A High-Density, Puzzle-Based Order Picking System

Kevin R. Gue and Onur Uludağ
Department of Industrial & Systems Engineering
Auburn University, Auburn, AL, USA

Abstract

We present a new semi-automated system for carton and piece picking operations in the distribution centers. The system features decentralized control based on a message-passing protocol in which cells communicate with their neighbors to determine action in each step. The result is a dynamically changing pick face that provides high sku density and reduced worker travel. We describe an example in which pickers are 20% to 50% more productive than they would be in an equivalent flow rack system.

1 Background and Motivation

Order picking is widely believed to be the most expensive task in distribution center operations, primarily due to the labor required to pick, pack, and assemble orders. Because pallet quantities are the most economical to handle and transport by truck and other forms of transportation, goods in a distribution center (DC) typically arrive on pallets. Yet, quantities shipped from a DC are often less than a pallet quantity, or several less-than-pallet quantities are picked and shipped together as a pallet quantity.

The unavoidable consequence of receiving in pallet quantities and shipping in smaller quantities is the need to pick cartons from pallets or items from cartons—a task most often accomplished by humans. A typical carton picking method is to use the bottom slots in a pallet rack as pick locations and upper slots as reserve. Each pallet location holds one sku. Order pickers travel throughout the space picking cartons off the pallets onto a shipping pallet. A typical piece picking method is to use flow rack. Again, each slot contains a single sku, and pickers traverse the space picking items into a tote or other container.

The theory behind both carton picking and piece picking operations is to increase the density of skus on the pick face, and therefore to reduce the amount of travel required between pick locations. Access to more skus per linear meter means orders can be gathered more quickly, which makes workers more productive.

In this paper, we introduce the GridPick system as an alternative to existing carton and piece picking systems. GridPick uses a "puzzle-based architecture" to increase sku density beyond one per slot, and therefore offers the potential to reduce travel distances for workers assembling orders.

2 Literature Review

Order picking is traditionally performed manually and workers travel within the aisles to pick from bin shelves, storage rack, flow rack, etc. In manual order picking, walking or travel time is the largest component of processing time [14, 17]. For this reason, heuristics and exact methods have been developed to build routes for order pickers [4, 11, 13, 14].

Automated Storage and Retrieval Systems (AS/RS) improve order picking productivity at the cost of the automation. Travel time models and methods of performance analysis for AS/RS have been introduced by Bozer and White [2] and Lee [10]. Travel time models for multiple shuttles and alternative configurations have also been explored [12, 15, 19].

High-density storage and retrieval systems have been studied by van den Berg and Gademann [18]. For a multi-deep AS/RS, travel time models and optimal system dimensions are described in Hu et al. [9] and de Koster et al. [3]. Yu and de Koster [20] extended the analysis of a multi-deep AS/RS by providing optimization of zone boundaries for class-based storage.

The system presented in this paper is based on a "puzzle-based storage system," which was introduced by Gue and Kim [8]. Material movement in a puzzle-based storage system requires that each cell be capable of conveying in the four cardinal directions. Alfieri et al. [1] consider the case of restricted puzzle-based movement, in which a limited number of vehicles move items among cells. Furmans et al. [6] also introduce a restricted puzzle based storage system with one empty cell, one robot and one Input/Output (I/O) point. Taylor and Gue [16] extended the puzzle based storage system by providing retrieval time performance analysis. Gue and Furmans [7] introduced a high-density storage system called GridFlow, which extends the puzzle-based paradigm to allow multiple empty cells and the simultaneous retrieval of multiple items to an exit row. Furmans et al. [5] describe the "plug and work" material handling paradigm, of which GridFlow is an example.

3 The GridPick System

We consider the problem of carton picking from pallets or piece picking from cartons to build orders in a distribution center. Current practice is to pick pieces from a forward storage area consisting of flow rack, for example, which increases the density of skus per meter of walking (Figure 1). Workers are more productive because they travel less between picks. Flow rack, bin shelving, and most other technologies used as a forward picking area have the limitation of one sku per slot or location in the pick face.

GridPick is based on the grid-based material handling model proposed in Gue and Furmans [7], in which items move in the four cardinal directions on a grid of

Figure 1: A flow rack representation, flow racks can have very long aisles in a large number of skus configuration.

unit-sized conveyor modules. Each module executes its control logic and decides what to do depending on its condition and the condition of its neighbors. Therefore, control is *decentralized*, which allows the system to be completely modular, flexible, and scalable. Items move to the pick face using the same logic for retrievals in the GridFlow system described in Gue and Furmans [7] (see Figure 2).

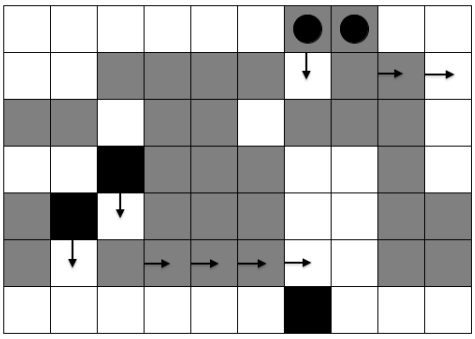

Figure 2: GridFlow for the retrieval of requested (items in black) and travel of replenishing items (gray, with black circles). Requested items move downward to leave the system; replenishing items move downward to their appropriate home rows. Interfering items (gray) move to the left and right. The GridPick system uses similar, but not identical, movement.

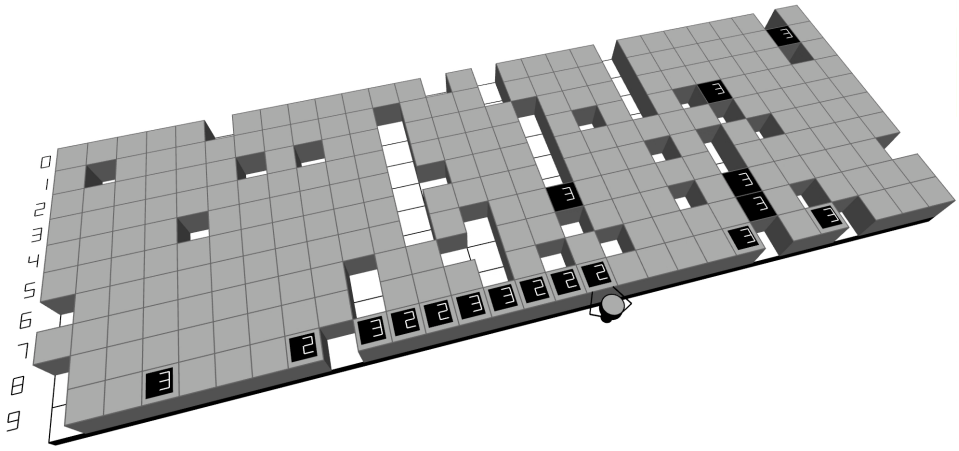

Figure 3: GridPick for carton picking from pallets.

3.1 System Description

The decentralized control scheme is based on an Assess-Negotiate-Convey cycle. In the Assess phase, each conveyor module determines its state (empty? occupied? etc.) and relays it to its four neighbors. The Negotiate phase consists of message passing between neighbor modules, in which modules communicate their needs ("I've got an item that needs to move to the pick face") and their ability to accommodate ("I am empty and can receive an item from the left"), among other things. At the end of the Negotiate phase, each module has decided whether or not it will convey, and if so, in which direction. In a carton-picking operation, for example, grid-based material movement allows the pick face to change dynamically, in response to skus required for the current order or orders soon to be picked. Therefore, each pallet position in the pick face effectively accommodates more than one sku, thereby increasing sku density and reducing worker travel.

Figure 3 illustrates a GridPick system for picking cartons from pallets. Gray pallets are stored and not needed in the current or next order. Black pallets contain skus for the order number indicated on top. As soon as all pallets for an order reach the pick face, pallets for the next order "activate" and begin to make their way to the pick face. The goal is to have all pallets for an order at the pick face before the worker begins picking that order.

3.2 The Algorithm

A detailed description of the GridPick algorithm is beyond the scope of this paper. Here we provide an overview of the control logic, and then illustrate the system with an example.

Overview: Each conveyor module contains and executes the same control logic in each iteration, but may take a different action depending on its local conditions (e.g., empty vs. occupied).

At each iteration there is an order currently being picked. Any items in the current order not at the pick face take on a "requested" state. In practice, the module containing that item would be told from a central authority that its item must move to the pick face. To maintain liveness (avoidance of deadlocks) in the system, downward movement of a requested item must be matched by the upward movement of an item not in the current order. We call an upward moving item a *balancing item*. Thus, in the long run, each row maintains a constant number of empty cells, which can be used to move interfering items out of the way of requested items.

It is possible that a balancing item meets a requested item—the one attempting to move up, the other down. In this case, we give balancing items priority, because their ability to maneuver affects the liveness of the system; whereas, the ability of a requested item to reach the pick face affects only the performance of the system. In the case of conflict, the requested item moves out of the way of the balancing item.

Rebalancing and movement to the pick face are accomplished by assigning and reassigning target rows. At each iteration, modules check the target row of their items; if the item's target row does not equal the row of the module, then events are initiated to move items to their target rows.

Key events in the GridPick algorithm are:

ReceiveReplenish: This event triggers the upward movement of a stored item in the below row when a requested item has not yet moved from its departure row. When the item on a module is first requested, it sends a message left and right seeking an empty module. When an empty module receives the message, it forwards it to its south neighbor. If the south neighbor is occupied with a stored (not moving) item, it responds with a willing message, which propogates back to the original requesting module. The requesting module responds on a first come, first served basis, which initiates the balancing movement by reassigning target rows: the willing southern item updates its target row to the row above it, and the downward requested item updates its departure row to the row below it. This process continues at each step as the requested item moves downward. Thus, at each step the downward moving item "fills the hole it left behind," and the system maintains its balance.

ClimbReplenish: In some cases, the module with a requested item is not able to find a stored item willing to move up in the *ReceiveReplenish* event. In this case, the requested still moves down, but keeps its original departure row. At each iteration, it attempts again to find an empty cell with a willing southern neighbor. When it finds one, the action is the same as *ReceiveReplenish*, except

Figure 4: Movement of a regular item to balance downward movement of the requested item.

that the balancing item will seek a row higher than the one immediately above it.

Tandem Movements: Ordinary implementation of the events above lead to a "sequential" rebalancing, in which balancing items leave holes behind them, which in the best case are then filled by further balancing items. However, each time an empty cell is created, it becomes a source of competition for other uses—namely, east-west movement—and this can delay rebalancing and system throughput. To address this issue, GridPick contains a message protocol that identifies tandem movements of rebalancing items, in which an entire block of stored items moves upward in one iteration, thereby achieving rebalancing slightly "ahead of time."

North-South and East-West Negotiations: These two negotiation phases executed in each iteration consecutively. The North-South negotiation facilitates the downward and upward movement of the requested and balancing items respectively. Negotiations are identical to that in GridFlow [7].

Convey Phase: At the conclusion of all message-passing and negotiations, each module knows whether or not it is to convey and if so, in what direction. Conclusion of the convey phase initiates another cycle.

Figure 5 shows several iterations of the GridPick algorithm. In the first iteration, items in Order 3 have already arrived, and the picker is processing Order 3 from left to right. At the same time, items for Order 4 have started to move down to the pick face. Items with a black dot are moving up to balance the downward movement of requested items. For instance, in the second iteration, the requested item in the eighth row moves down, and at the same time a balancing item in the ninth row moves up. Stored items move left and right to allow requested and balancing items to move. Requested items in the current order can also move right or left to allow downward movement of the next order's items.

In the third and fourth iterations, on the left side of the grid, balancing items move up and the requested item on the far left is able to move down after them in the

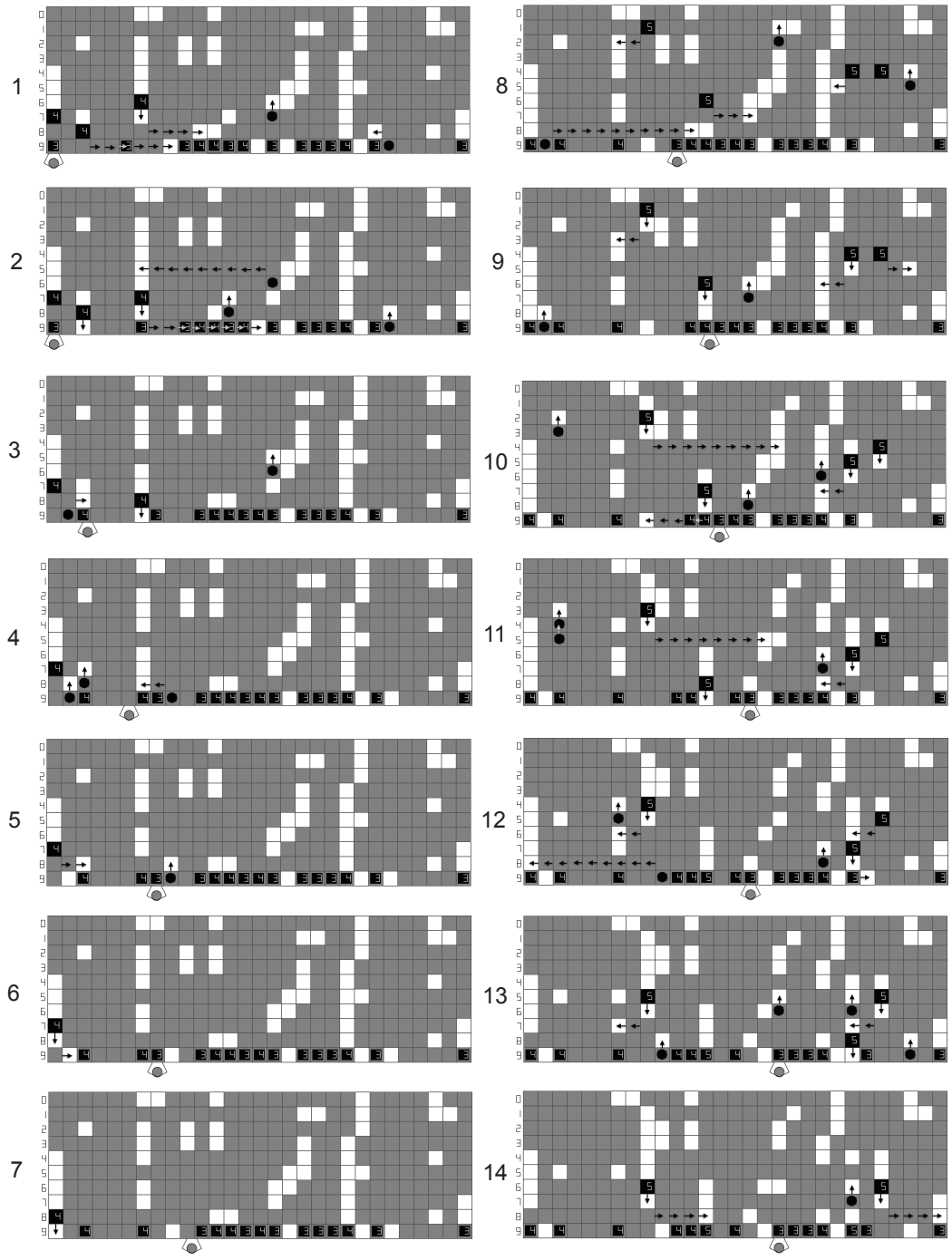

Figure 5: An example of the GridPick algorithm. Iterations proceed from top to bottom, left to right.

fifth iteration. Balancing items have priority over requested items to avoid possible congestion. In the eighth iteration, all items of Order 4 have reached the pick face; Order 5 is released and its items start their travel to the pick face.

4 Analysis

By increasing the effective sku density of the pick face, a GridPick system reduces the travel required to gather items for an order. To assess the level of benefit, we compared a GridPick system with an equivalent flow rack. Figure 6 shows a representation of the two systems. Note how much longer is the pick face for the flow rack system.

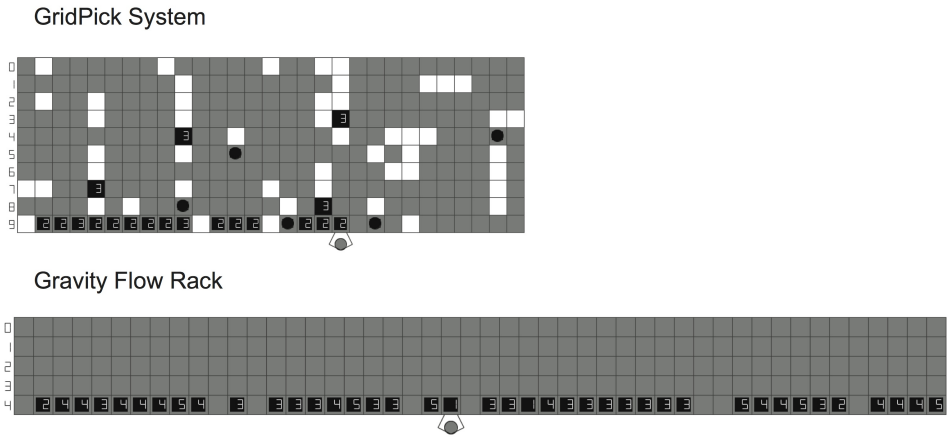

Figure 6: Comparison of the GridPick system and an equivalent flow rack.

Both systems contain 5 pallets each of 48 skus, for a total of 240 pallets. The nature of flow rack obliges a length of 48 slots, whereas this implementation of GridPick (other aspect ratios could be considered) has 29 slots. The GridPick system requires empty cells for each row to be able to execute puzzle-based movement. This configuration of the GridPick system has 10 rows, 29 columns, and 5 empty cells per row.

Figure 7 illustrates the average productivity improvement (in picks per hour) for a worker in a GridPick system. The level of benefit is higher for smaller orders because the ratio of travel time to pick time is greater in this case; because GridPick significantly reduces travel time (pick time for an order is fixed), it offers more benefit for smaller batches.

Figure 8 shows the difference in average walking time between picks for GridFlow and an equivalent flow rack. As order size increases, walking time per pick converges because workers in both systems walk very little between picks and spend most of their time picking. Order sizes of 5–20 are typical of most applications in industry.

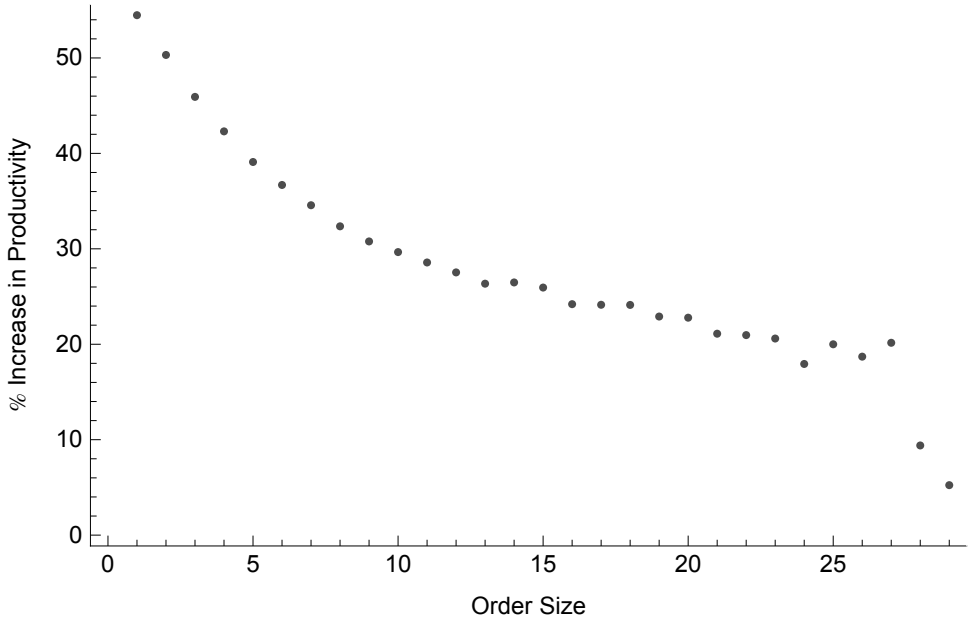

Figure 7: Estimated productivity improvement for GridPick over an equivalent flow rack system. Improvement falls off rapidly when the expected order size is greater than the number of slots in the GridPick system.

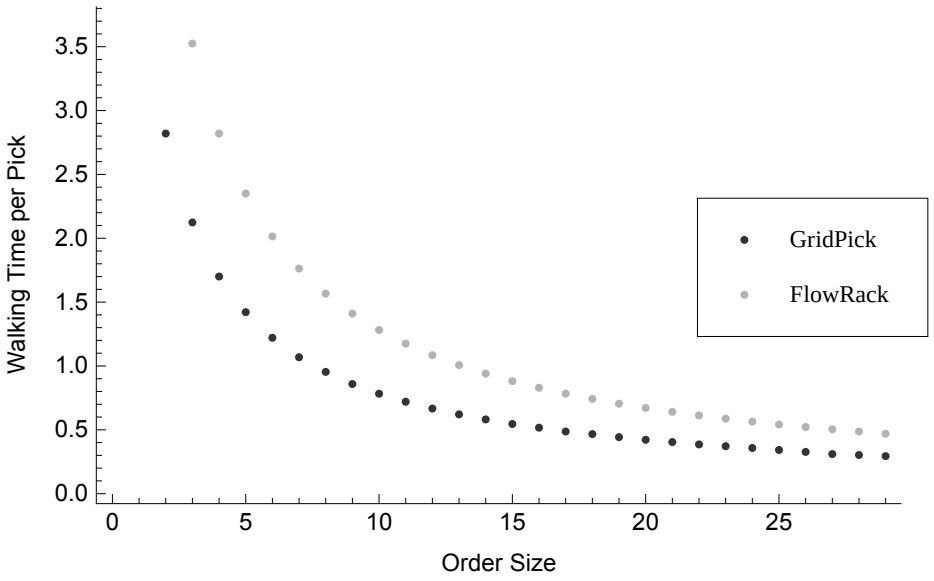

Figure 8: Walking time per pick for GridPick and an equivalent flow rack.

A Note on Deadlocks The decentralized control algorithms behind the GridPick system do *not* guarantee deadlock free operation. GridPick can deadlock when the number of items seeking the pick face exceeds the number of slots in the face. The precise conditions under which GridPick is guaranteed to be deadlock-free are under investigation.

5 Conclusions

GridPick is a new application of puzzle-based storage and material movement for order picking systems. It is important to note that the hardware requirements for GridPick are standard fare in the material handling industry, and so everything described in this paper could be implemented with current technologies. The GridPick system offers the potential to reduce order picking labor when compared with flow rack, by reducing the travel distances between items in an order. The cost, of course, lies in the automation. Whether or not GridPick would be cost competitive requires more research, and certainly would be application specific.

References

[1] A. Alfieri, M. Cantamessa, A. Monchiero, and F. Montagna. Heuristics for puzzle-based storage systems driven by a limited set of automated guided vehicles. *Journal of Intelligent Manufacturing*, pages 1–11, 2010. published online.

[2] Y. A. Bozer and J. A. White. Travel-time models for automated storage/retrieval systems. *IIE Transactions*, 16:329–338, 1984.

[3] R. B. M. de Koster, T. Le-duc, and Y. Yu. Optimal storage rack design for a 3-dimensional compact AS/RS. *International Journal of Production Research*, 46(6):1495–1514, 2008b.

[4] R. Dekker, R. B. M. de Koster, K. J. Roodbergen, and H. van Kalleveen. Improving order picking response time at Ankor's warehouse. *Interfaces*, 34(4): 303–313, 2004.

[5] K. Furmans, F. Schönung, and K. R. Gue. Plug and Work Material Handling Systems. In *Progress in Material Handling Research: 2010*, pages 132–142, 2010.

[6] K. Furmans, C. Nobbe, and M. Schwab. Future of Material Handling - modular, flexible and efficient. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[7] K. R. Gue and K. Furmans. Decentralized Control in a Grid-Based Storage System. In T. Doolen and E. Van Aken, editors, *Proceedings of the 2011 Industrial Engineering Research Conference*, 2011.

[8] K. R. Gue and B. S. Kim. Puzzle-based storage systems. *Naval Research Logistics*, 54(5):556–567, 2007.

[9] Y. H. Hu, S. Y. Huang, C. Y. Chen, W. J. Hsu, A. C. Toh, C. K. Loh, and T. C. Song. Travel time analysis of a new automated storage and retrieval system. *Computers & Operations Research*, 32:1515–1544, 2005.

[10] H. F. Lee. Performance analysis for automated storage and retrieval systems. *IIE Transactions*, 29:15–28, 1997.

[11] C. G. Petersen. An evaluation of order picking routing policies. *International Journal of Operations & Production Management*, 17(11):1098–1111, 1997.

[12] I. Potrč, T. Lerher, J. Kramberger, and M. Šraml. Simulation model of multi-shuttle automated storage and retrieval systems. *Journal of Materials Processing Technology*, pages 157–168, 2004.

[13] H. D. Ratliff and A. S. Rosenthal. Orderpicking in a rectangular warehouse: A solvable case of the traveling salesman problem. *Operations Research*, 31(3): 507–521, 1983.

[14] K. J. Roodbergen and R. de Koster. Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, 39(9):1865–1883, 2001.

[15] Z. Sari, C. Saygin, and N. Ghouali. Travel-time models for flow-rack automated storage and retrieval systems. *International Journal of Advanced Manufacturing Technology*, 25:979–987, 2005.

[16] G. D. Taylor and Kevin R. Gue. Retrieval Time Performance in Puzzle-Based Storage Systems. In A. Johnson and J. Miller, editors, *Proceedings of the 2010 Industrial Engineering Research Conference*, 2010.

[17] J. A. Tompkins, J. A. White, Y. A. Bozer, E. H. Frazelle, J. M. A. Tanchoco, and J. Trevino. *Facilities Planning*. New York: John Wiley & Sons, Inc., 2nd edition, 1996.

[18] J. P. van den Berg and A.J.R.N. Gademann. Simulation study of an automated storage/retrieval system. *International Journal of Production Research*, 38:1339–1356, 2000.

[19] J. Y. Wang and Y. Tih. Using neural networks to select a control strategy for automated storage and retrieval systems (AS/RS). *International Journal of Computer Integrated Manufacturing*, 10(6):487–495, 1997.

[20] Y. Yu and R. B. M. de Koster. Optimal zone boundaries for two-class-based compact three-dimensional automated storage and retrieval systems. *IIE Transactions*, 41:194–208, 2009.