A heuristic solution for order picking problem in unit-load automated storage and retrieval systems

Yacob Khojasteh (khojast@sophia.ac.jp)
Graduate School of Global Studies, Sophia University, Yotsuya, Tokyo 102-8554, Japan

Abstract

In this study, we consider an end-of-aisle unit load automated storage and retrieval system (AS/RS), where one storage and retrieval machine is dedicated to all aisles. The machine is able to travel in cross warehouse aisle through a transfer car called "traverser", so that it can enter any pick aisle. When retrieval requests are made for multiple items and the items are in multiple stock locations, there will be a huge number of feasible solutions with different retrieval times. The objective is to minimize the total time travelled by the machine to complete the retrieval process of customer orders.

Keywords: Automated storage and retrieval systems, Multi-aisle automated warehouse, Order picking

Introduction

The power of warehousing system to rapidly respond to customer demands participates an important function in the success of a supply chain. Automated warehouses are widely used in manufacturing, warehousing, and distributions applications. Automated storage and retrieval systems (AS/RSs) are basic components of automated warehouses. An AS/RS allows for more efficient utilization of warehouse space, increasing in speed and accuracy of operations, removing human factor from the equation, quicker response times, reducing the probability of damaging or losing goods, increasing security level, and reducing labour needed to operate warehouse productively.

One important operational aspect of the AS/RS is to minimize the total time or distance travelled by the storage and retrieval machine to complete the retrieval process of customer orders. Warehouse managers are interested in finding the most economical way of picking orders, which minimizes the costs involved in terms of travel distance or travel time. Order picking, which is a fundamental component of the retrieval function performed in warehouses, is a process by which products are retrieved from specified storage locations with respect to customer orders. By sequencing the retrievals in a smart way, improvements in the overall throughput of the AS/RS can be obtained.

Several approaches have been used to find solutions to the order picking problems: genetic algorithms (e.g., Krishnaiah and Sarveswar, 2003; Hsu et al., 2005; Khojasteh-Ghamari and Son, 2008), Petri nets (e.g., Amato et al., 2005), and neural networks (e.g., Wang and Yih, 1997). This study takes inspiration from our previous work (Khojasteh-Ghamari, 2012) and aims at continuing research in the same direction. In this paper, we focus on studying the current techniques in this field and ways to apply them to order picking problems and modify or improve them in order to achieve better results. First, we discuss and analyse the concepts of each studied methods. Second, we address possible techniques to improve them. Third, we conduct numerical experiments for performance assessment of proposed solutions, followed by discussions and conclusions.

Warehouse model and problem description

An end-of-aisle order picking automated storage and retrieval system is considered in which there are one or more aisles and each aisle contains a storage racks on both sides of an aisle. Input-output stations are located at the end of each aisle at the same level as the first rows of the racks. A single storage/retrieval (S/R) machine is employed which services all the aisles. The machine can simultaneously move in both horizontal and vertical direction. Due to this, machine can travel in accordance with Chebyshev's travel concept, so the travel time to some point in the automated warehousing system will be decided by the maximum travel time of horizontal and vertical travel

When retrieval requests are made for multiple items and the items are in multiple stock locations, there will be a huge number of feasible solutions with different retrieval times. The objective is to minimize the total time travelled by the S/R machine to complete the retrieval process of customer orders.

Solution methods

Enumeration

A simple genetic algorithm is presented to solve the problem. For illustration of the superior nature of the algorithms, it is imperative to contrast them against other techniques as well as optimal solutions. In view of this, an algorithm for obtaining the ideal solution to the problem known as the enumeration algorithm is presented. The outcomes are utilized for benchmarking solutions to compare the performance of the other suggested algorithm.

With this method, number of all feasible solutions to the problem that have to be evaluated has rapid rate of increase with increase in number of items stored in a warehouse or number of items in the order. With high number of items in the order or stored in the warehouse, this algorithm could have impractical calculation times or even impossible to compute with the level of current technology and in general. In order to still get the solution to the problem it will be required to use a different approach for such cases.

Genetic algorithm

We develop a genetic algorithm to solve the order picking problem.

Representation

For the order-picking problem, a chromosome denotes a possible solution in which each is considered a genetic sequence alongside its related allele. Notably, all genes within chromosomes denote the kind of item, whereas their related allele denotes the site of storage (location in the warehouse). In view of this, each possible solution is made up of one chromosome, where gene quantity is equivalent to the quantity of items within the order requested. Figure 1 illustrates a possible solution, *C1*, where five items #1, #2, #3, #4, and #5 are requested for collection.

#3[116]	#1[131]	#5[43]	#4[87]	#2[72]

Figure 1 – a chromosome C1

In the solution, items #1, #2, #3, #4, and #5, as well as their location numbers [131], [72], [116], [87], and [43] are marked for retrieval. Notably, the sequence for retrieving items has been taken into account within the representation. In the example, item #3 with [116] as the location code would be the first to be retrieved, followed by #1, #5, #4, and #2 alongside their respective location numbers [131], [43], [87], and [72].

Evaluation
At every generation, chromosomes are subject to a fitness function assessment. Since the issue is considered as a minimization problem, the value of objective function for every chromosome should be converted into fitness values, this has to be done for suitable chromosomes to have bigger fitness values.

Selection
Reproduction (selection) operator refers to the way individuals are chosen from a population when mutation and crossover parents combine to yield the following generation. A roulette wheel is utilized in the selection process for reproducing the following generation depending on the existing population where a suitable chromosome is better placed to reproduce in the following generation.

Crossover
Crossover operator allows the algorithm to obtain the ideal genes from various chromosomes and incorporating them to potentially viable offspring. As an example, consider this crossover of two random orders for 10 items. Randomly selected points: left 4 and right 6 as shown in Figure 2.

#5[182]	#4[48]	#3[151]	#1[157]	#7[173]	#9[193]	#8[77]	#2[51]	#6[57]

P1

#7[91]	#0[124]	#1[109]	#3[134]	#4[90]	#6[162]	#8[190]	#5[111]	#2[111]

P2

#9[193]	#7[173]	#3[151]	#1[157]	#4[48]	#6[57]	#8[77]	#5[182]	#2[51]

C1

#4[90]	#0[124]	#1[109]	#3[134]	#7[91]	#9[125]	#8[190]	#2[111]	#6[162]

C2

Figure 2 – Crossover operation

For the first child procedure goes as following: after copying all first parent's chromosomes, we then need to replace the one at position 5 (Item #7) with the item from the corresponding position of second parent (item 4). We locate Item 4 in first parent (position 2) and change it with Item #7 at position 5, while keeping the original indexes from the first parent for both of the items. The same operation is repeated for items #9 and #6. For the next Item #8 at position 7, however, there is no need to change anything, as both parents have the same item at this position. These operations are then repeated for the remaining position of the first child and for the second child, the only difference is that second child is taking second parent's chromosomes before the crossover.

Mutation
In order picking problem mutation changes only the location of the item, selecting from the list of the storage locations that have it. Mutation cannot impose changes over the sequence of the items. Position of the item to be mutated is selected randomly. Consider the following solution where position 2 (Item #9) has been randomly selected for mutation.

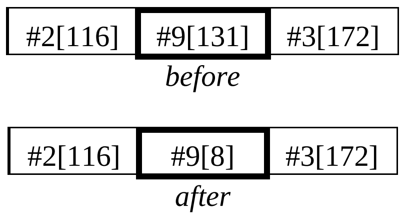

Figure 3 – Mutation operation

Available indexes for Item #9: 203 – 1 (One currently in use) = 202 possible positions. Randomly selected index 8 (it could have been any index from 1 to 203 range, except 131) out of available range.

Adaptive genetic algorithm (AGA)
Differences of AGA in current implementation:

- Adaptive max population with saw-edged population graph
- Reseeding with newly generated members at certain points
- Adaptive Pc and Pm (individually)
- Elitism operator

We consider combining these four techniques in particular in order to use their advantages to full extent, while allowing them covering disadvantages of each other (Xudong and Yunlong, 2013; Holland, 1992).

Numerical studies and findings

In order to evaluate and compare the performance of the proposed methods, we constructed a set of different warehouse configurations and different types of customer orders. Every sample is analysed with the help of enumeration, ordinary GA, Adaptive GA. Enumeration algorithm allows us to have an optimal solution, in order to be able to evaluate other algorithms' performance. 20 samples are considered for each warehouse configuration, making total sample number 180 for every order type.

We develop five scenarios based on five different types of customer orders, so that in the scenario number 1 to 5, the number of required items for retrieval is one to five, respectively. Initial locations of the all items in the warehouse are generated randomly.

Simulation model
Each rack has 35 number of columns and 12 Number of rows, so the total capacity of each rack is 420 items. Because racks are coming in pairs for one aisle, the total capacity of each aisle is 840 items. We also consider a warehouse with two, three and four aisles. Warehouse density and number of items are the two main parameters governing computation difficulty for this problem, as increase in these values can make number of feasible solutions skyrocket, especially with big orders, rendering enumeration algorithm unusable even for testing purposes. For warehouse density, such values as 0.6, 0.75, and 0.9 are suggested and used.

For each physical configuration of the warehouse, four different order structures are considered – for two, three, four and five items.

Table 1 – performance of the algorithms on the orders with two items; every tenth sample

Sample #	Number of aisles	Enumeration (optimal)		SGA		AGA	
		CPU time	Travel time	CPU time	Travel time	CPU time	Travel time
1	4	345	5.17	87	5.83	104	7.50
11	4	286	5.17	75	5.17	72	7.17
21	3	224	10.00	71	10.00	75	10.00
31	3	138	7.50	59	7.50	65	7.50
41	2	71	5.00	95	5.00	101	5.00
51	2	86	5.83	96	5.83	107	7.50
61	4	253	7.50	68	7.50	80	7.50
71	4	292	7.67	98	8.00	100	7.67
81	3	100	7.50	85	7.50	88	7.50
91	3	160	7.67	117	7.67	110	7.67
101	2	51	7.50	104	7.50	79	10.00
111	2	46	5.83	113	5.83	100	5.83
121	4	153	5.00	109	5.00	128	7.50
131	4	190	7.17	100	7.17	96	12.50
141	3	62	12.67	91	12.67	84	12.67
151	3	79	9.67	98	11.00	85	9.67
161	2	47	15.33	96	15.33	98	15.33
171	2	40	10.00	142	10.00	125	10.00

Total number of items varies depending on the order, as we are only interested in CPU time in this project, and having low number of items will increase quantity of each regardless of warehouse density, and, in turn, dramatically increase the total computation time of enumeration algorithm because of the need to use hard disk as a buffer memory and long I/O times associated with it.

Simulation results

Selected numerical performance measure results are presented in Table 1. It shows every tenth sample's analysis result out of total 180 randomly generated, and split in 20 samples for each configuration. CPU time is shown in milliseconds, while travel times are shown in abstract units of time it takes to complete an order.

For space limitation, we show only the results for the case in which customer orders include only two and four items, as shown in Table 1 and Table 2, respectively.

In Table 1, where the order includes only two items, 74.44% and 78.33% of solutions found by SGA and AGA are optimal, with AGA having much higher average difference at 29.68% (compared to 9.96%) for cases when it provided suboptimal solutions. However, in the next case, where the order includes four items, 58.89% and 91.67% of solutions found by SGA and AGA are optimal, with AGA having much higher average difference at 34.71% (compared to 10.09%) for cases when it provided suboptimal solutions.

Conclusions

We addressed an order picking problem in a multi aisle AS/RS, where items of some type can be found in several different locations. In addition to simple genetic algorithm, an adaptive genetic algorithm was developed. Enumeration algorithm was used in the simulation for comparison purposes and finding an optimal solution in order to assess the performance of already existing with the proposed algorithms. As CPU time for enumeration has an exponential increase speed, it takes very long time to compute a solution with complicated case, eventually rendering it completely unusable at the current

Table 2 – performance of the algorithms on the orders with four items; every tenth sample

Sample #	Number of aisles	Enumeration (optimal)		SGA		AGA	
		CPU time	Travel time	CPU time	Travel time	CPU time	Travel time
1	4	17503	42.50	450	42.50	342	42.50
11	4	21121	39.33	399	57.33	366	39.33
21	3	3363	21.50	259	63.50	253	21.50
31	3	3848	37.00	337	37.00	264	37.00
41	2	1137	60.67	232	60.67	199	60.67
51	2	397	60.50	251	63.00	201	60.50
61	4	5519	57.00	290	63.17	231	57.00
71	4	10369	39.33	391	40.00	296	39.33
81	3	3572	27.00	324	27.00	246	27.00
91	3	1437	40.00	325	40.00	255	40.00
101	2	182	56.00	240	56.00	184	56.00
111	2	453	59.83	232	59.83	184	59.83
121	4	1754	67.50	332	67.50	255	67.50
131	4	3275	42.83	282	42.83	241	42.83
141	3	1126	66.67	142	71.33	165	68.50
151	3	725	63.33	138	85.83	184	63.33
161	2	87	49.17	218	49.17	170	49.17
171	2	728	35.33	218	35.33	177	35.33

level of technological advance and in theory in general, because of the sheer amount of computing power needed.

As a result, while the genetic algorithm is based on effectively guessing the right answer, nevertheless, it proves surprisingly effective, frequently providing solutions with equal or near equal fitness values to the solutions given by enumeration algorithm. At the same time, genetic algorithm requires much less computation time than going through every possible solution with enumeration algorithm and the answer could be calculated in a much shorter and actually reasonable time, especially for more complicated cases.

References

Amato, F., Basile, F., Carbone, C. and Chiacchio, P. (2005), “An approach to control automated warehouse systems”, *Control Engineering Practice*, Vol. 13, pp. 1223-1241.

Holland, J.H. (1992), “*Adaptation in Natural and Artificial Systems: An Introductory Analysis with Application to Biology, Control and Artificial Intelligence*”, MIT Press.

Hsu, C.M., Chen, K.Y. and Chen, M.C. (2005), “Batching orders in warehouses by minimizing travel distance with genetic algorithms”, *Computers in Industry*, Vol. 56, pp. 169-178.

Khojasteh-Ghamari, Y. (2012), “Warehouse management: productivity improvement in automated storage and retrieval systems”, in Manzini, R. (Ed.), *Warehousing in the global supply chain*, London: Springer.

Khojasteh-Ghamari, Y. and Son, J.D. (2008), “Order picking problem in a multi-aisle automated warehouse served by a single storage/retrieval machine”, *International Journal of Information and Management Sciences*, Vol. 19, No, 4, pp. 651-665.

Krishnaiah C.O.V. and Sarveswar R.M. (2003), “Genetic algorithms for studies on AS/RS integrated with machines”, *International Journal of Advanced Manufacturing Technology*, Vo. 22, pp. 932-940.

Wang, J.Y. and Yih, Y. (1997), “Using neural networks to select a control strategy for automated storage and retrieval systems (AS/RS)”, *International Journal of Computer Integrated Manufacturing*, Vol. 10, No. 6, pp. 487-495.

Xudong, S. and Yunlong, X. (2013), “An improved adaptive genetic algorithm”, International Conference on Education Technology and Management Science (ICETMS), pp. 816-819.