Scheduling Twin Robots on a Line

Güneş Erdoğan[1], Maria Battarra[2], and Gilbert Laporte[3]

(1) *School of Management, University of Southampton*,
Highfield, Southampton, SO17 1BJ, United Kingdom
G.Erdogan@soton.ac.uk
(2) *School of Mathematics, University of Southampton*,
Highfield, Southampton, SO17 1BJ, United Kingdom
M.Battarra@soton.ac.uk
(3) *Canada Research Chair in Distribution Management*,
HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine,
Montreal, Canada H3T 2A7
gilbert.laporte@cirrelt.ca

September 5, 2013

Abstract

This paper introduces the *Twin Robots Scheduling Problem* (TRSP), in which two robots positioned at the opposite ends of a rail are required to deliver items to positions along the rail, and the objective is to minimize the makespan. A proof of $\mathcal{NP}$-hardness of the TRSP is presented, along with exact and heuristic algorithms. Computational results on challenging instances are provided.

Keywords: scheduling, robots, exact algorithms, approximation algorithms, Gantt chart.

1 Introduction

We study the problem of scheduling two robots, denoted as the *black robot* and the *white robot*. Each robot is assigned to a depot located at one of the ends of a rail; the depot of the black robot is called the *black depot* and that of the white robot is the *white depot*. The robots are responsible for a number of *tasks*, each of which consists of picking up an item at its depot, delivering the item to a predefined location along the rail, and returning to its depot. The objective is to minimize the makespan, while avoiding collisions between robots. To the best of our knowledge, this problem has not yet been studied. We call it the Twin Robots Scheduling Problem (TRSP).

An instance of TRSP has a trivial solution if the tasks of the black and white robots never obstruct each other. On the other hand, if the locations of the tasks are intertwined (i.e., the white robot has to cross over at least one item of the black robot and vice versa),

the schedule of the robots have to be synchronized and the optimal solution may include idle times.

The TRSP finds applications in the robotic industry where gantry robots are employed in the context of pick-and-place, assembly, and palletising or depalletising operations (Ranky 2003). A real-world example of the TRSP is depicted in Figure 1, where twin robots mounted on a rail are used to palletize boxes (All Glass s.r.l. 2013). Note that rail mounted gantry cranes are also employed in the loading and unloading operations of containers in maritime terminals (Gharehgozli et al. 2013). Similar scheduling issues arise in the context of automated parking garages (Mathijssen and Pretorius 2007), automated libraries (Dimitri et al. 2000) and narrow-aisle picking and retrieving operations (Lee et al. 1996, Dotoli and Fanti 2005), where the TRSP naturally extends to a multidimensional problem (i.e., robots are working on shelves or piles). As far as we are aware, the novel aspects of the TRSP are that two robots operate simultaneously, the operations are performed on a rail, and the tasks originate at a distinct depot for each robot.

Figure 1: Palletizing boxes with twin robots on a rail

An illustration of the problem is provided in Figure 2, where the white robot performs the task at a distance 4 from its depot, the black robot performs the task at a distance 4 from its depot. Note that these two tasks do not obstruct each other.

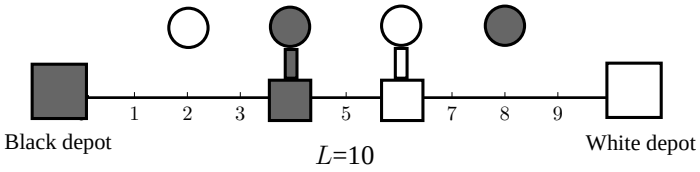

Figure 2: Illustration of TRSP.

The collision constraint is well known within the crane scheduling literature, and is referred to as a "spatial constraint" (Lim et al. 2004), "inter-crane interference" (Ng 2005), or more commonly "non-crossing" constraint (Zhu and Lim 2006, Lim et al. 2007, Meisel 2011, Boysen et al. 2012). However, these problems involve many other features such as workload allocation among the cranes, release times for the jobs, and sets of cranes that do not interact. We believe that the results of this study can be used for solving subproblems arising in some crane scheduling problems.

Robotic cells have been widely studied (we refer interested readers to Wilhelm 1987 and Dawande et al. 2005), but most of the studies refer to flow shop architectures (Chen and Su 1995) where a product has to be processed by a sequence of machines, and robots are required to transfer the product among work stations. Problems with a single or a *double-grip* gantry robot (Su and Chen 1996) have been studied in this context. Problems in which a single robot performs operations on a line have also been investigated (Dror et al. 1991, Stulman 1989), but we have not encountered any problem presenting the same characteristics as the TRSP.

We now proceed with a formal description of the problem. The black robot must perform the set of tasks $B = \{1, \ldots, m\}$, whereas the white robot is assigned the tasks $W = \{m+1, \ldots, n\}$. We assume that the positions of the tasks on the rail, are discrete, the two robots have the same speed, and the pickup and delivery times are negligible. Each task $i \in B$ is located at a distance b_i from the black depot, and each task $j \in W$ is located at a distance w_j from the white depot. The distance between the two depots is L. Without loss of generality, we assume that the tasks are indexed with respect to decreasing distances from their depot, i.e. $b_i > b_{i+1}$ and $w_j > w_{j+1}$. If this assumption is violated, the task lists may be sorted in $O(n+L)$ time using the *counting sort* algorithm, or $O(n \log n)$ time using the *heapsort* algorithm. The two robots are also expected to respect a security distance of one unit from each other to avoid collisions. Finally, no more than one task is associated with a position on the rail.

The remainder of this paper is organized as follows. In the following section, we derive some properties of the problem and of its solution, and we prove that the TRSP is $\mathcal{NP}$-hard. In Section 3, we present two alternative integer linear programming formulations

for the TRSP as well as a branch-and-bound algorithm based on an additive lower bound. In Section 4, we present two constructive heuristics, one of which has a worst-case performance guarantee of 3/2. In Section 5, we describe a set of benchmark instances and we compare the performance of the three exact algorithms and of two heuristic algorithms. Conclusions follow in Section 6.

2 Properties

Before proving that the TRSP is $\mathcal{NP}$-hard, we introduce an effective way of representing TRSP solutions and we show how to reduce the solution space by removing equivalent solutions. We will employ a *space-time Gantt chart*, in which the x-axis represents the time line and the y-axis represents the position of the robots on the rail. The black depot is assumed to be located at $(0, 0)$ and the white depot at $(0, L)$. A solution is depicted in Figure 3(a). Collisions between the robots can be detected by checking whether the vertical distance between the two lines is less than 1.

The first result we present allows us to restrict the search space to solutions in which both robots can only wait at their respective depots and never at intermediate positions on the rail, because a solution of type 3(a) has an equivalent solution of type 3(b).

Proposition 1 *Any optimal solution for the TRSP can be converted into an alternative optimal solution in which both robots only wait at their depot.*

Proof. Consider an optimal solution in which one of the robots does not wait at its depot. If the wait occurs after the delivery location or at the delivery location, it can be eliminated since there is no obstruction on the way back to the depot. If the wait is before the delivery, the departure of the robot can be postponed to guarantee that there will be no waiting, and the completion time of the task will not be affected. An example of such a rescheduling is depicted in Figure 3(b). Applying the rescheduling to all tasks with a wait at an intermediate location along the rail results in an optimal solution in which the robots only wait at their respective depots. □

In order to prove that the TRSP is $\mathcal{NP}$-hard we first introduce another combinatorial problem and prove that it is also $\mathcal{NP}$-hard. The *PARTITION with Distinct Weights* (PARTITION-DW) is a search problem that returns an affirmative answer when, given a set of positive integers $b_i, i \in S = \{1, \ldots, n\}$ and a positive integer K, there exists a partition $P \subset S : |\sum_{i \in P} b_i - \sum_{j \in S \setminus P} b_j| \leq K$.

Proposition 2 *The PARTITION-DW is $\mathcal{NP}$-hard.*

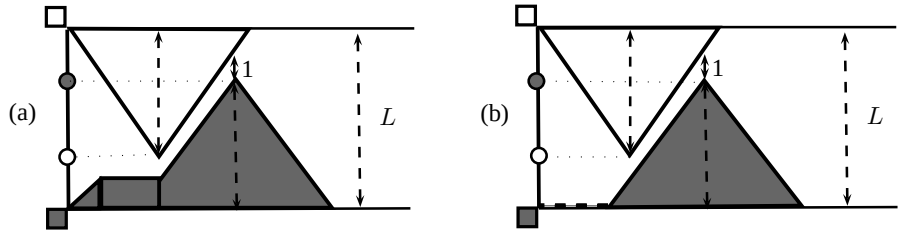

Figure 3: Waiting time at the depot.

Proof. We use a reduction from PARTITION. Given an instance of PARTITION with $a_i, i \in S = \{1, \ldots, n\}$, we define $M = n(n+1)$, $b_i = (a_i \times M) + i$, for all $i = \{1, \ldots, n\}$, and finally, $K = n(n+1)/2$. The PARTITION-DW instance has a feasible solution if and only if the corresponding PARTITION instance has a feasible solution.

A PARTITION solution $P \subset S$ is feasible if $\sum_{i \in P} a_i = \sum_{j \in S \setminus P} a_j$. Therefore the corresponding PARTITION-DW solution is feasible because $|\sum_{i \in P} b_i - \sum_{j \in S \setminus P} b_j| < n(n+1)/2 = K$. Assume that a PARTITION-DW solution $P \subset S$ is feasible, i.e. $|\sum_{i \in P} b_i - \sum_{j \in S \setminus P} b_j| \leq K$ but $\sum_{i \in P} a_i \neq \sum_{j \in S \setminus P} a_j$. Without loss of generality, we assume that $\sum_{i \in P} a_i > \sum_{j \in S \setminus P} a_j$. The minimum value of $\sum_{i \in P} b_i - \sum_{j \in S \setminus P} b_j$ is attained when $\sum_{i \in P} a_i = \sum_{j \in S \setminus P} a_j + 1$ and $P = \{1\}$. In this case, $\sum_{i \in P} b_i - \sum_{j \in S \setminus P} b_j = M \sum_{i \in P} a_i + 1 - (M \sum_{j \in S \setminus P} a_j + K - 1) = M - K + 2 = K + 2 > K$, resulting in a contradiction.

□

Proposition 3 *TRSP is $\mathcal{NP}$-hard.*

Proof. We use a reduction from PARTITION-DW. Given a PARTITION-DW instance, we construct a TRSP instance in which the white robot has to perform a task i at a distance $w_i = 2a_i$ from the white depot for each $a_i, i \in S$ and the black robot is assigned a single task at a distance $b_1 = 2\sum_{i \in S} a_i = 2A$ from the black depot. The value of L is set to $2A + 1$. The instance of PARTITION-DW has a feasible solution if and only if the corresponding instance of TRSP has a makespan of $4A + 2K$ or less. An example of transformation of an instance of PARTITION-DW with data $\{1, 2, 4\}$ and $K = 1$ is depicted in Figures 4 and 5.

Necessity: Assume that a partition exists such that $|\sum_{i \in P} a_i - \sum_{j \in S \setminus P} a_j| \leq K$. Without loss of generality, assume that $\sum_{i \in P} a_i \geq \sum_{j \in S \setminus P} a_j$, and hence $\sum_{i \in P} a_i \in [A/2, A/2 + K/2]$. If the tasks in the first partition are assigned to be completed before the pickup for the black robot, they will shift the makespan of this robot by $4\sum_{i \in P} a_i - 2A \leq 2A + 2K - 2A = 2K$ units, which will yield a makespan of $4A + 2K$. If they are

assigned to be completed after the pickup of the black robot, they will yield a makespan of $2A + 4\sum_{i\in P} a_i \leq 2A + 2A + 2K = 4A + 2K$.

Sufficiency: Assume that there exists a schedule with a makespan of $4A + 2K$ or less. Denote the set of tasks for the white robots scheduled before the pickup of the black robot as P. Then the makespan is computed as $\max\{4\sum_{i\in P} a_i - 2A + 4A, 2A + 4\sum_{j\in S\setminus P} a_j\}$, where the first term is the makespan determined by the black robot due to the shift of its only task by tasks of the white robot in the first partition, and the second term is the white robot's makespan resulting from tasks in the second partition.

If the first term is the maximum, then $4\sum_{i\in P} a_i - 2A + 4A \geq 2A + 4\sum_{j\in S\setminus P} a_j$, yielding $\sum_{i\in P} a_i \geq \sum_{j\in S\setminus P} a_j$, and hence $\sum_{i\in P} a_i \geq A/2$. We also have $4\sum_{i\in P} a_i - 2A + 4A \leq 4A + 2K$, reorganizing the terms of which gives $\sum_{i\in P} a_i \leq A/2 + K/2$. Then, $\sum_{i\in P} a_i \in [A/2, A/2 + K/2]$, which yields a feasible solution for PARTITION-DW.

If the second term is the maximum, then $4\sum_{i\in P} a_i - 2A + 4A \leq 2A + 4\sum_{j\in S\setminus P} a_j$, yielding $\sum_{i\in P} a_i \leq \sum_{j\in S\setminus P} a_j$, and hence $\sum_{j\in S\setminus P} a_j \geq A/2$. We also have $2A + 4\sum_{j\in S\setminus P} a_j \leq 4A + 2K$, which yields $\sum_{j\in S\setminus P} a_j \leq A/2 + K/2$ after reorganization of the terms. Then, $\sum_{j\in S\setminus P} a_j \in [A/2, A/2 + K/2]$, which yields a feasible solution for PARTITION-DW. □

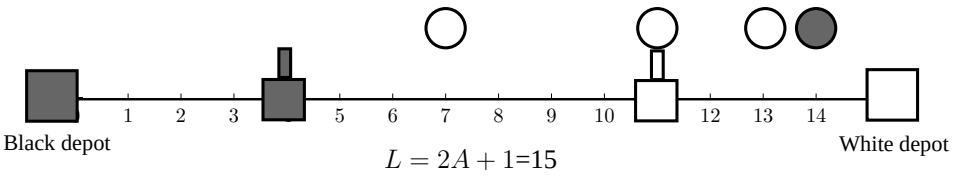

Figure 4: An instance of PARTITION-DW with data $\{1, 2, 4\}$, converted into an instance of TRSP.

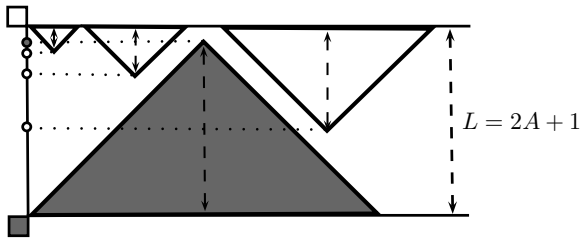

Figure 5: Space-time Gantt chart of the converted instance.

We can also prove the following more general result:

Proposition 4 *The generalization of TRSP which allows multiple items to be located at each location is $\mathcal{NP}$-hard in the strong sense.*

Proof. The proof follows from the fact that TRSP is a special case of the mentioned problem in which all data are restricted to be less than or equal to L. □

Although the TRSP is $\mathcal{NP}$-hard, it admits a straightforward lower bound as well as a trivial solution, which in conjunction yield a general performance guarantee.

Proposition 5 *Any heuristic for the TRSP which does not involve both robots waiting at intersecting time slots have a performance guarantee of 2.*

Proof. Given an instance of the TRSP with $w_i, i \in W$ and $b_i, i \in B$, a lower bound on the optimal solution value is $\max\{\sum_{i\in W} 2w_i, \sum_{i\in B} 2b_i\}$. In the worst case, all tasks of one robot can be performed in succession, yielding an upper bound of $\sum_{i\in W} 2w_i + \sum_{i\in B} 2b_i$. The ratio of the upper bound to the lower bound is

$$\bar{\rho} = (\sum_{i\in W} 2w_i + \sum_{i\in B} 2b_i)/\max\{\sum_{i\in W} 2w_i, \sum_{i\in B} 2b_i\} \tag{1}$$

$$\leq 2\max\{\sum_{i\in W} 2w_i, \sum_{i\in B} 2b_i\}/\max\{\sum_{i\in W} 2w_i, \sum_{i\in B} 2b_i\} = 2. \tag{2}$$

□

On the other hand, it can be proved that optimal solutions may require both robots to wait, therefore it is not possible to restrict the search space to solutions in which one of the robots never waits.

Proposition 6 *An optimal solution may involve both robots waiting at non-intersecting time slots.*

Proof. Consider the instance in Figure 6, where the white robot is assigned to 2 tasks at a distance of 10 and 11 from the depot, and black robot 3 tasks at distances 6, 7, and 8. The total distance between the robots is equal to 12. Note that the white robot has a workload of 20 + 22 = 42, equal to that of the black robot (12 + 14 + 16 = 42). The construction of the instance forces two conflicts between the robots, and allowing each robot to wait once results in an optimal solution, rather than having the same robot wait in both conflicts. The space-time Gantt chart for an optimal solution is given in Figure 6. The optimal solution value is 46. □

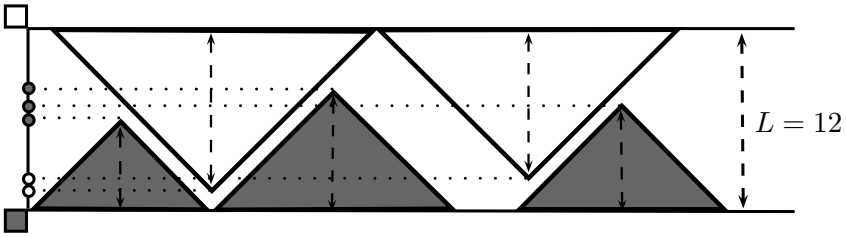

Figure 6: Space-time Gantt chart for the example in Proposition 6.

3 Exact algorithms

We now present two integer programming formulations and a branch-and-bound algorithm based on an additive lower bound. The first formulation is based on variables with time indices, whereas the second formulation uses precedence variables. The two formulations are solved by CPLEX 12.5 using as an initial upper bound the solution provided by the First Fit Decreasing heuristic of Section 4.2. This solution is also used to initialize the branch-and-bound algorithm.

3.1 Formulation with time indices

Our first formulation is based on two sets of binary variables, which are of pseudo-polynomial number, as well as an auxiliary variable u equal tothe makespan. Let $x_i^t \in \{0,1\}$ be equal to 1 if and only if the black robot starts performing task $i \in B$ at time t, and let $y_j^t \in \{0,1\}$ be equal to 1 if and only if the white robot starts performing task $j \in W$ at time t. Define I_{ij}^t as the set of incompatible time indices, such as the instants in which task j cannot start if task i started at time t. These sets can easily be defined a priori. The resulting formulation is:

$$(TRSP\ 1) \qquad \text{minimize } u \tag{3}$$

$$\text{subject to} \quad t\,x_i^t + 2b_i \le u \qquad i \in B, t \in \{0,\dots,T\} \tag{4}$$

$$t\,y_j^t + 2w_j \le u \qquad j \in W, t \in \{0,\dots,T\} \tag{5}$$

$$\sum_{t=0}^{T-2b_i} x_i^t = 1 \qquad i \in B \tag{6}$$

$$\sum_{t=0}^{T-2w_j} y_j^t = 1 \qquad j \in W \tag{7}$$

$$x_i^t + \sum_{q \in I_{ij}^t} x_k^q \leq 1 \qquad i,k \in B : i \neq k; t \in \{0,\ldots,T\} \tag{8}$$

$$y_j^t + \sum_{q \in I_{ij}^t} y_k^q \leq 1 \qquad j,k \in W : j \neq k; t \in \{0,\ldots,T\} \tag{9}$$

$$x_i^t + \sum_{q \in I_{ij}^t} y_j^q \leq 1 \qquad i \in B, j \in W, t \in \{0,\ldots,T\} \tag{10}$$

$$y_j^t + \sum_{q \in I_{ij}^t} x_i^q \leq 1 \qquad i \in B, j \in W, t \in \{0,\ldots,T\} \tag{11}$$

$$x_i^t \in \{0,1\} \qquad i \in B, t = \{0,\ldots,T\} \tag{12}$$

$$y_j^t \in \{0,1\} \qquad j \in W, t = \{0,\ldots,T\}. \tag{13}$$

The objective function minimizes the makespan. Constraints (4) and (5) impose that any black and white task must finish before the makespan is attained. Constraints (6) and (7) state that each task has to be performed once. Constraints (8)–(11) forbid incompatible pairs of tasks. Finally, (12) and (13) are the integrality constraints.

3.2 Formulation with precedence variables

A more compact formulation is obtained by using precedence variables. Let binary variable $z_{ij} \in \{0,1\}$ be equal to 1 if and only if task $i \in B \cup W$ starts before task $j \in B \cup W$, and let variable s_i represent the time at which task $i \in B \cup W$ is reached by the robot (the peak of the triangle associated with task i in the space-time Gantt chart). We also define M as a large positive constant and t_{ij}, $i \in B, j \in W$, as the minimum time between the white robot reaches location i and the black robot reaches location j in order to avoid a collision, or vice versa. The value of t_{ij} is zero whenever the two tasks are not conflicting, i.e. $b_i + w_j \leq L - 1$. If the tasks are conflicting, then $t_{ij} = |b_i - w_j + L - 1|$. The TRSP can be formulated as follows:

$$(TRSP\ 2) \qquad \text{minimize } u \tag{14}$$

$$\text{subject to} \quad s_i + b_i \leq u \qquad i \in B \tag{15}$$

$$s_j + w_j \leq u \qquad j \in W \tag{16}$$

$$s_i \geq b_i \qquad i \in B \tag{17}$$

$$s_j \geq w_j \qquad j \in W \tag{18}$$

$$z_{ij} + z_{ji} = 1 \qquad i,j \in B \cup W \tag{19}$$

$$s_i - s_j \geq (w_i + w_j) z_{ij} - M z_{ji} \qquad i,j \in W : i \neq j \tag{20}$$

$$s_i - s_j \geq (b_i + b_j)z_{ij} - Mz_{ji} \qquad i,j \in B : i \neq j \qquad (21)$$

$$s_i - s_j \geq t_{ij} - Mz_{ji} \qquad i \in B, j \in W \qquad (22)$$

$$s_i - s_j \leq t_{ij} + Mz_{ij} \qquad i \in B, j \in W \qquad (23)$$

$$z_{ij} \in \{0,1\} \qquad i,j \in B \cup W. \qquad (24)$$

The objective function minimizes the makespan, and constraints (15) and (16) ensure that all tasks are completed before the makespan. Constraints (17) and (18) set the earliest start time for a task as the time to reach the task location, whereas (20)–(23) avoid collisions among tasks. Constraints (24) force the precedence variables to be binary.

3.3 Additive lower bound

Formulation TRSP 1 involves a large number of variables, which limits its computational reach. Formulation TRSP 2 is more compact than TRSP 1, but a big-M term is necessary to linearize constraints (20)–(23), resulting in a weaker lower bound. Therefore, we have devised an additive lower bound as an alternative.

Denote a partial solution for the TRSP by the set $S = \{(r, i, u_i) : r \in \{B, W\}, j \in B \cup W\}$, where r specifies the robot, i specifies the task, and u_i specifies the start time of task i. Furthermore, $\overline{B}(S) \subseteq B$ and $\overline{W}(S) \subseteq W$ denote the set of tasks already assigned within the partial solution S, respectively. The individual makespans for the robots are denoted as $C_B(S)$, and $C_W(S)$. A lower bound for this partial solution is then

$$C_{lb}(S) = \max\{C_B(S) + \sum_{i \in B \setminus \overline{B}(S)} 2b_i, C_W(S) + \sum_{j \in W \setminus \overline{W}(S)} 2w_j\}. \qquad (25)$$

The resulting branch-and-bound algorithm is given below.

Branch-and-bound

$S = \emptyset$;

$List = \{S\}$; // Insert the empty partial solution in the *List*

$S_{Best} = S_{\text{FFD}}$; // *The result of the First Fit Decreasing heuristic*

$C_{Best} = z_{\text{FFD}}$; // *Best known solution value*

While $(List \neq \emptyset)$

Select a partial solution S from the *List*;

If $(C_{lb}(S) < C_{Best})$

If$(\overline{B}(S) = B$ and $\overline{W}(S) = W)$ // Feasible solution

C_{Best}=$C_{lb}(S)$;

S_{Best}=S;

Else

For $(i \in B \setminus \overline{B}(S))$

Determine u_i as the earliest possible start time of task i

$List = List \cup \{ S \cup \{(B, i, u_i)\} \}$;

For $(j \in W \setminus \overline{W}(S))$

Determine u_j as the earliest possible start time of task j

$List = List \cup \{ S \cup \{(W, j, u_j)\} \}$;

Observe that for the branching step, the earliest start times of all possible candidate tasks ($O(n)$) should be determined by checking against collisions with the tasks that have already started or are scheduled to start later on the other robot ($O(n)$). This results in a complexity of $O(n^2)$ per node. To alleviate this complexity, we have observed that a good strategy is to branch first on the tasks of the robot with the lower individual makespan.

4 Constructive heuristics

We have also developed two constructive heuristics for the TRSP.

4.1 PairMatch algorithm

We now present a constructive heuristic algorithm, and prove that it has a worst-case performance guarantee of $\rho = 3/2$. We first prove three lemmas to help us with the overall proof.

Lemma 1 *Given an instance with two tasks having distances b_i and b_j such that $b_i + b_j \leq L - 1$, removing them from the task list of the black robot and adding a single task at a*

distance $b_k = b_i + b_j$ to the task list B results in an instance, any feasible solution of which can be converted into a feasible solution for the original instance. A symmetric argument is also valid for the white robot and any pair of tasks.

Proof. Any feasible solution of the modified instance is still feasible if task b_k is decomposed into tasks b_i and b_j. □

Lemma 2 *Given an instance of TRSP with two tasks $b_1 \geq \lceil L/2 \rceil$, $w_1 \geq \lceil L/2 \rceil$, and $b_1 + w_1 \geq L + 1$, the optimal makespan is $C_{max} = 2b_1 + 2w_1 - (L - 1)$.*

Proof. We will analyze the makespan in two cases.

1. Assume that the white robot is scheduled to start immediately. For the optimal solution, the black robot must be at a distance of one unit from the white robot at time w_1, which is consequently at a distance of $L - 1 - w_1$ from the black depot. To reach this point, the black robot should start moving at time $w_1 - (L - 1 - w_1) = 2w_1 - (L-1)$. Finally, the black robot will be back at its depot at $2b_1 + 2w_1 - (L-1)$, determining the makespan.

2. Assume that the black robot is scheduled to start immediately. For the optimal solution, the white robot has to be at a distance of 1 unit from the black robot at time b_1, which is at a distance of $L-1-b_1$ from the white depot. To reach this point, the white robot should leave the depot at time $b_1-(L-1-b_1) = 2b_1-(L-1)$. Finally, the white robot will be back at its depot at time $2b_1 + 2w_1 - (L - 1)$, determining the makespan.

□

Lemma 3 *Given an instance of TRSP with two tasks $b_1 \geq w_1$ and $w_1 < \lceil L/2 \rceil$, the optimal solution value is $C_{max} = 2b_1$. Symmetrically, if $b_1 \leq w_1$ and $b_1 < \lceil L/2 \rceil$, the optimal solution value is $C_{max} = 2w_1$.*

Proof. In the first case, if both tasks start at time 0, the white robot arrives at its pickup point before the black robot, and the makespan is determined by the black robot. The proof of the second case is similar. □

The heuristic algorithm starts by aggregating tasks of the black and white robots as follows. If the distance of a task from its depot is less than $\lceil L/2 \rceil$, it is combined with

the next task of the same robot in the ordered list, while the resulting task has an overall processing time lower than $\lceil L/2 \rceil$. This process results in a new instance in which all tasks have a distance that is greater or equal than $\lceil L/2 \rceil$, with at most one task shorter than $\lfloor L/2 \rfloor$ for each robot. Lemma 1 guarantees that a solution for this instance (which may involve multiple items at the same location) can be translated into a feasible solution for the original instance. Denote the aggregated task lists as B' and W'. For simplicity, we reindex the two new task lists starting from 1, and we denote the distances associated with tasks in B' and W' as b'_i and w'_i, respectively.

Next, the tasks of the black and white robots with matching indices are paired (as long as both robots have an available task). If the tasks in the pair both have distances b'_i and w'_i greater than $\lceil L/2 \rceil$, they are scheduled according to Lemma 2, which leads to a makespan increase equal to $2b'_i + 2w'_i - (L-1)$. If, on the other hand, $b'_i + w'_i \geq L$ and $b_i < \lceil L/2 \rceil$ or $w_i < \lceil L/2 \rceil$, the pair of tasks can be scheduled according to Lemma 3 and the makespan increases by $\max\{2b'_i, 2w'_i\}$. Finally, if $b'_i + w'_i \leq L - 1$, both tasks can be appended to the end of the schedules of the associated robots without creating any conflict. The tasks that could not be paired because either $|B'| > |W'|$ or $|W'| > |B'|$ are performed at the end of the sequence of paired tasks, leading to an increase of the makespan equal to the sum of their processing times. We call this algorithm PairMatch, and provide the formal description below.

PairMatch

$C_{max} = 0$;
For$(i = 1, \ldots, B-1)$
  If $(b_i + b_{i+1} \leq L - 1)$
    $b_{i+1} = b_{i+1} + b_i$;
    $b_i = 0$;
For$(j = 1, \ldots, W-1)$
  If $(w_j + w_{j+1} \leq \lceil L - 1 \rceil)$
    $w_{j+1} = w_{j+1} + w_j$;
    $w_j = 0$;
// Remove jobs with $b_i = 0$ and $w_j = 0$ from B and W to obtain B' and W'
B', W';
For $(i = 1, \ldots, \min\{|B'|, |W'|\})$
  If $((b'_i \geq \lceil L/2 \rceil)$ and $(w'_i \geq \lceil L/2 \rceil))$ // Lemma 2
    $C_{max} = C_{max} + 2b'_i + 2w'_i - (L-1)$;
  Else If $(b'_i + w'_i \geq L)$ // Lemma 3
    If $(b'_i < \lceil L/2 \rceil)$ $C = C + 2b'_i$;
    Else $C_{max} = C_{max} + 2w'_i$;
  Else
    $C_{max} = C_{max} + \max\{2b'_i, 2w'_i\}$;
// Append remaining jobs
If $(|B'| > |W'|)$
  For$(i = |W'| + 1, \ldots, |B'|)$ $C_{max} = C_{max} + 2b'_i$;
If $(|W'| > |B'|)$
  For$(j = |B'| + 1, \ldots, |W'|)$ $C_{max} = C_{max} + 2w'_j$;

The aggregation operation, reordering, and scheduling steps all take $O(n)$ time, determining the complexity of the PairMatch algorithm as $O(n)$.

Proposition 7 *The PairMatch algorithm has a performance guarantee of* $\rho = 3/2$.

Proof. Without loss of generality, we assume that $\sum_{i \in B'} b_i \geq \sum_{j \in W'} w_j$. A valid lower bound on the solution value under this assumptions is $C_{lb} = \sum_{i \in B'} 2b_i$.

Case 1: $|B'| > |W'|$.

  Subcase 1a: $w_{|W'|} \geq \lceil L/2 \rceil$.

  The makespan is then

$$C_{max} = \sum_{i=1}^{|W'|}[2b_i + 2w_i - (L-1)] + \sum_{i=|W'|+1}^{|B'|} 2b_i = \sum_{i=1}^{|W'|}[2w_i - (L-1)] + \sum_{i=1}^{|B'|} 2b_i. \quad (26)$$

Since $w_i \leq (L-1)$ and $\sum_{i=1}^{|B'|} b_i \geq \sum_{i=1}^{|W'|} w_i$:

$$C_{max} \leq \sum_{i=1}^{|W'|} w_i + \sum_{i=1}^{|B'|} 2b_i \leq \sum_{i=1}^{|B'|} 3b_i. \quad (27)$$

Therefore the ratio is $\rho = C_{max}/C_{lb} = 3/2$. In each of the following subcases, we show that the ratio remains the same.

Subcase 1b: $w_{|W'|} < \lceil L/2 \rceil$. The makespan is:

$$C_{max} = \sum_{i=1}^{|W'|-1} [2b_i + 2w_i - (L-1)] + \sum_{i=|W'|}^{|B'|} 2b_1 \quad (28)$$

$$= \sum_{i=1}^{|W'|-1} [2w_i - (L-1)] + \sum_{i=1}^{|B'|} 2b_i \leq \sum_{i=1}^{|W'|} w_i + \sum_{i=1}^{|B'|} 2b_i \leq \sum_{i=1}^{|B'|} 3b_i. \quad (29)$$

Case 2: $|B'| = |W'|$.

Subcase 2a: $b_{|B'|} \geq \lceil L/2 \rceil, w_{|W'|} \geq \lceil L/2 \rceil$. The makespan is

$$C_{max} = \sum_{i=1}^{|B'|}[2b_i + 2w_i - (L-1)] \leq \sum_{i=1}^{|B'|}[2b_1 + w_i] \leq \sum_{i=1}^{|B'|} 3b_i. \quad (30)$$

Subcase 2b: $b_{|B'|} \geq \lceil L/2 \rceil, w_{|W'|} < \lceil L/2 \rceil$. The makespan is

$$C_{max} = \sum_{i=1}^{|B'|-1} [2b_i + 2w_i - (L-1)] + 2b_{|B'|} \leq \sum_{i=1}^{|B'|-1} [2b_i + w_i] + 2b_{|B'|} + w_{|B'|} \leq \sum_{i=1}^{|B'|} 3b_i. \quad (31)$$

Subcase 2c: $b_{|B'|} < \lceil L/2 \rceil, w_{|B'|} \geq \lceil L/2 \rceil$. The makespan is

$$C_{max} = \sum_{i=1}^{|B'|-1} [2b_i + 2w_i - (L-1)] + 2w_{|B'|} \leq \sum_{i=1}^{|B'|-1} [b_i + 2w_i] + b_{|B'|} + 2w_{|B'|} \leq \sum_{i=1}^{|B'|} 3b_i. \quad (32)$$

Subcase 2d: $b_{|B'|} < \lceil L/2 \rceil, w_{|B'|} < \lceil L/2 \rceil$. If $b_{|B'|} \geq w_{|B'|}$, the makespan is

$$C_{max} = \sum_{i=1}^{|B'|-1} [2b_i + 2w_i - (L-1)] + 2b_{|B'|} \leq \sum_{i=1}^{|B'|-1} [2b_i + w_i] + 2b_{|B'|} + w_{|B'|} \leq \sum_{i=1}^{|B'|} 3b_i. \quad (33)$$

Otherwise it is equal to

$$C_{max} = \sum_{i=1}^{|B'|-1} [2b_i + 2w_i - (L-1)] + 2w_{|B'|} \leq \sum_{i=1}^{|B'|-1} [b_i + 2w_i] + b_{|B'|} + 2w_{|B'|} \leq \sum_{i=1}^{|B'|} 3b_i. \quad (34)$$

Case 3: $|B'| < |W'|$.

Subcase 3a: $b_{|B'|} \geq \lceil L/2 \rceil$. The makespan is

$$C_{max} = \sum_{i=1}^{|B'|} [2b_i + 2w_i - (L-1)] + \sum_{i=|B'|+1}^{|W'|} 2w_i = \sum_{i=1}^{|B'|} [2b_i - (L-1)] + \sum_{i=1}^{|W'|} 2w_i \quad (35)$$

$$\leq \sum_{i=1}^{|B'|} b_i + \sum_{i=1}^{|W'|} 2w_i \leq \sum_{i=1}^{|B'|} 3b_i. \quad (36)$$

Subcase 3b: $b_{|B'|} < \lceil L/2 \rceil$. The makespan is

$$C_{max} = \sum_{i=1}^{|B'|-1} [2b_i + 2w_i - (L-1)] + \sum_{i=|B'|}^{|W'|} 2w_i = \sum_{i=1}^{|B'|-1} [2b_i - (L-1)] + \sum_{i=1}^{|W'|} 2w_i \quad (37)$$

$$\leq \sum_{i=1}^{|B'|} b_i + \sum_{i=1}^{|W'|} 2w_i \leq \sum_{i=1}^{|B'|} 3b_i. \quad (38)$$

□

As the final result, we now prove that the performance guarantee is tight.

Proposition 8 *The performance guarantee of the PairMatch algorithm is tight.*

Proof. Consider an instance of the TRSP with $L = 11$, $b_1 = 10$, $w_1 = 5, w_2 = 3, w_3 = 2$. The optimal solution value is 20, which is equal to the workload of both robots. The PairMatch algorithm will aggregate the tasks of the white robot, resulting a single task with $w'_1 = 10$. The two tasks will be scheduled according to Lemma 2 to yield a makespan of 30, and hence the performance ratio of 3/2 will be attained. Figure 7a depicts the

optimal solution and Figure 7b depicts the solution of the PairMatch algorithm for the instance. Consequently, Lemma 2 will schedule the two tasks with a total makespan of 30. □

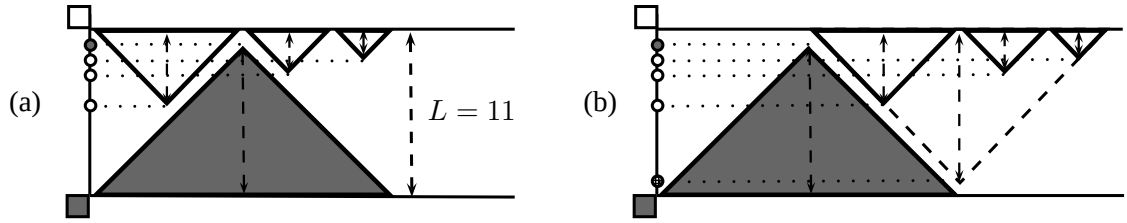

Figure 7: Instance described in Proposition 8.

4.2 First Fit Decreasing algorithm

We now present a second constructive heuristic which performs very well despite its simplicity. The idea of the heuristic is to schedule a single task at a time, on the robot with the shorter makespan. The task to be scheduled is selected as the one with the largest distance time among the tasks that can start the earliest. We call this algorithm as First Fit Decreasing (FFD), and provide the formal description below. In the pseudo-code below, we write u_k to denote the earliest possible start time of task $k \in B \cup W$. We denote the set of tasks already assigned as $\overline{B}$ and $\overline{W}$.

FFD
$S = \emptyset$;
While $(\overline{B}(S) \neq B)$ or $(\overline{W}(S) \neq W)$
  If $(C_B(S) < C_W(S))$ or $(\overline{W}(S) = W)$
    Determine $u_{min} = \min_{k \in B \setminus \overline{B}(S)}\{u_k\}$;
    Select $i \in B \setminus \overline{B}(S)$ such that $b_i = \max_{k \in B \setminus \overline{B}(S): u_k = u_{min}}\{b_k\}$;
    $S = S \cup \{(B, i, u_{min})\}$;
  Else
    Determine $u_{min} = \min_{k \in W \setminus \overline{W}(S)}\{u_k\}$;
    Select $j \in W \setminus \overline{W}(S)$ such that $w_j = \max_{k \in W \setminus \overline{W}(S): u_k = u_{min}}\{w_k\}$;
    $S = S \cup \{(W, j, u_{min})\}$;

Every time a new task is to be appended ($O(n)$ times), all possible candidate tasks ($O(n)$), should be checked for collisions against the tasks that have already started or are

scheduled to start later on the other robot. Since we select the robot with the shortest makespan at every iteration, there can be at most one such task. Consequently, the complexity of the FFD algorithm is $O(n^2)$.

5 Computational Results

The algorithms presented in the two previous sections have been implemented using C++ and CPLEX 12.5, and computational experiments were performed on the IRIDIS 4 computing cluster having 2.6 GHz cores with 4GB of memory per core.

Our first test instances were generated randomly, with each position having a 40% probability of housing a task for the black robot, 40% probability of housing a task for the black robot, and 20% probability to be empty. Initial experiments have shown that instances having an uneven workload between the two robots are extremely easy, and are solved to optimality within less than a second, even for hundreds of tasks for each robot. Consequently, we have focused our search on instances with a performance guarantee of $\bar{\rho} \geq 1.99$. Surprisingly, many of these instances also turned out to be very easy, and varying the sparsity of jobs did not seem to have an effect on the difficulty of the instance.

In order to generate difficult instances, we have linked the random instance generator with the solver, and we have specifically looked for instances that would take more than one CPU second to be solved. Our search resulted in 50 such instances, each of which required a few CPU seconds to be generated. Although having no more than 27 tasks in total, the instances force all three exact algorithms to struggle in order to prove optimality.

The results of the exact algorithms and the heuristic algorithms on the difficult instances are presented in in Tables 1 and 2, respectively. Table 1 also includes the details about the number of tasks each instance involves. The worst performance belongs to TRSP1, the root node relaxation of which could not be solved within two hours of CPU time for 11 out of the 50 instances. It successfully solves two instances to optimality, and the average gap for the remaining 37 instances is 57%. TRSP2 successfully solves 34 instances, and the average gap for the remaining 16 instances is 0.5%, with an overall average computing time of 2767 seconds. The branch-and-bound algorithm has the best performance. It successfully solves 40 instances, with an average gap of 0.6% for the remaining 10, and an overall average computing time of 1897 seconds. It should be noted that the unsolved instances for TRSP2 and the branch-and-bound algorithm do not coincide, i.e. TRSP2 can solve four instances that the branch-and-bound algorithm cannot solve, and the branch-and-bound algorithm can solve 10 instances that TRSP2 cannot solve. The optimal solution values proved are denoted in boldface in Table 2.

Regarding the heuristic algorithms, the PairMatch algorithm has a consistent perfor-

mance, with minimum, average, and maximum deviations of 31.9%, 39.7%, and 46.8% from the best known or optimal solution value, respectively. Evidently, the scheduling approaches described in Lemmas 2 and 3 restrict one of the robots to wait, which results in a performance that is quite close to the performance guarantee. The FFD algorithm has a much better performance, finding the optimal solution in eight out of 50 instances, and having an average deviation of 1.8% and a maximum deviation of 10.8% from the best known or optimal solution value. Both heuristics take less than 0.01 second, so no CPU times are reported for these algorithms.

6 Conclusions

We have introduced, modeled, and solved the problem of scheduling twin robots on a line. A proof of $\mathcal{NP}$-hardness was presented, along with some useful problem properties. Three exact algorithms and two heuristic algorithms were developed, one with a performance guarantee of 3/2. The results show that the problem can be solved efficiently for large size instances with uneven workload for the robots, can be rather challenging for some rare instances with equal workload. Among the three exact algorithms presented, the branch-and-bound algorithm exhibits the best computational performance. The Pair-Match heuristic, which has a worst-case ratio of 3/2, has an empirical performance that almost matches this ratio and is overwhelmingly overperformed by the FFD heuristic. A meaningful extension of this problem is the on-line version in which the tasks are revealed dynamically. For related problems, see Jaillet and Wagner (2008, 2010).

Acknowledgements: This study was partially supported by Centre for Operational Research, Management Science and Information Systems based within the University of Southampton, and by the Canadian Natural Sciences and Engineering Research Council under grant 39682-10. This support is gratefully acknowledged.

Table 1: Results of the exact algorithms on the difficult instances

					TRSP1		TRSP2		Branch-and-bound	
Inst.	n	L	$\|B\|$	$\|W\|$	Final Gap (%)	CPU time (sec.)	Final Gap (%)	CPU time (sec.)	Final Gap (%)	CPU time (sec.)
1	13	20	7	6	1.4	7200.0	0.0	20.2	0.0	14.8
2	16	20	8	8	19.0	7200.0	0.0	2668.2	0.0	5964.6
3	13	20	6	7	1.3	7200.0	0.0	15.4	0.0	2.9
4	15	20	8	7	9.8	7200.0	0.0	771.1	0.0	325.5
5	14	20	7	7	7.8	7200.0	0.0	76.2	0.0	25.9
6	13	20	7	6	0.8	7200.0	0.0	16.8	0.0	16.7
7	16	20	8	8	N/A	7200.0	0.0	3498.8	0.7	7200.0
8	12	20	5	7	N/A	7200.0	0.0	2.9	0.0	1.3
9	12	20	6	6	0.0	1918.1	0.0	4.4	0.0	3.5
10	14	20	7	7	0.0	6968.4	0.0	87.2	0.0	214.0
11	19	25	9	10	83.3	7200.0	0.7	7200.0	0.0	79.2
12	18	25	9	9	81.5	7200.0	0.0	0.1	0.0	5.4
13	20	25	11	9	89.3	7200.0	0.7	7200.0	0.0	2.4
14	21	25	11	10	91.0	7200.0	0.0	3.9	0.0	2.6
15	12	25	6	6	N/A	7200.0	0.0	4.8	0.0	1.3
16	13	25	6	7	3.6	7200.0	0.0	5.2	0.0	8.2
17	16	25	8	8	22.2	7200.0	0.0	0.0	0.0	1.4
18	13	25	5	8	N/A	7200.0	0.0	2.0	0.0	9.7
19	18	25	9	9	29.2	7200.0	0.0	0.0	0.0	18.6
20	17	25	9	8	27.7	7200.0	0.0	0.1	0.0	4.8
21	18	30	8	10	79.6	7200.0	0.4	7200.0	0.4	7200.0
22	19	30	10	9	87.8	7200.0	0.4	7200.0	0.4	7200.0
23	18	30	10	8	86.9	7200.0	0.0	4573.9	0.0	7.6
24	17	30	7	10	79.7	7200.0	0.7	7200.0	0.0	3.1
25	19	30	9	10	84.5	7200.0	0.6	7200.0	0.0	431.1
26	16	30	8	8	15.5	7200.0	0.0	5892.1	0.5	7200.0
27	14	30	7	7	5.3	7200.0	0.0	99.6	0.0	642.0
28	15	30	9	6	58.0	7200.0	0.0	432.3	0.0	1130.4
29	19	30	10	9	84.6	7200.0	0.3	7200.0	0.3	7200.0
30	15	30	8	7	12.2	7200.0	0.0	755.0	0.0	6120.6
31	19	35	9	10	91.5	7200.0	0.0	468.9	0.0	6.2
32	20	35	11	9	91.0	7200.0	0.0	2.2	0.0	522.1
33	24	35	13	11	91.8	7200.0	0.5	7200.0	0.0	104.8
34	19	35	11	8	82.2	7200.0	0.0	0.1	0.0	43.7
35	21	35	12	9	78.2	7200.0	0.0	0.1	2.6	7200.0
36	20	35	9	11	91.0	7200.0	0.5	7200.0	0.0	2.1
37	23	35	11	12	91.5	7200.0	0.5	7200.0	0.0	3.0
38	23	35	12	11	91.9	7200.0	0.0	0.5	0.0	3.0
39	23	35	11	12	N/A	7200.0	0.5	7200.0	0.0	2.9
40	24	35	12	12	92.2	7200.0	0.9	7200.0	0.0	34.3
41	22	40	10	12	N/A	7200.0	0.0	3742.7	0.0	2.8
42	26	40	12	14	90.6	7200.0	0.0	4.4	0.0	432.7
43	27	40	13	14	N/A	7200.0	0.0	0.2	0.0	5653.9
44	24	40	11	13	N/A	7200.0	0.2	7200.0	0.2	7200.0
45	27	40	13	14	N/A	7200.0	0.4	7200.0	0.0	3.5
46	24	40	12	12	91.0	7200.0	0.0	4.2	0.0	2.5
47	25	40	12	13	N/A	7200.0	0.4	7200.0	0.2	7200.0
48	21	40	11	10	90.0	7200.0	0.0	0.1	0.8	7200.0
49	20	40	11	9	87.7	7200.0	0.0	0.0	0.0	998.3
50	25	40	12	13	N/A	7200.0	0.2	7200.0	0.2	7200.0
Avg.					57.0	7089.7	0.2	2767.1	0.1	1897.1

Table 2: Results of the heuristic algorithms on the difficult instances

	PairMatch		FFD		Best known
Inst.	Value	Deviation (%)	Value	Deviation (%)	or Optimal
1	186	31.9	145	2.8	**141**
2	251	41.8	181	2.3	**177**
3	218	38.9	159	1.3	**157**
4	235	37.4	175	2.3	**171**
5	233	41.2	169	2.4	**165**
6	171	46.2	**117**	0.0	**117**
7	200	37.9	**145**	0.0	**145**
8	192	42.2	137	1.5	**135**
9	165	43.5	121	5.2	**115**
10	177	39.4	**127**	0.0	**127**
11	382	38.4	286	3.6	**276**
12	348	41.5	248	0.8	**246**
13	374	37.5	274	0.7	**272**
14	404	41.3	288	0.7	**286**
15	290	36.8	228	7.5	**212**
16	318	43.2	246	10.8	**222**
17	300	42.9	216	2.9	**210**
18	282	36.9	208	1.0	**206**
19	304	35.7	228	1.8	**224**
20	276	46.8	194	3.2	**188**
21	379	38.8	273	0.0	273
22	369	40.3	263	0.0	263
23	434	40.0	313	1.0	**310**
24	392	38.0	289	1.8	**284**
25	484	46.7	351	6.4	**330**
26	284	41.3	**201**	0.0	**201**
27	225	34.7	169	1.2	**167**
28	314	40.8	227	1.8	**223**
29	406	39.5	301	3.4	291
30	256	36.9	**187**	0.0	**187**
31	568	40.6	406	0.5	**404**
32	526	37.7	386	1.0	**382**
33	588	39.3	436	3.3	**422**
34	468	39.3	348	3.6	**336**
35	436	43.4	316	3.9	**304**
36	546	43.7	384	1.1	**380**
37	568	40.6	406	0.5	**404**
38	606	43.6	428	1.4	**422**
39	624	40.5	450	1.4	**444**
40	604	37.3	442	0.5	**440**
41	694	37.7	505	0.2	**504**
42	542	33.5	411	1.2	**406**
43	710	37.6	517	0.2	**516**
44	704	38.3	514	1.0	509
45	776	38.1	563	0.2	**562**
46	606	40.3	433	0.2	**432**
47	775	37.4	573	1.6	564
48	530	38.7	385	0.8	**382**
49	461	40.5	329	0.3	**328**
50	716	39.0	517	0.4	515
Avg.		39.7		1.8	

References

All Glass s.r.l. Robot onto rails, Accessed Sep 2013. URL `http://sito.allglass.it/macchine.php?m=50`.

N. Boysen, S. Emde, and M. Fliedner. Determining crane areas for balancing workload among interfering and noninterfering cranes. *Naval Research Logistics*, 59:656–662, 2012.

F.F. Chen and Q. Su. Scheduling single-gripper gantry robots in tightly coupled serial production lines: Optimum vs. push/pull concept based sequences. *Journal of Manufacturing Systems*, 14:139–147, 1995.

M. Dawande, H.N. Geismar, and S.P. Sethi. Sequencing and scheduling in robotic cells: Recent developments. *Journal of Scheduling*, 8:387–426, 2005.

K.E. Dimitri, F.D. Gallo, J.E. Kulakowski, R.J. Means, J.L. Thrall, and D.J. Winarski. Automated data storage library dual picker interference avoidance, 14 2000. US Patent 6,038,490.

M.G. Dotoli and M.P. Fanti. A coloured Petri net model for automated storage and retrieval systems serviced by rail-guided vehicles: A control perspective. *International Journal of Computer Integrated Manufacturing*, 18:122–136, 2005.

M. Dror, P. L' Écuyer, and M. Mayrand. Dynamic scheduling of a robot servicing machines on a one-dimensional line. *IIE Transactions*, 23:371–382, 1991.

A.H. Gharehgozli, G. Laporte, Y. Yu, and R. de Koster. Scheduling twin yard cranes in a container block. *Transportation Science*, 2013. Submitted for publication.

P. Jaillet and M.R. Wagner. Generalized online routing: New competitive ratios, resource augmentation, and asymptotic analyses. *Operations Research*, 56:745–757, 2008.

P. Jaillet and M.R. Wagner. Almost sure asymptotic optimality for online routing and machine scheduling problems. *Networks*, 55:2–12, 2010.

S.G. Lee, R. de Souza, and E.K. Ong. Simulation modelling of a narrow aisle automated storage and retrieval system (as/rs) serviced by rail-guided vehicles. *Computers in Industry*, 30: 241–253, 1996.

A. Lim, B. Rodrigues, F. Xiao, and Y. Zhu. Crane scheduling with spatial constraints. *Naval Research Logistics*, 51:386–406, 2004.

A. Lim, B. Rodrigues, and Z. Xu. A m-parallel crane scheduling problem with a non-crossing constraint. *Naval Research Logistics*, 54:115–127, 2007.

A. Mathijssen and A.J. Pretorius. Verified design of an automated parking garage. In L. Brim, B. Haverkort, M. Leucker, and J. Pol, editors, *Formal Methods: Applications and Technology*, volume 4346 of *Lecture Notes in Computer Science*, pages 165–180. Springer, Berlin Heidelberg, 2007.

F. Meisel. The quay crane scheduling problem with time windows. *Naval Research Logistics*, 58: 619–636, 2011.

W.C. Ng. Crane scheduling in container yards with inter-crane interference. *European Journal of Operational Research*, 164:64 – 78, 2005.

P.G. Ranky. Collaborative, synchronous robots serving machines and cells. *Industrial Robot: An International Journal*, 30:213–217, 2003.

A. Stulman. A "static" optimal strategy for a service robot: A one dimensional case. *Computers & Industrial Engineering*, 16:509–514, 1989.

Q. Su and F.F. Chen. Optimal sequencing of double-gripper gantry robot moves in tightly-coupled serial production systems. *IEEE Transactions on Robotics and Automation*, 12:22–30, 1996.

W.E. Wilhelm. Complexity of sequencing tasks in assembly cells attended by one or two robots. *Naval Research Logistics*, 34:721–738, 1987.

Y. Zhu and A. Lim. Crane scheduling with non-crossing constraint. *Journal of the Operational Research Society*, 57:1464–1471, 2006.