A High-Density System for Carton Sequencing

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

Kai Furmans, Institute for Conveying Technology and Logistics, Karlsruhe Institute of Technology, Germany.

Abstract

We present a new approach for building a required sequence of cartons to feed a palletizing robot, using a grid-based material handling system. The system features a decentralized control algorithm in which storage cells communicate to their neighbors to accomplish the sequencing. The proposed system has high storage density, and can be easily reconfigured or rescaled.

1 Introduction

Robotic pallet building is a relatively new means of reducing inventory and labor costs in the distribution industry. In a common application, robots pick cartons from a staging area and place them methodically on a pallet, in a sort of 3-dimensional packing problem. The objectives are (1) to achieve a high packing density, (2) to adhere to product constraints (e.g., lighter items on top of heavier items), (3) to minimize inventory costs by shipping less-than-pallet quantities, and (4) to reduce labor costs in stores by placing items nearby on the pallet that are nearby in the store.

To achieve these objectives, an algorithm must determine a precise sequence of cartons for placement on the pallet. Unfortunately, determining the sequence is only part of the problem, because cranes in an automated storage and retrieval system (AS/RS), which typically feeds robotic pallet-building stations, operate in parallel and so cannot guarantee the required sequence. For example, in a beverage distribution center, cranes in the AS/RS retrieve pallets of beverage cartons, which are de-palletized and fed via conveyor to multiple robotic pallet-building stations. Cartons enter the conveyor system batched by sku, and must be sequenced prior to being picked by the robot (Figure 1).

We introduce a new way to sequence cartons based on grid-based material handling (Gue & Furmans 2011). The system has a decentralized control algorithm, in which unit-sized conveyors execute a message passing protocol with their local neighborhoods. Each conveyor has its own sensors to detect its condition (empty, occupied, etc.), and its own controller to execute the same logic according to its condition and the conditions of its neighbors. The system is easily configurable and highly scalable.

This paper is organized as follows: in section 2, we review related literature, including sequencing algorithms that consider different factors to determine a sequence for the robotic palletizer. In section 3, we define the problem and describe a few other applications to achieve carton sequencing. In section 4, we explain the algorithm in detail. In section 5, we present performance results for different aspect ratios and numbers of empty columns per row. We offer conclusions in Section 6.

2 Literature

Many previous studies have considered how to determine a required sequence for the pallet-loading problem. The "manufacturer's pallet loading problem" is to find the sequence of boxes of identical size for each layer of the pallet (Hodgson 1982). The

"distributor's pallet loading problem" considers packing of non-homogenous items (Bischoff & Ratcliff 1995). The "multi-pallet loading problem" addresses packing onto multiple pallets (Terno et al. 2000). Rethmann & Wanke (1997) do not address the sequencing problem, but consider building multiple pallets in a workstation with a cyclic storage conveyor transferring items to input buffers. Finding the minimum number of pallets to load a certain number of boxes of various types is the objective for these kinds of problems. In this paper, we consider the manufacturer's pallet-loading problem (Hodgson 1982) that has a loading pattern for identical items. Ratcliff and Bischoff (1998) have studied optimization of the container-loading problem considering weight constraints. There are also several studies (Schuster et al. 2010; Terno et al. 2000; Kocjan & Holmstrom 2008) focusing on the stability of such pallets. Several authors have addressed three-dimensional loading patterns that solve the problem of maximizing volume utilization of a pallet (George & Robinson 1980; Gehring & Bortfeld 1997; Schuster et al. 2010). Scheithauer (1992), Sculli & Hui (1988), George & Robinson (1980), and Wu (2010) developed both exact and heuristic methods for the container-loading problem. Other authors have addressed control configurations for robotic palletizers (Dzitac & Mazid 2008; Warnecke & Baumeister 1990; Wurll & Schnoor 2003).

In automotive assembly plants, mixed-model assembly lines also need to sequence subassemblies or parts for final assembly. Parts should be delivered in sequence starting from suppliers for a mixed model production line. Several studies have addressed mixed model assembly line sequencing for conveyor stoppage and minimization of work overload (Xiaobo & Ohno 1997; Bolat 1997). A survey article is Boysen et al. (2007). Car sequencing is also considered in a number of studies (Fliedner & Boysen 2008; Gravel et al. 2005). In an automotive plant, parts can go out of sequence due to missing parts, delays, rework, parallel work stations, and so on, which requires resequencing of vehicle bodies and work pieces (Inman 2003; Gujjula & Günther 2009; Boysen et al. 2011). Inman (2003) describes the problem of sizing an ASRS for the assembly sequence before delivering to the final assembly line. His study is similar to ours in that it seeks a high-density way of accomplishing the sequence.

3 Problem Statement

We consider the problem of receiving a random sequence of cartons and delivering an exactly sequenced stream. Arriving items are numbered according to the required sequence, and the system stores them until their predecessors arrive. Items are allowed to leave immediately after their predecessors leave, although different departure rules exist (more on this below). Because there is a positive probability that the first item in the sequence arrives last in the incoming stream, it is necessary to have sufficient storage space for all items, in the worst case.

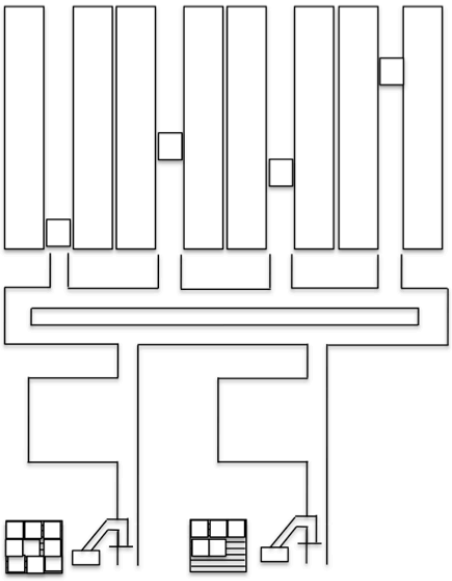

Figure 1: Sequencing required by automated material handling system. Items are retrieved from the AS/RS and fed to the robotic pallet building workstations. A sequencing system is needed to build an exact sequence for the robots.

In practice, companies address the sequencing problem by integrating it with the AS/RS. Requesting and releasing items from the AS/RS according to the required sequence is one way to approach this problem. A warehouse management system constructs a desired sequence and releases requests to the AS/RS accordingly. In practice, cartons often arrive out of sequence, requiring the palletizer to store temporarily items arriving out of sequence.

Another way to achieve the required sequence is to use a sortation conveyor. Items enter and loop around the conveyor until they are required in the stream. There are several disadvantages of this system, including increased cycle time of requested items, inflexibility with respect to throughput capacity, and poor utilization of floor space.

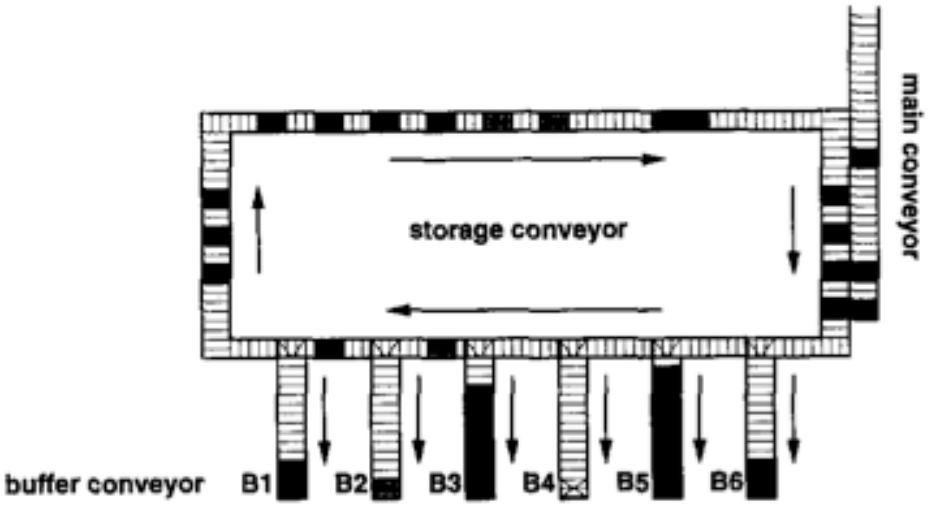

Figure 2: A sortation conveyor used to sequence cartons (Rethmann & Wanke 1997).

4 Solution Approach

Our approach is based on the GridFlow system (Gue & Furmans 2011), in which unit sized conveyors, or *modules*, form a grid to store items with high density, and items travel to the edge of the system when requested. Modules are able to convey in the four cardinal directions and to communicate with their north, south, east and west neighbors. Furthermore, modules have basic information about the topology of the grid, such as its size, and their own positions (row and column numbers). Each module has the same control logic and acts according to its parameters, the carton (or not) on top, and information from its neighborhood.

In the GridFlow system, requested and replenishing items move south only (not east or west); interfering items move east and west to get out of the way. Modules are synchronized to execute in a series of discrete events. At first, each module detects its own condition as empty, occupied, or seeking (meaning the module contains an item that must move south). Modules have sensors to detect the state of the carton (requested, occupied and replenished) on top of the module. The algorithm continues with north-south negotiation and east-west negotiation phases, which conclude with each module knowing whether or not it will convey, and in which direction. Details of these negotiations are below.

The series of synchronous movements result in what we call "puzzle based movement." The sequencing algorithm we describe below has logic similar to the GridFlow system for the interfering items, but it uses a restricted movement scheme to prevent items from going out of sequence after they reach their target positions.

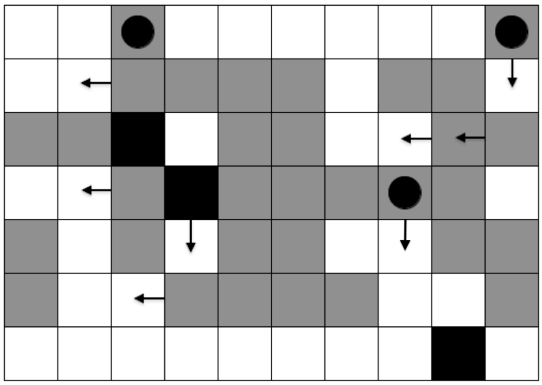

Figure 3: Representation of the GridFlow system. Stored items (gray) move left and right to allow requested items (black) to depart and replenishing items (black circle) to reach their target rows.

An example iteration of GridFlow is in figure 3, where black squares represent requested items that need to move down to an exit row on the bottom. Gray items with black circles are replenishing items that have left the grid already. Gray squares represent non-requested items, which move left and right to allow the passage of requested and replenishing items. Each row has a number of empty cells to allow east and west moves of the non-requested items.

4.1 System Description

GridSequence is a modified version of the GridFlow system in which there is one additional row at the top and one additional column to the left in the grid. The additional row is for travel to a target column. The additional column allows temporary repositioning of interfering items. The system can be modified to sequence any number of items with different aspect ratios and sizes of the grid.

The GridSequence system is based on having target positions for the cartons according to the required sequence. Cartons enter the system from the upper right corner and travel left in the top row until they reach their target columns. After reaching their appropriate columns, cartons move down to reach their target rows. Once in the correct position in the grid, items move left or right temporarily only to allow passage of other items seeking their own positions in the grid.

Items leave the system from the bottom right corner. So, the first item in the sequence should be placed in the most bottom right position. Other items are assigned to corresponding target positions. For instance, to sequence 100 items in an 11x11 grid, cartons 0-9 should be placed in the right most column, cartons 10-19 in the column to the left of the right most column, and so on.

Interfering items must move left and right to allow the downward movement of items traveling to their target rows. This means that they are leaving their target positions temporarily and that they should return to their target positions to keep the sequence. Items that have already moved to the side are not allowed to move farther away from the target position. So, there is a limited "puzzle based movement" for interfering items and it is restricted to one cell to the right or left.

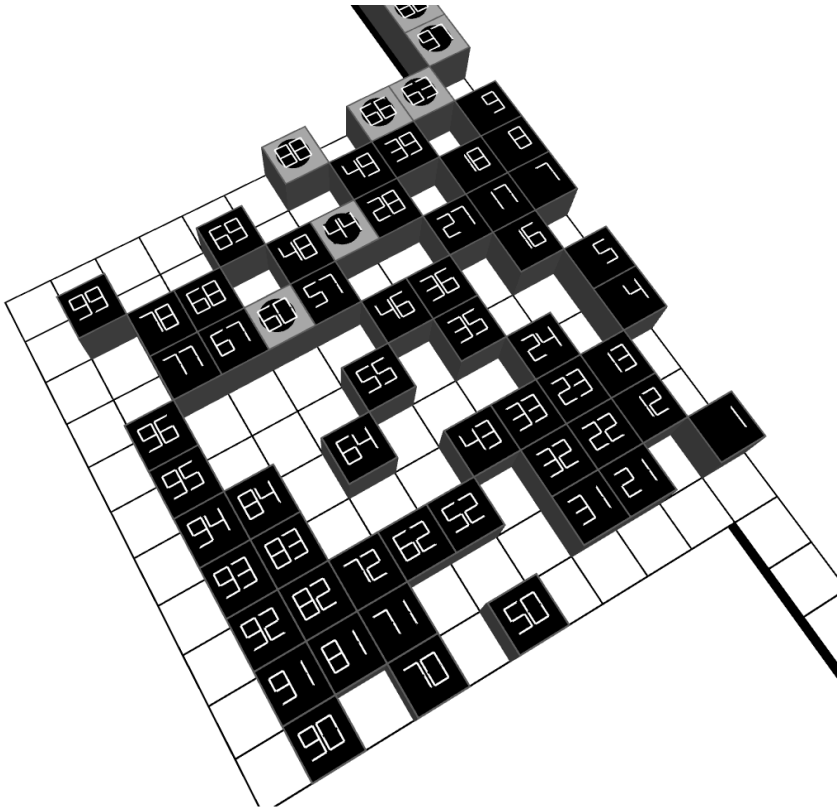

Figure 4: Snapshot of the GridSequence System. Items 44 and 60 are in the grid and traveling down to their target rows. Item 60 will travel to the bottom row. Items 67 and 68 are moved to the left to allow item 60 to pass by. Item 48 is moved to the left to allow passage of item 44. Item 88 is traveling to its target column on the top row.

4.2 Details of the Algorithm

Modules are synchronized to execute at the same time. Therefore, modules are in the same phase of the algorithm at a given time. The main steps of the algorithm are described below.

4.2.1 Setup Condition

At the start of the iteration, each module reads the RFID tag of the carton on top and updates its condition (empty, occupied, or seeking). This information can also be passed to the module with the carton itself, alleviating the need to read RFID tags.

After determining its initial condition, each module checks to see if it meets a "departure condition," meaning it checks to see if the sequence is complete and items are ready to leave. Below we describe two departure policies, each of which has a separate departure condition. The first policy requires that all cartons in the sequence be in the grid and sequenced before any cartons leave. In this case, the module asks its north and west neighbors if they have their required items; if all items meet this condition, then all modules also meet the departure condition. The second departure policy requires that (1) all modules in a column have their required items and (2) all columns to the right have met the departure condition. In both cases, if a module meets the departure condition, it no longer executes the negotiating events to follow.

4.2.2 North-South Negotiation

In this phase, the module carrying an item seeking its target position initiates a message to the south to see if it is empty or if it too contains an item moving down. If so, it can go to a condition to convey down and send its item to the south. If the south neighbor is occupied with an item not moving down, the module stays idle.

The north-south negotiation also places each module in a beginning state for the east-west negotiation. For example, when a module with an occupied (but not moving) item receives a message from its north neighbor that it wants to send its item southward, it will go to condition "request left or right."

4.2.3 East-West Negotiation

The east-west negotiation attempts to clear the way for items moving downward to their target positions. Modules in position "request left or right" send messages to their left and right neighbors, asking if they can receive an item passed in the appropriate direction. This message is propagated left and right until an empty module is found. A module in the empty condition responds that it can receive the item, and all modules between the module in "request left or right" and the empty module commit to convey in a "block movement." The module with the "request left or right" condition receives the response from one side and sends a commit message as a decision to move that way. At the end of the east-west negotiation, all modules are committed to convey left or right or down, or to remain idle.

4.2.4 Convey phase

This phase executes the movement decisions according to the final conditions of the modules. Negotiation rules are established such that modules cannot end up with a negotiation condition such as request left, request right, etc.

4.3 Departure Events

We consider two different departure policies. In the first, items may leave only after the entire sequence is ready. In the second, items start to leave when their column and all predecessors' columns are ready.

4.3.1 Depart When Entire Sequence Is Ready

This event checks to see if the sequence is ready for the entire grid. The first item leaving the system is at the bottom right position. Therefore, whenever the first item in the sequence arrives to its target position, the module carrying the item triggers a sequence check mechanism and sends messages both north and west. In this way, messages propagate all over the grid, and when the left most modules receive the message they respond back. If a module has not received its item, it does not respond to the incoming message. In this way, the message becomes interrupted and departure cannot start. If the message returns to the module in the bottom right position, the module knows that the entire sequence is ready and items can start to leave the grid.

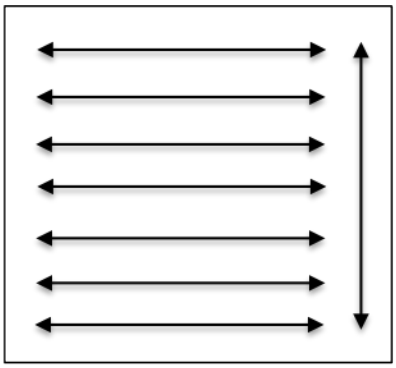

Figure 6: Message route for the sequence check

Figure-6 shows the message route starting from the bottom right module for the sequence check of the entire grid. If a module receives the message from the south neighbor and if it has the box with the correct target position, it forwards the message to the north and west. If a module receives the message from its east neighbor and if it has the box on top with the correct target position, it forwards the message to its west neighbor. When the message has arrived to the last position on the left, it responds back, and the message goes back following the same route.

If there is not yet an item in the target position, the module does not forward or response to the messages. Thus, departure cannot initiate.

4.3.2 Depart When Column Is Ready

An obvious enhancement is to allow items to depart as soon as their predecessors are in place. Again, the bottom right module initiates the messages if its row and column matches with the target row and column of the item on top. This time, it sends the message only to its north neighbor. This message propagates and returns from the top row, if all items are in their target positions in this column. If the bottom module receives the response message, it forwards the message to its west neighbor, and the west neighbor triggers a sequence check in its own column with a message to the north neighbor.

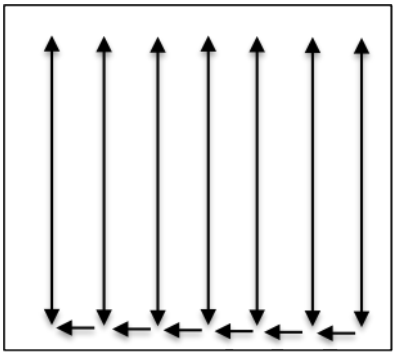

Figure 7: Message route with the cell based rules for the sequence check with a "leave when column ready" departure policy

Figure-7 shows the message route starting from the bottom right module. The sequence check always starts from the bottom right position, because it can leave without waiting for others in the ideal case. The sequence check is done separately for each column. When a column's sequence is ready, the bottom module forwards the message to the west neighbor for additional sequence checks. Again, if a module's row and column number does not match with the target row and column of the item on top, message passing stops and departure cannot start for the specified column.

4.3.3 Departure Negotiation

After the message passing for the sequence check, if the bottom right module understands that sequence is ready for the entire grid or the column, it will start a negotiation similar to north-south negotiation. It will send out a message to the south neighbor, and if it is empty it will send the box on top to the outgoing line. Also, it will forward the message to the north neighbor. In this way, items in the column will be able to do a block movement to the south. When the items in the first column have left entirely, the bottom right module will detect that it is empty and it will send a message to its west neighbor. This will initiate a block move from left to right, which is similar to the west-east negotiation. The only difference is there are two waves of messages in this case.

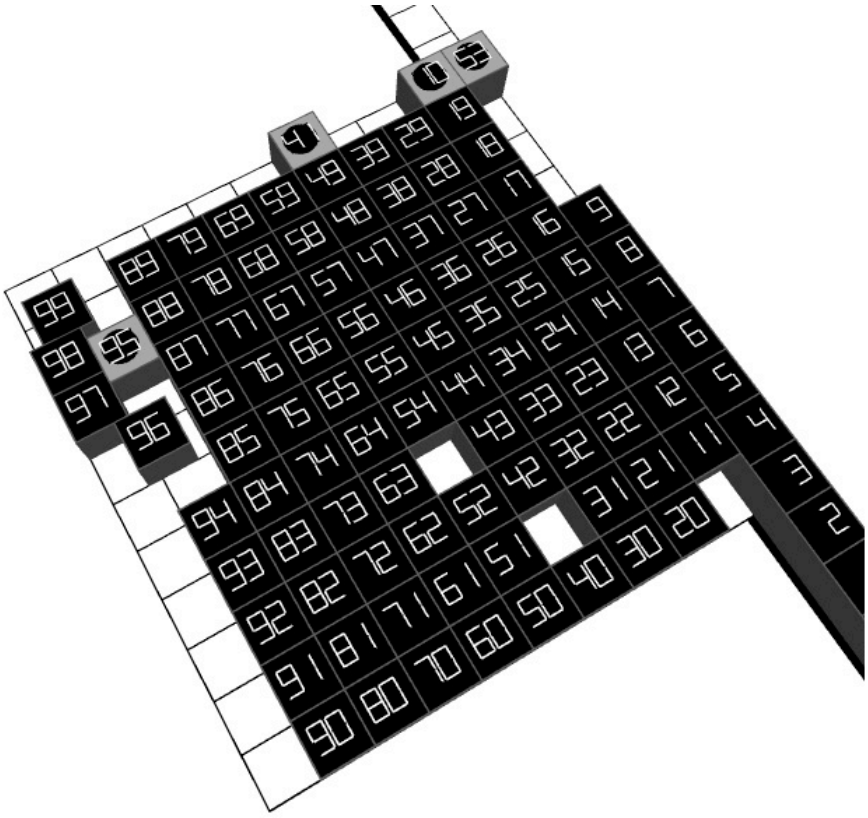

Figure 8: Snapshot of the model during departure during "leave when column ready." The first column is leaving from the right. A few items are still traveling to their target positions.

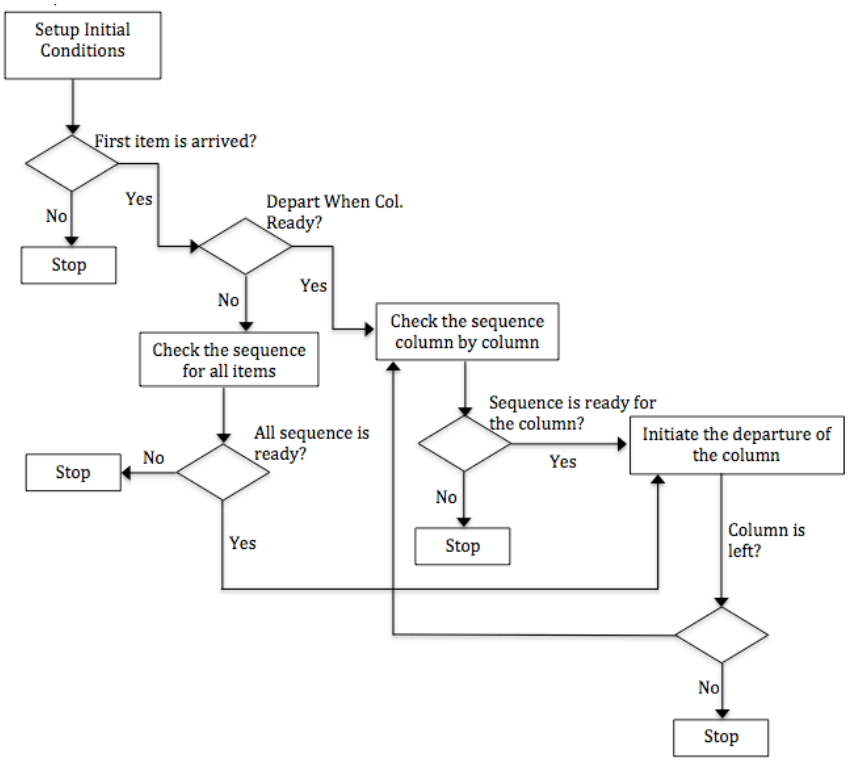

Figure 9: Flow chart showing the trigger steps of the departure mechanism. In each step the module initiates the message passing, and neighbors respond and/or forward these messages according to local conditions. Stop events mean that units are done with message passing for this iteration.

5 Analysis

In this section, we investigate performance of the GridSequence system with respect to changes in departure policies, aspect ratio, and empty columns. These factors are interdependent. For instance, when empty columns are added to the system, the grid becomes slightly wider. When the number of items per column decreases for wide systems, it can affect the performance of the departure policies.

5.1 Aspect Ratio Analysis with Two Departure Policies

More columns in a GridSequence system mean shorter columns, which would take less time to fill and be ready for departure. However, more columns also means more travel orthogonal to the departure path. Are more columns better, or not?

Table 1: Experiments with different aspect ratios. The required number of modules is smallest in the middle and highest for very small and large aspect ratios.

Rows	Columns	Aspect Ratio	Modules
1+1	96+1	0.02	194
2+1	48+1	0.06	147
3+1	32+1	0.12	132
4+1	24+1	0.20	125
5+1	20+1	0.29	126
6+1	16+1	0.41	119
7+1	14+1	0.53	120
8+1	12+1	0.69	117
9+1	11+1	0.83	120
10+1	10+1	1.00	121
11+1	9+1	1.20	120
12+1	8+1	1.44	117
14+1	7+1	1.87	120
16+1	6+1	2.43	119
20+1	5+1	3.50	126
24+1	4+1	5.00	125
32+1	3+1	8.25	132
48+1	2+1	16.33	147
96+1	1+1	48.50	194

Table 1 shows several configurations with different aspect ratios. All configurations sequence 96 items and have one additional top row and one additional column for the side moves. The aspect ratio changes with respect to the number of rows and number of columns.

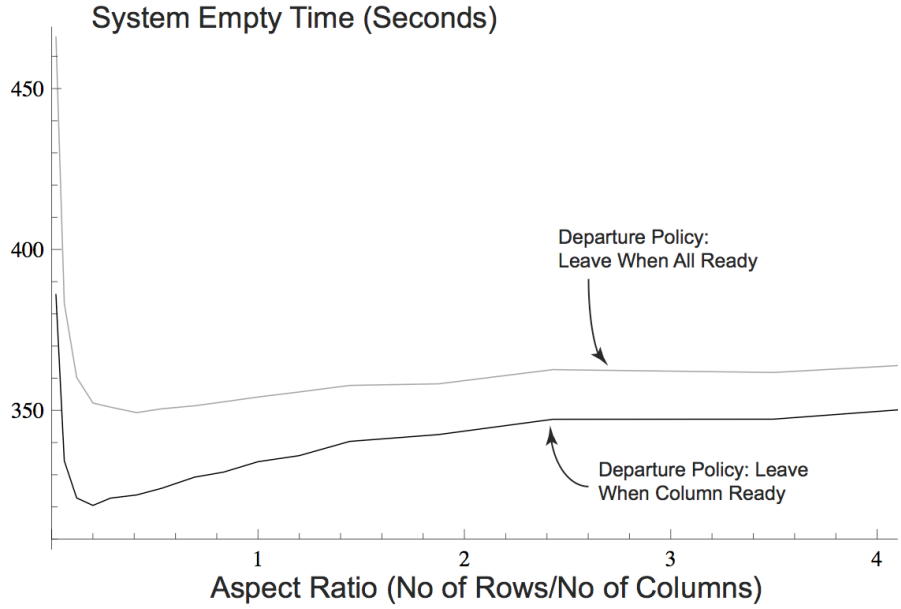

Figure 10: System empty time with respect to aspect ratio for two departure policies.

Figure 10 shows the amount of time it takes for the system to empty with different aspect ratios. For purposes of clarity, the figure omits ratios greater than 4. For configurations with a very low ratio of rows to columns, the system has the worst system

empty time because items travel excessively in the top row, and then must move back the same way during departure. System empty times drop dramatically when the system has more rows and fewer columns, reaching a minimum at ratio 0.41 for the "leave when all ready" policy and 0.2 for the "leave when column ready" policy. As the ratio increases, columns get fewer and longer, resulting in more congestion on the top row as items attempt to enter fewer columns. Moreover, longer columns (more rows) means more interfering items must move out of the way, which also increases the required time to reach target rows.

It is interesting to note that the most productive configurations also have the fewest required modules (see Table 1). Systems with fewer modules, of course, would be less expensive to implement in practice.

5.2 Number of Empty Columns for Puzzle Based Movement

So far, we have assumed that there is only one empty column to facilitate the puzzle-based movement of interfering items. Will performance improve if the number of empty columns is increased?

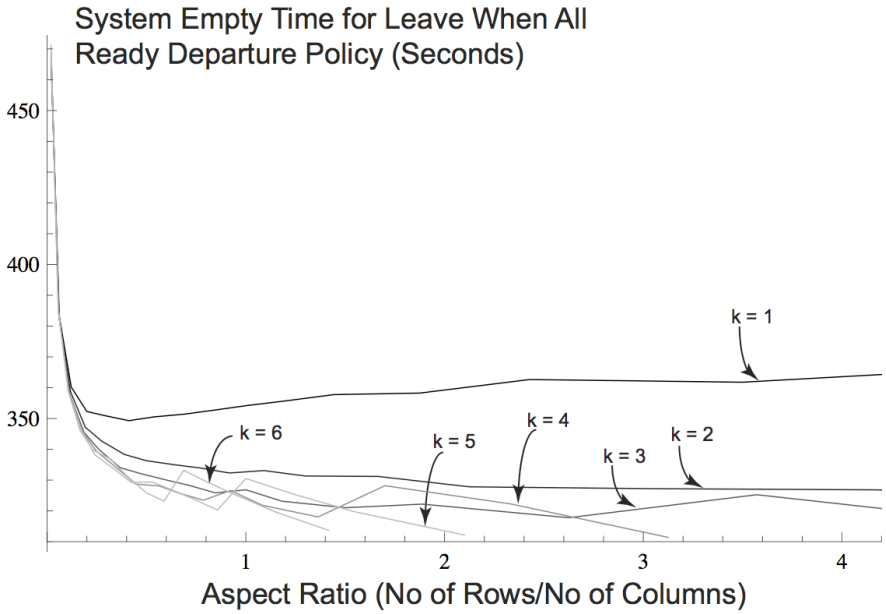

Figure 11: System empty time with respect to aspect ratio for different empty column configurations k = 1 through 6 and the "leave when all ready" departure policy. Irregularities in the curves for high aspect ratios are due to non-uniform distribution of empty columns for those configurations.

In Figure 11, k is the number of empty columns in the system, including the far left column. We place the empty columns in the grid uniformly. With more empty columns, horizontal moves of interfering items become easier, and the system gets wider. When we increase k from 1 to 2, the system empty time decreases significantly. Interestingly, additional empty columns seem not to improve system performance very much. Curves for k=4, 5, and 6 end when the number of empty columns equals the number of occupied columns.

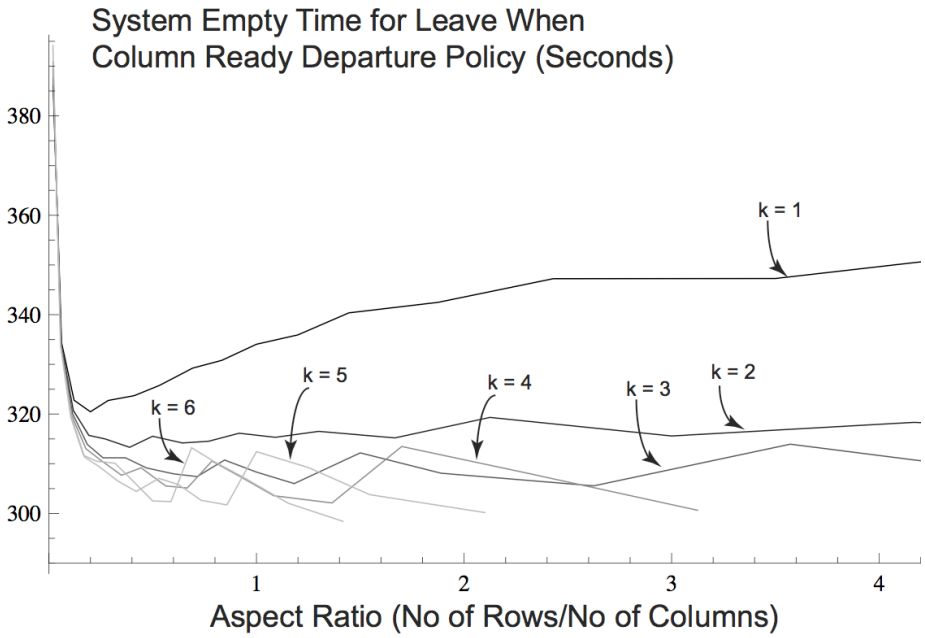

Figure 12: System empty time with respect to aspect ratio for different empty column configurations k = 1 through 6 and the "leave when column ready" departure policy.

Figure 12 shows similar results for the "leave when column ready" policy. Figures 11 and 12 suggest that a single additional empty column, located near the center of the grid is effective in reducing system empty time, but that additional empty columns add little value. The figures also suggest that system empty time is relatively insensitive to the aspect ratio (rows to columns), as long as the ratio is greater than about 1/4.

6 Conclusion

Carton sequencing is an important part of robotic pallet building systems. We have introduced a high-density solution to carton sequencing, which relies on puzzle-based movement with unit-sized conveyors. The system feature decentralized control of individual modules, and therefore is very flexible and scalable.

In general, grids with fewer columns perform better, because the top row tends to be a bottleneck. The more quickly items can find their columns; the faster they clear out of that row. Our results suggest that the aspect ratio (rows divided by columns) should be at least 10. It also seems beneficial to provide one additional empty column near the center of the grid to facilitate east-west movement of interfering items.

We should also add that carton sequencing could be used in other applications. For example, pallets in a crossdock might be sequenced for last-in, first-out delivery.

7 References

Bischoff, E. E. & Ratcliff, M. S. W. (1995): Issues in the development of approaches to container loading, OMEGA 23(4): 377-390.

Bischoff, E. E. (2006): Three-dimensional packing of items with limited load bearing strength, European Journal of Operational Research 168(3): 952–966.

Bolat, A. (1997): Stochastic procedures for scheduling minimum job sets on mixed model assembly lines, Journal of the Operational Research Society 48: 490-501.

Boysen, N., Fliedner, M. & Scholl, A. (2007): Sequencing mixed-model assembly lines: Survey, classification and model critique, Jena Research Papers in Business and Economics (JBE) 2/2007.

Boysen, N., Uli, G. & Rothlauf, F. (2011): The car resequencing problem with pull-off tables, Business Research Official Open Access Journal of VHB 4(2): 1-17.

Dzitac, D. & Mazid, A. M. (2008): An Efficient Control Configuration Development for a High-speed Robotic Palletizing System, IEEE International Conference on Robotics, Automation and Mechatronics (RAM 2008), Chengdu, China.

Fliedner, M. & Boysen, N. (2008): Solving the car-sequencing problem via branch and bound, European Journal of Operational Research 191: 1023-1042.

Furmans, K., Schönung, F. & Gue, K. R. (2010): Plug and work material handling systems, In Progress in Material Handling Research 2010: 132–142.

Gehring, H. & Bortfeld, A. (1997): A genetic algorithm for solving the container loading problem, International Transactions of Operational Research 4: 401-418.

George, J. A. & Robinson, D. F. (1980): A heuristic for packing boxes into a container, Computers and Operations Research 7: 147-156.

Gravel, M., Gagné, C. & Price, W. L. (2005): Review and comparison of the three methods for the solution of the car-sequencing problem, Journal of the Operational Research Society 56: 1287–1295.

Gue, K. R. & Furmans, K. (2011): Decentralized Control in a Grid-Based Storage System. In T. Doolen and E. Van Aken, editors, Proceedings of the 2011 Industrial Engineering Research Conference.

Gujjula, R. & Günther, H. O. (2009): Resequencing mixed-model assembly lines under just-in-sequence constraints, Proceedings of the 39th International Conference on Computers & Industrial Engineering, Troyes, France: 680-685.

Hodgson, T. J. (1982): A combined approach to the pallet loading problem, IIE Transactions 14 (3): 175-182.

Inman, R. R. (2003): ASRS sizing for recreating automotive assembly sequences, International Journal of Production Research 41(5): 847-863.

Kocjan, W. & Holmström, K. (2008): Generating stable loading patterns for pallet loading problems, The Fifth International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems CPAIOR08.

Ratcliff, M. S. W. and Bischoff, E. E. (1998): Allowing for weight considerations in container loading OR Spectrum, 20, 65-71.

Rethmann, J. & Wanke, E. (1997): Storage controlled pile-up systems, theoretical foundations, European Journal of Operational Research, 103(3): 515-530.

Scheithauer, G. (1992): Algorithms for the container loading problem, in: Proceedings in Operations Research, Springer, Berlin: 445-452.

Schuster, M., Bormann, R., Steidi, D., Reynolds-Haertle, S. & Stilma, M. (2010): Stable Stacking for the Distributor Pallet Packing Problem, In IEEE/RSJ International Conference on Intelligent Robots and Systems.

Sculli, D. & Hui, C. F. (1988): Three-dimensional stacking of containers, OMEGA 16(6): 585-594.

Terno, J., Scheithauer, G., Sommerwei, U. & Riehme, J. (2000): An efficient approach for the multi-pallet loading problem," European Journal of Operational Research 123: 372–381.

Warnecke, H. J. & Baumeister, K. (1990): Order-picking industrial robot, Robotica 8: 37-45, Cambridge University Press, UK.

Wu, Y., Li, W., Goh, M. & de Souza, R. (2010): Three-dimensional bin packing problem with variable bin height, European Journal of Operational Research 202: 347-355.

Wurll, C. & Schnoor, B. (2003): Robot picking system (RPS)-automated order picking with industrial robots, Proceedings of the IASTED International Conference on Robotics and Applications 64(9), ACTA Press, Anaheim, CA, USA.

Xiaobo, Z. & Ohno, K. (1996): Algorithms for sequencing mixed models on an assembly line in a JIT production system, Computer and Industrial Engineering 32(1): 47-56.