2012 IEEE International Conference on Systems, Man, and Cybernetics
October 14-17, 2012, COEX, Seoul, Korea

Application of NSGA-II with Local Search to Multi-dock Cross-docking Sheduling Problem

Yu Guo, Zhou-Rong Chen, Yong-Liu Ruan and Jun Zhang (Corresponding Author)
Sun Yat-sen University, P.R. China
Key Laboratory of Digital Life, Ministry of Education, P.R. China
Key Laboratory of Software Technology, Education Dept. of Guangdong Province, P.R. China
junzhang@ieee.org

***Abstract*—Cross-docking is now widely applied to trucking industry, for which the optimal schedule of the trucks is a crucial issue. In the cross-docking scheduling problem, the objectives of minimizing the operation cost and maximizing the possibility of punctuality are both important. In this paper, a non-dominated sorting genetic algorithm version II (NSGA-II) with a novel greedy local search strategy is proposed to solve the multi-objective optimization problem. NSGA-II can provide decision makers with flexible choices among the different trade-off solutions, while the local-search strategy is employed to accelerate the convergence speed. In the experiments, four criteria are applied to evaluate the strengths of the proposed algorithm. Experimental results on both small and large size of problems show the accuracy and efficiency of the propose strategy.**

Keywords- Cross-docking; just-in-time schdeule; NSGA-II; multi-objective

I. INTRODUCTION

Cross-docking scheduling [1][2] is a logistics strategy which arranges the inbound and outbound trucks sequences and parking positions to optimize the system performance. In the cross-docking system (as Fig. 1 shows), inbound trucks and outbound trucks can carry different kinds of products and have differential limited truckload. They arrive at the docks on different sides according to the arranged sequences. All of the products which are dedicated to the specific outbound truck are brought by inbound trucks and then reorganized and stored in a temperate storage [3]. When their destined outbound trucks arrive, these products are loaded onto the trucks and taken away.

Several researches have been done in this area in the last few decades. Traditionally, the objective is minimizing operation cost and enabling the products to go through the docking system as fast as possible. For example, Boysen [4] employed dynamic programming and simulated annealing to minimize operation cost and tardiness in a cross-docking model without temporary storage. S. Ley [5] used Genetic Algorithm (GA) to simulate the same model Boysen had worked on. But recently, a more comprehensive objective is proposed. It is to ensure the time punctuality which means products arrive at the outbound trucks as close to the time customer wanted as possible. This schedule can not only reduce the earliness which causes products be held in the temporary storage but also reduce the tardiness which causes the products shortage. Boloori [6] proposed three different methods including GA, particle swarm optimization (PSO), and differential evolution (DE) to solve the problem. Afterwards, Boloori [7] proposed another three famous multi-objective algorithms to solve the same problem, i.e., non-dominated sorting genetic algorithm-II (NSGA-II), strength Pareto evolutionary algorithm-II (SPEA-II), and sub-population genetic algorithm-II (SPGA-II).

This paper focuses on the multi-dock model with temporary storage and has an equal number of inbound and outbound docks on each side. The objectives are minimizing the operation cost and ensuring the punctuality. In this paper, the operation cost is measured by the total distances which products need to travel through from the inbound docks to the outbound docks. The longer distance means the more operation cost. To approach to the time customer wanted, we combine the earliness and tardiness to measure the time punctuality. The model is more complicated for it considers multi-dock and more realitic for both the distance cost and the time punctuality are taken into consideration.

Since NSGA-II [8] has been widely used for solving multi-objective problems for its promising performance, this paper employs NSGA-II to find a solution set which approximates to the optimal sequences of inbound and outbound trucks. Moreover, a greedy local search procedure is designed for this specific problem. The trucks with products which are needed soon are arranged to arrive the docks earlier. The local search helps to accelerate the convergence speed to improve the efficiency of NSGA-II.

In the experiment, the proposed strategy is tested on a system with three inbound docks and three outbound docks. Two different problem sizes are tested, small number (6-15) of trucks and large number (16-25) of trucks on inbound or outbound side. The results obtain by the proposed method are compared with an enumeration algorithm in a small problem size and with a greedy algorithm in a large problem size. The results show that the proposed algorithm not only obtains high solution accuracy, but also is very efficient. Since the proposed strategy is suitable for solving problems with different scales, it is very flexible and robust.

The rest of this paper is organized as follows. Section II presents the formulations of the multi-objective cross-docking system. Section III describes NSGA-II and the local-search scheme implementation. Experimental results of this algorithm

This work was supported in part by the National Science Fund for Distinguished Young Scholars No.61125205, National Natural Science Foundation of China No.61070004, and NSFC Joint Fund with Guangdong under Key Project U0835002.

are displayed in Section IV and section V draws a conclusion to the whole paper.

II. Formulation of Cross-docking System Model

This section describes three functions to determine storage cost (caused by earliness) and back-order loss (caused by tardiness) as well as the distance from inbound truck to outbound truck. These three factors are the keys to evaluate the performance of the cross-docking. Table I introduces the notations used in the objective functions and all subsequent formulations:

TABLE I NOTATION OF THE FUNCTION

Variables	Explanation
I	Number of inbound (outbound)trucks
N	Number of inbound (outbound) docks
L	The length of the docking platform
W	The distance between two neighbor docks on the same side
D_1	The changeover time for inbound trucks
D_2	The changeover time for outbound trucks
C_1	The storage cost for every product in every unit of time
C_2	The back-order loss for every product in every unit of time
$d[j]$	The due time for the *j*th outbound truck
$l[j]$	The leave time of the *j*th outbound truck
$t[i]$	The arriving time of the *i*th inbound truck
$T[j]$	The total cost of being late for the *j*th outbound truck
$E[j]$	The total cost of being too early for the *j*th outbound truck
$S_{in}[i]$	The arriving order of the *i*th inbound truck
$S_{out}[j]$	The arriving order of the *j*th outbound truck
$G[i][j]$	The number of products moving from the *i*th inbound truck to the *j*th outbound truck
S	The total operation cost in the system
$s[i][j]$	The distance from the *i*th inbound truck to the *j*th outbound truck in docking system
$P_{in}[i]$	The position of the *i*th inbound truck in the receiving docks
$P_{out}[j]$	The position of the *j*th outbound truck in the shipping docks

A. Distance Function:

In this particular problem, the operation cost is measured by the distance that products need to travel. To calculate the distance, the length of the docking platform and the distance between two adjacent docks on the same side are defined as L and W respectively. For example in Fig. 1, the distance s is calculated by using Pythagorean Theorem:

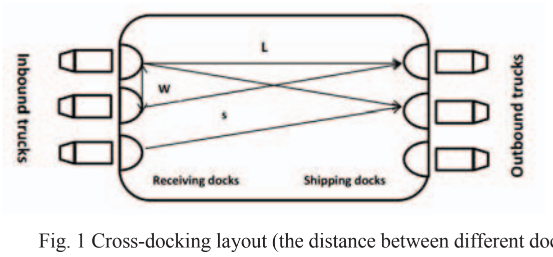


Fig. 1 Cross-docking layout (the distance between different docks)

The position $P_{out}[j]$ which the j-th outbound truck arrives at is determined in the process of simulation. Meanwhile, the i-th inbound truck will be arranged to arrive at the corresponding receiving docks $P_{in}[i]$ according to the schedule. $P_{in}[i]$ is calculated by the following function:

$$P_{in}[i]= S_{in}[i] mod(N) +1 \tag{1}$$

Therefore, the distance from the i-th inbound truck to the j-th outbound truck in docking system is represented by the following expression:

$$s_{[i][j]} = \sqrt{\left(P_{in}[i]-P_{out}[j]\right)^2 W^2 + L^2} \tag{2}$$

The notations of these two functions are described in Table I.

B. Earliness and Tardiness Function:

To send out the products as close to their due day (the day the customer wanted) as possible, we use the notations $E[j]$ (caused by earliness) and $T[j]$ (caused by tardiness) to represent the storage cost and back-order loss respectively. They are calculated by the functions below:

$$E[j]=\begin{cases}\sum_{i=1}^{I} C_1 \cdot G[i][j]\cdot(d[j]-l[j]) & if \quad l[j]<d[j] \\ 0 & else\end{cases} \tag{3}$$

$$T[j]=\begin{cases}\sum_{i=1}^{I} C_2 \cdot G[i][j]\cdot(l[j]-d[j]) & if \quad d[j]<l[j] \\ 0 & else\end{cases} \tag{4}$$

In the simulation, $d[j]$ is randomly generated and the leave time $l[j]$ is calculated by the function:

$$l[j]=\max\{\sigma_1,\sigma_2\} \tag{5}$$

where σ_1 represents the time outbound truck wait for the last product it needs and σ_2 represents the time when outbound truck arrives at the dock and pluses the total time to upload the product.

$$\sigma_1=\max\{k[i] \mid 1\le i\le I\} \tag{6}$$

In (6), $k[i]$ represents the time when the product in the inbound truck arrives and is evaluated by the following function:

$$k[i]=\begin{cases} t[i] & if \quad G[i][j]>0 \\ 0 & else\end{cases} \tag{7}$$

where $t[i]$ is the time when each inbound truck arrives:

$$t[i]=\frac{s_{in}[i]-1}{N}\times D_1 \tag{8}$$

σ_2 in (5) is calculated as followed:

$$\sigma_2=\begin{cases} l[j]+\sum_{i=1}^{I} G[i][j] & if \quad \sigma_1<l[j] \\ l[j] & else\end{cases} \tag{9}$$

Where $l[j]$ is:

$$l[j]=\begin{cases} l[j-1]+D_2+\sum_{i=1}^{I} G[i][j-1] & if \quad \sigma_1<l[j-1] \\ l[j-1]+D_2 & else\end{cases} \tag{10}$$

In (9), $l[j]$ is the time when the j-th outbound truck arrives at the dock and in (10) the changeover time of outbound trucks is D_2. If an outbound truck arrives at the dock before all the products it needs, the time of uploading the products onto this outbound truck can be neglected. Otherwise, the uploading time will be taken into consideration.

III. Scheduling Strategy

In [9], NSGA-II is demonstrated to be one of the most efficient algorithms for multi-objective optimization on several benchmark problems. Different from classical optimization methods that only find one solution in one simulation run, NSGA-II can find multiple optimal solutions in one simulation run. Such performance makes it more convenient to solve multi-objective optimization problems. The fitness function evaluation, the implement of NSGA-II algorithm on cross-docking systems and the scheme of local search are described in this section

A. General Description of NSGA-II

Some of the operators in NSGA-II are the same as those in GA. For example, in NSGA-II, the population undergoes initialization, crossover and mutation as usual. However there are three main differences: (1) each chromosome is sorted based on non-domination sorting into a front to obtain a fitness value; (2) crowding distance which represents the diversity in the population is employed to measure how close an individual is to its neighbors; (3) the population with the current population and current offspring (obtained by crossover and mutation) is sorted again based on the rank and the crowding distance. After that, the best N (population size) individuals are selected to be the next generation.

The procedure of NAGA-II is showed in Fig. 2.

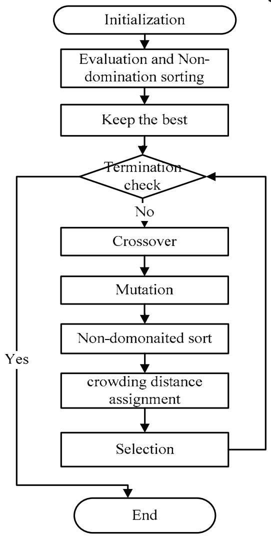


Fig. 2 The flowchart of NSGA-II

B. The Multi-objective Function

In the problem, the objectives are to find the minimal operation cost, earliness and tardiness. According to the formulations we discussed in section II, we establish two objective functions. The first function evaluates the punctuality by combining the earliness and tardiness. The function consists of two parts, one is the sum of the storage cost and the other is the sum of back-order loss. Therefore, the total function is:

$$F = \sum_{j=1}^{O} (E[j] + T[j]) \tag{11}$$

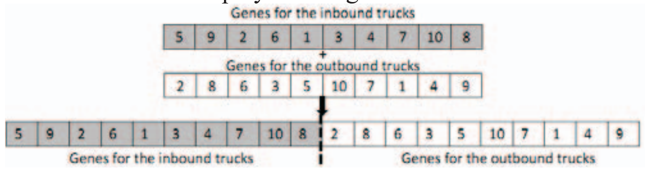
The second objective function is obtained by calculating the total operation cost.

$$S = \sum_{i=1}^{I} \sum_{j=1}^{O} G[i][j] \sqrt{(P_{in}[i] - P_{out}[j])^2 W^2 + L^2} \tag{12}$$

The total operation cost is represented by summing up all the distance that products go through from the inbound trucks to their destined outbound trucks.

C. Encoding and Decoding

To simulate cross-docking schedule, every truck schedule is modeled as a chromosome. The chromosome contains a schedule for both the inbound trucks and the outbound trucks. For example, a chromosome for ten inbound trucks and ten outbound trucks are displayed in Fig. 3.



Genes for the inbound trucks
5 9 2 6 1 3 4 7 10 8
+
Genes for the outbound trucks
2 8 6 3 5 10 7 1 4 9
5 9 2 6 1 3 4 7 10 8 2 8 6 3 5 10 7 1 4 9
Genes for the inbound trucks
Genes for the outbound trucks

Fig. 3 The encoding strategy for cross docking system problem

As shown in Fig. 3, we encode the inbound and outbound trucks separately with the integer numbers from 1 to N (the different number represents different truck). The first half represents the inbound trucks while the second half represents the outbound trucks. For example, Fig. 3 shows that the inbound trucks arrive in the order of 5,9,2,6,1,3,4,7,10,8. Furthermore, it should be noted that the number 5 in the first gene is different from the number 5 in the 15-th gene for they represent different trucks.

D. Non-domination Sorting

In NSGA-II, non-dominated sorting is used in the selection process. It obtains a Pareto-optimal set which is defined as a set of optimal solutions non-dominated by any feasible member of the search space. The pseudo-code of the non-domination sorting is displayed in Fig. 4.

```
Non-domination sorting
for each $p$ in $P$
  $S_P = \{ \}$
  $n_p = 0$
  for each $q$ in $P$                                   {Add it to the set of solution dominated
         if $(p<q)$ then $S_p=S_p \cup \{q\}$              by p}
         else if $(q<p)$ then $n_p=n_p+1$                  Increment the domination counter of p
  if $n_p=0$ then                                       p belongs to the first front
         $p_{rank}=1$
         $F_1=F_1 \cup \{p\}$
  $i=1$                                                 Initialize the counter for front
  while $F_i \neq \{\}$
         $Q=\{\}$
          for each $p$ in $F_i$                         Store the next front
               for each $q$ in $S_p$
                              $n_Q=n_q-1$
                              if $n_q=0$ then           It belongs to the next front
                  $q_{rank}=i+1$
                               $Q=Q \cup \{q\}$
         $i=i+1$
         $F_i=Q$
```

Fig. 4 The pseudo-code of Non-domination sorting process for NSGA-II

In the non-domination sorting process, three significant points needs to be taken into consideration: (1) it's possible to have several solutions belonging to the same level/rank; (2) the best set of solutions has rank 1, the second set has the rank 2 and so on. Each solution takes its rank as its fitness value. (3) If two solutions have the same rank, a crowding distance criterion is applied to measure the density of these two solutions. To maintain a better solution set, the key idea for this situation is to obtain an evenly distributed solution in the same rank. The pseudo-code of crowding distance is shown in Fig. 5.

crowding distance criterion

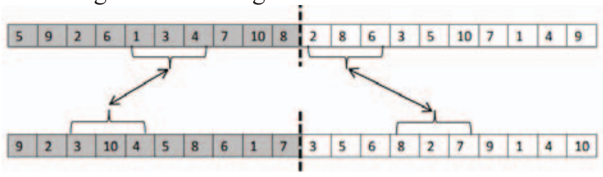
```
L=|T|                                   {Number of solution in T(non-
for each i in T                         dominated set) }
    $T[i]_{distance}=0$                 Initialize the distance of solution
for each objective S
    T=sort(T,S)                         Sort the non-dominated set
    $T[0]_{distance}=T[L]_{distance}=0$ Boundary points are selected
    for i=2 to (L-1)                    All the other points
        $T[i]_{distance}=T[i]_{distance}+(T[i+1].S-T[i-1].s)/(f_S^{max}-f_S^{min})$
```

Fig. 5 Algorithmic procedure of crowding distance criterion

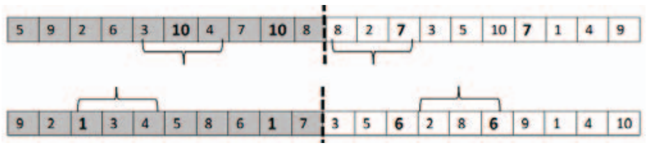
E. Crossover Scheme

The crossover operation has some constraints in this particular problem. For example, no duplicates in a chromosome are allowed after any kinds of operations. Also, the crossover operation must be performed separately on each half of the chromosome. This paper employs Partial-Mapped Crossover (PMX) which is first proposed by Goldberg and Lingle [10]. PMX is executed as the following steps:

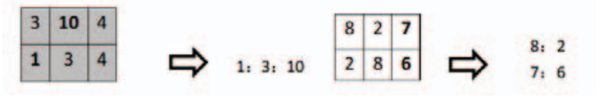
Step 1) Select two positions along the string uniformly at random and get the substrings.



Step 2) Exchange two substrings between parents to produce Pareto-children

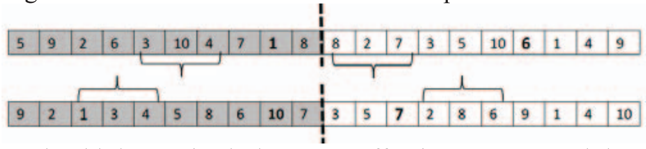


Step 3) Determine mapping relationship between the substrings



Step 4) Legalize offspring with mapping relationship

For example, 1 repeats in the second chromosome, then change 1 to 3 according to the map, if 3 is also repeated, change 3 to 10 and so on until there are no repeats.



It should be noticed that two offspring are created by exchanging the genes of the two selected individuals after a randomly selected locus but the first half and the second half executes the crossover separately. The procedure repeats until *N* new individuals are created.

F. Mutation Scheme

In this paper, interchange is applied for the mutation operator that means two points along the parent chromosome are randomly chosen to change with each other and genes in these two points will be reversed. As discussed in section E, the mutation has the same constraints as crossover. As Fig. 6 shows, the gene in different half can't exchange with each other, but the 3 and 10 in the same (first) half can exchange their positions.

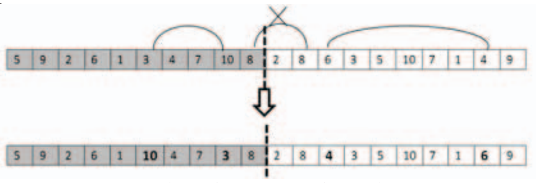


Fig. 6 the strategy of mutation

G. Local Search Scheme

To accelerate the convergence speed we employ the local search strategy by using a greedy algorithm designed for this specific problem. To minimize the earliness and tardiness, a number of N_1 successive genes in the first half and N_2 successive genes in the second half are chosen to execute the local search strategy. The inbound trucks with lager amount of products are supposed to arrive earlier than the others. Similarly, the outbound trucks whose due times are earlier are supposed to arrive earlier than others. For example, as Fig. 7 shows, 6, 1, 3 are the chosen inbound trucks and 1, 4, 9 are the chosen outbound trucks that undergo local search operation. Fig. 8 shows that the chosen inbound trucks have 40, 37 and 49 units of products respectively. According to our strategy that the inbound truck with larger number of product should come into the dock earlier, the gene segment is therefore changed into 3, 6, 1. Suppose that the outbound trucks due date is as shown in Fig. 9, the due date for the chosen outbound trucks are 620,382 and 752 respectively. For the outbound trucks that are set to leave earlier should arrive earlier, the gene segment is changed into 4, 1, 9.

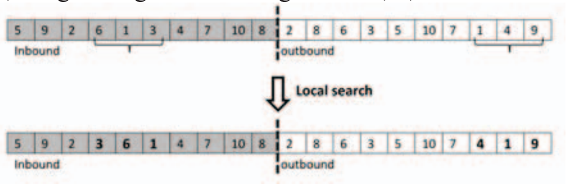


Fig. 7 the genes chosen to undergo local search

$$G=\begin{pmatrix} 0 & 0 & 3 & 16 & 0 & 7 & 0 & 0 & 11 & 0 \\ * & * & * & * & * & * & * & * & * & * \\ 15 & 0 & 0 & 8 & 0 & 15 & 0 & 0 & 0 & 11 \\ * & * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * & * \\ 0 & 19 & 0 & 12 & 0 & 0 & 9 & 0 & 0 & 0 \\ * & * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * & * \end{pmatrix} \begin{array}{l} \leftarrow \mathbf{37} \\ \\ \leftarrow \mathbf{49} \\ \\ \\ \leftarrow \mathbf{40} \\ \\ \\ \\ \end{array}$$

Fig. 8 the number of products transported from inbound trucks to outbound trucks

Outbound trucks' due-time
d[j]={**620** 324 298 **382** 357 232 130 296 **752** 306}

Fig. 9 the due date of each outbound truck

IV. Experiments and Discussion

As the search space is N!*N! (*N* is the number of inbound or outbound trucks) for an enumeration algorithm, time consuming becomes a severe burden as the number of trucks increases. So enumeration algorithm can only be carried out in small problem size for comparison experiment. Therefore, the results obtained by the propose method are compared with enumeration algorithm in small problem size in part C and compared with greedy algorithm in large problem size in part D. In the rest of this section, part A will introduce the way to generate the data that we used to simulate the process. And four major performance measurements that evaluate whether the results are good or not are discussed in part B.

A. Data Generation

To run the proposed algorithm, 10 problem sets were generated at random for each kind of experiments. For example, the number of products the inbound trucks bring and the destined outbound trucks each product will be sent to is randomly generated. But for being realitic, the due time of each outbound truck is generated according to the function (13).

$$d[j] = \frac{(s[i][j] \cdot \sum_{i=0}^{I} G[i][j] + D_1)}{random} \quad (1 < random < 1.5) \tag{13}$$

In this function, $s[i][j]$ is the distance that the products needs to travel from the *i*-th inbound truck to the *j*-th outbound truck which represents the flow cost. Meanwhile, $G[i][j]$ is the products in the *i*-th inound truck which the *j*-th outbound truck needs and D_1 is the changeover time. Therefore, the outbound trucks which require products with less flow time cost have higher possibility to leave earlier.

B. Performance Measurements

In this specific problem, we will have the non-dominated Pareto solution for each problem set. In order to evaluate the quality of each solution and compare the algorithm's performance, four performance measurements are employed because of their widely applications in [7] and [11].

1) MID (mean ideal distance): The closeness between Pareto solution and ideal point (0, 0).The function of MID is:

$$MID = \frac{\sum_{i=1}^{n} c_i}{n} \tag{14}$$

where $c_i = \sqrt{f_{1i}^2 + f_{2i}^2}$, f_{1i} (f_{2i}) means the value of the first (second) objective and *n* is the number of non-dominated set. In the function, c_i represents the distance to the ideal point. So the smaller value of MID means that the solution set is closer to the optimal solutions.

2) SNS: The spread of non-dominance solution:

$$SNS = \sqrt{\frac{\sum_{i=1}^{n} (MID - c_i)^2}{n-1}} \tag{15}$$

In (15) SNS represents the diversity of the solution set. The larger SNS is, the more diversity of the solutions and the better solution we obtains.

3) RAS: The rate of achievement to two objectives simultaneously

$$RAS = \frac{\sum_{i=1}^{n} \left(\frac{f_{1i} - F_i}{F_i} \right) + \left(\frac{f_{2i} - F_i}{F_i} \right)}{n}, F_i = \min\{f_{1i}, f_{2i}\} \tag{16}$$

RAS is based on the observance that suitable solutions full-fill the entire objectives concurrently. Therefore, having the least possible RAS values are our preferences.

4) ALC: Area under linear regression curves that fits to Pareto solution data with Excel Trendline function. Since the area is smaller when the solution set closer to the ideal point. The smaller values of ALC the better.

C. Case 1: Small Population Size

In this part, the experiment implements on the problem with small size. The number of the inbound (outbound) trucks is 6. Since the search space is not very large, we compare our results with the optimal solution obtained by using enumeration algorithm.

For fair comparison, the data generated above are the same in both NSGA-II and the enumeration algorithm. The population size of NSGA-II is set as 100 and terminates after 100 generations. Besides, crossover rate and mutation rate are set as 0.9 and 0.02 respectively.

We use the enumeration algorithm to find all possible solutions, totally there are 518400 (6!*6!). After that, we employ the non-domination sort to find all the solutions in rank 1 which represents a set of best solutions. With the set of optimal solutions, we then adapt four performance measurements to determine how close NSGA-II can proximate to them.

In our experiment, each of the problem set mentioned above is repeated 10 times. In each problem set, we obtain the minimum, average, and maximum values of the four measurements. After that, we calculate the mean values of the minimum, average and maximum values respectively to interpret the data.

Comparisons between two algorithms are shown in Table II. The values in the table are the percentage deviation of the minimal, average and maximal values calculated by the following function:

$$P = \frac{S_N - S_E}{S_E} \tag{17}$$

where *P* is the percentage deviation, S_N is the values of performance measured for NAGS-II and S_E is the values of performance measured for the optimal solution obtained by enumeration.

TABLE II Comparison between NSGA-II and Enumeration Algorithm using Four Performance Measurements

	MID	SNS	RAS	ALC
MIN	-3.40%	10.52%	-1.09%	-21.11%
AVERAGE	-0.50%	49.03%	0.69%	-5.95%
MAX	1.28%	113.26%	2.40%	16.53%

We can see that three of the measurement values are very close to the optimal measurement values, which reflects that NSGA-II can proximate to the optimal solution set. The reason that SNS of NAGA-II is much better than the optimal set is NSGA-II preserves the evenly distributed solutions in every generation.

Finally, in order to have a conceptual illustration of the non-dominated Pareto sets, Fig. 10 shows the best fronts which are obtained by NSGA-II and enumeration respectively in the first problem set. It shows that NSGA-II can find some of the best solutions and also the other solutions are close to the optimal ones.

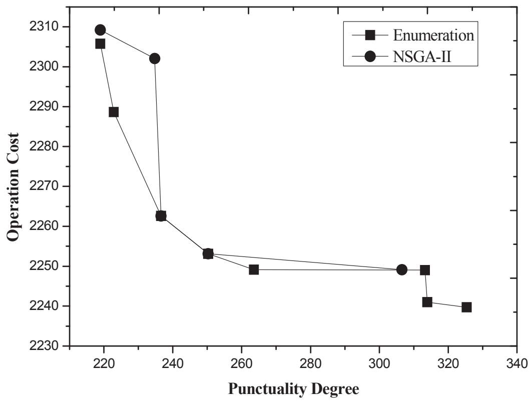


Fig. 10 Comparison between NSGA-II and Enumeration

The efficiency is obvious because the evaluation time of NSGA-II is only 10000 (100*100) and is only 19.3% of the searching space in the enumeration algorithm.

D. *Case 2: large Population Size*

For the large population size the number of the inbound (outbound) trucks is 21. Since the search space is too large for us to use the enumeration algorithm to find the optimal schedule, we employ a greedy algorithm which enables the inbound trucks with lager amount of products to arrive earlier than others. Similarly, the outbound trucks supposed to leave earlier are enabled to arrive earlier than others.

The data generated above are also the same in the both NSGA-II and greedy algorithms for fair comparison. The parameters of NSGA-II are the same as the previous experiment except for the generation is set as 1000. Fig. 11 shows the solutions NSGA-II obtained. All of them are better than the solution obtained by the greedy algorithm (the top-right point in the figure). The experiment results show that NSGA-II outperforms the greedy algorithm and finds the feasible solutions.

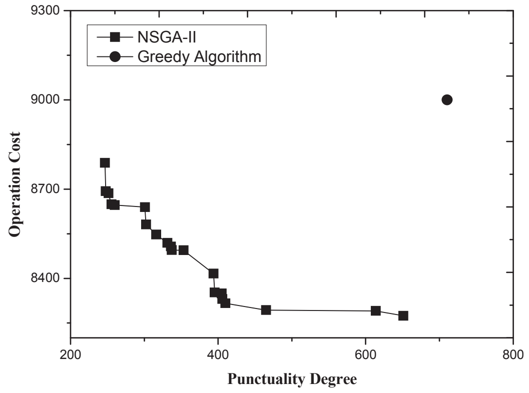


Fig. 11 Comparison between NSGA-II and Greedy Algorithm

V. Conclusion

This paper develops two objective functions considering punctuality delivery of products as well as the minimal operation cost that product needs to transport through the docking system. To solve this multi-objective problem, NSGA-II is employed for its robustness, accuracy and efficiency performance. At the same time, a local search strategy is designed for this particular problem to accelerate the convergence speed. Different from the traditional algorithm, NSGA-II can provide a set of solution instead of one proximate result. The solution can be selected according to different specific requirements. The result of the experiments shows that the proposed strategy outperforms the greedy strategy and approximates the optimal solution which is obtained by an enumeration algorithm. Furthermore, the proposed strategy is more efficient than the enumeration algorithm when the number of trucks increases.

References

[1] J. V. Belle, P. Valckenaers, D. Cattrysse "Cross-docking: State of the art," *Omega*, vol. 40, issue. 6, pp. 827-846, 2012

[2] W. Yu, "Operational Strategies for Cross Docking Systems", Iowa State University, Ph.D. Dissertation 2002.

[3] W. Yu, "Scheduling of inbound and outbound trucks in cross docking systems with temporary storage", *European Journal of Operational Research,* vol.184 , pp.377–396, 2008

[4] N. Boysen, "Truck scheduling at zero-inventory cross docking terminals". *Comput&Oper Res*, vol. 37, issue. 1, pp. 32–41, January 2010

[5] S. Ley, S. Elfayoumy, "Cross Dock Scheduling Using Genetic Algorithms" *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 416-420, June. 2007

[6] A. R. Boloori Arabani, S. M. T. Fatemi Ghomi, M. Zandieh, "A multi-criteria cross-docking scheduling with just-in-time approach," *Int J Adv Manuf Technol,*vol. 49, pp.741–756, 2010.

[7] A. R. Boloori Arabani, M. Zandiehb, S. M. T. Fatemi Ghomic, "Multi-objective genetic-based algorithms for a cross-docking scheduling problem," *Applied Soft Computing* vol.11, pp.4954–4970, 2011.

[8] N. Srinivas and K. Deb, "Multi-objective function optimization using nondominated sorting genetic algorithm," *Evol. Comput.*, vol. 2, no. 3,pp. 221–248, 1994.

[9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[10] David E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison–Wesley Publishing Company Inc, 1989

[11] J.Behnamian,S.M.T. Fatemi Ghomi,M. Zandieh, "A multi-phase covering Pareto-optimal front method to multi-objective scheduling in a realistic hybrid flowshop using a hybrid meta-heuristic", *Expert Systems with Applications* vol.36 ,no.8, pp.11057–11069, 2009.