Journal of Manufacturing Systems 40 (2016) 105–118

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

journal homepage: www.elsevier.com/locate/jmansys

A bi-objective model for integrated scheduling of production and distribution in a supply chain with order release date restrictions

Negin Jamili, Mohammad Ranjbar*, Majid Salari

Department of Industrial Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran

ARTICLE INFO

Article history:
Received 24 October 2015
Received in revised form 7 April 2016
Accepted 9 June 2016

Keywords:
Supply chain management
Scheduling
Routing
Bi-objective optimization

ABSTRACT

We study an integrated production and distribution scheduling model in a supply chain. In this problem, there is a set of orders from several customers that has to be processed on a single machine located in a factory. Each order has a release date imposed by the corresponding supplier for delivering the order. In the factory, orders must be batched and routed to be delivered to the corresponding customers. We propose a bi-objective model in order to find a joint schedule of production and distribution to optimize the customers' service level, measured as the mean delivery time and the total transportation cost. An integer linear programming model, in which a weighted sum of both objectives is converted to a single objective, is developed and solved. In addition, two local search algorithms and a metaheuristic algorithm based on tabu search are developed to solve large-scale problems at reasonable times. Moreover, two bi-objective solution methods are developed to find the Pareto solutions for the bi-objective problem. Finally, the performances of all developed solution approaches are analyzed and compared using randomly generated test sets and managerial insights are drawn from multiple numerical experiments.

1. Introduction

Depending on the number of stages in a supply chain, the supplier, manufacturer, and customers may be involved as decision makers. Supply chain activities cover the transportation of raw material, production processes and the transportation of finished goods. The products' cost consists of those imposed by converting raw materials into finished goods and the logistics expenditures such as delivering and services to customers. In particular, the main issue in the supply chain management is the coordination between decisions made at different stages, and supply chain scheduling integrates the three stages of material supply, production arrangement and product delivery to achieve optimal performance of the system through coordination of these stages. The management of the supply chain and the coordination between all stages usually lead to more savings in comparison to the case where each stage is managed separately. As a result, companies require an integrated plan of all activities in the supply chain.

Traditional scheduling models deal with the production scheduling problem without taking into account the other stages such as transportation. In these models, the assumption is that an unlimited number of vehicles are available for delivering finished products to the related customers, so that the products can be transported to customers without any delay. However, in several real world applications, the number or the capacity of the vehicles may be limited and, to increase their utilization, vehicles may need to distribute the products of multiple customers in a single trip.

The literature on the supply chain management can be divided into two main categories. Essentially, some studies focus on the two stages of a supply chain such as manufacturer and customers, while others deal with all three stages. Although there is an extensive literature on scheduling and vehicle routing problems, the integration of these problems has been rarely studied. Due to the complexity of the three-stage supply chain, publications in this area usually do not consider the vehicle routing problem. Chang and Lee [1] address a single/parallel machine scheduling problem, in which a vehicle with limited loading capacity delivers all the orders processed by the machine(s) to one or two customers. All orders require different amounts of storage space during delivery and the problem is to find a schedule for processing and delivering orders to minimize the last order's arrival time at its destination. Most articles published in this field consider a single machine manufacturing system. Selvarajah and Steiner [2] study a two-stage supply chain with a single machine system, considering the inventory holding cost and delivery cost for each batch. The objective is to minimize the costs incurred by batch scheduling and delivering to the customers. The problem proposed in [3] involves earliness, tardiness

* Corresponding author. Tel.: +98 9155591612.
E-mail addresses: negin.jamili@stu.um.ac.ir (N. Jamili), m_ranjbar@um.ac.ir (M. Ranjbar), msalari@um.ac.ir (M. Salari).

http://dx.doi.org/10.1016/j.jmsy.2016.06.005
0278-6125/

and inventory cost in a single machine and batch delivery problem. Condotta et al. [4] describe a metaheuristic method based on tabu search for combined production-transportation problem, in which orders have to be processed on a single machine and are delivered to a customer.

Moreover, there are studies that consider other machine environments such as parallel machine or flow shop. The proposed parallel machine scheduling problem by Wang and Cheng [5] focuses on optimizing the customer's service level and transportation costs. In this study, both job class setups for job processing and job delivery are considered at the same time. Lee and Chen [6] study scheduling problems in manufacturing and distribution systems with parallel machine environment and considering transportation of semi-finished orders, which is done between machines at the production site, and final products to the customer or warehouse. Soukhal et al. [7] investigate two-machine flow shop scheduling problem with a material handling system that considers the transferring of orders from processing facility to customers. Finally, Cakici et al. [8] focus on a two-stage supply chain problem in a single machine environment to optimize the trade-off between total weighted tardiness and distribution cost and provide a multi-objective analysis of the problem.

There are several publications which consider multiple customers and delivery routing decisions while delivering. Li et al. [9] study a single-machine scheduling problem in which all orders belong to one or two customers and the objective is to minimize the total orders' arrival time. Chen and Vairaktarakis [10] consider two machine environments, i.e. single machine and parallel machine, in the processing facility. The proposed objective function takes into account both arrival time and distribution cost. For the integrated production and distribution scheduling problem in the parallel machine environment, Chang et al. [11] present a problem in which the goal is to find a schedule that minimizes the overall cost, consisting of the total weighted job delivery time and the distribution cost. Ullrich [12] also adds some assumptions to a two-stage supply chain problem in which machine-dependent ready times, job-dependent processing times, delivery time windows, and service times were taken into account. The goal of the problem is to optimize the total tardiness of a production-distribution schedule proposed for a set of orders on parallel machines.

There are a few articles considering all three stages of a supply chain. Hall and Potts [13] address a problem consisting of a supplier, several manufacturers and several customers. The objective is to minimize the overall delivery cost and scheduling cost based on the orders' delivery time. The objective is achieved by scheduling the orders and forming them into batches, each of which is delivered to the next downstream stage. In the three-stage supply chain proposed in [14], the warehouse, factory and the customer are located at three different sites and the objective is to minimize the arrival time of the last delivered order to the customer. In [15], a logistic scheduling model is formulated where a manufacturer receives raw materials from a supplier, processes them on a single machine and delivers the finished product to a customer. The objective is to minimize the total work-in-process inventory and transportation costs by planning the size, arrival times and departure times of supply and delivery batches, respectively. It is worth mentioning that routing is not considered in any of these three-stage problems.

Another group of studies in supply chain management consist of coordination of product recovery and production processes. This field of study, namely closed-loop supply chain (CLSC), aims at integrating return products in the traditional supply chain process in order to save disposal cost and reduced environmental dilemmas. As regards these researches, there are some article connected with CLSC optimization. Priyan and Uthayakumar [16] address a problem to optimize the order quantity lead time and total number of deliveries in a supply chain, with the objective of minimizing system total cost. In [17] a supply chain model based on simulation and multi-objective optimization is proposed to optimize control policies for multi-echelon supply chain with returned products.

According to the literature review, all the integrated problems of scheduling and supply chain addressing routing decisions, consider only two stages of a supply chain. In this paper, we propose a three-stage logistic scheduling model in which a manufacturer receives raw materials from a supplier, processes them on a single machine, batches and distributes the finished products to a set of known customers using an unlimited number of capacitated vehicles. The objective is to minimize the mean delivery time and the transportation cost by determining the best production scheduling and routing of orders to the related customers. We formulate the problem as a bi-objective model and propose several heuristic algorithms. In the first algorithm, the two objective functions are integrated and converted into a single objective function using the weighted-sum method. This case has been modelled as a linear integer programming problem in Section 2 and a set of heuristic algorithms and a tabu search (TS) metaheuristic solution approach are developed to solve the problem (see Section 3). In Section 4, we develop several bi-objective heuristic algorithms based on the combination of the developed tabu search algorithm with the multi-objective adaptive memory programming method and the adaptive weighted-sum method. Section 5 is dedicated to the computational experiments. Finally, conclusions and future research directions are provided in Section 6.

2. Problem statement and formulation

The problem involves a production stage with a single machine environment which processes the orders to be transported to related customers. At the distribution stage, there is an unlimited number of homogeneous vehicles with a capacity constraint, to deliver the prepared orders. As there are several suppliers providing the material required for the production part, for each order a release date is considered; therefore, an order cannot be processed before its release date. We consider a set of orders $\boldsymbol{J} = \{1, 2, \ldots, n\}$ from k customers ($k \leq n$) located at different locations. Each order $j \in \boldsymbol{J}$ has a processing time p_j and a release date r_j imposed by the supplier $s \in \{1, 2, \ldots, \eta\}$ ($\eta \leq n$). So in each feasible solution, the process of each order $j \in \boldsymbol{J}$ cannot be started before r_j. In the transportation part, due to the capacity of each vehicle, the number of orders to be delivered in each trip of each vehicle cannot exceed the vehicle capacity (w). So all the orders are divided into a set of batches $\boldsymbol{B} = \{1, 2, \ldots, n\}$ where each batch $b \in \boldsymbol{B}$ constitutes the orders of a subset of customers which has to be transported by a dedicated vehicle in a single trip. The assumption of unlimited number of vehicles to deliver the orders is realistic when third-party logistics companies participate in the transportation and distribution stage of the supply chain. Departure time of each batch $b \in \boldsymbol{B}$, denoted by t_b, indicates the latest completion time of the orders dedicated to that batch. In addition, d_j denotes the delivery time of order j to its corresponding customer and can be obtained based on the route taken by the corresponding vehicle. The transportation cost for each batch consists of a fixed cost f and a variable cost dependent on the total distance of the route travelled by the vehicle. Finally, the travelling time (cost) from the customer i to the customer j is denoted by τ_{ij} (c_{ij}) in which a zero index indicates the manufacturer location. The problem is to find a schedule minimizing 1) the average of delivery times (D_{mean}), and 2) the distribution cost (T).

In this model, the two objectives are integrated and converted into a single-objective problem by using the parameter α ($0 < \alpha < 1$) which represents the decision maker's preference on the objectives.

Table 1
Description of parameters and decision variables of the problem.

Parameters	Definition
$\boldsymbol{J}=(1,2,\ldots,n)$	Set of orders with indices i and j.
$\boldsymbol{B}=(1,2,\ldots,n)$	Set of potential delivery batches with index b.
p_j	Processing time of order j.
τ_{ij}	Travelling time from the customer of order i to the customer of order j (zero index indicates the manufacturer location).
c_{ij}	Cost of travelling from the customer of order i to the customer of order j (zero index indicates the manufacturer location).
w	Capacity of each delivery truck.
f	Fixed cost incurred by each batch.
M	A sufficiently large number.
Variables	**Definition**
x_{jb}	Binary variable that takes the value 1 if order j is delivered in the batch b and takes 0, otherwise.
β_b	Binary variable which takes the value 1 if there is at least one order dedicated to batch b.
z_{ijb}	Binary variable which takes the value 1 if order j is delivered immediately after order i in the b^{th} batch. Also, z_{0jb} (z_{j0b}) takes the value 1 if order j is the first (last) order delivered in batch b.
s_{ij}	Binary variable which takes the value 1 if order j is processed immediately after order i. Also, s_{0j} ($s_{j,n+1}$) takes the value 1 if order j is the first (last) order which is processed on the machine.
y_j	Completion time of order j.
d_j	Delivery time of order j.
t_b	Departure time of batch b.
u_{ijb}	Auxiliary non-negative integers defined for sub-tour elimination in distribution route of batch b, including orders i and j.

The required parameters and decision variables are represented in Table 1.

The mathematical formulation of the problem reads as follows.

$$Min \quad \alpha \times \left(\frac{1}{n}\sum_{j\in \boldsymbol{J}} d_j\right) + (1-\alpha)\times\left(f\sum_{b\in \boldsymbol{B}}\beta_b + \sum_{i\in \boldsymbol{J}\cup\{0\}}\sum_{j\in \boldsymbol{J}} c_{ij}\sum_{b\in \boldsymbol{B}} z_{ijb}\right) \tag{1}$$

subject to:

$$\sum_{b\in \boldsymbol{B}} x_{jb} = 1 \quad \forall j \in \boldsymbol{J} \tag{2}$$

$$\beta_b \le \sum_{j\in \boldsymbol{J}} x_{jb} \quad \forall b \in \boldsymbol{B} \tag{3}$$

$$\sum_{j\in \boldsymbol{J}} x_{jb} \le w\beta_b \quad \forall b \in \boldsymbol{B} \tag{4}$$

$$\beta_{b+1} \le \beta_b \quad \forall b \in \boldsymbol{B}\backslash\{n\} \tag{5}$$

$$d_j \ge t_b + \tau_{0j} - M(1-x_{jb}) \quad \forall j \in \boldsymbol{J}, b \in \boldsymbol{B} \tag{6}$$

$$d_j \ge d_i + \tau_{ij} - M(1-z_{ijb}) \quad \forall i,j \in \boldsymbol{J}, b \in \boldsymbol{B} \tag{7}$$

$$x_{jb} = \sum_{i\in \boldsymbol{J}\cup\{0\}} z_{ijb} \quad \forall j \in \boldsymbol{J}, b \in \boldsymbol{B} \tag{8}$$

$$x_{jb} = \sum_{i\in \boldsymbol{J}\cup\{0\}} z_{jib} \quad \forall j \in \boldsymbol{J}, b \in \boldsymbol{B} \tag{9}$$

$$\beta_b = \sum_{j\in \boldsymbol{J}} z_{0jb} \quad \forall b \in \boldsymbol{B} \tag{10}$$

$$y_j \ge y_i + p_j - M(1-s_{ij}) \quad \forall i,j \in \boldsymbol{J}, \quad i \ne j \tag{11}$$

$$y_j \ge s_{0j}\times p_j \quad \forall j \in \boldsymbol{J} \tag{12}$$

$$y_j \ge r_j + p_j \quad \forall j \in \boldsymbol{J} \tag{13}$$

$$t_b \ge y_j - M(1-x_{jb}) \quad \forall j \in \boldsymbol{J}, b \in \boldsymbol{B} \tag{14}$$

$$\sum_{i=1\&i\ne j}^{n+1} s_{ji} = 1 \quad \forall j \in \boldsymbol{J}\cup\{0\} \tag{15}$$

$$\sum_{i=0\&i\ne j}^{n} s_{ij} = 1 \quad \forall j \in \boldsymbol{J}\cup\{n+1\} \tag{16}$$

$$\sum_{j\in \boldsymbol{J}} u_{0jb} = \sum_{j\in \boldsymbol{J}} x_{jb} \quad \forall b \in \boldsymbol{B} \tag{17}$$

$$\sum_{i\in \boldsymbol{J}\cup\{0\}} u_{ijb} - \sum_{i\in \boldsymbol{J}\cup\{0\}} u_{jib} = \sum_{i\in \boldsymbol{J}\cup\{0\}} z_{ijb} \quad \forall j \in \boldsymbol{J}, b \in \boldsymbol{B} \tag{18}$$

$$\sum_{j\in \boldsymbol{J}} u_{j0b} = 0 \quad \forall b \in \boldsymbol{B} \tag{19}$$

$$u_{ijb} \le w \times z_{ijb} \quad \forall i,j \in \boldsymbol{J}\cup\{0\}, \quad b \in \boldsymbol{B} \tag{20}$$

$$x_{jb} \in \{0,1\} \quad \forall j \in \boldsymbol{J}, \quad b \in \boldsymbol{B} \tag{21}$$

$$\beta_b \in \{0,1\} \quad \forall b \in \boldsymbol{B} \tag{22}$$

$$z_{ijb} \in \{0,1\} \quad \forall i,j \in \boldsymbol{J}\cup\{0\}, \quad b \in \boldsymbol{B} \tag{23}$$

$$s_{ij} \in \{0,1\} \quad \forall i \in \boldsymbol{J}\cup\{0\}, \quad j \in \boldsymbol{J}\cup\{n+1\} \tag{24}$$

$$y_j, d_j \in \mathbb{Z}^+ \quad \forall j \in \boldsymbol{J} \tag{25}$$

$$t_b \in \mathbb{Z}^+ \quad \forall b \in \boldsymbol{B} \tag{26}$$

$$u_{ijb} \in \mathbb{Z}^+ \quad \forall i,j \in \boldsymbol{J}\cup(0), \quad b \in \boldsymbol{B} \tag{27}$$

The objective function (1) minimizes the weighted sum of the D_{mean} and the T. Constraint (2) ensures that each order is delivered in only one batch. Constraints (3) and (4) describe the relation between x_{jb} and β_b and the restriction imposed by the capacity of the batch. In particular, batch b is active if and only if there is at least one order assigned to it. Constraint (5) guarantees that if there is no order placed in batch b, no orders can be placed in batch $b+1$, as well. Constraints (6) and (7) are related to the delivery times of the orders, in which parameter M is a sufficiently large positive value. Essentially, the delivery time of the first order, sent in each batch, must be greater than or equal to the departure time of that batch plus the required time for a trip from the manufacturer to the related customer. In addition, for each $i,j \in \boldsymbol{J}$ and $b \in \boldsymbol{B}$, if order j is delivered after order i in batch b, then we have $d_j \ge d_i + \tau_{ij}$, otherwise the constraint is redundant. Constraints (8)–(10) define the connectivity of each route. In particular, there exists exactly a single arc entering and a single arc leaving each visited customer in a route (see, Constraints (8) and (9)) and all the routes must be started at the manufacturer location (see (10)). Constraints (11)–(13) indicate the completion time of each order on the processing machine. Essentially, the completion time of an order must be greater than or equal to its processing time, plus the maximum value of its release date and the completion time of its preceding processed order. The departure time of a batch is bounded in Constraint (14) and cannot be less than the maximum completion time of the allocated orders. Constraints (15) and (16) stipulate that each order precedes one order and succeeds another one in the sequence of machine, unless it is the last or the first order to be processed. Constraints (17)–(19) are sub-tour elimination constraints that eliminate illegal sub-tours. These constraints were originally provided by Miller et al. [18] for the

well-known travelling salesman problem. The extended versions of these constraints are used for the vehicle routing problem and the location-routing problem in [19,20], respectively. According to these references, in these constraints, the u_{ijb} play a role similar to that of node potentials in a network and the inequalities involving them serve to eliminate tours that do not begin and end at the manufacturer. Finally, the last seven constraints indicate the type of the variables in which $\mathbb{Z}^+$ displays the set of non-negative integers.

3. Solution approaches for the single-objective problem

The problem considered in this work is NP-hard since it includes the NP-hard vehicle routing problem as a special variant (see, e.g. Lenstra and Kan [21] and Garey and Johnson [22]). As the computational tests show (see Section 5.2), the proposed model is not able to solve the large-size instances. So, in this paper a heuristic algorithm is introduced to solve the problem. In this section, we first show the solution representation followed by an initialization procedure to construct an initial feasible solution. Then, we develop two local search algorithms to improve the quality of the initial solution. In addition, we introduce a TS metaheuristic algorithm to find a solution.

3.1. Solution representation

Each solution s is represented by $\boldsymbol{S}_s = \{b_1^s, b_2^s, \ldots, b_n^s\}$ where $b_j^s \in \{1, \ldots, q'\}$ shows the batch number in which order j is delivered to its corresponding customer and q' is the total number of delivery batches. Since the sequence of delivering batches is based on increasing order of their numbers, the maximum value of b_j^s $(\forall j \in \{1, \ldots, n\})$ represents the total number of active batches. To illustrate the representation of a solution, consider an example with $n = 5$ and $w = 2$. In the initial solution $\boldsymbol{S}_0 = \{1, 1, 3, 2, 2\}$, orders 1 and 2 are assigned to the first batch, orders 4 and 5 are delivered by the second batch and order 3 is transported by the third batch.

3.2. Initialization procedure

In this section, we develop a heuristic procedure to generate an initial feasible solution which is based on the maximum utilization of each batch. We consider $q = \left\lfloor \frac{n}{w} \right\rfloor$ full batches in which $\lfloor x \rfloor$ indicates the greatest integer no larger than x and a single non-full batch if $v = n - w \times q > 0$. In this regard, $q' = \left\lceil \frac{n}{w} \right\rceil$ shows the total number of batches, in which $\lceil x \rceil$ indicates the smallest integer no less than x. In case of existence of a non-full batch (this happens if and only if n is not divisible by w), $q+1$ different solutions are evaluated based on all different possible positions of the non-full batch in the sequence of all the delivery batches. For this purpose, assume the non-full batch number is k where $k = 1, \ldots, q+1$. In order to specify the contents of each batch, the order j^* is selected as the first order of batch b if R_{j^*} has the minimum value among all R_j-values where $R_j = \alpha \times (max(time, r_j) + P_j) + (1-\alpha) \times c_{0j}$, j belongs to set of all unscheduled orders and the *time* parameter indicates the completion time of the last order of batch $b-1$; hence $time = 0$ when $b = 1$. At each step, for selecting the next order we have to increase the *time* parameter to the completion time of order j^* and replace c_{0j} by c_{j^*j}. Starting from $b = 1$ to $b = q+1$, this procedure is repeated until the capacity of batch b is full for each solution. For all the batches, we use the nearest neighbour algorithm (see, Kizilateş and Nuriyeva [23]) to construct the vehicle routes taken for delivering each batch. In Particular, among orders of each batch b, the first delivered order is the one that its corresponding customer has the nearest distance to the factory among other customers of this batch. Next, the second delivered order is the order that its corresponding customer has the nearest distance to the corresponding customer of the first order. As long as routes of all orders are not determined, we follow this algorithm to assign orders to the routes. After calculating the objective function according to the delivery times of the orders and the routes taken for delivering the batches, the solution with the minimum objective function is chosen as the final solution.

In the problems where n is divisible by w, all batches are considered to be delivered with their maximum capacity. This procedure generates one solution. It should also be mentioned that the idea of maximum utilization of vehicles may delete optimal solutions in some test instances.

3.3. Neighbourhood structure

Two kinds of moves, namely, insertion and swap are used to investigate the neighbours of the generated solutions efficiently. In the following, we describe the characteristics of the moves.

(a) *Swap*: a swap move is performed by selecting two orders, belonging to two different batches, and exchanging the batch number of the two orders. Starting from the first order of the first batch, all available orders will be considered for swap. For a given solution $\boldsymbol{S}_s$, in case of considering all possible swap moves at most $T_s = \sum_{b=1}^{q'-1} w'_b \sum_{b'=b+1}^{q'} w'_b$ new solutions will be available as the neighbours of $\boldsymbol{S}_s$, where w'_b is the number of orders assigned to batch b.

(b) *Insertion:* an insertion move is carried out by extracting one order from its current batch and inserting it into another feasible batch. For an order selected from batch b, we change its batch number repeatedly to find ρ other feasible neighbours, where ρ is an even number considered to speed up the algorithm by confining the search space. The search for finding feasible batches for each order is preferably done by converting its batch number to less and also larger numbers till finding $\rho/2$ different lower batch numbers and $\rho/2$ different higher ones, if possible. In special cases where $b = 1$ or $b = q'$, we replace the order in only ρ next or previous batches respectively. By applying this move, new delivery batches may be created in case of not finding ρ feasible active batches and for a solution containing n orders, there will be ρ^n neighbours to be investigated.

According to the definition of moves, searching the neighbourhood by applying insertion move, may lead to solutions containing more than q' batches, while applying swap will not change the total number of batches, so the search space would be limited.

3.4. Local search algorithms

In this section, we develop two local search algorithms, namely $H^f{}_1$ and $H^b{}_1$, based on the first improvement and the best improvement strategies.

3.4.1. Algorithm $H^f{}_1$

In this algorithm, we apply the swap and insertion moves to try to improve the quality of an initial feasible solution. To do so, the algorithm starts by investigating the swap moves. As soon as a move improves the quality of the solution, the corresponding swap leading to this improvement is accepted. The swap move is iterated until no improvement can be achieved by swapping all available orders. Following this step, we apply the insertion move. In particular, as soon as a move improves the cost of the solution, it will be accepted and the swap move is applied to the new found solution. This procedure iterates as long as no improvement can be found in the cost of the solution. The algorithm starts by using an initial solution generated by applying the heuristic method described in Section 3.2 and the whole procedure stops whenever we have no

improvement by applying the swap and insertion procedures. The general framework of this procedure is depicted in Algorithm 1.

Algorithm 1. Pseudo-code of the algorithm $H^f{}_1$

Input Data
$\boldsymbol{S}_0 = \{b_1^0, b_2^0, \ldots, b_n^0\}$: initial solution;
F_0: the objective value of $\boldsymbol{S}_0$;
T_0: total number of $\boldsymbol{S}_0$'s neighbours by considering swap moves;
Begin
SWAP:
$g = 1$;
calculate T_0;
while $(g \leq T_0)$
  Find the gth neighbourhood of $\boldsymbol{S_0}$ by applying the swap move;
  if $(F_g < F_0)$
    $\boldsymbol{S_0} \leftarrow \boldsymbol{S_g}$;
    $F_0 \leftarrow F_g$;
    Calculate T_0;
    $g = 1$;
  else
    $g = g + 1$;
INSERTION:
$g = 1$;
  Find the gth neighbourhood of $\boldsymbol{S_0}$ by applying the insertion move;
  if $(F_g < F_0)$
    $\boldsymbol{S_0} \leftarrow \boldsymbol{S_g}$;
    $F_0 \leftarrow F_g$;
    goto SWAP;
  else
    $g = g + 1$;
  if there is no neighbour solution
    return $\boldsymbol{S_0}$ as the final solution;
End

3.4.2. Algorithm H^{b_1}

This algorithm follows also the same strategy as that proposed for the $H^f{}_1$. The only difference is that the best neighbourhood leading to an improvement will be implemented when running each of the swap and insertion moves.

3.5. Tabu search algorithm

Proposed by Glover [24], to escape from local optimal solution, the TS algorithm always moves to the best neighbour of the working solution, even if the corresponding neighbour does not lead to an improvement in the objective function. To make TS fully effective, diversification is used as an algorithmic mechanism to ensure a broad exploration of the search space. Intensification strategy is also provided to focus on the promising portions of the search space in order to find local optimal solutions.

The proposed TS algorithm starts from the initial solution generated by utilizing the procedure developed in Section 3.2 and explores its neighbourhood by applying the moves already introduced in Section 3.3. After a specific number of consecutive non-improving iterations, the algorithm decides about applying intensification or diversification phase. In the following, more explanation about different components of the TS algorithm is given. These components are combined based on the general structure depicted in Fig. 1. The required notations are described in Table 2.

Table 2
Notations of the TS algorithm.

Notation	Definition
max_{itr}	The maximum number of consecutive non-improving iterations.
$count_{int}$	Total number of improving iterations.
$count_{div}$	Total number of repeatedly non-improving iterations.
$score^s_{int}$	Score of $\boldsymbol{S}_s$ obtained based on the intensification strategy.
$score^s_{div}$	Score of $\boldsymbol{S}_s$ obtained based on the diversification strategy.
L_T	Fixed length of the tabu lists.

3.5.1. Neighbourhood structure

Choosing a search space and a neighbourhood structure is one of the most critical steps in the design of a TS algorithm [25]. In this paper, the neighbourhood is explored by applying different moves introduced in Section 3.3. In particular, the working solution is always moved to the best non-tabu neighbour achieved by applying the insertion and swap procedures. In other words, in each iteration of this algorithm, both neighbourhoods are fully explored and the move leading to the best non-tabu solution is selected.

3.5.2. Tabu list and aspiration criterion

In our designed TS algorithm, we use two different tabu lists for each of the moves achieved by applying the insertion and swap procedures. In particular, if a new chosen non-tabu move is a neighbour solutions obtained by performing the swap or insertion procedure, related attributes are declared forbidden and will be stored in the corresponding tabu list, dedicated to the swap or insertion respectively. We have also used the aspiration criterion in which a tabu move is accepted if the objective value of the corresponding solution is better than the best-known solution.

3.5.3. Intensification

The intensification strategy is based on scoring all neighbour solutions of the current solution according to their components, i.e. assignment of orders to the batches. In other words, the batching plan for each improving solution is stored in a matrix to be available for further scoring. By saving the components of a solution in each improving iteration, from the beginning of the algorithm, the score of each neighbour solution is calculated. When applying the intensification procedure, the solution with the maximum score is chosen as the current solution.

3.5.4. Diversification

The diversification strategy is almost similar to the developed intensification strategy with some differences in the score calculation. In this procedure, the scores are assigned to the neighbour solutions according to the chosen solution in each iteration, regardless of whether the previous current solution has been an improved solution or not. The solution with the minimum score has components which have been explored rarely in the previous iterations. To implement the diversification procedure, this solution is replaced with the current one.

Let $count_{div}$ and $count_{int}$ represent the total number of non-improving and improving iterations of the algorithm, respectively. If there is no improvement in the best-known solution after max_{itr} number of consecutive iterations, the algorithm decides about applying intensification or diversification procedures, whether $count_{div}$ is less than or greater equal to $count_{int}$, respectively.

4. Solution approaches for the bi-objective problem

In this section we propose two heuristic algorithms for the introduced bi-objective problem in which the weighted sum of the mean delivery time (D_{mean}) and the total distribution cost (T) are considered as the two objectives. The first algorithm is based on the

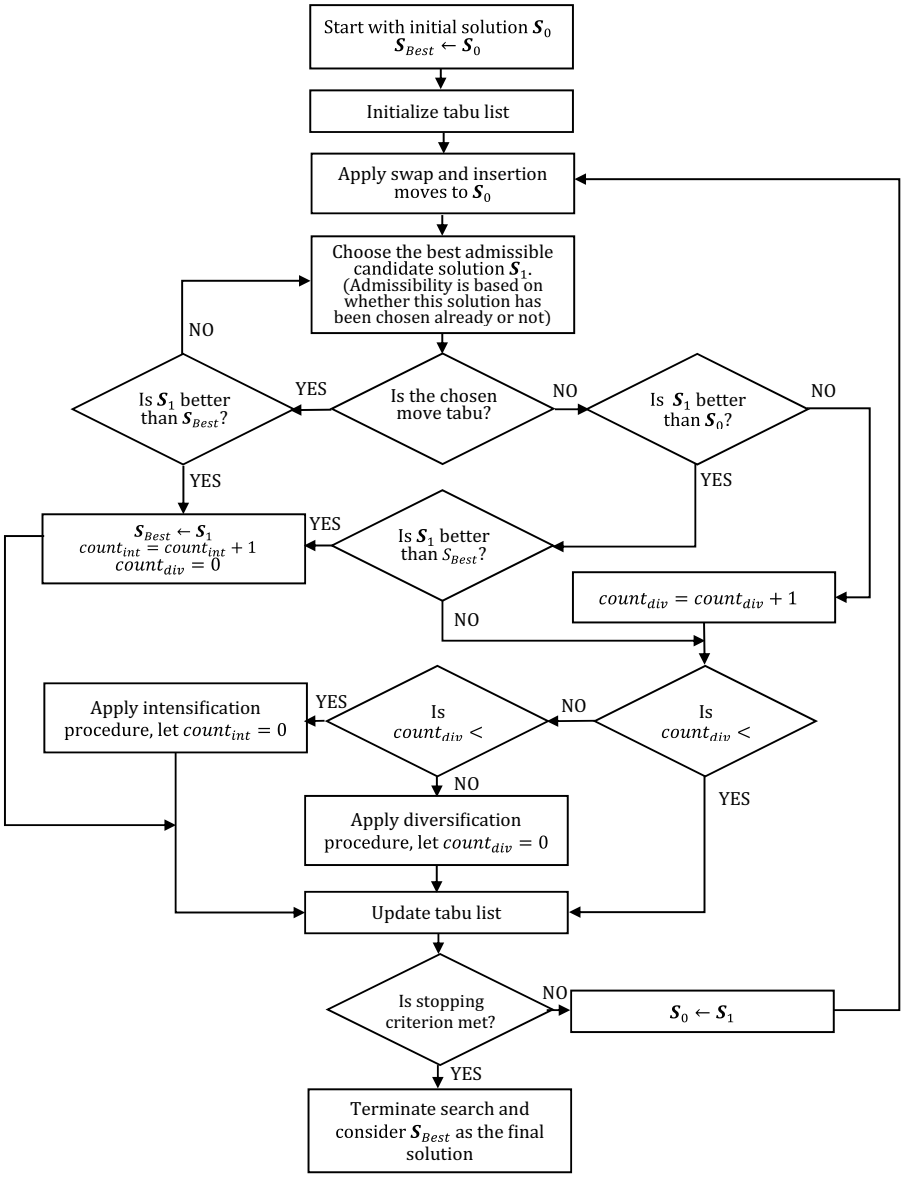

Fig. 1. General structure of the proposed tabu search algorithm.

multi-objective adaptive memory programming (MOAMP) developed by Caballero and Gandibleux [26] while the second algorithm is based on the adaptive weighted-sum approach (AWS), proposed by Kim and De Weck [27]. Some notations used to describe the bi-objective heuristic algorithms are introduced in Table 3.

4.1. Initial solution generation methods

We have developed two initialization procedures based on the minimization of the D_{mean} and T, respectively. In the following, the description of these algorithms is provided in details.

4.1.1. Initial solution generation method based on the minimization of D_{mean}

The initial solution generated with this method considers the minimization of D_{mean} and ignores the transportation cost T. As the required time for delivering an order from the manufacturer to its customer is a fixed distance-based value, this goal is achieved by minimizing the completion time of orders on the single machine and delivering them without any delay. For this purpose, we employ the shortest processing time first rule (SPT) which minimizes the total completion time on a single-machine scheduling problem when all release dates are considered to be zero (see, Pinedo [28]). Since there may be some positive release dates, we apply the SPT rule in each time moment t over all orders $j \in \boldsymbol{J}$ that $r_j \leq t$. It should be noticed that the proposed algorithm does not necessarily lead to an optimal solution, but it can obtain proper solutions in a very short computing time. Since the goal of this procedure is the minimization of D_{mean}, each processed order is transported immediately as a separate batch after its production completion time. So, there would be n batches for delivering n available orders.

4.1.2. Initial solution generation method based on the minimization of T

This method is similar to the greedy algorithm already described in Section 3.2 and by considering $\alpha = 0$ to fully emphasize the transportation cost.

Table 3
Notations of the MOAMP and AWS algorithms.

Notations of MOAMP algorithm	Definition
$\boldsymbol{P}$	Set of non-dominated solutions.
$\lvert\boldsymbol{P}\rvert$	Size of set $\boldsymbol{P}$.
max_{search}	The pre-specified number of iterations for phase 2.
f_1^i	The value of the first objective function (D_{mean}) for the ith solution of $\boldsymbol{P}$.
f_2^i	The value of the second objective function (T) for the ith solution of $\boldsymbol{P}$.
$f_1^{\min}$	The minimum amount of the first objective function during the procedure.
$f_2^{\min}$	The minimum amount of the second objective function during the procedure.
λ	A randomly generated weight factor between 0 and 1.
Notations of AWS algorithm	**Definition**
$\mu_{initial}$	Number of initial segments to start the AWS algorithm.
ε	The minimum allowable amount of Euclidean distance between solutions.
Γ	Constant value for further refinement in the AWS algorithm.
L_i	The Euclidean distance between solutions i and $i+1$.
μ_i	Number of further refinements in section i.
δ	The constant for determination of objectives' limitation.
$Limit\,F_1$	The maximum allowable amount of D_{mean} in search procedures.
$Limit\,F_2$	The maximum allowable amount of T in search procedures.

4.1.3. Random initialization procedure

Similar to the greedy initialization procedure in Section 3.2, in this algorithm there will be at most $q' = \lceil n/w \rceil$ number of batches. In each iteration, the procedure selects a random order and assigns that to a non-full randomly selected batch. The algorithm is repeated until all orders have been assigned to available batches.

4.2. The MOAMP method

Introduced by Caballero and Gandibleux [26] the MOAMP method consists of three phases. The first phase looks for efficient points by approximating the best solutions of the single-objective problems. The second phase consists of performing the TS which employs $f_\lambda = \max\left\{\lambda\frac{f_1^i - f_1^{\min}}{f_1^{\max} - f_1^{\min}}, (1-\lambda)\frac{f_2^i - f_2^{\min}}{f_2^{\max} - f_2^{\min}}\right\}$ as the objective function. In particular, it measures the distance between a trial point and the approximated ideal objective values, to evaluate the merit of each visited solution and to obtain an estimated Pareto front. Finally, the third phase is to search around the approximated Pareto set. The MOAMP method is a conceptual structure and there are different ways to apply it. García et al. [29] used a method based on the variable neighbourhood search to implement the first phase of the MOAMP. In contrast, Molina et al. [30] applied a TS for the first two phases, and a combination of scatter search and TS for the last phase. Our developed implementation of the MOAMP algorithm is almost similar to that proposed by Pacheco et al. [31]. Essentially, they used different forms of the TS to be applied for a bi-objective bus routing problem. The first phase is to provide $f_1^{\min}$ and $f_2^{\min}$, i.e. the minimum amount of each objective function, respectively, by executing two linked TS algorithms. The first one is a multi-start TS, presented in Section 4.2.1, with the objective of minimizing the value of D_{mean} and provides $\boldsymbol{S}_0$ as the final solution. The found solution $\boldsymbol{S}_0$ is considered as an initial solution for the TS algorithm, mentioned in Section 3.5, which tries to search with the objective of finding the minimum value of T. In the next step, the second search is conducted reversely. In other words, the multi-start TS is utilized to find the minimum amount of T and the final solution obtained by this algorithm is used as the starting point for a search with the goal of minimizing the D_{mean}. Implementing the procedures and maintaining all the non-dominated solutions visited during the search leads to an approximation of the efficient frontier ($\boldsymbol{P}$) and also $f_1^{\min}$ and $f_2^{\min}$. In the second phase, additional TS algorithms with the framework of what illustrated in Section 3.5 are implemented to search the neighbourhood of the current Pareto set by using f_λ as a criterion for evaluating the merit of each solution. After generating a random value for λ and finding a solution in $\boldsymbol{P}$ with the minimum amount of f_λ, the search is performed for a pre-specified number of consecutive iterations without any changes in the set $\boldsymbol{P}$. In each iteration, a randomly generated value of λ from uniform distribution $U(0, 1)$ affects the behaviour of the TS which starts from the best solution found in the previous iteration. Finally, in the third phase the algorithm searches around the solutions in $\boldsymbol{P}$ by using the two moves described in Section 3.3. The pseudo code of this method is presented in Algorithm 2.

Algorithm 2. Pseudo-code of the MOAMP algorithm

Input Data
$\boldsymbol{P} = \emptyset$: set of non-dominated solutions;
max_{search}: the number of consecutive iterations with no changes in $\boldsymbol{P}$;
Begin
PHASE 1: Find approximated efficient solution for each objective
Apply multi-start TS algorithm by considering D_{mean} as the objective function;
Apply TS algorithm over the solution obtained in the previous step by considering T as the objective function;
Apply multi-start TS algorithm by considering T as the objective function;
Apply TS algorithm over the solution obtained in the previous step and by considering D_{mean} as the objective function;
Update $\boldsymbol{P}$ with all non-dominated solutions;
PHASE 2: obtaining an approximation of Pareto front
Randomly generate $\lambda \in U(0,1)$;
$\boldsymbol{S_0}$ = Choose the solution in $\boldsymbol{P}$ with the minimum value for f_λ;
$count = 0$;
while $(count < max_{search})$
  $\boldsymbol{S_0}$ = Apply TS algorithm on $\boldsymbol{S}_0$ by considering f_λ as the objective function;
  Update $\boldsymbol{P}$;
  if ($\boldsymbol{P}$ *has changed*);
    $count = 0$;
  else
    $count = count + 1$;
  Randomly generate $\lambda \in (0,1)$;
PHASE 3: Intensifying
repeat
  Explore the neighbourhood of each unexplored solution in $\boldsymbol{P}$;
  Update $\boldsymbol{P}$ if necessary;
until $\boldsymbol{P}$ does not change;
return $\boldsymbol{P}$;
End

4.2.1. The multi-start TS algorithm

The multi-start TS algorithm aims to explore the search space more efficiently, by starting from multiple initial solutions and searching towards one objective. Different initial solutions were generated by utilizing the methods already introduced in Section 4.1.1 (D_{mean}'s minimization), Section 4.1.2 (T's minimization) and Section 4.1.3 (random procedure). Essentially, one solution is obtained by the method developed in Section 4.1.1 and at most $\lceil 1/\Delta\alpha \rceil + 1$ solutions are generated using the procedure explained in Section 4.1.2, by starting from $\alpha = 0$ and increasing this value in intervals of $\Delta\alpha$ until it reaches the value of 1. In case needed, the remained initial solutions are generated randomly according to Section 4.1.3.

4.3. AWS method

The AWS method solves the two main drawbacks in the traditional weighted-sum approach. As discussed in some studies such as [32], the first drawback is the probability of generating not an even distribution of solutions on the Pareto front, and the second is the disability of the traditional method for finding solutions of non-convex parts of the Pareto front. As proposed by Kim and De Weck [27] "in this approach, the weights are not predetermined, but they evolve according to the nature of the Pareto front of the problem". Starting from a small number of divisions with a large step size, $\Delta\alpha$, traditional weighted-sum method is used to generate a coarse representation of the solution. By calculating the distance between neighbour solutions, regions for further refinement are identified. Imposing additional inequality constraints in the objective space, these regions are known as feasible regions and the typical weighted-sum method is performed for identifying a new solution set. This procedure is repeated until all regions between two adjacent solutions reach a pre-specified resolution. In the following, the pseudo-code in Algorithm 3 shows the detailed procedure.

Algorithm 3. Pseudo-code of the AWS algorithm

Input Data
$\boldsymbol{P} = \emptyset$: set of non-dominated solutions;
$\mu_{initial}$: the number of initial segments;
ε: the Euclidean distance for determination of overlapping solutions;
Γ: the constant for further refinement;
δ: the constant for determination of objectives limitation;
Begin
$\alpha = 0$;
$\Delta\alpha = \frac{1}{\mu_{initial}}$;
while $(\alpha \leq 1)$
  Generate initial solution $\boldsymbol{S}_0$ based on greedy algorithm in Section 3.2;
  Apply TS on $\boldsymbol{S}_0$ by considering $\alpha D_{mean} + (1-\alpha)T$ as the objective function;
  Update $\boldsymbol{P}$ if necessary;
  $\alpha = \alpha + \Delta\alpha$;
REFINING:
Sort the members of $\boldsymbol{P}$ such that $f_1^i < f_1^{i+1}$ for all $i \in \{1, \ldots, |\boldsymbol{P}|\}$;
Calculate L_i as the length of segments for all $i \in \{1, \ldots, |\boldsymbol{P}|\}$;
Remove solutions of $\boldsymbol{P}$ with the distance less than ε from each other;
Calculate $n_i = \left\lfloor \Gamma \times \frac{L_i}{\bar{L}} + 0.5 \right\rfloor$;
$i = 1$;
while $(i \leq |\boldsymbol{P}|)$
  if $(\mu_i > 1)$
    $\theta = \tan^{-1}\left(\frac{f_2^i - f_2^{i+1}}{f_1^{i+1} - f_1^i}\right)$;
    $\alpha = 0$;
    $\Delta\alpha = \frac{1}{\mu_i}$;
    $LimitF_1 = f_1^{i+1} - \delta \times \cos\theta$;
    $Limit\ F_2 = f_2^i - \delta \times \sin\theta$;
    while $(\alpha \leq 1)$
      Apply TS on $\boldsymbol{S}_0$ by considering $\alpha D_{mean} + (1-\alpha)T$ as the objective function and imposing $\alpha D_{mean} \leq LimitF_1, T \leq LimitF_2$;
      Update $\boldsymbol{P}$ and $|\boldsymbol{P}|$;
      $\alpha = \alpha + \Delta\alpha$;
  $i = i + 1$;
if ($\boldsymbol{P}$ *has changed*);
  goto REFINING;
End

5. Computational experiments

Several tests have been conducted to evaluate the effectiveness of the proposed algorithms. In the following, the results are provided in details.

5.1. Test sets

We generated three groups of instances, including small, medium and large size data. All the developed algorithms were coded in C++ and run on a PC with an Intel Core i7 (2.93 GHz, 3.49 GB of RAM) and windows XP. We generated the test instances based upon the following parameters.

a) Number of orders (n) for small, medium and large size data are taken from sets $\{5, 6, 7\}$, $\{20, 30, 40\}$ and $\{100, 150, 200\}$, respectively. For all test instances, the processing time of each order j (p_j) is generated randomly from the discrete uniform distribution $U[1, 100]$.
b) The capacity of each delivery batch (or vehicle) is considered as $w = 2$ for the small instances and $w \in \{5, 10, 15\}$ for the other two sets.
c) For the single-objective case, weighting parameter in the objective function is considered as $\alpha \in \{0.2, 0.5, 0.8\}$.
d) Number of customers is considered as $k \in \{2, 3, 4, 5\}$ located in a square with the length of $\xi \in \{100, 200, 400\}$. In addition, τ_{ij} is defined as the integer part of the Euclidean distance between each pair of customers (i, j).
e) Number of suppliers, shown by η, is an integer value taken randomly from discrete uniform distribution $U\left[\left(1 - \frac{n}{3w}\right), \left(1 + \frac{n}{3w}\right)\right]$ and the release date imposed by each supplier is a random number from the discrete uniform distribution $U\left[0, \frac{1}{3}\sum_{j \varepsilon \boldsymbol{J}} p_j\right]$. The corresponding supplier of each order, is chosen randomly from the set of suppliers and the related release date is considered for that order, consecutively.

To generate the cost-related parameters, a scaling factor φ is used to make the scales of the two objectives comparable. Using preliminary tests, we found out that expected values for D_{mean} and T are close to $\frac{w\bar{p}n \times (\lceil \frac{n}{w} \rceil + 1) + \sqrt{2}\xi k}{2n}$ and $\left\lceil \frac{n}{w} \right\rceil \times 150 + \left\lceil \frac{n}{w} \right\rceil \frac{\sqrt{2}}{2} \times \frac{kw}{n}$, respectively in which $\bar{p}$ is the average of the processing times. Therefore, φ is the value generated by dividing the expected value of D_{mean} by the expected value of T. Consequently, the fixed transportation cost, incurred by each batch, is an integer value taken from $U[50\varphi, 250\varphi]$ and the travelling costs are random integer values drawn from $U\left[0.8\tau_{ij}\varphi, 1.2\tau_{ij}\varphi\right]$.

Finally, for each combination of the parameters, five random instances have been generated. As the result, we have 540, 1620 and 1620 test instances for the small, medium and large size groups, respectively.

5.2. Single-objective analysis

There are three parameters, namely L_T, ρ and max_{itr}, in the structure of the proposed TS algorithm. We have used the design of experiments techniques to achieve the best combination of parameters. Table 4 gives the best combination of parameters which is based on the preliminary experiments and the Wilcoxon tests.

We have used ILOG CPLEX 12.5 to run the exact model to optimality. In addition, we used time limits (TL) as the stopping criterion of the TS algorithm. Essentially, 1, 10, 30 and 100 s are put on the maximum run time of the algorithm. As the optimal results are

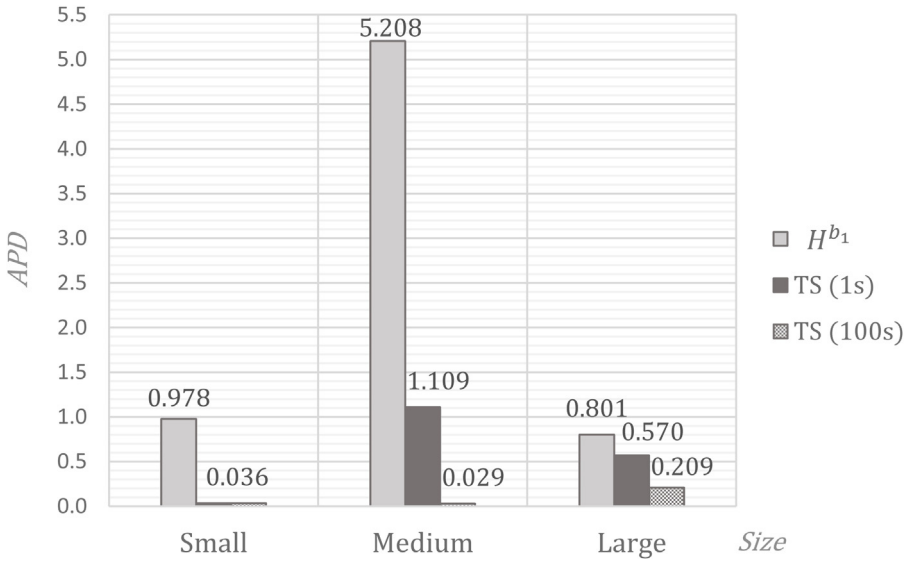

Fig. 2. The comparative results of TS and $H^b{}_1$ algorithms.

obtained only for the small size instances using ILOG CPLEX solver, the results of the TS for the small test instances are compared to the optimal solutions while for other test instances the results of TS are compared to the best solution obtained using all developed solution approaches. We define the percent deviation for a solution $\boldsymbol{S}_s$ as $\frac{F_s - F_{S*}}{F_{S*}} \times 100$ where F_{S*} indicates the objective value of the optimal or the best found solution.

The overall average running times of the CPLEX for the small instances were 5.5, 300 and 4800 s for those with $n = 5$, 6 and 8, respectively. Table 5 shows the average percent deviation (*APD*) of the solutions obtained by the TS algorithm from the optimal or the best found solutions. According to this table, for the small size instances, the TS algorithm is able to find considerably superior solutions in a very short computing time. Essentially, the overall average of *APD* is 0.036% and the results show that the algorithm is not able to improve the overall performance by using larger CPU run time. For the other two sets of instances, the performance of TS is improved by increasing the time limit.

Table 6 reports the *APD* between the results obtained by the local search algorithms $H^f{}_1$ and $H^b{}_1$ and the optimal or the best found solution achieved by all developed solution approaches. In addition, the average CPU time of each algorithm, over each test set, is reported inside the parenthesis.

Table 4
Setting of the TS parameters.

Parameters	Problem size		
	Small	Medium	Large
L_T	$2n$	n	n
ρ	2	4	6
max_{itr}	10	15	5

Table 5
Performance of the TS algorithm based on the *APD*.

Problem size	*TL* (s)			
	1	10	30	100
Small	0.036	0.036	0.036	0.036
Medium	1.209	0.559	0.330	0.129
Large	0.475	0.299	0.222	0.114

Table 6
The APD obtained by different local search algorithms.

Algorithm	Problem size		
	Small	Medium	Large
$H^f{}_1$	0.701(< 0.001)	5.173 (<0.001)	0.542 (1.007)
H^{b_1}	0.556(< 0.001)	5.083 (<0.001)	0.568 (2.07)

In this table, $H^f{}_1$ and $H^b{}_1$ show the *APD* obtained by performing these algorithms. Numerical results confirm the importance of choosing a qualified initial solution and indicate that the $H^b{}_1$ algorithm has better performance than $H^f{}_1$. It is to be noted that due to the complexity of large size instances, the TS algorithm needs by far more run time to be able to reach a good solution. In this regard, the results of local searches have some slight differences with the best solution found using various proposed methods and the APD obtained for large instances are less than that for medium size of problems. The comparative results of the TS and the most effective local search algorithm, i.e. $H^b{}_1$, are depicted in Fig. 2 based on the APD derived from each method. This figure indicates the superiority of the TS even with $TL = 1$ s.

5.3. Bi-objective analysis

In this section, we use the test problems described in Section 5.1. Since in the bi-objective problem we do not have the weighting parameter α, there will be a total of 1260 test instances including 180 small, 540 medium and 540 large size data. To measure the performance of the bi-objective heuristic algorithms, we use the following four parameters.

a) Number of Pareto solutions (NPS)
This metric represents the number of Pareto solutions obtained for a problem.

b) Spacing metric (SM)
This measure shows the distribution of the Pareto front by evaluating the extent of spread among the Pareto solutions. Introduced by Schott [33], it is determined by the following

relation in which $\boldsymbol{P}$ is the set of Pareto solutions:

$$SM = \sqrt{\frac{1}{|\boldsymbol{P}|} \times \sum_{i=1}^{|P|} (d_i - \bar{d})^2} \quad \text{where}$$

$$d_i = min_{j \in P} \left(\left| f_1^i - f_1^j \right| + \left| f_2^i - f_2^j \right| \right) \quad \text{and} \quad \bar{d} = \sum_{i=1}^{|P|} \frac{d_i}{|\boldsymbol{P}|}.$$

c) Non-uniformity of Pareto front (NPF)

Based on [34], this metric is used for measuring the non-uniformity distribution of a Pareto curve and is formulated as $NPF = \sqrt{\frac{1}{|\boldsymbol{P}|-1}} \times \sum_{i=1}^{|P|} \frac{\left(d_i - \bar{d}\right)^2}{\bar{d}^2}$ where $d_i = min_{j \in P} \sqrt{\left(f_1^i - f_1^j\right)^2 + \left(f_2^i - f_2^j\right)^2}$.

d) Maximum spread (MS)

Introduced by Ehrgott and Gandibleux [35], this metric measures the Euclidean distance between the first and the last solutions in Pareto front and is calculated by applying the following relation:

$$MS = \sqrt{\sum_{r=1}^{2} \left(max_{i=1}^{|\boldsymbol{P}|} f_r^i - min_{i=1}^{|\boldsymbol{P}|} f_r^i \right)^2}.$$

Using the design of experiments techniques we found that MOAMP algorithm is not very sensitive to values of its parameters. Thus, we choose to work with the minimum run time. Also, the sensitivity analysis performed for defining the AWS's parameters was based on the effect of those parameters on the abovementioned metrics. The final values reported in Table 7 are obtained for the parameters of the two bi-objective methods.

To compare the two provided bi-objective heuristic algorithms, an approximated set of efficient points, $\boldsymbol{P}^*$, is generated by combining the solutions obtained by both algorithms. The efficiency of each algorithm is evaluated as the percentage of points in $\boldsymbol{P}^*$ reached by the corresponding algorithm. Because of the high computational time of the AWS algorithm, we consider $TL = 200$ s as the total run time of these algorithms and all the TS procedures were run for 0.2 s. Table 8 shows the average size of $\boldsymbol{P}^*$, i.e. $|\boldsymbol{P}^*|$, the average of the percentage of points in $\boldsymbol{P}^*$ for each algorithm and also the mean run time of them for different data sets. Based on the results reported in this table, the MOAMP is more efficient for large instances, although relaxing the time limit of the AWS may lead to better performance.

Table 7
Setting of parameters for the bi-objective heuristic algorithms.

	Method					
	MOAMP				*AWS*	
Parameter	max_{start}	max_{phase}	δ	ε	Γ	$\mu_{initial}$
Value	5	5	0.5	0.25	1	3

Table 8
Comparative results of the bi-objective algorithms.

Problem size	Values				
	$\lvert\boldsymbol{P}^*\rvert$	%MOAMP	%AWS	$Time_{MOAMP}$	$Time_{AWS}$
Small	5.4	94	94	0.4	1.7
Medium	55.6	48	57	1.2	15.9
Large	73	63	38	121.1	203.5

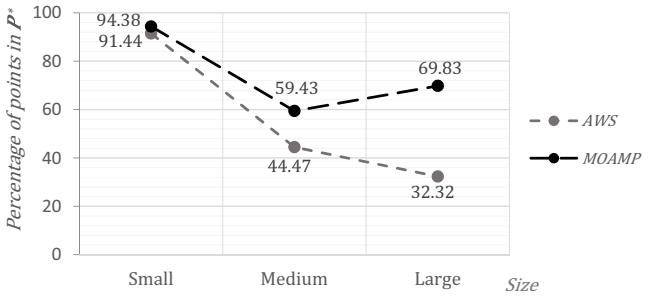

Fig. 3. Comparative results of MOAMP and AWS algorithms with $TL = 900$ s.

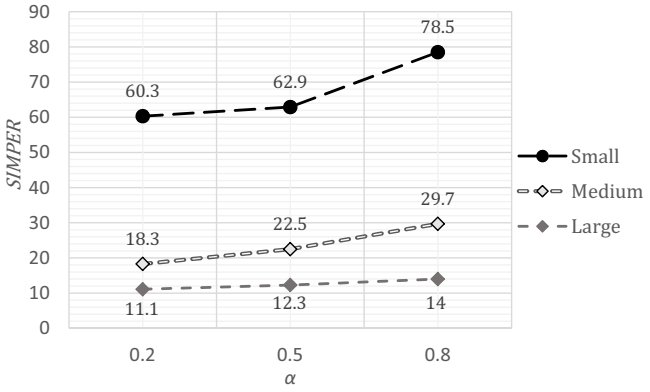

Fig. 4. Effect of α on the production scheduling.

For a fair comparison of the bi-objective heuristic algorithms, we have used the same run time limit $TL = 900$ s for both algorithms. Based on the results reported in Fig. 3, the MOAMP has a better performance with respect to the AWS algorithm. However, according to the random nature of the AWS, on some large instances better results have been obtained in comparison with the MOAMP.

5.4. Sensitivity analysis

In this section, several trade-offs and sensitivity analyses are presented to investigate the effect of some parameters on the objective functions and to ensure the validity of the proposed model and solution methods. Similar studies are presented in some papers such as [36,37].

5.4.1. Impact of α in the weighting method

We have used the parameter α as a constant for combining the two objectives based upon the decision maker's preference. Since the unified objective is defined as $\alpha D_{mean} + (1-\alpha)T$, any increase in the value of α will lead to prioritizing D_{mean} over T. Considering the preference of D_{mean}, the problem can be carried out as a scheduling problem which can be optimized using the SPT rule (Section 4.1.1). The obtained solutions show that in the case of $\alpha = 0.2$, 27% of answers can also be obtained using the SPT rule, while this value is 55% in the instances where $\alpha = 0.8$. For further comparison, the scheduling order in the final solutions of all instances were compared with the SPT rule. A similarity percentage, denoted by SIMPER, is used to measure the similarity between both answers. The increasing amount of this parameter as a result of increment of α from 0.2 to 0.8 is depicted in Fig. 4. It is shown that the growth is more evident in the small scale instances that can be attributed to the lower complexity of such instances

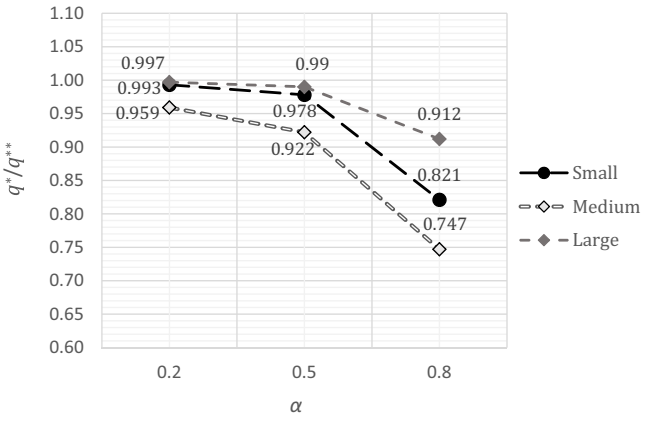

Fig. 5. Effect of α on the transportation.

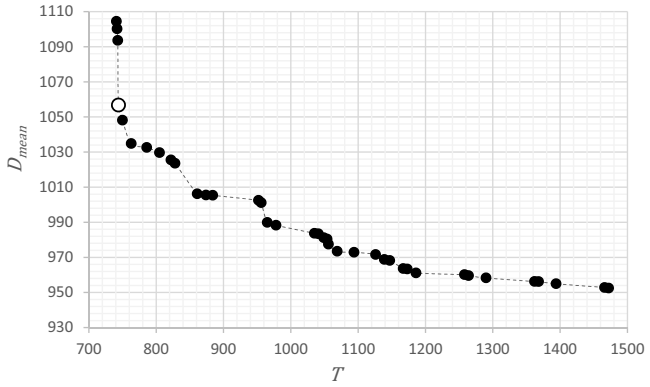

Fig. 8. Trade-off between D_{mean} and T.

On the other hand, reducing the value of α will increase the importance of transportation cost in minimizing the total objective function. Because of the fixed cost incurred by delivering each batch, it is probable that distributing the prepared orders in the minimum number of batches, leads to better solutions. If we consider $q^* = \lceil n/w \rceil$ as the minimum number of batches and q^{**} as the number of batches in a solution, the decreasing trend of ratio q^*/q^{**} confirms our conclusion (Fig. 5).

5.4.2. *Impact of the number of suppliers and the number of customers on number of Pareto solutions*

In this section, we change the parameters η and k separately in the medium size instances and evaluate their impacts on the number of Pareto solutions. For comparison, we generated P^*_{new} using MOAMP method. Defining P^*_{Total} as the union of non-dominated solutions of P^*_{new} and P^* and $\omega = \frac{|P^*|/\left|P^*_{Total}\right|}{|P^*_{new}|/\left|P^*_{Total}\right|}$, we evaluate the impact of η and k on the number of Pareto solutions by multiplying the number of suppliers and customers by 1.5, 2, 3 and 4. Figs. 6 and 7 show the increase in k will improve the quality of Pareto frontier as well as any decrease in η. As a result, it can be concluded that for any specific problem, increasing the number of customers or decreasing the number of suppliers result in better solutions. As the variety of suppliers leads to the more restrictions on the orders' release dates, contracting with many different suppliers can contribute to the poor quality of results.

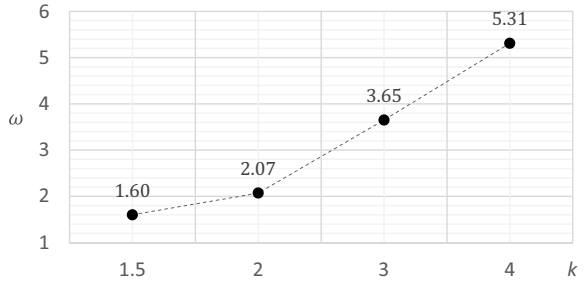

Fig. 6. Impact of k on the number of Pareto solutions.

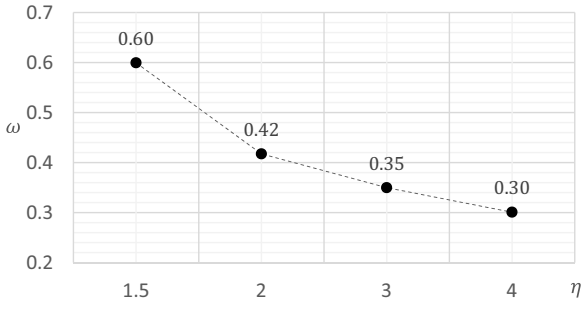

Fig. 7. Impact of η on the number of Pareto solutions.

5.4.3. *Trade-off between* D_{mean} and T

To show the trade-off between D_{mean} and T, we consider the non-dominated solutions of a test instance with 30 orders and 5 customers, which is solved by the MOAMP method. We know that when the number of batches increases, transportation costs are also increased but since orders are delivered to customers earlier, D_{mean} is decreased. Fig. 8 confirms this argument by showing the trade-off between D_{mean} and T.

5.4.4. *The influence of other key parameters on the Pareto front*

In this section, we have conducted sensitivity analyses on key assumptions such as transportation costs, vehicle capacity and release dates to see how these parameters affect the non-dominated solution set. Fig. 9 depicts the impact of doubling the fixed and variable costs on the Pareto set for a medium-size test instance, and it is shown that this parameter mostly influences the transportation cost.

Fig. 10 illustrates the impact of change in vehicle capacity on the non-dominated set. By increasing the capacity, more products are able to be transported in a single vehicle. Therefore, reducing the number of total vehicles by inserting more orders in each one leads to a decrease of the cost incurred for the delivery phase. However, the variable costs which are directly linked to the routing decisions may rise as the number of orders in each trip grows. Ultimately, since the amount of fixed cost is more than variable costs, doubling the vehicle capacity leads to a decrease in transportation cost and an increase in the mean delivery time, which is concluded from the delay incurred as a result of waiting for the batches to reach the maximum capacity and delivering more orders in each trip.

Also, a sensitivity analysis on release dates is depicted in Fig. 11. Since doubling the amount of release dates leads to an increase in the start times of the order processing, as well as the distance between release dates of different orders, we expect that the delivery times of orders and hence the first objective value is increased. Analysing the results gained from Fig. 11, it is concluded that by increasing this factor, orders are packed more separately or in fewer batches in order to balance the delivery times and maintain the customers' service level. When more batches are chosen for distribution phase, the amount of transportation cost and the second objective value is increased consequently.

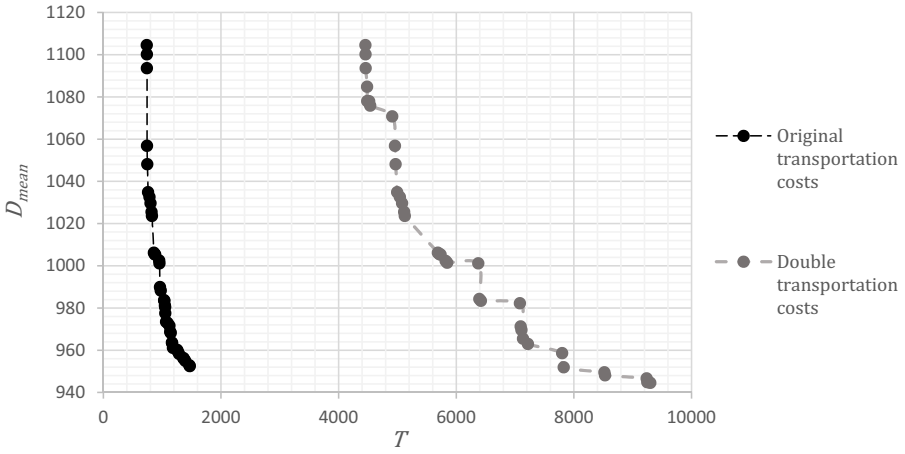

Fig. 9. The effect of fixed and variable cost on non-dominated solutions.

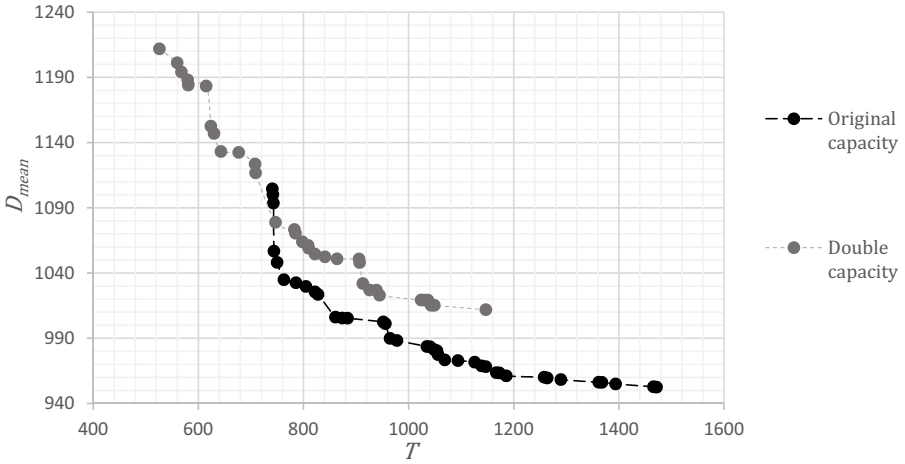

Fig. 10. The impact of vehicle capacity on non-dominated solutions.

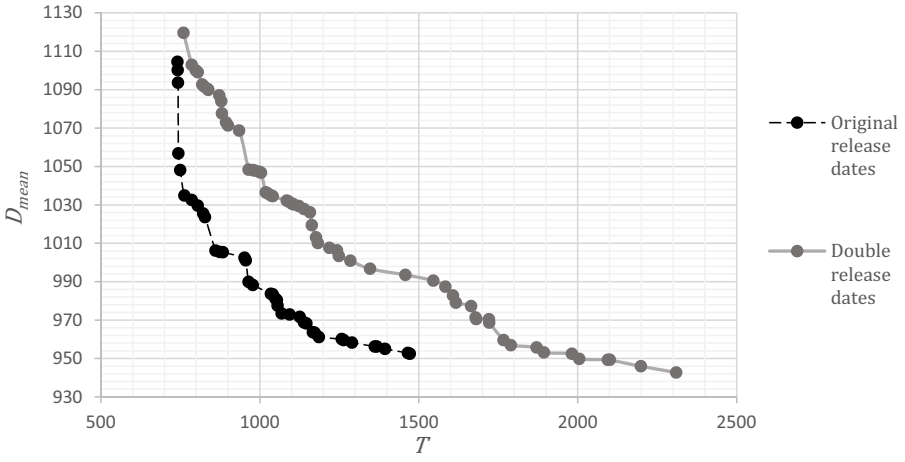

Fig. 11. The effect of release dates on non-dominated solutions.

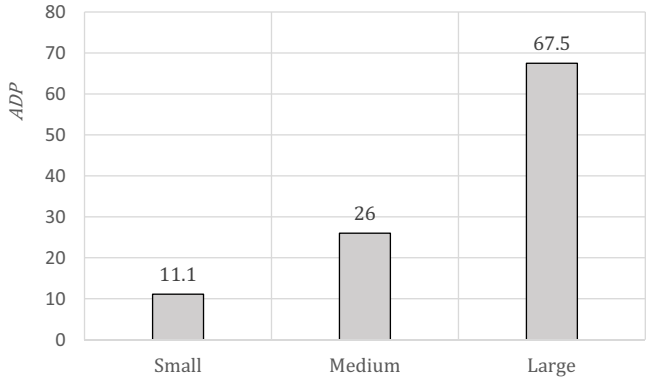

Fig. 12. Impact of integration.

5.5. Impact of integration

To illustrate the effectiveness of the proposed integrated scheduling model, in the following we provide a computational comparison of the proposed integrated approach with a hierarchical approach, in which production scheduling precedes distribution scheduling using a heuristic method. For the non-integrated model, in all generated instances, the SPT rule is employed for obtaining the optimal solution for scheduling the production stage to minimize the total completion time of all orders. Then, orders are batched in a way that at most one non-full batch is allowed to be available. Each delivery batch is transported immediately after completion of all its orders and the routing part of the problem is based on the nearest neighbour algorithm. Consecutively, the average of difference percentage (*ADP*) between the objective values has been calculated for the non-integrated model and the integrated one, which was solved by the TS algorithm with 1 s run time. Fig. 12 illustrates the *ADP* for the non-integrated solutions implying integration has more benefits for larger scale problems. Generalizing this analysis, it is highly efficient to consider the integrated problem, which leads to a dramatic decrease in costs, including transportation cost and delivery time of orders, especially for large-size problem.

6. Conclusions and future research

In this paper, we have developed an integrated scheduling of supplying, production and distribution in a supply chain, with consideration of the transportation cost and also the customer service level. This problem was proposed as a bi-objective problem and formulated as an integer linear programing model. Because of the complexity of the proposed model, two local search and a tabu search algorithm were developed to solve the single-objective version of the problem, which was based on the weighted sum method. For solving the bi-objective problem, we proposed two heuristic methods named AWS and MOAMP. By applying all the mentioned algorithms and comparing the obtained results, the efficiency of the TS algorithm was concluded. Also, the computational experiments for the bi-objective solution methods show that MOAMP outperforms AWS in short CPU run times while AWS probably leads to better solutions if CPU run time is considered as unlimited.

In addition, there are some directions on future research. Considering some other types of manufacturing environments such as parallel machines is an interesting research topic. Of course, it is to be noted that due to the complexity of this problem, considering multiple machines in the production stage will definitely increase the run time needed for solving the MILP model and decrease the efficiency of proposed model. In this regard, solution can be represented as a mn matrix in which m is the number of machines and n indicates the number of jobs in the problem. Another future research topics are differentiating orders by considering various sizes and studying other objective functions like minimizing the delivery tardiness for capturing more diverse applications and adding some other constraints to consider more real conditions of manufacturing and supply chains.

References

[1] Chang YC, Lee CY. Machine scheduling with job delivery coordination. Eur J Oper Res 2004;158(2):470–87.
[2] Selvarajah E, Steiner G. Batch scheduling in a two-level supply chain — a focus on the supplier. Eur J Oper Res 2006;173(1):226–40.
[3] Hamidinia A, Khakabimamaghani S, Mazdeh MM, Jafari M. A genetic algorithm for minimizing total tardiness/earliness of weighted jobs in a batched delivery system. Comput Ind Eng 2012;62(1):29–38.
[4] Condotta A, Knust S, Meier D, Shakhlevich NV. Tabu search and lower bounds for a combined production–transportation problem. Comput Oper Res 2013;40(3):886–900.
[5] Wang X, Cheng TCE. Heuristics for parallel-machine scheduling with job class setups and delivery to multiple customers. Int J Prod Econ 2009;119(1):199–206.
[6] Lee CY, Chen ZL. Machine scheduling with transportation considerations. J Sched 2001;4(3):24.
[7] Soukhal A, Oulamara A, Martineau P. Complexity of flow shop scheduling problems with transportation constraints. Eur J Oper Res 2005;161(1):32–41.
[8] Cakici E, Mason SJ, Kurz ME. Multi-objective analysis of an integrated supply chain scheduling problem. Int J Prod Res 2012;50(10):2624–38.
[9] Li CL, Vairaktarakis G, Lee CY. Machine scheduling with deliveries to multiple customer locations. Eur J Oper Res 2005;164(1):39–51.
[10] Chen ZL, Vairaktarakis GL. Integrated scheduling of production and distribution operations. Manage Sci 2005;51(4):614–28.
[11] Chang YC, Li VC, Chiang CJ. An ant colony optimization heuristic for an integrated production and distribution scheduling problem. Eng Optim 2014;46(4):503–20.
[12] Ullrich CA. Integrated machine scheduling and vehicle routing with time windows. Eur J Oper Res 2012;227(1):152–65.
[13] Hall NG, Potts CN. Supply chain scheduling: batching and delivery. Oper Res 2003;51(4):566–84.
[14] Wang X, Cheng TE. Production scheduling with supply and delivery considerations to minimize the makespan. Eur J Oper Res 2009;194(3):743–52.
[15] Wang X, Cheng TCE. Logistics scheduling to minimize inventory and transport costs. Int J Prod Econ 2009;121(1):266–73.
[16] Priyan S, Uthayakumar R. Two-echelon multi-product multi-constraint product returns inventory model with permissible delay in payments and variable lead time. J Manuf Syst 2015;36:244–62.
[17] Godichaud M, Amodeo L. Efficient multi-objective optimization of supply chain with returned products. J Manuf Syst 2015;37:683–91.
[18] Miller CE, Tucker AW, Zemlin RA. Integer programming formulation of traveling salesman problems. J ACM 1960;7(4):326–9.
[19] Desrochers M, Laporte G. Improvements and extensions to the Miller–Tucker–Zemlin subtour elimination constraints. Oper Res Lett 1991;10(1):27–36.
[20] Karaoglan I, Altiparmak F, Kara I, Dengiz B. The location-routing problem with simultaneous pickup and delivery: formulations and a heuristic approach. Omega 2012;40(4):465–77.
[21] Lenstra JK, Kan AHG. Complexity of vehicle routing and scheduling problems. Networks 1981;11(2):221–7.
[22] Garey MR, Johnson DS. Computer and intractability: a guide to the theory of NP-completeness. W.H. Freeman & Co; 1979.
[23] Kizilateş G, Nuriyeva F. On the nearest neighbour algorithms for the traveling salesman problem. Adv Comput Sci Eng Inform Technol 2013:111–8.
[24] Glover F. Future paths for integer programming and links to artificial intelligence. Comput Oper Res 1986;13(5):533–49.
[25] Glover F, Kochenberger GA. Handbook of metaheuristics. Springer Science & Business Media; 2003.
[26] Caballero R, Gandibleux X, Molina J. MOAMP: a generic multiobjective metaheuristic using an adaptive memory. Technical report. Valenciennes, France: University of Valenciennes; 2004.
[27] Kim IY, De Weck OL. Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. Struct Multidiscipl Optim 2005;29(2):149–58.
[28] Pinedo ML. Scheduling: theory, algorithms, and systems. Springer; 2012.
[29] García I, Pacheco J, Alvarez A. Optimizing routes and stock. J Heurist 2013;19(2):157–77.
[30] Molina J, Laguna M, Martí R, Caballero R. SSPMO: a scatter tabu search procedure for non-linear multiobjective optimization. Inform J Comput 2007;19(1):91–100.
[31] Pacheco J, Caballero R, Laguna M, Molina J. Bi-objective bus routing: an application to school buses in rural areas. Transport Sci 2013;47(3):397–411.
[32] Das I, Dennis JE. A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. Struct Optim 1997;14(1):63–9.

[33] Schott JR. Fault tolerant design using single and multicriteria genetic algorithm optimization (No. AFIT/CI/CIA-95-039). Wright-Patterson AFB, OH: Air Force Inst of Tech; 1995.
[34] Chambari A, Rahmati SHA, Najafi AA. A bi-objective model to optimize reliability and cost of system with a choice of redundancy strategies. Comput Ind Eng 2012;63(1):109–19.
[35] Ehrgott M, Gandibleux X. Approximative solution methods for multiobjective combinatorial optimization. Top 2004;12(1):1–63.
[36] Ghayebloo S, Tarokh MJ, Venkatadri U, Diallo C. Developing a bi-objective model of the closed-loop supply chain network with green supplier selection and disassembly of products: the impact of parts reliability and product greenness on the recovery network. J Manuf Syst 2015:76–86.
[37] Du F, Evans GW. A bi-objective reverse logistics network analysis for post-sale service. Comput Oper Res 2008;35(8):2617–34.