Rescheduling Policies for Large-Scale Task Allocation of Autonomous Straddle Carriers under Uncertainty at Automated Container Terminals

Binghuang Cai*, Shoudong Huang, Dikai Liu, Gamini Dissanayake

Centre for Autonomous Systems (CAS), Faculty of Engineering and Information Technology (FEIT), University of Technology, Sydney (UTS), PO Box 123, Broadway, NSW 2007, Australia.

Abstract

This paper investigates replanning strategies for container-transportation task allocation of autonomous Straddle Carriers (SC) at automated container terminals. The strategies address the problem of large-scale scheduling in the context of uncertainty (especially uncertainty associated with unexpected events such as the arrival of a new task). Two rescheduling policies - Rescheduling New arrival Jobs (RNJ) policy and Rescheduling Combination of new and unexecuted Jobs (RCJ) policy - are presented and compared for long-term Autonomous SC Scheduling (ASCS) under the uncertainty of new job arrival. The long-term performance of the two rescheduling policies is evaluated using a multi-objective cost function (i.e., the sum of the costs of SC travelling, SC waiting, and delay of finishing high-priority jobs). This evaluation is conducted based on two different ASCS solving algorithms - an exact algorithm [i.e., branch-and-bound with column generation (BBCG) algorithm] and an approximate algorithm (i.e., auction algorithm) - to get the schedule of each short-term planning for the policy. Based on the map of an actual fully-automated container terminal, simulation and comparative results demonstrate the quality advantage of the RCJ policy compared with the RNJ policy for task allocation of autonomous straddle carriers under

*Corresponding author. Tel: +61-2-9514-2676. Fax: +61-2-9514-2655.
E-mail Addresses: bhcai8@gmail.com, bhcai@ieee.org (B. Cai).
Postal Address: Centre for Autonomous Systems (CAS), Faculty of Engineering and Information Technology (FEIT), University of Technology, Sydney (UTS), PO Box 123, Broadway, NSW 2007, Australia (B. Cai).

uncertainty. Long-term testing results also show that although the auction algorithm is much more efficient than the BBCG algorithm for practical applications, it is not effective enough, even when employed by the superior RCJ policy, to achieve high-quality scheduling of autonomous SCs at the container terminals.

Keywords: Rescheduling policy, Uncertainty, Optimisation, Task allocation, Autonomous straddle carriers, Automated container terminals

1. Introduction

In the past several decades, the capacity and frequency of container ships arriving at container terminals has increased steadily due to both increasing containerization and world trade. Correspondingly, container terminals need to turn around larger ships carrying more containers as fast as possible to improve productivity and reduce the terminal operation costs. As a consequence, the terminals have a number of different types of yard resources to assist in the movement of containers, including for example, yard vehicles (for transporting containers around different yard areas) and yard cranes (for transporting containers in a fixed small area at the terminal) [1–3]. It is important for modern container terminals that these yard resources are used efficiently to load, unload and transfer containers during the transhipment process. Yard vehicles play a very essential role in container transportation because they are more flexible than yard cranes in being able to move freely within the yard. The rapid development of autonomous material handling vehicles and robots has facilitated the development and deployment of automated equipment for container terminals, such as Automated Guided Vehicles (AGVs) [4–6] and Automated Lifting Vehicles (ALVs) [7] as yard vehicles. Compared with human-operated vehicles, these automated material handling robots require a high degree of coordination and efficiency. The effective operation of automated yard vehicles hence becomes an essential issue, one that has been investigated by researchers and engineers in robotics and logistics. A number of methods and approaches have been proposed including, for example, two heuristic methods - one based on a flexible priority rule for AGV dispatching at highly automated container terminals in [5] and another based on a mixed-integer programming model for dispatching a small fleet of ALV [7]. Different approaches, such as simulated annealing, ant colony, auction algorithm, job grouping and column generation have

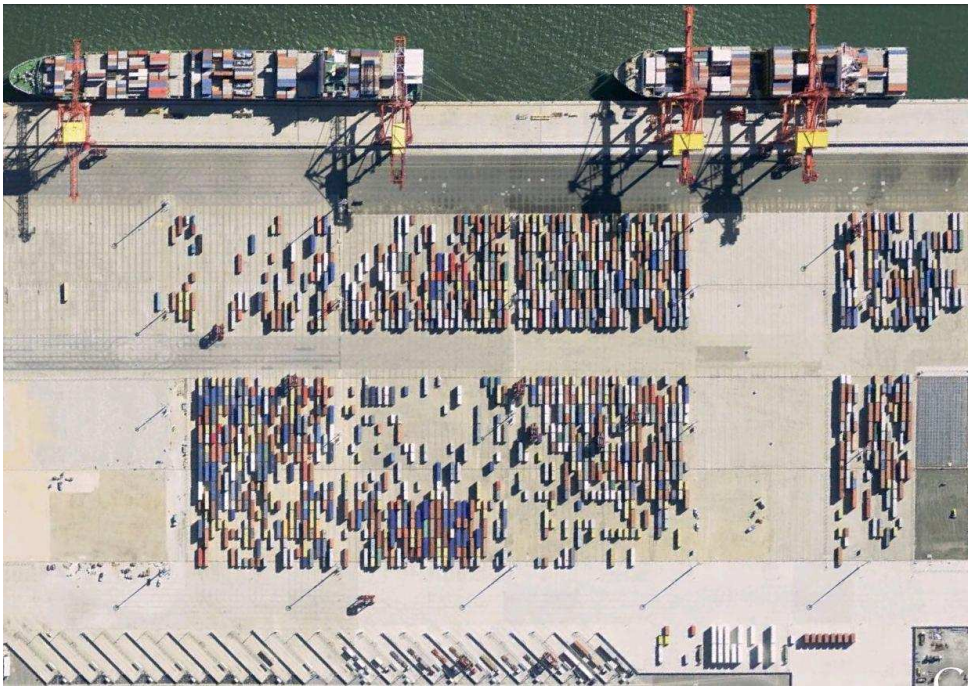

Figure 1: Patrick AutoStrad container terminal [Google Earth, 16 June, 2009].

been investigated for multiple autonomous vehicle operation at an automated container terminal by the authors [8–13].

The Patrick Autostrad container terminal (a fully-automated container terminal shown in Fig. 1) uses a type of ALV known as an autonomous Straddle Carrier (SC) [14–16]. The operation of the autonomous SCs plays a key role in enabling container transportation to increase the transhipment productivity of the terminal. A fundamental problem for the SC operation is SC task allocation. This is known as Autonomous Straddle Carrier Scheduling (ASCS): finding a feasible and efficient schedule for the straddle carriers to finish a list of container transportation jobs. The found schedule should satisfy the relations and constraints of yard vehicles, transhipment jobs, and seaport environments, as well as meet performance requirements of the terminal operation. The ASCS problem is very difficult to solve due to 1) large number of jobs, 2) large number of SCs and 3) complicated terminal environments. Scheduling algorithms have been developed and employed to solve such a problem and some of these algorithms (e.g., genetic algorithms and job grouping developed by the authors) have been or are being implemented

in the real operation system of the Patrick AutoStrad container terminal [11]. These methods are based on approximate methodology, which could get a feasible solution but may not be optimal. Note that optimal scheduling can greatly reduce the cost of the operation of the container terminal [14–16]. An exact algorithm based on column generation has been presented for the ASCS problem to obtain the optimal schedule [12, 13]. However, this column generation based algorithm is more time-consuming for large-scale problem than approximate methods. Thus, to improve algorithm efficiency and solution quality, there remains a need to investigate approaches for solving large-scale ASCS problem.

An important aspect of the task allocation problem of autonomous SCs is that the operation of autonomous SCs is subject to uncertainties that occur due to interaction with the real-world (autonomous vehicles can stop, delay or face problems like a new job arrival) and hence job and vehicle schedules need to be replanned [17]. Replanning of autonomous SC in dynamic container handling environments is essential for maintaining high productivity in the face of unexpected events. Thus, the ASCS problem should be solved in a way that it can react to such uncertainties. One of the most common and effective ways to do this is to formulate and solve small-scale ASCS problem from time to time using updated information based on the change of jobs and the state of the SCs. To some extent, this is similar to job-shop rescheduling in manufacturing systems [18, 19]. Different strategies and policies, such as new job rerouting, complete rerouting, periodic rescheduling and event-driven rescheduling, have been investigated for job-shop rescheduling [18, 19]. Dynamic replanning methods have been investigated and developed by robotic and logistic researchers in relation to the replanning of yard cranes and yard vehicles at container terminals [17, 20, 21]. However, their efficiency and effectiveness still require improvement for long-term operation performance, especially for application in autonomous SC task allocation under uncertainties.

This paper focuses on investigating replanning strategies for large-scale ASCS problems under the uncertainty of new job arrival and on providing related guidelines for the operation of automated container terminals. Two event-driven rescheduling policies are developed and compared for the long-term task allocation of autonomous straddle carriers to handle an unexpected event of new job arrival: 1) Rescheduling New Jobs (RNJ) policy (solving the ASCS problem with new arrival jobs only in each planning) and 2) Rescheduling Combination of new and unexecuted Jobs (RCJ) policy

(solving the ASCS problem with all the new jobs and unexecuted jobs in the previous planned schedule). To evaluate the performance of the two policies, a multi-objective cost function is used in the form of combining the costs of SC travelling, SC waiting and delay of finishing high-priority jobs via weighting factors. For each rescheduling policy, a Branch-and-Bound with Column Generation (BBCG) algorithm [12, 13] and a simple auction algorithm are alternatively employed to get the schedule in each small-scale planning. Simulations are conducted based on the practical map of the Patrick container terminal. These demonstrate the superior performance of the RCJ policy compared with the RNJ policy for large-scale SC scheduling. They further demonstrate the need for an improved auction algorithm to aid optimisation of large scale SC scheduling, especially in long time horizon.

The remainder of this paper is organized into five sections. Section 2 describes the large-scale ASCS problem under uncertainty. Section 3 explains the RNJ and RCJ rescheduling policies. The mathematic model of the ASCS subproblem in each planning and its solution algorithms are provided in Section 4. Simulation and comparative results are presented in Section 5. Section 6 concludes this paper with final remarks.

2. Large-Scale ASCS Problem under Uncertainty

The Patrick AutoStrad container terminal (with satellite view shown in Fig. 1) is a representative automated container terminal located in Brisbane, Australia, with Fig. 2 showing the schematic diagram of the static seaport environment. The seaport can be modelled as a yard environment map with 18380 positional nodes and 83155 predefined links [11, 14]. A fleet of autonomous SCs, capable of independently lifting a container, transferring the container from the pickup position and setting down the container at a destination, operate within the actual port environment for container transportation between different yard areas.

The ASCS problem is defined as assigning container transportation jobs to straddle carriers so that the SC and job constraints can be satisfied and the objective cost can be optimised. For each job, the SC should pickup a container from one position and transfer it to another position to set down. Both the pickup and setdown operations should be started within a given time window if required. In the solution of the ASCS problem, different job sequences are assigned to different SCs and a given job is only assigned to one SC. For the objective cost, the sum of costs of different yard resources, such

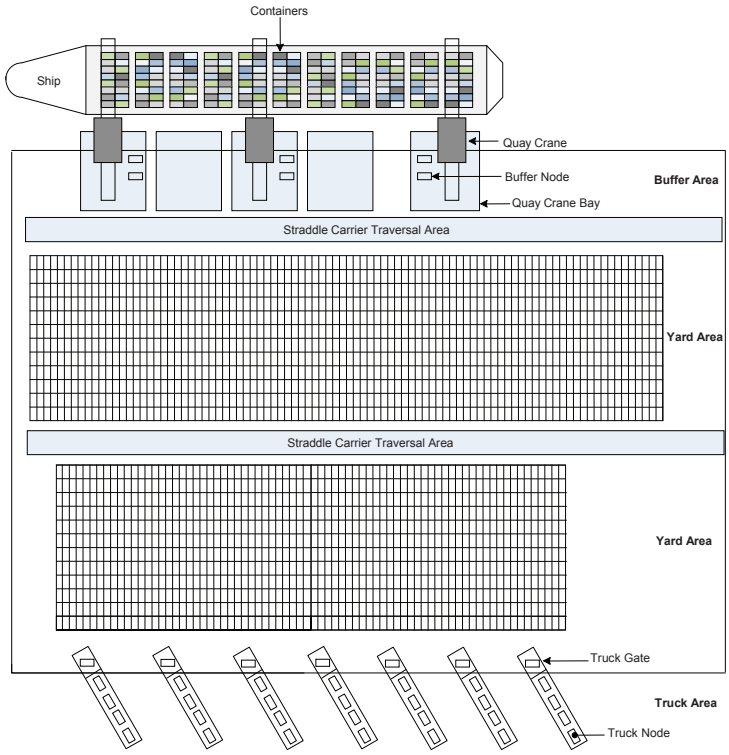

Figure 2: Schematic diagram of the static seaport environment [12].

as SC utilisation cost and delay cost of high-priority tasks, can be considered in the ASCS problem solving and the SC operation performance evaluation. For long-term performance, the main focus should be on actual benefit and cost, and the objective should be maximising the economic benefit. Thus, the unit of all the costs of the yard resources are needed to be the same (e.g. dollars) [22].

In the reality of SC task allocation, there are often many jobs to be allocated to many SCs, and hence the ASCS problem would always be in large-scale. Schedules would be very difficult or even impossible to get (especially, in the case of achieving the optimal solution) due to the large search space of the combination of jobs and SCs. According to [12, 13], solving the ASCS problem optimally is an integer-programming problem, which is an NP-hard (Non-deterministic Polynomial-time hard) problem. Exact algorithms (e.g., BBCG algorithm [12, 13]) can only efficiently solve small-scale ASCS problems, such as the ones with 10 jobs and 3 vehicles. For a large-scale ASCS

problem, the BBCG algorithm is very time-consuming and it may not be able to be applied in a real-time operation situation. For example, it took more than 2 hours to get the optimal solution of a 40-job 9-vehicle ASCS problem [12], which may not satisfy the requirement of solving the problem in couple of minutes for a practical SC operation. Generally speaking, exact (or even nearly-optimal) solution algorithms (e.g., BBCG algorithm) may not be efficient enough for large-scale SC task allocation problems at automated container terminals.

In addition to its large-scale, SC operation at automated container terminals is further characterized by the intermittent and unexpected arrival of new tasks. Expected new tasks (e.g. transporting containers being discharged from ships or trucks) may be taken into account in the ASCS problem at the beginning of planning. However, many of the new arrival jobs are unexpected tasks, and information about these jobs are unknown and unable to be considered into the ASCS problem at the beginning of planning. In this sense, the ASCS problem can not be solved in one go at the beginning. Note, however, that, from the problem point of view, new job arrival does not change the ASCS problem and does not increase the difficulty of solving the ASCS problem. By contrast, from the operation point of view, new jobs do affect the operation and need to be handled in different ways.

Replanning is commonly and effectively used to handle the large-scale ASCS problem and dynamic new job arrivals for the operations of a large fleet of straddle carriers. The large-scale ASCS problem can be divided into small-scale subproblems for planning based on the job arrival time. In each planning, the small-scale ASCS subproblem can be formulated and solved, significantly reducing the computational time and effectively handling the uncertainty of new job arrivals for large-scale SC task allocation. Furthermore, in the operation of an automated container terminal, long-term operation performance (e.g. operation cost and productivity) is key to evaluating the efficacy of the operation of a multiple SC system. As un-predicted events, especially new job arrivals, occur frequently, simple replanning strategies have been used in the practical implemented straddle carrier system. But which replanning strategy performs better, when replanning should be triggered, and how good the replanning strategy can perform in a long time horizon still needs to be determined.

3. Rescheduling Policies

To address the issues stated above, two event-driven rescheduling policies are presented and compared for large-scale task allocation of autonomous straddle carriers under job arrival uncertainty at an automated container terminal. The two rescheduling policies are Rescheduling New Jobs (RNJ) policy and Rescheduling Combination of new and unexecuted Jobs (RCJ) policy, with detailed descriptions given below.

3.1. RNJ policy

At the time of a new job arriving, the RNJ policy solves the ASCS problem with new arrival jobs only and keeps the planned schedules unchanged. Assuming n_{new} new jobs arrive at time t, RNJ policy will solve the ASCS subproblem (modelled in the next section) with the new number of jobs $n = n_{\text{new}}$. Note that, the number of SCs m is unchanged in the rescheduling, since we consider the same group of SCs during the whole scheduling operation in this paper. In the new ASCS subproblem, the initial position of SC v is updated as the final position of SC v in the previous planned schedule, i.e., the setdown position of the last job assigned to the SC in the previous schedule; while SC available time is updated as the finishing time of the previous planned schedule assigned to SC v, i.e., the servicing time of SC v at the setdown position of the last job assigned to SC v in the previous schedule. Other parameters (e.g., operation time windows and SC travel time) for the new ASCS subproblem can be updated based on the information of the new arrival jobs. Then, the new ASCS subproblem can be solved by solution algorithm (e.g., the BBCG algorithm or the auction algorithm to be developed in the next section) to get the new schedule for the new jobs.

3.2. RCJ policy

At the time of a new job arriving, the RCJ policy solves the ASCS subproblem with new arrival jobs and unexecuted jobs from the previous planned schedule, and the jobs being executed (i.e., the ones are being executed at the time of new job arrivals) will keep unchanged and not be taken into account in the new replanning. Assuming n_{new} new jobs arrive at time t and $n_{\text{unexecuted}}$ jobs have not yet been executed in the planned schedule at time t, RCJ policy will solve the ASCS subproblem with the new number of jobs $n = n_{\text{new}} + n_{\text{unexecuted}}$. In the new ASCS subproblem of the new planning, SC initial position (SC available time) is the setdown position (finishing time)

of the job being executed by the SC in the previous planned schedule, or the setdown position (finishing time) of the last job assigned to the SC in the previous plan if the SC has already finished all assigned jobs. Other parameters (e.g., operation time windows and SC travel time) for the new ASCS subproblem can be updated based on the information of the new arrival jobs and the unexecuted jobs. The new ASCS subproblem then can be solved by ASCS solution algorithm (e.g., BBCG algorithm or auction algorithm) for the new and unexecuted jobs.

In summary, the RNJ policy is to simply fix the already planned schedule and only plan the new arrived tasks (unscheduled tasks) with available straddle carriers. The RCJ policy is to fix the tasks already executed and being executed unchanged and replan all the unexecuted tasks (scheduled in the previous planning) together with the new arrived tasks (unscheduled tasks). The jobs being executed will keep unchanged and will not be taken into account in the newly rescheduling. For the RNJ policy, we can say that the current schedule is left in place for execution and new arrival jobs are rescheduled using available resources (i.e., straddle carriers). For the RCJ policy, the unexecuted jobs in the previous schedule will be rescheduled together with the new jobs in current scheduling, which is a kind of backtracking. It is worth mentioning that, in practice, the scheduling of straddle carriers is operated in real time. Once the straddle carrier is executing a job, it would not be able to change the job and it would not be free before the job is finished. Thus, the jobs currently being executed are kept unchanged in both polices. Moreover, both of the rescheduling policies are triggered by the event of the arrival of new tasks. The BBCG algorithm and the auction algorithm will be applied to obtain the optimal solution for the deterministic task allocation problem in each planning (i.e., ASCS subproblem). Furthermore, multiple objective cost function, combining costs of SC travelling, SC waiting and delay of high-priority job together, will be employed for the evaluation of long-term performance of the presented polices when new tasks keep arriving for a long time period. Note that the resource (i.e., straddle carriers) used for the new scheduling can be available at the current time and also at a later time. The available time of straddle carriers can be easily considered in the ASCS subproblem modeling in the next section by setting the time windows of the straddle carriers.

4. Model and Solution of ASCS Subproblem

Based on our previous work [12, 13], the ASCS subproblem in each planning is modelled and formulated as a Pickup and Delivery Problem with Time Windows (PDPTW) [23] in the form of Binary Integer Programming (BIP). Three types of costs for yard resource operation are considered as the objective to be minimised in the modelling. The first one is to minimise the total travelling cost of all SCs to finish all given jobs, which is proportional to the total SC Travelling Time (SCTT). The second, is to minimise the total cost of SC waiting during the transportation of a list of given jobs, being proportional to the total SC Waiting Time (SCWT), which can reduce the SC idle time and effectively use the SCs for container transhipment. The third, is to minimise the total cost of delaying high-priority jobs [defined as the proportional value of High-priority Job Finishing Time (HJFT)], since such jobs have to be finished as soon as possible to satisfy customers' requirements. As the three objective costs are all proportional to time, they are combined together via weighting factors (with the unit being dollar per second) to form the objective cost (in dollar) for the ASCS subproblem and the operation performance evaluation. Extending from [12, 13], the multi-objective ASCS subproblem is then modelled in mathematical formulation below.

4.1. Definitions of parameters and variables

The given parameters and defined variables for the multi-objective PDPTW-BIP model of the ASCS subproblem are described as follows.

4.1.1. Given parameters

- n: number of jobs.
- m: number of SCs.
- $V = \{1, \cdots, m\}$: SC set, indexed by v.
- $S = \{1, \cdots, m\}$: SC initial points.
- $F = \{m + 2n + 1, \cdots, 2m + 2n\}$: SC final points.
- $P^+ = \{m + 1, \cdots, m + n\}$: pickup point set.
- $P^- = \{m + n + 1, \cdots, m + 2n\}$: setdown point set.

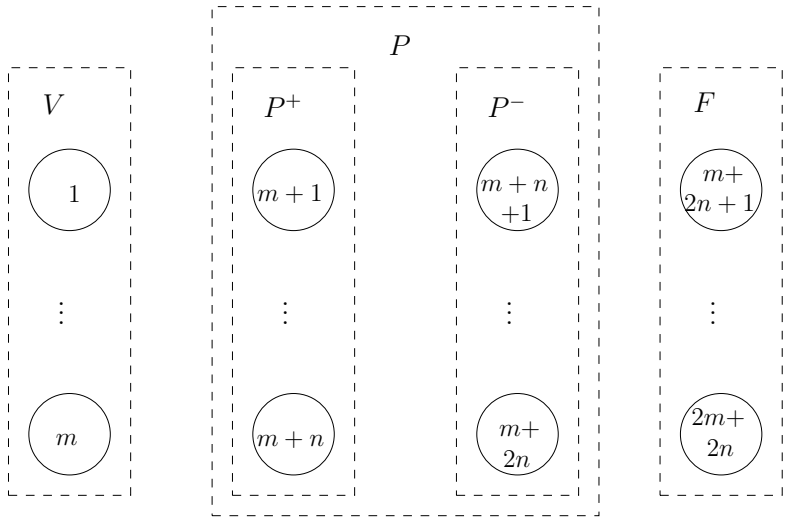

Figure 3: The definitions of vehicle and job point sets.

- $P = P^+ \cup P^-$: job (pickup and setdown) point set.
- $H^+ \subset P^+$: pickup point set of high-priority jobs.
- $N^v = P \cup \{S(v), F(v)\}$: point set including job points and initial and final points of a particular SC v.
- $[e_i, u_i]$: time window for point $i \in P \cup S \cup F$, where e_i is the earliest servicing time and u_i is the latest servicing time at point i.
- t_{ij}: travel time of a SC from point i to point j.
- A^v: set of all feasible point-links corresponding to SC v. Problem constraints (e.g., time windows) should be considered to obtain such a feasible point-links set.
- $h_{ij} \in \{0, 1\}, (i, j) \in A^v$: high priority job index. $h_{i,j} = 1$ if $i \in H^+$ and $j = n + i$, otherwise $h_{ij} = 0$.
- $\lambda_1,\ \lambda_2,\ \lambda_3$: weighting factors (in dollar per second) for SCTT, SCWT, and HJFT objectives, respectively. Such weighting parameters can be set as different combinations to meet different operational requirements. Detailed theoretical analysis and simulation verification can be found in [13].

Fig. 3 shows the parameter definitions of SC initial point set V, SC final point set F, and job point set P, including job pickup point set P^+ and job setdown point set P^-, in the ASCS problem based on PDPTW.

4.1.2. Defined variables

- $X_{ij}^v \in \{0,1\}$: binary integer decision variable, $v \in V$, $i,j \in N^v$, $i \neq j$. $X_{ij}^v = 1$ if SC v travels from point i to point j, otherwise $X_{ij}^v = 0$.
- T_i: the time at which SC services at point $i \in P \cup S \cup F$.

4.2. Model formulation

The ASCS subproblem of each planning can be modelled as the following binary integer programming formulation.

Minimise

$$\sum_{v\in V}\sum_{(i,j)\in A^v}\Big[\lambda_1 t_{ij} + \lambda_2(T_j - T_i - t_{ij}) + \lambda_3 h_{ij} T_j\Big]X_{ij}^v \tag{1}$$

Subject to

$$\sum_{v\in V}\sum_{j\in N^v} X_{ij}^v = 1,\ i \in P^+, \tag{2}$$

$$\sum_{i\in N^v} X_{ij}^v - \sum_{i\in N^v} X_{ji}^v = 0,\ j \in P,\ v \in V, \tag{3}$$

$$\sum_{j\in P^+\cup\{F(v)\}} X_{S(v),j}^v = 1,\ v \in V, \tag{4}$$

$$\sum_{i\in P^-\cup\{S(v)\}} X_{i,F(v)}^v = 1,\ v \in V, \tag{5}$$

$$\sum_{j\in P} X_{ij}^v - \sum_{j\in P} X_{j,n+i}^v = 0,\ i \in P^+,\ v \in V, \tag{6}$$

$$\sum_{j\in N^v} X_{ij}^v - X_{i,n+i}^v = 0,\ i \in P^+,\ v \in V, \tag{7}$$

$$T_i + t_{i,n+i} \leq T_{n+i},\ i \in P^+,\ v \in V, \tag{8}$$

$$X_{ij}^v(T_i + t_{i,j} - T_j) \leq 0,\ i,j \in A^v,\ v \in V, \tag{9}$$

$$e_i \leq T_i \leq u_i,\ i \in N^v,\ v \in V, \tag{10}$$

$$X_{ij}^v \in \{0,1\},\ (i,j) \in A^v,\ v \in V. \tag{11}$$

The objective function (1) is to minimise the weighted sum cost of SCTT (i.e., $\sum_{v\in V}\sum_{(i,j)\in A^v} t_{ij}X_{ij}^v$), SCWT (i.e., $\sum_{v\in V}\sum_{(i,j)\in A^v}(T_j - T_i - t_{ij})X_{ij}^v$), and HJFT (i.e., $\sum_{v\in V}\sum_{(i,j)\in A^v} h_{ij}T_jX_{ij}^v$), which in the long-term can maximise the economic benefit of container terminal operation. Constraint (2) forces one pickup point to be visited only once and by one SC only, which guarantee that all jobs are finished and each job is only done once. (3) is the flow conservation constraint, i.e., if a SC arrives at a pickup/setdown point, it should leave to other points after picking up/setting down a container. Constraint (4) forces a SC to start from its initial point. Constraint (5) forces SCs to travel to the final point after finishing the assigned jobs. Constraints (6) and (7) force the same SC to visit the pickup and setdown points of a job, and ensure the setdown point is visited immediately following the pickup point of the same job. Constraint (7) ensures that one SC can only transport one container at one time. Constraint (8) forces a visit to the pickup point before the corresponding setdown point for the same job. (9) is the time constraint in the case of SC travelling from point i to point j. (10) is the time window constraint of all SC and job points. (11) is the decision variable constraint. We see that the ASCS subproblem is very difficult to solve (especially for a large-scale problem), as integer programming problem is an NP-hard problem.

4.3. BBCG solution algorithm

To get the possible optimal schedule of SC task allocation in each planning, the exact algorithm based on branch-and-bound and column-generation developed in our previous work [12, 13] is employed to solve the multi-objective ASCS subproblem modelled as (1)-(11). The BBCG algorithm embeds the column-generation method into the branch-and-bound framework, with the solution structure shown in Fig. 4.

Specifically, according to the basic theory of Dantzig-Wolfe decomposition [24, 25], the multi-objective model (1)-(11) is decomposed into master problem and subproblem. Column generation is employed to solve the linear relaxation of the master problem [i.e., Restricted Master Problem (RMP)], while dynamic programming is employed as the subproblem solver to generate new columns for the RMP. The branch-and-bound method is used for the exploration of the integer solution for the master problem so as to get the solution of the PDPTW-BIP model for the ASCS subproblem. The main solving steps can be described as follows.

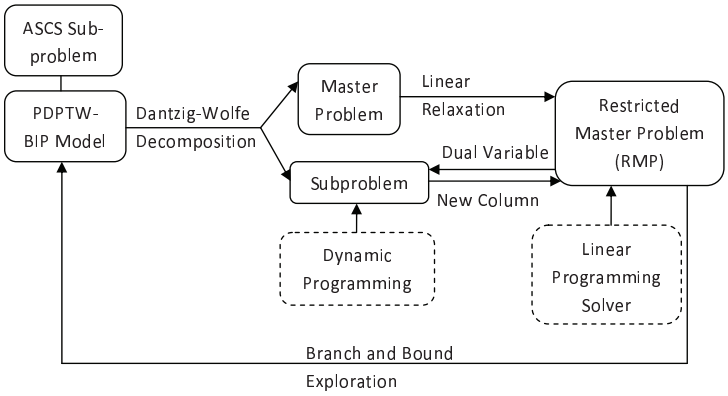

Figure 4: Solution structure of BBCG algorithm.

1) Solve RMP by column generation, and go to 5) if there is no feasible solution;
2) If the optimal solution to RMP is integer and better than the objective of the previous solution, update optimal solution and go to 5);
3) For a non-integer solution case, select one non-integer variable (closest to 0.5) and do branching;
4) Select one branch, update the searching space using the branching information, and construct a new RMP, then repeat step 1);
5) If all branches are all explored, output result and terminate the algorithm, otherwise repeat 4).

For detailed ASCS subproblem modelling and BBCG solution procedure, please refer to [12, 13].

4.4. Auction solution algorithm

In order to evaluate the performance of the two presented rescheduling policies based on different ASCS solving algorithms and to compare the solution quality with that obtained using the BBCG algorithm, an approximate algorithm, i.e., auction algorithm [26], is also developed to solve the ASCS subproblem modelled as (1)-(11). The main idea is assigning a job to the currently available SC with minimal cost introduced. Detailed steps are shown below.

1) Given the most recent available SC, calculate all the additional cost that may be introduced into the objective function (1) when assigning each job in the unassigned job list to this SC;

2) Assign the job with the minimal additional cost to the recent SC, and update the available time of the recent SC;
3) Update the available SC order list according to the new SC available time, and delete the assigned job from the unassigned job list;
4) If the unassigned job list is empty, output the result and terminate the algorithm, otherwise repeat 1).

5. Simulation and Comparative Results

In this section, the simulation and comparative results are presented to demonstrate the performance of the rescheduling polices (i.e., RNJ and RCJ polices) with different solution algorithms (i.e., BBCG algorithm and auction algorithm) for short-term and long-term task allocation of straddle carriers under the event of a new job arrival. The rescheduling policies and the solution algorithms for the ASCS problem are implemented in MATLAB 7.12.0 [27]. Computational experiments are conducted on a computing cluster computer with Linux operation system at the University of Technology, Sydney [28]. The specification of the cluster computer is 2x 3.46GHz Intel Xeon X5690 (6 Core) 6.4GT/s QPI, 96GB 1333MHz ECC DDR3-RAM.

5.1. Short-term performance testing

To test the quality and efficiency of the presented rescheduling policies for a short time horizon (i.e., in solving small-scale scheduling problem), 3 different-scale ASCS problems (called testing cases below) are tested in the simulation, with the total number of jobs being 18, 20, and 24, the total number of high-priority jobs being 6, 9, and 11, the number of new arrival jobs in each replanning being 6, 4, and 6, and hence the number of replannings being 3, 5, and 4, respectively. The number of SCs employed for the task allocation is 3. The test problems (i.e., job lists and time windows) are randomly generated based on the yard environment map of Patrick AutoStrad container terminal, mentioned in Section 2 and Figs. 1 and 2.

In each test case, we use fixed and random replanning time to test the performance of the RNJ and RCJ rescheduling policies. For the fixed one, replanning time is fixed as 300s, which means that new jobs arrive every 300s and the ASCS subproblem is solved every 300s. For the random one, replanning time is randomly generated based on uniformly-distributed function with the values in the range of [300,420]s, which means that the new job arriving interval is a random value of time between 300s to 420s, and

the ASCS subproblem is repetitively solved in different replanning intervals. Note that, the setting of new-job-arrival/replanning time in the simulation is just a representative example, which is used to show the capability of the rescheduling policies to handle different kinds of job arrival time. In real applications, the replanning time intervals are based on the arrival time of new jobs, as replanning is triggered by the event of new job arrival. Each test case with fixed/random replanning time is tested 100 times. For comparative purpose, we calculate a globally optimal solution by assuming that all jobs for each time of testing are known at the beginning of scheduling, although their arrival times are various. The globally optimal solution is obtained by solving the ASCS problem combining all jobs of all replanning using the BBCG algorithm. Note that, the objective cost used to evaluate the performance of the policies is the cost sum of SCTT, SCWT and HJFT with weighting factors in (1) setting as $\lambda_1 = \lambda_2 = \lambda_3 = 1$ dollar/s. The simulation results and comparisons between the two rescheduling policies are shown in Table 1. In each case, the ASCS subproblem is solved by the BBCG algorithm.

In Table 1, Columns "Average Computational Time (s)" show the average computational time of each time of testing. Specifically, for each time of testing, the computational time of the RNJ (or RCJ) policy is the sum of the calculation time of all replanning (including the time for ASCS subproblem updating and solving), while the computational time of the global problem is the computing time to get the optimal schedule for the ASCS problem with all jobs. From the table, we can see that to get the globally optimal solution takes much more time than the solution obtained by the RNJ (or RCJ) policy, e.g., 12.386s vs 1.2304s (or 10.146s) for the 18-job fixed-replanning-time testing case. This is because the computational complexity increases exponentially as the size of the ASCS problem increases, i.e., solving the ASCS problem with a large number of jobs needs much more time than replanning several times in terms of solving several small-size ASCS subproblems. In this sense, replanning several ASCS subproblems is much more efficient than computing the ASCS problem in one go, no matter which policy is used. Moreover, the average computational time of the RNJ policy to get all the schedules for the allocation of container transportation tasks to the three SCs is less than the average computational time of the RCJ policy, e.g., 1.1311s vs 9.4133s for the 20-job random-replanning-time test case. This is mainly because the size (number of jobs) of the ASCS subproblem for the RCJ policy in each replanning is mostly larger than the one for RNJ policy, since RCJ policy considers unexecuted jobs in additional to new arrival jobs.

Table 1: Performance of RNJ and RCJ rescheduling policies as compared with globally optimal scheduling for different-scale container-transportation task allocation using three SCs based on BBCG algorithm (each case is tested 100 times)

Total Number of Jobs (Number of High Priority Jobs)	Number of Replanning (Interval)	Average Computational Time (s)			Number of Tests RCJ= Global	Objective Cost Comparison		
		Global	RNJ	RCJ		RNJ/ Global	RCJ/ Global	RCJ/ RNJ
18 (6)	3 (Fixed)	12.386	1.2304	10.146	100	116.50%	100.00%	85.83%
	3 (Random)	12.448	1.2982	10.651	100	116.29%	100.00%	85.99%
20 (9)	5 (Fixed)	24.606	1.1293	13.301	85	142.67%	101.56%	71.19%
	5 (Random)	22.580	1.1311	9.4133	41	143.40%	105.67%	73.69%
24 (11)	4 (Fixed)	56.389	1.8418	38.617	85	132.98%	100.14%	75.30%
	4 (Random)	53.702	1.8545	30.601	49	133.43%	102.99%	77.19%

Moreover, Column "Number of Tests RCJ=Global" of Table 1 shows in how many of the 100 tests the RCJ policy gets the schedules with the same total cost as the cost of the globally optimal solution. From the column, we can see that the RCJ policy can achieve optimality in many tests and can get the optimal solution in all tests in the 18-job test cases. By contrast, from the simulation results, we find that the RNJ policy does not achieve optimality in all the tests and that the total costs of schedules from the RCJ policy are all better than the ones from the RNJ policy in each test. Thus, we can say that the RCJ policy is a better policy in view of the number of testings with better qualities, as compared with the RNJ policy.

Furthermore, the Columns "Objective Cost Comparison" of Table 1 show the comparisons of the cost values of the objective function between the globally optimal solution, the solution from RNJ policy, and the solution from RCJ policy. From Columns "RNJ/Global" (the cost of RNJ divided by the optimal) and "RCJ/Global" (the cost of RCJ divided by the optimal), we see that the RCJ policy has better performance since it can get the schedule with nearly the same cost as the globally optimal solution with less than 6%

Table 2: Performance of RNJ and RCJ rescheduling policies for long-term SC task allocation (each case tests 10 times) using BBCG algorithm and auction algorithm

Numbers of Jobs - High Priority Jobs - SCs	Number of Replanning (Replanning Interval)	Average Computational Time (s) of Each Planning				Objective Cost Comparison RCJ/RNJ		Objective Cost Comparison Auction/BBCG	
		RNJ BBCG	RNJ Auction	RCJ BBCG	RCJ Auction	BBCG	Auction	RNJ	RCJ
40 - 18 - 3	5 (Fixed)	1.1199	0.0085	92.960	0.0114	72.40%	91.10%	152.06%	191.34%
	5 (Random)	1.1050	0.0086	73.724	0.0116	73.38%	90.75%	152.57%	188.66%
48 - 20 - 4	6 (Fixed)	1.3427	0.0088	245.20	0.0108	75.07%	90.78%	170.38%	206.03%
	6 (Random)	1.3048	0.0085	243.03	0.0104	76.13%	89.97%	171.82%	203.02%
60 - 27 - 6	6 (Fixed)	4.3606	0.0123	1688.9	0.0141	83.59%	91.96%	185.70%	204.28%
	6 (Random)	4.1508	0.0121	1862.7	0.0128	82.54%	91.70%	186.91%	207.64%

bigger than the optimality, using less computational time than the global one as shown in the Columns "Average Computational Time (s)". In addition, from the last column [i.e., Column "RCJ/RNJ" (which means the objective cost of RCJ divided the one of RNJ)], we can see that the average total costs of the schedules from the RCJ policy can be reduced by maximally 26% and minimally 14% as compared with the RNJ policy. In this sense, the RCJ policy is a better rescheduling policy in terms of achieving small cost.

Above all, the two presented rescheduling policies (i.e., RNJ and RCJ policies) can effectively solve the task allocation problem of autonomous SCs with less computational time than the one of solving the whole ASCS problem with all the jobs. Comparison of the two rescheduling policies reveals the RCJ policy to be a better policy as it can achieve a higher quality of solution due to its considering both information from unexecuted jobs and from new jobs. Although the RCJ policy may take more computational time than the RNJ policy to get the solution, it is still feasible for short-term practical applications.

5.2. Long-term performance testing

For long-term scheduling, since the number of jobs to be scheduled would be very large and many jobs are unknown at the beginning of the scheduling, it is impossible to solve the global ASCS problem in the way (combining all jobs together and solving the problem in one go optimally) mentioned in the previous subsection. In this subsection, the RNJ and RCJ rescheduling policies are tested for the long-term task allocation of autonomous straddle carriers under the uncertainty of new job arrival.

Different-scale testing cases are used in the simulation, with the number of new jobs arriving in each replanning varying from 8 to 12, the number of replannings being 5 or 6, and the number of SCs being 3 to 6. The total number of jobs changes from 40 to 60, with the total number of high-priority jobs varying from 18 to 27. Details are shown in the first two columns of Table 2. The test problems (i.e., job lists and time windows) are also randomly generated based on the yard environment map. Fixed replanning time (300s) and random replanning time (uniformly-distributed in [300,420]s) are also used in the testing. Each test case is tested 10 times. The objective cost is also calculated based on Equation (1) with $\lambda_1 = \lambda_2 = \lambda_3 = 1$ dollar/s. For comparison purpose, both the BBCG algorithm and the auction algorithm are implemented to solve the ASCS subproblem so as to evaluate the presented rescheduling policies. The simulation results and comparisons are shown in Table 2.

For the efficiency of the two policies, Columns "Average Computational Time (s) of Each Planning" of Table 2 show that the average computational time of each planning via RNJ policy is much less than the one using the RCJ policy when using the BBCG algorithm. RNJ policy only takes less than 5 seconds to get the solution of each planning in all testings, while the RCJ policy takes much more time than that, e.g., about 30 minutes for the 60-job cases. On the other hand, when using the auction algorithm to solve the ASCS subproblem in each planning, the average computational times of RNJ and RCJ policies decrease greatly to almost the same length of time, i.e., about 0.01s. This shows that both the rescheduling policies can be efficient enough for real-time practical operation of straddle carriers if using an efficient solution algorithm for the ASCS subproblem in each planning.

For the quality of the two policies, the total objective cost of the schedules from the RCJ policy is much smaller than from the RNJ policy, with the cost value improving more than 16 percent using the BBCG algorithm

and 8 percent using the auction algorithm, as shown in the Columns "Objective Cost Comparison RCJ/RNJ" of Table 2. By considering the efficiency analysis together, using the auction algorithm, the RCJ policy can get better solution with a very similar computational time as compared with the RNJ policy. As compared with the BBCG algorithm, the auction algorithm takes much less time to get the solution in each planning, but the quality of the solution is not as good as the one from the BBCG algorithm. The objective cost of the solution from the auction algorithm is much bigger than the one from the BBCG algorithm as shown in the Columns "Objective Cost Comparison BBCG/Auction" of Table 2. For example, in a 48-job case, the cost of the auction solution is more than 1.7 times of the BBCG solution when using RNJ policy, while it is more than 2 times when using the RCJ policy. This shows that the simple auction algorithm is still not good enough to be employed in the rescheduling policies to achieve a high quality solution as it is still a bit far away from the optimality.

In summary, the simulation results show the short-term and long-term performance of the RNJ and RCJ rescheduling policies for large-scale task allocation of autonomous straddle carries under the uncertainty of new job arrival. The test results demonstrate the superiority of the RCJ rescheduling policy in terms of solution quality compared with the RNJ policy. From the results, we can also see that auction algorithm can be an efficient algorithm for the ASCS subproblem solving in each planning to meet practical real-time long-term operation requirement. However, the quality of the solutions from auction algorithm with the RCJ policy is still not good enough as compared with the ones from the BBCG algorithm. This shows a need for further investigations for a more efficient algorithm for the ASCS subproblem solving in each replanning to improve the solution quality for long-term operation of straddle carriers at automated container terminals.

6. Conclusions

This paper presented and compared two rescheduling policies, i.e., RNJ policy and RCJ policy, to solve large-scale scheduling problems and handle unexpected events (especially new job arrival) in the task allocation of autonomous SCs at automated container terminals. An optimisation model based on PDPTW, the exact BBCG algorithm or the auction algorithm have been employed to get the schedule for the ASCS subproblem in each planning. The sum of the costs of SC travelling, SC waiting, and delay of high-priority

jobs have been used as the objective function for the solving of the ASCS subproblem and for the evaluation of the performance of the rescheduling policies. Computational and comparative simulations have demonstrated the performance of the presented policies, the superiority of the RCJ policy to achieve small objective cost, and the need for an improved solution algorithm for the ASCS subproblem in each planning. Future research includes improvement of the rescheduling policies to handle other unexpected events (e.g., SC breakdown and delay), strategies for the selection of the replanning time for rescheduling policy, and improvement of the ASCS subproblem solving algorithm.

Acknowledgments

This work is supported by the ARC Linkage Grant (LP0882745), the Patrick Stevedores Holdings and the University of Technology, Sydney, Australia.

References

[1] S. Hartmann, A general framework for scheduling equipment and manpower at container terminals, in: Container Terminals and Automated Transport Systems, Springer, Berlin, 2005, Part 1, pp. 207-230.

[2] J. Zeng, W.-J. Hsu, Conflict-free container routing in mesh yard layouts, Robotics and Autonomous Systems, 56 (5) (2008) 451-460.

[3] I.F.A. Vis, R. de Koster, Transshipment of containers at a container terminal: an overview, European Journal of Operational Research, 147 (2003) 1-16.

[4] S. Hoshino, J. Ota, A. Shinozaki, H. Hashimoto, Hybrid design methodology and cost-effectiveness evaluation of AGV transportation systems, IEEE Transactions on Automation Science and Engineering, 4 (3) (2007) 360-372.

[5] M. Grunow, H.-O. Günther, M. Lehmann, Dispatching multi-load AGVs in highly automated seaport container terminals, OR Spectrum, 26 (2004) 211-235.

[6] E.K. Xidias, P.N. Azariadis, Mission design for a group of autonomous guided vehicles, Robotics and Autonomous Systems, 59 (1) (2011) 34-43.

[7] V.D. Nguyen, K.H. Kim, A dispatching method for automated lifting vehicles in automated port container terminals, Computers & Industrial Engineering, 56 (3) (2009) 1002-1020.

[8] D. Liu, A.K. Kulatunga, Simultaneous planning and scheduling for multi-autonomous vehicles, in: K.P. Dahal, K.C. Tan, P.I. Cowling (Eds.), Evolutionary Scheduling, Springer-Verlag, 2007, pp. 437-464.

[9] S. Yuan, H. Lau, D. Liu, S. Huang, G. Dissanayake, D. Pagac, T. Pratley, Simultaneous dynamic scheduling and collision-free path planning for multiple autonomous vehicles, in: Proceedings of the IEEE International Conference on Information and Automation (ICIA), Zhuhai/Macau, China, June 2009, pp. 522-527.

[10] D. Liu, X. Wu, A.K. Kulatunga, G. Dissanayake, Motion coordination of multiple autonomous vehicles in dynamic and strictly constrained environments, in: Proceedings of the IEEE International Conference on Cybernetics and Intelligent Systems (CIS), Bangkok, Thailand, June 2006, pp. 204-209.

[11] S. Yuan, B.T. Skinner, S. Huang, D. Liu, G. Dissanayake, H. Lau, D. Pagac, A job grouping approach for planning container transfers at automated seaport container terminals, Advanced Engineering Informatics, 25 (3) (2011) 413-426.

[12] B. Cai, S. Huang, D. Liu, S. Yuan, G. Dissanayake, H. Lau, D. Pagac, Optimisation model and exact algorithm for autonomous straddle carrier scheduling at automated container terminals, in: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), San Francisco, USA, Sep. 2011, pp. 3686-3693.

[13] B. Cai, S. Huang, D. Liu, S. Yuan, G. Dissanayake, H. Lau, D. Pagac, Multi-objective optimisation for autonomous straddle carrier scheduling at automated container terminals, IEEE Transactions on Automation Science and Engineering, 10 (3) (2013) 711-725.

[14] S. Yuan, B.T. Skinner, S. Huang, D. Liu, G. Dissanayake, H. Lau, D. Pagac, T. Pratley, Mathematical modelling of container transfers for a

fleet of autonomous straddle carriers, in: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), Anchorage, Alaska, USA, May 2010, pp. 1261-1266.

[15] G. Nelmes, Container port automation, in: Springer Tracts in Advanced Robotics: Field and Service Robotics, vol. 25, Springer, Berlin, 2006, pp. 3-8.

[16] H. Durrant-Whyte, D. Pagac, B. Rogers, M. Stevens, G. Nelmes, Field and service applications - an autonomous straddle carrier for movement of shipping containers - from research to operational autonomous systems, IEEE Robotics & Automation Magazine, 14 (3) (2007) 14-23.

[17] G. Paul, D. Liu, Replanning of multiple autonomous vehicles in material handling, in: Proceedings of IEEE International Conference on Robotics, Automation & Mechatronics, Thailand, June 2006, pp. 231-236.

[18] D. Ouelhadj, S. Petrovic, A survey of dynamic scheduling in manufacturing systems, Journal of Scheduling, 12 (4) (2009) 417-431.

[19] M. Merdan, T. Moser, P. Vrba, S. Biffl, Investigating the robustness of re-scheduling policies with multi-agent system simulation, International Journal of Advanced Manufacturing Technology, 55 (2011) 355-367.

[20] X.F. Yin, L.P. Khoo, C.-H. Chen, A distributed agent system for port planning and scheduling, Advanced Engineering Informatics, 25 (3) (2011) 403-412.

[21] T. Thurston, H. Hu, Distributed agent architecture for port automation, in: Proceedings of the 26th Annual International Computer Software and Applications Conference (COMPSAC), Oxford, England, 26-29th Aug., 2002.

[22] B.P. Gerkey, M.J. Mataric, A formal analysis and taxonomy of task allocation in multi-robot systems, International Journal of Robotics Research, 23 (9) (2004) 939-954.

[23] Y. Dumas, J. Desrosiers, F. Soumis, The pickup and delivery problem with time windows, European Journal of Operational Research, 54 (1991) 7-22.

[24] L.A. Wolsey, Integer Programming, John Wiley & Sons, New York, NY, 1998.

[25] D.-S. Chen, R.G. Batson, Y. Dang, Applied Integer Programming: Modeling and Simulation, John Wiley & Sons, Hoboken, NJ, 2010.

[26] B.P. Gerkey, M.J. Matarić, Sold!: Auction methods for multirobot coordination, IEEE Transactions on Robotics and Automation, 18 (5) (2002) 758-768.

[27] The MathWorks Inc. MATLAB 7.12.0 (R2011a), 2011.

[28] Faculty of Engineering and IT, University of Technology, Sydney, The High Performance Computing Linux Cluster, available at https://cluster.eng.uts.edu.au/.

Binghuang Cai was born in Chenghai, Guangdong, China, in 1981. He received the B.S. degree in electronic information engineering from Shantou University, Shantou, China, in 2004, the M.S. degree in signal and information processing from Shantou University, Shantou, China, in 2007, and the Ph.D. degree in communication and information systems from Sun Yat-sen University, Guangzhou, China, in 2010. He has been a Research Fellow with the Centre for Autonomous Systems, University of Technology, Sydney, Australia, from September 2010 to May 2012. He is currently a Postdoctoral Associate at the University of Pittsburgh. His research interests include robotics, optimisation, neural networks, and image/signal processing. His current research focuses on coordination and planning of autonomous robots.

Shoudong Huang received the Bachelors' and Masters' degrees in mathematics and the Ph.D. degree in automatic control from Northeastern University, Shenyang, China, in 1987, 1990, and 1998, respectively. He is currently an Associate Professor at the Centre for Autonomous Systems, Faculty of Engineering and Information Technology, University of Technology, Sydney (UTS), Australia. His research interests include nonlinear control systems and mobile robots simultaneous localisation and mapping (SLAM), exploration, navigation, and path planning.

Dikai Liu received the B.E. degree in mechanical engineering, the M.E. degree in mechatronics, and the Ph.D. degree in dynamics and control from the Wuhan University of Technology, Wuhan, China, in 1986, 1991, and 1997, respectively. He is currently a Professor of Mechanical and Mechatronic Engineering at the Centre for Autonomous Systems, Faculty of Engineering and Information Technology, University of Technology, Sydney (UTS), Australia. His research interests include novel methods and algorithms for intelligent machines, and intelligent robotic systems including autonomous robots for complex infrastructure maintenance, robot teams for material handling, assistive robots for assisted care, and bio-inspired robots.

Gamini Dissanayake received the graduate degree in mechanical / production engineering from the University of Peradeniya, Peradeniya, Sri Lanka, and the M.Sc. degree in machine tool technology and the Ph.D. degree in mechanical engineering (Robotics) from the University of Birmingham, Birmingham, U.K., in 1981 and 1985, respectively. He is currently the James N Kirby Professor of Mechanical and Mechatronic Engineering at the University of Technology, Sydney (UTS), Australia. He leads the Centre for Autonomous Systems at UTS. His current research interests include localisation and map building for mobile robots, navigation systems, dynamics and control of mechanical systems, cargo handling, optimisation, and path planning.