A Hybrid Scheduling and Control System Architecture for Warehouse Management

Byung-In Kim, Sunderesh S. Heragu, Robert J. Graves, and Art St. Onge

***Abstract*-- In recent years, the hybrid control framework has received much attention from the research community. Several variations of this control framework are available. In this paper, an actual industrial warehouse order picking problem where goods are stored at multiple locations and the pick location of goods can be selected dynamically in near real time, is considered. A hybrid intelligent agent based scheduling and control system architecture is presented for the order picking problem. The need for a higher level optimizer and communication between higher and lower level controllers is demonstrated. The presented architecture includes a higher level optimizer, a middle level guide agent, and lower level negotiation agents. A mathematical model and a genetic algorithm for the resource assignment problem are presented. Simulation results demonstrating efficiency of the new approach are also presented.**

***Index Terms*-- Hybrid framework, intelligent agent, warehouse management**

I. Introduction

A control framework is the backbone of any control system architecture and frameworks can be classified as *hierarchical*, *heterarchical*, and *hybrid* [1, 2]. The *hierarchical* framework assumes there is a hierarchy and a master/slave relationship between higher and lower levels of control. The higher level controller makes scheduling and other decisions based on a simplified model of lower level status. If there is no disturbance at the time of execution, this framework can provide a globally optimal or near-optimal set of decisions because the higher level controller has a global view of the system. As Szelke and Kerr [3] point out, however, external disturbance factors such as fast changing customer demands, urgent order insertion and order cancellation, and internal manufacturing disturbance factors such as machine breakdowns, missing tools, late deliveries, and rework occur frequently in today's manufacturing or service environment. These unforeseen disturbances invalidate the global plan and schedule at an operational level [4].

To provide robustness to the system, *heterarchical* frameworks have been proposed. However, they focus almost entirely on interactions between unit controllers (agents) to allow system flexibility, while ignoring those between higher and lower level controllers. Unit controllers negotiate with each other and make their own decisions. While this offers increased robustness against disturbance and reconfigurability against changes, it loses predictability and global perspective. It could also lead to chaos.

In recent years, developers of *hybrid* frameworks (for example, [2], [4]-[8]) have attempted to overcome some of the problems mentioned above by combining features of both hierarchical and heterarchical frameworks. They have developed hierarchical structures to enhance global performance while encouraging coordination between agents. Most of them focus on the generic conceptual framework.

In this paper, we propose a hybrid scheduling and control system architecture that achieves robustness and system wide objectives, and then apply it to an industrial warehouse order picking problem. We demonstrate the need for a higher level optimizer and communication between higher level and lower level controllers. A mathematical and a genetic algorithm for the resource assignment problem are also presented. Simulation experiments show the efficiency of the hybrid model compared with hierarchical and heterarchical models.

This paper is organized as follows. After we briefly review existing literature on reactive scheduling and real time control in section II, we describe an industrial warehouse order picking problem in section III. The need for a higher level optimizer and communication between higher and lower level controllers is presented in section IV. Our proposed scheduling and control system architecture, a genetic algorithm for the resource assignment problem, and experimental results follow in section V. Conclusions are presented in section VI.

II. Literature Review

Graves [9] surveys state of the art in the theory and practice of production scheduling until 1980. Parunuk [10] describes challenges to job shop scheduling and surveys existing literature. He categorizes papers as structural or operational depending upon what solutions they offer for the challenges. The challenges include desirability (some costs are more important than others), stochasticity (the real world changes unexpectedly), tractability (complex algorithms take too long to compute), chaos (small initial uncertainties lead to widely divergent outcomes), and decidability (no algorithm exists to predict the system's behavior).

Szelke and Kerr [3] survey artificial intelligence based

Manuscript received June 14, 2001. This research is supported by a grant from National Science Foundation, numbered DMI 9900039 and by St. Onge Company.

B. Kim, S. S. Heragu, and R. J. Graves are with the Department of Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, Troy, NY 12180 USA (e-mail: kimb4@rpi.edu, herags@rpi.edu, graver@rpi.edu).

A. St. Onge is with St. Onge Company, York, PA 17402 USA (e-mail: artstonge@stonge.com)

reactive scheduling approaches. They survey the definitions and characteristics of reactive scheduling, summarize the main requirements of reactive scheduling, and review problem-solving approaches. Reactive scheduling can be defined as continuous adjustment of given schedules to keep the system operating against disturbances. Shukla and Chen [1] classify the literature of real time FMS control according to the techniques: simulation, knowledge based, neural network, fuzzy logic, petri nets, and hybrid approaches.

Sadeh and Fox [11] and Sycara et al. [12] present a pure heterarchical distributed scheduling framework. In their framework, each agent makes a schedule for a group of orders. Agents share their future resource demands with each other so as to enhance global convergence. Using the shared data, each agent finds the most critical resource and its corresponding critical operation, and allocates the resource to the operation. Agents share their updated future demand information again. If there are capacity and time conflicts between assignments, agents backtrack to a previous feasible status. The above process is repeated until a global feasible solution can be achieved. The focus of this framework is on scheduling speed and not on system performance.

Szelke and Monostori [13] point out that since advance (predictive) scheduling uses a static model of the shop status, it cannot handle unplanned disturbances. They assert that any effective scheduling system must be able to predict and quickly react to changes. They define reactive scheduling as a process of incrementally revising the execution of an outdated schedule and proactive scheduling as a process of adjusting a current schedule to prevent expected system failures.

Smith [14] presents the OPIS (Opportunistic Intelligent Scheduling system) that is a blackboard based constraint directed reactive scheduling system. It has three steps for incrementally revising an existing schedule: recognizing the conflicts or opportunities that are introduced into the schedule as a result of changes, selecting the most important conflict or opportunity, and selecting a scheduling action and executing it to resolve conflicts or to exploit opportunities.

Bongaerts [15] proposes a hybrid framework for shop floor scheduling and control. To achieve high and predictable performance while offering robustness in the event of a disturbance, a reactive scheduling agent (holon), an on-line manufacturing control (OMC) agent and their interaction mechanisms are proposed. The reactive scheduling agent (highest level agent) makes an off-line schedule with feedback from the OMC agent, periodically or when specific events occur. The OMC agent (middle level agent) takes the schedule as advice and not as a command and dispatches it to the shop floor. It can deviate from the original schedule during a short range of time when there are disturbances. Thus, while the OMC agent handles real time shop floor control, the scheduling agent makes a schedule for the future. The lowest level agents are resource and order agents. They receive commands from the OMC holon, but can override the command if they deem it necessary. Our proposed architecture is a revised version of this framework. Ours gives ultimate authority to the higher or middle level agents, whereas Bongaerts' architecture gives it to the lower level agents. For additional literature on warehouse planning and control, see [16] and [17].

III. Industrial Warehouse Order Picking Problem

In this section, we provide details of the actual industrial warehousing problem to which the proposed scheduling and control architecture is applied. In order to protect business interests of the company, fictitious company names, places and products are used in the description. Lareal Company is a manufacturer of cosmetics. It has a large warehouse in central Florida which receives cosmetic products from several plants in the US and overseas locations. The warehouse functions are primarily twofold. The first is to receive the cosmetic products - several thousand stock keeping units (SKUs) in various quantities from the plants and store them in the warehouse. The second is to receive customer orders and fill them from the SKUs stored in the warehouse. We focus on the latter in this paper.

Customer orders are received via mail and the Internet. Orders that must be filled the next day are processed through an Order Analysis and Planning System (OAPS) which creates the next day's order sequence and pick plan. The OAPS interacts with a Finite Scheduling System (FSS) whose primary function is to take the next day's order sequence as well as pick plan and to convert them into a detailed scheduling plan for the gantry robots which reside within the Gantry Picking Complex (GPC). The detailed scheduling plan generated by the FSS includes specific pick tasks for each material handling resource. The sequences of orders for the picking resources are determined by the OAPS and the feeding of orders to the order wave is done by the FSS.

The GPC has 16 pick zones and 16 replenishment zones (see figure 1). The pick zones are the areas where actual picking takes place for customer orders and the replenishment zones are the areas where the layout of product inventory for the next picking cycle is being built. Four pick zones and their corresponding replenishment zones make a sub-zone as shown in figure 1. Currently, the replenishment and the picking cycle require twenty hours, and the pan swapping between the replenished totes and pick totes requires four hours. Each sub-zone has a replenishment robot and each pair of sub-zones on the same side of the conveyor (i.e. sub-zones A and B or sub-zones C and D in figure 1) has one pan swap crane.

The pick zones are arranged in a serial order and there is a common conveyor among them so each order tray passes sequentially from the first pick zone to the last (figure 1). Each pick zone has one picking gantry robot and 19 pick totes with each pick tote having 48 compartments (figure 2). A compartment stores only one SKU type at any given time and has a capacity of 30 units of the product. While a typical SKU is contained in only one pick zone, some popular SKUs may be spread over more than one pick zone and are distributed across multiple pick zones located within the same sub-zone.

A pick zone also contains eighty-five drop buffers where products picked by the gantry robot are placed. The products wait here until their designated order trays arrive (figure 2). Each drop buffer can contain only one product at a given time. The conveyor speed indexes at a constant interval as each order tray progresses through the system allowing the drop buffer to deposit its contents on the designated order tray. The gantry robot must not only pick the items for the current order in its zone, but also place them in drop buffers prior to the order tray entering its zone. If the item is deposited in a drop buffer after the tray designated to receive it has passed that buffer, a pick error occurs. Each order tray will contain one customer's order and can accommodate multiple line items in the order. Each order's line item is a single unit of a product.

Since our scheduling architecture is limited to the order filling activity of the FSS and the replenishment activity is handled by a separate material handling system, the replenishment activity is not described in this paper. The order filling task of the FSS is an activity that must be completed within 20 hours and begins with the transfer of an entire day's order stream from OAPS to FSS. Once the order stream is received, the FSS segments the order stream into batches. Each batch is released one by one to the invoice printing system at least two hours prior to its scheduled pick start time. This allows mailbags to be setup at appropriate sort points within the sortation system well in advance of the time an order is scheduled to arrive.

Once the invoice packet pertaining to an order is put into an empty order collection tray, the FSS takes control of the tray constantly monitoring its path and progress via scanning devices located at strategic locations. When an order tray passes a designated upstream point (called trigger point), the FSS issues picking instructions in advance to each gantry robot that must fill the current order tray.

In order to use the gantry robots efficiently, 80 trays are logically formed into a 'train'. The first of the 80 trays is called the 'locomotive' and the remaining 79 make up the train pulled by the locomotive. When the locomotive of a train reaches the trigger point for a pick zone's gantry, the 80 order trays in the train are scanned to see what items, if any, must be picked from the zone. A work queue for the order train is then made. To minimize the gantry movement, the work queue is sorted based on the compartments' positions of the SKU storage units. Thus in each pick zone, the entire work queue for the order train is served in one pass of the corresponding gantry robot. The gantry robot then picks items in its zone and places them at the nearest drop buffer so that items are correctly dispensed into order trays when they pass the drop buffer location.

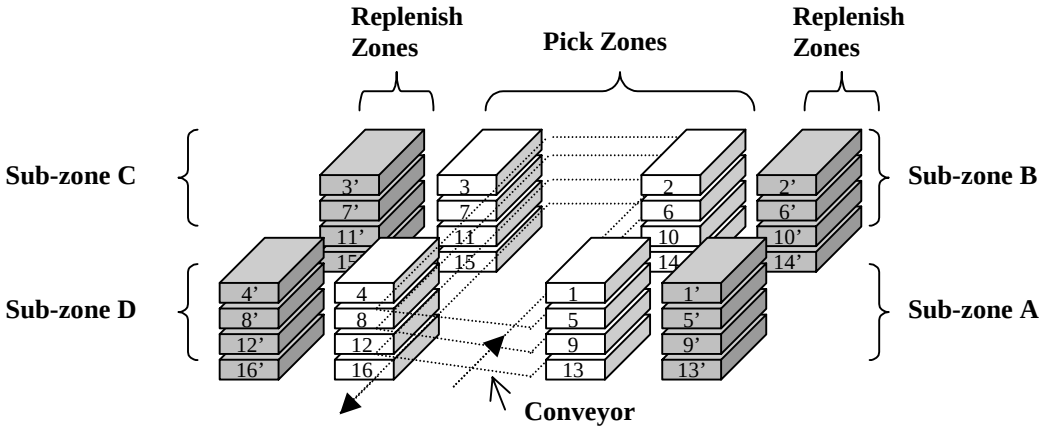


Fig. 1. Gantry Picking Complex (GPC) layout

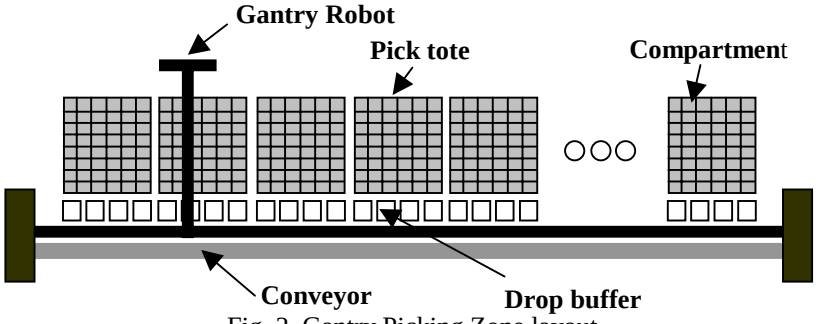


Fig. 2. Gantry Picking Zone layout

This industrial warehouse order picking problem mainly includes three tasks: order sequencing, resource assignment for each line item, and generating the order picking sequence for each gantry robot. The final task is strongly related to the other two. It would therefore be desirable to formulate the three tasks together using a single model. However, such a model will be too complex to use in real time decision making environments. In fact, modeling the combined problem of resource assignment for each line item and order picking sequence generation for each gantry robot itself is too complex to solve optimally in reasonable time. Thus we use a simple heuristic for the latter problem. Since the length of a pick zone is significantly longer than the width, the sequencing of pick locations based on the locations' x-coordinates is likely to give a near optimal solution (see Appendix for a detailed discussion on this aspect). Because the order picking is done in batches, order train by order train, we formulate the problem for each order train. The linear integer programming model is shown below.

Let X_{pij} $\begin{cases} 1 \text{ if } i \neq j \text{ and the tour uses a link from location } i \text{ to } j \text{ in pick zone } p \\ \quad \text{where, } i \leq j \text{ and x\_coordinate of location } i \leq \text{x\_coordinate of location } j \\ \geq 1 \text{ if } i = j \text{ and the tour picks items from location } i \\ 0 \text{ otherwise} \end{cases}$

O_k = number of units of SKU type k to be picked in order train

L_k = set of locations containing SKU type k

q_{pi} = number of units at location i in pick zone p

N_p = number of locations, which have SKUs for the order train in pick zone p

$X_{p0\,j}, X_{piN_p+1}$ = special variables for subtour elimination

C_{pij} = gantry movement time (Tchebychev metric) from location i to j in pick zone p

The order picking problem,

Objective: min $MaxTSP$

Constraints:

$$\sum_{j\geq i}^{N_p+1} X_{pij} \leq q_{pi} \quad \text{for each } i,\ p \tag{1}$$

$$\sum_{p=1}^{16} \sum_{i\in L_k} \sum_{j\geq i}^{N_p+1} X_{pij} = O_k \quad \text{for each } k \tag{2}$$

$$MaxTSP \geq \sum_{i=1}^{N_p-1} \sum_{j=i}^{N_p} C_{pij} X_{pij} \quad \text{for each } p \tag{3}$$

$$\sum_{j=i+1}^{N_p+1} X_{pij} \leq 1 \quad \text{for each } i,\ p \tag{4}$$

$$\sum_{i=0}^{j-1} X_{pij} \leq 1 \quad \text{for each } j,\ p \tag{5}$$

$$\sum_{i=0}^{j-1} X_{pij} = \sum_{k=j+1}^{N_p+1} X_{pjk} \quad \text{for each } j,\ p \tag{6}$$

$$\sum_{j=1}^{N_p} X_{p0\,j} = 1 \quad \text{for each } p \tag{7}$$

$$\sum_{i=1}^{N_p} X_{piN_p+1} = 1 \quad \text{for each } p \tag{8}$$

$$\sum_{j\geq i}^{N_p+1} X_{pij} + M(1\text{-}Z_{pii}) \geq 1 \quad \text{for each } i,\ p \tag{9}$$

$$\sum_{j\geq i}^{N_p+1} X_{pij} - M(Z_{pii}) < 1 \quad \text{for each } i,\ p \tag{10}$$

$$X_{pii} = \sum_{j\geq i}^{N_p+1} X_{pij} - Z_{pii} \quad \text{for each } i,\ p \tag{11}$$

$$X_{pij} = \text{non negative integer} \tag{12}$$

$$Z_{pii} = 0 \text{ or } 1 \tag{13}$$

While minimizing the maximum gantry movement time of the sixteen gantry robots as well as their variance simultaneously is a reasonable objective function, we minimize only the maximum gantry movement time in our objective to obtain a linear objective function. Variable X_{pij} represents whether the link between location *i* and location *j* in pick zone *p* is included in the gantry robot tour. If X_{pij} is a positive integer, it means the gantry robot of pick zone *p* will pick a SKU from location *i*, drop it to an available drop buffer in that pick zone, and move to location *j* and pick a SKU. If *i* and *j* are different, the possible maximum value of X_{pij} is 1. This is to prevent cycling and is enforced by constraints (4) and (5). While the value of a link is zero or one in a typical traveling salesman problem (TSP), since more than one SKU from one particular location can be picked for the order train, X_{pii} is allowed to have a larger integer value. For example, $X_{pii} = 3$ means that 3 items are picked from location *i* (pick a SKU, transfer it to a drop buffer, and back to location *i*) in pick zone *p*. Notice that since a gantry robot can pick only one SKU at a time and it has to bring the SKU to a drop buffer, the movement time from location *i* to itself is not zero. X_{p0j} and X_{piN_P+1} are introduced only for modeling purposes. Every gantry robot tour starts at a virtual location 0 and ends at N_P+1.

Constraint (1) is to ensure that the total number of picks from a location cannot exceed the number of units stored in it (capacity constraint). Constraint (2) is an order fulfillment constraint. Constraints (3) is an expression of the maximum gantry movement time of the sixteen gantry robots. Constraints (4) and (5) are introduced to prevent cycling. Constraint (6) ensures that if a gantry robot arrives at an intermediate location, it must leave that location. Notice that each gantry robot tour starts at a virtual location 0 and ends at N_P+1. Constraints (9), (10), (11), and (13) are introduced to eliminate sub tours at each location. If there is more than one pick at a location, the number of links from location *i* to itself should be one less than the total number of picks. Without these constraints, it is possible for a gantry robot to have links from location *i* to itself without ever entering or leaving that location.

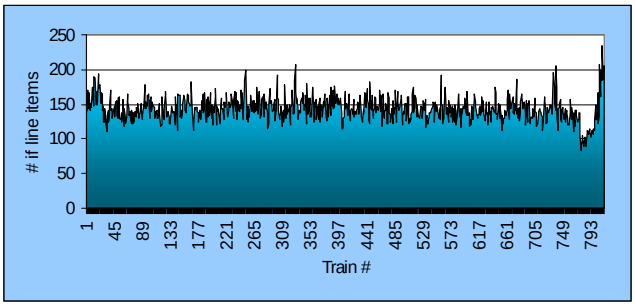
The model is complex. For a sample order stream and SKU location data, we get an average of $1.31(10^{47})$ possible ways to assign resources for an order train with 80 order trays. If the problem has *P* pick zones, *K* SKU types in an order train, and N_p locations which have SKUs for the order train in pick zone *p* (assume $N = N_p$ for each *p*), there are $NP(N+1)/2 + NP + 2$ variables, $7NP + 3P + K$ constraints. Thus, if $P = 16$, K=140, and N=16, there are 2,434 variables and 4,412 constraints (including 2,432 integer constraints). LINGO 7.0 (LINDO Systems Inc.) takes approximately 30 minutes to get to within 2% of the optimal solution for this problem on a 650 MHz Pentium machine. Since there are more than 800 order trains per day, the model solution would take more than 400 hours, making it impractical to solve using a general purpose integer programming solver in real time environments.

Kim et al. [18] apply an intelligent agent-based hybrid modeling framework to this problem to determine the pick zone assignment for each line item. By dynamically adjusting conveyor speed using a carefully constructed negotiation mechanism, they demonstrate a higher level of performance and reactivity compared with the existing hierarchical framework. In this paper, we extend the proposed conceptual framework to a scheduling and control architecture, and show an even higher level of performance and reactivity compared to a pure hierarchical or pure heterarchical architecture.

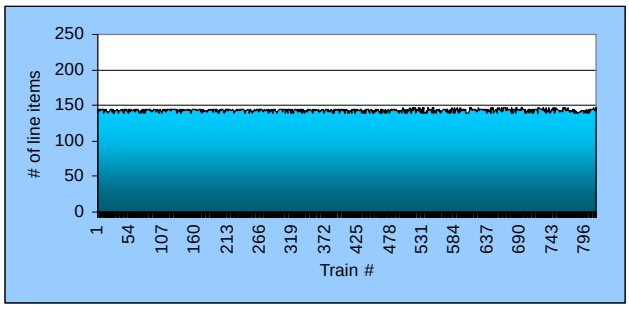
IV. Need for Higher Level Optimizer and Communication Between Higher and Lower Level Controllers

Even though a heterarchical framework, in which unit controllers negotiate with each other and make their own decisions without higher level intervention, can provide robustness against disturbance, a pure heterarchical system has difficulty in providing a globally optimized solution because it does not have a global system perspective. To provide a global system view, a hierarchy is needed. Two problems arising in the context of this industrial application are discussed to demonstrate this need.

The first problem has to do with the workload variability in the input order stream. Consider the input order stream for an example day shown in figure 3a. The higher degree of workload variability between order trains increases the likelihood of pick errors. The system, therefore, cannot operate at a higher speed as this will increase the number of pick errors even more. To balance the workload as shown in figure 3b, we need a higher level agent with a global system perspective.



(a)



(b)

Fig. 3. (a) Unbalanced Order Sequencing, (b) Balanced Order Sequencing

The second problem has to do with the order sequence in an order train. Consider figures 4a and 4b. The sequence of line

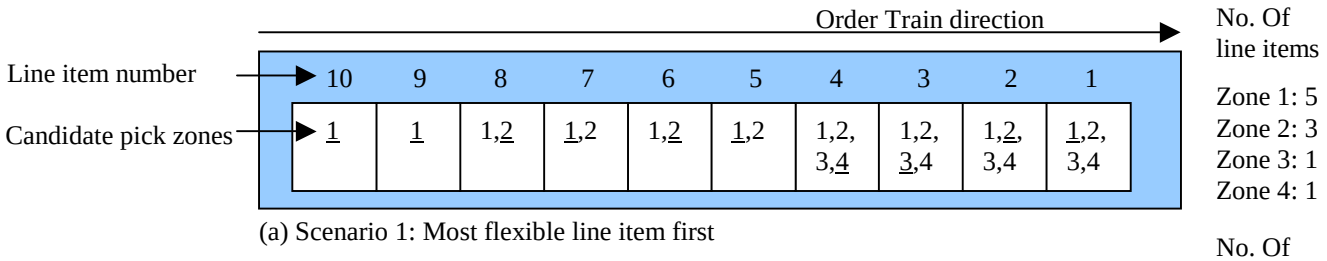
items presented for order picking in figure 4a is such that line items with the most picking flexibility (i.e., those items stored in multiple pick zones) are presented first. The numbers in each small box show the candidate pick zones. For example, the line item number 5 can be filled either at pick zone 1 or pick zone 2. If a negotiation based mechanism, such as the ones developed in [18], [21], or [22], is used to determine the specific gantry that must pick a specific line item, the line item sequence in figure 4a may lead to inferior pick zone/gantry to line item assignments. While five and three line items are assigned to pick zone 1 and 2 respectively, only one line item is assigned to pick zone 3 and 4 for the order sequence in figure 4a, causing uneven workload amongst the pick zone gantry robots. On the other hand, pick zones 1, 2, 3, and 4 will fulfill three, three, two, and two line items, respectively for the order sequence in figure 4b. Clearly, the second sequencing method allows workload to be more evenly balanced amongst the four gantries. A higher degree of workload variability between gantry robots increases the likelihood of pick errors. A higher level agent with a broader perspective thus generates a better order sequence.

The first workload variability problem can be solved by swapping orders between order trains according to the number of line items of the orders. An order from the order train with the maximum workload containing the most number of line items is swapped with the order containing the least number of line items from the order train that has the least workload. The swapping process is repeated until the workload variability is within a predefined range. The second resource assignment problem is more difficult to solve. If each order has only one line item such as previous example, the sequencing based on flexibility and assigning resources to each order using negotiation can generate a well balanced schedule. However, since each order may have multiple line items and they have different flexibility in terms of number of candidates, generating an order sequence in increasing order of flexibility is not always possible. We could use an integer programming model (such as the one presented in the previous section) to determine the resource for each line item, but solving such a model is not practical as pointed out in the previous section. We use the genetic algorithm (GA) to solve this problem and it is described in the next section.

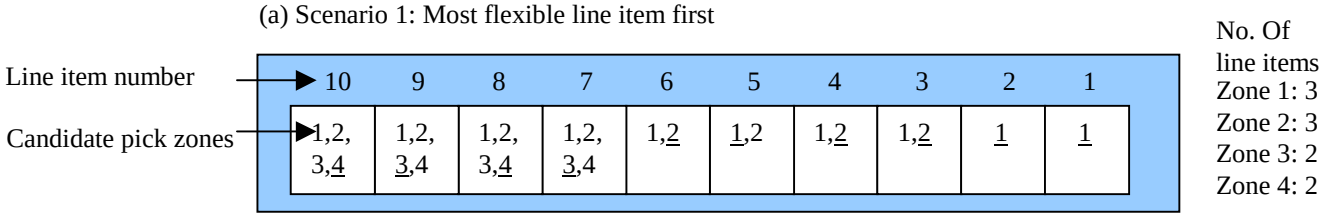
The two problems presented above highlight the need for a higher level agent to make decisions globally. However the higher level global optimizer assigns resources for each line item under the assumption that all pick zones operate continuously. As Szelke and Kerr [3] point out, however, in today's manufacturing or service environment, fast changing customer demands such as urgent order insertion and order cancellation induce external disturbance, while unexpected or unplanned events such as machine breakdowns and rework, induce internal disturbance. These unforeseen disturbances invalidate the plan and schedule at an operational level. Thus, the reactivity of the heterarchical framework is still needed. In other words, a hybrid modeling framework such as the one outlined in Heragu et al. [2] is required. A scheduling and control architecture that combines hierarchical and heterarchical features is presented in the next section.

V. Scheduling and Control Architecture

Figure 5 shows the proposed scheduling and control architecture that can provide robust solutions while achieving system wide objectives. While this architecture is for the industrial warehouse problem described in section III, it can be generalized to job shop scheduling and other manufacturing control environments. The higher level agent, which has a global perspective, prepares an order sequence so that (i) the number of line items is evenly balanced amongst the order trains and (ii) inflexible line items are ahead of the flexible ones in each order train. Resources are then assigned for each line item. While the middle level guide agent and lower level agents are busy handling the current order picking cycle, the higher level agent can prepare the next. It has to prepare the resource replenishment plan as well as the other tasks described above.



(a) Scenario 1: Most flexible line item first



(b) Scenario 2: Most inflexible line item first

Fig. 4. Importance of decision sequence (numbers underlined represent assigned pick zones)

The middle level guide agent takes the schedule and guides the lower level agents. The order agents know the preliminary resources (gantries/zones) assigned to them when they enter the system. However, the order agent can reselect resources based on the real time shop floor conditions by negotiating with the gantry robot agents. It announces its tasks on the task board, gets bids from the available resources (pick zones/gantry robots), and selects the best bid. If the gantry robot selected is different from the one assigned by the higher level agent, the order agent asks the middle level guide agent for permission to change its assigned resource. The order agent must provide information on the current participants, especially the originally assigned resource, the best bidder, and bid data to the middle level agent so that the middle level agent can decide whether or not to grant permission. The logic behind this is that since the middle guide agent has a more global view, the final decision has to be made by the middle level agent rather than the lower level order agent.

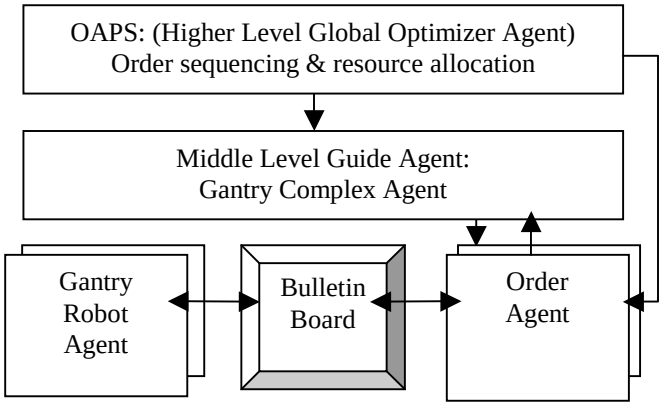


Fig. 5. Scheduling and Control Architecture

The middle level guide agent maintains updated aggregate planning information such as the number of line items assigned to each resource. When it receives a request to change the original assignment, it examines the projected workload of all 16 gantry robots based on the line items in the current train, and grants the request only if the assignment change is necessary or desirable. For example, suppose pick zones 1 and 5 have 11 and 8 items respectively to pick for the current order train in the original plan. Upon further negotiation amongst the lower level agents, let us assume that an order originally assigned to pick zone 1 is now reassigned to zone 5. The order agent now requests permission from the middle level agent to execute the resource change. Since this change results in a better balanced plan (from 11:8 to 10:9), the middle agent grants the request. Thus the hybrid architecture requires interactions between higher level, middle level, and lower level agents.

While the hybrid architecture attempts to maintain the plan of the higher level global optimizer because it can provide higher performance under normal operating conditions, it also provides flexibility in reacting to disturbances. Moreover this architecture can be easily converted to a pure hierarchical or pure heterarchical architecture if necessary. If the middle level agent is bypassed, we have a heterarchical structure. If negotiation at the lowest level is prohibited, we have a hierarchical architecture.

For the resource assignment, the higher level agent uses a genetic algorithm (GA) whose procedures are as follows [19].

Step 0: Obtain the population size P and the desired number of generations G from the user, generate P solutions for the first generation's population randomly, and represent each solution as a string. Set generation counter $N_{gen} = 1$.

Step 1: Determine the fitness of each solution in the current generation's population.

Step 2: Generate solutions for the next generation's population as follows. Retain 0.2 P of the solutions with the best fitness in the current population. Generate 0.78 P solutions via mating, and select 0.02 P solutions from the retained 0.2 P population randomly and mutate them.

Step 3: Update $N_{gen} = N_{gen} + 1$. If $N_{gen} \# G$, go to step 1. Otherwise, stop.

If an order train has L line items to be picked, a solution string (chromosome) will be of size L. Each character (integer) in the string represents the compartment index for the line item and each compartment (see figure 1b) has a unique index. For example, a solution string {10, 2, 801, 90, 37, 90, …, 567} indicates that line item #1 will be fulfilled from compartment 10, line item #2 from compartment 2, etc. The only compartments that have proper SKUs are the candidates for the specific character. For example, if line item #1 needs a specific SKU type and only compartments 10 and 25 have that SKU type, these are the only candidates for the first character in the string. In step 0, the GA generates feasible solutions by random selection of candidates for each character of the string.

From a feasible solution, workload (assigned line items) for each pick zone can be determined. Then the fitness of the solution can be calculated. Any linear or non-linear evaluation function such as variance of gantry moving times, the maximum gantry moving time among 16 gantry robots, or a combined one, can be used as a fitness function. We use the maximum gantry moving time of 16 gantry robots as a fitness function in our experiments.

We have used generally recommended values for the various parameters in our implementation of the GA (see [19]). Experimentation has found that these values work well for our application. For the next generation's population, 20% of the solutions with the best (smallest) fitness in the current population are selected and retained. 78% of the next generation's population is generated by mating between the selected 20% population. We use a two-point crossover mating method. In the two-point crossover, given randomly selected two parent chromosomes $\{x_1, x_2, \ldots, x_L\}$ and $\{y_1, y_2, \ldots, y_L\}$, two integers r and s for each pair of parents such that $1 \# r < s \# L$, are randomly selected and the genes in positions r to s of one parent are swapped with those of the other to get two offspring: $\{x_1, x_2, \ldots, x_{r-1}, y_r, y_{r+1}, \ldots, y_s, x_{s+1}, x_{s+2}, \ldots, x_L\}$ and $\{y_1, y_2, \ldots, y_{r-1}, x_r, x_{r+1}, \ldots, x_s, y_{s+1}, y_{s+2}, \ldots, y_L\}$. The remaining

2% of solutions are generated by mutation. In the mutation process, given a randomly selected parent chromosome $\{x_1, x_2, \ldots, x_L\}$, an integer r is randomly selected and the gene in the position r is altered randomly among candidates. Since mating and mutation operations only change numbers within candidates for each character, they generally generate feasible solutions. However, there are situations where a generated solution is not feasible. For example, suppose the current order train needs two units of SKU type A for two line items and there is a compartment that has only one remaining SKU type A in it. Since the compartment is a candidate for the two line items, a generated solution may have two of the compartment index, rendering the solution infeasible. In that case, one of them must be changed to another candidate compartment. We randomly select one among the other available candidates to resolve the problem in our implementation.

The above procedure is continued until a specified number of generations have been examined or there is no improvement in the fitness of the best individuals for several generations. The best solution obtained is then the final resource assignment solution.

For an example order train, the objective values of the best solutions of the integer programming model and Genetic Algorithm (GA) are 31.68 and 33.02, respectively. GA takes 50 minutes and 10 seconds for a one-day order (815 order trains) on a 350 MHz Pentium pro machine with 128 MB RAM and the average of objective values is 33.51.

In order to compare the effectiveness of the proposed architecture with others, a simulation study is conducted. The hierarchical and heterarchical models are obtained from the hybrid model by bypassing the middle-level agent and by prohibiting any change from the original plan respectively, as we described at the beginning of this section.

For the study, the Finite Scheduling System (FSS) provided by the Lareal Company is used. It is implemented in Gensym Corporation's G2, which is a simulation and real time control system language. The simulation and real time control capability is a distinctive feature of the G2 system. Since FSS has been applied to the industrial warehousing problem, its validity is already demonstrated. Since the change of the FSS for our study is limited to the logical resource assignment part, the validity of our model is also demonstrated. It runs on the same machine described above.

One day's real order stream generated by the Order Analysis and Planning System (OAPS) is also provided by the Lareal Company. Figure 3(a) shows the stream in terms of number of line items in each order train. This order stream is modified to a more balanced one as shown in figure 3(b) in the hierarchical and hybrid models. However, in order to preserve the heterarchical aspects, the original input order stream is not modified in the heterarchical model. The order stream balancing and the GA based resource assignment algorithm are implemented in C++ programming language. The modified order stream for the hierarchical and hybrid models and the original order stream generated by OAPS for the heterarchical model are the input order streams to the FSS simulator. In order to ensure repeatability of the simulation model and study the effect of order variability, two additional order streams for one day are generated by changing the sequence and quantity of orders from the example order stream. 5% and 10% of line items are eliminated randomly from the example order stream.

Table 1 compares the performance of the three models by experimenting with different conveyor speeds. There are 116,589 line items to be picked for the simulation day. A conveyor feed interval of 0.65 seconds means that there is a conveyor movement every 0.65 seconds. Thus a conveyor feed interval of 0.65 seconds represents slower movement than a 0.60 seconds interval. All three models are similar with respect to the first two moments of resource utilization levels. The mean utilization in the table refers to the average utilization of picking robots and the standard deviation shows the variability of the robots' average utilization levels. Further, hierarchical and hybrid models show similar results with respect to the number of pick errors. Results show that the pick errors of the hierarchical and hybrid models are always less than those of the pure heterarchical negotiation based model. Zero errors for the hierarchical and hybrid models means that all orders are filled completely without any line items missing. In other words, the gantry robots have enough time to pick all line items in advance before order trays reach their pick zones. There is some variation in the number of pick errors for the heterarchical model for the three replications in table 1 because of the random seeds used in the G2 software program.

In order to investigate the robustness of each model against disturbances, an artificial machine breakdown scenario is simulated. Gantry robots 1, 2, 3, and 4 are down between 10,000 and 20,000 'simulation' seconds. Table 2 shows the higher schedule quality of the hybrid architecture compared with a hierarchical or heterarchical model with machine breakdowns in terms of pick errors. In the heterarchical and hybrid models, when the four gantries are out of order, the remaining 12 are able to absorb the tasks of the down gantries based on the real time negotiation mechanism provided they have the needed SKUs. The hybrid model has 2779 pick errors, whereas the hierarchical and heterarchical have 5985 and 2807 errors respectively. The new hybrid model consistently outperforms the pure hierarchical or heterarchical model when machine breakdowns are considered in the experiment.

VI. CONCLUSIONS

Using an industrial order picking problem, the need for a higher level global optimizer and the need for communication between higher level and lower level controllers are demonstrated. A hybrid scheduling and control system architecture for achieving robustness and system wide objectives, is proposed. It has features of hierarchical and heterarchical models.

Table 1. Schedule quality comparison between hierarchical, heterarchical and hybrid model

Unit: %

	Order stream 1 with conveyor feed interval of 0.65 sec			Order stream 2 with conveyor feed interval of 0.63 sec			Order stream 3 with conveyor feed interval of 0.60 sec								
							Replication 1			Replication 2			Replication 3		
Zone	hier.	heter.	hybrid	hier.	heter.	hybrid	hier.	heter.	hybrid	hier.	heter.	hybrid	hier.	heter.	Hybrid
1	91.6	91.8	91.4	90.8	91.6	91.3	92.0	92.8	92.5	92.0	92.9	92.6	92.6	92.6	92.6
2	90.8	91.4	92.1	90.6	91.3	91.7	91.8	92.5	93.0	91.5	92.4	92.6	91.6	92.4	92.7
3	90.6	91.7	91.7	90.6	91.3	91.6	91.7	92.4	92.5	92.3	92.7	92.9	91.5	92.7	92.8
4	91.3	91.8	91.7	90.4	91.7	91.7	92.0	92.8	92.9	92.2	92.8	92.8	92.4	92.9	92.7
5	90.1	90.2	90.5	90.2	90.1	89.9	91.2	91.2	91.4	91.1	91.4	91.6	90.7	91.4	91.8
6	90.0	90.2	90.2	89.9	89.8	90.2	91.4	91.1	90.8	91.1	90.7	91.3	91.2	90.9	91.5
7	91.1	90.9	91.1	90.2	91.0	90.3	91.6	91.4	91.9	91.2	91.2	91.8	91.4	91.2	91.8
8	90.7	90.6	90.3	90.8	90.3	89.8	91.6	91.2	91.4	91.2	91.6	91.4	91.3	91.6	91.4
9	90.6	91.0	91.4	90.1	90.4	90.9	91.3	91.4	92.2	91.1	91.3	92.0	91.4	91.6	92.0
10	90.8	91.3	91.4	90.6	90.9	90.9	91.4	92.4	92.4	92.0	92.5	92.3	91.5	92.3	92.6
11	91.2	90.9	91.5	90.4	90.7	90.7	91.5	91.6	92.1	91.3	91.4	92.3	91.8	91.7	92.4
12	90.5	90.9	91.3	90.3	90.4	90.9	91.5	92.1	92.4	91.4	92.5	92.4	91.5	91.9	92.3
13	90.8	90.8	91.3	90.5	90.5	90.8	91.3	91.5	91.9	91.4	91.3	91.6	91.0	91.2	91.6
14	90.6	90.1	90.5	90.5	90.2	90.1	91.2	91.0	91.5	91.4	91.3	90.9	91.7	91.1	91.2
15	90.9	91.0	91.4	90.5	90.5	90.9	92.0	91.8	91.9	92.0	91.8	91.9	91.7	91.5	91.8
16	90.2	90.5	90.8	90.0	90.7	90.7	91.7	91.8	92.0	91.9	91.5	92.0	92.0	91.6	92.2
Average Util. (%)	90.7	90.9	91.2	90.4	90.7	90.8	91.6	91.8	92.1	91.5	91.8	92.0	91.5	91.8	92.1
Util Std dev (%)	0.4	0.5	0.6	0.3	0.5	0.6	0.3	0.6	0.6	0.4	0.7	0.6	0.5	0.6	0.5
No. of errors	0	329	0	0	159	0	0	272	0	0	234	0	0	222	0

Table 2. Schedule and control quality comparison under machine breakdown, with conveyor interval 0.75 sec

	Replication 1			Replication 2		
Architecture	Hier.	Heter.	Hybrid	Hier.	Heter.	Hybrid
No. of errors	5985	2807	2779	5983	2921	2716
Average Utilization (%)	75.3	76.1	76.1	74.6	76.2	76.2
Standard Deviation of Util.	6.5	8.3	8.2	7.1	8.3	8.3

The higher level optimizer agent, with its global perspective, makes a balanced and synchronized order sequence and efficiently assigns resources to each order. A genetic algorithm is presented for the resource assignment problem. The middle level guide agent takes the resource assignment decision from the higher level agent and guides the lower level agent to achieve better system wide objectives. The lower level agents make their decisions based on real time conditions, and suggest the alteration of predetermined resource assignments but have to obtain permission from the middle level agent. The hybrid architecture is applied to an industrial warehouse order picking problem and its effectiveness is demonstrated using test problems.

The proposed architecture can be applied to different warehousing and manufacturing systems. A more generalized hybrid framework is presented in our previous work [2]. Furthermore, the architecture can be easily integrated into a warehouse management system (WMS). The OAPS and the FSS in this paper are in fact, parts of a WMS system.

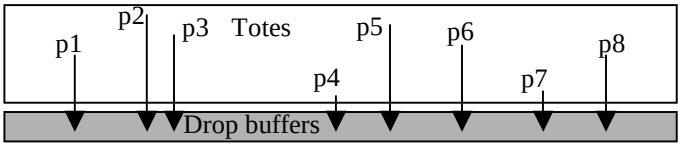
APPENDIX

Generation of the order picking sequence for gantry robot

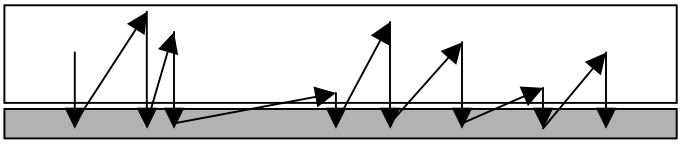
After the line items that need to be fulfilled from a pick zone are determined for an order train, the order picking sequence needs to be generated for the picking gantry robot of that pick zone. In general, the order picking sequence problem can be modeled as a traveling salesman problem [19]. The gantry robot deposits a product at the nearest drop buffer after picking it, making the gantry robot sequencing problem a special type of traveling salesman problem (figure A-1-a).

Since the length of a pick zone is significantly longer than

the width in our application, the sequencing of pick locations based on their x-coordinates gives an optimal solution in many instances (see figure A-1-b). Specifically, when the pick locations are sparse enough such that the horizontal movement time of the gantry robot between any two possible locations (one is the drop buffer for the previous location and the other is the next location) is always longer than the vertical movement time, the x-coordinate based sequencing method generates an optimal solution (figure A-2). Because the gantry robot has two independent motors, one permitting movement in the horizontal direction and the other in the vertical direction, the movement time between two locations depends on the longer of the horizontal and vertical moving times. Thus when the pick locations are sparse enough such that the horizontal movement time between any two locations of the gantry robot is always longer than the vertical movement time, the gantry moving time depends only on the horizontal movements. (Note that the vertical movement from a pick location to a drop buffer is the same for any order picking sequence.) The x-coordinate based sequencing method generates an optimal solution since the sum of the movement times between two consecutive locations equals the movement time between the start (far west) location and the end (far east) location, which is the minimum possible time (figure A-2).

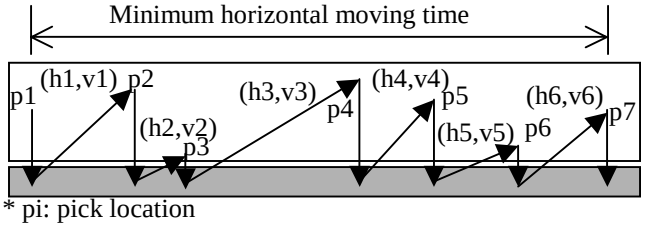


(a) Gantry robot order picking sequence problem



(b) x-coordinates based sequencing solution

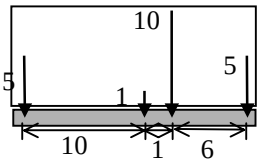
Fig. A-1. Gantry robot order picking sequence problem



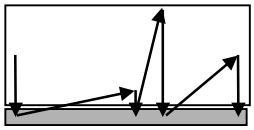
* pi: pick location
* (hi, vi): horizontal and vertical moving time from the drop buffer for pi to location i+1

Fig. A-2. Case when an optimal solution is guaranteed by the x-coordinate based sequencing

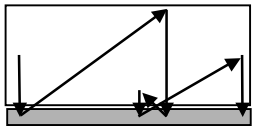
When the sparsity condition for the pick locations is not met, as in fig. A-3, the x-coordinate based sequencing method cannot guarantee optimality. The numbers in figure A-3a correspond to movement times between various points. Because the horizontal length of a pick zone is at least 19 times longer than the vertical length, in our application, the likelihood of meeting the sparsity condition is high. Thus the x-coordinate based sequencing heuristic often provides an optimal solution. A more through investigation of this aspect as well as a more efficient algorithm for generating the gantry robot's order picking sequence is provided in [23]. Experimental results in [23] indicate that the x-coordinate based heuristic provides near-optimal results. Furthermore, because the order pick sequence generation is a relatively minor problem amongst the others considered in this paper and also because the same heuristic is used in all three models, the results comparing the different frameworks are not significantly affected.



(a) problem



(b) moving time=47
=5+10+1+10+10+6+5



(c) moving time=40
=5+11+10+1+1+7+5

Fig. A-3. Case when the x-coordinate based sequencing is not an optimum

REFERENCES

[1] C. S. Shukla and F. F. Chen, "The state of the art in intelligent real-time FMS control: a comprehensive survey," *Journal of Intelligent Manufacturing*, vol. 7, pp.441-455, 1996

[2] S. S. Heragu, R. J. Graves, B. Kim, and A. St. Onge, "Holonic/Intelligent agent based framework for manufacturing systems control," DSES Technical Report #38-00-480, Rensselaer Polytechnic Institute, Troy, NY., Sep. 2000, Available at http://www.rpi.edu/~herags

[3] E. Szelke and R. M. Kerr, "Knowledge-based reactive scheduling," *Production Planning and Control*, vol. 5, no. 2, pp. 124-145, 1994

[4] H. V. Brussel, J. Wyns, P. Valckernaers, and L. Bongaerts, "Reference Architecture for Holonic Manufacturing Systems: PROSA," *Computers in Industry*, vol. 37, pp. 255-274, 1998

[5] F. Maturana, W. Shen, and D. H. Norrie, " MetaMorph: an adaptive agent-based architecture for intelligent manufacturing," *International Journal of Production Research*, vol. 37, no. 10, pp. 2159-2173, 1999

[6] T. A. Ottaway and J. R.Burns, "An adaptive production control system utilizing agent technology," *International Journal of Production Research*, vol. 38, no. 4, pp. 721-737, 2000

[7] M. M. T. Giebels, " EtoPlan: a Concept for Concurrent Manufacturing Planning and Control – building holarchies for manufacture-to-order environments," PhD thesis, University of Twente, Netherlands, 2000

[8] R. Tawegoum, E. Castelain, and J.C. Gentina, "Hierarchical and Dynamic Production Control in Flexible Manufacturing Systems," *Robotics & Computer-Integrated Manufacturing*, vol. 11. no. 4, pp. 327-334, 1994

[9] S. C. Graves, "A Review of production scheduling," *Operations Research*, vol. 29, no. 4, pp.646-675, 1981

[10] H. V. D. Parunak, "Characterizing the manufacturing Scheduling Problem," *Journal of Manufacturing Systems*, vol. 10, no. 3, pp.241-259, 1991

[11] N. Sadeh and M. S. Fox, “Focus of attention in an activity-based scheduler,” in *Proc. NASA Conf. On Space Telerobotics*, NASA, Bethesda, MD., pp.425-434, 1989

[12] K. P. Sycara, S.F. Roth, N. Sadeh, and M.S.Fox, “Resource allocation in distributed factory scheduling,” *IEEE Expert*, vol.6, no.1, pp.22-40, 1991

[13] E. Szelke and L. Monostori, “Reactive Scheduling in Real Time Production Control,” In *Modeling Manufacturing Systems: from aggregate planning to real time control*, P. Brandimarte and A. Villa (eds.), New York: Springer, 1999, ch. 4, pp. 65-113

[14] S. F. Smith, “OPIS: a methodology and architecture for reactive scheduling,” in *Intelligent Scheduling*, M. Zweben and M. S. Fox (eds.), Morgan Kaufmann Publishers, San Fransisco, 1994, ch. 2, pp. 29-66

[15] L. Bongaerts, “Integration of scheduling and control in holonic manufacturing systems,” PhD thesis, Katholieke Universiteit Leuven, Belgium, 1998

[16] B. Rouwenhorst, B. Reuter, V. Stockeahm, G. J. van Houtum, R. J. Mantel, and W. H. M. Zijm, “Warehouse design and control: Framework and literature review,” *European Journal of Operational Research*, vol. 122, pp. 515-533, 2000

[17] J. P. Van den Berg, “A literature survey on planning and control of warehousing systems,” *IIE Transactions*, vol. 31, pp. 751-762, 1999

[18] B. Kim, R. J. Graves, S. S. Heragu, and A. St. Onge, "Intelligent Agent Based Model for an Industrial Order Picking Problem," in *Proc. of the 10th Annual Industrial Engineering Research Conference*, May 20-22, 2001, Dallas, TX

[19] S. S. Heragu, *Facilities Design*, PWS Publishing Company, Boston, MA, 1997

[20] L. J. Bain and M. Engelhardt, *Introduction to Probability and Mathematical Statistics*, Belmont, California: Duxbury Press, 1992

[21] A. D. Baker, “Manufacturing Control with a Market-Driven Contract Net,” PhD Thesis, Rensselaer Polytechnic Institute, Troy, New York, 1991

[22] G. Y. Lin and J. J. Solberg, “Integrated Shop-Floor Using Autonomous Agents,” *IIE Transactions*, vol. 24, no.3, pp. 57-71, 1992

[23] B. Kim, “Intelligent Agent Based Hybrid Control Architecture for Warehouse Management,” Unpublished Ph.D dissertation, Rensselaer Polytechnic Institute, 2002