Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 316908, 18 pages
doi:10.1155/2012/316908

Research Article

A Hybrid Genetic Algorithm for the Multiple Crossdocks Problem

Zhaowei Miao,[1] Ke Fu,[2] and Feng Yang[1]

[1] *School of Management, Xiamen University, Xiamen 361005, China*
[2] *Lingnan (University) College, Sun Yat-Sen University, Guangzhou 510275, China*

Correspondence should be addressed to Ke Fu, fuke@mail.sysu.edu.cn

Received 8 December 2011; Revised 1 April 2012; Accepted 15 April 2012

Academic Editor: John Gunnar Carlsson

We study a multiple crossdocks problem with supplier and customer time windows, where any violation of time windows will incur a penalty cost and the flows through the crossdock are constrained by fixed transportation schedules and crossdock capacities. We prove this problem to be $\mathcal{NP}$-hard in the strong sense and therefore focus on developing efficient heuristics. Based on the problem structure, we propose a hybrid genetic algorithm (HGA) integrating greedy technique and variable neighborhood search method to solve the problem. Extensive experiments under different scenarios were conducted, and results show that HGA outperforms CPLEX solver, providing solutions in realistic timescales.

1. Introduction

As companies seek more profitable supply chains, there has been a desire to optimize distribution networks to reduce logistics costs. This includes finding the best locations for facilities, minimizing inventories, and minimizing transportation costs. A distinct recent industry example is the successful implementation of crossdocking strategy at Wal-Mart, whose crossdocks require coordinating 2000 dedicated trucks over a large network of warehouses, crossdocks, and retail points [1]. While there is a rich literature on conventional facility location problems, crossdocking strategies—which minimize inventory by processing goods quickly for reshipment—have recently attracted the attention of researchers (see, e.g., [2–4]). In conventional transshipment-inventory models, a common assumption is that demand (usually stochastic) that cannot be met from one supply point can be fulfilled through some other point. The objective is then to evaluate a control policy for replenishment. Work on this subject has been extensive and can be found in, for example, Krishnan and Rao [5], Karmarkar and Patel [6], Karmarkar [7], Tagaras [8], Robinson [9], Rudi et al. [10], Grahovac and Chakravarty [11], Herer and Tzur [12], Herer et al. [13], Axsäter [14], and Axsäter [15]. For the general n location transshipment model, heuristics were proposed by Robinson [9] who developed a large-scale LP by discretizing demand. Although these studies considered inventory and transshipment costs, they did not address time constraints that occur during the transshipment process, for example, constraints imposed by transportation schedules. There has been relatively limited research on distribution and system design, which includes crossdocks. Some recent attempts can be found in Donaldson et al. [16], Ratcliff et al. [17], Gumus and Bookbinder [4], Li et al. [3], Miao et al. [18], and Boysen et al. [19]. In particular, Gumus and Bookbinder [4] modeled location-distribution networks that include crossdock facilities to determine the impact on the supply chain. Li et al. [3] developed a heuristic algorithm to find JIT schedules within a single crossdock. Miao et al. [18] and Boysen et al. [19] studied how to schedule the inbound and outbound trucks to achieve high operational efficiency within a single crossdock.

Our work differs from the above research in that we study a kind of multiple crossdocks problem where transportation is available at fixed schedules, and where both shipping and delivery at supply and demand locations can be executed within specified time windows with normal transportation cost, and any shipment that cannot be met will be fulfilled through external channels, which causes penalty costs. Supplier time windows allow, for example, flexibility in planning for the best shipping times to fit production and operating schedules. Time windows at demand points satisfy customer requirements, for example, when service deadlines must be met. Moreover, in the real-world applications, sometimes the time windows are allowed to be violated. There are two primary reasons for this: one is exogenous, for example, it might not be practical to satisfy all the time windows constraints when the demands are too high during a certain period; the other is endogenous, for example, some shipments are emergent and have to be executed outside the normal time windows. Clearly, such abnormal arrangements usually incur additional costs we call penalty in this paper. Furthermore, we consider inventory cost at the crossdocks, which includes storing cost and handling cost and are one of the cost sources in any transshipment strategy.

To the best of our knowledge, there are some papers closet to our work, including Lim et al. [20], Chen et al. [21], and Ma et al. [22]. Lim et al. [20] studied the complexity of different types of multiple crossdocks problems where transportation schedules such as flexible schedules and fixed schedules, and time constraints at manufacturers and customers are included. Our model extends one of the cases studied by them, which is called single shipping and single delivery by fixed schedules. However, their problem requires all the demands should be satisfied by the suppliers, which is very difficult to achieve even to find a feasible solution. We relax this constraint and allow demands of some customers to be unfulfilled, but penalty cost will be incurred. In addition, most importantly, in the earlier works, they did not set up any mathematical formulation and provide any implementable algorithms that were able to solve this practical problem. We model this problem as an integer programming problem, prove it to be $\mathcal{NP}$-hard in the strong sense, and then focus on developing efficient heuristic algorithms. Chen et al. [21] extended another case studied by Lim et al. [20], which is called single shipping and single delivery by flexible schedules, simplified the problem by discretizing the time horizon, and designed metaheuristic algorithms to solve it. Despite the constraint of single shipping and single delivery, Ma et al. [22] took into consideration setup cost of each vehicle in their multiple crossdocks problem, which is also strong $\mathcal{NP}$-hard and solved by a two-stage heuristic algorithm developed by them.

While one of the main objectives in this paper is to develop an efficient heuristic algorithm to solve the above multiple crossdocks problem, we find that numerous researches have been dedicated to design meta-heuristic algorithms to solve transportation problems. For example, Hai and lim [23] used a Tabu-embedded simulated annealing algorithm that restarted from the current best solution after several nonimproving iterations to solve the pickup and delivery problem with time windows. Chen et al. [21] developed three kinds of meta-heuristics, namely, simulated annealing, tabu search, and integrated simulated annealing and tabu search with local search technique to solve multiple crossdocks problems with inventory and time windows. In a dynamic vehicle dispatching problem with pickups and deliveries, Michel et al. [24] proposed a tabu search with neighborhood search based on ejection chains to explore the proposed problem. All these meta-heuristic algorithms are focused on two aspects, one is how to generate initial feasible solutions efficiently according to the specific structure of the problems, and the other is how to improve the current best solution. Based on this, neighborhood search and integrated meta-heuristic algorithm are then developed. In this paper, we adopt the basic idea of the genetic algorithm and meanwhile take advantage of the problem structure to develop a hybrid genetic algorithm (HGA) integrating greedy technique and variable neighborhood search to solve the proposed problem. The unique feature of HGA is that it makes assignment of crossdocks and routes to suppliers and customers in *two stages* and is further refined by the greedy technique and variable neighborhood search. Since the problem we consider is a fairly general transshipment problem with complicated constraints, we believe that our method could be useful to solve other transshipment problems of this sort. We conduct various computational experiments with different problem scales, and the results show that the proposed HGA can yield better solutions for various problem instances of different scales, especially for the large-sized problems compared with the CPLEX solver, which gives solutions before getting terminated within the stipulated time limit of execution.

The rest of this paper is organized as follows. In Section 2, we formulate the problem as an integer programming problem and provide complexity analysis. In Section 3, we explore the problem structure and develop HGA. Section 4 demonstrates HGA with computational results for various problem instances of different scales. The paper is concluded in Section 5.

2. The Multiple Crossdocks Problem

In this section, we describe the multiple crossdocks problem and introduce basic notation in Section 2.1. We then formally formulate the problem as an integer programming problem and provide complexity analysis in Section 2.2.

2.1. Problem Description

The problem studied here extends the well-known transshipment problem to include constraints imposed by time and inventory considerations, which arise in applications. The following assumptions are made. First, a supplier (customer) is allowed to ship goods to crossdocks (receive goods from crossdocks) within a specified time window with a normal cost level; however, if the shipment cannot be met, a much higher cost called penalty here is incurred. When the shipments take place outside the time windows, then it means that the current transportation network is unable or too busy to fulfill this shipment requirement and external channels have to be used to ship the cargos at a much higher cost, which may include the higher transportation costs and the order earliness or lateness costs through the external channels. Second, shipped goods can be delayed at crossdocks, which is helpful to satisfy time windows constraints and also helpful to potential consolidation. Third, shipping schedules offered by transportation providers are fixed, that is, departure and arrival times of any schedule are fixed. For example, the schedules in the railway network or in airline operations are usually fixed. We assume that each schedule has a set of associated shipping costs and route capacities. Fourth, the setup cost of each shipment sometimes is very high in real world, so in order to reduce setup cost as much as possible, it requires each supplier make only one batch shipment to some crossdock within its specified time window, and each customer can receive goods only one time from some crossdock within its time window, which is called single shipping and single delivery case [20], and for this case, consolidation is quite important. Finally, the objective of our problem is to satisfy the demands of the customers with minimum total costs including shipping cost, inventory cost and penalty cost without violating the capacity constraints of crossdocks and routes through given fixed schedules.

The underlying problem can be represented by a network. Let $\Sigma := \{1, \ldots, n\}$ be the set of supply nodes (suppliers) where, for each $i \in \Sigma$, s_i units of goods are available, which can be shipped (released) in the time window $[b_i^r, e_i^r]$, $\Delta := \{1, \ldots, m\}$ the set of demand nodes (customers) where each $k \in \Delta$ requires d_k units of goods, which must be delivered (accepted) within the time window $[b_k^a, e_k^a]$, and $\mathbf{X} := \{1, \ldots, l\}$ the set of crossdocks, where each $j \in \mathbf{X}$ has inventory capacity c_j and inventory cost h_j per unit per time. Take S_1 to denote all fixed scheduled routes between points in Σ and points in $\mathbf{X}$, that is, routes serviced by transport providers, each with a scheduled departure (begin) and arrival (end) time, capacity and unit transportation cost. Similarly, let S_2 denote the set of fixed schedules between the crossdocks $\mathbf{X}$ and customers Δ.

2.2. Mathematical Formulation

We now introduce more notation that is used in our formulation as follows

$S_{i,j}$: set of fixed transportation schedules between supplier i and crossdock j, and $|S_{i,j}| = \gamma_{i,j}$, where $|\cdot|$ represents the cardinality of a set.

$S'_{j,k}$: set of fixed transportation schedules between crossdock j and customer k, and $|S'_{j,k}| = \gamma'_{j,k}$.

$(b^r_{i,j,q}, e^r_{i,j,q})$: qth fixed transportation schedule in $S_{i,j}$, where $b^r_{i,j,q}$ and $e^r_{i,j,q}$ are the beginning time point and ending time point of this fixed schedule, respectively.

$(b^a_{j,k,q}, e^a_{j,k,q})$: qth fixed transportation schedule in $S'_{j,k}$, where $b^a_{j,k,q}$ and $e^a_{j,k,q}$ are the beginning time point and ending time point of this fixed schedule, respectively.

$c_{i,j,q}$: unit shipping cost from supplier i to crossdock j through qth fixed transportation schedule in $S_{i,j}$.

$c'_{j,k,q}$: unit shipping cost from crossdock j to customer k through qth fixed transportation schedule in $S'_{j,k}$.

P_i: unit penalty cost for supplier i if its cargo cannot be shipped out.

P'_k: unit penalty cost for customer k if its demand cannot be met.

T_j: set of ending time points of all fixed transportation schedules in $\cup_{i=1}^n S_{i,j}$, that is, $T_j = \{e^r_{i,j,q} : 1 \le i \le n, 1 \le q \le \gamma_{i,j}\}$.

T'_j: set of beginning time points of all fixed transportation schedules in $\cup_{k=1}^{m} S'_{j,k}$, that is, $T'_j = \{b^a_{j,k,q} : 1 \le k \le m, 1 \le q \le \gamma'_{j,k}\}$.

$\widetilde{T}_j$: set of time points when the inventory level of crossdock j is likely to be changed, that is, $\widetilde{T}_j = T_j \cup T'_j$. Let $|\widetilde{T}_j| = \tau_j$ and all the elements in $\widetilde{T}_j$ are sorted in an increasing order, and let $t_{j,g}$ $(g = 1, 2, \ldots, \tau_j)$ correspond to these τ_j time points such that $t_{j,1} \le t_{j,2} \le \cdots \le t_{j,\tau_j}$. Using this notation, we can easily formulate the set of flow conservation constraints later.

$\text{CAP}_{i,j,q}$: shipping capacity of qth fixed transportation schedule in $S_{i,j}$.

$\text{CAP}'_{j,k,q}$: shipping capacity of qth fixed transportation schedule in $S'_{j,k}$.

$\theta_{i,j,q}$: a binary parameter, which is 0 if the beginning time point of qth fixed transportation schedule in $S_{i,j}$ is within the time window of supplier i, that is, $b^r_{i,j,q} \in [b^r_i, e^r_i]$, and 1 otherwise.

$\theta'_{j,k,q}$: a binary parameter, which is 0 if the ending time point of qth fixed transportation schedule in $S'_{j,k}$ is within the time window of customer k, that is, $e^a_{j,k,q} \in [b^a_k, e^a_k]$, and 1 otherwise.

The following are decision variables.

$x_{i,j,q}$: binary, which is 1 if to deliver cargos from supplier i is bound for crossdock j through qth fixed transportation schedule in $S_{i,j}$, and 0 otherwise.

$x'_{j,k,q}$: binary, which is 1 if to receive cargos from crossdock j to customer k is through qth fixed transportation schedule in $S'_{j,k}$, and 0 otherwise.

$y_{j,t_{j,g}}$: integer, which is inventory level in crossdock j at time $t_{j,g}$, where $t_{j,g} \in \widetilde{T}_j$.

We are now ready to formulate the transshipment problem, which hereafter is called problem (P):

$$\min \text{COST}_{\text{Transportation}} + \text{COST}_{\text{Penalty}} + \text{COST}_{\text{Inventory}}, \tag{P}$$

where

$$\begin{aligned}
\text{COST}_{\text{Transportation}} &= \sum_{i=1}^{n}\sum_{j=1}^{l}\sum_{q=1}^{\gamma_{i,j}} c_{i,j,q} s_i x_{i,j,q} + \sum_{k=1}^{m}\sum_{j=1}^{l}\sum_{q=1}^{\gamma'_{j,k}} c'_{j,k,q} d_k x'_{j,k,q}, \\
\text{COST}_{\text{Penalty}} &= \sum_{i=1}^{n} P_i s_i \left(1 - \sum_{j=1}^{l}\sum_{q=1}^{\gamma_{i,j}} x_{i,j,q}\right) + \sum_{k=1}^{m} P'_k d_k \left(1 - \sum_{j=1}^{l}\sum_{q=1}^{\gamma'_{j,k}} x'_{j,k,q}\right), \\
\text{COST}_{\text{Inventory}} &= \sum_{j=1}^{l}\sum_{g=1}^{\tau_j} h_j \left(t_{j,g} - t_{j,g-1}\right) y_{j,t_{j,g-1}},
\end{aligned} \tag{2.1}$$

s.t.

$$x_{i,j,q} \le 1 - \theta_{i,j,q} \quad (1 \le i \le n,\ 1 \le j \le l,\ 1 \le q \le \gamma_{i,j}), \tag{2.2}$$

$$x'_{j,k,q} \le 1 - \theta'_{j,k,q} \quad \left(1 \le j \le l,\ 1 \le k \le m,\ 1 \le q \le \gamma'_{j,k}\right), \tag{2.3}$$

$$\sum_{j=1}^{l} \sum_{q=1}^{\gamma_{i,j}} x_{i,j,q} \le 1 \quad (1 \le i \le n), \tag{2.4}$$

$$\sum_{j=1}^{l} \sum_{q=1}^{\gamma'_{j,k}} x'_{j,k,q} \le 1 \quad (1 \le k \le m) \tag{2.5}$$

$$\begin{gathered} s_i x_{i,j,q} \le \mathrm{CAP}_{i,j,q} \quad (1 \le i \le n,\ 1 \le j \le l,\ 1 \le q \le \gamma_{i,j}), \\ d_k x'_{j,k,q} \le \mathrm{CAP}'_{j,k,q} \quad \left(1 \le k \le m,\ 1 \le j \le l,\ 1 \le q \le \gamma'_{j,k}\right), \end{gathered} \tag{2.6}$$

$$\begin{gathered} y_{j,t_{j,g}} \le c_j (1 \le j \le l,\ 1 \le g \le \tau_j), \\ y_{j,t_{j,0}} = 0 (1 \le j \le l, t_{j,0} = 0), \end{gathered} \tag{2.7}$$

$$y_{j,t_{j,g}} = y_{j,t_{j,g-1}} + \sum_{i=1}^{n} \sum_{\{q: e^r_{i,j,q} = t_{j,g}\}} s_i x_{i,j,q} - \sum_{k=1}^{m} \sum_{\left\{q: b^a_{j,k,q} = t_{j,g}\right\}} d_k x'_{j,k,q} (1 \le j \le l,\ 1 \le g \le \tau_j), \tag{2.8}$$

$$\begin{gathered} x_{i,j,q} \in \{0,1\} \quad (1 \le i \le n,\ 1 \le j \le l,\ 1 \le q \le \gamma_{i,j}), \\ x'_{j,k,q} \in \{0,1\} \quad \left(1 \le j \le l,\ 1 \le k \le m,\ 1 \le q \le \gamma'_{j,k}\right), \\ y_{j,t_{j,g}} \in \mathbb{N} \quad (1 \le j \le l,\ 1 \le g \le \tau_j). \end{gathered} \tag{2.9}$$

In the above formulation, the objective is to minimize total cost, including transportation cost, penalty cost, and inventory cost. Note that we impose the penalty cost on both supplier and customer sides here because unfulfilled demands have different impact on each side in general. Constraint (2.2) ensures that each available fixed transportation schedule is within the time window of suppliers. Similarly, the available fixed transportation schedule of customers is given by (2.3). Constraint (2.4) ensures that each delivery is fulfilled within each supplier specified time window at most once and (2.5) forces each customer to receive cargos within its time window for no more than one time, which is required by single shipping and single delivery constraint. The capacity constraints of fixed schedules are given by (2.6). The capacity constraint of every crossdock is restricted by (2.7) and we also set a zero initial inventory for each crossdock. The changes of inventory level of each crossdock are recorded in (2.8), which ensures cargo flow conservation.

We have the following proposition whose proof is given in the appendix.

Proposition 2.1. *The multiple crossdocks problem* (P) *is $\mathcal{NP}$-hard in the strong sense, even if supply and demand time windows and crossdock and route capacities are relaxed.*

From the above proposition, we know that to find minimum cost of this problem is $\mathcal{NP}$-hard in the strong sense. Hence, it is unlikely to find a polynomial or pseudopolynomial time algorithm to solve the problem unless $\mathcal{P}=\mathcal{NP}$. As a result, we focus on efficient heuristics to solve problem (P). In the next section, we describe a heuristic that exploits the problem structure and solves the problem efficiently.

3. Hybrid Genetic Algorithm

Genetic algorithm (GA) has become a well-known and powerful metaheuristic approach for hard combinatorial optimization problems. Genetic algorithm is based on the ideas of natural selection and has been applied to numerous combinatorial optimization problems successfully. However, basic genetic algorithms have limitations to attain the optimal solution. We propose a hybrid genetic algorithm (HGA) integrating greedy technique and variable neighborhood search to solve problem (P) as described in the previous section. The proposed HGA simulates the natural selection process; in addition, it incorporates the special structure of problem (P). In particular, in problem (P), we expect to assign crossdocks and routes to *both the suppliers and customers* in the most cost-effective manner. Clearly, the assignments of crossdocks and routes for suppliers will affect the assignments for customers due to capacity and time windows constraints and vice versa. As a result, simultaneously assigning crossdocks and routes for suppliers and customers is not effective. Observing this fact, we propose a two-stage assigning approach as follows. In the first stage, we assign crossdocks and routes for suppliers; in the second stage, by taking advantage of the former assignments for suppliers, we then assign crossdocks and routes for customers. This principle is used throughout the process of HGA including the initial solution generation and the solution updates. We describe the overall procedure of HGA briefly as follows, the formal procedure will be presented in Section 3.2 after all the components of HGA are discussed in Section 3.1.

Step 1. Generate initial solutions (chromosomes) by greedy technique. As described above, there are two stages to generate initial solutions. In the first stage, we apply *cost saving priority* to the assignments of crossdocks and routes to suppliers. In the second stage, for customers, the procedure is also based on *cost saving priority*, and then adjust the solution by *time match criterion*.

Step 2. Evaluate the fitness of each individual with respect to the objective function.

Step 3. Select a group of best individuals as the population pool, which guarantees that the best genes can be preserved in offsprings.

Step 4. Apply two-opt strategy as the crossover operator to generate offsprings.

Step 5. Apply mutation to diversify the pool by changing some genes in specified chromosomes.

Step 6. Apply variable neighborhood search to each new generated offspring, and go to Step 2 until one of the termination conditions is satisfied.

Note that both crossover and mutation operators are only applied when assigning for suppliers and any change in assignments for suppliers will trigger changes in assignments for customers. Furthermore, in HGA, variable neighborhood search is applied to improve the solution. This evaluation-selection-reproduction-local search cycle is repeated until one

v_1	χ_1	χ_2	χ_3	χ_4	χ_5	χ_6	χ_7	χ_8
v_2	ψ_1	ψ_2	ψ_3	ψ_4	ψ_5	ψ_6	ψ_7	ψ_8

Figure 1: Two-vector chromosome for $n = m = 4$.

of the termination conditions is satisfied, namely, either the maximum number of iterations is reached or the best solution cannot be improved within a certain number of iterations.

3.1. Components of HGA

3.1.1. Solution Representation

The chromosome is an important component in GA, which has a great influence on the algorithm output. In the basic GA, a chromosome is usually encoded as *one* sequence to represent a solution. However, since our problem involves assigning *both the crossdocks and routes*, we construct the chromosome by two vectors. The first vector represents the assignments of crossdocks to suppliers and customers, and the second vector represents the assignments of routes to suppliers and customers. Formally, the two vectors are as follows.

(1) crossdocks assignment vector (hereafter called v_1),

(2) routes assignment vector (hereafter called v_2).

For 1, crossdocks assignment vector is represented as $v_1 = (\chi_1, \chi_2, \ldots, \chi_{n+m})$, where n and m are the number of suppliers and that of customers, respectively; each $\chi_i \in \{1, 2, \ldots, l\}$ (recall that there are l crossdocks) represents an assignment of crossdock χ_i to suppler i ($i = 1, \ldots, n$) or customer $i - n$ ($i = n + 1, \ldots, n + m$). That is, supplier i ships cargos to crossdock χ_i ($i = 1, \ldots, n$), or customer $i-n$ receives cargos from crossdock χ_i ($i = n+1, \ldots, n+m$). For 2, routes representation vector v_2 is designed in the way similar to v_1, $v_2 = (\psi_1, \psi_2, \ldots, \psi_{n+m})$, where ψ_i means supplier i ($i = 1, 2, \ldots, n$) or customer $i - n$ ($i = n + 1, n + 2, \ldots, n + m$) chooses route ψ_i to release or receive cargos. Note that ψ_i represents an available fixed schedule between any two points. In particular, when $1 \le i \le n$, ψ_i means supplier i chooses ψ_i^{th} route among all the available routes $\{1, 2, \ldots, \gamma_{i,\chi_i}\}$ to ship cargos to crossdock χ_i (which has been assigned in vector v_1 already); similarly, when $n + 1 \le i \le n + m$, ψ_i means customer $i - n$ chooses ψ_i^{th} route among all the available routes $\{1, 2, \ldots, \gamma'_{\chi_i,i}\}$ to ship cargos to crossdock χ_i (which again has been assigned in vector v_1). The whole chromosome for a problem instance with four suppliers and four customers is illustrated in Figure 1.

The initial sequences are generated randomly. However, given such a chromosome sequence, we cannot guarantee the feasibility of the solution because the time windows of suppliers and customers may conflict with each other when proper transportation schedules do not exit or the capacity constraints of crossdocks or routes are violated in the cargo transferring process. In order to overcome these difficulties and find a feasible solution efficiently, a greedy technique is applied to identify a relatively better solution. More details about generation of initial solution will be given next.

3.1.2. Generation of Initial Solution

Common integer programming methods usually fail for large-scale problems. In view of the complexity of the crossdock problems, in order to obtain a much better solution, our approach is to attain initial solutions using greedy method. At first, we can combine the transportation cost and the penalty cost together in the objective function by some algebra as follows

$$\sum_{i=1}^{n}\sum_{j=1}^{l}\sum_{q=1}^{\gamma_{i,j}}(c_{i,j,q}-P_i)s_i x_{i,j,q}+\sum_{k=1}^{m}\sum_{j=1}^{l}\sum_{q=1}^{\gamma'_{j,k}}\left(c'_{j,k,q}-P'_k\right)d_k x'_{j,k,q}+C, \tag{3.1}$$

where constant $C=\sum_{i=1}^{n}P_i s_i+\sum_{k=1}^{m}P'_k d_k$ and each coefficient is negative because we assume that the unit penalty cost is higher than unit transportation cost. Let $C_{i,j,q}$ represent the cost supplier i can save if he ships cargos to crossdock j by $(b^r_{i,j,q}, e^r_{i,j,q})$ and $C'_{j,k,q}$ represent the cost customer k can save if he receives cargos from crossdock j by $(b^a_{j,k,q}, e^a_{j,k,q})$, where $C_{i,j,q}=P_i-c_{i,j,q}$ and $C'_{j,k,q}=P'_k-c'_{j,k,q}$, respectively.

From (3.1), we can see that, for a supplier, the most important point is saving cost $C_{i,j,q}$, which is the primary factor that determines which crossdock to ship to and which route to be chosen between these two locations. This decision subsequently affects the holding cost in the corresponding crossdock it ships to. To reflect this fact, we set probabilities for supplier-crossdock assignments so that a higher cost saving of an assignment would result in a higher probability for that assignment to be chosen. Formally, the probability that supplier i ships cargos to crossdock j by schedule $(b^r_{i,j,q}, e^r_{i,j,q})$ is calculated as follows:

$$\text{Prob}_{i,j,q}=\frac{C_{i,j,q}}{\max_{j',q'}\{C_{i,j',q'}\}} \quad (1\le i\le n,\ 1\le j\le l,\ 1\le q\le \gamma_{i,j}). \tag{3.2}$$

For customers, we know which crossdocks have cargo after the first stage, and then we assign one of these crossdocks to each customer and choose a route to deliver. The assignment strategy of crossdocks and schedules for customer is similar to that of the suppliers, and we also can calculate the probability similar to (3.2). Only one difference is that the customers just can be assigned to those crossdocks that have been already assigned to suppliers, instead of all the crossdocks. After that, we need adjust the solution according to time match criterion to guarantee feasibility. During adjustment, we needs to eliminate those infeasible issues such as time conflicts, overflow of capacity, and nonconservation of cargo flows. By adjustment, infeasible solutions will scarcely be generated. However, for some infeasible solutions that are too difficult to repair, we just need to unfulfill those customers who incur infeasibility to get a feasible solution.

3.1.3. Crossover

In order to preserve efficient genes in a chromosome, the two-point crossover operator is applied to generate offspring, which is widely adopted in GA (see, e.g., [3, 18]). The crossover operator is illustrated by Figure 2. First, two individuals, we call *parent 1* and *parent 2*, are selected randomly from the population pool, and then two points are randomly selected between genes representing assignment for suppliers, and because crossover operators

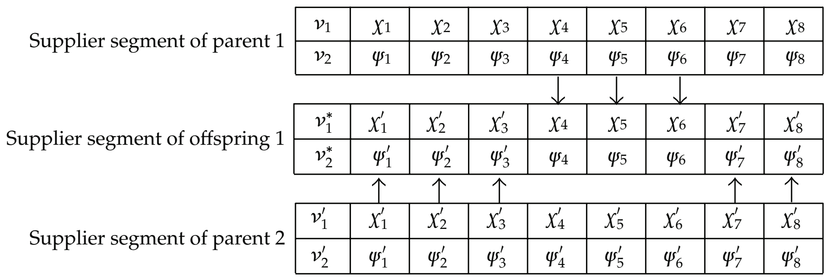
Supplier segment of parent 1
ν1 χ1 χ2 χ3 χ4 χ5 χ6 χ7 χ8
ν2 φ1 φ2 φ3 φ4 φ5 φ6 φ7 φ8
Supplier segment of offspring 1
ν1* χ'1 χ'2 χ'3 χ4 χ5 χ6 χ'7 χ'8
ν2* φ'1 φ'2 φ'3 φ4 φ5 φ6 φ'7 φ'8
Supplier segment of parent 2
ν'1 χ'1 χ'2 χ'3 χ'4 χ'5 χ'6 χ'7 χ'8
ν'2 φ'1 φ'2 φ'3 φ'4 φ'5 φ'6 φ'7 φ'8

Figure 2: Crossover operator.

applied among customers will generate lots of infeasible solutions, assignments for customers are determined by assignments for suppliers. The symbols outside the two crossover points are directly inherited from *parent 1* to *offspring 1*, and the other genes of *offsprings 1* are transferred from the symbols of *parent 2* in corresponding positions. After crossover, crossdocks and schedules are reassigned for customers using the aforementioned greedy technique. After changing the roles of parents, the same procedure is applied to generate *offspring 2*.

3.1.4. Mutation

It is obvious that the initial population generated by two-stage greedy method has poor ability to carry the genetic diversity because the cost saving priority and time match priority in the greedy technique reduce the chance for crossdocks and schedules with relatively low cost saving to be chosen. As a result, the greedy technique causes population pool homogeneity. In order to overcome this limitation, we use the mutation operator with a given individual mutation probability P_{im} to mutate every individual and apply the greedy technique to mutation of some gene representing the assignment of crossdock to some supplier with gene mutation probability P_{gm} calculated by (3.2) in the selected individual. After that, we also need to adjust the new solution to be feasible by the strategy which is similar to that of initial solutions. Different from prior crossover operator slightly, the mutation operator may deteriorate the current solution in terms of fitness. However, its goal is not only to preserve the best genes but also to attain inferior genes with some probability to diversify the pool.

3.1.5. Variable Neighborhood Search

GA is a global search technique but is poor in local search. Therefore, we use variable neighborhood search technique to improve local search ability. A basic component of any local search is neighborhood search. A solution is said to be a neighbor of another solution s if it can be obtained from s through a neighborhood move. We develop several such moves suitable for this problem to find neighborhood solutions. These moves are key to the successful implementation of these heuristics. Next are the key moves and strategies for neighborhood search.

Vary Crossdock for Supplier

In our algorithm, the crossdock assignment and route choosing problem has a special feature that a sequence may not cover all the crossdocks, which is different from other order-based problems. So single-point change strategy is applied here rather than two-opt strategy. First, we randomly select a gene in supplier segment in one chromosome and change its assigned crossdock to another that is different from the current one with probability calculated by (3.2) and repeat this procedure until a better solution is obtained. It should be emphasized that, in each search process, we use the greedy technique mentioned in Section 3.1.2 to assign crossdocks and schedules to customers after the reassignment for each supplier. This is an efficient strategy to guarantee the feasibility of the new solution.

Change Route for Supplier

Routes selection is the most difficult decision in our problem, especially for suppliers. There is no good method available that can efficiently select routes, which can guarantee the feasibility besides the saving cost, not only in generating initial solution but also in generation of new population. However, routes selection has a great influence on total cost. In addition, it affects the inventory level of each possible time point in crossdocks, which determines the customers crossdock assignment and route selection in our algorithm. In order to obtain a better route, our strategy is to not only change the routes for suppliers, but also keep the crossdock assignment unchanged. In this move, we change the assigned route of a randomly selected supplier to a new one, and repeat this procedure until a better solution is obtained. It must be of concern that, for any change in suppliers route, we must conduct reassignment of crossdocks and schedules for customers to ensure the feasibility.

Swap Crossdocks for Customers

In the assignment process, we assign crossdocks and routes for customers one by one, which would reduce the probability for latter customers to choose a particular crossdock. For example, suppose that the inventory at certain time of a *crossdock j* could meet the demand of *customer 1* or *customer 2* individually if there exist available routes for both of them and the probabilities for them to choose the crossdock are close to each other. If *customer 1* chooses *crossdock j* first, then *customer 2* would have small possibility to also choose *crossdock j* because the remaining inventory of *crossdock j* may not meet its demand. This move is designed to overcome this difficulty. That is, we give priority to *customer 2* if that helps reduce the total cost. Our strategy for this move is to select two customers randomly whose crossdocks are different from each other, swap their crossdocks, and then repeat this procedure until a better solution is obtained.

Change Route for Customer

The goal for this move is the same as the third strategy, that is, eliminating the ordering effect for assigning crossdocks and routes for customers by the greedy method. The difference is that this move focuses on two customers whose crossdocks are the same. It is obvious that, if two customers choose the same crossdock but are assigned to receive cargos in order, for example, supposed *customer 1* and *customer 2* select the same crossdock, *customer 1* has an advantage to ship out cargos in time; however, the total cost may be much lower if *customer*

```
Generate initial solutions by two-stage greedy method.
for $iter \leftarrow 1$ to #$maximum\_iter$ do
    for $off \leftarrow 1$ to #$crossover$ do
        Randomly select Parent 1 and Parent 2.
        Crossover Parent 1 and Parent 2 to produce a new Offspring.
    end for
    for each offspring do
        Mutate offspring with individual mutation probability $P_{im}$ and gene mutation probability
        $P_{gm}$
        Apply neighborhood search to each newly-produced Offspring.
    end for
    Select the best #pop individuals from all the Individuals including all current parents and
    newly produced offsprings.
    Update current best solution.
    if the best solution could not improve within #$terminate\_iter$then
        Consider the solution as the goal optimal solution.
        break
    end if
end for
output the best solution and escaped time
```

Algorithm 1: HGA to solve problem (P).

2 can ship out cargos earlier than *customer 1* when the penalty cost of *customer 2* is relatively high. So the strategy aims to give a priority to *customer 2*.

3.2. Framework of HGA

With these components, we now outline HGA framework in Algorithm 1. In this algorithm, #pop denotes the number of populations, #crossover denotes the number of crossovers we will do, #terminate_iter denotes the maximum number of iterations the current best solution cannot be improved, #maximum_iter denotes the maximum number of iterations, and P_{im} and P_{gm} are defined in Section 3.1.4.

4. Computational Experiments

We generate a great variety of problem instances and apply HGA to solve them. For comparison purposes, we also use ILOG CPLEX 11.0 solver to solve the instances, which is widely adopted by many papers (see, e.g., [3, 21, 22]). Both HGA and the CPLEX solver were run on a personal computer with an Intel 2.4 GHz Pentium 4 CPU and 1G memory. The test data generation, parameter settings of HGA, and detailed computational results are reported in the following content.

4.1. Test Data Generation and Experimental Parameter Setting

Because crossdocking problems are relatively new, there are no benchmarks test sets available. As a result, we generated our own data. The data sets are generated randomly in such a way that they can represent realistic situations and can cover different scenarios, which is suggested by Chen et al. [21], Li et al. [3], and Ma et al. [22], and the parameters in HGA are based on numerous computational experiments, and they are effective to attain desirable results.

The test data generation procedure requires three basic parameters: the number of suppliers n, the number of customers m, and the number of crossdocks l. The time horizon is fixed at 48 hours (2 days) in the test sets; note that this is usually the longest-time shipments by railways between two cities. The n start points of supplier i time window b_i^r $(1 \leq i \leq n)$ were then randomly generated from a uniform distribution $U[0, 12]$. The end points of supplier i time window e_i^r were also randomly generated from a uniform distribution $U[12, 36]$. For customers, their time windows are generated as $[b_k^a, e_k^a]$, where $b_k^a \sim U[12, 24]$ and $e_k^a \sim U[24, 48]$. The number of fixed transportation schedules between two points is randomly generated in the interval [6, 8]. Meanwhile, the beginning time of the first fixed transportation schedule from supplier to crossdock is generated according to penalty cost and transportation cost, so the fixed schedule is as $[begin, end]$, where $begin \sim U[(b_i^r \times P_i)/(P_i - c_{i,j,1}), 12]$ and $end \sim U[13, 24]$, which means that a higher penalty cost provides a supplier with a higher motivation to ship out cargos. Other schedules are generated as $[begin, end]$, where $begin \sim U[the first\ schedule\ begin\ time, 12]$ and $end \sim U[13, 24]$. Similarly, the arrival time of the last delivery schedule for customers is generated according to penalty cost and delivery cost, so the time window of the last delivery schedule is as $(begin, end)$, where $begin \sim U[12, 35]$ and $end \sim U[36, (e_k^a \times (c'_{j,k,\gamma'_{j,k}} + P'_k))/P'_k]$. For others, the time window is as $[begin,\ end]$, where $begin \sim U[12, 35]$ and end: $U[36$, the last schedule arrival time$]$. Next, because pickups usually follow deliveries within short times, we take the inventory cost at crossdocks to be small relative to transportation costs. This reflects the fact that handling costs are usually smaller than transportation costs. Based on this, the transportation cost per unit cargo of each fixed scheduled route is uniformly generated in the interval [10, 30] and inventory handling cost per unit per hour is uniformly generated in the interval [1, 3], which on average is 1/10 of transportation cost. The penalty cost is set to be relatively higher compared to the transportation cost, which can enforce suppliers and customers to deliver cargoes on time, so the penalty cost per unit cargo is uniformly generated in the interval [30, 90]. Lastly, the amount of supplied cargo s_i (demanded cargo d_k) is uniformly generated in the interval [100, 500]. The capacity of each crossdock is set to $\alpha\Sigma_{1\leq i\leq n} s_i$, where α is randomly generated from a uniform distribution $U[0.5, 0.8]$. Also the capacity of each route is set to $\beta\Sigma_{1\leq i\leq n}(s_i/n)$, where β is randomly generated from a uniform distribution $U[2, 5]$. The following values of parameters are used: # max_iter = 10^4, #terminate_iter = 100, #pop = 200, and #crossover = 80. The mutation probability P_{im} is taken to be 0.02, which is proved to be effective in experiments.

4.2. The Results and Analysis

Based on the number of suppliers, crossdocks, and customers, we designed three categories of problem instances to test HGA: small, medium, and large scale. The results are presented in Tables 1, 2 and 3. Each category has 40 test instances, sorted into 8 groups where each group has 5 instances. The first row of each table specifies the instance size. $n \times l \times m$ denotes that there are n suppliers, l crossdocks, and m customers for this instance group. The rest of each table provides the computational result of CPLEX and HGA. For each instance, the following key values were reported: the average objective value, the average computational

Table 1: Result of CPLEX and HGA on random instances with small scale.

Problem size		10×4×10	12×4×12	14×4×14	16×4×16	18×4×18	10×6×10	12×6×12	14×6×14
CPLEX	LBs	104818	105058	139556	167451	144959	114548	104734	145931
	Objective	111094	107805	142217	174498	152658	123788	108595	155670
	Time(s)	>3600	>3600	3301.1	>3600	>3600	>3600	>3600	>3600
	Gap	5.52%	2.21%	1.85%	4.14%	4.02%	6.24%	3.36%	5.71%
HGA	Objective	110834	107808	142124	173815	150214	122921	108113	154486
	Time(s)	472.93	781.9	530.79	922.69	562.30	726.26	608.8	1084.69
	Gap	5.30%	2.21%	1.80%	3.79%	2.81%	5.59%	2.97%	5.14%

Table 2: Result of CPLEX and HGA on random instances with medium scale.

Problem size		20×4×20	22×4×22	16×6×16	18×6×18	20×6×20	16×8×16	18×8×18	16 × 10 × 16
CPLEX	LBs	153165	197595	143037	161171	185072	131554	149797	132087
	Objective	158295	202441	154712	169924	195823	142704	161564	150179
	Time(s)	>5000	>5000	>5000	>5000	>5000	>5000	>5000	>5000
	Gap	2.64%	2.37%	7.48%	4.95%	5.17%	7.74%	6.90%	12.06%
HGA	Objective	157928	202398	153922	167337	194017	138126	159152	147352
	Time(s)	1080.17	1741.77	1139.11	677.49	1326.22	675.84	1224.98	609.23
	Gap	2.48%	2.34%	6.97%	3.69%	4.40%	4.40%	5.71%	10.22%

time, the gaps between value attained by both CPLEX and HGA, and the low bound attained by CPLEX when terminated.

(1) Small-size instances: the results are shown in Table 1. In this category, eight small scale instance groups are generated with the size n and m ranging from 10 to 14, and l ranging from 4 to 6. We use these instances to compare the performance of the CPLEX solver and HGA. We find that, only in one group, CPLEX solver reaches the LB within time limit set as 3600 s; for other cases, CPLEX fails to get the better solutions within 3600 s comparing HGA, which gets better solutions more quickly, and of which the average gaps are apparently smaller than CPLEX solver.

(2) Medium-size instances: the results are reported in Table 2. In this category, eight instance groups with the size n and m ranging from 16 to 22 and l ranging from 4 to 10 are tested. Time limit is set to more than 5000 s. Also CPLEX fails to get the better solutions within the time limit for all the instance groups, while HGA performs well in no more than 1200 s.

(3) Large-size instances: the results can be found in Table 3. In this category, large-scale instance groups are generated and categorized into 8 groups with the size n and m ranging from 20 to 24 and l ranging from 8 to 12. The CPLEX solver is unable to obtain the better solutions within the time limit, which is set to 7200 s; only in one group CPLEX gets a better solution than HGA. However, HGA can attain much better solutions in no more than 1800 s in the other seven cases.

All the three categories of 24 instance groups show that HGA performs fairly well and is preferable over the commercial CPLEX solver.

The main feature of our proposed HGA is to integrate variable neighborhood search (VNS) into a general GA framework so that it has the ability to get better solutions, especially

Table 3: Result of CPLEX and HGA on random instances with large scale.

Problem size		20×8×20	22×8×22	24×8×24	18 × 10 × 18	20 × 10 × 20	22 × 10 × 22	22 × 12 × 22	24 × 12 × 24
CPLEX	LBs	179309	189188	196688	1495661	162114	170686	208110	192499
	Objective	195510	207942	217234	160386	186614	194127	232170	227976
	Time(s)	>7200	>7200	>7200	>7200	>7200	>7200	>7200	>7200
	Gap	7.68%	8.64%	9.20%	6.38%	13.12%	11.80%	10.36%	14.95%
HGA	Objective	194431	204953	216949	160445	183441	189251	227295	221557
	Time(s)	1159.11	1718.59	1696.08	1314.28	1298.37	1316.45	1419.08	1545.45
	Gap	7.30%	7.34%	9.07%	6.43%	11.71%	9.77%	8.44%	12.96%

Table 4: Gap between HGA and GA.

Problem size		30×10×30	34×10×34	36×10×36	40×10×40	36×15×36	40×15×40
HGA	Objective	261081	299320	296109	347677	306791	334069
	Time(s)	1365.87	1626.23	1801.35	1661.80	1738.46	1823.73
GA	Objective	280881	309925	314313	355154	334459	357053
	Time(s)	1338.44	1303.33	1405.94	1691.24	1332.00	1386.86
Gap		7.58%	3.54%	6.15%	2.15%	9.02%	6.88%

for large-scale problem instances. Hence, by comparing the results of HGA and GA without VNS, we can identify how much the solutions can be improved for large-size problem instances. The numerical results are reported in Table 4, where each problem size has 5 instances, and Gap is defined by

$$\frac{\text{Objective\_GA} - \text{Objective\_HGA}}{\text{Objective\_HGA}} * 100\%. \tag{4.1}$$

The results show that our proposed HGA provides better solutions without sacrificing much computational efforts compared with the GA. Specifically, the results show that HGA outperforms GA for all the cases in terms of solution quality. Although the speed of HGA is slower than GA, it is reasonable because HGA requires more time to search for a better solution by applying VNS. This gives us a clearer idea of the performance of the proposed HGA for the large-sized problems. That is, in general, HGA can provide high-quality solutions in realistic timescales for large-size problems.

Note that our problem has many characteristics (e.g., fixed transportation schedules, inventory capacity, etc.) different from the vehicle routing problem (VRP), although the VRP also considers how to find an optimal transportation scheme to satisfy customer demands. The heuristic algorithms that can be very effective for the VRP cannot be applied to our problem because these two types of problems have different structures and constraints.

5. Conclusions

In this paper, we consider multiple crossdocks problem through fixed transportation schedules with time windows, capacity, and penalty. The objective is to minimize the total costs including shipment costs, penalty cost, and inventory cost. Since we prove that the problem is $\mathcal{NP}$-hard in the strong sense, we focus on developing an efficient heuristic algorithm. Based on the problem structure, we propose an HGA to solve the problem efficiently. In HGA, we employ two vectors (two sequences) to represent a solution including crossdock assignment and route assignment, and we use a greedy method to generate initial solutions that can help the solutions achieve feasibility. We apply variable neighborhood search to eliminate the limitations caused by greedy method and accelerate convergence rate of HGA. Experiments are conducted by using a wide range of test data sets that reflect various realistic scenarios with different problem sizes. Computational results demonstrate that HGA is preferable over the commercial CPLEX Solver.

Our main contribution is threefold. First, the problem we consider represents a class of transshipment problems that arise from real-world applications, which may help industrial practitioners to improve the transshipment decisions within multiple crossdock networks. Second, we set up an integer programming model for this special problem and show that its complexity is strongly $\mathcal{NP}$-hard, which implies that it is unlikely to find a polynomial or pseudopolynomial time algorithm to solve the problem unless $\mathcal{P} = \mathcal{NP}$. Third, we propose a hybrid genetic algorithm integrating greedy technique and variable neighborhood search method that exploits the problem structure and is able to solve the problem effectively and efficiently. The proposed heuristic sheds light on solving many other related complex multiple crossdocks problems.

There are a few directions for further research. Firstly, it may be worthwhile to consider different cost structures, for example, discounted transportation costs based on the shipping amount. Secondly, lateral transportation between crossdocks may be considered. Finally, the current problem can be extended to the multicommodity consolidation problem with repacking consideration, in which various types of goods or freight are considered in a given supply chain transshipment network, as well as the packing problem.

Appendix

Proof of Proposition 2.1

We provide a reduction of the strongly $\mathcal{NP}$-complete *3-partition* problem: given positive integers, w, D, and $\Gamma = \{1, 2, \ldots, 3w\}$ with positive integer values $\gamma(i)$ where, for each $i \in \Gamma$, $\sum_{i\in\Gamma} \gamma(i) = wD$ and $D/4 < \gamma(i) < D/2$ for $i \in \Gamma$, can Γ be partitioned into w disjoint sets $\Gamma_1, \Gamma_2, \ldots, \Gamma_w$ such that $|\Gamma_k| = 3$ and $\sum_{i\in\Gamma_k} \gamma(i) = D$ for $k = 1, \ldots, w$? From an arbitrary instance of *3-partition*, we consider a polynomial reduction to an instance of our multiple crossdocks problem and ask if there exists a feasible solution whose objective value is no greater than $2wD$. For w suppliers given in Σ (let $\Sigma = \{1, 2, \ldots, w\}$) and $3w$ customers in Δ (let $\Delta = \Gamma$), let s_i be the supply and $s_i = D$ for $i \in \Sigma$ with unit penalty cost 3, while for each $k \in \Delta$, let $d_k = \gamma(k)$ be the demand also with unit penalty cost 3. Exactly one crossdock, χ, say, with inventory holding cost 1 per unit product per time, exists linking suppliers with customers. For each supplier i ($i \in \Sigma$), there is only one fixed transportation schedule $(i, i + 1)$ with unit transportation cost 1. On the other hand, for each customer k ($k \in \Delta$), there is w fixed transportation schedule $\{(q + 1, q + 2) : q = 1, \ldots, w\}$ connected with crossdock χ also with unit shipping cost 1.

We now show that a feasible schedule exists whose objective value is no greater than $2wD$ if and only if the *3-partition* has a feasible solution. On the one hand, if *3-partition* has a feasible solution $\Gamma_1, \ldots, \Gamma_w$, note that we needs pay attention to the single shipping and single delivery condition, and hence we should ship all goods provided by supplier i ($i \in \Sigma$) to χ

through fixed schedule $(i, i+1)$, respectively, and transship all of them to customer j $(j \in \Gamma_i)$ through $(i+1, i+2)$, which satisfies the demand $\gamma(j)$ for customer j $(j \in \Gamma_i)$ exactly. It is easy to verify that such a schedule is feasible and total cost is $2wD$. On the other hand, if a feasible schedule exists with objective no greater than $2wD$, then it is optimal since it is easy to prove that $2wD$ is the lower bound of our instance, whose reason is because the total transportation cost is $2wD$ at least, and if any cargo is delayed in crossdock or any demand is unfulfilled, then the total cost is definitely greater than $2wD$. Hence, this optimal solution must satisfy the following two conditions: (1) there is no inventory in crossdock at any time; (2) no penalty cost is incurred. We can then construct a partition by setting Δ_i to be the subset of $k \in \Delta$ whose demand is satisfied by supplier i for $1 \leq i \leq w$. Because of conditions (1) and (2), the demand of customer k $(k \in \Delta)$ is $\gamma(k)$ which should be satisfied immediately by fixed schedule $(i+1, i+2)$. Moreover, because of the single shipping and single delivery condition, we have $\sum_{k \in \Delta_i} d_k = D$. Since $D/4 < d_k < D/2$ for $k \in \Delta$, we have $|\Delta_i| = 3$. Hence, $\Delta_1, \ldots, \Delta_w$ is a feasible partition for the instance of *3-partition* and this completes the proof.

Acknowledgments

The authors wish to acknowledge one anonymous referee whose comments have helped to greatly improve this paper. This research was supported in part by NSFC (70802052, 70701039, 71072090), MOE (NCET-10-0712, NCET-10-0847), the Fundamental Research Funds for the Central Universities (2010221025, 10wkpy20), the Academic Outstanding Youthful Research Talent Plan of Fujian Province (JA10001S), and Soft Science Projects of Fujian Province (2011R0081).

References

[1] D. Simchi-Levi, P. Kaminsky, and E. Simchi-Levi, *Designing and Managing the Supply Chain*, McGraw-Hill—Irwin, Boston, Mass, USA, 2nd edition, 2003.

[2] T. Brockmann, “21 Warehousing trends in the 21st century,” *IIE Solutions*, vol. 31, pp. 36–40, 1999.

[3] Y. Li, A. Lim, and B. Rodrigues, “Crossdocking—JIT scheduling with time windows,” *Journal of the Operational Research Society*, vol. 55, no. 12, pp. 1342–1351, 2004.

[4] M. Gumus and J. H. Bookbinder, “Cross-docking and its implication in location-distribution systems,” *Journal of Business Logistics*, vol. 25, no. 2, pp. 199–228, 2004.

[5] K. Krishnan and V. Rao, “Inventory control in N warehouses,” *Journal of Industrial Engineering*, vol. 16, pp. 212–215, 1965.

[6] U. S. Kamarkar and N. R. Patel, “The one-period n-location distribution problem,” *Naval Research Logistics*, vol. 24, no. 4, pp. 559–575, 1977.

[7] U. S. Karmarkar, “The multilocation multiperiod inventory problem: bounds and approximations,” *Management Science*, vol. 33, no. 1, pp. 86–94, 1987.

[8] G. Tagaras, “Effects of pooling on the optimization and service levels of two-location inventory systems,” *IIE Transactions*, vol. 21, no. 3, pp. 250–257, 1989.

[9] L. W. Robinson, “Optimal and approximate policies in multiperiod, multilocation inventory models with transshipments,” *Operations Research*, vol. 38, no. 2, pp. 278–295, 1990.

[10] N. Rudi, S. Kapur, and D. F. Pyke, “A two-location inventory model with transshipment and local decision making,” *Management Science*, vol. 47, no. 12, pp. 1668–1680, 2001.

[11] J. Grahovac and A. Chakravarty, “Sharing and lateral transshipment of inventory in a supply chain with expensive low-demand items,” *Management Science*, vol. 47, no. 4, pp. 579–594, 2001.

[12] Y. T. Herer and M. Tzur, “The dynamic transshipment problem,” *Naval Research Logistics*, vol. 48, no. 5, pp. 386–408, 2001.

[13] Y. T. Herer, M. Tzur, and E. Yücesan, “Transshipments: an emerging inventory recourse to achieve supply chain leagility,” *International Journal of Production Economics*, vol. 80, no. 3, pp. 201–212, 2002.

[14] S. Axsäter, “A new decision rule for lateral transshipments in inventory systems,” *Management Science*, vol. 49, no. 9, pp. 1168–1179, 2003.

[15] S. Axsäter, “Evaluation of unidirectional lateral transshipments and substitutions in inventory systems,” *European Journal of Operational Research*, vol. 149, no. 2, pp. 438–447, 2003.

[16] H. Donaldson, E. L. Johnson, H. D. Ratliff, and M. Zhang, “Schedule-driven crossdocking networks,” Research Report, Georgia Institute of Technology, 1999.

[17] D. H. Ratcliff, J. van de Vate, and M. Zhang, “Network design for load-driven cross-docking systems,” Research Report, Georgia Institute of Technology, 1999.

[18] Z. Miao, A. Lim, and H. Ma, “Truck dock assignment problem with operational time constraint within crossdocks,” *European Journal of Operational Research*, vol. 192, no. 1, pp. 105–115, 2009.

[19] N. Boysen, M. Fliedner, and A. Scholl, “Scheduling inbound and outbound trucks at cross docking terminals,” *OR Spectrum*, vol. 32, no. 1, pp. 135–161, 2009.

[20] A. Lim, Z. Miao, B. Rodrigues, and Z. Xu, “Transshipment through crossdocks with inventory and time windows,” *Naval Research Logistics*, vol. 52, no. 8, pp. 724–733, 2005.

[21] P. Chen, Y. Guo, A. Lim, and B. Rodrigues, “Multiple crossdocks with inventory and time windows,” *Computers and Operations Research*, vol. 33, no. 1, pp. 43–63, 2006.

[22] H. Ma, Z. Miao, A. Lim, and B. Rodrigues, “Crossdocking distribution networks with setup cost and time window constraint,” *Omega-International Journal of Management Science*, vol. 39, no. 1, pp. 64–72, 2011.

[23] L. Hai and A. Lim, “A metaheuristic for the pickup and delivery problem with time windows,” *International Journal on Artificial Intelligent Tools*, vol. 12, no. 2, pp. 173–186, 2003.

[24] M. Gendreau, F. Guertin, J. Y. Potvin, and R. Séguin, “Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries,” *Transportation Research Part C*, vol. 14, no. 3, pp. 157–174, 2006.