Proceedings of the 2016 Winter Simulation Conference
T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds.

TASK SCHEDULING IN A FULL ROAMING SHUTTLE SYSTEM

Martijn P.O.J. Gootzen
Ivo J.B.F. Adan

Eindhoven University of Technology
PO Box 513
Eindhoven, 5600 MB, THE NETHERLANDS

Jorine W.E. Heling
Bruno van Wijngaarden

Vanderlande
Vanderlandelaan 2
Veghel, 5466 RB, THE NETHERLANDS

ABSTRACT

A new concept in automated storage and retrieval systems is the full roaming shuttle system, the distinguishing feature of which is that its material handling shuttles are not aisle-captive, but can easily switch aisles and levels in the storage area. A consequence of this flexibility is that shuttles more often overtake each other and deliver product totes in a different sequence than they are requested. In case of strict sequence requirements for delivered totes, this leads to more waiting time of shuttles and thus to loss of throughput capacity. In this paper we propose heuristics to assign tasks to shuttles that aim at minimizing the number of out-of-sequence occurrences and at maximizing the throughput capacity. These heuristics are evaluated through simulation. The results suggest that, in comparison to first-in first-out task assignment, substantial throughput improvement can be achieved by employing smart task assignment heuristics.

1 INTRODUCTION

Distribution centers are important for all companies handling many products that have to be distributed or stocked. One of the main activities in a distribution center is order picking, which can be defined as the process of retrieving products from the warehouse to fulfill customer orders. Other operations in distribution centers include receiving, putting away, storing and shipping, though order picking is the most expensive one (Tompkins, White, Bozer, and Tanchoco 2010). A detailed overview of operations in distribution centers can be found in (Bartholdi and Hackman 2014), and a description of different types of order picking systems is provided in (de Koster, Le-Duc, and Roodbergen 2007).

To reduce operational costs and to increase throughput capacity, many companies invest in automating the order picking process, in particular in automated storage and retrieval systems (ASRS). In ASRS product totes containing the items that have to be picked are automatically transported to workstations where operators (or robots) will pick one or more items from the product tote, after which the tote will again be transported to and stored in the storage area. A novel solution of such goods-to-man systems is the full roaming shuttle system (FRS), an example of which is the 3D storage system (ADAPTO 2015) of Vanderlande. This system is suitable for slow-moving products in retail and wholesale distribution centers. The distinguishing feature of an FRS is that its material handling devices, i.e. roaming shuttles, are not aisle-captive, but they can easily switch aisles and even levels (through lifts) in a multi-level storage area. A consequence of increased routing flexibility is that shuttles will more often overtake each other and deliver product totes in a different sequence than they are released. Hence, when totes have to be delivered in a strict sequence (for example, due to buffer or loading constraints), this flexibility may lead to more waiting time of shuttles and operators, and thus to loss of throughput capacity. For companies, however, it is important that maximum throughput is achieved for the existing or new equipment. This may achieved by developing intelligent control strategies.

(Matthijssen 2012) investigates the performance of FRS where order lines are released to shuttles according to first-in first-out (FIFO). This is a simple strategy, suitable in case there are no sequencing requirements. However, when sequencing is required, he shows that this strategy leads to longs waiting times at the lifts. (van Dorst 2012) investigates the performance of a more intelligent task assignment method based on the assumption of fixed inter-arrival times (FIAT) of shuttles, called the required arrival time (RAT) method, and shows that significant throughput improvement can be achieved with RAT. However, RAT has not been validated in case of conflicts (i.e., interference delays) between shuttles, and a drawback of RAT is that it requires an accurate estimate of the throughput. To overcome this drawback, Van Dorst recommends to develop scheduling heuristics that use information about the current state of shuttles in the FRS in order to predict their arrival times. That is what we intend to in the current paper: We propose smart task assignment heuristics that aim at minimizing the number of out-of-sequence occurrences, by using information on estimated arrival times of shuttles. The performance of these heuristics will investigated through simulation, which takes into account stochastic interference delays.

This paper is organized as follows. In the next section we describe the FRS. Afterwards, in Section 3, we describe the smart scheduling heuristics, by using a running example. In Section 4 the simulations, which are used to evaluate the heuristics, are discussed, followed by a section showing the results of these simulations. Finally, the simulation model and its results are discussed, followed by a conclusion.

2 SYSTEM DISCRIPTION

ASRS consist of multiple aisles where product totes are stored in racks. Product totes are stored and retrieved from the racks with material handling devices. The ASRS considered in this paper is the full roaming shuttle system (FRS). An FRS typically consists of multiple levels, connected by lifts, where each level has multiple aisles connected by cross-aisles, and these systems can be build modularly. The distinguishing feature of an FRS is that its material handling devices, i.e., roaming shuttles, are not bound to a specific part of the storage area, but can reach every location in the storage area. Hence, an advantage of the FRS system is that multiple storage and retrieval machines (SRMs) can simultaneously retrieve product totes from any aisle at any level. The FRS recently developed by Vanderlande is (ADAPTO 2015).

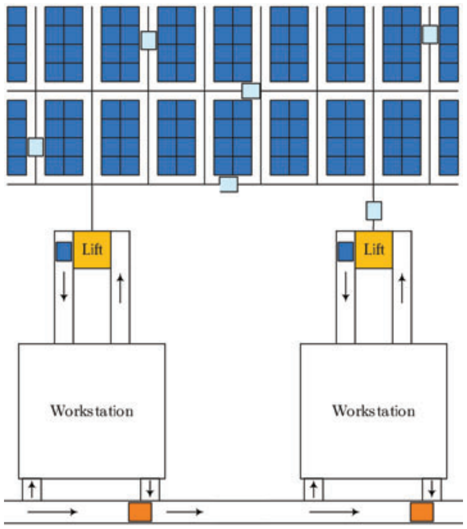


Figure 1: Schematic top view of an FRS showing the layout of a single level (in blue) of the storage area.

Figure 1 displays a schematic representation of an FRS, with two cross-aisles at each level, two lifts and two workstations. Shuttles can move along the cross-aisle to the desired storage-aisle. In the storage-aisle a shuttle travels along the aisle, to first store and then retrieve a product tote. Afterwards the shuttle travels to the front cross-aisle, and then, via this cross-aisle, to the lift. The lift is connected to the cross-aisle with a rail, which is used as buffer offering space for exactly one shuttle. In this buffer, the shuttle can wait for the lift to arrive. In case more than one shuttle needs to wait for the lift, the additional ones wait in (and block) the cross-aisle, and it is not possible to change the sequence of waiting shuttles. When the lift arrives, the shuttle waiting in the buffer moves in the lift and is then transported to the workstation level, where the shuttle first unloads the product tote and then loads another one. After loading, the lift transports the shuttle back to a possibly different level and the cycle of the shuttle repeats. This process is referred to as dual cycle (combining storage and retrieval operations in a single cycle), as opposed to inbound (storage) and outbound (retrieval) cycles.

The FRS setup considered in this paper consists of a dual-cycle multi-level system, with a single cross-aisle at each level. We will now describe the dual cycle of shuttles in more detail.

Shuttles can retrieve product totes from any location in the system and transport them, by using the lift, to the workstation. After unloading the product tote at the workstation, the shuttle will receive a new retrieval task as well as a new product tote from which the required item(s) have been extracted, to store it back in the FRS. We assume that it is always possible to store this product tote in an open location in the same aisle as (and on its way to) the next product tote that has to be retrieved. This assumption is reasonable, since the fill rate of the storage area (i.e., percentage of locations occupied by totes) is typically at most 95%. It implies that no additional travel time is needed for the storage activity of shuttles and that the travel time is determined by the location of the product tote that has to be retrieved. Note that product totes can be claimed by one shuttle at the time. To avoid delays due to shuttles competing for the same product, fast-moving products are stored in multiple totes.

The different components of a shuttle cycle are chronologically listed in Figure 2, and they are explained below:

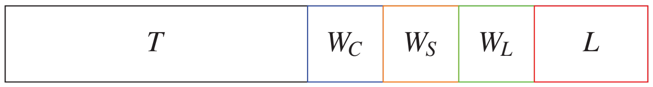


Figure 2: Dual cycle of a shuttle in an ASRS with full roaming shuttles.

- T is the travel time without conflict (i.e., interference delay) on a level of the storage area. This travel time is determined by the location of the product tote that has to be retrieved.
- W_C is the waiting time due to conflicts that the shuttle encounters on a level during a cycle.
- W_S is the waiting time of the shuttle due to being out-of-sequence.
- W_L is the waiting time for the lift. The lift is not available, but serving other shuttles, while this shuttle has the right sequence.
- L is the lift time. The lift time is the time from picking up the shuttle from the level in the ASRS until putting this shuttle back at the desired level for the next retrieval.

The conflict time W_C is stochastic and can happen anywhere during the travel time T on a level. This conflict time is due to interference delays between shuttles moving around at the same level of the ASRS. Two types of conflicts can occur. Either the storage aisle or cross-aisle the shuttle needs to visit is not available, because another shuttle has already claimed this aisle. These conflicts and resulting delays depend on the level of congestion in the system and they are difficult to predict and estimate. In Section 4 we will explain how the conflict time W_C is modeled in the simulation study.

The lift time L consists of multiple components. First, the lift picks up the shuttle on level A say, and transports it to the level of the workstation. The workstation unloads the product tote from the shuttle and loads it with a another one to be stored again in the storage area. Then the lift transports the shuttle to level B, where the shuttle leaves the lift, and then the lift moves to level C to pick up the next shuttle. Hence, the lift time L consists of entering and leaving, loading and unloading, and traveling between three levels A, B and C, though it will not be modeled in full detail in the simulation study.

The travel time T depends on the location of the tote to be retrieved, which can be anywhere in the system, and therefore the travel time T can strongly vary among the retrieval tasks. Hence it is possible that shuttles overtake each other. Since strict sequence restrictions are required by the workstation, shuttles that have overtaken other shuttles will have to wait in the input buffer in between the cross-aisle and the lift. As observed by (Matthijssen 2012), when retrieval tasks are assigned to shuttles according to FIFO, this may result in long waiting times in the input buffer and thus to loss of throughput capacity. Therefore, we are interested in more intelligent scheduling heuristics that attempt to produce task assign sequences leading to less out-of-sequence events. These scheduling heuristics are the topic of the next section.

3 SCHEDULING HEURISTICS

In this section we will explain the scheduling heuristics, and include a running example supported by graphs. The setting that we consider is a single lift (and single workstation) with a fixed number of shuttles dedicated to this lift. The number of shuttles assigned to the lift is indicated by k. Such systems are referred to as closed-loop systems (as opposed to pooled systems where multiple lifts share a pool of shuttles). There is an unlimited list of orders, i.e., product totes that have to be retrieved from the storage area. Hence, shuttles will immediately receive a new task when they deliver a product tote at the workstation, and thus never idle. This list of totes to be retrieved also prescribes the strict sequence in which the product totes have to be delivered at the workstation (so the first tote on the order list has to be delivered first, then the second one and so on).

The system throughput, that is the number of tasks performed per unit of time, depends on the sequence in which tasks are assigned to arriving shuttles. The most basic algorithm to schedule tasks is FIFO. Tasks are assigned to arriving shuttles in sequence of the list of orders, i.e., FIFO simply assigns the ith order on the list to the ith arriving shuttle. The advantage of this heuristic is its ease of understanding and implementation. A disadvantage is that it may lead to a lot of overtaking, resulting in long waiting times due to sequencing restrictions, and thus to loss of throughput capacity.

FIFO does not use any information on the duration of the shuttle cycles nor on the arrival times of shuttles: We now propose heuristics that use this information. To compute the duration of shuttle cycles, we will ignore the stochastic component W_C and use (an estimate of) the mean lift time $E(L)$ instead of L. The heuristics look ahead and consider the first N tasks on the list of retrieval tasks, that have not yet been assigned. The parameter N is called the (rolling) scheduling horizon, and the list of these N tasks is referred to as the scheduled list. The first heuristics is:

3.1 Brute Force

To decide which task should be assigned to the next arriving shuttle, this heuristic considers *all sequences* of the N tasks on the scheduled list, and for each sequence, it computes the makespan, which is the time required to complete all tasks on the scheduled list. To compute the makespan of a sequence, we have to estimate the shuttle cycles, in which we ignore (as mentioned above) the stochastic component W_C and use $E(L)$ instead of L. The heuristic then selects the sequence with minimal makespan, and the first task in this sequence will be assigned to the next arriving shuttle. Note that this task may not be the first one on the list of tasks, that have not yet been scheduled.

The Brute Force heuristic is illustrated for the example in Table1 with horizon $N = 3$. The tasks on the scheduled list are numbered $1,2,3$, and T_i is the travel time (in seconds) of task i. The lift time L is assumed to be exactly 2 seconds for each cycle. The number of shuttles is $k = 3$ and a_i is the time of the ith arrival of a shuttle, starting from the current time $t = 0$. Note that, since the second arrival is at time

$t = 6$, this shuttles occupies the lift from time $t = 4$ to $t = 6$, and from $t = 6$ to $t = 8$ it is occupied by the third shuttle.

Table 1: Scheduling example with $N = 3$.

i	$T_i[s]$	$a_i[s]$
1	4	0
2	7	6
3	10	8

There are $3! = 6$ possible sequences for which the makespan needs to be evaluated. Figures 3–5 illustrate the calculation of the makespan for three sequences: **a** = [1, 2, 3] (FIFO), **b** = [3, 2, 1] and **c** = [2, 3, 1]. Note that sequence **a** in Figure 3 corresponds to FIFO, and that for this sequence, the waiting W_L after T_1 is due to the fact the lift is busy with the other shuttles. The makespan is 20 seconds for sequence **a**,18 seconds of **b** and 19 seconds of **c**. Hence, in case the makespan of the other three sequences is greater than 18 seconds, the Brute Force method selects sequence **b** and assigns task T_3 to the shuttle arriving at time $a_1 = 0$. Then it deletes task 3 from the scheduled list $\{1,2,3\}$, adds task 4, so the scheduled list becomes $\{1,2,4\}$, and sets $a_4 = 18$. Then time jumps $a_2 = 6$ seconds ahead to the next arrival, and the Brute Force method repeats.

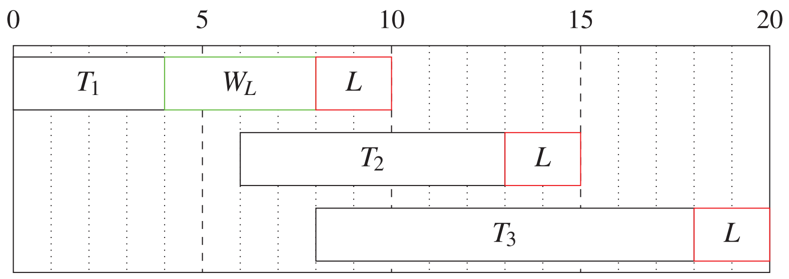


Figure 3: Retrieval scenario **a** = [1, 2, 3] (FIFO).

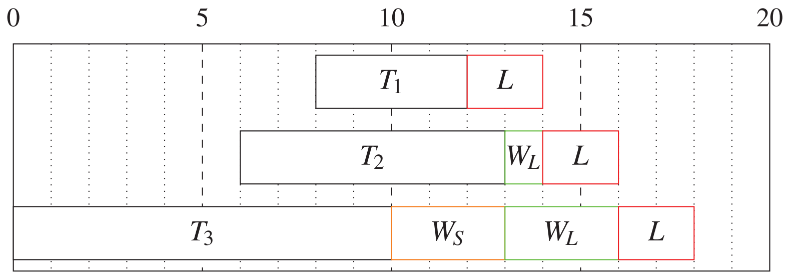


Figure 4: Retrieval scenario **b** = [3, 2, 1].

The Brute Force method evaluates all possible task assign sequences of the scheduled list of size N. Since this number is $N!$, the computational effort of Brute Force explodes as N grows. For large-sized systems, scheduled lists of size $N > 10$ yield good makespan performance, though require excessively long computation times (especially for PLC implementations). Therefore, a more efficient method is developed which is explained below.

3.2 Last Scheduled

Let us assume that the best sequence used for the previous arrival is $[i_1, i_2, \ldots, i_N]$. Hence, task i_1 is assigned to the shuttle and removed from the scheduled list. Then the next task i_{N+1} is added to the list $\{i_2, \ldots, i_N\}$

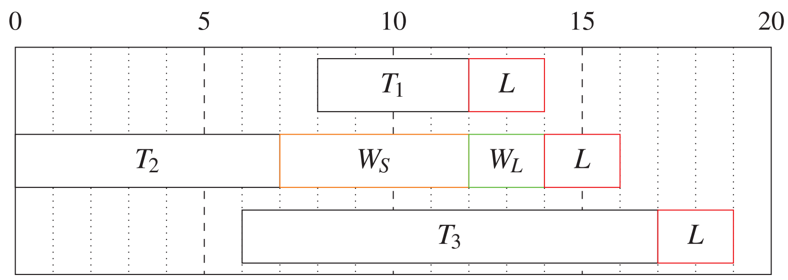


Figure 5: Retrieval scenario **c** = [2, 3, 1].

and time moves to the next arrival. To decide which task should be assigned to the next arriving shuttle, the Last Scheduled heuristic takes the sequence $[i_2, \ldots, i_N]$ as starting point and only evaluates the makespan of the sequences obtained by inserting task i_{N+1} at the N possible positions in sequence $[i_2, \ldots, i_N]$. The heuristic then selects the sequence with minimal makespan, and the first task in this sequence will be assigned to the next arriving shuttle. Clearly, this method is much more efficient than Brute Force, since it evaluates only N instead $N!$ sequences. However, it is possible that it omits task sequences with a significantly lower makespan.

To illustrate this method, we consider the example in Table 1 again, and suppose that [1, 3, 2] is the optimal sequence at time $t = 0$. Then task 1 is assigned to the shuttle arriving at time $t = 0$ and it is removed from the list. Task 4 is added, and to decide which task should be assigned to the shuttle arriving at time $a_1 = 6$, the Last Scheduled method calculates the makespan of three sequences [3, 2, 4] [3, 4, 2] and [4, 3, 2], from which the one with minimal makespan is selected.

3.3 Brute Force and Last Scheduled with Fixed Inter Arrival Times

The makespan calculations in Figures 3–5 are based on the actual arrival times of shuttles. In fact, these arrival times are *estimates* of the actual times, since in the shuttle cycle times, the stochastic component W_C is ignored and only the mean of L is used. Moreover, the complexity of the calculation of these arrival times increases as the number of shuttles k increases, and the calculation becomes unworkable, in the situation where multiple lifts share a pool of shuttles. Heuristics using estimated arrival times of the fixed pool of k shuttles in order to schedule tasks are referred to as closed-loop schedules. An easier and more robust approach to estimate the arrival times of shuttles is to assume a Fixed Inter Arrival Time (FIAT) of shuttles. This means that arrival time $a_i = i \times \text{FIAT}$. Both methods described above, i.e., Brute Force and Last Scheduled, can be equipped with the FIAT assumption for the calculation of the arrival times of shuttles. Such schedules will be referred to as FIAT schedules, and they can be used in closed-loop as well as pooled systems.

For the example in Table 1, Figures 6–8 illustrate the retrieval scenarios **a**, **b** and **b** based on the FIAT assumption with FIAT = 3 seconds. The makespan is 18 seconds for **a** and 16 seconds for both **b** and **c**.

4 SIMULATION MODEL

To evaluate the heuristics, we have developed a simulation model of the system shown in Figure 1. However, this simulation model describes the FRS system at a high level of abstraction: The travel time T, conflict time W_C and lift time L are described by i.i.d. random variables, the parameters of which are obtained from a detailed simulation model of the complex FRS system in Figure 1, developed by Vanderlande in the programming environment (AUTOMOD 2014). The high level simulation model has been developed in the simulation language (χ3.0 2014). In this simulation model we implemented an elegant algorithm due to (Fuchs 2016) to generate all permutations of the scheduled list (required for Brute Force), and we developed a fast algorithm to calculate the makespan of task assign sequences (required for each of the smart heuristics); the reader is referred to (Gootzen 2015) for a detailed description of this algorithm.

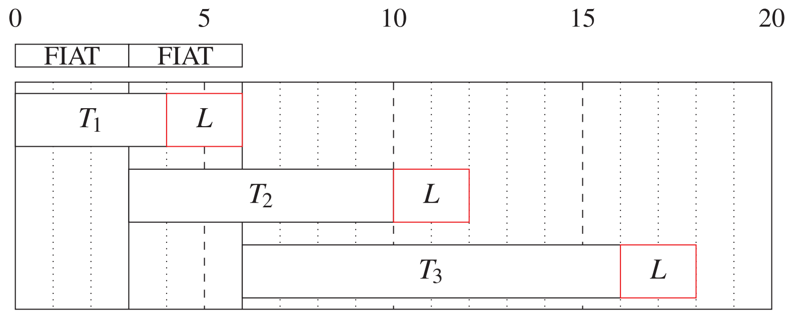


Figure 6: Retrieval scenario **a** with FIAT.

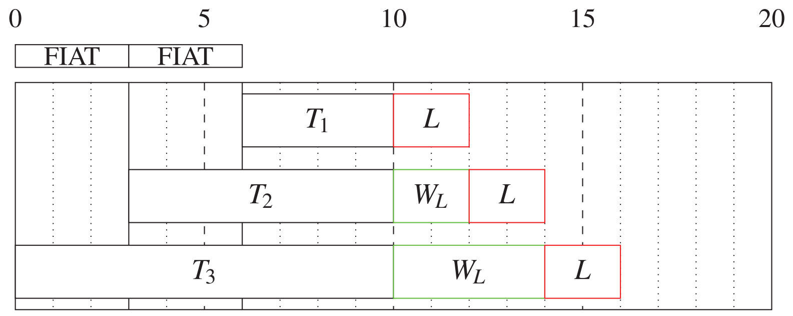


Figure 7: Retrieval scenario **b** with FIAT.

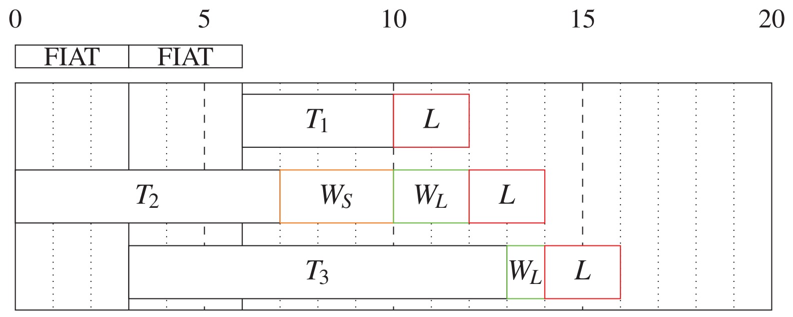


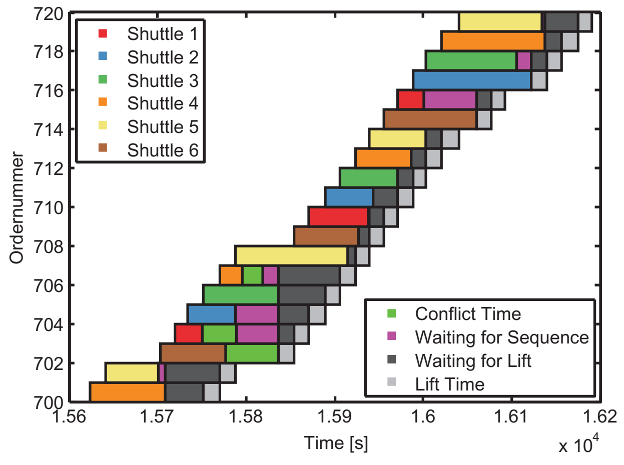
Figure 8: Retrieval scenario **c** with FIAT.

The FRS studied in this paper consists of 20 levels, with at each level, 10 storage aisles with a dept of D locations. The lift has k dedicated shuttles to retrieve and store product totes. The totes that have to be retrieved are randomly located in the storage area (i.e., at a random location in a random storage aisle at a random level). The travel time T, conflict time W_C and lift time L are modeled as follows:

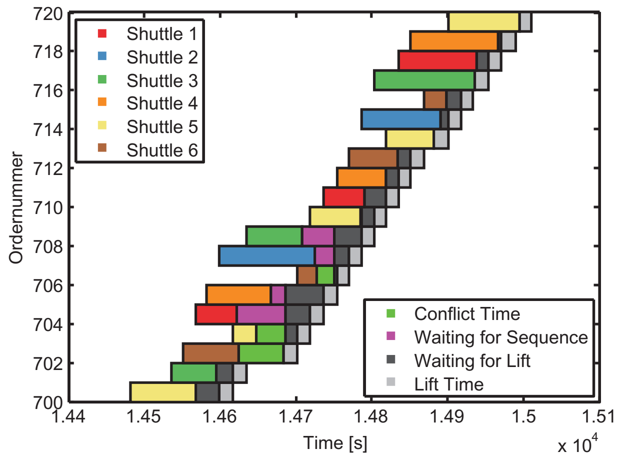
- Travel time T: The shuttle needs to travel in the cross-aisle and storage aisle. The travel time in the cross-aisle is assumed to be Uniform[0, 30] seconds. The travel time in the storage-aisle is Uniform[0, $0.625 \times D$] seconds. In addition, the travel time includes a fixed time of 18.5 seconds, describing the storage and retrieval of totes and the communication time with the system supervisor.
- Conflict time W_C: Not all shuttles experience conflicts during a cycle, but only a fraction p. Hence, $W_C > 0$ with probability p, and $W_C = 0$ otherwise. The conditional conflict time $W_C | W_C > 0$ is assumed to be Gamma distributed. The mean $E(W_C | W_C > 0)$ and coefficient of variation $c_{W_C | W_C > 0}$ of the conditional conflict time are estimated from data of the detailed simulation model.

- Lift time L: A lift cycle consists of loading and unloading of shuttles and traveling between levels. The fixed time of loading and unloading of shuttles is large in comparison to the travel time between levels. The total lift time is assumed to be Uniform[13.5, 19.5] seconds.

The high level simulations can be visualized with Gannt charts. Figure 9(a) shows the resulting cycle times of orders 700 up to 720 by using $k = 6$ shuttles operating under FIFO. The results of a smarter scheduling in Figure 9(b) clearly demonstrate that tasks are no longer assigned FIFO, resulting in a steeper graph and thus higher throughput. As explained in Section 3, in the simulations the scheduling heuristics do not take into account stochastic conflict times W_C and stochastic lift times L in order to estimate shuttle cycle times, i.e., the heuristics simply ignore W_C and use $E(L)$ instead of L.



(a) FIFO



(b) Smart Scheduling Heuristic

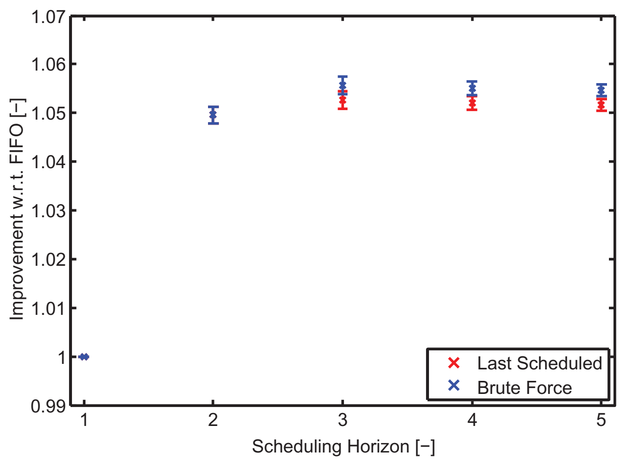
Figure 9: Gantt chart of retrieval scenario.

5 SIMULATION EXPERIMENTS

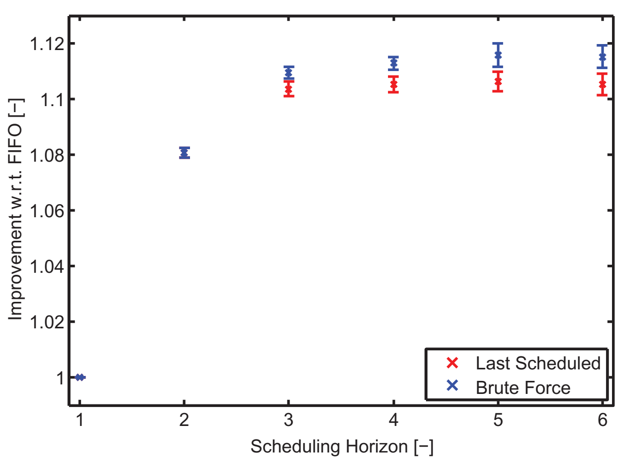
In all experiments presented in this section we generated a list of 20.000 tasks. To avoid startup symptoms the first 500 task completion times are not taken into account. Comparison of scheduling heuristics is always based on exactly the same list of tasks, generated by common random numbers (i.e., using the same seed values). The bars in the figures displayed in this section indicate 95% confidence intervals.

5.1 Performance of Closed-Loop Scheduling with no Conflicts

We start to conduct simulations assuming shuttles experience no conflicts at all (so $p = 0$). In the absence of conflicts, shuttle arrival times can be predicted accurately. Figure 10 shows the throughput improvement of Brute Force and Last scheduled based on closed loop scheduling and different scheduling horizons N. Figure 10(a) illustrates that for a small-sized system ($D = 80$, $k = 5$) a scheduling horizon $N = 2$ results in a throughput improvement of around 5.5% compared to FIFO. A longer scheduling horizon does not result in significantly higher throughput. Figure 10(b) shows that in a larger-sized system ($D = 145$, $k = 6$) a longer scheduling horizon will result in more throughput improvement. It also illustrates that in larger systems more throughput improvement can be achieved, 11% versus 5.5%. Both figures indicate that Brute Force performs slightly better than Last Scheduled. However, since the computational effort of Last Scheduled is substantially less, we will use Last Scheduled (LS) in the remainder of this section.



(a) Small-sized system: $D = 80, k = 5$.

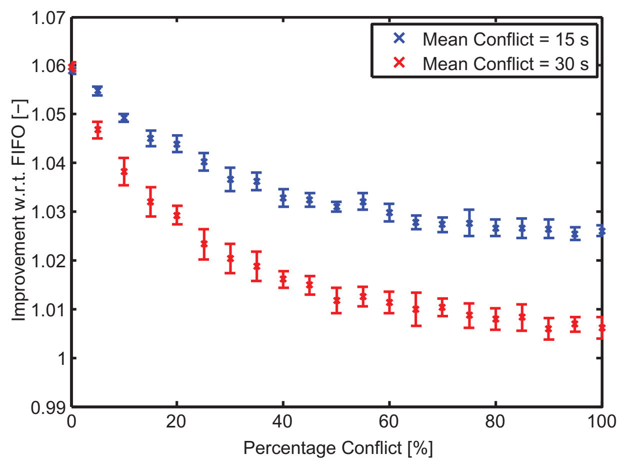


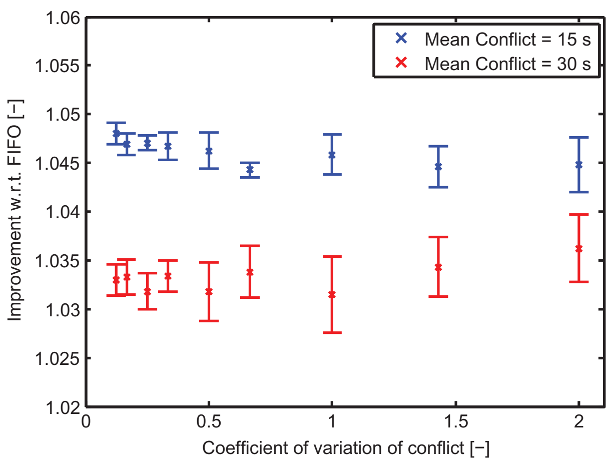
(b) Large-sized system: $D = 145, k = 6$.

Figure 10: Performance of closed-loop scheduling heuristics in the absence of conflicts.

5.2 Influence of Conflicts

For a small-sized system ($D = 80$, $k = 5$), Figure 11 illustrates the influence of conflicts on the throughput performance of the closed-loop LS heuristic with $N = 2$. Figure 11(a) shows the effect of the conflict probability p, for mean conditional conflict times $E(W_C|W_C > 0)$ of 15 and 30 seconds, respectively, and coefficient of variation $c_{W_C|W_C>0} = 1$ (so Exponential conflict times). Figure 11(b) demonstrates the effect of variability of conflict times, for conflict probability $p = 0.15$.





(a) Throughput gain as function of p for $c_{W_C|W_C>0} = 1$. (b) Throughput gain as function $c_{W_C|W_C>0} = 1$ for $p = 0.15$.

Figure 11: Effect of conflict time on throughput improvement under closed-loop LS.

As expected, Figure 11 shows that the more shuttles experience conflicts, i.e., the greater p, the less throughput improvement can be achieved with a smart scheduling heuristic. The same is true for the mean and variability of the conflict times.

5.3 Closed-Loop versus FIAT Scheduling

We now compare the throughput performance of closed-loop scheduling with FIAT scheduling. Instead of calculating the arrival times of shuttles at the workstation, as done in closed-loop scheduling, FIAT scheduling simply assumes a Fixed Inter Arrival Time of shuttles. Hence, this scheduling strategy uses less system information, and it has the advantage that it can also be applied in pooled systems. Figure 12 shows the throughput improvement of FIAT with respect to FIFO for various FIAT values in a system with $D = 145$ and $k = 6$. The scheduling heuristic is LS with horizon $N = 6$. The conflict parameters are $p = 0.33$, $E(W_C|W_C > 0) = 20.2$ seconds and $c_{W_C|W_C>0} = 1.098$. Figure 12 demonstrates that FIAT based scheduling outperforms closed-loop scheduling for FIAT values in the range of 20-40 seconds, and that its performance is fairly robust with respect to the choice of FIAT as long as it is not too small. Clearly, a lower bound for FIAT is the lift time (since shuttles cannot arrive faster), and Figure 12 suggests that $1.75 \times E(L) \approx 29$ seconds is a safe choice for FIAT.

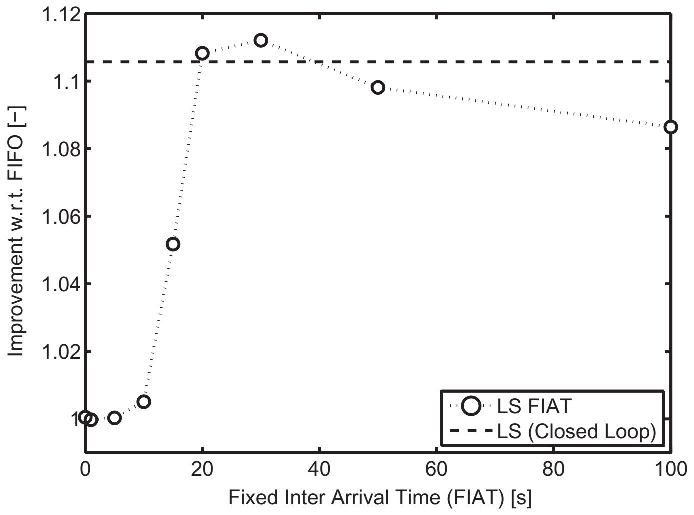


Figure 12: Throughput performance of FIAT and closed-loop LS scheduling.

5.4 Cycle Time Components

So far we focussed on throughput. Now we investigate the effect of smart scheduling on the waiting time components of the shuttle cycle, shown in Figure 2. For a system with $D = 145$ and $k = 6$, Table 2 shows the mean values of the cycle time components for FIFO and LS, which are based on $N = 6$ and FIAT = $1.75 \times E(L)$. For a system without conflicts (table at the left), the reduction of the mean out-of-sequence waiting time $E(W_S)$ is more than 70%, the total mean waiting time in front of the lift $E(W_S) + E(W_L)$ is reduced by 45% and the total cycle time is reduced by around 10%. For a system with conflicts (table at the right), the parameters of which are $p = 0.33$, $E(W_C|W_C > 0) = 20.2$ seconds and $C_v = 1.098$, the reduction of $E(W_S)$ is around 38%, $E(W_S) + E(W_L)$ is reduced by 25% and the total cycle time is reduced by around 6%. Hence, the benefits in case of stochastic conflicts are less, but still considerable.

Table 2: Cycle times (in seconds) of FIFO and LS, with (right) and without (left) stochastic conflicts.

	FIFO	LS
$E(T)$	78.79	78.79
$E(W_C)$	0.00	0.00
$E(W_S)$	10.19	2.95
$E(W_L)$	18.49	12.92
$E(L)$	16.48	16.48
Total	123.94	111.14

	FIFO	LS
$E(T)$	78.54	78.54
$E(W_C)$	7.10	7.10
$E(W_S)$	13.41	8.34
$E(W_L)$	19.48	16.55
$E(L)$	16.50	16.50
Total	135.04	127.04

6 CONCLUSION

In this study we developed scheduling heuristics that can potentially improve the throughput of an FRS system. The proposed heuristics estimate shuttle cycle lengths and arrival times over a rolling horizon and attempt to use this information to optimally assign tasks to arriving shuttles. Through a high-level simulation model we investigated the throughput performance of various scheduling heuristics. Simulation experiments suggest that: (i) smart scheduling heuristics indeed improve the throughput capacity, (ii) more improvement is possible for larger systems, (iii) variability in shuttle cycle times due to, e.g., interference delays, reduces throughput capacity, (iv) LS with FIAT is a simple, robust and well-performing scheduling heuristic, with the advantage that it can be used in both closed-loop and pooled systems.

REFERENCES

ADAPTO 2015. https://www.vanderlande.com/warehouse-automation/innovative-systems/storage-asrs/adapto/.

AUTOMOD 2014. http://www.appliedmaterials.com/global-services/automation-software/automod.

Bartholdi, J., and S. Hackman. 2014. *Warehouse and Distribution Science*. Release 0.96 http://www.warehouse-science.com/.

χ3.0 2014. *Documentation of the χ3.0 Language*. http://chi.se.wtb.tue.nl/.

de Koster, R., T. Le-Duc, and K. Roodbergen. 2007. "Design and Control of Warehouse Order Picking: A Literature Review". *European Journal of Operational Research* 182:481–501.

Fuchs, P. 2016. *Scalable Permutations: A Permutation of Agreeable Ideas*. http://www.quickperm.org/ScalablePermutations.php.

Gootzen, M. 2015. *Optimization of Task Scheduling in a Full Roaming Shuttle System*. Internship report, Eindhoven University of Technology.

Matthijssen, I. 2012. *Integration of Automated Item Picking in Multi Workstation Order Completion Goods to Man Systems*. Master thesis, Eindhoven University of Technology.

Tompkins, J., J. White, Y. Bozer, and J. Tanchoco. 2010. *Facilities Planning*. JohnWiley and Sons.

van Dorst, N. 2012. *Task scheduling in Full Roaming Shuttle Automatic Storage and Retrieval Systems*. Internship report, Eindhoven University of Technology.

AUTHOR BIOGRAPHIES

MARTIJN P. O. J. GOOTZEN is an M.Sc. student in Mechanical Engineering at the Eindhoven University of Technology. He has studied the simulation of scheduling heuristics as part of his master programm. His email address is mgootzen@gmail.com.

IVO J. B. F. ADAN is a Professor of the Department of Mechanical Engineering at the Eindhoven University of Technology. His current research interests are in the modeling and design of manufacturing systems, warehousing systems and transportation systems, and more specifically, in the analysis of multi-dimensional Markov processes and queueing models. His email address is i.j.b.f.adan@tue.nl.

JORINE W. E. HELING is Group Leader Systems Simulation at Vanderlande. The Systems Simulation group performs computer simulations on the automated material handling system designs of Vanderlande in order to validate system performance. Her email address is Jorine.Heling@vanderlande.com.

BRUNO VAN WIJNGAARDEN is System Architect at Vanderlande. His current activities are requirements management, system design support, product platform and system architecture development and knowledge management. His e-mail address is bruno.van.wijngaarden@vanderlande.com