Paper Title: Natural Optimization Algorithms for the Cross-Dock Door Assignment Problem

Journal: IEEE Transactions on Intelligent Transportation Systems. H Index 112, Rank: Q1

Citation: Tarhini, A., Yunis , M., Chamseddine, M. (2016). Natural Optimization Algorithms for the Cross-Dock Door Assignment Problem, IEEE Transactions on Intelligent Transportation Systems, 17(8), pp. 2324 – 2333.

Link: https://ieeexplore.ieee.org/document/7434649

Natural Optimization Algorithms for the Cross-Dock Door Assignment Problem

Abbas Tarhini, abbas.tarhini@lau.edu.lb
Lebanese American University, Beirut, Lebanon

Manal Yunis, myunis@lau.edu.lb
Lebanese American University, Beirut, Lebanon

Mohamad Mohamad Jaafar Chamseddine, mchamseddine@matrixdesign.com.lb
Matrix Design Foundation, Beirut, Lebanon

***Abstract*—Cross-docking is a practice in logistics in which shipments are directly moved from an inbound truck into an outbound truck. A recognized problem in this domain is the assignment of trucks to doors in a way that the distance to be traveled between the doors is minimized. This problem is known as the Cross-Dock Door Assignment Problem (CDAP). A lot of research has been conducted regarding this topic still, up to our knowledge, none used Scatter Search (SS). In this paper, we implemented this evolutionary metaheuristic algorithm and tested it, then compared the results with those of another evolutionary algorithm, Genetic Algorithm (GA). The results indicate that the SS outperformed the GA.**

Keywords — Cross-docking, Door Assignment Problem, Scatter Search, Metaheuristics, Logistics.

I. Introduction

Cross-docking is a new model in logistics in which an inbound truck, usually with less-than-truckload (LTL) shipment, directly loads its cargo into a specified outbound truck without the need for temporary storage. In case storage was required, materials are stored for twenty four hours at maximum. In fact, this limited presence of storage will mitigate a cumbersome process common in traditional inventories. At some stages LTL shipments created a financial problem as it was causing losses. This loss is due to the violation of a basic economical concept known as the "economy of scale" which states that the truck is to be filled completely so that the expenses of shipping -which are fixed- will be divided on the items being shipped. Consequently, the main goal of the model is to sort and consolidate shipments from different suppliers into a fully loaded truck (LT) in an attempt to achieve economies of scale. An illustration of Cross-docking is shown in Fig. 1. Fig.1. (a) shows the process inside a distribution center. Fig. 1. (b) shows the case when the truck is not fully loaded and thus the supplier will encounter a higher increase in the price. Fig. 1. (c) shows the creation of a hub in which shipments from different suppliers are sorted based on their destination and then consolidate into one fully loaded truck (LT) and then moved to the consumer thus achieving economy of scale.

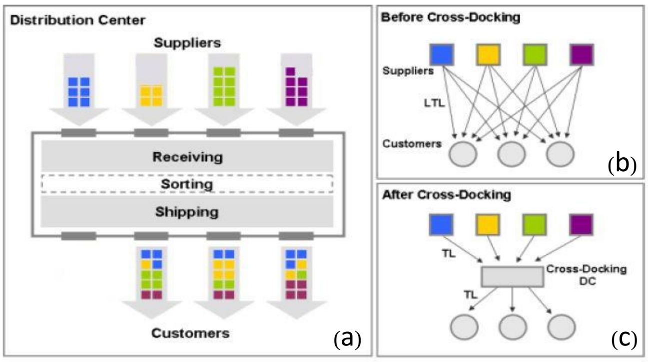

Fig. 1. Process inside a Cross-dock (left); The Distribution Industry before and after the emergence of Cross-Docking (right)

The decreased operational costs and improved response and delivery time made cross-docks a desirable practice for transportation and other similar industries. Still, with the fierce competition and low profit margins, cross-dock managers need to take actions on different decision levels to further increase effectiveness and efficiency. One crucial decision is to be taken at the operational level and is related to assigning trucks to doors. This problem is known as the Cross-dock Door Assignment Problem (CDAP) which is a nondeterministic polynomial time (NP)-complete problem. Such problems may be too computationally-intensive to find their exact solution. In these situations evolutionary techniques can be effective. In fact, due to their random nature, evolutionary algorithms are never guaranteed to find an optimal solution for any problem, but they will find a good solution if one exists.

After reviewing the recent available literature related to the cross-docking, this paper proposes a novel evolutionary heuristic for solving the CDAP. Specifically, a Scatter Search (SS) algorithm is presented to solve this problem. In fact the selection of the SS algorithm was motivated by Glover's perception [19] which states that "in contrast to other evolutionary methods like genetic algorithms, scatter search is founded on the premise that systematic designs and methods for creating new solutions afford significant benefits beyond those derived from recourse to randomization; further SS uses strategies for search diversification and intensification that have proved effective in a variety of optimization problems". Thus, we presented Scatter Search (SS) algorithm and tested it with randomly generated input which simulates actual data in cross-docks with up to 192 doors. The goal is to minimize the traveling distance of the handling machines when moving cargo from an inbound truck to an outbound truck. The results of the SS are then compared to those of another evolutionary algorithm (Genetic Algorithm). The test cases results clearly indicate that the SS solutions outperformed those of the GA for problem sizes greater than 24.

The rest of this paper is organized as follows. The next section is devoted for the literature review. Section 3 represents the mathematical formulation. The Scatter Search-based heuristic is discussed in details in Section 4. Then Section 5 presents the modified SS version. Section 6 presents the empirical results. Finally, the conclusion is found in Section 7.

II. Literature Review

One of the leading studies in the domain of door assignment problem in cross-docking was carried out by Tsui and Chang in 1990. Later they improved their work through presenting a more advanced approach for the solution [1]. The problem formulation included fixed origin doors on one side of the cross-dock and fixed destination doors at the opposite side taking into consideration that the cross-dock shape is rectangular. The goal was to determine which of the doors are to be classified as destination doors, and which are to be classified as origin doors so that the travel time of the forklifts that move the shipments is minimized. They proposed a bilinear programming model with the usage of branch and bound algorithm.

Other works [2] [3] focused on the assignment of the trucks to doors in a dynamic way in which both inbound and outbound doors are not fixed. This way, not only the material handling workload was minimized (i.e. the objective function), but also the congestion problem inside the terminal was mitigated. They presented their problem as a Mixed Integer Programming Model and used Simulated Annealing Algorithm to solve the model.

Aickelin and Adewunmi [4] proposed a Memetic algorithm based on the combination of both Local Search and Genetic Algorithms. The aim of the model is to minimize the traveling distance of the handling vehicles.

Bermudez [5] used Genetic Algorithm (GA) aiming at minimizing the weighted travel distance. The GA was then compared with a pair wise exchange algorithm. The results of the case studies showed an acceptable performance for the GA. The research also included an extensive study on the impact of parameter changes (especially those related to crossover and mutation) in the GA.

Another known problem in the domain of cross-docking is the scheduling problem in which the sequence of trucks in a cross-dock is to be determined. Such arrangement is performed in order to decrease the time makespan therefore minimizing costs. In their paper, Hazzoury et al. [6] present a reformulation for the integer programming model that was introduced by Zhaowei Miao et al. The main objective of the research is minimizing the number of trucks that leave without being fully filled, and at the same time minimizes the operations cost related to the shipping process.

Arabani et al. [7] used simulated annealing algorithm to schedule the arrival and departure of trucks at the cross-docking terminal. They only discussed one case in which there is only one cross-docking terminal.

Larbi et al. [8] proposed a graph based model to approach the transshipment scheduling problem with the goal of minimizing the cost. Transshipment operations are those related to: first, the cost resulting from storing certain items when their outbound truck is not available, second, the cost resulting from the move of the trucks from the doors to the parking zone and vice versa. The model was tested on a case of one outbound and one inbound door.

Apart from scheduling and door assignment, Agustina et al. [9] covers almost all of the significant literature related to cross-docking in an impressive and comprehensive manner. They divided the highlighted biography into: first, those discussing the operational level in which short term decisions have to be taken, e.g. assignment and scheduling problems; second, the research focusing on the tactical level, e.g. those dealing with the layout of the cross-dock; third, those related to the strategic level such as determining the location of the cross-dock or number of vehicles in the network.

Bartholdi and Gue [10] proposed a model for the layout of a cross-dock taking into consideration the door assignment problem. The main goal of the work is to change the layout of the inventory in order to reduce the labor cost through decreasing the traveling distance and congestions. One important outcoming was the suggestion of clustering high flow inbound doors with their relative outbound doors therefore decreasing the traveling distance, additionally, disperse these clusters among the cross-dock in an attempt to reduce congestion. They claim after implementing their model there was an increase in the productivity of the cross-dock by 11%. Later Bartholdi and Gue [11] focused on another paper on the best physical shape for a cross-docking inventory. They reached a conclusion that the "I" shape is needed for cross-docks with doors equal to or less than 150. And that shape "T" is adequate for those of size between 150 and 200. Finally, shape "X" is the best for inventories with more than 200 doors.

Additionally, Vis and Roodbergen [12] presented a model for the storage of products in a cross-dock. This model is designed for cases in which some items are to be stored for a short period of time until the outbound truck is available to load them out. They used a minimum cost flow problem model.

III. Cross-Docking Door Assignment Problem

A. *Preliminary*

With a fierce competition with rivals, transportation and logistics companies noticed a decrease in the profit margins if some of their trucks are not fully loaded. Accordingly, efficient and effective measures had to be taken at the operational level by assigning trucks to doors. The effect of such assignment is illustrated in the following example on cross-dock inventory. This example shows two different permutations to assign four trucks to four doors with their corresponding costs.

The first permutation is illustrated in Fig. 2. In this permutation, the four trucks 1, 2, 3, 4 are assigned to the corresponding four doors 1, 2, 3, 4.

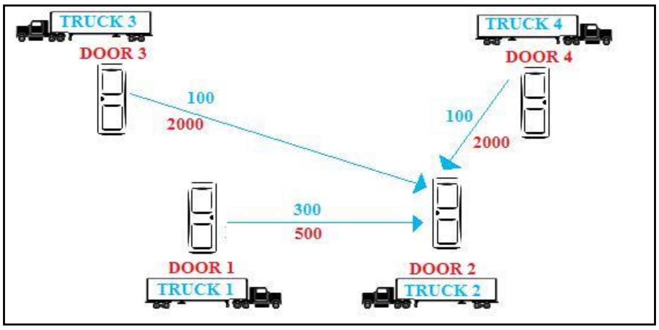

Fig. 2. Truck-door assignment, first permutation

It has four trucks and four doors where trucks 1, 2, 3, and 4 are assigned to doors 1, 2, 3, and 4 respectively. Fig. 2 clearly shows that trucks 1, 3, and 4 are inbound trucks, and truck 2 is an outbound truck. The blue numbers above each arrow represent the exact flow quantity between two trucks. For example, the flow quantity between trucks 1 and 2 is 300. As for the red numbers below the arrows, they represent the distance between the doors. In Fig. 2, the distance between door 1 and 2 is 500. It is important at this stage to mention that trucks 1 and 2 having the highest flow among them are assigned deliberately to doors 1 and 2 respectively, which in their turn have the smallest distance spanning them. The cost of such a permutation would be calculated as follows (this formula will be verified in the following section): $\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{m=1}^{n} \sum_{n=1}^{n} f_{mn}\, d_{ij}\, x_{mi}\, y_{nj}$ = (300*500) + (100*2000) + (100*2000) = 550,000. Were n is the total number of trucks in this case equals to 4. "i" and "j" stand for the inbound and outbound doors, "m" and "n" represent the inbound and outbound trucks. "f_{mn}" serves as the flow between two trucks. "d_{ij}" represents the distance between two doors. "x_{mi}" indicates whether truck "m" is assigned to door "i" or not through taking a Boolean value either 0 or 1. Similarly, "y_{nj}" determines whether truck "n" is assigned to door "j".

Obviously, the combination of the flow and distance has a tremendous effect on the total cost. Therefore it is rational to assume that if we assigned the trucks with the highest flow between, on the doors separated with the least distance, we would get an optimized solution. To explain this concept further, consider the following figure Fig. 3 which represents a different permutation of that given in Fig. 2 where all trucks are assigned to different doors. For example, truck 4 was assigned to door 1 instead of door 4. The reason behind this reallocation is to assign trucks 1 and 2 that have the highest flow amount between two new doors that have the largest distance spanning them. This is completely the opposite of what is given in Fig. 2 in which doors 1 and 2 with the shortest separating distance were chosen. The cost of the new permutation will be as follows: (300*3000) + (100*2000) + (100*500) = 1,150,000 which is extremely greater than the first cost which was 550,000 thus making no room for any doubt regarding the logic that will be used as the building block for the creation of the initial population.

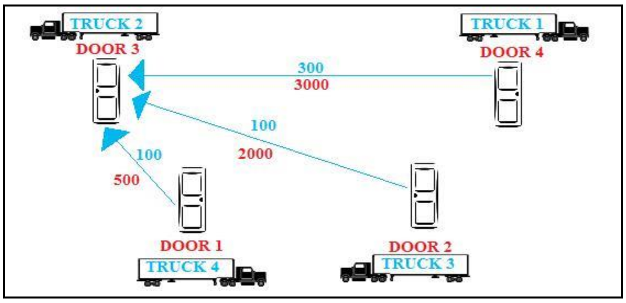

Fig. 3. Truck-door assignment, second permutation.

B. Mathematical Formulation

The mathematical model of the Cross-dock Door Assignment Problem is based on [1] and is as follows:

1. Parameters:
   - ***M*** number of inbound trucks;
   - ***N*** number of outbound trucks;
   - ***I*** number of inbound doors;
   - ***J*** number of outbound doors;
   - f_{mn} the flow between the inbound and outbound trucks;
   - d_{ij} distance between inbound and outbound doors;
2. Decision Variables:
   - x_{mi} = 1 if inbound truck **m** is assigned to inbound door **i**, else $x_{mi} = 0$;
   - y_{nj} = 1 if outbound truck **n** is assigned to outbound door **j**, else $y_{nj} = 0$;

Minimize:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{m=1}^{n} \sum_{n=1}^{n} f_{mn}\, d_{ij}\, x_{mi}\, y_{nj} \qquad (1)$$

Subject to:

$$\sum_{i=1}^{I} x_{mi} = 1 \quad \text{for } m = 1, 2, \ldots, M \qquad (2)$$

$$\sum_{m=1}^{M} x_{mi} = 1 \quad \text{for } i = 1, 2, \ldots, I \qquad (3)$$

$$\sum_{j=1}^{J} y_{nj} = 1 \quad \text{for } j = 1, 2, \ldots, J \qquad (4)$$

$$\sum_{n=1}^{N} y_{nj} = 1 \quad \text{for } n = 1, 2, \ldots, N \qquad (5)$$

$$X_{mi} = 0 \text{ or } 1 \quad \text{for all } m, i$$

$$Y_{nj} = 0 \text{ or } 1 \quad \text{for all } n, j$$

Notice that (1) is the objective function or the cost. Constraint (2) ensures that each inbound truck is assigned to one inbound door. Constraint (3) ensures that each inbound door is assigned one inbound truck. Constraint (4) ensures that each outbound truck is assigned to one outbound door. Constraint (5) ensures that each outbound door is assigned one outbound truck.

Additionally, the above presentation is a bilinear program with high complexity. Furthermore, it is a variant of the Quadratic Assignment Problem making it an NP complete problem [1].

IV. A CLASSICAL SCATTER SEARCH-BASED HEURISTIC

In his paper “A Template for Scatter Search and Path Relinking” [13], the father of the Scatter Search Algorithm Fred Glover defines it as an evolutionary method that combines solutions in order to create new ones. Similarly, [14] defines SS as a heuristic that creates initial solutions “purposely” and not randomly, and then explores the population in a systematical manner to produce new solutions mainly through combination. Such technique is used for solving combinatorial and nonlinear optimization problems and has so far proven its effectiveness in doing so [15].

SS was first introduced by Fred Glover in 1977 and is based on concepts of his work conducted in 1963 which included methods that were related to combining decision rules and problem constraints. These methods formulate the building blocks for the combination of solutions in SS [16].

Additionally, Glover asserts that keeping a small population of elite solutions, in the Reference Set, to be combined and improved will aid in the process of shifting towards the desired optimal solution space iteration after iteration [14]. An illustration of SS algorithm is shown in Fig. 2.

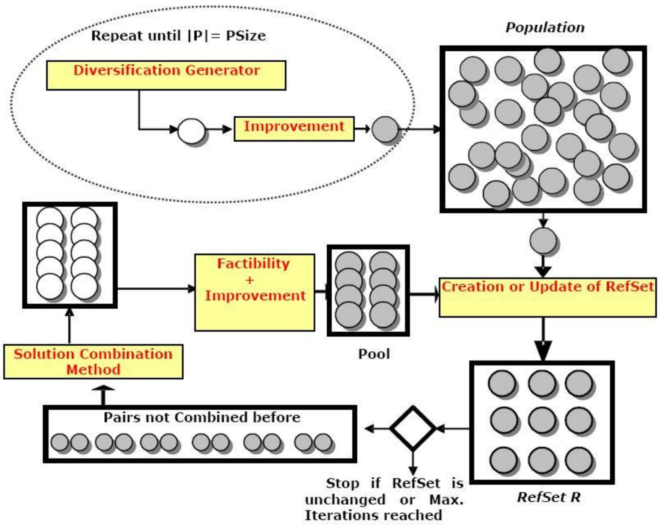

Fig. 4. Scatter Search Template

In [16] an outline for the classical SS algorithm is provided and is as follows:

1. Start by creating the initial population using the Diversification Generation Method keeping in mind the goal of maintaining diversity. The Improvement method might be implemented at this stage to further enhance the current solutions. Next, select the best of these solutions present in the population and insert them into the Ref Set using the Reference Set Update Method. Notice that the best solutions does not mean only those with the best cost or objective value, but also include other solutions that have an undesired cost but believed to play a role in maintaining diversity.

2. Start the combination process through identifying the subsets that are going to be combined. The Subset Generation Method prepares these subsets and delivers them to the Solution Combination Method that will undertake the combination process based on the linear combination of the combined solutions and not through random selection. Again, after combining a solution we need to improve the newly combined one so we use the Improvement Method.

3. Select the best newly combined and improved solutions to be added to the Ref Set. The selection process of these best solutions should take into consideration two factors, the cost of the solution itself, and the level of diversity it embeds.

4. Repeat steps 2 and 3 until either reaching the specified number of iterations or reaching the level in which no change is occurring in the Ref Set.

V. THE PROPOSED ALGORITHM

Our SS heuristic adheres to the basic outline of the Scatter Search Algorithm and has the following procedure which is further detailed in the below subsections:

1. *Diversification Generation Method:* responsible for the creation of the pool of diversified solutions and is divided into two main phases.
2. *Creating and Updating the Reference Set:* This is a sub-set of the initial pool. It consists of the elite solutions of the starting pool. The Reference Set is updated continuously when Combination and Improvement is carried out. This is done through replacing solutions in the Reference Set with better new solutions produced after combination and improvement.
3. *Subset Generation Method:* responsible for preparing the input solutions for the Combination Method. It is created in its simplest form in which the Reference Set is divided into pairs processed in order.
4. *Combination Method:* the input is the pairs from the Subset Generation Method. The combination is done in two phases detailed below.

5. Improvement Method: enhances intensification. It depends on switching the doors of two trucks randomly.

The detailed methods for the proposed SS heuristic are as follows:

A. Diversification Generation Method (DGM)

This method is divided into two main phases. Phase I which is the guided selection process that aims at creating an elite initial solution. And Phase II which adds diversity to the selection through the insertion of randomly created solutions.

1) DGM: Phase I

This phase of the method is based on a special technique which will guide the selection process of the candidate solutions. The technique is based on the following:

a) First, create a new matrix based on the given flow matrix and has three elements for each index. The first element is dedicated for the flow amount f_{mn}, the second element is for *m* (inbound truck) and the last element is for *n* (outbound truck). Then sort the newly created matrix in ascending order based on the flow amount f_{mn}. Use the same process to create a similar matrix for the distance matrix. Notice that the cases in which $f_{mn} = d_{ij} = 0$ are not included in the new distance and flow matrices.

b) Second, divide horizontally each of the two matrices into four equal sub-lists. Thus, now the focus will be on eight sub-lists.

c) Third, randomly and without replacement, select an index from the first flow sub-list and obtain its contents; i.e., trucks *m* and *n* in addition to the flow (f_{mn}) residing on this index.

d) Forth, after identifying both trucks *m* and *n* we need to determine which one of the following cases apply:

   i. Case 1: both of the selected trucks have been selected earlier and assigned to two doors. Therefore, we should reselect two trucks again (repeat the third step i.e. step 1.c).

   ii. Case 2: both of the selected trucks have not been selected yet, therefore not assigned any door. At this stage, select randomly and without replacement an index from the first distance sub-list then determine doors *i* and *j* in addition to the distance d_{ij} residing on this index. For the doors, we are to encounter one out of two possibilities:

      1. First, one or both of the selected doors have been selected previously and assigned a truck, hence the selection process should be repeated –and keep doing so- until we get a pair of doors that has not been assigned any trucks yet.

      2. Second, both of the selected trucks have not been selected before, thus assign the chosen pairs of trucks to those of selected doors i.e. assign truck *m* to door *i* and truck *n* to door *j*. It is crucial at this level to carry out some procedures that make sure that the trucks and doors are not selected again as such a reselection will lead to the assignment of the same truck to two different doors at the same time, or assigning two different trucks to the same door, thereafter breaking the imposed constraints for the Assignment Problem.

   iii. Case 3: one of the two trucks has been selected earlier. Do the following: First, mark up the door that already holds truck *m* as door *i*. Then go back to the four distance sub-lists and locate all of the indexes that have door *i* as an element then copy those indexes into a new matrix. Next, select randomly an index from the newly created matrix then identify doors *i* and *j*. Again, one of two cases is to be encountered:

      1. Both door *i* and *j* are already assigned trucks, therefore the random selection of a new pair of doors *i* and *j* should be repeated.

      2. Second, one of the doors is free thus the assignment of the unassigned truck to the unassigned door can be done.

e) Repeat until all the trucks are assigned to all the doors

2) DGM: Phase II

Still, in order to keep the Initial population diverse, we decided to mix it with a random population that forms around 50% of the total pool size (i.e. 150 solutions).

B. Reference Set Creation

From the 300 candidates in the initial population there are solutions with high costs, therefore being undesired ones, and candidates with low cost creating excellent solutions. And since Scatter Search is based on the concept of combining and improving elite solutions instead of manipulating both bad and good solutions, we are to create a mini-pool of outstanding candidate solutions to perform the predefined procedures on. This mini-pool is called the Reference Set and it contains from twenty to thirty solutions selected using the Reference Set Update Method. Such small pool will reduce the complexity of calculations and the cost for performing them including computational power and time.

C. Combination Method

Combination Method in SS is not limited to the combination of only two solutions. It grants the combination of up to five solutions simultaneously. In our algorithm, we focused on the combination of two solutions at once, performing improvement, and then moving to the next pair of solutions already prepared by the Subset Generation Method through simply selecting the next pair of solutions from the Reference Set.

The outline of the Combination Method in our algorithm is divided into two main phases. The first phase starts with identifying the door that holds the same trucks in both solutions. If applicable, then move them into a new solution array. The solution array is incomplete and requires some missing trucks to be assigned to the free doors. This is going to take place in the second phase.

1) Combination: Phase I

Take the first pair of solution from the Subset Generation Method (for simplicity call them solution 1 and solution 2) then identify the doors that hold the same truck in both solutions, i.e. a door *i* that is assigned the same truck *m* in both solutions 1 and 2. After identifying these doors, we create a new array for the combined solutions in which the trucks common to doors are clearly identified, and so are the remaining trucks and doors that are still free. Simultaneously, create another list containing the trucks that are free.

2) Combination: Phase II

The free doors and trucks remaining from Phase I are going to be assigned based on the following algorithm:

a) Start from the first/next element in the new combined solution array.

b) Check if that door is free or already assigned a truck from the first combination phase.

c) If it turned out that it is already assigned a truck then move to the next element (door).

d) Else, if the door was not assigned, label it as door *j* then select randomly either 1 or 2. These values stand for whether we are going to choose solution 1 of the pair to continue the process with, or continue with solution 2.

e) Let us assume solution 2 was selected, what we care about in solution 2 is the truck number that resides on the door *j* determined in step 4 (i.e. *d*).

f) Identifying truck *n* on door *j* in solution 2 is not enough to finish the process and assign that truck to the free door since it could be the case that this truck has been already assigned to a door. If it was so, the following is to be done:

1. Check the list the contains the free doors and select randomly a truck from it.
2. After selecting the truck, assign it to the free door *j* in the newly combined array.
3. Then, remove that truck from the list containing the free trucks so that it would not be selected again.

g) Else, if the truck was not assigned before:

1. Assign it to the free door *j*.
2. remove it from the list containing the free trucks.

h) If some elements (doors) are still free, repeat the process starting from step 1

i) Else, in case all items were selected, then stop this Combination process and add the newly combined solution to an Intermediate Reference Set with solutions 1 and 2, then move to the next pair selected by the Subset Generation Method to combine starting from phase I.

D. Improvement Method

The Improvement Method in Scatter Search is again an extremely important method specially when comes to the intensification procedures that aim to further investigate a current solution. During intensification, the elite solutions (in our case those solutions from the Reference Set and the newly combined ones that are found in the Intermediate Reference Set) are being examined thoroughly. Therefore, intensification is completely opposite to diversification. Diversification focuses on exploring solutions that has not been discovered before [17] [18].

Back to our program, the Improvement method was implemented in its simplest form:

a) Start with the first / next solution in the Intermediate Reference Set.

b) Choose a door *i* randomly and determine the truck *m* assigned to it.

c) Choose another door *j* randomly and again determine the truck *n* assigned to it.

d) Exchange the trucks selected, i.e. assign truck *m* to door *j* and truck *n* to door *i.*

e) Calculate the cost.

f) Repeat this process thirty times for this solution then go back to step one if still unimproved solutions, else

g) Stop the Improvement Method and call the Update Reference Set Method that will select the best 30 solutions and create therefore the new Reference Set

h) After the new updated Reference Set is created, repeat combination and improvement methods on each pair of the Reference set and keep iterating until the iteration number is fulfilled.

VI. Experimental Results

We tested our SS algorithm on a set of five test cases; each of these test cases has a different number of trucks. each instance is written in a file of type DATA. Each file has 3 main components. The first one is the size of the instance (i.e. N) found in the first line of the file. Then comes the flow matrix (N*N) followed by an empty line and after it comes the final part which represents the distance matrix (N*N). Notice that all the distance matrices are symmetric (e.g. if the distance from door 5 to door 10 is 200 unit of measure, then the distance from door 10 to door 5 is also 200 unit of measure). The details of each of the five test cases are presented in TABLE I.

TABLE I. DETAILS ABOUT THE FIVE INSTANCES

Test Cases	**Number of trucks**	***Inbound trucks***	***Outbound trucks***	***Flow range***	***Distance range***
TC 1	12	5	7	10 to 400	10 to 300
TC 2	24	10	14	7 to 500	10 to 310
TC 3	48	20	28	10 to 200	10 to 200
TC 4	96	45	51	5 to 300	5 to 300
TC 5	192	90	102	10 to 100	10 to 100

To further illustrate the above table consider the following explanation for the first instance of size 12, i.e. there are 12 trucks to be assigned to 12 doors. 5 out of these 12 trucks are inbound trucks and the remaining 7 trucks are outbound. The flow quantities range from 10 to 400 (e.g. the flow from truck 1 to truck 9 is 10 and the flow from truck 2 to truck 9 is 400) whereas the distance values range from 10 to 300.

Computation results of the 5 test cases are shown in TABLE II which compares the results of the SS to those of the GA.

TABLE II. RESULTS FOR THE 5 TEST CASES

		SS	**GA**
TC 1 - Iterations: 1500	Total time	0:01:43.100	0:02:05.85
	Best Cost	335300	**333900**
	Time to best	0:00:07.07	0:00:01.85
	Best at iter.#:	217	139
TC 2 - Iterations: 1000	Total time	0:03:11.82	0:02:49.68
	Best Cost	1,528,370	1,528,370
	Time to best	**0:00:40.68**	0:01:12.02
	Best at iter.#:	232	823
TC 3 - Iterations: 1500	Total time	0:10:42.0	0:03:02.95
	Best Cost	**5,014,864**	5,026,594
	Time to best	0:09:47.18	0:02:59.18
	Best at iter.#:	1358	1482
TC 4 - Iterations: 1500	Total time	0:24:11.59	0:02:51.38
	Best Cost	**46,607,952**	47,887,658
	Time to best	0:23:24.64	0:02:38.45
	Best at iter.#:	1453	1417
TC 5 - Iterations: 1500/3000	Total time	1:28:31.66	0:30:58.01
	Best Cost	**7,773,033**	7,806,548
	Time to best	1:25:39.90	0:30:46.56
	Best at iter.#:	1451	2985

The GA has a population of 200 chromosomes. Two chromosomes are selected randomly. Crossover is performed using a swipe window method. Next, Mutation is carried out through a random selection of two trucks and then the substitution of their doors. At this level four candidates are present, the two parent chromosomes and the two offspring. Only two candidates are to be returned back to the original pool. The first is the best among them and the other is randomly selected.

The SS algorithm and GA were both implemented using VB.Net. Furthermore, the tests were carried out on a laptop Pentium(R), dual-core CPU T4300- 2.10 GHz, and 2 GB RAM with Windows Vista SP 1.

For each instance other characteristics are shown and also compared such as the total time, best found cost, time taken to find this best cost, and at which iteration the best known cost was found. For the first problem set of size 12, the GA got a better solution. Both SS and GA got the same best known solution for problem size 24, still the SS found it in less time. For the third problem of size 48, the fourth problem of size 96, and the final problem of size 192, the SS found the best known solutions. Therefore it is clear that the GA was performing better or equal to the performance of the SS in small problem sizes (less than 24) but the SS outperformed the GA in large problem sizes although the computational time was longer.

Fig. 3. compares the best solution found of SS and GA algorithm at each iteration of problem with size 12.

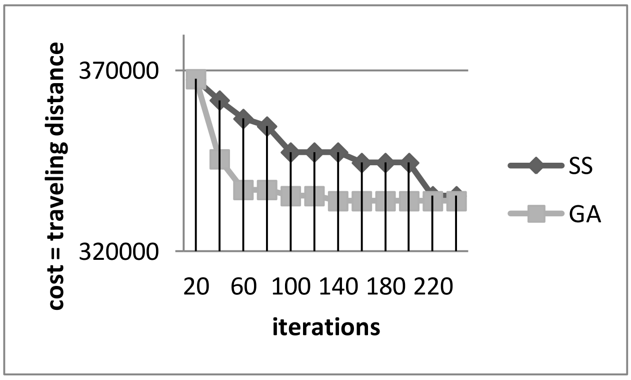

Fig. 3. Best solution-found at each iteration - problem size 12

The below figures, 4, 5 and 6 compares the results of SS and GA algorithm regarding the best solution found at each iterations of sizes 24, 96, and 192.

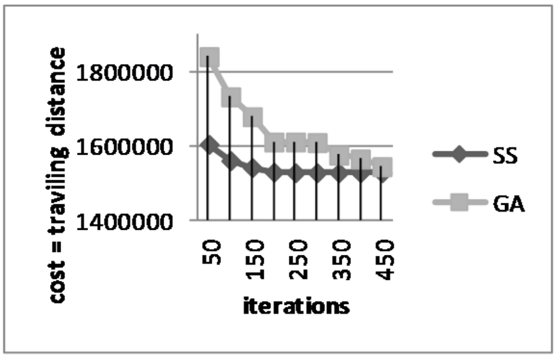

Fig. 4. Best solution found at each iteration - problem size 24

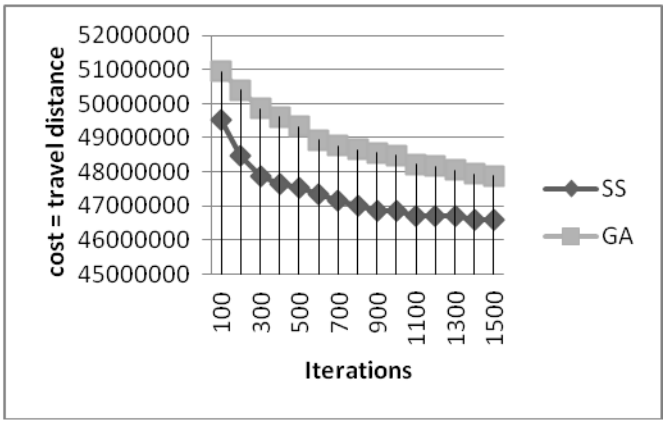

Fig. 5. Best solution-found at each iteration - problem size 96

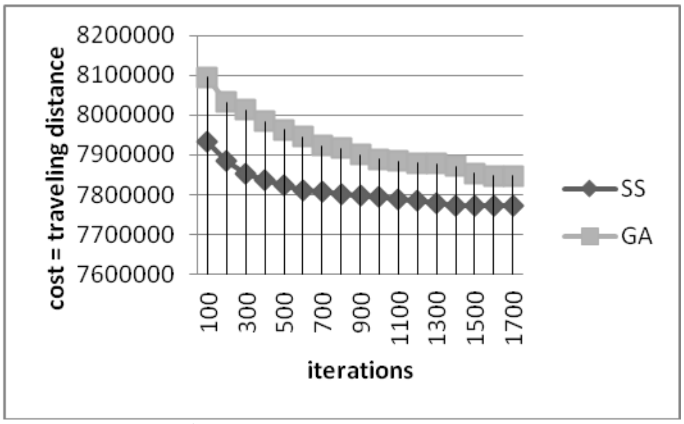

Fig. 6. Best solution-found at each iteration - problem size 192

VII. CONCLUSION

Scatter Search is an evolutionary algorithm that has been utilized to solve the assignment problem in cross-docking inventory. The problem entails the distribution of inbound and outbound trucks to the doors of the inventory in the best manner possible so that the cost of transportation of goods and items are minimized. Based on that, the methods of the SS have been enhanced to solve this problem.

Mainly, the focus was on the Diversification Generation Method. A unique method has been used to boost the quality of the initial population, consequently, the quality of the Reference Set. Clearly, this process affected the final results as it aided in yielding the best known solution at earlier stages of the iterations giving an edge when compared to other related work and research that did not focus on generating such enhanced solutions. Additionally, the Combination and Improvement Methods have been also reinforced to give better results.

The paper major contribution is through presenting and testing a unique algorithm (SS) that has not been used before for solving the assignment problem in cross-docks. The results indicate that our algorithm SS-based outperformed other researched algorithms.

REFERENCES

[1] L. Y. a. C.-H. C. Tsui, "An optimal solution to a dock door assignment problem," *Computers & Industrial Engineering,* vol. 23, no. 1, pp. 283-286, 1992.

[2] H. J. Carlo, Door assignment and sequencing problems in crossdocks and container terminals, vol. 68, 2007.

[3] Y. A. a. H. J. C. Bozer, "Optimizing inbound and outbound door assignments in less-than-truckload crossdocks," *IIE Transactions,* vol. 40, no. 11, pp. 1007-1018, 2008.

[4] U. a. A. A. Aickelin, "Simulation optimization of the crossdock door assignment problem," in *In: UK Operational Research Society Simulation Workshop 2006 (SW06), Leamington Spa. (Unpublished)*.

[5] R. a. M. H. C. Bermúdez, "A genetic algorithm approach to door assignments in breakbulk terminals," *University of Arkansas, Mack-Blackwell National Rural Transportation Study Center,* 2001.

[6] F. B. T. a. F. S. Hazzoury, "Heuristic for Cross Dock Problem," École Centrale de Lille, *2010*.

[7] A. B. F. R. a. S. N. R. Arabani, "Applying Simulated Annealing *Algorithm* for Cross-Docking Scheduling," In Proceedings of the World Congress on Engineering and Computer Science, vol. 2, 2009.

[8] R. e. a. Larbia, "Scheduling of transhipment operations in a single strip and stack doors crossdock," in *Proceedings of the 19th International Conference on Production Research*, 2007.

[9] D. C. K. M. L. a. R. P. Agustina, "A Review: Mathematical Modles for Cross Docking Planning," *International Journal of Engineering Business Management, vol*. 2, no. 2, pp. 47-54, 2010.

[10] J. J. Bartholdi, a. K.R. Gue, "Reducing labor costs in an LTL crossdocking terminal," *Operations Research,* vol. 6, no. 48, pp. 823-832., 2000.

[11] K. R. Gue. a. J. J. Bartholdi, "The best shape for a crossdock," *Transportation Science,* vol. 38, no. 2, pp. 235-244, 2004.

[12] I. F. Vis, a. K. J. Roodbergen. "Positioning of goods in a cross-docking environment," *Computers & Industrial Engineering,* vol. 54, no. 3, pp. 677-689, 2008.

[13] F. Glover, "A template for scatter search and path relinking," *Artificial evolution,* p. Springer Berlin Heidelberg, 1998.

[14] M. Laguna *and* R. Marti, "SCATTER SEARCH," in *Scatter search: methodology and implementations in C,* vol. 1, 2003.

[15] F. M. L. a. R. M. Glover, "Scatter search and path relinking: *Advances* and applications," *International series in operations research and management science,* pp. 1-36, 2003.

[16] F. M. L. a. R. M. Glover, "Fundamentals of scatter search and path relinking," *Control and cybernetics,* vol. 39, no. 3, pp. *653*-684, 2000.

[17] F. Glover and M. Laguna, "Tabu Search," *Boston: Kluwer academic publishers,* vol. 22, pp. 1-18, 1997.

[18] Y. Rochat and É. D. Taillard, "Probabilistic diversification and intensification in local search for vehicle routing," *Journal of Heuristics,* vol. 1, no. 1, pp. 147-167, 1995.

[19] Glover, F., M. Laguna and R. Martí. “Scatter Search”. Theory and Applications of Evolutionary Computation: Recent Trends, A. Ghosh and S. Tsutsui (Eds.) Springer-Verlag(2003).