Computers & Industrial Engineering 142 (2020) 106342

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

A memetic algorithm for joint production and distribution scheduling with due dates

Ece Yağmur, Saadettin Erhan Kesen*

Konya Technical University, Faculty of Engineering and Natural Sciences, Department of Industrial Engineering, Selçuklu/Konya, Turkey

A R T I C L E I N F O

Keywords:
Permutation flow-shop
Vehicle routing
Due date
Mathematical modelling
Memetic algorithm

A B S T R A C T

Integration between production and distribution phases in supply chain has attracted close attention of many researchers over the last decade as companies have to juggle these activities for survival in increasingly competitive market conditions. In this paper, we study a joint production and distribution problem where a single manufacturer has committed to processing jobs (i.e., customer orders) on permutation flow-shop environment and subsequently distributing them by a single capacitated vehicle. Customers locate geographically-dispersed points and place their orders with pre-determined due dates. Since a single vehicle is available, customer orders should be consolidated in order to reduce the total trip time spent by the vehicle but this may result in failing to meet some of customer orders before their due dates. The objective is therefore minimizing the total travelling time plus total tardiness. We first develop a mixed integer linear programming to formulate the problem. Due to intractability matters, mathematical formulation suffers to find optimal solutions even in moderate number of customers. Thus, we present a memetic algorithm (MA) to find good or near-optimal solutions in an acceptable amount of time. In order to evaluate the effectiveness of the algorithm, we compare CPLEX results with that of the MA on a wide range of randomly generated test instances. Results indicate that MA is capable of finding solutions to optimality in a quite short time for most of the small-sized instances. For medium and large-sized instances, MA is still well-performing and yields better solutions as compared to CPLEX solutions found 3 h time limit.

1. Introduction

One of the most important performance criteria in supply chain management is to meet customer expectations at maximum service level with limited resources. Just-In-Time (JIT) policy has been widely used for many companies as it has an ultimate aim to eliminate all wastes and reduce in process buffers in supply at all levels. The philosophy of JIT shed lights on effective use of two important supply chain phases, namely production and distribution, by removing intermediate inventory phase, which might lead to excessive system cost. Beside JIT philosophy, make-to-order production policy also necessitates linkage between production and distribution functions with no or less stock since almost every product is unique and keeping these products in stock is needless.

Traditionally, production and distribution functions are treated standalone but recent studies show that joint decisions yield higher customer expectations with less system cost (Moons et al., 2017 and Chen, 2004). While production decisions include which order will be processed on which machine with the information of order starting and completion time, distribution decision involves customer visitation sequences along with delivery times at each customer.

In practical point of view, there exist many application areas where production and distribution decisions should jointly be made. For example, perishable or time-sensitive products (e.g., newspapers, food catering, ready-mixed concrete, and nuclear medicine) must be delivered to customers just after their production completes within a very limited time.

In many manufacturing and assembly facilities operating in sectors like electronic, telecommunication etc., each job consists of a series of operations and each operation is performed on different machines setup in series with a certain amount of time. This implies that each job follows the same route. Let's say machine 1, machine 2, and so forth. This kind of shop is referred to as flow-shop environment. One special case of flow-shop is that order of the jobs waiting to be processed in front of each machine is the same. This shop is usually known as permutation flow-shop.

The aim here is to generate a joint production and distribution schedule for a manufacturer who is responsible for processing and

* Corresponding author.
E-mail addresses: ecyagmur@ktun.edu.tr (E. Yağmur), sekesen@ktun.edu.tr (S.E. Kesen).

https://doi.org/10.1016/j.cie.2020.106342
Received 16 August 2019; Received in revised form 12 December 2019; Accepted 3 February 2020
Available online 04 February 2020

distributing orders. For any planning horizon, once all customer orders are collected, the manufacturer begins processing them with limited number of machines setup in series and then subsequently distributing the orders to the related customers by a single capacitated vehicle. As the orders are placed with a particular due date, manufacturer might fail to deliver some orders before due dates. The manufacturer may want to visit as many customers as possible in each trip to reduce total trip time subject to vehicle capacity. On the other hand, visiting more customers in a trip is likely to increase delivery times and it will lead some orders to be tardy. The objective is therefore minimizing the summation of two conflicting objectives, total trip time and total tardiness, both of which are measured in time unit. Any subset of customer orders to be delivered in a single trip is called "batch" and underlying assumption is that after production of any sequence of the orders in a batch complete, they are distributed to the related customers in the same sequence.

In order to formulate the abovementioned problem, we first develop a mixed integer linear programming (MILP) model. The problem consists of two operational level problems, namely permutation flow-shop scheduling and vehicle routing, both of which are proven to be NP-hard in strong sense. Not surprisingly, the intractability of the problem does not allow for the MILP model to provide optimal solutions even in moderate size of customers. We therefore introduce a Memetic Algorithm (MA) to obtain good or near optimal solutions in a shorter amount of time.

The main contributions of our study can be summarized as follows. First, we investigate the permutation flow-shop machine environment which is studied in few works in the literature although this shop is implemented in a broad range of real life cases. Second, we study, for what we believe to be the first time, an objective function composing of total trip time plus total tardiness. Finally, we present a memetic algorithm with different decoding strategies.

The rest of the paper is organized as follows: Subsequent section reviews the relevant studies on the integrated production and distribution scheduling problem including routing decisions. Section 3 describes the formal definition of integrated problem along with an illustrative example. The proposed MILP formulation for the problem is explained in Section 4. Section 5 is devoted to revealing the details of MA. Section 6 presents the generation of random test instances. In Section 7, experimental studies are presented, in which the comparative results of the CPLEX and MA are discussed. Conclusions and future directions are stated in Section 8.

2. Overview of the relevant literature

In recent years, more attention has been paid in literature to the coordination of production and distribution activities in the supply chain management. Although this research area has received much interest from researchers, scant studies have examined relatively more complex production and distribution environments. Based on the survey study of Chen (2010), the existing literature can be categorized into two areas regarding the type of delivery method. In the first area, researchers study simple delivery methods, such as (*i*) immediate and individual shipping of orders upon its completion, (*ii*) batch delivery to a single customer by direct shipping, and (*iii*) batch delivery to multiple customers by direct shipping. In the second area, however, researchers consider delivery methods in which orders belonging to different customers can be shipped together by routing method. In this study, we limit our literature review to the studies in which routing decisions are involved as shown in Table 1. Interested readers are also referred to the study of Moons et al. (2017). We classify the relevant studies according to machine configuration, vehicle number and type, objective function and solution method. It is obvious from Table 1 that while most of the studies concern with single and parallel machine production environments, few works deal with relatively more complex production environment such as flow-shop environment.

The first studies were related to newspaper printing and distribution problem. Hurter and Van Buer (1996) proposed the joint decision of production and distribution in a newspaper company and make sensitivity analysis for various parameter levels, whereby using a greedy algorithm with forward looking strategy to construct routes. Similarly, Russell, Chiang and Zepeda (2008) also applied the joint decision mechanism for a big newspaper company in order to minimize not only the number of vehicles required, but also the total travel distance incurred by the fleet of vehicles.

Chen and Vairaktarakis (2005) studied two classes of problems in terms of customer service level which is measured by the average and maximum time when the jobs are delivered to related customers and make worst case analyses for different variants of the problem. Devapriya et al. (2006) formulated two mixed integer programming models to solve the single plant and two-plant versions and provided heuristics based on evolutionary algorithms so as to find good quality solutions in a more reasonable time.

Geismar et al. (2008) developed a two-phase heuristic and subsequently developed a lower bound on the problem with the objective of determining the minimum time required to produce and deliver the products to customers. For the same problem setting, Karaoğlan and Kesen (2017) developed a branch-and-cut algorithm to further improve the results of Geismar et al. (2008). In a more recent study, Lacomme et al. (2018) extended the same problem by permitting the use of multiple vehicles and demonstrated the effectiveness of the proposed algorithm against the results of Geismar et al. (2008) and Karaoğlan and Kesen (2017) for some problem instances.

In the food industry, Chen, Hsueh and Chang (2009) formulated a mixed-integer nonlinear programming model for fresh food products under stochastic demands and proposed a solution algorithm based on decomposition concept. The objective of the model was to maximize the expected total profit of the supplier. In a study conducted in food industry more recently, Farahani, Grunow and Günther (2012) proposed a mixed integer linear programming model for real settings of a food catering company.

Park and Hong (2009) developed a hybrid genetic algorithm with several local optimization techniques for single-period inventory products and compared the integrated model with the uncoordinated one. Belo-Filho, Amorim and Lobo (2015) proposed an adaptive large neighborhood search to tackle large size instances for the problem presented in Amorim, Belo-Filho, Toledo, and d., Almeder, C., & Almada-Lobo, B. (2013) where job splitting decisions are taken into consideration in a parallel machine environment. In the study of Devapriya, Ferrell and Geismar (2017) three heuristics based on genetic and memetic algorithm were developed for the model, which determines the fleet size as well as trucks' routes.

Ullrich (2013) integrated production and distribution process in order to minimize total tardiness and compared the genetic algorithm and two decomposition approaches for parallel machine environment. Kesen and Bektaş (2019) describe linear programming models for two variants of the problem which includes parallel machine environment in production phase with time windows consideration.

Low, Chang and Gao (2017) proposed a nonlinear mathematical model considering heterogenous fleet in order to minimize the total cost which includes transportation cost, vehicle arrangement cost and penalty costs, subjected to satisfy all demands of each customer. Lee et al. (2014) applied an integrated model to radioactive materials for nuclear medicine so as to minimize the total cost including production costs, fixed vehicle costs and travel costs. Viergutz and Knust (2014), extended the problem in Armstrong, Gao, and Lei (2008) addressing a product with a short lifespan and a predefined customer sequence and proposed tabu search algorithm.

Li et al. (2016) proposed a non-dominated genetic algorithm with the elite strategy (NSGA-II) for the problem with the objective of minimizing the vehicle delivery cost and the total customer waiting time. Kergosien, Gendreau and Billaut (2017) addressed the

Table 1
An overview of existing literature.

Paper	Machine Configuration	Vehicle number/ type	Objective	Solution method
Hurter and Van Buer (1996)	Single	Limited/ Homogenous	Cost	Heuristic
Chen & Vairaktarakis (2005)	Single/Parallel	Sufficient/ Homogenous	Cost/ Service	Exact/ Heuristic
Devapriya et al. (2006)	Single	Sufficient/ Homogenous	Cost	Exact/ Heuristic
Armstrong et al. (2008)	Single	Limited (single)	Service	Exact
Geismar et al. (2008)	Single	Limited (single)	Service	Exact/ Heuristic
Russell, Chiang and Zepeda (2008)	Parallel	Limited/ Heterogenous	Cost/ Service	Heuristic
Chen, Hsueh and Chang (2009)	Single	Limited/ Homogenous	Profit	Exact
Park and Hong (2009)	Single	Limited/ Homogenous	Cost	Exact/ Heuristic
Scholz-Reiter et al. (2011)	Flow shop	Limited/ Heterogenous	Cost	Exact
Farahani, Grunow and Günther (2012)	Parallel	Limited/ Homogenous	Cost/ Quality	Exact/ Heuristic
Amorim et al. (2013)	Parallel	Sufficient/ Homogenous	Cost	Exact
Ullrich et al. (2013)	Parallel	Limited/ Heterogenous	Service	Exact/ Heuristic
Lee et al. (2014)	Parallel	Limited/ Heterogenous	Cost	Exact/ Heuristic
Low, Chang and Gao (2017)	Single	Sufficient/ Heterogenous	Cost	Exact/ Heuristic
Meinecke and Scholz-Reiter (2014)	Flow shop	Limited/ Homogenous	Cost	Heuristic
Viergutz and Knust (2014)	Single	Limited (single)	Service	Exact/ Heuristic
Belo-Filho, Amorim and Lobo (2015)	Parallel	Sufficient/ Homogenous	Cost	Exact / Heuristic
Ehm and Freitag (2016)	Flow shop	Limited/ Heterogenous	Cost/ Service	Exact
Li et al. (2016)	Single	Limited/ Homogenous	Cost/ Service	Exact / Heuristic
Mohammadi et al. (2018)	Flow shop	Limited/ Heterogenous	Service	Exact / Heuristic
Devapriya, Ferrell and Geismar (2017)	Single	Sufficient/ Homogenous	Cost	Exact / Heuristic
Karaoglan and Kesen (2017)	Single	Limited (single)	Service	Exact
Kergosien, Gendreau and Billaut (2017)	Parallel	Limited (single)	Service	Exact/ Heuristic
Ramezanian, Mohammadi and Cheraghalikhani (2017)	Flow shop	Limited/ Heterogenous	Cost	Exact/ Heuristic
Lacomme et al. (2018)	Single	Limited (single)	Service	Exact/ Heuristic
Kesen and Bektaş (2019)	Parallel	Limited/ Heterogenous	Service	Exact
Mohammadi et al. (2019)	Job shop	Limited/ Heterogenous	Cost/ service	Exact/ Heuristic
Our study	Flow shop	Limited (single)	Service	Exact/ Heuristic

chemotherapy production and delivery problem where independent jobs are prepared by pharmacy technicians working in parallel and used Benders decomposition-based heuristic to find feasible solutions and lower bounds.

Mohammadi, Al-e-Hashem and Rekik (2019), considered an integrated problem with time windows, in which production is assumed to be performed in a flexible job-shop system for a furniture manufacturing company as a case study. They made a use of ε-constraint method and particle swarm optimization for multi-objective model.

Mohammadi, Cheraghalikhani and Ramezanian (2018) addressed the problem in permutation flow-shop system with minimizing the time of the last order delivered to relevant customer by the last vehicle and that vehicle returning to production center. They proposed the imperialist competitive algorithm for this problem. Ramezanian, Mohammadi and Cheraghalikhani (2017) handled the problem studied by Mohammadi et al. (2018) by different objective function minimizing total production and distribution cost.

Scholz-Reiter et al. (2011) and Meinecke and Scholz-Reiter (2014), also consider the integrated problem in flow shop production environment. These studies differ from the above-mentioned studies as they consider inventory decisions. Scholz-Reiter et al. (2011) is the first paper that explicitly mentions that inventory can be stored before the first production level, between consecutive production levels, and before the departure of a vehicle trip, and takes holding costs into account (Moons, 2017). Ehm and Freitag (2016), proposed a mixed integer programing formulation for separate and integrated approach to show the benefit of integration.

3. Formal definition of the problem

The problem can be formally stated as follows: We consider a single plant owned by a manufacturer in which there is a set of machines $\{1, \cdots, M\}$ in series. A set of orders $\{1, \cdots, N\}$arrive at the plant where each order has to visit each of the machines exactly once in the same sequence. The processing time of order $i \in \{1, \cdots, N\}$ on machine $m \in \{1, \cdots, M\}$ is denoted by p_{im}. Each order belongs to a unique customer, for which reason we use the terms orders and customers interchangeably as there is a one-to-one correspondence between them. We assume that each order i is non-splittable, ready at time 0 and once its operation on machine m starts, it cannot be interrupted until the completion (i.e., preemption is not allowed). Each customer i locating in different region has a demand size of d_i, for which p_{im} units of time is required to complete on machine m as a linear function of d_i. We also assume that orders are produced in batches and each batch composes of several different orders. Once processes of all orders in a particular batch complete, a single vehicle with a capacity of Q delivers them to the associated customers by considering due date of each order j (denoted as dd_j). Due to capacity limitation of the vehicle, some batches may have to temporarily wait at the plant for delivery in the next trip. For the sake of simplicity, we further assume that orders constituting a particular production batch are delivered in the same trip. The routing part of the problem is modelled on a graph with $\{0, 1, \cdots N\}$ as the set of nodes in which 0 denotes the plant and remaining nodes represent customers. The travel time between nodes i and j is denoted by t_{ij} and assumed to be constant. Since the vehicle capacity is limited, it can be used for multiple trips. The problem is to determine which orders will be assigned to which batches in which sequence on each of the M machines and to find the sequence of customers' visit along with delivery times to the associated customers for each trip so as to minimize the summation of total trip time and total tardiness.

3.1. Illustrative example

In this section, we aim to explain the problem with the help of an illustrative example. Table 2 shows dataset generated for an instance involving 3 machines and 7 customer orders. In Table 2, first column represents the customer nodes. Node *0* indicates the plant, which is located in the middle of the two-dimensional plane. Second and third columns give the coordinates of the associated customer nodes. We should note that the transportation time between each pair of nodes $i$and j is calculated using Euclidian distances. While d_i represents the demand size of customer i, p_{i1}, p_{i2}, and p_{i3} are the processing times of order i on machines 1, 2, and 3, respectively. The vehicle capacity Q is set to 180. Last two columns represent two different due date

Table 2
Random dataset for the illustrative example.

i	X coordinate	Y coordinate	d_i	p_{i1}	p_{i2}	p_{i3}	dd_{wide}	dd_{tight}
0	0	0	–	–	–	–	–	–
1	2	33	75	77	76	69	434	446
2	7	−51	92	87	97	93	795	630
3	52	−15	65	65	69	59	910	701
4	−33	−42	76	73	76	84	521	311
5	54	49	89	91	85	98	451	464
6	−48	55	60	63	60	66	552	556
7	2	−46	89	83	94	88	628	402

parameters, namely dd_{wide} and dd_{tight} to be able to see the effect of due dates on the solution. In the problem examined under dd_{wide} parameter, due dates are relatively wider than the problem examined under dd_{tight} parameter.

Fig. 1 displays feasible solutions for the illustrative example in accordance with two due date settings. For a particular tour, the load capacity of the vehicle is not exceeded. According to the Fig. 1(a) which considers dd_{wide} parameters, vehicle completes its delivery service with four tours as follows: 0–5–0 (tour 1), 0–1–6–0 (tour 2), 0–7–4–0 (tour 3) and 0–2–3–0 (tour 4). And the case in which solution generated based on dd_{tight} parameters depicted in Fig. 1(b), vehicle meets the customer demands with five tours as follows: 0–4–0 (tour 1), 0–7–0 (tour 2), 0–1–6–0 (tour 3), 0–2–3–0 (tour 4) and 0–5–0 (tour 5).

Fig. 2 represents Gantt charts for the solutions in Fig. 1 with order arrival times (A_j) and due dates (dd_j) of related customers. As mentioned before, any particular batch constitutes the same orders for both production and distribution. When we look at Fig. 2(a) based on the first due date setting dd_{wide}, the production of the order 5 is completed at time 274, at which vehicle starts the delivery of the first tour. The vehicle returns to the depot at time 420 after making the delivery of the order 5. Despite the production completion time of the second batch including orders 1 and 6 is 409, the delivery of the second batch starts only after vehicle is ready (i.e., at time 420). The vehicle returns to the depot at time 582 after making the delivery of the third batch including customer orders 1 and 6. Although the third production batch completes its last operation on the last machine at time 581, the delivery of this batch can only start at time 582, which is the returning time of vehicle after making the delivery of second batch. While the vehicle returns to the plant at time 719, the production completion time of the last batch which consists of orders 2 and 3 is 733. So, the starting time of the distribution for the last batch at time 733 and under this production and distribution sequences, the returning time of vehicle after fulfilling all deliveries is found as 898. From Fig. 2(a), we can see that order 1, order 4 and order 7 become tardy. For this solution total tardiness time is calculated as 165 ($T_1 = 20$, $T_4 = 144$and $T_7 = 1$) and total trip time is calculated as 610. So, the objective function, summation of total trip time and total tardiness, is found to be 775 for the illustrative example.

The Gantt chart for the second solution shown in Fig. 2(b) can be followed in a similar fashion. As seen from Fig. 2(b) only order 4 is delivered before its due date. It is possible to make some shifts in the schedule without disrupting the flow of subsequent orders. For example, order 7 is served exactly at its due date, although it may be delivered before this time. For this solution, total trip time is calculated as 675 and total tardiness time is calculated as 577 ($T_1 = 61$, $T_2 = 57$, $T_3 = 44$, $T_5 = 409$and $T_6 = 6$). So, the objective function is found to be 675 + 577 = 1252 for the solution shown in Fig. 2(b).

While distributing the orders, idle times could occur in two different cases. In the first case the vehicle which is ready in the plant waits until the completion of production of the orders in the next batch. Or adversely, the batch whose production is completed waits until the vehicle visits all customers in its previous trip and returns to the plant. In the illustrative example, idle waiting times for both production and distribution occur.

4. A mathematical formulation

In this section, we will present a mathematical formulation for integrated production and distribution scheduling problem. We now describe the parameters, decision variables and MILP formulation for considered model.

Parameters

d_i : demand of customer i ($i = 1, \cdots, N$)

p_{im} : processing time of order i on machine m ($i = 1, \cdots, N$; $m = 1, \cdots, M$)

Q : the capacity of vehicle

t_{ij} : travel time between customers i and j ($i, j = 0, 1, \cdots, N$; $i \neq j$; index0indicatesdepot)

dd_j: due date for order j

H: sufficiently large number

Decision variables

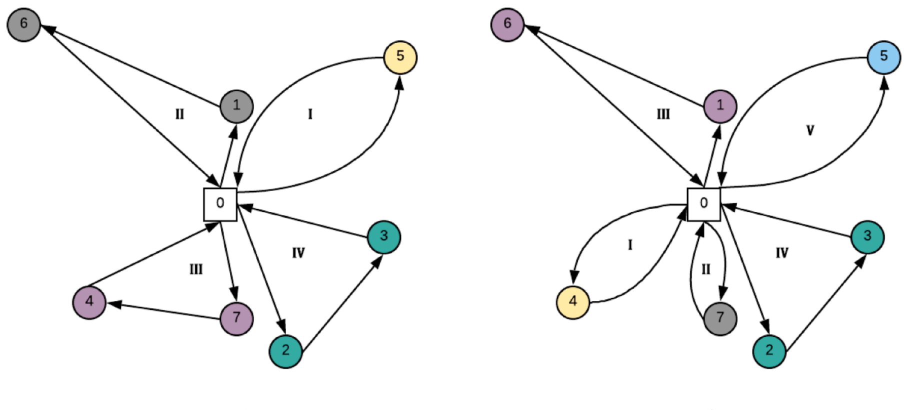
X_{ij} = 1 if vehicle goes directly from node i to node j ($i, j = 0, 1, \cdots, N$; $i \neq j$), 0 otherwise.

W_{ij} = 1 if vehicle completes preceding tour with node i and starts succeeding tour with node$j$ ($i, j = 0, 1, \cdots, N$; $i \neq j$), 0 otherwise.

u_i: the total size of load on vehicle just before visiting node i ($i = 0, 1, \cdots, N$)

S_{im}: process starting time of order i on machine m ($i = 1, \cdots, N$; $m = 1, \cdots, M$)

C_{im}: process completion time of order i on machine m



(a) *wide* (b) *tight*

Fig. 1. Feasible solutions for the illustrative example with different due date settings.

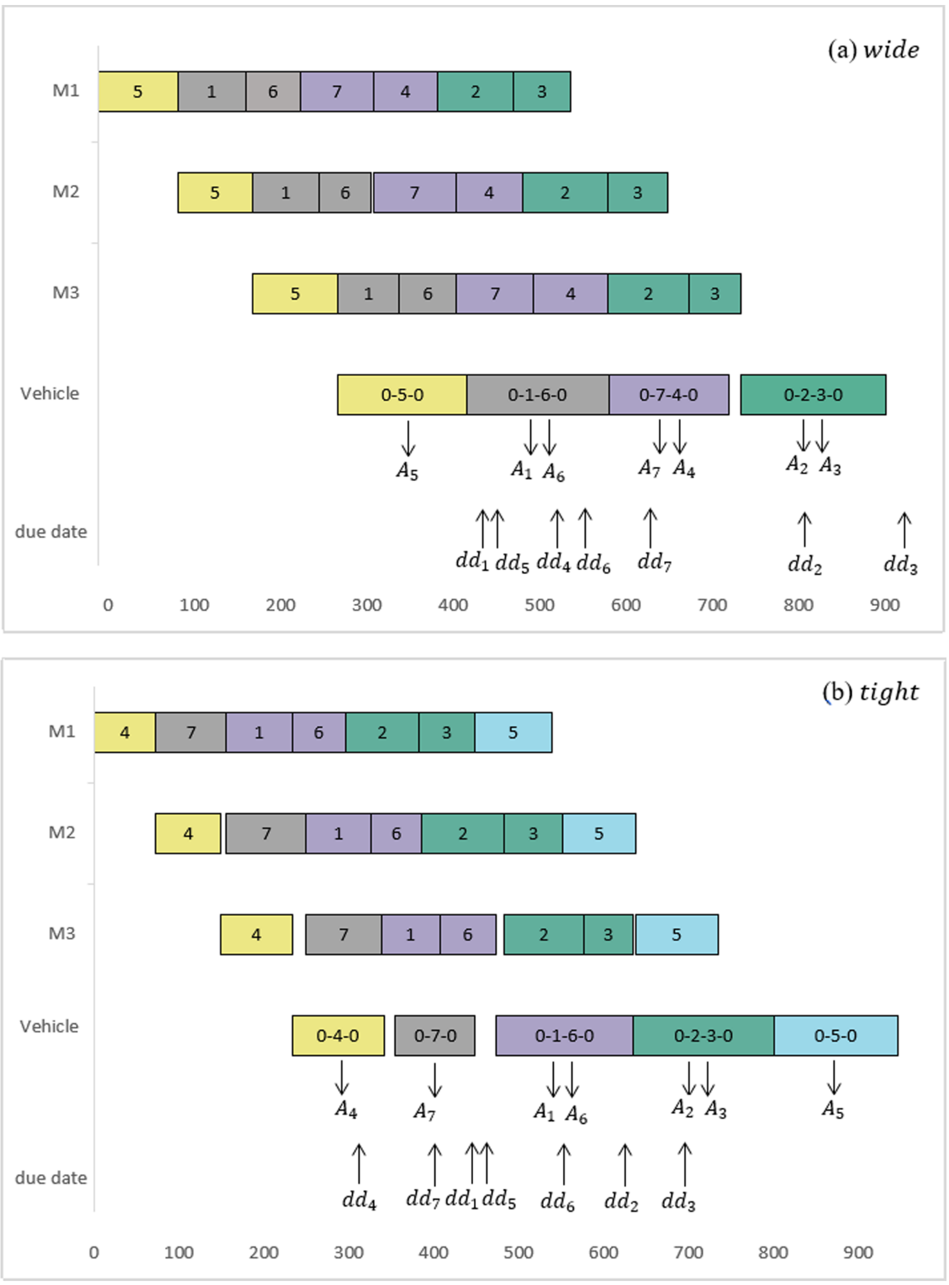


Fig. 2. Gantt charts of feasible solutions for the illustrative example with different due date settings.

$(i = 1,\cdots,N;\ m = 1,\cdots,M)$

A_i: The arrival (or delivery) time of vehicle at customer i $(i = 1,\cdots,N)$

Y_i: Production completion time of a batch to which order i belongs $(i = 1,\cdots,N)$

T_i: Tardiness for customer$i$ $(i = 1,\cdots,N)$

The mathematical formulation is given as follows:

$$minimize \sum_{i=0}^{N}\sum_{j=0}^{N} t_{ij}X_{ij} + \sum_{i=1}^{N} T_i \tag{1}$$

subject to

$$\sum_{i=1}^{N} X_{ij} = 1 j = 1,\ \cdots,N \tag{2}$$

$$\sum_{j=0}^{N} X_{ij} = \sum_{j=0}^{N} X_{ji} i = 0,\ \cdots,N \tag{3}$$

$$u_j - u_i + QX_{ij} + (Q - d_i - d_j)X_{ji} \le Q - d_i i, j = 1,\ \cdots,N;\ i \ne j \tag{4}$$

$$u_i \ge d_i + \sum_{j=1;j\ne i}^{N} d_jX_{ji} i = 1,\ \cdots,N \tag{5}$$

$$u_i \le Q - (Q - d_i)X_{i0} i = 1,\ \cdots,N \tag{6}$$

$$\sum_{j=1;i\ne j}^{N} W_{ij} \le X_{i0} i = 1,\ \cdots,N \tag{7}$$

$$\sum_{i=1,i\ne j}^{N} W_{ij} \le X_{0j} j = 1,\ \cdots,N \tag{8}$$

```
Algorithm 1 - Proposed Algorithm
Input: Parameters of population size, tournament size, crossover rate, mutation rate, maximum number of generations
Output: The best individual in all generations
Step 1: Initialize population size individuals
        Step 1.1: Generate first individual by Clarke and Wright's Saving Algorithm
                  Generate a giant tour for second individual by Nearest Neighbor Algorithm
                  Generate giant tours for (population size/2)-2 individuals by Random Task Heuristic
                  Generate giant tours for population size/2 individuals by fully random
        Step 1.2: Decode each giant tour by Prins' splitting algorithm for evaluation (Obtain feasible solution)
WHILE number of generations is less than maximum number of generations DO
        Step 2: Compute the fitness of each individual
        Step 3: Perform the selection process with binary tournament among randomly selected individuals as much as tournament size
        Step 4: Use the order crossover operator with probability of crossover rate and generate two offspring
        Step 5: Mutate the resulting offspring by swap or 2-opt swap procedure with probability of mutation rate
        Step 6: Check feasibility of new generated offspring by different decoding strategies
                Decode new generated offspring by Prins' splitting algorithm
                Decode new generated offspring by direct shipping policy
                Decode new generated offspring by random splitting
        Step 7: Evaluate new offspring
                ELSEIF both of the offspring already exist in population THEN
                        Apply mutation operator to either of the offspring and RETURN Step 7
                ELSEIF either of the offspring already exist in population THEN
                        Select the offspring not existing in the population to compare with the individuals.
                ELSE
                        Select the offspring with the best objective value to compare with the individuals in population
        Step 8: Apply local search procedure
        Step 9: Update the population for the next generation
END
Step 10: Return the best solution
```

Fig. 3. Pseudocode of proposed algorithm.

$$\sum_{j=1}^{N} X_{0j} - \sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij} = 1 \tag{9}$$

$$Y_i \geq C_{iM}\, i = 1, \cdots, N \tag{10}$$

$$Y_i \leq C_{iM} + H(1 - X_{i0})\, i = 1, \cdots, N \tag{11}$$

$$Y_j - Y_i \leq H(1 - X_{ij} - X_{ji})\, i, j = 1, \cdots, N;\ i \neq j \tag{12}$$

$$A_i - A_j + HX_{ij} + (H - t_{ij} - t_{ji})X_{ji} \leq H - t_{ij}\, i, j = 1, \cdots, N;\ i \neq j \tag{13}$$

$$A_i - A_j + HW_{ij} \leq H - t_{i0} - t_{0j}\, i, j = 1, \cdots, N;\ i \neq j \tag{14}$$

$$A_j \geq HW_{ij} - H + Y_j + t_{0j}\, i, j = 1, \cdots, N;\ i \neq j \tag{15}$$

$$A_j \geq Y_j + t_{0j} - H\left(1 - X_{0j} + \sum_{i=1}^{N} Wij\right) j = 1, \cdots, N;\ i \neq j \tag{16}$$

$$A_j \leq Y_j + t_{0j} - H\left(1 - X_{0j} + \sum_{i=1}^{N} Wij\right) j = 1, \cdots, N;\ i \neq j \tag{17}$$

$$C_{im} - S_{jm} \leq H(1 - X_{ij} - W_{ij})\, i, j = 1, \cdots, N;\ i \neq j;\ m = 1, \cdots, M \tag{18}$$

$$S_{i(m+1)} \geq C_{im}\, i = 1, \cdots, N;\ m = 1, \cdots, M - 1 \tag{19}$$

$$C_{im} = S_{im} + p_{im}\, i = 1, \cdots, N;\ m = 1, \cdots, M \tag{20}$$

$$T_i \geq A_i - dd_i\, i = 1, \cdots, N \tag{21}$$

$$A_i,\ u_i,\ S_{im},\ C_{im}, Y_i,\ T_i \geq 0 \tag{22}$$

$$X_{ij} \in \{0, 1\}\, i, j = 0, \cdots, N \tag{23}$$

$$W_{ij} \in \{0, 1\}\, i, j = 1, \cdots, N \tag{24}$$

The objective, which is given in Eq. (1) is to minimize total trip time and total tardiness. Eq. (2) ensures that each node is visited exactly once. Eq. (3) indicates that the number of arcs entering and leaving any node must be the same. Eqs. (4)–(6) are capacity and sub-tour elimination constraints. In particular, Eq. (4) states that the total load on vehicle in any tour must not exceed the capacity limit of the vehicle. Eq. (5) and Eq. (6) indicate the lower and upper limit for auxiliary variables of u_i. Eq. (7) and Eq. (8) determine the successive tour combinations, using the last customer (let's say customer i) of the preceding tour and the first customer (let's say customer j) of the succeeding tour. Eq. (9) guarantees that difference between the total number of tours and the total number of successive tour combinations is exactly one. Eq. (10) and Eq. (11) state that production of any particular batch must be completed after operation of the last order i in the batch is performed on the last machine M. Eq. (12) indicates that all orders in a batch must wait until the completion of the last order i in a production to get ready for distribution. Eq. (13) determines the arrival time of the successive orders in the same batch at the associated customers. Eq. (14) along with Eq. (15) inter connect the arrival time of the first order j in the succeeding batch with the last order i in the preceding batch if $W_{ij} = 1$. Eq. (14) ensures that the vehicle can only start the delivery of the first customer i in the succeeding batch after completing the delivery of last order j in the preceding batch and returning to the depot. Eq. (15) states that the vehicle can only start the delivery of customer j after production of the batch to which order j belongs completes. Eq. (16) and Eq. (17) determine the delivery time of the first customer in the first tour. Eq. (18) determines production starting time of the successive orders in the same batch and also indicates the production starting time of the first order in the successive batches. Eq. (19) guarantees that processing of any order on succeeding machine can only start after its processing completes on preceding machine. Eq. (20) ensures that completion time of any order i on machine m equals to starting time of order i plus its processing time on machine m (p_{im}). Eq. (21) simply calculates the tardiness value. Eqs. (22)–(24) represent the non-negativity and integrality restrictions on the variables.

5. Memetic algorithm

The integrated problem comprises of two well-known NP-hard problems, namely permutation flow-shop scheduling and vehicle routing. Finding a solution to the integrated problem is at least as hard as finding a solution to any of the problems standalone. We, therefore, develop a heuristic approach based on memetic algorithm to find good or near-optimal solutions in a reasonable amount of time due to intractability matter of the problem. Memetic algorithm which use a local search procedure is an extension of genetic algorithm and was first introduced by Moscato (1989). It has already been compared to classical evolutionary algorithms on numerous combinatorial optimization problems and experimental results indicated that the memetic algorithms found much better solutions than standard genetic algorithms (Garg, 2010; Elbeltagi et al. ,2005). Details of the proposed MA are given in Fig. 3.

5.1. Encoding schemes and evaluation

The chromosome is usually expressed in a string of variables, each element of which is called a gene. The variable can be represented by binary, real number, or other forms and its range is usually defined by the problem specified (Tang et.al., 1996). In the algorithm, we use real numbers on permutation encoding including all customer without tour information and a solution vector is represented by a giant tour which belongs to Travelling Salesman Problem (TSP) or Vehicle Routing Problem (VRP) with infinite capacity. However, since the vehicle capacity is limited, it cannot be expected that the vehicle can serve all customer in a single tour. As a result, it is also necessary to decide which customer belongs to which batch. Decoding is a key step in which information from a chromosome is read to build a feasible solution. In decoding phase, we use different decoding strategies to create a feasible solution considering capacity constraint to evaluate fitness function value.

Prins (2004) shows that a chromosome can be converted into an optimal VRP solution (subject to chromosome sequence) at any time, thanks to its splitting procedure and we also find useful this strategy when we consider the classical objective of VRP which take part of our objective as a minimization of total trip time. The splitting procedure can be seen in Fig. 4 and the readers are referred to works of Prins (2004, 2009) for more detail.

Prins' splitting algorithm is basically designed to minimize the total trip time which arises in the classical VRP. However, objective here is to minimize the total trip time and total tardiness. As Prins' algorithm disregard tardiness objective in nature, any solution with relatively lower total trip time does not necessarily produce a low total trip time plus total tardiness. This requires us to introduce other decoding schemes which can search for different areas in solution space for diversification. For this purpose, we present other two decoding schemes, namely direct shipping and random splitting. In direct shipping, as its name suggests, the vehicle visits only one customer in each tour. In random splitting, for a given giant tour, customer demands are collected from first customers to onwards until vehicle capacity is exceeded. For this customer subset, the number of customer to be visited

```
set $V_0 = 0$ and $V_i = \infty$ (where $V_i$ is cost of path to node $i$, $n$ is number of nodes)
for $i \leftarrow 1$ to $n$ do
    for $j \leftarrow i$ to $n$ while subsequence/route $(T_i, T_{i+1}, \dots, T_j)$ feasible
        compute route cost (where $z_{i-1,j}$ is cost of arc $(i-1, j)$)
        if $V_{i-1} + z_{i-1,j} < V_j$ then
            $V_j \leftarrow V_{i-1} + z_{i-1,j}$
        end if
    end for
end for
return the batches
```

Fig. 4. Prins' splitting algorithm.

is uniformly selected with a lower value of 1 and maximum value of the number of customers in subset. This encoding scheme executes until the last customer. Let's reconsider the illustrative example given in Section 3.1 and assume a giant tour as follows: $(0-4-2-7-1-3-5-6-0)$. According to the giant tour, orders 4 and 2 can be visited in the first tour without violating vehicle capacity. If order 7 was included in the first tour, total amount of demand would be 257, leading to violation of the capacity limit of 180. Then, we need to decide how many customers will be visited in the first tour. In this case, let's select integer number of 2, meaning that customers 4 and 2 will be visited in the first tour. Remaining customers to be assigned to subsequent tours will be determined according to this procedure.

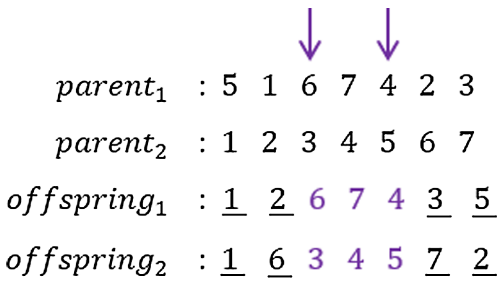
Following the implementation of three decoding schemes, evaluation process is applied to each solution to calculate its fitness value (or objective value).

5.2. Population structure

The first step in the implementation of MA is generation of initial population. A population comprises of a group of chromosomes and each chromosome represents a solution for the problem. In many applications the initial population is generated randomly. In this study, both random solutions and solutions produced according to specific principles are included in initial population. The construction of initial population with size of N can be described as follows: In initial population, the first chromosome is generated by The saving algorithm of Clarke and Wright (1964). Other chromosomes are created based on route first-cluster second methods. Route mechanism is relied upon well-known techniques involving nearest neighbor and random task. Nearest Neighbor (NN), which is an approximate algorithm to solve TSP, works in a fashion that vehicle visits the unvisited nearest customer to its current customer location until all visitation of customers are completed. Random Task (RT), which is first propounded by the work of Prins (2009), is a slightly modified version of NN. In RT, the vehicle randomly selects the candidate customer for its next visit among the unvisited customers based on a threshold value of $(0 \le \theta \le 1)$. Based on the selection of θ, the vehicle can visit a restricted number of unvisited customers. The higher the value of θ is chosen, the more likely the relatively farther customers to vehicle's current position can be visited. Finally, other half of the initial population is randomly generated.

5.3. Selection and crossover

Following the population generation, we use binary tournament mechanism in selection phase. First the number of individuals with a size of tournament are randomly chosen from the population to apply tournament mechanism. Then, the winner of tournament which has the best fitness value is selected as a parent for recombination procedure. A number of crossover operators were proposed in the literature. Here, we



$parent_1$: 5 1 6 7 4 2 3
$parent_2$: 1 2 3 4 5 6 7
$offspring_1$: 1 2 6 7 4 3 5
$offspring_2$: 1 6 3 4 5 7 2

Fig. 5. The OX procedure.

opt to choose order crossover (OX) procedure, which is developed by Davis (1985) and consistent with permutation encoding for recombination phase. An example of OX procedure is shown as in Fig. 5. First, we select two cutting points randomly. Then, genes between these points in *parent₁* (*parent₂*) are copied to *offspring₁* (*offspring₂*). The remaining genes (i.e., out-of-cutting points) needed to form *offspring₁* (*offspring₂*) are copied from *parent₂* (*parent₁*) according to the order of appearance. The resulting two offspring are generated as can be seen from Fig. 5.

5.4. Mutation

To maintain diversity, mutation operator is used for escaping from local optima in memetic algorithm. In proposed algorithm, mutation operator is used based on swap and 2-opt swap procedures. In swap procedure, two genes are randomly selected and these selected genes are interchanged. In 2-opt swap procedure, two genes are randomly selected. The genes out of cutting points are kept intact but in between, the genes are reversed. For instance, let $(1-2-6-7-4-3-5)$ be the chromosome to which 2-opt swap applies. Let's genes 6 and 3 are selected as cutting points. After 2-opt is applied to the chromosome, the resulting chromosome would be $(1-2-3-4-7-6-5)$.

5.5. Local search

After the decision of what orders to be included in each batch is made, delivery sequence among these batches is of importance in terms of tardiness value. Although delivery sequence of batches does not change the total trip time, the change in batch sequences can normally yield different total tardiness value. We, thus, need to use local search mechanism to improve fitness value of the newly selected offspring, whereby randomly changing the sequence of batches for possible reduction in total tardiness. Let's consider the illustrative example given in Section 3.1. From Fig. 2(a) vehicle makes four delivery tours, including 0–5–0 (tour 1), 0–1–6–0 (tour 2), 0–7–4–0 (tour 3) and 0–2–3–0 (tour 4). Let tour 1 and tour 4 are imposed to pairwise interchange (that is, tour 4 is performed first and tour 1 is performed fourth). According to this change, the resulting Gantt chart is shown as in Fig. 6. As can be seen, total trip time remains unchanged (i.e., 610) but there exists a substantial change in total tardiness value, which increased from 165 to 868. After pairwise interchange, new tardiness value of each customer is calculated as ($T_1 = 101$, $T_6 = 38$, $T_7 = 82$, $T_4 = 225$ and $T_5 = 422$). Since this random pairwise interchange makes the total tardiness value worse than the current one, this is not permitted.

5.6. Updating population

Diversity can only be sustained for future generations only if a population in a particular generation consists of unique chromosomes. To achieve this aim, we use an updating mechanism during the inclusion of newly generated offspring into population. After crossover and mutation operators are used to generate two new offspring, three possible situations can occur: (*i*) both of the offspring (*ii*) either one of the offsprings or (*iii*) none of the offspring may already exist in population. In case (*i*), one of the offspring is randomly selected and mutated. In case (*ii*), the offspring not existing in the population is selected to compare with the individuals. In case (*iii*), the offspring with the best objective value is selected to compare with the individuals in population. If selected offspring in each of three cases have better fitness value than the chromosome with worst fitness value in the population, the offspring is replaced with that chromosome and added into the population.

6. Generation of test instances

In order to assess the performance of the heuristic over MILP reliably, the generation of test instance is of great importance. During the design of test instances, we classify them into three groups, namely small, medium, and large sized instances. Small sized instances contains customer numbers of 5, 6, 7 and machine numbers of 2 and 3. Medium sized instances involve customer numbers 10 to 15 with an increment of 1 and machine numbers of 3, 4, and 5. Large size instances cover customer numbers 20 to 50 with an increment of 10 and machine numbers of 4, 5, and 6. In order to see the impact of due dates on solution, three levels are chosen as tight, medium, and wide. Another parameter which is thought of having an impact on solution is the variability of processing times on machines. Impact of the variability is measured by the inclusion of a parameter μ. Two levels are chosen for parameter of μ as 0.1 and 0.3. For each combination, five test instances are generated. We, therefore, generate $3 \times 2 \times 3 \times 2 \times 5 = 180$ instances for small sized groups, $6 \times 3 \times 3 \times 2 \times 5 = 540$ instance for medium sized groups, and $4 \times 3 \times 3 \times 2 \times 5 = 360$ instances for large sized groups. In total, $180 + 540 + 360 = 1080$ test instances are generated. The parameters with their chosen levels are shown in a tabular form given in Table 3.

We are inspired by the work of Ullrich (2013) during our test instance generation procedure. First, we set a general ρ value which will be a base in generating all other parameters. Table 4 shows the instance generation procedure. As seen in the first row, the demand size d_i of order i is assumed to be uniformly distributed and represented by $UNIF(\rho/2, \rho)$. While generation of processing time p_{im} of order i on machine m, first unit processing time value for each order on each machine is drawn from a discrete uniform distribution $UNIF(1-\mu, 1+\mu)$ then p_{im} is calculated as demand size of order i multiplied by unit processing time. As regards to vehicle capacity, we reasonably assume that it cannot be lower than maximum demand size. The upper limit for capacity is set half of the total demand size of all customer orders. In generation of the travel time matrix, we first randomly select x and y coordinates of each customer on a two-dimensional plane and then calculate the Euclidian distances between each pair of customers. The distances are assumed to be symmetric (i.e., $t_{ij} = t_{ji}; \forall\, i, j = 0, \cdots, N$). We generate the customer locations in a fashion that the maximum travelling time between each pair of customers does not exceed the threshold value of α. In order to effectively manage both production and distribution operations, balancing between processing and travelling times are significant. If processing times are relatively larger than travelling times, machine scheduling problems turns out to be trivial and the integrated problem boils down to the vehicle routing problem. In practice, however, processing and travelling times are well-fitted. While we use a coefficient $(M + N - 1)$ as multiplier in production stage, we use $(N + 1)$ where M and N denote the number of machines and orders respectively in distribution stage. After calculation of α, customers are randomly located in this geographical region. Thus, triangle inequality are preserved. In the last row, tight, medium and wide due date generations are calculated based on related expressions.

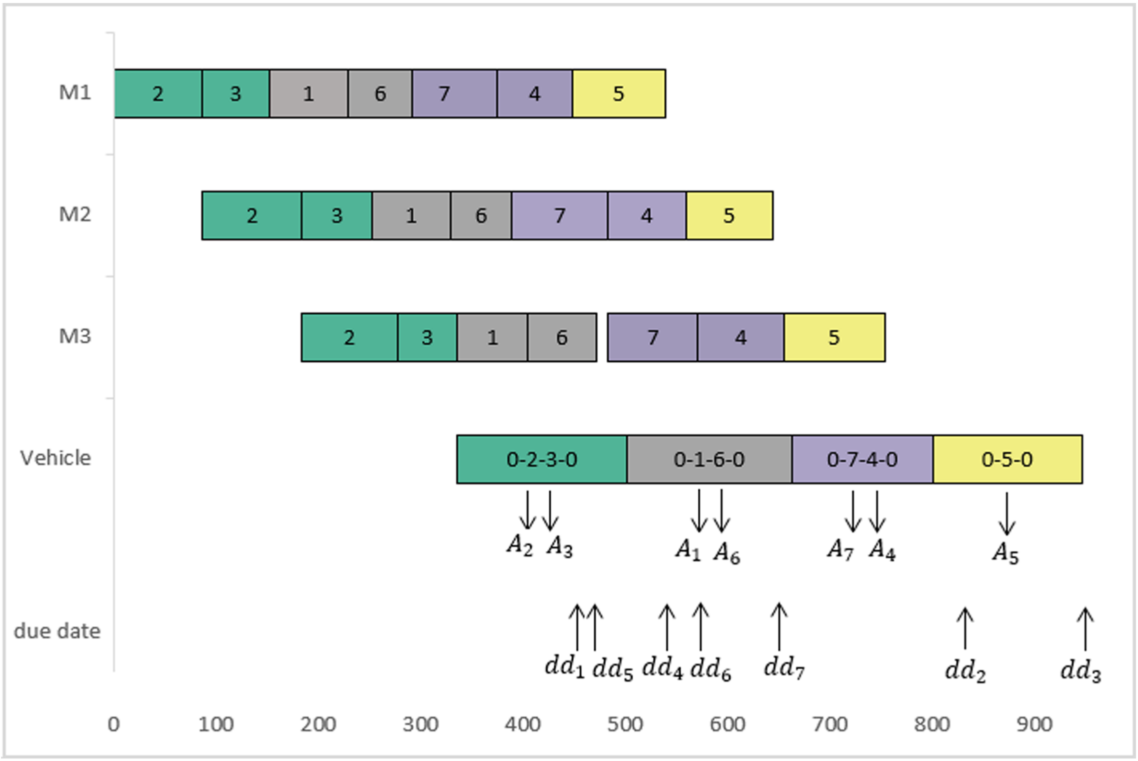


Fig. 6. Effect of batch exchanging for tardiness.

Table 3
Test instances structure.

Instance Type	Number of customers	Number of machines	dd_i	μ
Small sized	5, 6, 7	2, 3	tight, medium, wide	0.1, 0.3
Medium sized	10, 11, 12, 13, 14, 15	3, 4, 5	tight, medium, wide	0.1, 0.3
Large sized	20, 30, 40, 50	4, 5, 6	tight, medium, wide	0.1, 0.3

Table 4
Generation of test instances.

Parameter	Equations
d_i	$UNIF\left(\frac{\rho}{2}, \rho\right)\rho = 100$
p_{im}	$UNIF(1-\mu, 1+\mu)d_i\mu = 0.1, 0.3$
Q	$UNIF\left(maxd_i, \frac{\sum d_i}{2}\right)$
α	$= \lfloor\frac{3\rho(M+N-1)}{2(N+1)}\rfloor$
(X_0, Y_0)	$= \left(\frac{a}{2}, \frac{a}{2}\right)a = \sqrt{\frac{maxt_{ij}^2}{2}}$
(X_i, Y_i)	$UNIF(0, a)$
dd_i	$tight = \sum_m p_{im} + t_{oi} + UNIF\left(0, \frac{\pi}{4}\right)medium = \sum_m p_{im} + t_{oi} + UNIF\left(0, \frac{\pi}{3}\right)\pi = \rho[(M+N-1)+(N+1)]wide = \sum_m p_{im} + t_{oi} + UNIF\left(0, \frac{\pi}{2}\right)$

7. Experimental results

Computational experiments are conducted on a workstation with two four-core Intel Xeon E31245 at 3.30 GHz with 8 GB RAM. CPLEX solver embedded to GAMS version of 24.2 is used as the MILP-solver. The maximum computational time for CPLEX solver is set to 3 h (10800 sec.). This section is divided into three subsections.

7.1. Effects of varying number of customers and machines on performance of CPLEX

This section is devoted to interpreting the overall performance of CPLEX with respect to different number of customers and machines. Table 5 reports the computational results obtained by CPLEX. First and second column represent the customer (N) and machine number (M), respectively. Third column (*Objective_value*) shows the best feasible solution found by CPLEX until the time limit of 10,800 sec. The fourth column (*Lower_bound*) represents the lower bound value provided by CPLEX and is found by relaxing the integer/binary decision variables. Therefore, a lower bound does not guarantee feasibility unless lower bound and upper bound found are the same. The fifth column (*%GAP*) represents the percentage gap value and is calculated as $100 \times (Objective\_value - Lower\_bound)/(Objective\_value)$. The value of 0 for %GAP means that optimal solution is found for a particular test

Table 5
Computational results obtained by CPLEX.

N		*M*	*Objective_value*	*Lower_bound*	*%GAP*	*# of optimal*	*CPU in sec.*
Small	***5***	***2***	519.47	519.47	0.00	30/30	0.39
		3	623.93	623.93	0.00	30/30	0.32
	6	***2***	460.27	460.27	0.00	30/30	3.08
		3	786.40	786.40	0.00	30/30	3.54
	7	***2***	617.47	617.47	0.00	30/30	49.28
		3	775.17	775.17	0.00	30/30	33.60
Medium	***10***	***3***	1187.67	868.14	23.69	15/30	7118.17
		4	1120.67	618.42	37.66	6/30	9004.55
		5	1477.23	713.17	40.79	7/30	8595.79
	11	***3***	1268.37	556.88	45.54	3/30	10350.61
		4	1432.90	695.20	42.39	4/30	10213.25
		5	1439.80	889.14	29.05	10/30	7528.41
	12	***3***	1626.27	511.09	57.19	1/30	10757.46
		4	1734.00	827.65	40.10	6/30	8773.78
		5	1712.97	813.51	44.08	4/30	9687.43
	13	***3***	1820.80	689.39	49.84	2/30	10081.33
		4	1656.23	564.42	59.33	0/30	10800.00
		5	1937.47	698.93	55.86	2/30	10081.28
	14	***3***	2120.63	744.11	53.09	0/30	10800.00
		4	2035.13	872.34	46.70	6/30	9414.67
		5	1931.73	743.75	53.02	0/30	10800.00
	15	***3***	2035.40	633.04	57.13	4/30	9978.09
		4	2307.00	507.97	70.48	0/30	10800.00
		5	2385.27	947.18	48.63	2/30	10083.67
Large	***20***	***4***	3473.60	717.75	72.69	0/30	10800.00
		5	3690.90	720.00	73.26	0/30	10800.00
		6	3573.47	622.96	73.06	0/30	10800.00
	30	***4***	7078.117	879.28	78.86	0/30	10800.00
		5	7661.13	556.58	85.93	0/30	10800.00
		6	10129.17	1167.07	84.00	0/30	10800.00
	40	***4***	17671.10	585.24	93.58	0/30	10800.00
		5	17614.63	629.87	93.53	0/30	10800.00
		6	19638.07	640.37	96.08	0/30	10800.00
	50	***4***	30396.30	1235.38	94.94	0/30	10800.00
		5	32186.60	646.03	97.67	0/30	10800.00
		6	30076.97	1126.43	95.24	0/30	10800.00

instance. The sixth column (*# of optimal*) shows the number of solutions found to optimality among 30 instances for each combination of customer and machine. The last column (*CPU in* sec.) represents CPU time in sec. When CPU time is found to be lower than maximum time limit of 10,800 sec., this indicates that solution to optimality is found for that instance.

It is seen from Table 5 that CPLEX can provide optimal solutions for all small sized instances in quite short times. In medium sized instances, increasing number of customer leads to a dramatic decrease in the number of optimal solutions and a substantial increase in CPU times. As regards to large sized instances, CPLEX cannot reach optimal solution in any instances within the given time limit. In addition to this, the GAP value has increased dramatically. Another finding from Table 5 is that increasing number of machines has no impact on %GAP, and CPU times but expectedly leads to an increase in objective value.

7.2. Effect of variability on processing times: Comparison of CPLEX and MA results

In this chapter, we discuss the relative performances of CPLEX and MA under two different processing time variability factors. Before proceeding results, we give details about the selection of parameter values of the heuristic.

The heuristic algorithm is coded using C# programming language. The performance of the algorithm is fine-tuned, whereby best levels of parameters are determined based on pilot runs. Based on pilot runs, best values of crossover and mutation rates are set to 0.90 and 0.10. Population size consists of 30 chromosomes. Maximum number of iteration is selected as a function of customer number (N). This number is chosen $n \times 50$, $n \times 100$, $n \times 200$ for small sized, medium sized, and large sized instances, respectively.

Table 6 presents the comparative results of MA and CPLEX results. The first column (μ) denotes the variability coefficient of processing times. As indicated in Section 6, processing time p_{im} is calculated as $p_{im} = UNIF(1-\mu, 1+\mu)d_i$. Hence, an increase in μ also increases the variability in processing times. The values of 0.1 and 0.3 are selected to

Table 6
Comparative results of the MA and CPLEX under different μ values.

		MA				*CPLEX*		
μ	*N*	*% GAP_best*	*% GAP_worst*	*GAP_avg.*	*CPU in sec.*	*% GAP*	*# optimal*	*CPU in sec.*
0.1	***5***	0.00	0.29	0.16	0.22	0.00	30/30	0.52
	6	0.07	0.93	0.45	0.23	0.00	30/30	3.85
	7	0.62	1.87	1.26	0.25	0.00	30/30	44.44
	10	3.39	7.88	5.52	0.80	0.01	15/45	8187.65
	11	4.16	9.50	6.75	0.93	0.06	11/45	9154.68
	12	5.69	9.59	7.56	1.15	0.07	6/45	9720.72
	13	6.77	11.89	9.06	1.23	0.06	2/45	10320.79
	14	7.95	13.54	10.61	1.48	0.13	3/45	10100.31
	15	11.63	15.70	13.80	1.73	0.10	3/45	10297.29
	20	10.73	18.05	14.39	6.40	0.45	0/45	10800.00
	30	7.88	11.80	9.60	14.27	9.19	0/45	10800.00
	40	0.01	0.17	0.10	25.22	69.70	0/45	10800.00
	50	0.00	0.00	0.00	35.99	146.70	0/45	10800.00
0.3	***5***	0.03	0.07	0.04	0.23	0.00	30/30	0.20
	6	0.18	0.90	0.48	0.23	0.00	30/30	2.77
	7	0.56	2.34	1.36	0.25	0.00	30/30	38.44
	10	3.88	7.42	5.73	0.82	0.00	13/45	8291.36
	11	4.80	9.54	7.21	0.98	0.03	6/45	9573.51
	12	8.00	11.53	9.70	1.17	0.04	5/45	9758.39
	13	9.51	15.06	11.90	1.25	0.26	2/45	10320.78
	14	10.96	15.99	13.15	1.49	0.27	3/45	10576.18
	15	14.93	19.25	17.00	1.76	0.54	3/45	10277.26
	20	14.81	21.05	18.05	5.22	0.39	0/45	10800.00
	30	12.60	17.73	14.90	12.39	9.44	0/45	10800.00
	40	0.39	1.32	0.60	21.75	46.09	0/45	10800.00
	50	0.00	0.00	0.00	30.68	132.69	0/45	10800.00

Table 7
Comparative results of the MA and CPLEX under different due date levels.

		MA				*CPLEX*		
dd_i	*N*	*% GAP_best*	*% GAP_worst*	*GAP_avg.*	*CPU in sec.*	*% GAP*	*# optimal*	*CPU in sec.*
wide	***5***	0.00	0.00	0.00	0.28	0.00	20/20	0.23
	6	0.08	0.23	0.14	0.23	0.00	20/20	1.71
	7	0.65	2.05	1.22	0.25	0.00	20/20	27.71
	10	2.31	6.79	4.47	0.75	0.00	18/30	5430.47
	11	3.02	7.95	5.27	0.88	0.07	14/30	7260.08
	12	5.34	11.07	8.03	1.05	0.10	9/30	8011.44
	13	6.37	13.64	9.54	1.17	0.05	4/30	9362.25
	14	6.97	13.47	9.82	1.36	0.00	6/30	9414.67
	15	14.52	20.19	17.24	1.60	0.13	6/30	9261.74
	20	17.39	29.53	23.69	5.21	0.44	0/30	10800.00
	30	19.85	29.29	24.12	12.80	10.08	0/30	10800.00
	40	0.59	1.96	0.90	22.53	97.54	0/30	10800.00
	50	0.00	0.00	0.00	32.31	296.70	0/30	10800.00
medium	***5***	0.00	0.06	0.02	0.22	0.00	20/20	0.38
	6	0.27	0.89	0.65	0.23	0.00	20/20	3.14
	7	0.40	2.18	1.22	0.25	0.00	20/20	19.60
	10	4.02	8.15	5.91	0.84	0.01	6/30	9133.94
	11	5.81	12.59	9.44	0.97	0.00	3/30	10032.17
	12	9.05	12.66	10.84	1.21	0.02	2/30	10407.18
	13	10.82	16.20	13.23	1.26	0.20	0/30	10800.00
	14	14.30	21.02	17.53	1.52	0.11	0/30	10800.00
	15	16.46	21.07	19.11	1.78	0.48	0/30	10800.00
	20	12.80	17.62	15.18	6.14	0.75	0/30	10800.00
	30	8.49	11.47	9.64	13.30	9.89	0/30	10800.00
	40	0.00	0.00	0.00	24.30	55.86	0/30	10800.00
	50	0.00	0.00	0.00	33.69	82.30	0/30	10800.00
tight	***5***	0.04	0.48	0.28	0.21	0.00	20/20	0.46
	6	0.02	1.62	0.61	0.23	0.00	20/20	5.08
	7	0.72	2.08	1.48	0.25	0.00	20/20	77.01
	10	4.57	8.00	6.49	0.85	0.00	4/30	10154.10
	11	4.61	8.01	6.23	1.01	0.06	0/30	10800.00
	12	6.15	7.95	7.02	1.23	0.04	0/30	10800.00
	13	7.23	10.60	8.67	1.30	0.23	0/30	10800.00
	14	7.10	9.80	8.30	1.56	0.49	0/30	10800.00
	15	8.85	11.16	9.84	1.85	0.36	0/30	10800.00
	20	8.10	11.50	9.80	6.09	0.08	0/30	10800.00
	30	2.38	3.52	3.00	13.87	7.98	0/30	10800.00
	40	0.01	0.27	0.15	23.61	20.29	0/30	10800.00
	50	0.00	0.00	0.00	34.00	40.09	0/30	10800.00

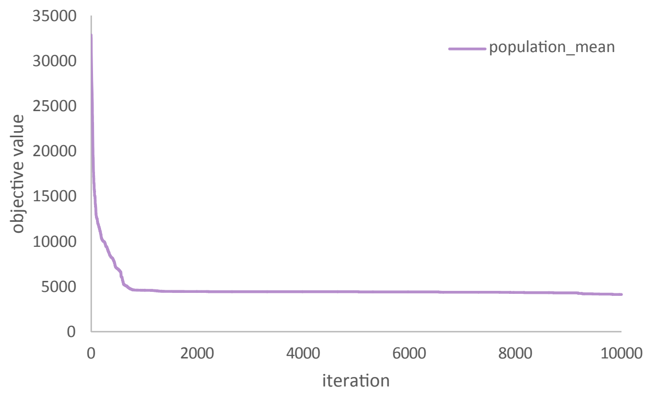


Fig. 7. Convergence of MA.

see the effect of this parameter. The second column represents the number of customer. The heuristic algorithm is run three times for each instance. Third (*% GAP_best*), fourth (*% GAP_worst*) and fifth (*% GAP_avg.*) columns give the average percentage gap values of MA against CPLEX. While sixth and ninth columns CPU times provided by heuristic algorithm and CPLEX, eight column represents the number of solutions to optimality found by CPLEX. *%GAP_best*, *%GAP_worst*, and *% GAP_avg.* represent the best, worst, and average solution provided by the heuristic among three runs for a given instance. These values are accordingly calculated as $100 \times (Heuristicsolution - CPLEXsolution)/(CPLEXsolution)$ if solution found by heuristic is worse (higher) than the solution by CPLEX. Finally, *%GAP* is calculated as $100 \times (CPLEXsolution - Heuristicsolution)/(Heuristicsolution)$ if solution found by CPLEX is worse (higher) than the solution by heuristic.

Findings from Table 6 can be summarized as follows. The heuristic can find solution to optimality for most of the small sized instances generated. For medium sized instances, relative performance of heuristic algorithm over CPLEX worsens. This is because CPLEX is still capable of finding optimal solutions within time limit of 3 h for some of the instances. The likelihood of finding optimal solution for the heuristic diminishes due to enlarged solution space. The situation reverses the for large sized instances, though. The solution performance of the heuristic overwhelmingly outperforms CPLEX. In particular, *%GAP* value dramatically increases to 69.70 and 146.70 when $\mu = 0.1$ and customer numbers are 40 and 50. Similar results occur when $\mu = 0.3$. Another remark worth emphasizing is associated with different values of μ. When value of μ is increased from 0.1 to 0.3, performance of both heuristic and CPLEX degrades. This means that increased variability in processing times also leads to an increase in complexity of the problem. Finally, it is clear that the heuristic can provide solutions less than a minute even for the largest size of instance. But, solution time of CPLEX is computationally forbidden and is not amenable for practical purposes.

7.3. Effect of due dates: Comparison of CPLEX and MA results

Another important parameter that should be investigated to see its effect on results provided by both CPLEX and the heuristic algorithm is due dates of the customers. As tardiness values are calculated based on due dates of customers, it is worthwhile to observe the changes in objective value with varying levels of due dates. We, thus, choose three different levels of due dates, labeled wide, medium, and tight while generating instances. While wide due date designates a longer latest time for delivery to customer, tight due date represents shorter latest time. The calculation formula for each these levels can be seen in Table 4.

In Table 7, all column labels are exactly the same in Table 6, excluding first column, which now represents due date level. It is seen From Table 7 that when due date level is changed from wide to medium and from medium to tight number of solution found to optimality notably decreases and solution times rapidly increases. Since tardiness value is positive difference between arrival time and due date for any customer and total tardiness contributes to objective function, non-zero tardiness values for any customer lead to more space search for CPLEX. As regards to relative performances of the heuristic and CPLEX on solution quality following remarks can be stated. For small sized instances under all due date levels, *%Gap_best* is very close to zero, indicating the favorable performance of the heuristic. For medium sized instances, when due date level is changed from wide to medium, the relative performance of the heuristic over CPLEX slightly worsens. Conversely, when due date level is changed from medium to tight the relative performance of the heuristic against CPLEX rises up again more remarkably. For large sized instances, When due date is changed from wide to medium and medium to tight, the *%Gap_best* value decreases, indicating that relative performance of the heuristic over CPLEX increases. In terms of computational time, the heuristic finds solutions in quite shorter times as compared to CPLEX.

7.4. Convergence of MA

The success of any algorithm is frequently measured how rapidly it converges. This sub section is to show the convergence speed of the heuristic developed. Fig. 7 demonstrates convergence behavior for a big sized instance involving customer number of 50. As maximum iteration number is calculated as $50 \times 200 = 10,000$, Fig. 7 draws the average objective values of all chromosomes (in y-axis) against each iteration (in x-axis). It is obvious that the developed heuristic can rapidly converges around iteration 1000, after which small improvements also are observed until the end of maximum iteration number.

8. Conclusions and future directions

In this paper, we studied a joint production scheduling and distribution planning problem where a manufacturer has committed to processing customer orders on a permutation flow-shop environment and subsequently delivering them to the customers by a single capacitated vehicle. As each customer place its order with a pre-determined due date, tardiness may occur for an order which is delivered to destined customer after due date. Owing to its limited capacity, vehicle makes as many order deliveries as possible to reduce the total trip time. This, however, leads some orders to become tardy. The objective function is therefore to minimize the total trip time plus total tardiness. In order to cope with high complexity of the problem, we assume that production and distribution sequence of a batch composing of several orders are the same.

We formulate the problem as a mixed integer linear programming (MILP). Inherent complexity of the problem does not allow mathematical formulation for providing optimal solution even in test instances with a moderate number of customers. We, therefore, present a Memetic Algorithm (MA) which is capable of providing good or near-optimal solutions in acceptable amount of time. Our experimental studies rely on comparative results of MA and CPLEX solver, whereby pointing out the impact of variability in processing times and expansion of due dates.

Based on experimental study, the following remarks can be stated. The increasing variability in processing time has a negative impact on solution quality and number of optimal solutions provided by MA and CPLEX. Likewise, tighter due date also deteriorates the solution qualities provided both of the methods. The number of machines is observed to have no effect on the problem complexity. The last, but not the least, remark is that MA finds good quality solutions and outperforms the CPLEX results. In particular, it can find better solutions than the ones provided by CPLEX in quite shorter computational times. It is also seen that MA converges quite rapidly.

This study can be extended in several points in order to shed lights on future studies. The sensitivity analysis can be done to see the effect of varying vehicle capacity on the problem. The impact of multiple vehicles with homogenous/heterogeneous capacity can be studied. Other metaheuristic combined with effective local search methods seems worthwhile to study in the future research.

References

Amorim, P., Belo-Filho, M., Toledo, F. M. B. d., Almeder, C., & Almada-Lobo, B. (2013). Lot sizing versus batching in the production and distribution planning of perishable goods. International Journal of Production Economics, 146(1), 208–218.

Armstrong, R., Gao, S., & Lei, L. (2008). A zero-inventory production and distribution problem with a fixed customer sequence. *Annals of Operations Research, 159*(1), 395–414.

Belo-Filho, M., Amorim, P., & Almada-Lobo, B. (2015). An adaptive large neighbourhood search for the operational integrated production and distribution problem of perishable products. *International Journal of Production Research, 53*(20), 6040–6058.

Chen, Z. L. (2004). Integrated production and distribution operations. In Handbook of quantitative supply chain analysis (pp. 711–745). Springer, Boston, MA.

Chen, Z. L., & Vairaktarakis, G. L. (2005). Integrated scheduling of production and distribution operations. *Manage. Sci. 51*(4), 614–628.

Chen, H.-K., Hsueh, C.-F., & Chang, M.-S. (2009). Production scheduling and vehicle routing with time windows for perishable food products. *Computers & Operations Research, 36*(7), 2311–2319.

Chen, Z.-L. (2010). Integrated production and outbound distribution scheduling: Review and extensions. *Operations Research, 58*(1), 130–148.

Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research, 12*(4), 568–581.

Davis, L. (1985, July). Job shop scheduling with genetic algorithms. In Proceedings of an international conference on genetic algorithms and their applications (Vol. 140).

Devapriya, P., Ferrell, W., & Geismar, N. (2006). Optimal fleet size of an integrated production and distribution scheduling problem for a perishable product. Paper presented at the IIE Annual Conference. Proceedings.

Devapriya, P., Ferrell, W., & Geismar, N. (2017). Integrated production and distribution scheduling with a perishable product. *European Journal of Operational Research, 259*(3), 906–916.

Ehm, J., & Freitag, M. (2016). The benefit of integrating production and transport scheduling. *Procedia CIRP, 41*, 585–590.

Elbeltagi, E., Hegazy, T., & Grierson, D. (2005). Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics, 19*(1), 43–53.

Farahani, P., Grunow, M., & Günther, H.-O. (2012). Integrated production and distribution planning for perishable food products. *Flexible Services and Manufacturing Journal, 24*(1), 28–51.

Garg, P. (2010). A Comparison between Memetic algorithm and Genetic algorithm for the cryptanalysis of Simplified Data Encryption Standard algorithm. arXiv preprint arXiv:1004.0574.

Geismar, H. N., Laporte, G., Lei, L., & Sriskandarajah, C. (2008). The integrated production and transportation scheduling problem for a product with a short lifespan. *INFORMS Journal on Computing, 20*(1), 21–33.

Hurter, A. P., & Van Buer, M. G. (1996). The newspaper production/distribution problem. *Journal of Business Logistics, 17*(1), 85.

Karaoğlan, İ., & Kesen, S. E. (2017). The coordinated production and transportation scheduling problem with a time-sensitive product: A branch-and-cut algorithm. *International Journal of Production Research, 55*(2), 536–557.

Kergosien, Y., Gendreau, M., & Billaut, J.-C. (2017). A Benders decomposition-based heuristic for a production and outbound distribution scheduling problem with strict delivery constraints. *European Journal of Operational Research, 262*(1), 287–298.

Kesen, S. E., & Bektaş, T. (2019). Integrated Production Scheduling and Distribution Planning with Time Windows. *Lean and Green Supply Chain Management* (pp. 231–252). Cham: Springer.

Lacomme, P., Moukrim, A., Quilliot, A., & Vinot, M. (2018). Supply chain optimisation with both production and transportation integration: Multiple vehicles for a single perishable product. *International Journal of Production Research,* 1–24.

Lee, J., Kim, B.-I., Johnson, A. L., & Lee, K. (2014). The nuclear medicine production and delivery problem. *European Journal of Operational Research, 236*(2), 461–472.

Li, K., Zhou, C., Leung, J. Y., & Ma, Y. (2016). Integrated production and delivery with single machine and multiple vehicles. *Expert Systems with Applications, 57*, 12–20.

Low, C., Chang, C. M., & Gao, B. Y. (2017). Integration of production scheduling and delivery in two echelon supply chain. *International Journal of Systems Science: Operations & Logistics, 4*(2), 122–134.

Meinecke, C., & Scholz-Reiter, B. (2014). A heuristic for the integrated production and distribution scheduling problem. *International Science Index, 8*(2), 290–297.

Mohammadi, S., Cheraghalikhani, A., & Ramezanian, R. (2018). A joint scheduling of production and distribution operations in a flow shop manufacturing system. Scientia Iranica. *Transaction E Industrial Engineering, 25*(2), 911–930.

Mohammadi, S., Al-e-Hashem, S. M., & Rekik, Y. (2019). An integrated production scheduling and delivery route planning with multi-purpose machines: A case study from a furniture manufacturing company. International Journal of Production Economics.

Moons, S., Ramaekers, K., Caris, A., & Arda, Y. (2017). Integrating production scheduling and vehicle routing decisions at the operational decision level: A review and discussion. *Computers & Industrial Engineering, 104*, 224–245.

Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Caltech concurrent computation program, C3P Report, 826, 1989.

Park, Y.-B., & Hong, S.-C. (2009). Integrated production and distribution planning for single-period inventory products. *International Journal of Computer Integrated Manufacturing, 22*(5), 443–457.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research, 31*(12), 1985–2002.

Prins, C., Labadi, N., & Reghioui, M. (2009). Tour splitting algorithms for vehicle routing problems. *International Journal of Production Research, 47*(2), 507–535.

Ramezanian, R., Mohammadi, S., & Cheraghalikhani, A. (2017). Toward an integrated modeling approach for production and delivery operations in flow shop system: Trade-off between direct and routing delivery methods. *Journal of Manufacturing Systems, 44*, 79–92.

Russell, R., Chiang, W.-C., & Zepeda, D. (2008). Integrating multi-product production and distribution in newspaper logistics. *Computers & Operations Research, 35*(5), 1576–1588.

Scholz-Reiter, B., Makuschewitz, T., Novaes, A. G., Frazzon, E. M., & Lima, O. F., Jr (2011). An approach for the sustainable integration of production and transportation scheduling. *International Journal of Logistics Systems and Management, 10*(2), 158–179.

Tang, K. S., Man, K. F., Kwong, S., & He, Q. (1996). Genetic algorithms and their applications. *IEEE Signal Processing Magazine, 13*(6), 22–37.

Ullrich, C. A. (2013). Integrated machine scheduling and vehicle routing with time windows. *European Journal of Operational Research, 227*(1), 152–165.

Viergutz, C., & Knust, S. (2014). Integrated production and distribution scheduling with lifespan constraints. *Annals of Operations Research, 213*(1), 293–318.