Cross-docking centre operation optimization using simulation-based genetic algorithm

Y Wu[1], **M Dong**[1]*, and **D Yang**[2]
[1]Department of Operations Management, Antai College of Economics & Management, Shanghai Jiao Tong University, Shanghai, People's Republic of China
[2]College of Management, Donghua University, Shanghai, People's Republic of China

The manuscript was received on 17 May 2010 and was accepted after revision for publication on 11 October 2010.

DOI: 10.1177/2041297510393623

Abstract: The operational improvement of cross-docking centres in the literature involves job scheduling, layout design and dock assignment. The mentioned research is mainly concerned with makespan, number of tardy jobs or travel distance. However, these measures may not fully represent the main advantages of cross-docking operations. Little has been done to optimize the major performance measures of a cross-docking centre, which substantiates the main advantages of cross-docking technology. In this paper, inventory holding cost, transportation cost and backorder penalty cost are aggregated into total cost and a solution framework has been developed by integrating simulation, genetic algorithm (GA) and smart computing budget allocation (SCBA) for a comprehensive total cost minimization problem. This problem has huge search space even for medium-sized problem scenarios. To address this difficulty, the framework employs simulation to estimate the total cost, GA to search for better design and SCBA to efficiently allocate the simulation budget. Numerical experiments show that the proposed framework operates effectively in cases of different problem scales.

Keywords: cross-docking, simulation optimization, genetic algorithm

1 INTRODUCTION

Considered to be one of the most effective lean logistics strategies, cross docking is known for the great advantage of tremendous inventory cost reduction without sacrificing the customer service level. In addition, cost reduction in transportation is another major incentive as economies in transportation cost can be achieved by consolidation of different products intended for the same destination to full truck loads [**1**]. Some industries, in which logistics play a key role, have taken great advantages from this strategy. One of the most successful examples can be seen in the retail giant, Walmart [**2**]. Other companies which have benefited from using this strategy include express mail/package delivery company, UPS [**3**], automobile manufacturing leader, Toyota [**4**], and less than truckload (LTL) logistic providers [**5**].

Abundant academic research has started to emerge especially in recent years. However, the literature to date mainly focuses on either operation scheduling or design optimization of a cross-docking centre. Little research has been done to optimize the major performance measures of a cross-docking centre, which demonstrates the main advantages of cross-docking technology. The lack of optimization of comprehensive performance measures might lead to misjudgement of the impacts of cross docking. An optimal trade-off among inventory cost, transportation cost and customer service level serves as the motivation to fill this research gap. The complex dynamics of the

*Corresponding author: Department of Operations Management, Antai College of Economics & Management, Shanghai Jiao Tong University, 535 Fahua Zhen Road, Shanghai, 200052, People's Republic of China.
email: mdong@sjtu.edu.cn*

whole cross-docking system makes an analytical study extremely hard, if not impossible. Discrete event simulation is thus employed here to evaluate the performance measures of such systems. Along with high modelling flexibility, simulation brings a cumbersome computational burden. Furthermore, the huge solution search space could be another disaster. To tackle these challenges, a solution framework that integrates a genetic algorithm (GA) with the smart computing budget allocation (SCBA) method is proposed.

As more industries have adopted lean strategy nowadays, they implement a 'pull' strategy to manage their physical flows throughout the supply chain. However, this trend has not been fully discussed in related cross-docking research. To fill this gap, the cross-docking centres (CDC) under study in this paper are assumed to operate in a 'pull' strategy which rarely appears in existing literature. The details of the operational rules will be given in Section 3.

The rest of the paper is organized as follows. Section 2 presents a brief literature review on both cross docking and optimal computing budget allocation (OCBA). A description of the cross-docking centre operation problem is presented in Section 3. The solution framework which integrates simulation, GA and SCBA is developed in Section 4. Section 5 presents extensive numerical tests to verify the efficiency and effectiveness of the proposed method. Summary and conclusions are provided in Section 6.

2 LITERATURE REVIEW

Studies on cross docking have been proposed from different points of view for nearly two decades. They can be categorized into four main subjects, namely layout design of cross-docking centres, logistics networks design, inbound and outbound job scheduling, and the impacts of cross docking on the supply chain system performance. As comprehensive performance optimization is one of the major contributions of this research compared to the existing literature, attention will be paid to the performance measures and optimization methodologies used in the publications reviewed.

2.1 Layout design

This is also known as the truck dock assignment problem in the cross-docking environment. The labour cost, which is estimated by worker travel distance, is one of the main performance measures used in such a problem. Gue [**5**] compared two different scheduling policies for assigning incoming trailers to open docks based on this performance measure. Bartholdi and Gue [**6**] proposed a model to reduce labour cost by balancing the worker travelling distances and worker waiting time caused by congestions. Bartholdi and Gue [**7**] conducted computational experiments with respect to labour costs to determine the best shape for cross docks of various sizes. Lim *et al.* [**8**] and Miao *et al.* [**9**] formulated an integer programming model to describe the truck dock assignment problem with operational time and total capacity constraints. The objective of their model was to minimize the total cost, which comprises the operational cost and the penalty cost, where the operational cost is similar to the labour cost described above and the penalty cost will be incurred for all the unfulfilled shipments.

2.2 Network design

This particular topic focuses on the logistics or supply chain network design in which one or more cross-docking centres are involved. Sung and Song [**10**] and Sung and Yang [**11**] study a cross-docking supply chain network design problem. The cross-docking network consist of three phases including origin, intermediate (CDC) and destination phase. The setup cost of CDC and transportation cost incurred by moving items between phases are minimized by a Tabu search-based algorithm and branch-and-price algorithm, respectively. The fixed costs to open the CDC and costs to deliver products in a network comprising of a central manufacturing plant, multiple CDCs and retail stores were minimized by Ross and Jayaraman [**12**] using simulated annealing integrated with TABU search.

2.3 Job scheduling

Some studies treat the inbound and outbound job scheduling problems as machine scheduling problems. Li *et al.* [**13**] proposed such a scheduling model with time window constraints. The objective of minimizing the penalty cost of job earliness and tardiness was achieved by a genetic algorithm. Song and Chen [**14**] studied a two-stage cross-docking scheduling problem and minimized the makespan using proposed heuristics. A special form of the same problem in which only one machine exists in each stage was solved by a branch-and-bound algorithm in Chen *et al.* [**15**]. Boysen *et al.* [**16**] studied a similar problem with just one inbound and outbound dock. The inbound and outbound truck-scheduling problem is divided into two sub-problems, namely fix a particular inbound sequence and then find the optimal outbound sequence. A more general problem applied in the frozen food industry is presented by Boysen [**17**] where zero-inventory is an essential prerequisite. Yu and Egbelu [**18**] keep makespan as the objective although extending the problem with more detailed issues. The product assignment from inbound trucks to outbound trucks and the sequences of the inbound and outbound trucks are optimized simultaneously.

2.4 System performance

Little research has been done to optimize the major performance measures of a cross-docking centre. The impact of cross docking on retailers' system-wide inventory holding cost was examined by Waller *et al.* [**19**] while the same customer service level is kept in the retail stores.

As simulation optimization is employed as the solution framework, a brief review of this methodology is given below. A general framework can be defined as

$$\min_{\theta \in \Theta} J(\theta) \equiv E[L(\theta, \varepsilon)] \tag{1}$$

where $J(\theta)$ is the performance measure of the studied system, $L(\theta, \varepsilon)$ is the sample performance, ε represents the stochastic effects in the system, θ is a vector of decision variables and Θ is the set of all feasible solutions.

The above simulation optimization problem has been extensively studied in recent decades. Several review papers are available for this particular area [**20–23**].

A relatively new approach, called optimal computing budget allocation (OCBA), is proposed and shown to be timely efficient with higher accuracy by Chen and Lee [**24**]. However, this method evaluates all feasible solutions and allocates fixed number of additional simulation runs to these solutions step-by-step based on the present performance results obtained so far. As the alternative solution number becomes really huge, this method is no longer practical. Thus, the key idea of allocating the simulation budget more efficiently is integrated with meta-heuristics to tackle the problem. The essential of meta-heuristics is to balance the trade-off between intensification and diversification; herein accuracy of the performance evaluations for different solutions is added, which will lead to a more sophisticated control strategy.

This solution framework has been successfully implemented to solve several real life problems recently, such as adjusting nurses' schedules in hospital emergency departments to improve service quality [**25**], improving robustness of the flight schedule [**26**] and allocating aircraft spare parts [**27**]. In this paper, a novel solution framework, which integrates GA and SCBA method, is proposed to optimize the operational control of the cross-docking centre (CDC).

While the importance of the cross-docking distribution has been recognized, little research has been done for cross-docking performance optimization. We present a GA-based simulation optimization approach with embedded SCBA to optimizing the cross-docking performance in a supply chain network.

3 PROBLEM DESCRIPTION

The CDC may serve multiple types of customers with multiple types of products. Let $P = \{1, 2, \ldots, m\}$ denote the set of all product type indices. For any subset of product $K \subseteq P$, we say an order is of type K if it consists of one unit of each product in K and zero units in $P \backslash K$. Customer orders of type K arrive at the system following a stationary Poisson process, denoted as $\{A_K(t),\ t \geq 0\}$, with rate λ_K. It is assumed that each order's type is independent of the other orders' types and of all other events. $D = \{1, 2, \ldots, n\}$ is used to denote the set of all order types.

For each product i, let D_i denote the family of subsets of D that contains i. Then it is clear that the demand process for product i is also a Poisson process with rate:

$$\lambda_i = \sum_{K \in D_i} \lambda_K \tag{2}$$

Upon the arrival of any customer order, the customer order is transformed into several product orders according to the subset of products this customer order requires. As CDC carries no inventory ahead, a replenishment order for a specific product type will not be placed to the supplier until a certain amount (order point, x_i) of backorders for this product type is accumulated. The unfilled order is backlogged and is charged at a penalty cost rate of b_K for order type K. Sufficient transportation capacity, i.e. an unlimited number of trucks, is assumed thus once the suppliers receive the replenishment order, the amount of products required will be delivered to the CDC in a truck. The load volume of each truck is considered and the maximum load number of each product type is assumed to be $x_i^{\max}$. The transportation cost consists of constant cost, C_i (for product i) and variable cost related to the product amount. As the variable cost corresponds with only total replenish amount, it does not vary when different order points are chosen, i.e. how many deliveries have to be made, thus only constant transportation cost is included in total cost. Orders are filled when all the products that they require arrive, on

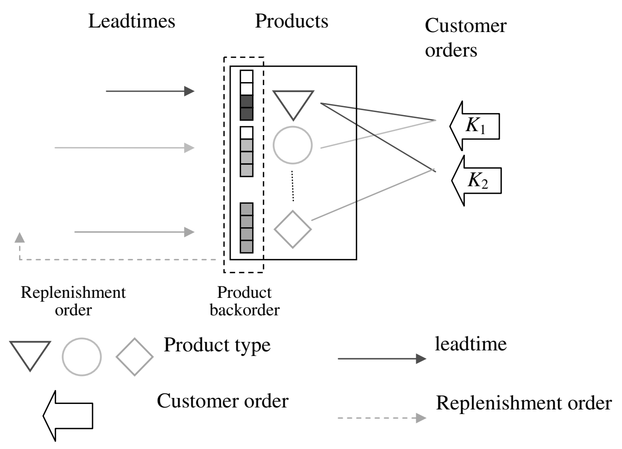

Fig. 1 Cross-docking centre operation

a first-come-first-served basis. Figure 1 gives an illustration of the operation. Different shapes, such as an inverted triangle, circle and rhombus, represent different types of products. Each order of order type K_1 requires a product of both types of inverted triangle and circle. Each order of order type K_2 requires a product of both types of inverted triangle and rhombus.

As the product demand rates, λ_i, and lead-times, l_i, may differ in product types, items filling one customer order may not arrive simultaneously. The items arriving earlier are stored in the CDC until all the remaining items required arrive, and the stored items make up the inventory. Inventory holding cost rate is h_i per item for product type i. Moreover, the product unloading and loading time are assumed to be constant and could be negligible. Thus, the order is immediately satisfied once the trucks arrive at the CDC. The notations are given in Appendix 1.

The objective of this problem is to determine the order point for each product type to minimize the average total cost that includes inventory holding cost, transportation cost and backorder penalty cost. The mathematical model for the problem can be described as below.

$$\underset{x_i \le x_i^{\max}, \forall i \in P}{\mathrm{MIN}} \sum_{i=1}^{m} h_i E(I_i) + \sum_{i=1}^{m} C_i S_i + \sum_{K \in D} b_K E(B_K) \qquad (3)$$

where $E(I_i)$ and $E(B_K)$ in equation (3) are average on-hand inventory level of product i and average backorder level of order type K, respectively. Unfortunately, no analytical solution is available for the performance measures of such a complex system and simulation is thus employed to estimate the system performance that is being transformed into average total cost. All the operation details and constraints are integrated in the simulation model.

4 A SOLUTION FRAMEWORK INTEGRATING GA AND SCBA

The solution framework mainly constitutes of three parts, which are simulation, GA and SCBA, respectively. As no analytic solution for such a system is available, simulation is employed to evaluate average on-hand inventory level and average backorder level in equation (3). However, as simulation can only evaluate the performances of given design alternatives, it lacks the ability to explore more promising solutions in the search space. Thus, a GA integrated with a newly developed SCBA is proposed to tackle this problem. GA has been shown to be efficient when integrated with computing budget allocation [**26**, **27**]. Generally, GA explores the unsearched space by generating new solutions whereas SCBA allocates more computing budget to the more promising solutions. The work progress of the whole framework is shown in Fig. 2.

4.1 Coding scheme

Each chromosome, which represents a set of order points of all types of product, is represented as follows $X = (x_1, x_2, \ldots, x_m)$, where m is the number of product type. Each gene in a chromosome, x_i, represents the order point of a specific product type i, $i \in P$.

4.2 Initialization

The population size is chosen to be 100, which is widely adopted in the literature and the genetic algorithm starts the search by randomly generating a population of candidate solutions. Each gene, x_i, is generated from an integer uniform distribution in the interval of $[1, x_i^{\max}]$.

4.3 Chromosome evaluation

A computer program is developed to evaluate different order point settings. The program will generate the performance measures, $E(I_i)$, S_i and $E(B_K)$, and calculate the average total cost which is then used to calculate the fitness value for the setting.

4.4 Fitness function

A simple fitness function (equation (4)) is adopted. The main shortcoming of this function will be overcome by the child generation strategy, thus it is hired

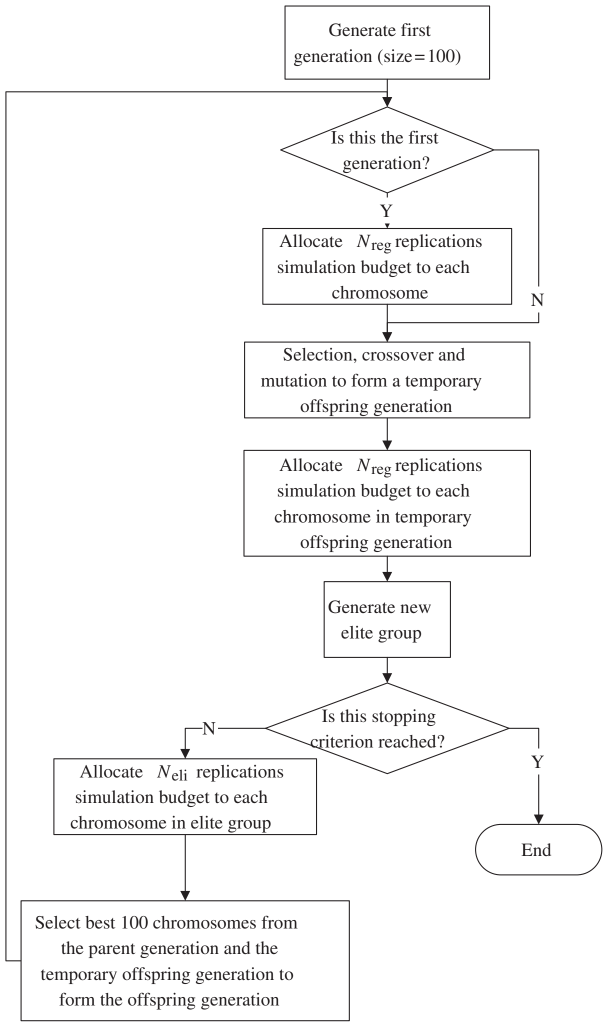
Generate first generation (size = 100)
Is this the first generation?
Y
N
Allocate N_{reg} replications simulation budget to each chromosome
Selection, crossover and mutation to form a temporary offspring generation
Allocate N_{reg} replications simulation budget to each chromosome in temporary offspring generation
Generate new elite group
Is this stopping criterion reached?
N
Y
Allocate N_{eli} replications simulation budget to each chromosome in elite group
End
Select best 100 chromosomes from the parent generation and the temporary offspring generation to form the offspring generation

Fig. 2 Flow chart of the solution framework

for its simplicity. The lower the average total cost the higher the fitness value.

$$fitness(X) = \frac{1}{E(TC)} \tag{4}$$

where $E(TC) = \sum_{i=1}^{m} h_i E(I_i) + \sum_{i=1}^{m} C_i S_i + \sum_{K \in D} b_K \times E(B_K)$ is the average total cost for setting X.

4.5 Parent selection

The Roulette wheel parent selection is used. The chance of each chromosome being selected is directly proportional to its fitness value.

4.6 Crossover operator

The string-of-change crossover, which is a modified single point crossover, is used here. This operator will pick the crossover point from all genes equally except those points, which may result in no change between parents and offspring. This will slow down the convergence and diminish the searched settings.

4.7 Mutation operator

Mutation is randomly applied to each of the genes in a new chromosome. If a gene is picked to be mutated, its value will either go up or go down by 1 with equal probability of 0.5. Two exceptions are if the value of gene equals to 1, it will increase by 1 with probability 1, or the value of gene equals to $x_i^{\max}$, it will decrease by 1 with probability 1.

4.8 Formation of child population

After generating an offspring population, chromosomes from both offspring and parent population will be ranked and the best 100 will be placed in the child population, which will in turn be the parent population for the next generation. By selecting the best 100 chromosomes, the algorithm ensures that the best members of the population will be presents in the next generation. For some application examples, this approach might cause the problems of premature convergence. However, in this case, since the evaluation of the chromosomes is performed through simulation, the problem will be greatly mitigated since the simulation generates estimations for the performance evaluation instead of the exact values. The premature convergence could be avoided due to this randomness.

4.9 SCBA

The key idea of SCBA is to allocate more of the simulation budget on the critical designs rather than waste it on non-critical designs. Chen *et al.* [**15**] first implemented this idea by allocating several pilot simulation replications to all the alternative designs and then gradually increased the computing budget, which was allocated according to the current values obtained from allocated computing budget. This framework works perfectly in the case with relatively small design space as the example used in Chen *et al.* [**15**]. However, this strategy is not applicable when a huge number of alternative designs need to be evaluated. Thus, a smart computing budget allocation strategy is developed here to tackle such a problem.

The idea is still to allocate more simulation replications to potentially good designs, thus an elite group, the members in which will receive more simulation replications, is proposed to keep the best several results obtained so far. Since GA is employed as the search technique, the optimal solution is not expected to be found within the random first population in the early stage, especially in cases with huge search space. Therefore, the replication number for the elite group increases a certain amount with generation. With more replications, the designs in the elite group will have more accurate performance estimations. Those new designs which were generated at each iteration have to prove themselves by running more replications of simulations in order to replace the existing members in the elite group.

4.10 Termination rule

In order to illustrate the effectiveness and efficiency of the proposed solution framework, the results from the proposed framework will be compared to that from the exhaustive method in the small size case and to that from random search methods and a regular genetic algorithm in medium and large size cases. To make different algorithms comparable, it is of interest to see which algorithm offers the best results when given the same number of simulation replications. This means the computation times will be roughly the same. Therefore, running out of total replication number will be adopted as the termination rule for all the algorithms in this paper except the exhaustive method. In other words, the algorithms will stop when a maximum amount of time has been spent.

4.11 GA integrated with SCBA

Step 0: Initialization: Randomly generate an initial population of size N_{pop}; set elite group $G_{\text{eli}} = \varnothing$, assign total replication number $R_{\text{total}} = M$ (M could be any positive integer, e.g. 50 000).

Step 1: Evaluation: Estimate the performance of each design in the parent population by running R_{pop} replications of simulation, set $R_{\text{total}} = R_{\text{total}} - N_{\text{pop}} * R_{\text{pop}}$.

Step 2: Fitness assignment: Assign a fitness value using equation (4) to each of the designs in the parent population.

Step 3: Initial elite group formation: Select top N_{eli} designs from the parent population according to the estimations of their performance; set the current total elite simulation replication number $TR_{\text{eli}} = R_{\text{pop}}$, set the performance threshold to be the estimation of the worst design in the elite group.

Step 4: Parent selection: Select a pair of parents using the Roulette wheel parent selection method.

Step 5: Crossover: Perform string-of-change crossover to generate a pair of offspring.

Step 6: Mutation: For each offspring, apply mutation to the genes.

Step 7: Checking: Check if the number of offspring is equal to N_{pop}. If there is not a sufficient number of offspring, go to Step 4, else, proceed to Step 8.

Step 8: Evaluation: Estimate the performance of each offspring by running R_{pop} replications of simulation, set $R_{\text{total}} = R_{\text{total}} - N_{\text{pop}} * R_{\text{pop}}$.

Step 9: Challenging elite group: Select all the offspring (say the number is N_{cha}), whose estimated performances are better than the elite performance threshold, allocate $TR_{\text{eli}} - R_{\text{pop}}$ replications of simulation to each of these designs and if the estimated performance is still better than the elite performance threshold then replace the worst design in elite group with the current new design, $R_{\text{total}} = R_{\text{total}} - N_{\text{cha}} * (TR_{\text{eli}} - R_{\text{pop}})$. Check if $TR_{\text{eli}} < TR_{\text{eli}}^{\max}$ ($TR_{\text{eli}}^{\max}$ is a predefined number that the elite group should be replicated at last), if so, allocate R_{eli} replications of simulation to each design in the new elite group and set $TR_{\text{eli}} = TR_{\text{eli}} + R_{\text{eli}}$ and $R_{\text{total}} = R_{\text{total}} - N_{\text{eli}} * R_{\text{eli}}$, then update the elite performance threshold with the estimated performance of the new worst design in elite group.

Step 10: Formation of child population: The newly generated offspring and the parent population are combined together and the best N_{pop} are then selected. These designs make up the child population.

Step 11: Termination: If $R_{\text{total}} \leq 0$, end the algorithm. Otherwise, replace the parent population by the child population and return to Step 4.

5 NUMERICAL EXPERIMENTS

The numerical results are presented in this section to verify the efficiency of the proposed solution framework by comparing it to exhaustive method in small size case and to random search method and regular genetic algorithm in medium and large size cases, respectively. All the algorithms and simulation were tested on an Intel Core 2 Duo computer with CPU 2.33 GHz and RAM 2048 MB. The following sub-section describes the three benchmark methods.

5.1 Other approaches

5.1.1 Exhaustive method

This method evaluates all the feasible designs with same amount of simulation replications and generates the optimal design based on the estimated performance. As problem size grows, the alternative space will grow exponentially. Thus, this method is only practical in very small cases.

5.1.2 Random search method

The random search method works as follows:

Step 0: Randomly generate a feasible design $X_0 = \left(x_1^0, x_2^0, \ldots, x_m^0\right)$ and evaluate this design with R replications of simulation, set the current design to be the best design and the minimum total cost $TC_{\min}$ to be the estimated total cost obtained from simulation.

Step 1: Generate new design: Choose x_i^t from all the order points with equal probability,

$$\text{If } x_i^t = 1,\ x_i^{t+1} = x_i^t + 1;$$

$$\text{If } x_i^t = x_i^{\max},\ x_i^{t+1} = x_i^t - 1;$$

$$\text{If } 1 < x_i^t < x_i^{\max},\ x_i^{t+1} = \begin{cases} x_i^t + 1, \\ x_i^t - 1, \end{cases} \text{with probability of } 0.5.$$

Set $x_j^{t+1} = x_j^t$, where $j \in P$, $j \neq i$.

Step 2: Evaluate new design: Evaluate the new design with R replications of simulation, if the estimated total cost $TC_t < TC_{\min}$, set $TC_{\min} = TC_t$ and the best design to be X_t.

Step 3: Check whether or not the termination condition is satisfied. If yes, end algorithm. Otherwise, go to step 1.

5.1.3 Regular genetic algorithm

The regular genetic algorithm is the same as that described in Section 4, except no SCBA is embedded, which means each design will be allocated a predefined amount of simulation replications.

5.2 Parameters setting

5.2.1 Problem parameters

This problem was studied in three different sizes: small, medium and large sizes, respectively. The numbers of product types and order types are given in Table 1.

The detailed parameters, such as order arrival rate, backorder cost rate, maximum load number, constant transportation cost, replenish lead-time and inventory holding cost rate, are presented in Appendix 2.

Table 1 Problem scale parameters

Size	Product type	Order type
Small	4	3
Medium	40	40
Large	100	60

5.2.2 Simulation parameters

The main parameters to be determined here were the warm-up length and the total simulation length. In the present study, the replication/deletion approach was applied when generating simulation outputs for the performance measures of each design. The order waiting time and product inter-arrival time of randomly chosen type were monitored in both small and medium size cases. From Figs 3–5, it can be seen that the system reached stability very fast; this is because of the sufficient transportation capacity assumption. Therefore, a relatively short warm-up length (1 day) was chosen and the total simulation length was fixed to be a week (7 days).

5.2.3 Solution framework parameters

The population size was set to be 100 ($N_{\text{pop}} = 100$) and the elite group size was set to be 20 ($N_{\text{eli}} = 20$). For each design in newly generated population, $R_{\text{pop}} = 2$ replications were assigned. $R_{\text{eli}} = 2$ replications of simulation were assigned to each design in the elite group every generation until $TR_{\text{eli}}^{\max} = 50$ replications was reached. In all the other methods (exhaustive, random search and regular GA), 50 replications were assigned to each selected design. All the parameters mentioned above were obtained through some trial numerical experiments in which different levels of parameters were tested and those parameters which generate less overall total cost were chosen.

5.3 Numerical results

The solution framework was tested using different cases of small, medium and large sizes, respectively. In small size cases, the solution framework was compared with the exhaustive method and the results are listed in Tables 2 and 3. Deviations were the optimal total cost value gaps between the exhaustive method and the proposed solution framework (represented by SCBA).

$$\text{Dev} = \frac{|\text{Total cost}_{\text{SCBA}} - \text{Total cost}_{\text{exhaustive}}|}{\text{Total cost}_{\text{exhaustive}}} \cdot 100\%$$

In medium and large size problems, the comparisons among the proposed solution framework, random search method and regular GA are shown in Figs 6 and 7. Total replications of 50 000 runs were assigned in each method, thus the computation times for different methods were roughly the same, which were about 3–4 h for the medium

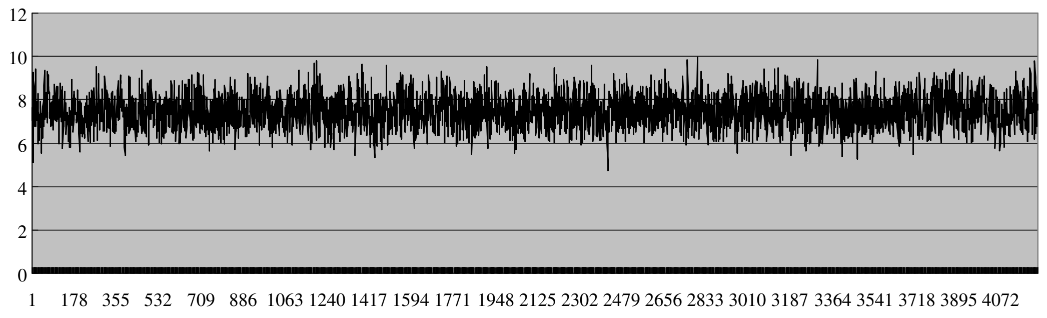

Fig. 3 Order delay time from 50 independent replications in small size case

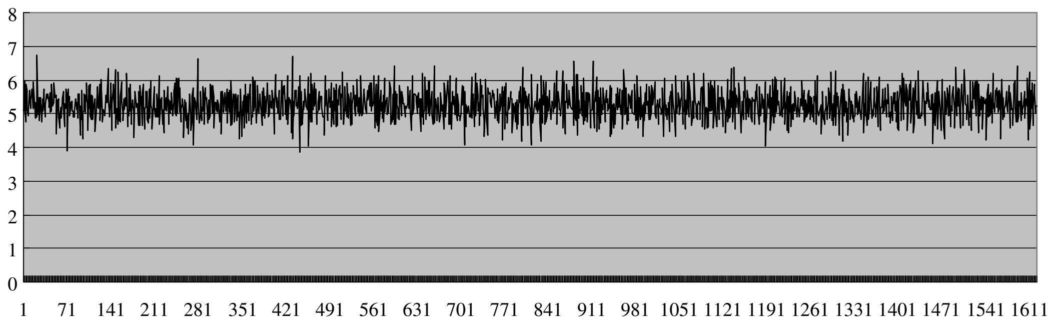

Fig. 4 Product inter-arrival time from 50 independent replications in small size case

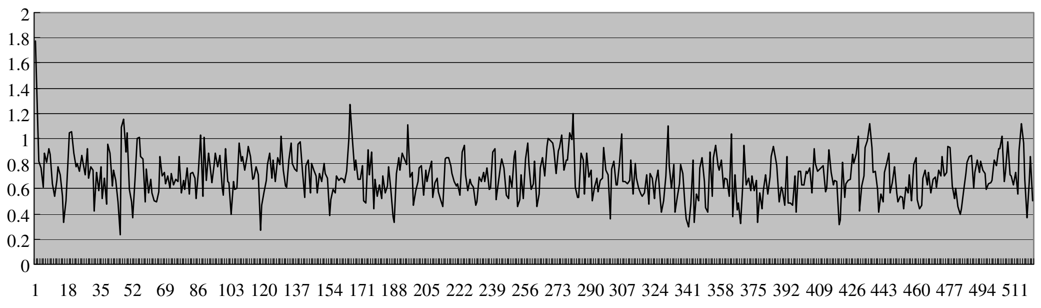

Fig. 5 Order delay time from 50 independent replications in medium size case

Table 2 Small-size case 1

Method	Exhaustive	SCBA	Dev
Unit time total cost	26.22	27.86	6.25%
		26.58	1.37%
		26.36	0.53%
		27.06	3.20%
		27.75	5.84%
Compute time	700 s	8 s	

Table 3 Small-size case 2

Method	Exhaustive	SCBA	Dev
Unit time total cost	28.30	28.42	0.42%
		30.87	9.08%
		28.98	2.40%
		30.24	6.86%
		28.89	2.08%
Compute time	800 s	8 s	

size problem and 60 h for the large size problem. It is quite obvious from Figs 6 and 7 that the proposed method outperforms the other two methods for its lower total cost and performance variance.

To show the better convergence of the proposed solution framework, experiments with total replication of 2 000 000 runs were carried out for the medium size problem, the optimization process is recorded in Fig. 8. Clearly, SCBA converges much faster than the regular GA. Each instance of these experiments took about 80 h.

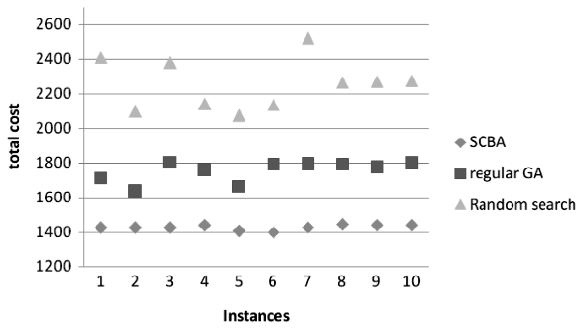

Fig. 6 Results of the medium size problem

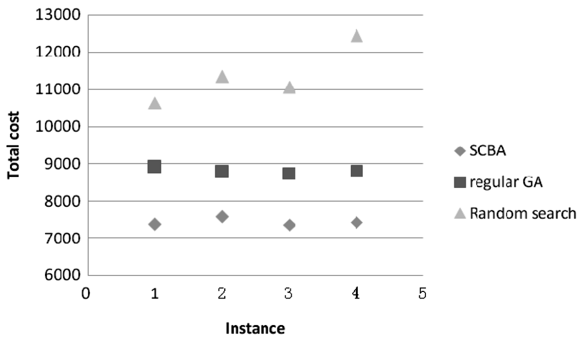

Fig. 7 Results of the large size problem

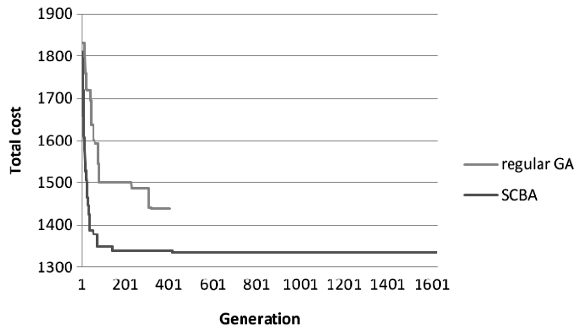

Fig. 8 Convergence comparisons

6 CONCLUSIONS

This investigation studied the performance optimization of cross-docking centre operations, which is widely recognized as an efficient technology with great potential to reduce the inventory and transportation cost in distribution networks. The system is modelled with a focus on the product order point optimization, which plays a key part in the daily operations. Unfortunately, no exact method is available for the performance evaluation for such complex operations especially when there exists multiple product types and order types. Thus, simulation is employed to estimate the system performance from which the total cost can be obtained. A novel solution framework integrating simulation, genetic algorithm (GA) and smart computing budget allocation (SCBA) is proposed to search for the optimal order point for each product type. Numerical results show that the proposed solution framework works well in all cases including small, medium and large size problems. In the small size cases, the differences between the results obtained from the proposed solution framework and exhaustive method were within 10%; however, the new approach only takes about 1% of the computation time that the exhaustive method requires. In the medium and large size cases, the framework also outperformed the regular genetic algorithm and random search method when roughly same amount of time was assigned to each method. In a future study, instead of studying the total cost, multiple performance measures, which may conflict with each other will be treated separately. In that case, a multi-objective optimization approach should be integrated into the framework in this paper.

ACKNOWLEDGEMENTS

The work presented in this paper has been supported by the grants from National Natural Science Foundation of China (70571050 and 70871077) and National High Technology Research and Development Program ('863' Program) of China (2008AA04Z104).

REFERENCES

1 **Apte, U. M.** and **Viswanathan, S.** Effective cross docking for improving distribution efficiencies. *Int. J. Logistics*, 2000, **3**, 91–302.
2 **Stalk, G., Evans, P.,** and **Shulman, L. E.** Competing on capabilities: the new role of corporate strategy. *Harvard Business Rev.*, 1992, **70**(2), 57–69.
3 **Forger, G.** UPS starts world's premiere cross-docking operation. *Modern Mater. Handling*, 1995, **50**, 36–38.
4 **Witt, C. E.** Crossdocking: Concepts demand choice. *Mater. Handling Engng.*, 1998, **53**(7), 44–49.
5 **Gue, K. R.** The effect of trailer scheduling on the layout of freight terminals. *Transportation Sci.*, 1999, **33**, 419–428.
6 **Bartholdi III, J. J.** and **Gue, K. R.** Reducing labor costs in an LTL crossdocking terminal. *Operat. Res.*, 2000, **48**(6), 823–832.
7 **Bartholdi III, J. J.** and **Gue, K. R.** The best shape for a crossdock. *Transportation Sci.*, 2004, **38**(2), 235–244.
8 **Lim, A., Ma, H.,** and **Miao, Z.** Truck dock assignment problem with time windows and capacity constraint in transshipment network through crossdocks. *Lecture Notes Computer Sci.*, 2006, **3982**, 688–697.

9 **Miao, Z., Lim, A.,** and **Ma, H.** Truck dock assignment problem with operational time constraint within crossdocks. *Eur. J. Operat. Res.*, 2009, **192**, 105–115.

10 **Sung, C. S.** and **Song, S. H.** Integrated service network design for a cross-docking supply chain network. *J. Operat. Res. Soc.*, 2003, **54**, 1283–1295.

11 **Sung, C. S.** and **Yang, W.** An exact algorithm for a cross-docking supply chain network design problem. *J. Operat. Res. Soc.*, 2008, **59**(1), 119–136.

12 **Ross, A.** and **Jayaraman, V.** An evaluation of new heuristics for the location of cross-docks distribution centers in supply chain network design. *Computers Industr. Engng.*, 2008, **55**, 64–79.

13 **Li, Y., Lim, A.,** and **Rodrigues, B.** Crossdocking – JIT scheduling with time windows. *J. Operat. Res. Soc.*, 2004, **55**, 1342–1351.

14 **Song, K.** and **Chen, F.** Scheduling cross docking logistics optimization problem with multiple inbound vehicles and one outbound vehicle, Proceedings of the IEEE International Conference on Automation and Logistics, *ICAL 2007*, 2007, pp. 3089–3094.

15 **Chen, F.** and **Lee, C. Y.** Minimizing the makespan in a two-machine cross-docking flow shop problem. *Eur. J. Operat. Res.*, 2009, **193**, 59–72.

16 **Boysen, N., Fliedner, M.,** and **Scholl, A.** Scheduling inbound and outbound trucks at cross docking terminals. *OR Spectrum*, 2009, **32**(1), 135–161.

17 **Boysen, N.** Truck scheduling at zero-inventory cross docking terminals. *Computers Operat. Res.*, 2010, **37**(1), 32–41.

18 **Yu, W.** and **Egbelu, P. J.** Scheduling of inbound and outbound trucks in cross docking systems with temporary storage. *Eur. J. Operat. Res.*, 2008, **184**, 377–396.

19 **Waller, M. A., Cassady, C. R.,** and **Ozment, J.** Impact of cross-docking on inventory in a decentralized retail supply chain. *Transportation Res. Part E*, 2006, **42**, 359–382.

20 **Fu, M. C.** Optimization via simulation: A review. *Ann. Operat. Res.*, 1994, **53**, 199–247.

21 **Fu, M. C.** Optimization for simulation: theory vs. practice. *INFORMS J. Comput.*, 2002, **14**(3), 192–215.

22 **Swisher, J. R., Hyden, P. D., Jacobson, S. H.,** and **Schruben, L. W.** A survey of recent advances in discrete input parameter discrete-event simulation optimization. *IIE Trans.*, 2004, **36**, 591–600.

23 **Tekin, E.** and **Sabuncuoglu, I.** Simulation optimization: A comprehensive review on theory and applications. *IIE Trans.*, 2004, **36**, 1067–1081.

24 **Chen, C. H., Lin, J. W., Yucesan, E.,** and **Chick, S. E.** Simulation budget allocation for further enhancing the efficiency of ordinal optimization. *Discrete Event Dynam. Syst.: Theory Applic.*, 2000, **10**, 251–270.

25 **Yeh, J. Y.** and **Lin, W. S.** Using simulation technique and genetic algorithm to improve the quality care of a hospital emergency department. *Expert Syst. Applic.*, 2007, **32**, 1073–1083.

26 **Lee, L. H., Lee, C. U.,** and **Tan, Y. P.** A multi-objective genetic algorithm for robust flight scheduling using simulation. *Eur. J. Operat. Res.*, 2007, **177**, 1948–1968.

27 **Lee, L. H., Chew, E. P., Teng, S.,** and **Chen, Y.** Multi-objective simulation-based evolutionary algorithm for the aircraft spare parts allocation problem. *Eur. J. Operat. Res.*, 2008, **189**, 476–491.

APPENDIX 1

Notation

m	number of product types
P	set of all product indices
N	number of order types
D	set of all order types
λ_K	arrival rate of order type K, $K \in D$
b_K	backorder cost rate of order type K, $K \in D$
B_K	backorder level of order type K, $K \in D$
$x_i^{\max}$	maximum load number for product i, $i \in P$
λ_{i}	demand rate for product i, $i \in P$
C_i	constant transportation cost of product i, $i \in P$
l_i	replenish lead-time for product i, $i \in P$
h_i	inventory holding cost rate of product i. $i \in P$
I_i	on-hand inventory level of product i, $i \in P$
S_i	replenish time rate of product i, $i \in P$

Decision variable

x_i	order points of products i, $i \in P$

APPENDIX 2

All the parameters were randomly generated with practical use. All the order inter-arrival times and replenishment lead-times were exponentially distributed. The mean values of the order inter-arrival times were generated from an uniform distribution with an interval [2, 4], the backorder cost rates were uniformly generated from an interval [2, 3], constant transportation costs were uniformly generated from an interval [15, 25], replenishment lead-times were uniformly generated from an interval [1, 3], inventory holding cost rates were uniformly generated from an interval [0, 2], and the maximum load numbers were

Table A1 Product parameters for small-size case 1

Product type	h	C	$E[l]$ (h)	$x_i^{\max}$
1	1	20	2	10
2	1	20	2	10
3	1	20	2	10
4	1	20	2	10

Table A2 Order parameters for small-size case 1

Order type	$1/\lambda(h)$	b
1	2.5	2
2	4	2
3	2	2

Table A3 Order-product matrix for small-size case 1

	Order 1	Order 2	Order 3
Product 1	1	0	1
Product 2	1	1	0
Product 3	1	1	0
Product 4	0	1	1

Table A4 Product parameters for small-size case 2

Product type	h	C	$E[l]$ (h)	x_i^{max}
1	3	20	2	8
2	2	20	2.5	10
3	1	20	2.5	15
4	1.5	20	3	12

generated from a discrete uniform distribution with an interval [6, 15]. The detailed parameters of three scales are presented in Tables A1 to A9 (all the notations are consistent to those in Section 3).

The order-product matrix contains the information about what product types a certain order type requires. In Table A3 and A7, rows in the matrix represent product types and columns represent order types. For example, number 1 in row 3 and column 2 means an item of product type 3 is required by each order type 2.

Small size case 2 has the same order parameters and same order-product matrix as case 1. The product parameters are listed in Table A4.

The order–product matrix for large size problem is omitted for the sake of space.

Table A5 Product parameters for medium-size problem

Product type	h	C	$E[l]$	x_i^{max}	Product type	h	C	$E[l]$	x_i^{max}
1	0.764	20.58	1.707	8	21	0.744	15.57	1.015	14
2	0.201	19.00	2.722	11	22	0.711	21.25	2.434	7
3	1.193	21.28	2.286	7	23	1.821	22.19	1.674	11
4	1.798	17.79	1.434	7	24	0.932	24.06	1.429	10
5	1.769	15.44	2.471	14	25	0.852	23.41	2.847	11
6	1.917	24.14	2.924	14	26	0.608	18.33	2.217	14
7	0.029	18.48	2.862	15	27	1.951	16.22	1.209	11
8	0.815	23.09	2.177	7	28	1.613	15.10	1.994	6
9	1.726	24.53	2.766	12	29	1.982	24.44	2.137	10
10	0.277	15.01	2.635	14	30	0.513	22.44	1.874	9
11	0.490	22.21	1.410	13	31	1.903	23.38	2.185	9
12	0.091	20.47	1.080	10	32	0.107	21.76	1.621	12
13	0.065	23.84	1.679	7	33	1.410	20.07	2.408	11
14	0.328	21.03	1.272	12	34	1.633	18.67	1.155	11
15	0.439	17.16	2.655	9	35	1.945	15.56	1.408	9
16	0.034	22.00	2.016	13	36	0.933	22.08	2.711	15
17	0.570	23.83	2.421	11	37	0.600	20.07	2.993	12
18	0.686	16.06	1.702	7	38	1.500	21.12	1.450	9
19	1.107	21.02	1.497	7	39	0.703	15.10	1.536	15
20	0.715	21.82	2.801	6	40	1.551	20.08	1.814	14

Table A6 Order parameters for medium-size problem

Order type	$1/\lambda$	b	Order type	$1/\lambda$	b
1	2.498	2.310	21	2.464	2.826
2	3.837	2.630	22	3.726	2.706
3	2.732	2.279	23	3.367	2.066
4	2.955	2.430	24	3.167	2.347
5	3.642	2.391	25	3.597	2.846
6	2.270	2.630	26	3.811	2.549
7	3.051	2.962	27	2.680	2.648
8	3.517	2.088	28	2.462	2.799
9	2.261	2.049	29	2.186	2.295
10	2.912	2.532	30	2.086	2.590
11	3.401	2.322	31	3.311	2.967
12	2.340	2.708	32	3.459	2.706
13	2.101	2.360	33	2.765	2.411
14	2.677	2.262	34	2.933	2.431
15	3.570	2.623	35	3.163	2.760
16	3.434	2.897	36	2.282	2.728
17	2.001	2.823	37	2.454	2.110
18	3.477	2.859	38	3.487	2.155
19	3.575	2.633	39	2.347	2.797
20	2.525	2.188	40	3.074	2.991

Table A7 Order-product matrix for medium-size problem

0	0	0	1	1	0	0	0	1	0	1	1	0	1	0	0	0	1	1	1	0	1	1	1	1	0	0	1	0	0	0	1	0	1	0	1	0	1	1	0
1	0	1	1	1	0	0	0	1	1	1	1	1	1	0	0	0	1	1	0	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	1	0	1	0	0
1	0	1	0	0	0	1	1	1	0	0	1	1	0	1	0	1	0	1	1	0	1	1	0	0	0	1	1	1	0	0	0	0	1	0	0	0	1	0	1
1	0	0	1	0	1	0	1	1	0	1	0	1	0	1	1	1	1	1	0	0	1	1	1	1	0	1	1	1	0	0	1	1	0	0	1	1	1	0	0
1	0	0	1	0	1	1	0	1	0	1	0	1	1	1	0	0	1	1	1	0	0	1	1	0	1	0	1	1	0	1	1	1	1	0	1	0	0	1	0
0	0	0	0	0	1	1	1	0	1	1	1	1	0	0	0	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	0	0	1	1	1	0
1	0	1	0	1	0	0	0	1	1	0	0	1	0	1	1	1	1	1	1	0	1	1	0	1	0	0	1	0	0	0	0	0	1	0	1	0	1	0	1
1	0	1	1	1	0	0	0	1	0	1	1	1	1	1	1	0	1	0	1	0	1	1	0	0	1	1	1	0	0	0	0	0	1	1	0	1	0	1	0
0	0	0	0	1	0	0	1	0	0	1	0	0	1	1	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	1	1	1	0	0	0	0	1	1	0
0	0	0	1	1	1	1	1	1	0	1	0	0	1	1	0	1	0	0	1	0	0	1	1	0	0	0	0	1	0	1	0	1	0	0	1	1	1	1	0
1	1	0	1	0	1	1	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0
1	0	1	1	0	1	0	1	1	0	0	1	0	1	1	1	1	0	1	1	0	0	1	0	1	0	1	0	1	1	0	1	0	0	0	0	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	1	1	1	1	0	0	1	1	0	0	1	0	1	1	1	1	0	0	0	1	1	0	1	0	0	0	1	0	0
1	0	1	0	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	0	1	0	1	0	0	1	1	1	1	0	0	0	0	0	1	0	1	0	0
0	0	0	0	1	1	0	0	1	1	1	1	0	1	1	0	1	1	1	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	0	0	1	1	0
0	1	0	0	1	0	1	1	0	1	0	0	0	0	1	0	1	1	0	1	1	0	1	1	0	0	0	1	1	1	0	0	1	0	0	1	1	0	0	0
0	0	1	1	0	0	0	0	0	0	1	1	0	1	0	0	1	1	1	1	1	1	0	1	1	1	0	0	0	1	0	1	1	1	0	0	0	1	1	1
1	0	0	1	0	0	1	0	1	0	1	1	0	1	0	0	0	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	0	1	0	1	0	1	0	1
1	1	0	0	1	1	0	0	1	1	1	0	1	1	1	1	0	1	1	0	0	1	1	0	1	0	1	1	1	1	0	0	0	1	1	1	0	1	1	1
0	0	1	1	1	0	0	1	1	0	0	1	1	1	1	1	0	1	0	0	0	1	1	0	1	0	1	0	1	1	0	1	0	0	1	0	0	1	1	1
0	0	1	1	1	0	1	1	1	0	1	0	1	1	1	0	1	1	0	1	0	1	1	1	1	0	0	1	0	0	0	1	1	0	0	1	0	0	1	0
0	0	1	1	0	0	1	0	0	0	1	1	1	1	0	0	1	0	0	1	1	0	1	1	1	0	0	1	1	1	0	0	0	0	0	1	0	1	1	0
0	0	1	0	1	1	1	1	1	1	0	1	0	0	1	0	0	0	1	0	0	0	1	1	0	0	1	0	1	1	1	0	1	0	0	0	1	1	0	1
0	0	1	1	1	1	1	0	0	1	0	1	1	1	1	1	1	1	1	0	0	1	1	0	0	1	1	0	1	0	1	1	1	0	0	1	0	1	1	1
0	0	1	0	1	1	1	0	1	1	0	1	1	1	1	0	0	1	0	1	0	1	0	1	1	0	1	1	1	0	1	0	0	1	0	1	0	1	1	0
0	1	0	0	1	0	1	0	1	1	1	1	0	1	1	0	0	1	0	0	1	1	1	1	0	0	1	1	0	0	1	0	0	1	0	1	1	1	1	0
1	1	0	1	1	0	1	0	1	1	1	1	0	1	1	0	1	1	1	1	1	0	1	0	0	1	1	0	0	1	1	1	0	1	0	1	0	1	0	0
1	0	1	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	0	0	1	0	1	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0
0	0	0	0	1	0	1	1	1	1	1	1	1	0	1	0	0	1	0	1	1	0	0	0	0	0	1	1	0	1	1	0	1	1	0	1	0	1	0	0
1	0	1	0	0	0	1	1	1	1	1	1	0	1	1	0	1	1	1	1	0	1	1	1	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	1
0	1	0	0	0	0	1	0	1	1	0	0	1	1	1	1	1	1	1	1	0	1	1	0	0	0	1	1	1	1	0	0	1	1	0	0	0	1	0	0
0	1	0	1	0	1	1	1	1	0	1	1	0	0	0	0	0	1	1	1	0	1	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0	1	1	0
0	0	0	0	0	1	1	0	0	0	1	1	1	1	1	0	1	1	1	1	0	0	0	1	1	1	1	1	0	1	0	0	1	1	0	0	0	0	1	1
1	0	1	0	0	1	1	0	1	0	1	0	0	1	0	1	0	1	1	1	1	1	1	1	0	1	0	1	1	1	1	0	0	1	0	0	0	1	0	0
1	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	1	0	1	1	1	0	0	0	0	1	1	1	0	1	0	0	1	1	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	1	1	0	1	0	0	0	0	1	1	1	1	1	1	0	0	0	1	1	1	1	0	0	1	0	1	0	1	1	0
1	0	1	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	0	0	1	1	0	1	1	0	0	1	1	0	0	0	0	1	0	1	1	1	0	0
0	0	0	1	1	1	1	0	1	1	0	1	0	0	1	0	1	1	0	0	0	1	1	1	0	1	0	1	1	0	1	0	1	1	0	0	0	0	1	1
0	0	0	1	1	1	1	0	1	0	1	0	0	1	0	1	0	1	0	1	0	0	1	1	1	0	1	0	1	1	0	0	1	1	0	1	0	1	1	0
0	0	1	0	0	1	1	0	1	1	1	1	0	0	1	1	0	0	1	0	0	1	1	1	1	0	1	0	0	0	0	1	1	1	1	1	1	1	1	0

Table A8 Product parameters for large-size problem

Product type	h	C	$E[l]$	$x_i^{\max}$	Product type	h	C	$E[l]$	$x_i^{\max}$
1	1.348	18.00	2.089	12	51	1.409	23.15	1.811	6
2	1.687	21.74	1.647	9	52	0.552	16.71	2.254	10
3	1.033	21.61	1.933	9	53	1.931	21.52	1.784	11
4	0.355	20.53	2.323	8	54	1.286	18.18	2.797	14
5	1.944	17.87	2.040	9	55	1.996	17.40	1.409	6
6	0.620	19.50	1.341	6	56	0.792	19.76	2.227	6
7	0.950	20.74	1.728	10	57	0.861	24.71	1.984	6
8	1.826	22.22	1.746	6	58	0.564	22.42	2.559	13
9	1.774	16.50	1.019	11	59	0.896	20.52	2.873	14
10	0.670	19.58	1.628	12	60	1.062	20.76	2.093	9
11	1.374	21.21	2.278	14	61	1.620	16.87	2.100	7
12	1.262	22.84	2.616	8	62	0.162	21.65	1.599	12
13	0.069	16.06	2.585	8	63	0.732	22.15	1.977	12
14	1.806	24.16	1.187	11	64	0.910	18.61	2.923	7
15	0.402	15.44	2.476	9	65	0.294	22.98	2.734	13
16	1.355	23.24	2.463	9	66	0.627	20.88	1.677	12
17	0.799	24.67	2.525	14	67	0.439	22.25	1.765	9
18	0.990	21.65	2.530	10	68	0.783	23.42	1.625	6
19	1.549	24.72	2.171	9	69	0.792	17.54	2.052	13
20	1.373	24.90	2.587	14	70	1.495	17.03	2.615	12
21	1.713	16.82	1.389	8	71	0.635	23.89	2.795	9
22	1.292	16.82	1.641	14	72	0.155	19.61	1.445	8
23	1.634	15.16	1.497	10	73	1.702	21.65	2.534	13
24	1.750	23.37	1.328	10	74	0.958	18.83	1.315	11

(continued)

Table A8 Continued

Product type	h	C	$E[l]$	$x_i^{\max}$	Product type	h	C	$E[l]$	$x_i^{\max}$
25	1.842	19.29	2.940	14	75	1.859	19.77	2.337	12
26	1.647	24.15	1.888	11	76	1.068	17.35	1.647	14
27	0.786	23.73	1.388	12	77	0.925	24.77	2.187	11
28	0.321	21.36	1.759	9	78	0.227	22.80	1.365	7
29	0.017	18.01	1.319	12	79	0.732	20.51	1.809	12
30	1.538	18.66	1.766	10	80	1.896	19.15	2.428	6
31	1.619	22.20	2.606	8	81	1.637	21.42	2.045	12
32	1.045	18.20	1.869	7	82	1.900	24.47	1.781	9
33	1.971	16.02	2.752	11	83	0.107	17.35	2.797	11
34	1.947	23.03	1.031	7	84	1.142	22.88	2.136	11
35	1.646	21.83	1.576	14	85	0.138	18.12	2.771	10
36	1.001	17.06	2.291	10	86	0.642	18.75	2.463	9
37	0.586	18.06	2.601	6	87	0.272	17.15	2.937	6
38	0.861	15.37	2.797	9	88	0.082	18.09	2.848	14
39	0.003	16.55	2.356	13	89	1.171	20.88	2.165	14
40	0.490	16.60	1.654	9	90	1.501	18.70	2.917	13
41	0.846	21.85	1.576	8	91	1.265	22.49	2.001	6
42	0.667	16.37	2.149	6	92	0.075	16.92	1.378	7
43	0.523	17.50	1.046	11	93	1.252	18.62	2.393	12
44	0.045	18.54	2.601	9	94	0.156	16.55	1.811	10
45	0.059	21.62	1.922	12	95	1.361	24.69	1.481	10
46	1.727	20.20	2.837	8	96	1.340	16.32	1.773	11
47	0.246	16.79	1.951	9	97	1.583	22.73	2.758	6
48	1.158	17.81	2.549	11	98	0.429	16.24	1.921	9
49	0.915	23.96	1.386	9	99	1.806	22.52	1.991	7
50	1.469	18.00	2.616	11	100	1.017	23.15	2.352	7

Table A9 Order parameters for large-size problem

Order type	$1/\lambda$	b	Order type	$1/\lambda$	b
1	2.772	2.364	31	2.033	2.374
2	2.327	2.390	32	2.289	2.622
3	2.349	2.187	33	3.537	2.936
4	3.791	2.758	34	2.073	2.483
5	3.227	2.896	35	3.644	2.868
6	3.758	2.222	36	2.339	2.659
7	2.278	2.663	37	2.191	2.876
8	3.497	2.664	38	3.326	2.679
9	3.151	2.448	39	3.351	2.736
10	2.303	2.948	40	3.478	2.190
11	3.369	2.742	41	2.019	2.398
12	3.018	2.432	42	2.054	2.214
13	3.290	2.802	43	2.710	2.481
14	2.541	2.890	44	2.251	2.921
15	2.434	2.977	45	2.732	2.228
16	3.276	2.350	46	2.064	2.885
17	3.803	2.758	47	2.465	2.302
18	2.198	2.151	48	2.905	2.018
19	2.666	2.021	49	3.153	2.093
20	2.560	2.070	50	3.224	2.335
21	3.215	2.028	51	3.024	2.292
22	2.555	2.088	52	3.220	2.620
23	2.683	2.839	53	3.945	2.809
24	2.660	2.095	54	2.514	2.917
25	2.238	2.408	55	3.517	2.485
26	2.211	2.907	56	2.563	2.837
27	3.466	2.543	57	3.723	2.908
28	3.281	2.404	58	2.958	2.935
29	2.794	2.524	59	3.281	2.765
30	3.280	2.544	60	2.700	2.144