HAL
open science

Optimizing Transportation Sequence in Warehouse with Genetic Algorithms

Xuan-Thuong Tran, Thanh-Do Tran, Hwan-Seong Kim

To cite this version:

Xuan-Thuong Tran, Thanh-Do Tran, Hwan-Seong Kim. Optimizing Transportation Sequence in Warehouse with Genetic Algorithms. AETA 2013: Recent Advances in Electrical Engineering and Related Sciences, Dec 2013, Ho Chi Minh City, Vietnam. 10.1007/978-3-642-41968-3_40 . hal-01058036

HAL Id: hal-01058036

https://inria.hal.science/hal-01058036v1

Submitted on 25 Aug 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Optimizing Transportation Sequence in Warehouse with Genetic Algorithms

Xuan-Thuong Tran[1], Thanh-Do Tran[2], and Hwan-Seong Kim[3]

[1] Graduate School, Korea Maritime University, South Korea
[2] Inria Lille – Nord Europe and Université Lille 1, France
[3] Department of Logistics, Korea Maritime University, South Korea

Abstract. Optimizing transportation sequence is crucial to reduce material handling costs in warehouse operations and thus in total logistic costs. Transportation sequence is the ordering of storage and retrieval jobs that a material handling device has to perform to finish an order list. In many studies, the optimization of transportation sequence has been simplified as an order-picking problem, and accordingly solved as a classical traveling salesman problem. However, transportation sequence is a double-cycle storage and retrieval problem (DCSRP) in itself, meaning that the combination of storage and retrieval jobs into double cycles has to be considered simultaneously. In this paper, we propose formulating the DCSRP as a permutation problem and applying several genetic algorithms to solve the formulated problem. Extensive computational experiments were performed to demonstrate the capability of the approach. The experimental analysis confirms that our approach could solve the problem efficiently on the one hand, and addresses the question of which genetic operators are best applied to the formulated DCSRP on the other hand.

Keywords: Transportation sequence, genetic algorithms, permutation

1 Introduction

Typical functions of a warehouse include receiving, storage, order picking, and shipping. In [3], the authors provided a comprehensive review of research on warehouse operation, in which various decision support models and solution algorithms for each of the functions were discussed. One of the repeated activities that absorbs significant costs in total warehouse operational costs is material handling. This activity comprises order picking, loading and unloading goods, and transporting to another location for unloading and loading, etc.

These material handling activities are unavoidable since Stock Keeping Units (SKUs) are stored in different locations in a warehouse while orders may come from various customers and/or departments. Transportation costs, both external and in-house transportation, contribute as the highest cost of total logistic costs ($> 40\%$). To reduce these costs, it is required to reduce traveling distance (and also traveling time) of transporters—which are forklift trucks in our work. In warehouses, a single transport order is described as a movement of one storage unit from one location (source) to another location (sink). Such orders are

transferred to the subordinate control of a forklift operating in a warehouse aisle. Thereby the warehouse management system (WMS) can control the sequence (i.e. ordering) of operations to be performed by a forklift in an aisle. In each aisle, loads (e.g. pallets) to be stored and retrieved by the forklift are buffered in the warehouse pre-storage zone [4].

A straightforward method to optimize transportation sequence in an aisle is to combine simultaneous storage and retrieval jobs into multiple cycles. Fig. 1 illustrates the comparison between single- and double-cycle operation modes. There are two transport orders in this illustration. The order 1 is a storage job and the order 2 is a retrieval job. By combining two orders into double cycle operation shown in Fig. 1b, the total traveling distance to complete two orders is 16 length units in comparison with the single-cycle case in Fig. 1a with a total of 26 length units.

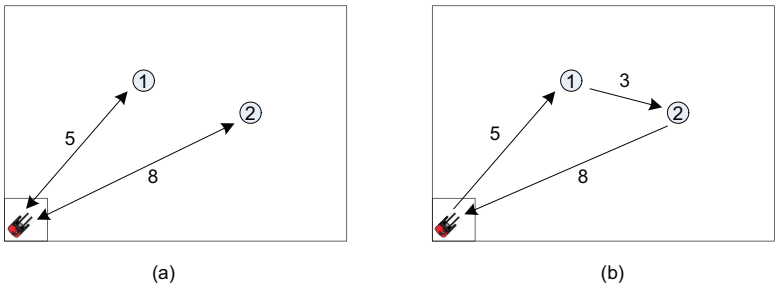

Fig. 1. Single vs double cycle operation mode.

The DCSRP is one of the sequencing problems that occurs in many fields of material flow. Another problem with similar characteristics is the well-known order-picking problem [1, 2]. The objective of these problems is to find an optimal sequence of transportation (or order-picking) that yields the *minimal* total traveling distance of the material handling transporter (or picker). Such kind of sequencing problems can be considered as the traveling salesman problem (TSP), which is among the most popular NP-hard combinatorial problems [6]. In fact, several existing investigations into these sequencing problems have been formulated via the TSP [7, 11].

Specifically in warehouse operations, a large portion of research has been performed to tackle the order-picking problem. For example, in [11], the authors use TSP heuristic algorithms to address the sequencing problems of order pickers in conventional multi-parallel-aisle warehouse systems. Another example is the use of TSP-based k-interchange method for solving the problem of routing order pickers in single-block warehouses [7]. Meanwhile, the issue of double-cycle transportation in warehouses has not been considered rigorously in the operations research literature. Motivated by previous research on using TSP heuristics for solving the order-picking problem, in this paper, we discuss the optimization of transportation sequence in warehouses by solving the double-cycle storage and retrieval problem. To be more specific, we aim to find an optimal combination

of all double cycles that the forklift has to follow to finish an order list with minimal traveling distance.

One of the benefits of transferring the order-picking problem into the TSP is the existence of various solution approaches for this problem. Recent development in metaheuristics, including the hybridization of evolutionary algorithms with heuristics as a local search strategy, have provided promising solution approaches to the TSP. Among evolutionary algorithms, genetic algorithms are the most popular technique and have certain success in solving NP-hard problems, including the TSP [5]. In this paper, we formulate the DCSRP as a permutation problem and employ different genetic algorithms to solve the resulting problem.

The remainder of this paper is organized as follows. In Sect. 2, the double-cycle storage and retrieval problem in a multi-parallel-aisle warehouse system is described and then formulated as permutation problem. Next, in Sect. 3, we present an approach to solving the formulated problem using genetic algorithms (GAs) and review different genetic operators to be used. Then, in order to evaluate the effectiveness of GAs to our problem, extensive computational experiments were implemented and the results are analyzed in Sect. 4. Finally, some conclusions and possible extensions for future research are discussed.

2 Problem Description and Formulation

2.1 Problem Description

A conventional multi-parallel-aisle warehouse system is shown in Fig. 2. This warehouse consists of r racks and l aisles. In this paper, we consider a warehouse containing only full-sized pallets, which are stored on the racks and in both sides of the aisles. In this figure, those pallets that need to be stored are denoted by filled rectangles and those that need to be retrieved are denoted by crossed rectangles. Buffer zone is the location where pallets are released and also where a forklift picks the pallets that need to be stored in warehouse.

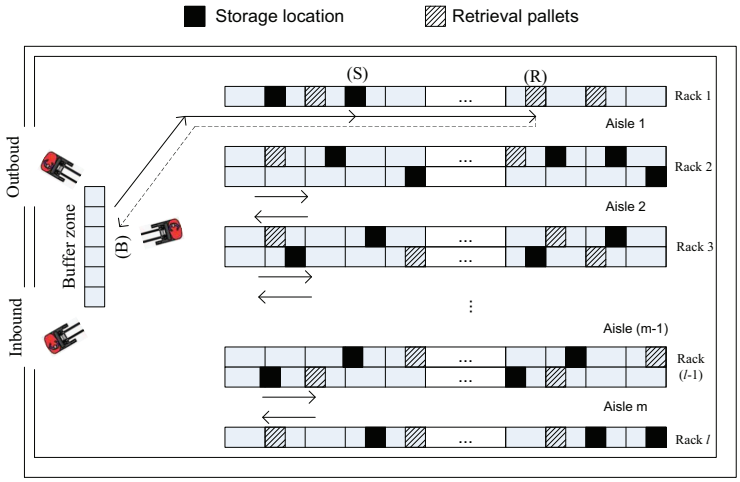

Fig. 2. Multi-parallel-aisle warehouse system.

A double cycle is executed as follows. First, the forklift starts from the buffer zone (B), loads a pallet, moves to the designated storage location (S), and stores the pallet. Then, it approaches the pallet to be retrieved at the retrieval location (R), picks this pallet, travels back to the buffer zone, and unloads the pallet. At this point, the forklift has completed a double cycle and the process is repeated until all transport orders are finished.

Given the DCSRP as described above, the aim of a warehouse manager is to find a sequence of transportation that minimizes the total traveling distance of the forklift. In other words, the goal is to determine a combination of double cycles that results in minimal traveling distance. Such a sequencing problem can be viewed as a permutation problem. Permutation is one of the NP-hard problems for which there does not exist any exact algorithms that can give optimal solutions in polynomial time. For this class of problem, (meta-) heuristics are commonly employed to approximate the optimal solutions. In the following, we show that our formulated problem has the computational complexity of factorial, and thus call for the use of evolutionary algorithms as a solution approach.

Assuming that a symmetrical order volume consists of n storage and n retrieval jobs; therefore, n double cycles have to be executed by the forklift in a consecutive sequence. The first of the n storage jobs may be followed by one of the n retrieval jobs. Then, the second may be followed by one of the $(n-1)$ remaining retrieval jobs. Similarly for the rest of the n storage jobs, there will have a total of $n!$ possible sequences of double cycles that can be chosen. Since any of the n storage jobs can be chosen first in the sequence, the number of all possible double-cycle sequences theoretically amounts to $(n!)^2$.

However, the ordering of double cycles normally has no effect on the overall distance [4]. This means that the two double-cycle sequences 1–4–7 and 7–4–1, for example, are considered to be the same. Thanks to this property, the number of possible sequences of a symmetrical order volume with n storage and n retrieval jobs is reduced to $n!$.

2.2 Problem Formulation

To facilitate the formulation, some assumptions and notations are needed.

Assumptions:

- First of all, the number of storage jobs and retrieval jobs in each aisle are equal and this symmetric order volume is known beforehand.
- The forklift starts and ends a trip at the buffer zone; so, there is no need to add the traveling cost from the ending point of a trip to the starting point of the next trip.
- The forklift can identify the pallets that need to be picked up or retrieved.

Notations:

S: the set of storage jobs, with $|S| = n$ being the no. of storage jobs in S.

R: the set of retrieval jobs, with $|R| = n$ being the no. of retrieval jobs in R.

B: buffer area where stored pallets and retrieved pallets are located.

$d(B \to S_i)$: distance from the buffer area to storage position S_i, $S_i \in S$.

$d(S_i \to R_j)$: distance from storage position S_i to retrieval position R_j.

$d(R_j \to B)$: distance from retrieval position R_j to the buffer position.

P_k: the k-th permutation of the set R, indicating an ordering (i.e. sequence) of retrieval jobs, where $k = 1, 2, \ldots, n!$. For example, with $R = \{R_1, R_2, R_3, R_4\}$, a permutation of R could be $\{R_2, R_1, R_3, R_4\}$. P_k is an ordered set.

P_Σ: the set of all possible permutations, i.e. $P_\Sigma = \{P_k\}_{k=1}^{n!}$

$C_j(P_k)$: the cost of double cycle j, where $j = 1, 2, \ldots, n$. Note that there are exactly n double cycles in each sequence.

$TC(P_k)$: total cost to complete n double cycles associated with P_k.

With the above notations, let us consider a DCSRP with symmetrical order volume consisting of n storage jobs and n retrieval jobs. The cost of a double cycle j in the sequence corresponding to the permutation P_k, where $k = 1, 2, \ldots, n!$, is calculated by Eq. 1:

$$C_j(P_k) \;=\; d(B \to S_i) \;+\; d(S_i \to R_j) \;+\; d(R_j \to B), \tag{1}$$

where $S_i \in S$ with $i = 1, 2, ..., n$, and $R_j \in P_k$ with $j = 1, 2, ..., n$. Finally, the total cost to complete n double cycles in permutation P_k is computed by Eq. 2:

$$TC(P_k) = \sum_{j=1}^{n} C_j(P_k). \tag{2}$$

The objective of solving the DCSRP is to find a permutation $P^* \in P_\Sigma$ of the set R [of retrieval jobs] that minimizes the total traveling distance of the forklift:

$$P^* \;=\; \arg\left[\min_{P_k \in P_\Sigma} \left(TC\left(P_k\right)\right)\right]. \tag{3}$$

Eventually, the optimal solution of the DCSRP is a sequence of n double cycles. Each double cycle i of the sequence consists of a pair of the storage job S_i and the retrieval job R_j, where R_j is the i-th element of P^*. Let us consider an example with 4 storage jobs $S = \{S_1, S_2, S_3, S_4\}$ and 4 retrieval jobs $R = \{R_3, R_2, R_1, R_4\}$. Assuming that the optimal solution of this DCSRP is $P^* = \{R_3, R_2, R_1, R_4\}$, then the optimal sequence of double cycles is:

$$(B \to S_1 \to R_3 \longrightarrow B \to S_2 \to R_2 \longrightarrow B \to S_3 \to R_1 \longrightarrow B \to S_4 \to R_4 \to B)$$

It is worth noting that the permutation of the set S of n storage jobs does not affect the solution of the problem. Note also that the DCSRP is subject to a number of constraints. The first constraint is that each storage job position or retrieval job position can only be visited once. Second, in each double cycle, the forklift must firstly visit a storage location to unload the pallet and then move to a retrieval position to load the retrieval pallet; and this order must be respected. Third, after loading a retrieval pallet, the forklift must always move to the buffer area where the pallet is to be pre-stored. In the next section, we will discuss the use of genetic algorithms to solve the formulated DCSRP.

3 Genetic Algorithms for the DCSRP

Genetic algorithms (GAs) [8] are perhaps among the most widely used evolutionary optimization techniques. GAs are adaptive randomized searchers that simulate the genetic inheritance and the Darwinian principle of striving for survival in nature. One might find GAs simple to implement, fun to use, and versatile to solve a wide range of problems. GAs are often useful to problems for which there is no algorithm available or the computation to reach the exact optimum is unaffordable. The pseudocode of typical GAs is given in Alg. 1.

Algorithm 1 Pseudocode of Genetic Algorithms

```
INITIALIZATION: randomly generate a population of N individuals
EVALUATION: evaluate the initial population
while (not STOPPING_CRITERIA) do
    MATING SELECTION: select parents to reproduce offsprings
    CROSSOVER: apply crossover operator to the mating pool to generate offsprings
    MUTATION: mutate the offsprings by [genetic] mutation operator
    EVALUATION: evaluate the offspring population
    SURVIVOR SELECTION: select individuals for the next generation
end while
```

First of all, we have to define how a candidate solution to the problem is represented in GAs. For the DCSRP formulated in Sect. 2.2, it is quite straightforward to use integer numbers from 1 to n (with n being the number of retrieval jobs) to directly encode a retrieval job's number. By this way, a candidate solution to the DCSRP is just a series of unique integers; and the ordering of the numbers in this series determines which retrieval job to be handled together with which storage job in a double cycle. More precisely, the value (from 1 to n) of an integer in the series indicates the retrieval job's number, while the position (also from 1 to n) of this integer in the series indicates the companion storage job's number. In other words, a permutation of the series generates a candidate solution to the problem. With this encoding scheme, it is trivial to create an initial population of N individuals for GAs, just by performing N permutations.

Given a candidate solution, it is essential to compute the objective value (i.e. the cost, or the total traveling distance in this case) associated with it. Using the above-mentioned encoding strategy and the problem formulation given in Sect. 2.2, the cost of a solution is computed by summing up the costs induced by all component double cycles (see Eq. 2). The cost of each double cycle, in its turn, is—as denoted in Eq. 1—the summation of three components: the distance from the buffer to the storage location, the distance between the storage and retrieval jobs, and the distance from the retrieval job to the buffer. By these calculations, every *feasible* solution (i.e. a series of *unique* integers from 1 to n) can be evaluated for its objective value.

In addition to the representation and evaluation function, the determination of proper genetic operators, including crossover and mutation, are of paramount importance to a GA. These operators are strongly dependent on the solution representation. For the permutation encoding described above, there exist a

number of crossover and mutation operators that can be applied. In this work, *five* crossover operators: (1) order crossover (OX), (2) partially mapped crossover (PMX), (3) cycle crossover (CX), (4) position-based crossover (PBX), and (5) modified order crossover (MOX); and *two* mutation operators: (1) reciprocal exchange (or swap) mutation (S) and (2) inversion mutation (I), were selected to deploy. Due to the space limitation, complete details about these operators are not given in this paper; a full description is therefore redirected to [10, 8, 5].

Such operators have been widely tested on the TSP and scheduling problem. As pointed out in p. 242 of [8], both problems are of sequencing type but they differ in characteristics and thus require different operators. The adjacency information (i.e. distances between cities) is important for the TSP but not applicable to the scheduling problem, while the relative order of items is not important for the TSP yet is of great concern in the scheduling problem. It is clear from the above encoding that the formulated DCSRP displays different properties than both the TSP and the scheduling problem, as no adjacency information exists and the relative order of integers in the series is also not a decisive factor. The motivation here is therefore to evaluate several operators and to learn about their performance on the DCSRP.

In order to complete the nuts and bolts of a GA, some other components need also to be defined. For mating selection, in this work, we employ the binary tournament selection strategy. This tournament selection could favor a low selection pressure [9], which has been known to be useful in preventing premature stagnation during the evolution. Additionally, it is required to have another selection mechanism in between two consecutive generations. In this study, when a new population is generated and evaluated, GAs evolves to the next generation by discarding the parent population except for its best individual. The best of the past population replaces the worst of the current population. This survivor selection mechanism is often referred to as the generational-with-elitism strategy.

4 Experiments and Discussions

In this section, an extensive analysis is presented to verify the validity of the proposed approach to the DCSRP and to answer the question of which genetic operators are most suitable for the formulated problem. For problem instances, we used two real warehouse datasets, having pallets stored in one and two aisles, respectively. For the dataset of one aisle, different instances with the symmetric order volumes of 10, 20, and 30 pallets were simulated by sampling randomly from the aisle, repeated twice for each volume, resulting in six instances. The same procedure was applied to the dataset of two aisles, but with the symmetric order volumes of 10, 20, 30, 40, 50, 60, and 70 pallets being sampled, resulting in 14 instances. Consequently, we solved a total of 20 problem instances. It should be noted that the four instances with a ten-pallet volume are practically small; they are still possible to be solved to optimality by an enumeration method. Their true optimal solutions could therefore be known exactly.

For algorithms, we considered five crossovers and two mutations as listed in Sect. 3, which amounts to a total of ten GAs. Each GA is then referred to by

its crossover's and mutation's abbreviations; the S+OX, for example, thereby refers to the GA using swap mutation and order crossover. All the GAs used a crossover rate $p_{cross} = 1$, a mutation rate $p_{mut} = 0.15$, and a fixed population size $popsize = 100$. On each problem instance, each GA was launched 30 times with different initial populations to mitigate its random effect. With 20 instances and ten GAs, we performed a total of 6000 runs. As a simple stopping criterion, we terminated a run after a fix number of generations, which was set empirically to 200 for those instances with a volume of 30 or less, and to 500 for those with larger volumes. It is worth noting that, for evaluating a solution, the distances in Eq. 1 were computed prior to the GA runs using a shortest path method; these distances thereby reflect the real traveling distances in the warehouse.

The performance of ten GAs on three typical instances are presented in Fig. 3. It can be clearly observed that the GAs with an OX or MOX crossover performs worst, no matter which mutation it goes with. For the GAs with PMX, CX, and PBX, the performance depends greatly on mutation: those GAs with inversion mutation perform worse than their counterparts with swap mutation; and for the GAs using swap mutation, those equipped with PMX and CX impressively outperform all the other GAs. The difference between S+PMX and S+CX is however unobvious. These trends are invariant across the three instance sizes.

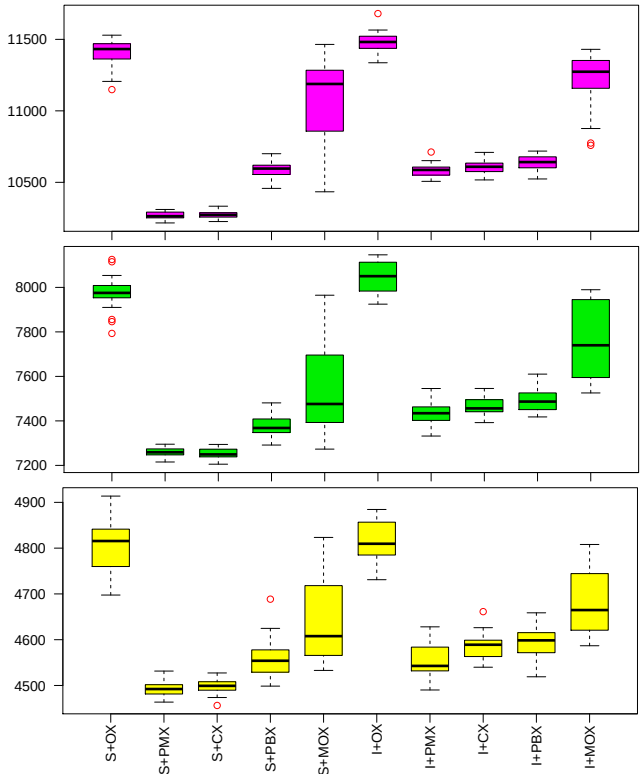

Fig. 3. Solutions in 30 runs of ten GAs across three instance sizes 30, 50, and 70, which are presented in the lower (yellow), middle (green), and upper (magenta) boxplots, respectively. In each plot, the lower a box, the better is the corresponding algorithm.

Table 1 summarize the best algorithm (i.e. the winner out of the ten GAs) in terms of its median-quality solution across 30 runs. This table is compiled from the results of all the 6000 runs of the ten GAs. Three main remarks can be drawn from this table. First, the GA with swap mutation and PMX crossover perform best on most of the instances and datasets. Second, the same GA but with CX crossover perform better on the instances of size 50; and this could also be observed in the middle boxplot in Fig. 3. Interestingly, for the two 30-pallet instances from the one-aisle dataset, the two mentioned GAs dominate each other by random. Third, for small instances with only 10 pallets, all GAs could finally find the true optima. The computational budget for the GAs here is much smaller than the enumeration method's. These results could therefore reflect somewhat the effectiveness and efficiency of the proposed approach.

Table 1. The best algorithms in terms of median over 30 runs of the best achieved objective value in each run. For each of the datasets (one and two aisles), two random order lists were sampled to create two random instances. A '–' indicates the unavailability of an instance for the corresponding size. An 'ALL' denotes that all ten algorithms could equally solve the instance to optimality, thus they are incomparable in the end.

Instance	10	20	30	40	50	60	70
One aisle 1	ALL	S+PMX	S+PMX	–	–	–	–
One aisle 2	ALL	S+PMX	**S+CX**	–	–	–	–
Two aisles 1	ALL	S+PMX	S+PMX	S+PMX	**S+CX**	S+PMX	S+PMX
Two aisles 2	ALL	S+PMX	S+PMX	S+PMX	**S+CX**	S+PMX	S+PMX

Finally, Fig. 4 provides some insight into the evolution of the ten GAs. The profiles of S+PMX and S+CX, in addition to the plots in Fig. 3, once again confirm their suitability for the formulated problem. The swap mutation provides better convergence than the inversion. And more importantly, the PMX and CX crossovers exhibit a persuasive success on the DCSRP.

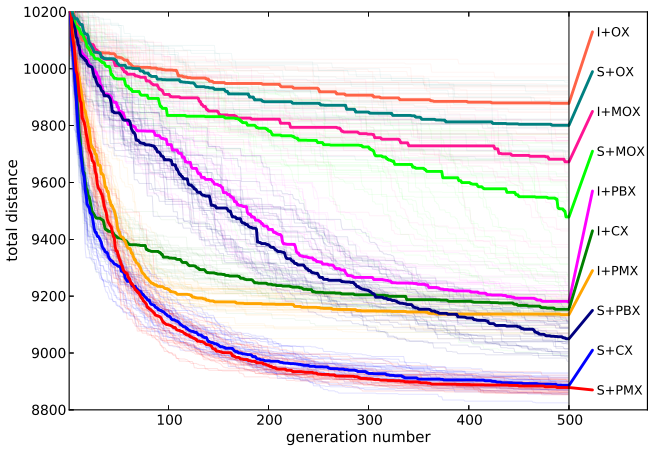

Fig. 4. The 30 run profiles of ten GAs on an instance of size 60. At each generation in a run, the best-so-far objective value of each GA was recorded, forming a profile for that run. Thirty profiles for each GA are plotted in the same transparent color in the background. The median of the 30 profiles is plotted by a thick line of the same color.

5 Conclusions

The double-cycle storage and retrieval problem is an important problem in warehouse operations. Notwithstanding its relevance and importance, there is a lack of general-purpose solution approaches to this problem in the literature. In this work, we have formulated the DCSRP as a permutation problem and solved it using genetic algorithms. Since this problem was not given due attention in the evolutionary optimization community, less or no knowledge about which genetic operators work best on it. Through an extensive experiment, we could draw a conclusion that the GA equipped with a swap mutation coupled with a PMX or CX crossover clearly outperforms the GAs using any of the other genetic operators under consideration. The second conclusion is that, for instances of as small size as 10 with the exact optimum being known, GAs can efficiently solve the instances to optimality within a limited number of function evaluations. What still remain interesting to know are the impact of other GA parameters, such as crossover and mutation rates, and the effectiveness of the algorithm when the problem size grows to a much larger number of items, e.g. hundreds or even thousands. These questions deserve further investigations into the DCSRP.

Acknowledgement. This research was supported by a grant (11 Transportation System – Logistics 02) from the Transportation System Efficiency Program funded by the Ministry of Land, Infrastructure and Transport of the Korean government. TDT is supported by a CORDI-S Doctoral Fellowship of the French National Institute for Research in Computer Science and Automation.

References

1. De Koster, R., Le-Duc, T., Roodbergen, K.J.: Design and control of warehouse order picking: A literature review. Eur. J. Oper. Res. 182(2), 481–501 (2007)
2. Goetschalckx, M., Donald Ratliff, H.: Order picking in an aisle. IIE Transactions 20(1), 53–62 (1988)
3. Gu, J., Goetschalckx, M., McGinnis, L.F.: Research on warehouse operation: A comprehensive review. Eur. J. Oper. Res. 177(1), 1–21 (2007)
4. ten Hompel, M., Schmidt, T.: Warehouse Management: Automation and Organisation of Warehouse and Order Picking Systems. Intralogistik, Springer (2007)
5. Larrañaga, P., Kuijpers, C., Murga, R., Inza, I., Dizdarevic, S.: Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artificial Intelligence Review 13(2), 129–170 (1999)
6. Lawler, E.L., Lenstra, J.K., Kan, A.R., Shmoys, D.B.: The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley, New York (1985)
7. Makris, P.A., Giakoumakis, I.G.: k-interchange heuristic as an optimization procedure for material handling applications. App. Math. Model. 27(5), 345–358 (2003)
8. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs (3rd ed.). Springer-Verlag, London, UK (1996)
9. Sokolov, A., Whitley, D.: Unbiased tournament selection. In: Proceedings of the GECCO 2005. pp. 1131–1138. ACM, NY, USA (2005)
10. Starkweather, T., McDaniel, S., Mathias, K., Whitley, D.: A comparison of genetic sequencing operators. In: Proc. of 4th ICGA. p. 69–76. Morgan Kaufmann (1991)
11. Theys, C., Bräysy, O., Dullaert, W., Raa, B.: Using a TSP heuristic for routing order pickers in warehouses. Eur. J. Oper. Res. 200(3), 755–763 (2010)