2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Optimization of the Picking Sequence of an Automated Storage and Retrieval System (AS/RS)

Rolf Dornberger, Thomas Hanne, Remo Ryter and Michael Stauffer

Abstract— **In this paper we consider the problem of an optimal picking order sequence in a multi-aisle warehouse that is operated by a single automatic storage and retrieval system (AS/RS). The problem is solved by using a genetic algorithm (GA) similar to the one in the earlier research [3]. The problem and the solution approach are implemented in the *OpenOpal* software which provides a suitable test bed for simulation and optimization (see http://www.openopal.org/). As a result it becomes evident that the genetic algorithm can be improved by changing the selection method and introducing an elitism mechanism.**

I. Introduction

This paper is based on results of Khojasteh-Ghamari and Son [3], in which a solution for an optimal picking order sequence for a multi-aisle warehouse, controlled by a single automated storage and retrieval machine (AS/RS) is described. This routing problem is a special case of the known Travelling Salesman problem, which is proven to be NP-hard [8]. Khojasteh-Ghamari and Son [3] developed a genetic algorithm (GA) that can approximately solve this known NP-hard problem. This paper implements and enhances the GA and the used problem representations using the optimization and simulation software *OpenOpal* (see http://www.openopal.org/) and [2]. Of course, other solution approaches such as, for instance, differential evolution or various metaheuristics might be suitable for the considered problem as well.

The optimal picking sequence not only depends on the used algorithm but also on the distribution of the items to be picked within the warehouse. This is exactly where the work of Khojasteh-Ghamari and Son [3] lacks of some further explanation as they only mentioned that an item can occur several times within a warehouse. However, in most warehouses the stored items occur in a certain distribution which leads to the fact that not all the items are stored in the same amount. Another obstacle in [3] is the too strong abstraction of reality, resulting in many simplifications in their proposed GA. This paper analyses where exactly the GA needs to be improved in order to better represent real world conditions. Moreover, it needs to be investigated whether the proposed GA of Khojasteh-Ghamari and Son [3] can be improved or not regarding performance, especially the rate of convergence, to find a nearly ideal picking order.

Rolf Dornberger is with the University of Applied Sciences and Arts Northwestern Switzerland, School of Business, Institute for Information Systems, Peter Merian-Str. 86, CH-4002 Basel, Switzerland (phone: +41-61-279-1790; fax: +41-61-279-1798; e-mail: rolf.dornberger@fhnw.ch). Thomas Hanne, Remo Ryter, and Michael Stauffer are with the University of Applied Sciences and Arts Northwestern Switzerland, School of Business, Institute for Information Systems, Riggenbach-str. 16, CH-4600 Olten, Switzerland (e-mail: {thomas.hanne, remo.ryter, michael.stauffer}@fhnw.ch).

The focus of this paper lies on the fine tuning of the different parameters of a GA such as the selection method of the next generation and the probability of crossover and mutation. There is a high probability to optimize the GA by adapting the different parameters as [3] only focused on specific settings of parameter (roulette wheel selection method, crossover and mutation probability 100 percent) and they did not provide a sufficient analysis or discussion of respective parameter settings.

The research results are divided into three constructive parts the *Analysis*, the *Implementation* and the *Evaluation of the Implementation*. The *Analysis* reveals the missing information in the work in [3] that is necessary in order to successfully implement the GA in *OpenOpal*. Such knowledge gaps are the missing warehouse item distribution as mentioned before. A solution can then be modelled and implemented in *OpenOpal* based on the results of the *Analysis*. In the end the implementation is analysed and optimized in order to improve the performance of the GA.

II. Literature Review

The paper by Khojasteh-Ghamari and Son [3] is the basis of our study as it was the starting point for the general idea of the algorithm. The implementation first tried to follow their approach as closely as possible. However, there were some implementation-relevant parts not described. In those cases, assumptions and approximations were made to cope with this problem. The biggest gap of information was the missing description of the initial warehouse item distribution. This has a high impact on the overall performance of the order picking and should therefore be specified.

A small and simple overview of different types of selection for genetic algorithms. The webpage of NeuroDimension [4] includes a short description of the selection type which explains the basic idea of the method.

The tutorial of Obitko [5] describes the different parts of genetic algorithms in general. For our paper the part of the steady-state selection was particularly interesting since it is not included in [4].

To close the gap of information related to the initial warehouse item distribution another paper had to be taken as source of knowledge. In [1] the storage locations of products inside a warehouse are analysed. During their experiments, different combinations of storage assignment policies were tested and evaluated. They state that the random storage assignment is comparable in performance to an ABC class-based one. This confirms that the assumption of a normal distribution combined with a random storage assignment is a valid approximation.

Further papers which deal with optimization problems in AS/RS control under different assumptions are [6], [7], [8], and [9]. In [6] the order picking sequence of an AS/RS is formulated in a TSP like manner and optimized by a recursive algorithm. Based on a simulation model, the design of an AS/RS operated warehouse is optimized by performing experiments in [7]. In [8] a more complex study is undertaken which combines simulation and optimization approaches for order picking problems. In [9] the assignments of storage locations are optimized in an AS/RS operated warehouse which is modeled using the simulation software FLEXSIM.

III. Analysis

There are many research papers available regarding the picking order problem in a warehouse, but [3] has revealed a research need regarding the situation where one particular item can be stored in *several storage locations* within the same warehouse. They tried to evaluate whether their approach of using a GA to optimize the picking order of an AS/RS machine is suitable or not. Subsequently, a brief listing is provided which topics are covered in this paper:

- Description of the warehouse design (I/O stations, AS/RS machine, racks, aisles, etc.)
- Explanation of the GA (representation of chromosome, initialization of population, crossover, mutation, evaluation and selection)
- Test results with the settings and evaluation (models, results and analysis)

[3] is precise in the description of the warehouse design and the GA. However, in the section on simulation some information necesssary to reproduce their research is missing. In particular the initial storage of the items in the warehouse is completely missing in the simulation model description. Only the settings of the warehouse design are mentioned (warehouse capacity, warehouse density, and shape factor), these parameters are described in more detail in the section IV-A. Therefore, further parameters are needed in order to develop a GA which better represents real world conditions, such as:

- The number of different items in the warehouse in total
- The number of positions in the warehouse for each item
- The distribution of items in the warehouse
- The population size for the GA
- The number of generations of the GA

The last detail to be clarified is the starting position of the AS/RS machine to pickup the items of a new order. In [3] it is mentioned that the starting position is depending on the storage location of the last picked item of the previous order. This means that the starting position is the I/O station of the aisle in which the previous order was finished. Since there is no information about the previous order, the I/O station in the first aisle is picked as starting point.

IV. Implementation

To align the *Warehouse Problem* as well as the *Warehouse GA* as described in [3] with the architectural principle of *OpenOpal*, both parts are separated from each other as shown in Figure 1. Nevertheless, there is a need for a common part in the form of the so called *Problem Properties*. Subsequently, each part of the model is explained in a programming language neutral way.

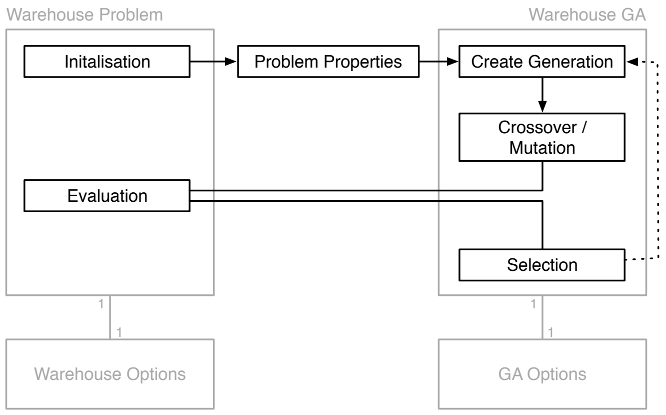

Fig. 1. Implementation model of the warehouse problem and GA in *OpenOpal*

A. Warehouse Options

Even though the *Warehouse Options* are a separate class due to the design pattern of *OpenOpal*, they are tightly coupled to the *Warehouse Problem*. This class is mainly necessary to offer all the parameters to initialize a warehouse in the *Warehouse Problem* class. An overview of the needed parameters is provided subsequently:

- Warehouse Capacity (Integer)
- Number of Aisles (Integer)
- Warehouse Density (Float)
- Number of Different Items (Integer)
- Shape Factor (Float)

The *Warehouse Capacity* is proportional to the *Number of Aisles* since the *Number of Storage Locations per Rack* is firmly defined as 780 in [3]. For this reason, the *Warehouse Capacity* directly depends on the *Number of Aisles* and can only be a multiple of 1560 (two racks per aisle). In the simulation described in [3] a warehouse with one, two, three or four aisles is evaluated. The reason for this limitation of aisles to four or fewer lies in the fact that a high number of aisles decreases the practical efficiency of a warehouse system that is served by a single AS/RS machine only [3].

The *Warehouse Density* describes the percentage of used *Number of Storage Locations per Rack* compared to the overall *Warehouse Capacity*. A setup of 60%, 75% or 90% can be chosen [3].

Due to the parameter *Number of Different Items* the number of different items that are stored in a warehouse

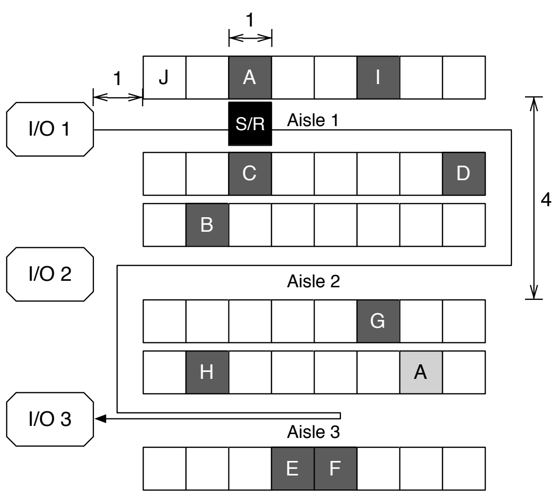

Fig. 2. Warehouse scheme (top-view)

can be defined. This parameter has a strong influence on the picking duration since a high number of *Number of Different Items* leads to fewer picking possibilities in a warehouse and therefore the chance to pick all needed items in a single aisle becomes smaller. This parameter can have an arbitrary positive integer value that is smaller or equal to the effective *Warehouse Capacity*.

The *Shape Factor* describes the ratio of height and length of the rack (see Figure 3), a rack that is for example 6m high and 10m long has a shape factor of 0.6. In the GA a shape factor of 0.6, 0.73 or 1 can be chosen [3].

B. Warehouse Problem

The problem of an optimal picking order sequence of a multi-aisle warehouse by a single AS/RS machine (*Warehouse Problem*) is already described in [3]. However, this paper has a strong focus on the description of the *Warehouse GA* only (see sub subsection IV-E), without explaining the initialization and structure of the warehouse in general.

For clarification and simplification the following constraints need to be considered for the *Initialization* as well as the *Evaluation* of the *Warehouse Problem* [3]:

- One or more aisles can be chosen
- An aisle has on each side a multilevel storage rack
- Each item can be stored in several locations
- A picking sequence consists of one or more different items
- One single automated storage/retrieval system (AS/RS)
- Starting position of the AS/RS is the I/O station of the aisle where the last item of the previous order has been picked
- The AS/RS can move simultaneously in horizontal and vertical direction
- The AS/RS has a constant velocity

In addition to the constraints proposed in [3] the following constraints are defined:

- There is an input/output (I/O) station at each aisle
- The AS/RS can switch aisles on each side of the aisle
- The AS/RS has an infinite storing capacity

1) Initialization: The algorithm to initialize a warehouse consists of three steps:

Step 1 - Initialization: As already mentioned in the warehouse constraints, each aisle consists of two racks per aisle, therefore the total *Number of Racks* can easily be calculated by multiplying it by the *Number of Aisles*. Even though the *Number of Storage Locations per Rack* is defined as 780 items per rack in [3], it is better to choose a *Number of Rows and Columns per Rack* where the square root leads to an integral number. By doing so it can be guaranteed that there are now half rows or columns. For this reason the *Number of Storage Locations per Rack* is defined as 784 as well as the *Number of Rows and Columns per Rack* is defined as 28. In the next step the effective *Warehouse Capacity*, known as *Total Items To Place*, can be calculated according to the chosen *Warehouse Density*.

Step 2 - Randomly calculate number of items to place: An entry is generated for each item in a list (ItemsToPlaceList) based on the *Number of Different Items*. As a next step, an item is randomly chosen out of this list and added to a map (WarehouseNrOfStorageMap) that determines the total *Number of Storage Locations per Item* in the warehouse. This random position within the linked map guarantees that the position itself has no influence on the number of storage locations that is generated subsequently.

In the next step, a random number with a standard deviation of σ is generated. This random number has a range of -1 to +1 (full width at half maximum) and a mean of 0. This means that 68.27% of all generated random values are in-between this range of -1 to +1 and that the rest of 31.73% is outside this range. As we divide all warehouse items within the range from -1 to +1 it is necessary to throw away every random number that is outside this range.

This distribution of items is an approximation of the well known ABC distribution. There are different publications regarding the optimization of the warehouse inventory and positioning of items. In [1] the ABC distribution is extended with specific factors related to the order details. For our purpose it is sufficient to just approximate a regular ABC distribution since the order is highly artificial and so are the warehouse items.

The increase of storage locations per item is then done according to a threshold that is increased step by step from -1 to +1. It will be checked whether the random number is below this threshold and if so, the corresponding item counter in the map will be increased by 1, this means that the selected item receives an additional storage location within the warehouse.

Through this normal distribution it can be guaranteed that not all items are stored in the same number of positions, e.g. item A is stored at 10 different warehouse locations whereas item B is only stored at 3 different warehouse locations. If this would not be the case, the chance that an AS/RS needs to change its aisle would be reduced significantly as there is a high chance that all items are stored at least once within the same aisle (under the settings discussed below). Finally, this also leads to a more realistic warehouse in which not all items are available in the same number, e.g. according to an ABC distribution.

Step 3 - Randomly calculate item positions in warehouse: In the last step, an item position is generated for each of the possible storage locations according to the overall *Warehouse Capacity*. All possible positions are stored in a list (WarehousePositionList) where from for each of the *Number of Different Items* the number of storage locations is retrieved from the map (WarehouseNrOfStorageMap) generated in the previous step. According to the retrieved number of storage locations random positions are removed from the position lists (WarehousePositionList) until for each item the corresponding warehouse position is placed.

2) Evaluation: Based on this initialized warehouse the *Warehouse GA* can now generate and test different chromosomes to probably find a good solution after several generations.

During the evaluation, a picking order sequence represented in the form of a chromosome is proposed by the GA to evaluate how well the solution performs according to the chosen fitness function. In the case of the *Warehouse Problem*, the fitness value is simply the minimal distance to pick up all required items in the given sequence.

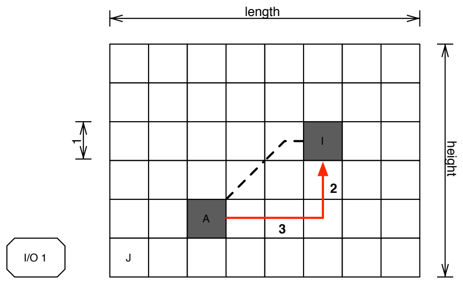

Fig. 3. Storage rack scheme

To accurately calculate the fitness of a solution, further constraints are necessary and listed below:

- The distance is measured in integer values without any specific unit
- The distance between two adjacent items is one (e.g. Figure 2 the distance between item E and item F)
- The distance between two obverse items is zero (e.g. Figure 2 the distance between item C and item A)
- The distance between the I/O station and the first column in the storage rack is one (e.g. Figure 2 the distance between I/O 1 and item J)
- The distance between two aisles is four (e.g. Figure 2 the distance between I/O 1 and I/O 2)
- The I/O station is positioned on the ground level

According to the constraints mentioned before the total distance of a picking order sequence can be calculated as in algorithm 1 below:

```
Data: Picking order sequence
Result: Total distance of picking order sequence
totalDistance t = 0;
t += Distance from last I/O station to first item;
for $i \leftarrow 1$ to Number of Different Items $- 1$ do
    t += Distance from Item i to Item i+1;
end
t += Distance from item to next I/O station;
```

Algorithm 1: Calculation of total distance of a picking order

Additionally, the distance between different items or between an item and an I/O station needs to be calculated as mentioned in Algorithm 2 below. Given the fact that the AS/RS machine can move simultaneously at constant velocity, the distance between two travel points is the maximum of the horizontal or the vertical travel distance. In Figure 3, the distance between item A and item I is therefore three as the AS/RS machine has already travelled the vertical distance before the horizontal distance was reached [3]. The dashed line illustrates the actual path of the AS/RS machine.

```
Data: Position of Item1 and Item2
Result: Distance between Item1 and Item2
if Item1 and Item 2 are in stored in the same aisle then
    Calculate Vertical Distance between Items;
    Calculate Horizontal Distance between Items;
    return min. of Vertical and Horizontal Distance;
end
else
    Calculate Left Move;
    Calculate Vertical Distance from left side to Item2;
    Calculate Horizontal Distance from left side to
    Items2;
    Left Distance = Left Move + min. of Vertical and
    Horizontal Distance;
    Calculate Right Move;
    Calculate Vertical Distance from right side to Item2;
    Calculate Horizontal Distance from right side to
    Items2;
    Right Distance = Right Move + min. of Vertical and
    Horizontal Distance;
    return min. of Left and Right Distance;
end
```

Algorithm 2: Calculation of distance between two items

Algorithm 2 for the calculation of the distance between two items simply checks whether two items are in the same aisle or not, If this is the case, it calculates the *Vertical Distance* and *Horizontal Distance* between the two items and returns the smaller value of both. If the two items are not in the same aisle there are theoretically two possible ways which the AS/RS machine can take, one way goes left around to switch the aisle and the other way goes right around to switch the aisle. For this reason the *Left Move* and the *Right Move* from the current aisle of item 1 to item 2 needs to be calculated. In the case of the *Left Move* this is done by first calculating the vertical distance from item 1 to left outer border of this aisle, secondly the change of the aisle is calculated by multiplying the number of aisles to change by the predefined constant of 4 (distance between two aisles). Finally the distance from the outer border to item 2 is exactly as if the item were in the same aisle. The same needs to be done for the right move and finally the smaller of both distances needs to be returned. The distance from the I/O station to an item is done in the same way as if two items were located in the same aisle.

In Figure 2 an exemplary picking order sequence is illustrated. The order is sorted according to the pickup position and consists of the following items A, C, I, D, G, B H, E and F. The total distance of this picking order sequence is calculated as follows:

	From	To	Distance
+	I/O 1	A	3
+	A	C	0
+	C	I	3
+	I	D	2
+	D	G	9
+	G	B	4
+	B	H	9
+	H	E	3
+	E	F	1
+	F	I/O3	5
=			35

TABLE I
CALCULATION OF PICKING ORDER SEQUENCE OF FIGURE 2

All items are on the same height so there are only horizontal movements in the calculation above to simplify the distance calculation. The total distance of this picking order sequence of 35 will then be returned to the *Warehouse GA*, where from in a next iteration new chromosomes are provided to evaluate, this process continues until a certain termination criterion is fulfilled, e.g. after a certain number of iterations.

C. Problem Properties

It is necessary to store the different positions at which an item is stored all over the warehouse after the initialization. The *Warehouse GA*, respectively its crossover and mutation, depends on the information of the different positions at which a certain item is stored in the warehouse.

D. GA Options

To define the behaviour of the GA there are some parameters required:

- Number of generations (Integer)
- Number of individuals per generation (Integer)
- Selection method (Integer)
- Mutation method (Integer)
- Crossover method (Integer)
- Probability of crossover (Float)
- Probability of mutation (Float)
- Percent of elitist (Float)

These parameters exceed the minimum requirements to reproduce the solution of [3]. Only the selection, crossover and mutation methods are specified. The probabilities if a crossover or a mutation happens are not discussed in [3] and it is assumed that the probability is equal to 1.0. The idea of elitism is also not considered in [3]. The idea is that at least a small percentage of the population forms an elite (best solutions found in one generation). Such solutions are directly transferred to the next generation without using the selection method, i.e. the best solutions are preserved for the next generation. Since this is not specified, it is assumed to be equal to 0 percent. However, these options can clearly improve the convergence rate (performance), especially when the population is large.

E. Warehouse GA

The goal of the algorithm is now to minimize the travel time of the *AS/RS* machine to complete the retrieval process of the orders [3]. This means to find the optimal sequence of item positions, which the *AS/RS* machine has to visit before the order is completely collected and can be delivered to the nearest I/O station. This is not trivial since the items can be stored at several storage locations and the path of the *AS/RS* machine has to be optimized inside the current aisle and rack as well as the number of aisle changes which should be as small as possible.

1) Create Generation: To start with the GA it is required to randomly create a starting population. This has to be done in accordance with the warehouse since there are restrictions regarding the possible items and their storage locations. The necessary information from the warehouse is shared via the *Problem Properties* with the GA. The goal of the developed approach is to avoid infeasible solutions, like for example inexistent items or positions.

2) Mutation: The mutation faces the same problem since it should change the position of an item. Therefore, it is necessary to know the possible positions for the particular item in order to avoid infeasible individuals [3]. The procedure is fairly simple, one just has to pick another position for this item. But again, knowledge about the actual warehouse is required. Figure 4 shows an example encoding of a mutated chromosome.

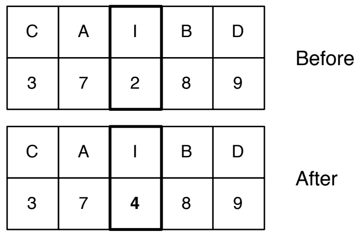

Fig. 4. Example of a mutation, assuming item I to be available at locations 2 and 4

3) Crossover: During the crossover, two possible pickup sequences are combined. Since the order of the items can be very different, it is highly probable that the combined sequence will contain duplicate items or miss some items which are required to be picked up. To avoid this, the partially matched crossover method is used in [3]. We have adapted this approach for our study. An example of using this genetic operator is shown in Figure 5.

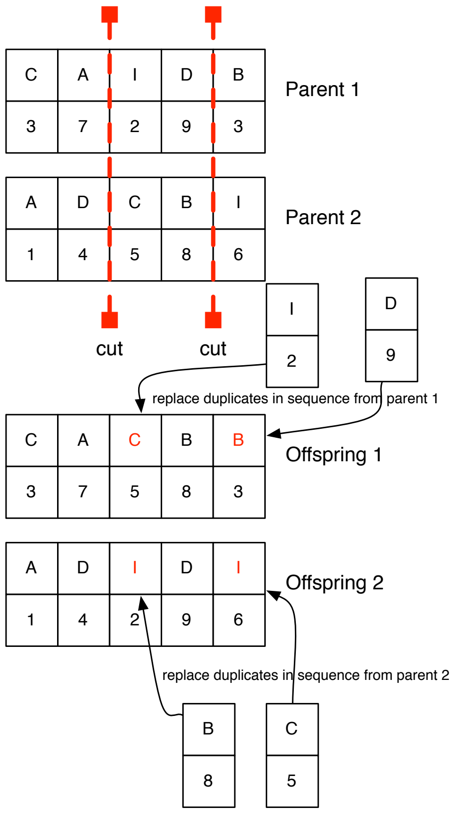

Fig. 5. Example of the partially matched crossover

4) Selection: The only criterion of the fitness of an individual is the total distance the *AS/RS* machine has to travel to pick up all demanded items. The selection of the fittest individuals relies solely on this evaluation. [3] defines that the *Roulette Wheel* is used as a selection method. This method spins a virtual roulette wheel and stops at a random position. The wheel itself is divided into sections, each section is assigned to an individual of the population. The fitter an individual, the bigger its section on the wheel and therefore the more likely its selection for the next generation. Figure 6 shows that individual 3 is the fittest and has therefore the highest probability to be picked for the next generation.

V. Evaluation of the Implementation

To prove whether the implemented *Warehouse GA* has provided a nearly ideal solution or not, it would be necessary to enumerate all possible picking order sequences and then select the solution with the best fitness value to compare it

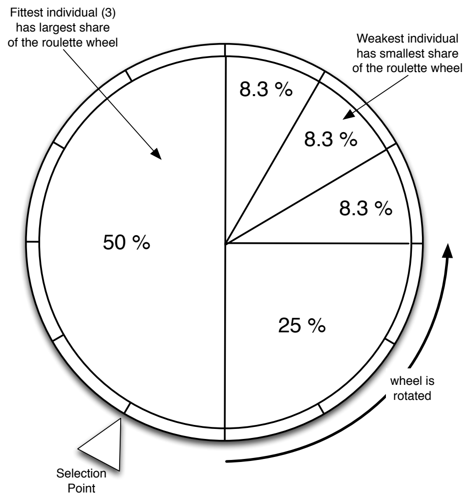
Fittest individual (3) has largest share of the roulette wheel
Weakest individual has smallest share of the roulette wheel
8.3 %
8.3 %
8.3 %
50 %
25 %
wheel is rotated
Selection Point

Fig. 6. Roulette wheel selection method

with the proposed solution. This is exactly what is described in [3], even though this would be ideal for comparison, it is very expensive regarding computational time as the number of possible combinations increases exponentially. Due to the considered problem size and time limitations this could not be proved in the same way. The current implementation of the *Warehouse GA* in *OpenOpal* allows settings that result in a much higher number of possible pickup sequences. The number of feasible solutions can be calculated by [3]:

$$n! \prod_{i=1}^{k} \binom{m_i}{n_i} = n! \prod_{i=1}^{k} \left(\frac{m_i!}{n_i! \, (m_i - n_i)!} \right) \tag{1}$$

Where k is the *Number of Different Items*, m_i is the total inventory of the ith item (e.g. *Number of Storage Locations per Item*) in the warehouse and $n = \sum_{i=1}^{k} n_i$. With the maximum settings ($k = 1000$, $m_i = 123$ (average, rounded), $n_i = 1$ and $n = 1000$) the number of possible solutions results to

$$1000! \prod_{i=1}^{1000} \left(\frac{123!}{1! \, (123-1)!} \right) = 1000! \cdot 123^{1000}$$

.

Due to this high number of different possible solutions, it is not possible to find the best solution in a reasonable computation time. Even though it cannot be proved that the newly implemented *Warehouse GA* generates a nearly ideal solution, it is possible to check whether the proposed settings for the *Warehouse GA* [3] are ideal or not.

To compare different parameter settings for the *Warehouse GA* it is necessary to have fixed settings for the *Warehouse Problem*. For this reason, the following default settings [3] are chosen:

- Rack Capacity : 784
- Number of Aisles : 4
- Warehouse Density : 75%
- Number of Different Items : 5

- Shape Factor : 1

For this warehouse capacity and number of different items it is recommended to take a *Number of Generations* of 5000 with a *Number of Individuals per Generation* of 100. However, this is just a rule of thumb and depending on the *Warehouse Problem*, it is recommend to test the *Warehouse Problem* with different settings so that a nearly optimal solution can be reached.

The first valuable observation is that Khojasteh-Ghamari and Son [3] did not implement any kind of elitism in their *Warehouse GA* at all. Even though they have implemented a *Roulette Wheel* that increases the probability that a good solution is selected for the next population, they have elitism neither recommended nor used. However, it became evident that this selection mechanism is not enough to steadily improve the fitness value with the increasing number of generations, the result is an oscillating fitness value in relation to the increasing number of generations. Moreover, apart from avoiding fitness deteriorations, elitism often leads to a faster convergence and is usually an essential assumption in convergence proofs. It is therefore recommended to implement an elitism mechanism in the originally proposed *Warehouse GA* in [3] to minimize the fluctuation of fitness during the population construction.

The second improvement that became evident during the evaluation of the *Warehouse GA* is that the proposed selection method *Roulette Wheel* [3] seems not to lead to optimal fitness value. Any other selection method that is implemented in OpenOpal like *Best 20%* [4], *Steady State* [5] and *Tournament* [4] not only leads to better solutions, they also have a higher convergence speed. Finally, the best selection method seems to be *Tournament* which found the best solution with a distance of 13 after 700 generations as shown in Figure 8. In Figure 7 it is shown that the *Roulette Wheel* found the best solution with a distance of 23 after 1800 generations! It is therefore recommended to implement the *Tournament* selection method in the originally proposed *Warehouse GA* in [3].

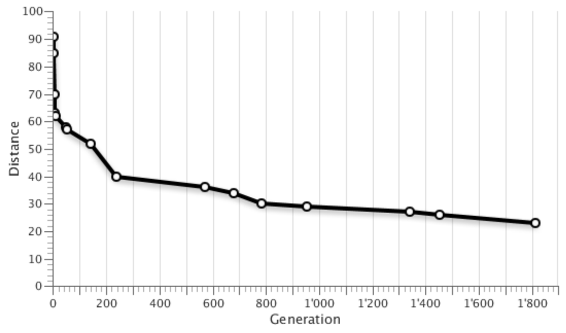

Fig. 7. Convergence of the fitness function of the roulette wheel selection method

The final observation is that a higher *Probability of Crossover* and *Probability of Mutation* only leads to higher computational time but does not result in better solutions. It is therefore recommended to have only a small *Probability of*

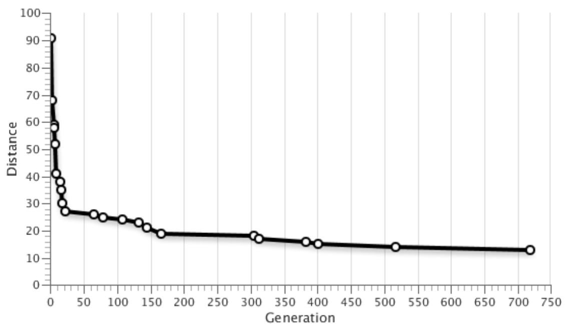

Fig. 8. Convergence of the fitness function of the tournament selection method

Crossover and *Probability of Mutation* as this not only leads to better solutions, but also to an increase of the convergence speed. Of course, the probabilities have to be greater than zero, otherwise no evolution takes place.

VI. Conclusion

In our study, we considered the problem of optimally controlling an AS/RS for picking orders from a multi-aisle warehouse. By doing so we mainly followed the earlier contribution [3]. In this paper, a number of significant details are missing or described insufficiently. This, in particular, concerns the problem set-up, i.e. the specification of the warehouse including the location of items. We investigated this issue in more details and provided a realistic approach for the problem set-up including an item distribution in accordance with an ABC scheme. Moreover, our experiments with the employed genetic algorithm led to some substantial improvements, e.g. an elitist based selection scheme.

For future investigations there are still several aspects to be studied in more details, e.g. a more detailed investigation into the effects of parameter settings of the genetic alghorithm. Apart from that, the problem scenario could be extended or refined. This could include a more realistic representation of an AS/RS, with a limited capacity for picked items or with more realistic movement assumptions, e.g. non-constant velocities. Moreover, in a typical warehouse usually several AS/RS are used, e.g. one AS/RS per aisle, and often with further restrictions, e.g. a missing possibility to change aisles. Scenarios like that will be subject to future research.

References

[1] F. T. Chan and H. Chan, "Improving the productivity of order picking of a manual-pick and multi-level rack distribution warehouse through the implementation of class-based storage," Expert Systems with Applications, vol. 38, no. 3, pp. 2686–2700, Mar. 2011.

[2] R. Dornberger, T. Hanne, and L. Frey, "The way to an open-source software for automated optimization and learningOpenOpal," In 2010 IEEE Congress on Evolutionary Computation (CEC), (pp. 1-8). IEEE, 2010.

[3] Y. Khojasteh-Ghamari and J.-D. Son, "Order picking problem in a multi-aisle automated warehouse served by

a single storage/retrieval machine," Int. J. Inf. Manag. Sci., vol. 19, no. 4, pp. 651–665, 2008.

[4] NeuroDimension, "Genetic Algorithms," 2002. [Online]. Available: http://www.nd.com/products/genetic/selection.htm.

[5] M. Obitko, "Genetic Algorithms," Tutorial, 1998. [Online]. Available: http://www.obitko.com/tutorials/genetic-algorithms/selection.php.

[6] M. Shuhua and H. Yanzhu, "Research on the order picking optimization problem of the automated warehouse," In Control and Decision Conference, 2009. CCDC'09. IEEE, 2009, pp. 990-993.

[7] S. Takakuwa, "Module modeling and economic optimization for large-scale AS/RS", In Proceedings of the 21st Conference on Winter Simulation. ACM, 1989, pp. 795-801.

[8] M. Yu, Enhancing Warehouse Performance by Efficient Order Picking, Rotterdam: Erasmus University Rotterdam, 2008.

[9] G. Zhou and L Mao. "Design and Simulation of Storage Location Optimization Module in AS/RS Based on FLEXSIM." International Journal of Intelligent Systems and Applications (IJISA), vol. 2, 2010, pp. 33-40.