented in the terminal scheduling system and the live testing results show that the GA-based approach can reduce the overall time-related cost of container transfers at the automated container terminal.

1. Introduction

Both the capacity and frequency of container carrying ships arriving at seaport container terminals have increased steadily during the past several decades (Günther & Kim, 2006; Steenken, Voß, & Stahlbock, 2004). The efficiency of container transfers must be as high as possible so as to reduce costs to terminal operators and increase the productivity of the terminal. This requires efficient use of yard vehicles to load, unload and transfer containers during the transportation process.

Automating container terminals to some extent is a trend in container terminal operations (Kim, Won, Lim, & Takahashi, 2004; Liu, Jula, Vukadinovic, & Ioannou, 2004). Increasing automation of yard vehicles not only reduces the labour costs of terminal operators, but also has the potential to increase the efficiency of container transport. Nevertheless, as compared with human operated yard vehicles there is an ongoing requirement to ensure a high degree of coordination and efficiency for all material handling equipment participating in the transportation process. As a result of this, automated yard vehicles for transporting containers within the ports. The Straddle Carrier (SC) is one of the most common types of yard vehicles for such container transfer. They can pick-up/set-down containers autonomously without a human operator in the yard area. At the Patrick AutoStrad container terminal (Fig. 1), a fleet of fully autonomous SCs are used for container transport (Nelmes, 2005) (Fig. 2). More than twenty autonomous SCs are issued high level commands from a central computer system located at the port. The navigation system allows the SCs to travel along any planned trajectory while performing collision detection. These SCs are the only vehicles to transfer containers among QC area, yard area and the truck (TK) area which makes this terminal unique when compared with current seaports that use human operated SCs for container transporting.

At the Patrick AutoStrad container terminal, like many other seaport terminals, one of the main tasks of a fleet of SCs is to serve the QCs such that the maximal QC turn-around rate can be achieved (Nelmes, 2005; Yuan et al., 2009). This will reduce the ship berthing time and increase the terminal productivity. Moreover, the SCs also need to serve the trucks which are used
to transfer containers between customers and the seaport terminal such that the truck waiting time is minimised. Furthermore, the seaport operator would like SCs to perform less urgent yard-to-yard container transportation jobs (e.g., yard maintenance) as part of their yard management strategy.

We presented a mathematical model of the integrated SC path planning and task allocation in yard operations (yard jobs) in (Liu & Kulatunga, 2007; Yuan et al., 2010). This mathematical formulation was extended in (Yuan et al., 2011) by including QC and TK related jobs and then solved by a proposed job grouping strategy (Yuan et al., 2011). The job grouping approach groups jobs based on time-critical requirements, so as to enhance the time related performance. One advantage of the job grouping approach is that it integrates both the scheduling and path planning algorithms for all jobs and SCs. However, this grouping approach has a short planning horizon, which may end up with a local optimum solution. The employed path planning algorithm (Halpern, 1977; Lau, Pratley, Liu, Huang, & Pagac, 2008) has been well-studied and can effectively deal with routing and collision avoidance in actual terminals. Solving the scheduling problem with longer planning horizon using a global optimisation approach is expected to give better solution (higher productivity) in automated container terminals such as the Patrick AutoStrad Terminal.

This paper presents a genetic algorithm (GA)-based optimisation approach to solve the SC scheduling problem for container handling in the Patrick AutoStrad Terminal. In this paper, the problem of scheduling SCs has subtle but significant differences to the multiple travelling salesman problem (MTSP) (Bektas, 2006) and Pick-up and Delivery Problem (PDP) (Savelsbergh & Sol, 1995). Firstly, both MTSP and PDP do not have sequence constraints on visiting cities or nodes and salesmen or vehicles are normally allowed to conduct jobs in any order. In our problem the job sequence has to be taken into account particularly for QC and TK related jobs. Secondly, MTSP and PDP do not have to deal with constraints such as servicing QCs and TKs, which tends to add complex timing dependence for related jobs. Due to the timing dependence of all resources (i.e. SCs, QCs and TKs), it is not suitable to use time-window based approaches (e.g. MTSP with time-window and PDP with time-window) to deal with the correlated job sequence and timing constraints. Thirdly, our performance metrics are based on practical operations at the Patrick AutoStrad Terminal which are different from the general MTSP and PDP.

Objectives in our model are not only related to SCs (e.g.: minimising total travel time which is a linear function), but also include the performances of QCs and TKs (e.g.: minimising QC waiting time and TK waiting time which are non-linear functions). Previous approaches (Bektas, 2006; Dumas, Desrosiers, & Soumis, 1991; Li & Lim, 2001; Park, 2001; Ruland & Rodin, 1997; Savelsbergh & Sol, 1995) to solving the MTSP and PDP are often designed for linear objective functions with linear constraints and thus cannot be directly used to with our model and solve the practical problem at Patrick AutoStrad Terminal. In this paper, the comprehensive problem model presented in (Yuan et al., 2011) is modified as a job scheduling problem which deals with correlated job sequence and timings. Five practical performance metrics are formulated which are directly related to the operations at the Patrick container terminal. A modified genetic algorithm (GA) is presented to handle the job sequence and timing constraints and improve the scheduling performance of container transfers at the automated container terminal.

The paper is organised as follows. Section 2 provides a brief overview of related work in scheduling problems for the transporation process within automated seaport container terminals. Section 3 presents the modified mathematical model of the job scheduling. In Section 4, we discuss the GA-based approach. Section 5 presents the experiments and the compared performance. Finally, Section 6 provides a conclusion and summary of the study.

2. Literature review

2.1. Container handling in transhipment

Within a seaport environment an effective schedule for the transhipment of containers requires efficient allocation and scheduling of quay-side, yard and land-side resources. Although the operation of these resources is typically decoupled from each other the efficiency of the overall transhipment process is significantly affected by their operation, particularly if there is a high level of automation involved.

A quay crane scheduling problem (QCSP) considers the starting and finishing time of each job (e.g., loading and unloading a container) in a set of jobs assigned to quay cranes servicing a vessel. Kim and Park (2004) proposed an analytical formulation of the QCSP. Using a meta-heuristic search algorithm called GRASP, near optimal solutions were found. In addition, a lower bound on the optimal performance was proposed using the branch and bound method. However, these studies assume complete availability of yard resources when transporting containers from the yard to the quay crane loading area, i.e., the scheduling of yard vehicles are not taken into account.

A yard vehicle scheduling problem considers the starting and finishing time of each yard job (e.g., moving a container from its current position to a different position in the yard or to a QC or a truck) in a set of yard jobs assigned to a fleet of autonomous straddle carriers servicing quay cranes and trucks. The problem is to minimise the utilisation of straddle carriers in the transhipment process while finding a feasible and efficient schedule. This
requirement aims to reduce costs associated with performing yard-side operations while solving the allocation and scheduling subproblem for yard resources, namely a fleet of fully autonomous straddle carriers. In a recent study, Kim and Nguyen (2009) discussed a method for dispatching a small fleet of automated guided vehicles (AGVs) for efficient operations of quay cranes and automated yard cranes (AYCs). The problem was introduced as a scheduling problem with both precedence and buffer constraints. These constraints arise due to the variable buffer space allocated in the apron and yard upon which QCs, AYCs, and ALVs can all release containers. A mixed-integer linear programming (MILP) model and heuristic algorithm were proposed for the ALV scheduling problem with the aim of minimising both the total travel time of ALVs and total delay in QC operations. The heuristic algorithm was used to find solutions to this NP-hard scheduling problem with a procedure applied to convert buffer constraints into time-window constraints. The effect of dual cycling at buffers of varying sizes was also analysed, which suggested that as the proportion of dual cycling operations increased the ALV total travel time, QC total delay time and vessel completion (berthing or handling) time decreased. However, the authors suggested the performance of the algorithms proposed in their study need to be assessed in a simulated and dynamic environment. This study only addressed the dispatching of ALVs and not the complete scheduling problem, which also encompasses ALV planning, automated yard cranes, QC operations and TK operations.

As one of the enabling technologies, scheduling of automated guided vehicles (AGVs) has attracted considerable attention and many AGV scheduling algorithms have been proposed. Recently, more container terminals have started utilising automated transporters like AGVs. Therefore the research on the scheduling of AGVs has become more important. Grunow, Hans-otto, and Lehmann (2004) proposed a novel heuristic dispatching algorithm for a fleet of multi-load AGVs. The analysis performed by the authors is based on the distinction of different degrees of AGV availability and a definition of assignment patterns which promoted alternative AGV tours to be generated with very limited computational effort. Furthermore, the problem is also formulated as a MILP model. The performance of the MILP model is compared to the heuristic dispatching algorithm for different AGV fleet sizes. Due to the port layout this study only investigated the assignment problem for a small fleet of AGVs (maximum of six vehicles). The assignment problem is solved while minimising the total lateness of the AGVs only. Additional objectives related to the quay crane or land-side operations were not considered. In a similar study, Kim and Bae (2004) examined AGV dispatching methods by utilising information about locations and times of future delivery tasks. A MILP model is provided for assigning optimal delivery tasks to a small fleet of AGVs. A heuristic algorithm is suggested for overcoming the excessive computational time needed for solving the MILP model. In their study, the dispatching problem is reduced to an assignment problem by defining the exact pickup and delivery times of each container for a single quay crane.

On land-side operations containers are moved to/from the yard stack and to/from other modes of transportation like road and rail as suggested by Vis and Koster (2006). The Patrick AutoStrad container terminal has been designed to support road transportation in its land-side operation. Similar to berthed ships, trucks can either be importing or exporting containers between the yard and the Truck Import Area (TIA). Transporting containers between the yard and the TIA requires the use of autonomous straddle carriers. Minimising the turnaround time of trucks at the TIA can provide greater throughput for the overall process. Since truck movements are booked using the Vehicle Booking System (VBS) and free TIA bays are common at the Patrick AutoStrad container terminal, the allocation of trucks to TIA bays is greatly simplified as a first-come-first-served (FCFS) basis. Ballis and Abacoumkin (1996) presented a simulation model for the port of Piraeus, Greece which permitted performance evaluation of land-side operations using trucks. Here, the trucks are serviced by a fleet of straddle carriers. By incorporating expert knowledge from operators into the model the simulator assesses different configuration of five terminal characteristics. The resulting simulations suggest that the ‘observed strategy leads to shorter truck service time but increase the traffic conflicts in the terminal’s internal transport networks’.

2.2. Related solution techniques

There are many research papers related to the modelling of different operational aspects for seaport container terminals (Stahlbock & Voß, 2008). In a recent study, Kozan and Preston (2007) examined mathematical modelling and optimisation of the sea interfaces using GA and tabu search, specifically the transfer of export containers from the storage to berthed ships and presented an optimisation algorithm capable of handling large problems that arise at the quay-side operation. This study deals exclusively with import and export containers between the ship and the yard and does not include yard-to-yard and yard-truck transportation or other practicalities.

Hartmann (2004) proposed a general model for various scheduling problems that occur in container terminal logistics. The scheduling model consists of the assignment of jobs to vehicles and the temporal arrangement of jobs subject to precedence constraints and sequence-dependent setup times. The model was applied to solve problems for SCs, AGVs, stacking cranes, and workers who handle reefer containers in the port of Hamburg. Solutions were obtained using a vehicle constrained GA.

An attempt at performing comprehensive scheduling of different resources at a seaport container terminal was performed by Meersmans and Wagelmans (2001). They used static and a dynamic variation of the beam search algorithm to model the problem. A form of rescheduling, variable job horizons and different vehicle dispatching algorithms were also investigated in this study. In summary, longer planning horizons (50 containers per quay crane) provided better performance on average. The static beam search algorithm performed better than the dynamic variation of algorithm and better than a number of typical dispatching rules while avoiding deadlock situations.

Lee, Chew, Tan, and Wang (2010) addressed a dispatching problem for vehicles in a transshipment hub by considering the capacity of quay cranes and yard cranes, while minimising the makespan time at the quay-side by using the neighbourhood search and GA with minimum cost flow (MCF) network model. The experimental results showed the superiority of the GA–MCF-based algorithm over the neighbourhood search algorithm. In the work by Zacharia and Aspragathos (2005), the scheduling problem was extended from the TSP by considering the multiple solutions of the inverse kinematics problem. Then a GA-based method was introduced to determine the optimum sequence of task scheduling and an innovative encoding method was introduced by taking into account the multiple solutions of the inverse kinematic problem.

In this paper, we consider five performance metrics which are currently used at the Patrick AutoStrad Terminal: quay crane waiting cost, truck waiting cost, SC travel cost, SC waiting cost and high priority job finishing time. We present a GA-based optimisation approach to solve the container transfer problem with those multiple objectives combined in a linear function.

3. Problem formulation

We presented a mathematical model in (Yuan et al., 2010) which deals with both job scheduling and SC path planning with
collision avoidance. In that model, many practical challenges are taken into account including the presence of multiple levels of container stacking and sequencing, variable container orientations and vehicular dynamics that require finite acceleration and deceleration times. Our mathematical model was extended in (Yuan et al., 2011) by considering quay crane and truck related jobs which makes the model a more comprehensive representation of port operations. However, in this paper, we only focus on the scheduling optimisation for container transfers due to the complexity of solving the integrated problem.

3.1. Overview of the problem

For convenience a map depicting the yard environment has been developed to model the actual Patrick AutoStrad container terminal. Fig. 3 illustrates a schematic diagram of the static seaport environment. In order to reflect the operational environment the map is represented by a graph that contains 18,380 positional nodes and 332,620 predefined links (Yuan et al., 2010), through which any SC can travel. All links are bi-directional and connect two neighbouring nodes. This graph representation of the map is currently implemented at the Patrick terminal for trajectory planning. Using a graph to represent the seaport map allows for the accurate determination of SC position and trajectory information at any time. However, as we focus on the scheduling problem we assume that there are no collisions between SCs and travel times are computed using the Dijkstra’s algorithm (Dijkstra, 1959).

The map of the Patrick terminal contains three key functional areas namely the buffer area, yard area and truck area as shown in Fig. 3. Firstly, the buffer area is located on the quay-side and is used for the temporary storage of containers which are eventually transported between SCs and berthed ships. Secondly, the yard area, including the reefer area and container temporary storage areas (MXA and MXB) is the largest area of the terminal due to the accumulation of containers stored in this area. As a result, the yard area is relatively confined for SCs transportation. Lastly, the truck area is used for the import and export of containers between trucks and the yard area.

Typically, containers discharged from a berthed ship would become truck loading jobs after being temporarily stored in the yard. Similarly, containers unloaded from trucks would most likely become ship loading jobs after being temporarily stored in the yard. Consequently, different job types are not isolated from each other, but loosely coupled through the implicit use of yard vehicles.

A list of container transfers jobs need to be performed by a fleet of identical SCs. For each job the Yard Management System (YMS) provides both an initial node and a destination node and also specifies the container stack level and container alignment for both pick-up location and set-down location. In this paper, we divide the container transferring jobs into five different categories based on their initial and destination nodes:

- Buffer-to-Yard (B2Y): transport a container from a buffer node to a yard node for QC discharging.
- Yard-to-Buffer (Y2B): transport a container from a yard node to a buffer node for QC uploading.
- Truck-to-Yard (T2Y): transport a container from a truck area node to a yard node TK exporting.
- Yard-to-Truck (Y2T): transport a container from a yard node to a truck area node for TK importing.
- Yard-to-Yard (Y2Y): transport a container from a yard node to another yard node for yard management.

![Fig. 3. Schematic diagram of the static seaport environment showing berth, quay cranes, bays, special nodes (buffers and truck gates), truck import area and nodes in the yard (Yuan et al., 2010).](http://dx.doi.org/10.1016/j.cie.2012.08.012)
For each node in the yard area there is a two-level stack where a container can be stored. That is, each yard node can be occupied by two containers vertically. This adds significant complexity to the problem as both set-down and pick-up sequencing must be considered. Each job is described using an initial node and a destination node. T2Y jobs are the basic house-keeping tasks which usually have the lowest priority and they have no critical timing requirements. However, B2Y and Y2B jobs are QC related jobs which must be performed before a predefined time to negate delays in the QC operations. Moreover, T2Y and Y2T jobs are TK related jobs which differ from T2Y jobs in that the initial nodes for T2Y jobs and destination nodes for T2Y jobs require job sequencing operation. Additionally, QC related jobs and TK related jobs share the same attribute as quay-side areas are associated with transferring containers between ships and yard, while truck areas are related to transferring containers between trucks and yard. Hence, QC related jobs and TK related jobs can be treated in a similar manner for scheduling. In our current problem formulation it is assumed that each job must be performed by one and only one SC. That is, when a SC picks up a container as part of an assigned job the container must be transported to the destination by the same SC.

3.2. Definitions of model parameters and variables

This section further describes the model parameters and variables. Considering container transfers by SCs we can establish the following definitions.

**Resource acronyms:**

There are three different terminal resources discussed in this model:

- SC: Straddle Carrier.
- QC: Quay Crane.
- TK: Truck.

**Indices:**

- v: index of the SC.
- q: index of the QC.
- g: index of the TK.
- j: index of the job.
- s: index of the schedule. Each schedule contains a list of jobs which need to be finished in order, and each schedule can be assigned to any SC, but one schedule cannot be shared by multiple SCs.

**Sets:**

- V: set of all SCs, |V| is the total number of SCs in the fleet.
- Q: set of all QCs, |Q| is the total number of QCs.
- G: set of all TKs, |G| is the total number of TKs.
- J: set of all jobs, |J| is the total number of jobs.
- J^H: set of jobs with high priority for scheduling and these jobs usually are time critical or considered as urgent tasks and J^H ⊂ J.
- S: set of all schedules.

**Vectors:**

- J_s: vector of jobs in schedule s. That is, a list of jobs that will be done by a SC in the defined order.
- J_q: vector of jobs associated with discharging QC q.
- J_g: vector of jobs associated with uploading QC g.
- J_t: vector of jobs associated with exporting TK g.
- J_k: vector of jobs associated with importing TK g.

**Parameters:**

- m_q: the total number of discharging containers for QC q.
- n_q: the total number of uploading containers for QC q.
- c_q: the total number of containers for exporting TK g.
- r_q: the total number of containers for importing TK g.
- \( t_p \): the initial position of SC v.
- \( a_{vb} \): parameter for defining pick-up job sequence. \( a_{vb} \) ∈ (0, 1]. Let \( a_{vb} = 1 \) if and only if job a need to be picked up before job b picked up, else \( a_{vb} = 0 \).
- \( b_{vb} \): parameter for defining set-down job sequence. \( b_{vb} \) ∈ (0, 1]. Let \( b_{vb} = 1 \) if and only if job a need to be set-down before job b set-down, else \( b_{vb} = 0 \).
- \( u_j \): the pick-up position of job j. The position is generated by yard management system.
- \( d_j \): the set-down position of job j. The position is generated by yard management system.
- \( y_j \): a binary parameter. \( y_j = 1 \) if and only if the candidate schedule s has job j, otherwise \( y_j = 0 \).
- \( t_0 \): plan starting time.
- \( \Delta t^{SC} \): the turnaround time of each QC unloading/uploading a single container.
- \( \Delta t^{TK} \): the turnaround time of a SC performing TK importing/exporting for a single container within the truck area.
- \( t_D^{q} \): the starting time of QC q operations (discharging/uploading) which is predefined a priori as part of the QC schedule. For each discharging QC we assume that no container from QC q can be picked up by a SC at the buffer before the time \( t_D^{q} + \Delta t^{SC} \).
- \( t_E^{q} \): the starting time of QC q operations (uploading) which is predefined a priori as part of the QC schedule. For each uploading QC we assume that no container can be set-down at buffer by a SC before the time \( t_E^{q} + \Delta t^{SC} \).
- \( t_I^{g} \): the starting time of TK g operations (exporting) which is predefined a priori as part of the TK schedule. For each exporting TK we assume that no container from TK g can be set-down by a SC at the TK gate before the time \( t_I^{g} + \Delta t^{TK} \).
- \( t_F^{g} \): the starting time of TK g operations (importing) which is predefined a priori as part of the TK schedule. For each importing TK we assume that no container from TK g can be picked up by a SC at the TK gate before the time \( t_F^{g} + \Delta t^{TK} \).
- \( w_o \): the minimum theoretical travel time between position o and position z. A look-up table based on Dijkstra’s algorithm (Dijkstra, 1959) is used for all positional nodes.

**Decision variable:**

- \( X_s \): binary decision variable. \( X_s = 1 \) if and only if the candidate schedule s is selected for SC v, otherwise \( X_s = 0 \).

**Dependent variables**

The following time related values are dependent on the decision variable \( X_s \) and a set of constraints will be defined in Section 3.3 to ensure their feasibility.

- \( t_{j}^{d} \): planned start time of job j.
- \( t_{j}^{f} \): planned finish time of job j.
- \( t_{j}^{x} \): planned time to pick-up the final container for discharging QC q.
- \( t_{j}^{y} \): planned time to set-down the final container for uploading QC q.
- \( t_{j}^{z} \): planned time to pick-up the final container for exporting TK g.
- \( t_{j}^{w} \): planned time to set-down the final container for importing TK g.
3.3. Problem formulation

This section describes the overall objective function and the various constraints that need to be enforced as part of the model.

\[
\text{Minimise} \sum_{i \in V} \sum_{s \in S} \left( x_{i,s} \right) \text{TravelTime} + \sum_{i \in V} x_{i,s} \text{SC\_waiting} + \lambda_s x_{i,s} \text{QC\_waiting}
\]

subject to:

\[
\sum_{i \in V} x_{i,s} = 1, \quad \forall v \in V
\]

\[
x_{i,s} \in \{0,1\}, \quad \forall v \in V, \quad \forall s \in S
\]

\[
T_q < T_f, \quad \forall j \in J
\]

\[
\alpha_{ab} = 1, \quad \forall (a,b) \in J - T_q < T_s
\]

\[
\beta_{ab} = 1, \quad \forall (a,b) \in J - T_a < T_b
\]

\[
y_{s,j} x_{j,s} = 1, \quad \forall v \in V, \forall s \in S - T_j > W_{s,a} + t_0
\]

\[
T_j - W_{s,a} > 0, \quad \forall j \in J
\]

\[
y_{s,j} x_{j,s} = 1, \quad \forall v \in V, \forall s \in S - T_j > W_{s,a} + t_0
\]

\[
W_{s,a} = t_0 + T_v < T_s, \quad \forall (a,b) \in J - T_q < T_s
\]

\[
T_q - T_a > \Delta T Q, \quad T_a - T_s > \Delta T C, \quad \forall (a,b) \in J - T_q < T_s
\]

\[
T_q - T_s > \Delta T Q, \quad T_s - T_a > \Delta T C, \quad \forall (a,b) \in J - T_q < T_s
\]

\[
T_q - T_s > \Delta T Q, \quad T_s - T_a > \Delta T C, \quad \forall (a,b) \in J - T_q < T_s
\]

\[
T_q - T_s > \Delta T Q, \quad T_s - T_a > \Delta T C, \quad \forall (a,b) \in J - T_q < T_s
\]

where \(x_{i,s}, \lambda_s, \alpha_{ab}, \beta_{ab}\) are predefined parameters used to normalise the contributions from each function. These parameters provide a means to allocate relative importance to each of the individual costs. Typically, \(\lambda_s, \alpha_{ab}, \beta_{ab}\) will have larger values since QC and TK waiting are of high importance and their contribution in the overall objective function must be amplified.

Eq. (2) ensures that each job \(j\) is included in one and only one selected schedule \(s\) with the assigned SC \(v\) while Eqs. (3) and (4) ensure that each SC \(v\) is assigned to only one selected schedule \(s\). Timing constraints in Eqs. (5)–(11) ensure feasibility in all selected schedules should be taken into account as well. Eq. (5) ensures that the planned start time must precede the corresponding job finish time for each scheduled job. Pick-up and set-down job sequencing is required for all QC and TK related jobs and some Y2Y jobs that require multi-tiered stacking. Pick-up sequencing requirements are expressed by Eq. (6). Here, \(\alpha_{ab}\) is a predefined parameter that indicates if two jobs \((a, b)\) have a pick-up sequencing requirement. Similarly, set-down sequencing operations are expressed in Eq. (7). Here, \(\beta_{ab}\) indicates if two jobs \((a, b)\) have a set-down sequencing requirement. Since the number of buffer nodes for each associated QC is two, for any two B2Y or Y2B jobs, \(\alpha_{ab} = 1\) or \(\beta_{ab} = 1\) indicates the two jobs \((a, b)\) are dependent on timings and must be picked up or set-down at the same buffer node. On the other hand, waiting times at QCs and TKs is caused by late pick-up or set-down of a container by an SC. While waiting times for SCs occur when an SC arrives too early and the corresponding QC or TK, is not ready to process the container. Hence \(\alpha_{ab}\) and \(\beta_{ab}\) are used as the indicator of timing sequences when calculating the waiting time of QCs, TKs and SCs. Eq. (8) ensures that job starting times would not be less than the travel time from the initial position of assigned SC \(v\) to the pick-up position of job \(j\). Eq. (9) ensures that the difference between planned starting and finishing time for a job is not less than the theoretical travel time from the job pick-up to set-down positions. Eq. (10) ensures that the time difference between starting and finishing a job is not less than travel time from the job pick-up to set-down position. Eq. (11) ensures that a SC can only transfer one container at the same time. Eqs. (12)–(15) ensure that any two jobs associated with a same QC or TK should take into account the related turnaround time of QC or TK and any QC or TK job should not start or finish earlier than the related starting time of QC or TK.

The cost of travel time \(c_{\text{TravelTime}}\) is used to reflect the utilisation of SCs for a schedule and the cost is calculated based on Dijkstra’s algorithm.

\[
c_{\text{TravelTime}} = W_{p,u} + W_{d,u} + W_{d,v} + W_{d,\text{q}}
\]

where \(p_u\) is the initial position of the SC \(v\) which is assigned with the schedule \(s\). \(f\) is the final job in the selected schedule \(s\).

Vehicle waiting time \(c_{\text{SC\_waiting}}\) is an important performance metric which represents the efficiency of vehicle usage. In general, less waiting time indicates a better SC usage and less fragmentation of the schedule. The calculation is based on the difference between planned timings (starting and finishing) and the theoretical shortest travel time.

\[
c_{\text{SC\_waiting}} = (T_{1e} - W_{p,u} + T_{1e} - W_{d,u})
\]

\[
+ (T_{2e} - W_{d,u} + T_{2e} - W_{d,v} + W_{d,u} + W_{d,v})
\]

Quay crane waiting time \(c_{\text{QC\_waiting}}\) is a critical cost at container terminals and the following equation minimises the total waiting time. The QC waiting time can be calculated based on the last job of the QC. It is assumed that all QCs have a constant turnaround time for each job.

\[
c_{\text{QC\_waiting}} = \sum_{j \in J} y_{j} \left( T_{q} - T_{f} - m_{q} \times \Delta T Q \right)
\]

\[
+ \sum_{j \in J} y_{j} \left( T_{q} - T_{f} - n_{q} \times \Delta T Q \right)
\]

Similarly, truck waiting time \(c_{\text{TK\_waiting}}\) is an important consideration at container terminals and the following equation minimises the total waiting time. The TK waiting time can be calculated based on the last job in the TK gate. It is assumed that a constant turnaround time is applied for each job at TK gate.

\[
c_{\text{TK\_waiting}} = \sum_{j \in J} y_{j} \left( T_{s} - T_{f} - e_{q} \times \Delta T Q \right)
\]

\[
+ \sum_{j \in J} y_{j} \left( T_{s} - T_{f} - r_{q} \times \Delta T Q \right)
\]

Occasionally there is a requirement for a particular job to be finished as early as possible. As such, it is regarded as a high priority job. The finishing time \(c_{\text{HP\_finishing}}\) of high priority jobs can be defined as:

\[
c_{\text{HP\_finishing}} = \sum_{j \in J} y_{j} \times y_{j}
\]

4. Optimisation of scheduling

In this paper, our main focus is on the scheduling problem as it plays relatively more important role in the port operation than path planning with collision-avoidance which has already been successfully implemented.

The current system has adopted a well-studied collision-free path planner which implements a prioritised multi-vehicle path
planning algorithm (Lau et al., 2008). This planning algorithm is extended from a single-vehicle time-window-based algorithm (Halpern, 1977) to calculate feasible time and cost windows for each vehicle to arrive at and depart from each node (subject to time-dependent node and link availability). Such time windows are propagated iteratively from the known starting time of the vehicle at the start node until a feasible arrival window is found at the destination. The key feature of this algorithm is that the paths generated will consider the motion of all other active SCs and as a result will cause them to go around or give way (via waiting at a node or shunting aside and subsequently resuming) to SCs with already planned paths. Significantly, this is more complex than simplified vehicle routing problems where path lengths only need to be calculated once regardless of the changing occupancy of the various nodes and links in the graph.

This section presents two different solution approaches only for scheduling container transfers. We first present our modified GA approach and then a sequential job scheduling method is presented as an alternative solution.

4.1. The modified GA approach

In general, GAs are global optimisation techniques (Bo, Hua, & Yu, 2006; Cus & Balic, 2003; Goldberg, 1989; Holland, 1992; Smith & Smith, 2002) that avoid many of the shortcomings that exist in classical local search techniques on difficult search spaces. GAs use operators such as reproduction, crossover and mutation as means of preserving beneficial information with the overall goal of finding a better solution to the problem. In addition, genetic algorithms work using codification of the parameter space rather than the parameters themselves. The objective function can be easily defined as a measure of fitness for solution performance which allows the genetic algorithm to retain useful solutions and inhibit those which are less useful. Based on these features a GA-based optimisation method is proposed here to solve our modelled problem. We adopt the two-part chromosome representation (Carter & Ragsdale, 2006) and introduce a chromosome validation and repair method. We have designed an effective strategy to handle the sequence and timing constraints during the fitness calculation.

4.1.1. Two-part chromosome representation

The key to finding a good solution using a GA lies in developing a good chromosome representation of candidate solutions to the problem. A good GA chromosome should reduce or eliminate redundant chromosomes from the population. Redundancy in the chromosome representation refers to a solution being able to be represented in more than one way and appearing in the population multiple times. These multiple representations increase the search space and slow the search. So far, the two-part chromosome technique (Carter & Ragsdale, 2006) has been viewed as the best representation with minimum redundancy for MTSP. The MTSP maps very well to our scheduling problem as each SC can be regarded as a salesman and each job (including pick-up and set-down operations) can be viewed as a city. However, the major difference is that we take into account complex constraints (e.g. job sequence and timings) and practical performance metrics.

The two-part chromosome technique, as the name implies, divides the chromosome into two parts. The first part of length \( n \leq |J| \) represents a permutation of \( n \) jobs and the second part of length \( m = |V| \) gives the number of jobs assigned to each SC. The total length of the chromosome is \( n + m \) in this representation. The \( m \) values present in the second part of the chromosome must sum to \( n \) in order to represent a valid solution. In the example of Fig. 4, the shaded area shows the second part of the chromosome. Here the first SC (SC1) will process jobs 6, 9, 1 and 7 in that order, the second SC (SC2) will process jobs 8 and 4 in that order and the third SC (SC3) will conduct jobs 5, 2 and 3 in that order.

Using the two-part chromosome for solution representation, there are \( n! \) possible permutations for the first part of the chromosome. The second part of the chromosome represents a positive vector of integers \( \{x_1, x_2, \ldots, x_m\} \) satisfying \( x_1 + x_2 + \ldots + x_m = n \), where \( x_i \geq 0, i = 1, 2, \ldots, m \). There are \( \binom{n + m - 1}{n} \) distinct positive integer-valued \( m \) vectors that satisfy this requirement. Hence, the solution space for the two-part chromosome is of size \( n! \binom{n + m - 1}{n} \). Moreover, compared to MTSP with the two-part chromosome representation, the SCs in our problem have different initial depots and they do not have to travel back to their starting depots after finishing all jobs. For our problem which was modelled in Section 3, each candidate schedule for a SC can be regarded as a


Fig. 4. Example of two-part chromosome representation for a nine-job schedule with three SCs.

Fig. 5. An example of calculating SC and QC waiting times and updating related timings.
two-part chromosome in the GA population. In Fig. 4, for example, a candidate schedule for SC1 is represented by four jobs $J_i = \{6, 9, 1, 7\}$ and the related decision variable $X_{1i} = 1$.

4.1.2. Chromosome validation and repair operation

Due to time related constraints (Eqs. (6) and (7)) in scheduling some containers must be picked up or set-down following a fixed order by a single SC particularly for QC/TK associated jobs. Within a randomly created two-part chromosome or via the crossover or mutation operations, however, it is possible to have a few jobs which are not consistent with the predefined order of picking up and setting down. Hence, based on each SC it is necessary to check whether the generated chromosome is feasible in terms of satisfying all timing constraints. If any part chromosome is inconsistent with any fixed order for container pick-up or set-down then we apply a chromosome repair operation so as to ensure the chromosome represents a feasible schedule.

The pseudo code of the chromosome validation and repair operation algorithm is:

**Chromosome validation and repair operation:**

1. **Input:** Any unchecked chromosome.
2. **Output:** A chromosome representing a feasible schedule.
3. **For** $v = 1$:NumSCs/\ NumSCs: total number of SCs
   1. **For** $j = 1$:chromo (SecondPartHead + $v$)/\ SecondPartHead: head index of second part chromosome
      1. If the $j$th job is associated with a predefined order
         1. Store the index and its position in the chromosome;
         2. End
      2. If any two indexes are inconsistent with the predefined order
         1. Sort all indexes according to the predefined order;
         2. For $k = 1$:chromo (SecondPartHead + $v$)
            1. Adjust the job’s position in the chromosome;
            2. End
         3. Else
            1. Keep the part in the original chromosome;
            2. End

The above algorithm is designed to check the order of jobs based on each SC and also repair any inconsistent part according to a predefined order. In the second part of the chromosome, the algorithm finds the assigned jobs for each SC in the first part chromosome based on the predefined order and then checks if the assigned jobs are consistent with the predefined sequence. If not, then we perform a repair operation by adjusting job positions in the first part of the chromosome. After all iterations a valid chromosome is generated which represents a feasible schedule for container transfers.

4.1.3. Fitness calculation strategy

Our fitness function is based on the objective function in Eq. (1), however, the fitness calculation cannot be directly applied on a chromosome. This is because some performance metrics are not independent and cannot be calculated simply based on the chromosome, particularly for the QC or TK associated jobs. During fitness evaluation the total SC waiting time is dependent on QC and TK operations. QC uploading time or discharging time is directly related to the total QC waiting time while TK exporting or importing time is associated with the total TK waiting time. Both QC waiting and TK waiting times are also dependent on SC pick-up time and set-down time. As can be seen all types of timings are closely correlated.

To evaluate the overall fitness of a schedule we initially calculate the total SC travel time by summing up all job setup times and job processing times, which all are based on Dijkstra's algorithm (Dijkstra, 1959). The job setup time refers to the duration of empty travelling time for a SC moving to a particular job pick-up position while the processing time refers to the duration of time for a SC transporting a container from a job pick-up position to the set-down position.

Then, the total SC waiting time, the total QC waiting time and the total TK waiting time are calculated together. We transform the two-part chromosome into a set of groups based on each SC and then each job which is assigned to a SC is mapped into different slots. A pair of raw (estimated) pick-up times and raw set-down times is created initially for each job by assuming there is no SC waiting time. By assuming there is no QC waiting time a set of raw QC uploading times and QC discharging times are created for each QC based on the QC starting time $(t_0^k$ and $t_1^k$) and turnaround time $(\Delta t_k^b)$. Likewise, TK raw timings are created in the same way based on the TK starting time $(t_0^k$ and $t_1^k$) and turnaround time $(\Delta t_k^b)$. For each slot we find all QC/TK related jobs with raw timings and update the dependent jobs (via $\delta_{ab}$ or $\delta_{ab}$). If its dependent job has been updated then we check the job raw timings and identify the SC waiting, QC waiting or TK waiting times and then update all related timings. If its dependent job has not been updated yet then we ignore the job until its dependent job has been updated. However, since each QC has two buffer nodes the evaluation on QC waiting time and related SC waiting time should take into account the two buffer nodes. That is, for each uploading QC job $j$ $(j < k, j > 2)$ its QC waiting time should be calculated based not only on the current raw QC timings but also the set-down time of the second previous job $(j - 2)$ $t_0^k$. Likewise, for each discharging QC job $j$ $(j < k, j > 2)$ the QC waiting time should be calculated based not only on the current raw QC timings but also the pick-up time of the second previous job $(j - 2)$ $t_0^k$. After all iterations for the slots has been performed the SC waiting time, QC waiting time and TK waiting time can be calculated and all related timings updated. An example is provided in Fig. 5 to further illustrate the process.

In Fig. 5, $J_5$ and $J_3$ are TK exporting jobs, $J_4J_5$ and $J_6$ are QC uploading jobs and the other jobs are Y2Y jobs. As each QC has two buffer nodes, $J_5$ is dependent on $J_4$ and $J_6$ must be set-down before setting down $J_5$ at the same buffer. Here, $J_5$ can be set-down at another buffer as long as it satisfies the timing of uploading QC for $J_3$. $J_5$ is required to be picked up before picking up $J_6$. The raw timings of TK and QC are calculated using the starting times of QC and TK and the turnaround time.

In the first slot $(S_1)$, $J_3$ and $J_5$ need to be checked for related raw timings and $J_6$ should be checked after checking $J_5$. The raw QC uploading time for $J_5$ is 22 and the raw set-down time of $J_5$ is 15, therefore SC1 needs to wait for seven time units. Accordingly, all pick-up and set-down timings of $J_2$ and $J_3$ should be updated by adding the seven time units. By checking the timings of $J_6$, SC2 needs to wait for two time units as the raw pick-up time is 8 and raw TK exporting time is 10. Accordingly, all pack-up and set-down timings of SC2 should be updated by adding 2 time units.

In the second slot $(S_2)$, $J_4$ and $J_6$ need to be checked for related timings. The latest updated set-down time of $J_4$ is 21 and the raw QC uploading time is 12 for $J_4$ so the QC needs to wait for nine time units. The QC discharging time for $J_6$ should be updated to 31 from 22. As $J_6$ is at the previous slot and dependent on $J_4$ and $J_5$, it should be checked before $J_5$ by $J_4$ in this slot. The latest updated QC uploading time of $J_4$ is 21 and the raw set-down time of $J_6$ is 15 and then SC1 needs to wait for six time units. Accordingly, the pick-up and set-down timings of $J_6$ $J_3$ and $J_5$ should be updated by adding the six time units. Regarding $J_6$, the raw TK exporting time is 20 and the pick-up time for SC is updated to 22 and TK needs to wait for two time units.
units. For the third and fourth slots it is not necessary to check and update the timings because there is no QC/TK related job. Overall, the total SC waiting time is 15, the total TK waiting time is 2 and the total QC waiting time is 9.

The performance of high priority job finishing times can be evaluated based on all previously updated timings. The total value of high priority job finishing times is the summation of set-down times $\left( T_j^s, j \in P_H \right)$ of all high priority jobs.

The following pseudo code shows the chromosomal fitness calculation of a valid two-part chromosome using the objective function.

**Fitness calculation**:

**Input**: A valid two-part chromosome.

**Output**: Fitness value ($f$) for all performance metrics.

Variable and parameters initialisations for the chromosome and set $f = 0$;

Generate the raw timings for all QCs and TKs based on their turnaround times;

for $v = 1$ : numSCs
  for $j = 1$ : chromo(SecondPartHead + v) // for each assigned job of the SC
    Calculate and store the ideal pick-up time $\left( T_j^p \right)$ and set-down time $\left( T_j^s \right)$ for the assigned job $j$;
    $f = f + \lambda_1 \times$ the travel time for the job setup and job processing;
  end
  end

maxNumSlots = max (values of the second part chromosome);

for $i = 1$ : maxNumSlots
  Get QC and TK related jobs which have not been checked yet in slot ($i$);
  for $k = 1$ : numUnchecked // search unchecked job, check if the job's dependent job has been updated;
    if the job's dependent job has been updated
      Compare the raw timings of pick-up/set-down with the related QC and TK timings;
      Calculate the waiting time and update all raw timings and all QC and TK related jobs related to jobs;
      if the waiting time is caused by SC
        $f = f + \lambda_2 \times$ SCWaiting;
      elseif QC needs to wait
        $f = f + \lambda_3 \times$ QCWaiting;
      elseif TK needs to wait
        $f = f + \lambda_4 \times$ TKWaiting;
      end
    end
  end

for $j = 1$ : numHighPriorityJobs
  $f = f + \lambda_5 \times T_j^f$;
end

**4.1.4. Selection and crossover**

We utilise the rank-based roulette-wheel (Goldberg, 1989) selection method in our GA. With this approach each chromosome is assigned a portion of an imaginary roulette wheel which is based on the chromosome rank. Chromosomes which have a higher fitness value are allocated a larger segment of the roulette wheel. Selection of a parent requires random generation of a number between 0 and 1. The chromosome occupying the section of the roulette wheel covered by the randomly generated percentage is chosen as a parent. The second parent is selected in the same manner.

A modified crossover operator is required to handle the two-part chromosome. For the first part of the two-part chromosome our modified crossover operator is consistent with the ordered crossover method (Carter & Ragsdale, 2006), which it is commonly used for planning with sequence constraints. The parent chromosomes are selected using the rank-based roulette-wheel scheme described above. Given two parent chromosomes two random crossover points are selected which partitions the chromosome into a left, middle and right portion. The ordered two-point crossover behaves in the following way: Child A inherits its middle section from Parent A, and its left and right sections are determined, as shown in Fig. 6.

For the second part of the two-part chromosome we adopt a single point asexual crossover operator as shown in Fig. 7. This method simply cuts the second part of the chromosome into two sections and reverses the order in which the two pieces were arranged. This type of crossover ensures that the second part of the chromosome remains feasible (with the sum of the values in the chromosome equalling $n$). After the crossover operations for both the first and second parts of the two-part chromosome it is necessary to perform chromosome validation and repair for all generated children chromosomes.

**4.1.5. Mutation**

The mutation operator used in our GA is based on the swap mutation (Goldberg, 1989) which consists of randomly swapping two genes in the first part of the two-part chromosome. This swap mutation operator performs a very vital task because it provides the means by which jobs are exchanged to seek improvement. Crossover has the ability to drastically impact the groupings of jobs, but has little impact on the ordering of the jobs within each group. A low mutation probability is used to help explore alternative solutions and maintain diversity of the solutions in the population. After each mutation operation it is also necessary to perform a chromosome validation and repair for all newly generated chromosomes.

**4.1.6. Replacement**

We use the replacement policy of Steady-State GA (Dejong, 1975). Parents are selected to produce offspring and then a decision is made as to which individuals in the population to select for deletion to make room for the newly created offspring. Steady-State GA is an overlapping system since parents and offspring
compete for survival. A percentage of the replacement determines how much of the population each generation is replaced by the newly generated children.

4.2. Sequential job scheduling approach

Sequential job scheduling is often considered as a typical solution in manufacturing systems that deal with many different components particularly for the job scheduling problem with priority constraints (Kim, Naa, & Chenb, 2003). In order to reduce quay crane waiting and truck waiting times a sequential job scheduling based approach is an alternative method to handle the scheduling problem. The basic idea of the sequential job scheduling is illustrated in Fig. 8.

This approach aims to ensure quay crane related jobs and truck related jobs will have the higher priority of scheduling than Y2Y jobs. Essentially all jobs are sorted according to their predefined pick-up/set-down times. At each iteration of the algorithm the first job in list is allocated to its nearest vehicle so as to ensure no delay for quay cranes or trucks and also minimise SC travel time. The allocated job is removed from the job list and the algorithm iterates until the job list is empty.

The full Patrick scheduling system applies some more sophisticated techniques including pre-optimisation for the job schedule structure and considers additional business requirements. However, we implemented a sequential job scheduling approach with its well-understood nearest-vehicle selection policy and comparable performance in smaller problems and compared the performance with the GA-based approach. The advantage of the sequential job scheduling method is that high priority jobs are always serviced first and QC/TK related jobs are treated as more important than Y2Y jobs during scheduling. However, the downside of the sequential job scheduling approach is that some SCs have unnecessary waiting time since they may arrive early at the work point and wait for a long time until other higher priority quay crane and truck jobs are transferred to the yard area. Furthermore, this approach limits SC utilisation so as to avoid introducing quay crane and truck delay.

5. Experimental results

This section presents performance comparisons between the GA-based approach and the sequential job scheduling method via simulation experiments and live testings in Patrick AutoStrad scheduling system. In our simulation experiments both the GA approach and sequential job scheduling algorithms have been implemented in C++. All experiments were performed on a High Performance Computing Linux Cluster configured using 2 x 3.33 Ghz Quad Core Xeon with 24 GB of DDR3-1333 ECC Memory. Live testings in Patrick AutoStrad terminal was accomplished by integrating the GA-based algorithm with the scheduling system.

5.1. Simulation testings

Firstly, we conducted the comparison on performance of the GA-based method and the sequential job scheduling approach for scheduling a small amount of jobs (24 jobs). Table 1 shows the parameter values for the experiment on the simple scheduling scenario which includes QC and TK related jobs and Y2Y jobs. We have attempted to base the parameter values on those typically found in seaport terminal operations. The 24 mixed jobs consist of B2Y (6 jobs are associated with one discharging QC), Y2B (six jobs are associated with one uploading QC), T2Y (two jobs are associated with one exporting TK), T2T (two jobs are associated with one importing TK) and Y2Y (eight jobs), and there are eight available SCs.

Experimental results include different performance metrics used to evaluate the effectiveness of the proposed algorithms. These metrics include total SC travel cost, total SC waiting cost, total TK waiting cost, total QC waiting cost and total high priority job
finishing cost which have been defined in Section 3.5. Since the GA
is a stochastic search algorithm, it is necessary to perform at least
30 independent runs to obtain the average values (GA_Average) and
the best result (GA_Best) found by GA.
As shown in Fig. 9 the GA-based approach outperforms the
sequential job scheduling algorithm (SQT) for the total cost of
scheduling the 24 jobs for both GA_Average and GA_Best. The solution
found by GA improved the overall performance in this experiment by
42.50% (GA_Average) and 45.10% (GA_Best). The GA achieves a better result than SQT for SC total waiting cost, TK total waiting
cost, while SQT has only slightly better values for SC total travel
cost and high priority job (HPJ) finishing cost. For the QC total
waiting cost, the performance of SQT and GA are similar because
the weight of scheduling QC related jobs is large (k3 = 20)
and both algorithms try to reduce QC total waiting cost as much
as possible. The GA sacrifices a little SC total travel cost and HPJ fin-
ishing cost so as to gain much more reduction in SC total waiting
cost and TK total waiting cost.

We also compared the performances of the GA-based method
and the sequential job scheduling approach for scheduling a large

time-related value (ms)

Comparison on SQT and GA

Performance metrics

![Comparison on SQT and GA](image)

Table 3 summarises the average performance and standard deviation of the GA and SQT algorithms for the total cost. A significant improvement in performance of the GA with respect to the SQT algorithm is indicated by the $t$-value. According to the critical value ($t = -2.00$), the results show the performance of GA is statistically significantly better than the performance of SQT algorithm for both 24 mixed jobs and 80 mixed jobs.

Table 3: Comparison of GA vs sequential job scheduling algorithm for total cost.

<table>
<thead>
<tr>
<th></th>
<th>MEAN</th>
<th>STDEV</th>
<th>$t$-Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 Mixed jobs</td>
<td>SQT</td>
<td>7.90E+06</td>
<td>0.00E+00</td>
</tr>
<tr>
<td></td>
<td>GA</td>
<td>4.50E+06</td>
<td>1.11E+05</td>
</tr>
<tr>
<td>80 Mixed jobs</td>
<td>SQT</td>
<td>3.11E+07</td>
<td>0.00E+00</td>
</tr>
<tr>
<td></td>
<td>GA</td>
<td>2.30E+07</td>
<td>1.50E+05</td>
</tr>
</tbody>
</table>

**5.2. Live testing results at Patrick AutoStrad terminal**

Our GA-based approach has been implemented on a trial basis in the scheduling system of Patrick AutoStrad Terminal in Brisbane, Australia. To show the effectiveness of the GA-based approach we obtained a set of random live testing results from Patrick AutoStrad Terminal and compared to the actual scheduling optimizer (OP) of Patrick.

Table 4 shows the general settings for the GA throughout the testing. All the problems listed in Table 4 are a random snapshot based on typical operations at the Patrick AutoStrad terminal. The planning performance of OP and GA-based approach are calculated by an evaluation function in Patrick’s system. Both OP and GA are integrated with a local optimising process which conducts the local optimisation for each schedule based on a local hill climbing mechanism. Moreover, a number of additional practical requirements are taken into account in the actual operating system including SC contention and the MIN/MAX number of SCs. SC contention is used to penalise the job scheduling when some of the SCs are likely to travel on paths which are close together and might cause some traffic delay. The MIN/MAX number of SCs needs to be ensured in some particular sets of jobs. Fundamentally, the MIN number is to ensure that for the duration of the planning horizon the throughput of container movements respects a certain level of expected throughput that roughly equates to the number of straddle carriers set as the MIN for any order of work While the MAX number applies a large cost to any period of time on the planning horizon in which more than MAX SCs are simultaneously working on the same order of work. The precise details are not presented in this paper due to commercial considerations. Nevertheless.

Table 4: GA settings.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA population size</td>
<td>100</td>
</tr>
<tr>
<td>Number of generations</td>
<td>50 (Approx. 2 min)</td>
</tr>
<tr>
<td>Crossover probability rate</td>
<td>0.85</td>
</tr>
<tr>
<td>Mutation probability rate</td>
<td>0.01</td>
</tr>
<tr>
<td>Selection method</td>
<td>Roulette-wheel selection</td>
</tr>
<tr>
<td>Mutation method</td>
<td>Swap</td>
</tr>
<tr>
<td>Replacement in GA</td>
<td>Steady-state GA</td>
</tr>
<tr>
<td>Replacement percentage</td>
<td>50%</td>
</tr>
</tbody>
</table>

Table 5: Various testing results from Patrick AutoStrad scheduling system.

<table>
<thead>
<tr>
<th>Snapshot problems</th>
<th>Score of OP</th>
<th>Score of GA</th>
<th>Raw improvement by GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1 (SCs = 16, Jobs = 21)</td>
<td>471,781</td>
<td>471,704</td>
<td>77</td>
</tr>
<tr>
<td>Case 2 (SCs = 20, Jobs = 32)</td>
<td>104,447</td>
<td>104,109</td>
<td>338</td>
</tr>
<tr>
<td>Case 3 (SCs = 14, Jobs = 32)</td>
<td>280,304</td>
<td>277,861</td>
<td>2443</td>
</tr>
<tr>
<td>Case 4 (SCs = 18, Jobs = 14)</td>
<td>105,647</td>
<td>97,355</td>
<td>8292</td>
</tr>
<tr>
<td>Case 5 (SCs = 14, Jobs = 21)</td>
<td>361,782</td>
<td>360,192</td>
<td>1590</td>
</tr>
<tr>
<td>Case 6 (SCs = 18, Jobs = 9)</td>
<td>58,896</td>
<td>58,853</td>
<td>43</td>
</tr>
<tr>
<td>Case 7 (SCs = 18, Jobs = 10)</td>
<td>69,908</td>
<td>69,282</td>
<td>626</td>
</tr>
<tr>
<td>Case 8 (SCs = 16, Jobs = 3)</td>
<td>55,675</td>
<td>55,675</td>
<td>0</td>
</tr>
<tr>
<td>Case 9 (SCs = 19, Jobs = 17)</td>
<td>53,057</td>
<td>53,040</td>
<td>17</td>
</tr>
<tr>
<td>Case 10 (SCs = 13, Jobs = 20)</td>
<td>81,647</td>
<td>81,564</td>
<td>83</td>
</tr>
<tr>
<td>Case 11 (SCs = 11, Jobs = 60)</td>
<td>119,716</td>
<td>119,664</td>
<td>52</td>
</tr>
<tr>
<td>Case 12 (SCs = 14, Jobs = 47)</td>
<td>70,870</td>
<td>70,747</td>
<td>123</td>
</tr>
</tbody>
</table>

less, the GA-based approach outperforms the OP method with positive improvements through the 12 testings as shown in Table 5.

For the 12 different port scenarios the GA outperformed OP. Variations in the improvement are because each case has different conditions, such as the number of SCs, number of jobs, pick-up and set-down locations of jobs and SC initial position. In case 8 as the problem is relatively simple (i.e. there are only three jobs and 16 available SCs) the performance scores of GA and OP are the same, while the most significant difference is made by GA at case 4, where the performance score is improved by 8292. Overall, the effectiveness of the GA-based approach is shown by numerical experiments and live testing results on different sets of data.

6. Conclusion

This paper has presented a modified mathematical model derived from the problem of container transfers at the Patrick Auto-Strad container terminal located in Brisbane, Australia. This model incorporates QC related operations, SC scheduling and TK related operations. Additionally, a practical contribution of this paper is the GA-based approach which was presented to solve the job scheduling problem and was compared to a sequential job scheduling method for different scheduling scenarios. The proposed approach has been fully implemented on a trial basis in the live scheduling system at Patrick container terminal and it effectively improves the performance of the seaport container terminal.

Our future work will focus on two main areas. Firstly, as uncertainty is a critical concern in port operations we will investigate and further improve the performance of GA by considering a replanning scheme for both short term and long term planning. Secondly, we would develop a parallel computing architecture to boost the convergence velocity and solution quality of the GA. This would permit more improvements on the productivity of the transshipment process.

Acknowledgments

This work is supported by the ARC Linkage Grant [LP0882745], the ARC Centre of Excellence program (funded by the Australian Research Council (ARC) and the NSW State Government), the Patrick Stevedores Holdings and the University of Technology, Sydney, Australia.

Moreover, we would like to thank anonymous reviewers for their comments and suggestions, which have really helped us improve the quality and presentation of the paper further.

References


