2014

Real-time Dock Door Monitoring System Using a Kinect Sensor

Hector J. Carlo
UPR- Mayaguez, hector.carlo@upr.edu

Cristina Pomales- Garcia
UPR- Mayaguez, cristina.pomales@upr.edu

Yeiram Martinez
UPR- Mayaguez

JOB SEQUENCING IN A MINILOAD SYSTEM

Lennart Baardman
Operations Research Center, MIT, Cambridge

Kees Jan Roodbergen
Faculty of Economics and Business, University of Groningen

Héctor J. Carlo
Industrial Engineering Department, University of Puerto Rico

Abstract

More and more warehouses shift towards the use of automated systems. One such system is a miniload system. In this system, a crane retrieves bins from their storage locations and brings the bins to the picking station, where a worker takes the requested products from the retrieved bins. After a product has been picked, the bin is put back by the crane in its original location. A buffer is present at the picking station to absorb fluctuations in speed between the picker and the crane. The problem of scheduling jobs in this system has received little attention in the literature. We describe system properties and give insight in the performance of a number of heuristics.

1 Introduction

Traditionally, many warehouses have been using manual order picking systems, since these require a lower investment. Furthermore, manual systems are appreciated for their flexibility. For example, in peak periods, it is possible to increase throughput by increasing the number of workers, provided the picking area is large enough and provided sufficient pick carts are available. The major drawback of manual picking systems, however, is the high operational cost due to the, potentially high, number of workers required. Despite the introduction of newer systems, like shuttles and Kiva, more traditional automated systems such as miniloads are a viable option for automating warehouses as well.

The miniload system we consider can be described as follows. It consists of an aisle with storage locations on both sides and a picking station at the end of the aisle. More aisles and more picking stations can be combined to form a larger system. The crane starts at the I/O-point, moves to a storage location, retrieves a bin from that location, and brings the bin to the I/O-point. At the I/O-point, the bin enters a horseshoe conveyor that takes the bin to the worker. The worker retrieves one or more items from the bin in response to a customer order, after which the bin continues on the horseshoe conveyor, and returns to the I/O-point. After some time, the crane picks up the bin from the I/O-point and puts it back at its original storage location.

The system is designed such that the actions of the crane and the worker need not be synchronized. The horseshoe conveyor functions as a buffer between the crane and the worker.

Hence, the crane can be retrieving a new bin and can deliver this bin at the I/O-point while the worker is retrieving items from another bin. We assume the horseshoe conveyor to have a capacity of m bins. This implies that once there are m bins on the horseshoe conveyor, first the oldest bin will have to be removed by the crane before a new bin can be retrieved from the rack. When the system is in use, this will result in dual-command cycles for the crane. That is, a tour of the crane starts at the I/O-point, where a bin is picked up for storage in the rack. This is a bin from which recently items have been taken by the worker. The bin is stored in one of the rack locations. Next, the crane moves empty to another location, to retrieve a bin that is needed for an order. The crane moves back to the I/O-point with the bin. Thus each cycle of the crane consists of two actions, one storage request and one retrieval request. A graphical presentation of the system with a buffer size of 2 is given in Figure 1.

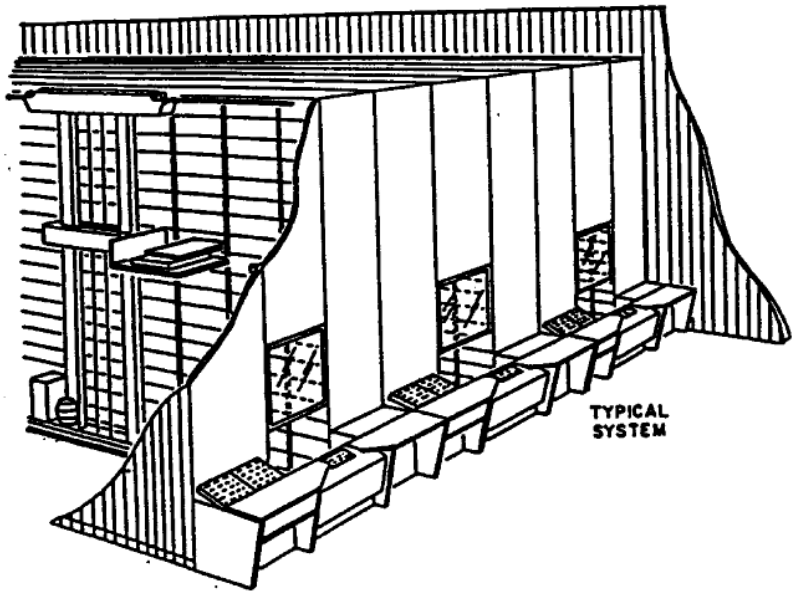

Figure 1: Graphical representation of a miniload system consisting of three aisles, with a crane in each aisle, and a picking station at the end of each aisle. The size of the buffer is exactly 2 bins at each picking station in this figure (one to the left and one to the right of the aisle). Alternatively, a horseshoe conveyor may provide more buffer capacity at the pick station. This figure was previously published in [1] and [2].

To ensure the system functions efficiently, it is vital to distinguish between the system as a whole, and its two components, the worker and the crane. Optimizing only the crane or only the worker could cause suboptimal solutions for the system as a whole. We therefore focus on total system performance, while also studying the role of pure crane scheduling within this context. Some aspects of the system are not taken into account in this paper. Most notably, we assume that a bin is returned to its original storage location after items have been taken from it by the worker. It may be possible to achieve additional efficiency gains by optimizing the locations for returned bins. However, we aim to optimize the system during its most critical period, i.e., when the system needs to achieve it maximum throughput. This may, for example, be in the last few hours before an

order cut-off time. To be able to achieve maximum output, bins in the system must be positioned at the most appropriate locations beforehand. This prepositioning can be done during off-peak periods, without impacting throughput times. Hence, bins are in this situation best returned to their original storage location.

To the best of our knowledge, previous research on this topic is limited to only three articles [1], [2], and [3]. In [1] and [2], a miniload system is studied in the exact setting as described above, only the buffer size is fixed at 2. In [3] the buffer can have any size, as we consider here. Both [1] and [2] present constructive heuristics for scheduling the retrieval sequence. In [3] a simulated annealing heuristic is presented. Other related research concerns the estimation of performance by means of analytical methods [1], methods to determine appropriate storage locations for products [4], and methods to determine a dwell-point for the crane [5].

The remainder of this paper is organized as follows. In Section 2, we present a formal description of the system and a linear integer programming formulation. In section 3, we describe properties of the model. Heuristics are described in Section 4, computational insights in Section 5, and conclusions in Section 6.

2 Problem definition

We have a set of n requests for bins from which items must be taken. These bins must be retrieved from their respective storage locations, brought to the I/O-point -and after the appropriate items have been taken from the bins- returned to their storage locations. The buffer (horseshoe conveyor) can contain m bins. We assume that the number of bins in the buffer is constant at m. This assumption is logical from a practical point of view, since when a system must achieve maximum performance, it must be operating with dual-command cycles. This implies that for every bin that is removed from the buffer, a new one is retrieved in the same cycle, hence resulting in a constant number of bins in the buffer. Since processing of the bins by the worker does not alter the sequence, a direct consequence is that a bin will have to be included as a storage request exactly m cycles after it was retrieved. Or stated differently, the sequence of storage requests is determined uniquely by the sequence of retrieval requests. This is consistent with the assumptions in [1], [2], and [3].

We present a mathematical description of the problem below. The objective is to minimize the makespan for a given set of requests. This is an alternative formulation for the model presented in [3]. The formulation presented here takes advantage of the structural properties of the crane's Chebychev travel by using the linearization strategy for the rectilinear Quadratic Assignment Problem from [6] and then selecting the maximum of the horizontal and vertical distances. This model, which unlike the model in [3] is limited to Chebychev travel, is outperformed by the formulation presented in [3] in most test instances. However, given that no other mathematical formulations for this problem exist, we opted to include the formulation in this paper.

Structural parameters and variables:

n Number of requests

m Buffer size at picking station

i, j Indices for the requests ($i, j = 1, \ldots, n + m$)

k Index for cycles ($k = 1, \ldots, n + m$)

d_{ij} Distance (or time) between the storage locations of request i and j

α_i Horizontal coordinate of centroid of storage location of request i

β_i Vertical coordinate of centroid of storage location of request i

[illegible] 1 if request i is retrieved in cycle k; and 0 otherwise

Model representation:

$$\min Z = \sum_{k=1}^{m} u_k + \sum_{k=m+1}^{n} w_k + \sum_{k=n+1}^{n+m} v_k \tag{1}$$

$$\text{such that } u_k \geq \sum_{i=1}^{n} \alpha_i x_i^k \qquad k = 1, ..., m \tag{2}$$

$$u_k \geq \sum_{i=1}^{n} \beta_i x_i^k \qquad k = 1, ..., m \tag{3}$$

$$w_k \geq (a_k^+ + a_k^-) \qquad k = m+1, ..., n \tag{4}$$

$$w_k \geq (b_k^+ + b_k^-) \qquad k = m+1, ..., n \tag{5}$$

$$v_k \geq \sum_{i=1}^{n} \alpha_i x_i^{k-m} \qquad k = n+1, ..., n+m \tag{6}$$

$$v_k \geq \sum_{i=1}^{n} \beta_i x_i^{k-m} \qquad k = n+1, ..., n+m \tag{7}$$

$$a_k^+ - a_k^- = \sum_{i=1}^{n} \alpha_i x_i^{k-m} - \sum_{i=1}^{n} \alpha_i x_i^k \qquad k = m+1, ..., n \tag{8}$$

$$b_k^+ - b_k^- = \sum_{i=1}^{n} \beta_i x_i^{k-m} - \sum_{i=1}^{n} \beta_i x_i^k \qquad k = m+1, ..., n \tag{9}$$

$$a_k^+, a_k^-, b_k^+, b_k^- \geq 0 \qquad k = m+1, ..., n \tag{10}$$

$$\sum_{i=1}^{n} x_i^k = 1 \qquad k = 1, ..., n \tag{11}$$

$$\sum_{k=1}^{n} x_i^k = 1 \qquad i = 1, ..., n \tag{12}$$

$$x_i^k \in \{0, 1\} \qquad i, k = 1, ..., n \tag{13}$$

The Chebychev metric is enforced by Equations (2) and (3) for the distance from the I/O-point to retrieval locations. In Equation (2) the Chebychev travel distance is ensured to be equal or larger

than the horizontal travel from retrieval location *i* to the I/O-point, while in Equation (3) it is ensured that the Chebychev travel distance is equal to or larger than the vertical travel from location *i* to the I/O-point. Similarly, Equations (6) and (7) ensure the Chebychev travel distances are used for the path from the I/O-point to the storage location, while Equation (4), (5), (8) and (9) take care of the Chebychev travel between the storage location and the retrieval location. Equations (11) and (12) require that each storage request and each retrieval request is uniquely assigned to a single cycle *k*. That is, each storage request and each retrieval request is executed exactly once, and each cycle *k* contains exactly one storage request and one retrieval request. Non-negativity constraints and binary requirements are given by Equations (10) and (13).

3 Properties

In [3] the request sequencing problem for a miniload system is compared to the multiple Traveling Salesmen Problem (mTSP). The mTSP considers *n* locations that must each be visited exactly once by one of the *m* salesmen. In the setting of this basic description, actually it would be optimal to use only one salesman, and leave the other *m*-1 salesmen idle. Therefore, in the most common presentation of the problem there are two bounds present. The upperbound *U* specifies the maximum number of locations that may be visited by any one salesman, and the lowerbound *L* specifies the minimum number of locations that must be visited by any salesman. A special case of this problem, is the multiple Traveling Salesmen Problem with equal visits (mTSPEV) as introduced in [3]. This problem is achieved by setting $L = U = n/m$. Note that the mTPEV is technically speaking only defined if n/m returns an integer value. However, when n/m is not integer, it is straightforward to introduce *t* dummy locations with coordinates equal to the depot's coordinates such that $(n + t)/m = \lceil n/m \rceil$ without altering the objective value.

As it appears, the mTSPEV has several properties that distinguishes it from other routing problems. Most notably we have the following property.

Property 1 (see [3]). There exist instances of the mTSPEV for which it holds that the optimal total tour length strictly decreases if extra locations are added to these instances.

This property is special since normally in routing problems, adding extra locations to an instance would either result in an increase in the total tour length, or -at best- results in the tour length remaining equal. For the mTSPEV, however, the total tour length may be strictly decreasing when adding locations.

4 Solution methods

In this section we describe three heuristics and provide a comparison of their performance. The first heuristic we describe here originates from [1] and is based on the nearest neighbor strategy. The second heuristic described below is taken from [2], while the third heuristic is from [3]. In this paper, so far we have assumed that the crane never has to wait for the picker. This assumption can be realistic for situations in which picking times of the worker are negligible compared to the crane operating times. Also, when the buffer is sufficiently large, the probability of waiting by the crane is small, in which case the model without waiting times is appropriate. In other cases, it may be useful to explicitly include waiting times of the crane and the worker when calculating makespan.

Formulas for this are available in [3]. The heuristics described in this paragraph are all capable of finding solutions for settings with and without waiting times.

4.1 The Nearest Neighbor strategy

This strategy was first published in [2] for a miniload system with a buffer size of $m = 2$. Later this was extended for any m in [3], which is the method we describe here. We start with an empty buffer. The first retrieval request to be on the list is the request for which the storage location is closest to the I/O-point. The second retrieval request, is the request for which the storage location is second closest to the I/O-point. Etcetera. This continues until we have m retrieval requests on the list. At this point the buffer is full, and in any following cycle, first a bin must be taken from the buffer and stored in its original storage location. These storage request are fixed by the retrieval sequence, hence they cannot be changed any more at this stage.
In cycle m+1, we store the bin that was retrieved in the first cycle. Generally, in cycle f, we store the bin that was retrieved in cycle f-m. This storage request will be combined with a retrieval request to form a dual-command cycle. We let each storage request be followed by the retrieval request with its location nearest to that storage request (to be selected from the retrieval requests that have not been scheduled yet).

4.2 The BW heuristic

As with the nearest neighbor strategy this heuristic was first published for systems with a buffer of size 2 only (see [1]). The adapted version for any buffer size as given in [3] is described here. Again we assume to start with an empty buffer. First, we randomly select m request from the total set of requests to serve as the first m requests in the sequence. Now define $t_{(i)j}$ as the time needed by the crane for storing load i (numbered according to the sequence newly determined by this heuristic; hence the brackets in the subscript) and retrieving load j (numbered according to the original numbering, that served as input for the heuristic). And define $p_{(i)}$ as the time the worker needs to take items from bin i in the new sequence. $C_{(i)j} = \max\{0, t_{(i)j} - p_{(i)}\}$ serves as a measure for worker idle time. Now, the next retrieval request will be request j, chosen from the non-handled requests, such that j minimizes $C_{(i)j}$. This process is continued until all requests have been scheduled.

4.3 Simulated annealing heuristic

This method is based on the simulated annealing scheme presented in [7], and the description we give here follows the description given for the miniload scheduling problem in [3]. The local search procedure used is based on pairwise exchange of requests. We only consider exchanges of retrieval requests. This suffices since the corresponding storage request must be scheduled in cycle i if the retrieval request was scheduled for cycle i-m. Solutions are accepted by definition if they are better than the current solution, and are accepted otherwise with the Boltzmann acceptance probability. The nearest neighbor solution serves as the initial solution.

4.4 Optimal solutions and lower bound

Optimal solutions can be obtained from solving the integer linear programming formulation given

in Section 2, or from the model from [3] in CPLEX. However, since the problem is NP-hard, it is to be expected that only small instances can be solved to optimality this way. To benchmark heuristic performance, a lowerbound can be used. For this purpose, the integer linear programming model can be converted to a linear programming formulation by lifting the binary restrictions on appropriate variables. Two additional sets of constraints are to be added to the model to ensure that the variables will be larger than or equal to 0, as well as smaller than or equal to 1.

5 Computational insights

In this section, we present the results of comparisons of the various solution methods. First, we analyze the system, assuming that there are no waiting times for the crane. Table 1 present a performance comparison between the nearest neighbor strategy and optimal solutions, as well as a comparison between the simulated annealing heuristic and optimal solutions. Half of the instances are for $n = 20$, the other half for $n = 30$. Locations were randomly generated on a rack with dimensions 20x20.

Table 1: Percentage difference between respectively the nearest neighbor strategy (NN) and optimal solutions (column 2) and the simulated annealing heuristic and optimal solutions (column 3) for different values of m.

m	NN	SA
2	24.6%	0.8%
3	26.3%	0.7%
4	27.2%	0.5%
5	26.6%	0.1%

As can be seen from Table 1 the simulated annealing heuristic [3] outperforms the nearest neighbor strategy [2] significantly. We did not include the BW heuristic [1] in these experiments since the BW heuristic is specifically designed for situations with waiting times. Next, in our experiments we allow for waiting times of the crane. Since now we have no optimal solution available, we compare performance of the nearest neighbor strategy and simulated annealing heuristic to the BW heuristic in Table 2.

Table 2: Percentage difference between respectively the nearest neighbor strategy and the BW heuristic (column 2) and the simulated annealing heuristic and BW heuristic (column 3) for different values of m.

m	NN	SA
2	-6.1%	-9.5%
3	-5.4%	-9.8%
4	-3.9%	-8.7%
5	-2.4%	-7.8%

Table 2 demonstrates that the BW heuristic is outperformed by both the nearest neighbor strategy as well as the simulated annealing heuristic. The nearest neighbor strategy is on average 4.5% better than the BW heuristic and the simulated annealing heuristic 9.0%.

6 Conclusions

We have revisited the problem of scheduling requests for a miniload system where bins must be placed back after picking at the same location they were retrieved from. This problem was previously studied in [1] and [2], but only for a buffer size of 2 bins. In [3] the setting was extended for any size buffer. We discussed in this paper the analogy of the problem of sequencing request in a miniload with the multiple Traveling Salesman Problem with Equal Visits (mTSPEV), which was first introduced in [3]. The mTSPEV can be seen to be equivalent to the miniload request sequencing when there is no waiting time for the crane. In the computational insights section, we presented comparisons that demonstrate that the simulated annealing method of [3] significantly outperforms the heuristics from [1] and [2] in all situations considered. Furthermore, for situation without crane waiting times, it can be seen that the simulated annealing heuristic performs close to optimality.

References

[1] Bozer, Y. A. and White, J. A., "A generalized design and performance analysis model for end-of-aisle orderpicking systems," *IIE Transactions*, 28, 4, 271-280 (1996).

[2] Mahajan, S., Rao, B. V. and Peters, B. A., "A retrieval sequencing heuristic for miniload end-of-aisle automated storage/retrieval systems," *International Journal of Production Research*, 36, 6, 1715-1731 (1998).

[3] Baardman, L., Roodbergen, K. J. and Carlo, H. J., "A special case of the multiple traveling salesmen problem in end-of-aisle picking systems," working paper (2016).

[4] Chen, L., Langevin, A. and Riopel, D., "A tabu search algorithm for the relocation problem in a warehousing system," *International Journal of Production Economics*, 129, 1, 147-156 (2011).

[5] Van den Berg, J. P., "Analytic expressions for the optimal dwell point in an automated storage/retrieval system," *International Journal of Production Economics*, 76, 1, 13-25 (2002).

[6] Bozer Y. A. and Carlo H. J., "Optimizing inbound and outbound door assignments in less-than-truckload crossdocks," *IIE Transactions,* 40, 11, 1007-1018 (2008).

[7] Eglese, R. W., "Simulated annealing: A tool for operational research," *European Journal of Operational Research*, 46, 3, 271-281 (1990).