Contents lists available at GrowingScience

Uncertain Supply Chain Management

homepage: www.GrowingScience.com/uscm

Applying meta-heuristic algorithms for an integrated production-distribution problem in a two level supply chain

Maedeh Bank[a], Mohammad Mahdavi Mazdeh[a*] and Mahdi Heydari[a]

[a]*Department of Industrial Engineering, Iran University of Science and Technology, Narmak, Tehran, Iran*

C H R O N I C L E

Article history:
Received July 14, 2019
Received in revised format July 28, 2019
Accepted August 5 2019
Available online August 5 2019

Keywords:
Scheduling
Supply chain
Lifespan
Simulated Annealing
Genetic Algorithm

A B S T R A C T

Supply Chain Management (SCM) is the set of approaches used for the appropriate integration and utilization of suppliers, manufacturers, warehouses and retailers to ensure the production and delivery of products to end users in the right quantities and at the right time. Integration of the stages in the supply chain can make it more effective and profitable as a whole. In the present study, an integrated production and distribution problem in a two-stage supply chain is considered. The supply chain consists of *m* manufacturers with different locations and rates of production, and a distributer that delivers the ordered products to customers in different locations. Here, products are seasonal and perishable and must be delivered before a specified time. To characterize the problem, a Mixed Integer Programming (MIP) model is proposed and to solve the proposed model, a Hybrid Simulated Annealing (HSA) and a Genetic Algorithm (GA) with mixed repair and penalize strategies are introduced. Computational results of HSA are compared with those of the GA algorithm as the current best algorithm for solving similar problems in the literature.

1. Introduction

Supply chain (SC) is the network of organizations, people, activities, information and resources involved in the physical flow of products from suppliers to customers (Guo et al., 2016). Supply Chain Management (SCM), thus, is the process of integrating and utilizing suppliers, manufacturers, warehouses and retailers for the production and subsequent delivery of products to end users at the right quantities and at the right time. Implementation of a SC has crucial impact on the organizations' financial performance. Manufacturing and distribution companies require generic and customized software packages for the effective management of their logistics and SC activities through the selection of strategies, asset configurations, participants and operating policies. SC can be made more effective and profitable through coordinating its stages via information sharing. In other words, given all SC stages optimize their costs independently, the SC total costs will increase due to a lack of coordination. Conversely, the total costs will decrease in a coordinated SC in which individual elements may face increased costs. A total cost reduction increases the SC total sales and turnover, and profit for individual SC elements will increase in spite of their increased costs.

* Corresponding author Tel:+982173222005
E-mail address: mazdeh@iust.ac.ir (M. Mahdavi Mazdeh)

doi: 10.5267/j.uscm.2019.8.004

Integration of manufacturers and distributers is an important aspect of such coordination which has become more practical and has attracted the attention of both industry practitioners and academic researchers. In this paper, an integrated production and distribution problem in a two-stage supply chain is considered. This SC has *m* manufacturers with different rates of production and locations, and a distributer that delivers the ordered products to customers in different locations. Here, the products are seasonal and perishable, and must be delivered before a specified time. The problem hereby addressed in this paper can be reduced to a similar problem originally introduced by Chang and Lee (2004). Their problem was shown to have NP-hard complexity and the problem in this paper is also NP-hard. NP-hard problems are a class of problems in the complexity theory for which obtaining an optimal solution within a reasonable time is not possible. NP-hard problems must therefore be solved by means of heuristic or meta-heuristic approaches.

A Hybrid Simulated Annealing (HSA) and a Genetic Algorithm (GA) are proposed for solving the present problem. This is a Low-level Co-evolutionary Hybrid (LCH) algorithm. Low-level means that a part or a function of one meta-heuristic method is used in the other, giving rise to a hybrid algorithm. Co-evolutionary means that a meta-heuristic method is used as a sub-algorithm to the first one, for example as a local search. The proposed HSA algorithm uses mutation, crossover and selection concepts of GA to perform local search in the SA algorithm. The computation results obtained from the algorithm are compared with those of GA, which is the current best algorithm for solving similar problems in the literature.

The organization of this paper is as follows. A thorough investigation of literature on supply chain scheduling problems is presented in section two, the proposed mixed integer programming model of the study is described in section three, the proposed hybrid algorithm and its parameters are given in section four, and results of the computational analysis are presented in section five. Finally, the study is concluded and future work is outlined in section six.

2. Literature Review

A thorough review of literature on supply chain scheduling is presented in the following. Lee and Chen (2001) studied machine scheduling problems with explicit transportation considerations. In their models, two types of transportation situations were considered. Type-1 transportation involves intermediate transportation of jobs from one machine to another for further processing and Type-2 transportation involves the delivery of finished jobs to their destinations. Here, the transporter(s) delivered products in batches and it was assumed that the same physical space needed to be allocated to all products in the transporter. Both transportation capacity and transportation times were considered in these models. Moon et al. (2002) and Lee et al. (2002) proposed an integrated process planning and scheduling model for multi-plant supply chain which behaves as a single machine company through strong coordination. The problem was formulated mathematically by considering alternative machines and sequence-dependent setup times and due dates with the objective of minimizing total tardiness. A genetic heuristic-based algorithm was proposed for solving this problem. Hall and Potts (2003) introduced the concept of supply chain scheduling and considered a three-stage supply chain process with a supplier, a manufacturer and several customers. Here, the problem was targeted from a supplier, manufacturer and supply chain perspectives, respectively. In order to solve the first two problems (i.e. supplier and manufacturer perspectives), polynomial algorithms were presented and complexity analysis was also given for the coordination between the supplier and manufacturer. Findings of this paper demonstrated a reduction in the costs in the case of coordinated decision-making. Here, special cases with polynomial algorithms and general case complexity analysis were presented. Chang and Lee (2004) studied an extension of Lee and Chen (2001) Type-2 transportation models in which the physical space occupied by each product in a transport vehicle may be different. Three different scenarios were discussed. A proof of NP-hardness and a heuristic with worst-case analysis was provided for the problem in which jobs are processed on a single machine and delivered by a single vehicle to one customer area. At most 100% error can be caused by the heuristic under worst-case situations with a tight bound for the problem in which jobs are processed by either one of two parallel machines and delivered by a single vehicle to one customer area. Another heuristic that is 100% error bound is provided for the scenario in which jobs are processed by a single machine and delivered by a single vehicle to two customer areas.

Ryu et al. (2004) proposed a bi-level programming approach for integration, production and distribution purposes. Their goal was to determine production and inventory levels in plants and distribution centers such that production, transportation and warehousing costs would be minimized. It was hereby assumed that plants would share the available resources. Chan et al. (2005) discussed distributed scheduling problems in multi-factory and multi-product environments. Lejeune (2006) investigated the means by which costs would be minimized in a three-stage supply chain comprised of supplier, production and distribution phases. After modeling the problem by a mixed integer programming approach, the author developed an algorithm based on variable neighborhood decomposition search. Zhong et al. (2007) examined two scheduling problems with product delivery coordination. Here, each product demands a different storage space during transportation. In the first problem, the best possible approximation algorithm was presented for jobs that were processed on a single machine and delivered by one vehicle to a customer. In the second problem, which differed from the first in that jobs were processed by two parallel machines instead, an improved algorithm was given.

Mazdeh et al. (2007) considered scheduling as a set of jobs on a single machine that would deliver to customers in batches or to other machines for further processing. Here, the scheduling objective was to minimize the sum of flow times and delivery costs. Structural properties of the problem were investigated and used to devise a branch-and-bound solution scheme. Armstrong et al. (2008) studied the zero-inventory production and distribution problem with a single transporter and a fixed sequence of customers. In their problem, the product lifespan starts upon completion of production for a customer's order. The objective of this work was to maximize the total demand satisfied, without violating the product lifespan, the production/distribution capacity, and the delivery time window constraints. Several fundamental properties of the problem were analyzed and it was shown that these properties can lead to a fast branch-and-bound search procedure for practical problems. Zegordi et al. (2010) proposed a mixed integer programming model for a scheduling problem in the context of a two-stage supply chain environment with the objective of minimizing the make span. They introduced a gendered genetic algorithm named GGA with two different chromosome structures for solving the proposed problem. Fahimnia et al. (2012) developed a mixed integer non-linear formulation for a two-echelon supply network (i.e. a production-distribution network) considering the real-world variables and constraints. GA was utilized for optimizing the developed mathematical model due to its ability to effectively deal with a large number of parameters.

Yin et al. (2013) addressed a batch delivery single-machine scheduling problem in which jobs have an assignable common due window. They showed that the problem can be optimally solved in O (n^8) time by a dynamic programming algorithm under a reasonable assumption for the relationships between the cost parameters. They also show that some special cases of the problem can be optimally solved by lower order algorithms. Low et al. (2014) studied the integration of production scheduling and batch delivery problems with heterogeneous fleet of vehicles to minimize the total cost. They proposed two adaptive genetic algorithms and compared them with single plant models. Hao et al. (2015) studied a static integrated production-distribution scheduling problem with multiple independent manufacturers and developed a mixed integer programming model to maximize the weight sum of profit for each manufacturer in the supply chain under the constraint that all orders should be completed before a common deadline and that all manufacturer profits are non-negative. They used CPLEX to solve the problem. Chang et al. (2015), considered orders to be processed by unrelated parallel machines without being stored in the production stage and then, delivered to the customers by vehicles with limited capacity. The goal was to reduce the total cost, considering customer service level and the total distribution cost.

Karaoğlan and Kesen (2017) intended to integrate the production and transportation decisions in short lifespan production. The products were distributed to the customers by a single vehicle having limited capacity before the lifespan. The objective function was to determine the minimum time required to produce and deliver all customer demands. They designed a branch-and-cut algorithm for the problem. Taheri and Beheshtinia (2019) considered the problem of minimization of total tardiness and earliness of orders in an integrated production and transportation scheduling problem in a two-stage supply chain. Moreover, several constraints are also considered, including time windows due dates, and suppliers and vehicles availability times. After presenting the mathematical model of the problem, a developed version of GA called Time Travel to History (TTH) algorithm was proposed to solve the problem.

Jia et al. (2019) investigated a production-distribution scheduling problem on parallel batch processing machines with multiple vehicles. In the production stage, the jobs with non-identical sizes and equal processing time are grouped into batches, which are processed on batch processing machines. In the distribution stage, there are vehicles with identical capacity arriving regularly to transport the batches to the customers. The objective function in this paper is to minimize the total weighted delivery time of the jobs a deterministic heuristic (Algorithm H) and two hybrid meta-heuristic algorithms based on ant colony optimization (HACO, MMAS) are proposed to solve the problem. Change and lee (2004) and Zegordi et al. (2010) have the most relevance to our research. In these two problems, two-stage supply chain scenario is considered in which jobs have different sizes, manufacturers are located in a geographical zone, and vehicle travel time is taken into account. In this paper we will extend the problem by assuming that the supply chain comprises *m* production companies that act as suppliers with different production speed in the first stage. Moreover, we consider product lifespan for each job that begins upon completion of the production for a customer's order and is a real and practicable assumption in perishable industries.

3. Problem Definition

3.1. Assumptions

The proposed problem is an integrated production and distribution problem in a two-stage supply chain. The first stage in the supply chain comprises *m* manufacturers with different production rates. The second stage assumes a single vehicle with a given speed for distribution of orders from suppliers to customers. Suppose there exists *n* jobs in different sizes and the customers are in different locations. This implies a traveling time from manufacturer to customer for a job that depends on the job number and manufacturer, since every job has its own loading time and the locations of the manufacturers are different. For simplicity, we assume that that inner transportation time is negligible in comparison with the outer one (transportation time from the manufacturers to the customers. It is assumed that the vehicle is located in the distributer zone at time zero and can carry products from one manufacturer to the customers in a single batch. This is essentially a scheduling problem in which each manufacturer is considered a single machine. Products considered in this study are seasonal and perishable and have a specified lifespan. It is therefore necessary that they are delivered to the end users before this specified time. Also, the vehicle delivers the orders and returns to the distributer for the next dispatch. The objective function of the current problem aims to minimize the overall throughput in order to minimize the worst-case maximum completion time for all jobs (i.e. the make-span).

3.2. Mathematical Model

Parameters:

i	*Job index*

s	*Manufacturer index*
b	*Batch index*
vol_i	*Size of job i*
p_{is}	*Processing time for job I on manufacturer s*
cap	*Capacity of the vehicle*
t_{dis}	*Travelling time of the vehicle between supplier to cutomer*
B_i	*Life span of job i*
pr_s	*Production rate of manufacturer s*
v	*Vehicle travelling speed*

Variables:

$c_{1\mathrm{i}}(c_{2\mathrm{i}})$	*Completion time for job i during manufacturer stage (distributer stage)*
av_{b}	*Vehicle availibility time for traveling to the supplier to load bth batch*
x_{si}	1, *if job i is assigned to supplier s*, 0, *otherwise*
y_{iw}	1, *if job i produces before job w*, 0, *otherwise*
z_{ib}	1, *if job i is assigned to the bth batch*, 0, *othersie*

$min\ C_{max}$

subject to

$$\textstyle\sum_{s=1}^{m} x_{si} = 1 \qquad \forall\ i \tag{1}$$

$$c_{1i} \geq p_{is} - \textstyle\sum_{j=1}^{n} p_{js}\,(1 - x_{si}) \qquad \forall\, i, s \tag{2}$$

$$c_{1i} + \textstyle\sum_{j=1}^{n} p_{js}\,(2 + y_{iw} - x_{si} - x_{sw}) \geq p_{is} + c_{1w} \qquad \forall i, w, s\,; i < w \tag{3}$$

$$c_{1w} + \textstyle\sum_{j=1}^{n} p_{js}\,(3 - y_{iw} - x_{si} - x_{sw}) \geq p_{ws} + c_{1i} \qquad \forall i, w, s\,; i < w \tag{4}$$

$$c_{2i} \geq av_b + 2t_{dis} - M(1 - z_{ib}) \qquad \forall\, i, b \tag{5}$$

$$\textstyle\sum_{b=1}^{n} z_{ib} = 1 \qquad \forall\ i \tag{6}$$

$$\textstyle\sum_{i=1}^{n} z_{ib} \times vol_i \leq cap \qquad \forall\, b \tag{7}$$

$$av_{b+1} \leq c_{2i} + M(1 - z_{ib}) \qquad \forall\, b,\mathrm{i} \tag{8}$$

$$av_{b+1} \geq c_{2i} - M(1 - z_{ib}) \qquad \forall\ b, i \tag{9}$$

$$c_{2i} \leq c_{1i} + B_i \qquad \forall\, i \tag{10}$$

$$C_{max} \geq c_{2i} \qquad \forall\, i \tag{11}$$

$$\textstyle\sum_{b=1}^{n} z_{ib+1} \leq M \sum_{b=1}^{n} z_{ib} \qquad \forall i \tag{13}$$

$$y_{iw}, z_{ib}, x_{si} \in \{0,1\} \tag{12}$$
$$c_{2i}, c_{1i}, C_{max} \geq 0$$

Here, constraint (1) determines that every job is assigned to just one manufacturer. Constraint (2) forces the jobs' completion time in the manufacturer stage to be more than its processing time. Constraints (3) and (4) guarantee that if job *i* precedes job *j* at a same manufacturer, its completion time has to be more than job *j*. Constraint (5) shows the relationship between job completion time and vehicle availability times. Constraint (6) assures that each job is assigned only to one vehicle. Constraint (7) expresses the vehicle capacity limitation. Constraints (8) and (9) specify the time in which the vehicle becomes available for processing batch *b* + 1 as being equal to the completion time of jobs that are assigned to batch b of the vehicle. Constraint (10) ensures that difference between delivery time of each job (completion time in second stage) and its completion time in manufacturer site is less than the job's lifespan and constraint (11) ensures that C_{max} reflects the maximum delivery time (completion times for all jobs at the second stage). Finally, constraint (13) ensures that a batch cannot be filled if its previous one has been field before. As mentioned above Chang and Lee (2004) proved that their investigated problem which has just one manufacturer and one capacitated vehicle had NP-hard complexity in the strong sense. Thus, our developed model also belongs to the NP-hard class, which means that obtaining optimal solution for this problem will be challenging ever for moderate size problems. Hence heuristic or metaheuristic methods can be employed to solve the problem.

4. The Proposed Hybrid Algorithm

GA is applied to a vast array of research problems that uses meta-heuristic methods for solving integrated production-distribution problems. The aim of these methods is to obtain a near optimum solution for a given problem. The wide usage of GA algorithms together with a lack of application of different meta-heuristic methods to such problems prompted the use of Simulated Annealing methods in the current problem. Additional justifications for this selection are:

- This algorithm has proven of capable of making an escape from local minimum likely (by allowing jumps to higher energy states),
- Guarantees to find an optimum solution statistically,
- Obtains good solutions in relatively short computation times (in comparison to other methods).

To improve the ability of the SA algorithm in finding good solutions, it is hybridized with the GA (as the most frequently applied algorithm in such problems) (Hamidinia et al., 2012). This hybridization can aid the accuracy of algorithm's search in the solution space. The proposed algorithm is a Low-level Co-evolutionary Hybrid (LCH) one. Low-level means that a part or a function of one meta-heuristic method is used in the other, giving rise to a hybrid algorithm. Co-evolutionary means that a meta-heuristic method is used in the middle of the other, for example as a local search. The proposed SA algorithm uses mutation, crossover and selection concepts of GA to perform local search in the SA algorithm.

4.1. Simulated Annealing (SA)

Simulated annealing (SA) is a generic probabilistic meta-heuristic algorithm used in global optimization problems that requires locating a good approximation to the global optimum of a given function in a large search space. This algorithm can be applied in discrete spaces and combinatorial optimization problems. The SA works on the basis of temperature. The temperature is updated at each iteration of the algorithm according to an annealing schedule. The SA algorithm functions as follows: at each step in the algorithm, SA considers some neighboring state s' of the current state s, and probabilistically decides between moving the system to state s' or staying in state s. These probabilities ultimately lead the system to states of lower energy. Typically, this step is repeated iteratively until the system reaches a state that is good enough for the application, or until a given computational budget has been exhausted. The equilibrium state determines the number of iterations in each temperature. Fig. 1 shows a pseudo code of applied Simulated Annealing Algorithm.

4.2. Solution Representation

The design process for any iterative meta-heuristic requires an encoding (representation) of a solution. This is a fundamental design question and an essential design step in the development of a meta-heuristics. The encoding plays a major role in the efficiency and effectiveness of any meta-heuristic. The encoding must be suitable for and relevant to the optimization problem to be tackled. Moreover, the efficiency of a representation is also related to the search operators applied (neighborhood, recombination, etc.). Upon defining a representation, it is important that one bears in mind how the solution will be evaluated and how the search operators will operate. In the problem hereby addressed in this study, a sequence should be found for each manufacturer and distributer, respectively. Therefore, a sequencing problem is targeted and the permutation representation is used. Given m manufacturers and a single distributer, a permutation matrix with m+1 rows and *n* columns needs to be formulated.

```
Begin
Input the problem data
Initialize the initial temperature $t_0$( it has to be tuned during design of the algorithm)
Generate an initial solution and name it as s
Best=s;
Bestfun=f(s);
  counter=0;
      while ($t<t_f$)
       i=0;
      while ( i<n(t))
          generate a new solution by performing a local search and name it s'
      if (f(s')<f(s))
          replace s with s';
     if (f(s')<Bestfun)
          Best=s';
         Bestfun=f(s');
     end
     else
       calculate $\Delta f$=f(s')-f(s)
      Generate A random uniform number between 0 and 1 and name it v;
        if (v< exp ( $\Delta f$ / t ))
           replace s with s';
           i = i + 1 ;
    end while
counter=counter +1;
$t_{counter}$(temperature in iteration number counter)=T(counter)

End While

Return (Best and Bestfun)
```

Fig. 1. SA Pseudo-Code

The specific representation of this problem is composed of several strings (rows) that represent each manufacturer and the vehicle. Each cell indicates a job; suppose that there exists five jobs, two manufacturers and only one vehicle. Then, a feasible solution structure is presented in Fig. 2 as shown.

Manufacturer 1	1				
Manufacturer 2	2	3	5	4	
Vehicle	5	3	1	2	4

Fig. 2. Solution representation

This solution suggests that job 1 is assigned to manufacturer 1. Moreover, jobs 2, 3, 5 and 4 are assigned to manufacturer 2, to process job 2 first, job 3 second, job 5 third and job 4 fourth. On the other hand, the vehicle must transport jobs 5, 3, 1, 2 and 4 from the manufacturer to customers according to the proposed priority.

4.3. Initial Solution

The current problem is considered a two-stage flow shop problem. The first stage of this flow shop consists of several identical machines which are not similar to each other. The second stage of this flow shop consists of a single machine and the processing time for each job in this stage depends on its manufacturer in the first stage. The initial solution for this problem is generated by modifying the Johnson rule and the proposed heuristic algorithm is described as below:

Step 0: Consider Ω as the set of all jobs to be sequenced j=1, 2... n

Step 1: For jobs yet to be sequenced (j Є Ω), find their minimum processing times for all manufacturers and the distributor.

Step 2: If the minimum processing time is associated with manufacturers, place the corresponding job in the earliest possible position in the sequencing priority list and in case of a tie, select the job with the least job type index. If the minimum processing time is associated with the distributer, place the corresponding job in the latest possible position in the sequencing priority list and in case of a tie, select the job with the least job type index.

Step 3: Repeat *Step 2* until Ω become an empty set. The obtained sequencing priority list is an initial solution for the distributer. To find a sequencing priority list for the manufacturers, go to *step 4* and Set t=1.

Step 4: Choose the tth job from the distributer sequencing priority list and assign the job to the manufacturer with the minimum total processing time, then replace t with t+1.

Step 5: Repeat *Step 4* until all jobs are assigned to manufacturers.

4.4. SA Parameters

Although SA exhibits great capability in deriving good solutions, it is a parameter-sensitive algorithm. Performance and computation time of the SA algorithm depend heavily on parameter tuning. Prior experience with other problems proves that SA has very good computation time and yields very good solutions.

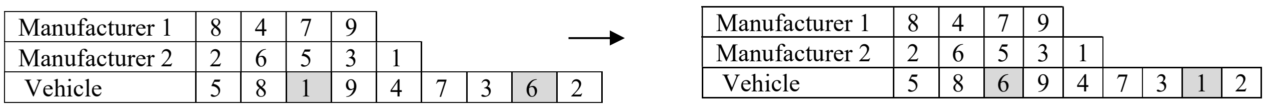
Parameters for the SA algorithm are:

- Neighborhood structure (local search)
- Initial temperature
- Production schedule (Temperature function)
- Number of neighborhood searches for each temperature (NNS)
- Stopping criteria

The following subsections describe parameters of the SA.

4.4.1. Local Search

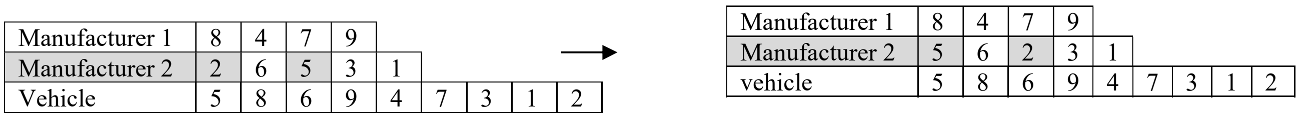
As explained before, Genetic Algorithm (GA) is the most commonly applied algorithm in integrated production and distribution problems. Additionally, it was also stated that choice of the SA algorithm for the particular problem hereby addressed is due to its appropriate computational speed and because it was not used in such problem setting before. In order to construct the most effective algorithm, it was proposed that a local search be performed using concepts of GA (i.e. mutation, crossover and selection). The local search applied in this algorithm has three stages; the first stage consists of mutation operators including insertion, inversion and swap. The second stage consists of a crossover operator that performs a reproduction for each parent pair obtained from the first stage. Finally, the third stage consists of a selection mechanism among the results of the previous stages. These three stages in the local search are described thoroughly in the following:



Manufacturer 1	8	4	7	9					
Manufacturer 2	2	6	5	3	1				
Vehicle	5	8	1	9	4	7	3	6	2

Manufacturer 1	8	4	7	9					
Manufacturer 2	2	6	5	3	1				
Vehicle	5	8	6	9	4	7	3	1	2

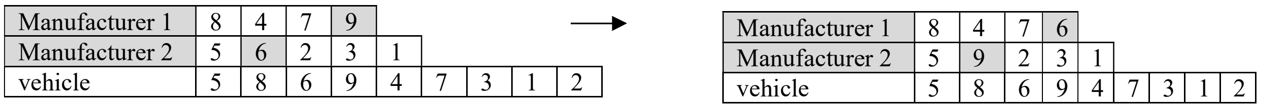
Fig. 3.a Distributer Swap Operator



Manufacturer 1	8	4	7	9					
Manufacturer 2	2	6	5	3	1				
Vehicle	5	8	6	9	4	7	3	1	2

Manufacturer 1	8	4	7	9					
Manufacturer 2	5	6	2	3	1				
vehicle	5	8	6	9	4	7	3	1	2

Fig. 3.b Manufacturer Swap Operator



Manufacturer 1	8	4	7	9					
Manufacturer 2	5	6	2	3	1				
vehicle	5	8	6	9	4	7	3	1	2

Manufacturer 1	8	4	7	6					
Manufacturer 2	5	9	2	3	1				
vehicle	5	8	6	9	4	7	3	1	2

Fig. 3.c Swap Operator between Manufacturers

Fig. 3. The applied Swap Operator

Mutation Stage

The mutation stage includes swap, insertion and inversion operators which are applied to the proposed solution representation as below:

Swap Operator

The Swap Operator changes the position of two randomly selected jobs in a sequence list. In this case, this operator is applied in three parts of the representation:

1. Performing Swap Operator in the distributer sequence. (As shown in Fig. 3.a)
2. Performing Swap Operator in each manufacturer. (As shown in Fig. 3.b)
3. Performing Swap operator between Manufacturers (As shown in Fig. 3.c). In this case, two different jobs are selected from two different manufacturers, and job positions are displaced.

In Fig. 3 it is supposed that two manufacturers, one distributer and nine jobs exist.

Insertion Operator

Insertion operator selects two jobs at random and replaces the second job to a position subsequent the first one. Here, the remainders of the jobs are shifted following this replacement. This operator is applied to the solution three times as described below:

1. Performing Swap Operator in the distributer sequence. (As shown in Fig. 4.a)
2. Performing Swap Operator for each manufacturer. (As shown in Fig. 4.b)
3. Performing Swap operator between Manufacturers (As shown in Fig. 4.c). In this case, two different jobs are selected from two different manufacturers and the second is inserted in the position after the first.

Manufacturer 1	8	4	7	9					
Manufacturer 2	2	6	5	3	1				
Vehicle	5	8	1	9	4	7	3	6	2

$\longrightarrow$

Manufacturer 1	8	4	7	9					
Manufacturer 2	2	6	5	3	1				
vehicle	5	8	1	6	9	4	7	3	2

Fig. 4.a Distributer Insertion Operator

Manufacturer 1	8	4	7	9					
Manufacturer 2	2	6	5	3	1				
Vehicle	5	8	1	6	9	4	7	3	2

$\longrightarrow$

Manufacturer 1	8	4	7	9					
Manufacturer 2	2	6	3	1	5				
vehicle	5	8	1	6	9	4	7	3	2

Fig. 4.b Manufacturer Insertion Operator

Manufacturer 1	8	4	7	9					
Manufacturer 2	2	6	3	1	5				
Vehicle	5	8	1	6	9	4	7	3	2

$\longrightarrow$

Manufacturer 1	8	4	7	6	9				
Manufacturer 2	2	3	1	5					
vehicle	5	8	1	6	9	4	7	3	2

Fig. 4.c insertion Operator between Manufacturers
Fig. 4. The applied Insertion Operator

Manufacturer 1	8	4	7	9					
Manufacturer 2	2	6	5	3	1				
vehicle	5	8	1	9	4	7	3	6	2

$\longrightarrow$

Manufacturer 1	8	4	7	9					
Manufacturer 2	2	6	5	3	1				
vehicle	5	8	6	3	7	4	9	1	2

Fig. 5.a Distributer Inversion Operator

Manufacturer 1	8	4	7	9					
Manufacturer 2	2	6	5	3	1				
vehicle	5	8	6	3	7	4	9	1	2

$\longrightarrow$

Manufacturer 1	8	4	7	9					
Manufacturer 2	2	1	3	5	6				
vehicle	5	8	6	3	7	4	9	1	2

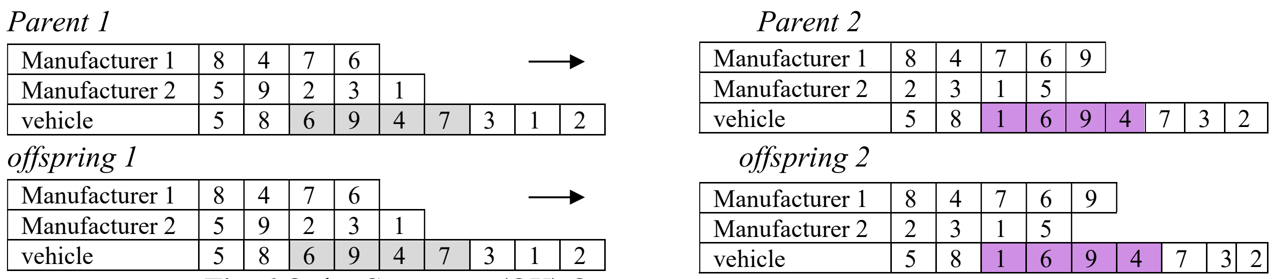
Fig. 5.b Manufacturer Inversion Operator
Fig. 5. The applied Inversion Operator

Inversion Operator

The inversion operator selects two positions in a sequence list and inverts the jobs between these two positions. This operator is applied to the distributer and manufacturers sequence list shown in Fig. 5.

Crossover Stage

The three solutions obtained from swap, insertion and inversion operators are used as the parents in the crossover stage in which the crossover operator is applied to each pair of the obtained solutions. The crossover operator used in the current algorithm has two stages. In the first stage, an Order Crossover (OX) is applied on the distributer sequence list. The OX operator selects a substring from one parent at random, produces an offspring by copying the substring into its corresponding position, and deletes values already in the substring from the second parent and finally, places the remaining values into the unfixed positions of the offspring from left to right according to the order of the sequence. The OX Operator is shown in Fig. 6.



Parent 1

Manufacturer 1	8	4	7	6					
Manufacturer 2	5	9	2	3	1				
vehicle	5	8	6	9	4	7	3	1	2

Parent 2

Manufacturer 1	8	4	7	6	9				
Manufacturer 2	2	3	1	5					
vehicle	5	8	1	6	9	4	7	3	2

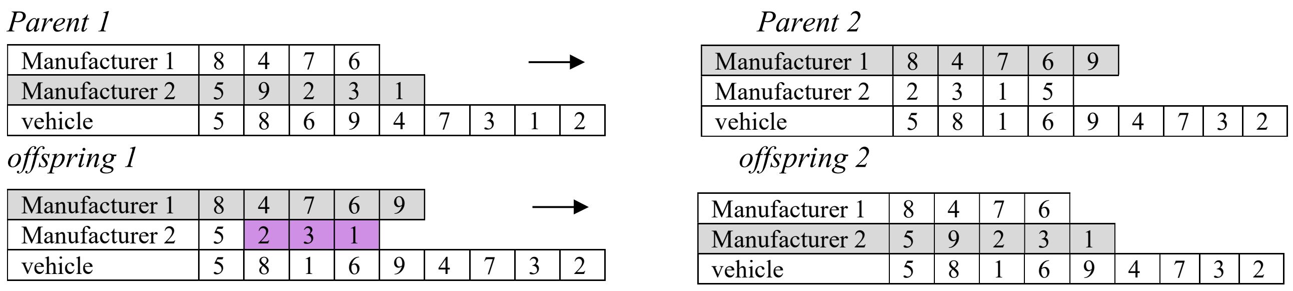
offspring 1

Manufacturer 1	8	4	7	6					
Manufacturer 2	5	9	2	3	1				
vehicle	5	8	6	9	4	7	3	1	2

offspring 2

Manufacturer 1	8	4	7	6	9				
Manufacturer 2	2	3	1	5					
vehicle	5	8	1	6	9	4	7	3	2

Fig. 6 Order Cross over (OX) Operator



Parent 1

Manufacturer 1	8	4	7	6					
Manufacturer 2	5	9	2	3	1				
vehicle	5	8	6	9	4	7	3	1	2

Parent 2

Manufacturer 1	8	4	7	6	9				
Manufacturer 2	2	3	1	5					
vehicle	5	8	1	6	9	4	7	3	2

offspring 1

Manufacturer 1	8	4	7	6	9				
Manufacturer 2	5	2	3	1					
vehicle	5	8	1	6	9	4	7	3	2

offspring 2

Manufacturer 1	8	4	7	6					
Manufacturer 2	5	9	2	3	1				
vehicle	5	8	1	6	9	4	7	3	2

Fig. 7. The proposed crossover

The second stage of the crossover operator involves the crossover between manufacturers' sequence list. Here, a manufacturer is selected randomly from one of the two parents and its sequence list is copied into the corresponding manufacturer sequence list of the offspring. Other sequences of this offspring will be filled by the sequence list of the second parent. If this newly generated solution is infeasible, it should be removed.

Selection

Following the crossover stage, nine solutions are available from which three are the parents obtained from swap, insertion and inversion operators and six are the off springs obtained from the crossover stage. Since only one solution is desirable, the Roulette Wheel selection mechanism is applied in order to select a single solution from amongst all candidate solutions.

Fig. 8. Shows Pseudo-code of the proposed local search.

```
begin(solution)
produce the parents by swap, insertion and inversion operator;
Copy the parents in the mating pool;
perform cross over between each parent pair;
copy the offspring in the mating pool;
obj=evaluate members of mating pool;
for (i=1:mating pool size)
    p(i)=p(i-1)+obj(i)/sum;
end
v=rand;
while (p(i)<v)
    i=i+1;
end
solution=i th member of mating pool;
return(solution)
```

Fig. 8. Local search pseudo-code

4.4.2. Number of neighborhood searches in each temperature (NNS)

In most previous researches, the number of neighborhood searches at each iteration of the temperature was constant. However, as mentioned before, the search scope is extensive in the initial stages of the algorithm and then gradually converges to one point or schedule. Selecting a constant number for NNS may degrade the solution quality or increase the running time of the algorithm. To avoid these disadvantages, NNS is considered a linearly decreasing function of temperature in the present study. This implies that temperature would increase as it decreases in the algorithm. The shape of this function is obtained from the literature and parameters of the decreasing function are tuned by trial and error.

4.4.3. Temperature

Initial and final temperatures play important roles in the SA algorithm. These two temperatures and the annealing schedule described in the following determine the search scope during algorithm implementation. High initial temperature permits the algorithm to replace the current solution by its worse neighboring solution. As such, the algorithm can extend its search scope in the initial steps. Also, low final temperature causes the algorithm to narrow down its search scope and find an acceptable solution. In the present study, the initial and final temperatures are equal to 10 and 0.001, respectively. These values are obtained from the literature and are adapted to the problem by trial and error.

4.4.4. Annealing Schedule

Generally speaking, two types of annealing schedules are presented in the literature: linear and exponential. In the linear schedule, for which two approaches are possible, temperature decrement at each stage is equal to a constant obtained from the difference between initial and final temperatures divided by the number of observed temperatures. In the exponential approach, the temperature is decreased by multiplying it with a constant less than 1.

4.4.5. Stopping Criteria

The algorithm shows improved performance when the linear schedule is used. Therefore, this schedule is also used in this paper for obtaining new temperatures at each stage. As described in the temperature subsection, a very small value is assigned to the final temperature. In this algorithm, the stopping criteria is set to this final temperature.

4.5. Constraint Holding

An important characteristic of an algorithm is the way it deals with infeasible solutions; this is called constrained handling. Constraint handling strategies can be classified as reject strategies, penalizing strategies, and repairing strategies. Reject strategies represent a simple approach, where only the feasible solutions are kept during the search and the infeasible solutions are automatically discarded. Repairing strategies consist in heuristic algorithms, transforming an infeasible solution into a feasible one. A repairing procedure is applied to infeasible solutions to generate feasible ones. Penalizing strategies consider infeasible solutions during the search process. The unconstrained objective function is extended by a penalty function that will penalize infeasible solutions. This is the most popular approach but many alternatives may also be used to define the penalties. To deal with a solution that contravenes the lifespan constraint, a hybrid repair and penalize strategy is proposed. The algorithm tries to repair the solution by changing the manufacturer of the job that contravenes the constraint and by changing the location of that job in the distributer sequence list to lessen its delay. In changing the manufacturer, the algorithm selects the manufacturer which minimizes the travelling time of the distributer for processing of a given job. For changing the location of the job on the distributer, a swap operation is performed between that job and the job with maximum earliness in comparison with its lifespan. If the process described above fails to make the solution feasible, it will be penalized by the following function:

$$f_p(x) = f(x) + \sum w_i d_i$$

where d_i is the delay of job i and w_i is the weight of the constraint. Here, all n lifespan constraints have the same weight. Fig. 9 shows the pseudo-code for the repair process described above.

```
Begin(solution)
for (i=1:n)
     s=solution;
     Set k as the job in position (i,m+1);
     If (job k is delivered after its lifespan)
        man=manufacturer job k;
        newman= manufacturer which causes to minimum processing time for job k on the distributer;
        s= insertion ((man,k),(newman,k),s);
        if ( penalty(s)< penalty(solution)
           solution=s;
        end
     end
end
for (i=1:n)
     s=solution;
     Set k as the job in position (i,m+1);
     If (job k is delivered after its lifespan)
        j=find job with maximum earliness in comparison with lifespan;
        s= swap ((m+1,k),(m+1,j),s);
        if ( penalty(s)< penalty(solution)
           solution=s;
        end
     end
end

Return (solution)
```

Fig. 9. Repair process pseudo-code

5.1. Test Problem

One possible way for obtaining problem instances is achieved by means of a random generator that enables production of many instances that share the same characteristics and allows gradual traversing of a range of characteristics (hardness). Moreover, randomly generated instances can be shared, allowing comparisons across different researches. Given the above-mentioned advantages of random generation, a random generator is used to obtain the test problems. The following five factors were adjusted in the present experiments: processing times (P_i), manufacturers production speed (Pr_j), number of manufacturers (m), number of jobs (n), lifespan of each job (B_i), vehicle travel speed(v) and distance between manufacturers and customers (d_{ij}). To obtain bounds for random generation, literature data were used and the problem specifications were considered, The number of jobs (n) are 20, 50 and 100 (For meta heuristic algorithms comparison) and 10 up to 15 (For algorithms validation) Manufacturers production rate and vehicle travel speeds (v) follow a uniform distribution U [1, 3] and the number of manufacturers are 5, 10 and 15 (For meta heuristic algorithms comparison) and 2,3,4 (For algorithms validation). In addition, the processing times for each job (P_i) follow a uniform distribution U [10, 30] and distances between the manufacturer and the customers (d_{ij}) are generated using a uniform distribution U [4, 10]. Finally, the lifespan also follows a uniform distribution U [5, 15]. Nine problem variants (three different sets of job numbers and three different sets of manufacturer numbers) can be generated by combining different values across the factors. Ten datasets can be randomly generated for each type (according to above ranges), leading to 90 problems altogether. Because of the stochastic nature of the meta-heuristic methods, the algorithm will be run five times for each of the 90 different problems.

Table 1
Error Percentage of two proposed algorithms

	Average		**Max**	
N	**SA-GA**	**GA**	**SA-GA**	**GA**
10	0	0	0	0
11	0	0	0	0
12	0.002922	0.008104	0.023089	0.040018
13	0.010147	0.016627	0.072157	0.081274
14	0.003251	0.018326	0.02344	0.072471
15	0.005544869	0.014485	0.055156841	0.076108

Table 2
Average, max and standard deviation of make-span obtained from the proposed algorithms.

M×J	**Average**		**Standard Deviation**		**Max**	
	SA-GA	GA	SA-GA	GA	SA-GA	GA
5×20	1308.178	1486.25	43.05913	61.783	1383.141	1598.816
10×20	1015.118	1200.34	37.61278	42.768	1091.148	1324.876
15×20	903.4638	1150.63	36.80761	33.81	950.3528	1137.25
5×50	7277.358	8263.230	107.371	121.547	7432.973	8445.324
10×50	5348.457	5367.18	108.2254	178.374	5595.14	6272.928
15×50	4694.718	5367.45	54.29885	53.997	4760.53	5514.444
5×100	27641.54	32598.17	265.3282	283.7532	27907.16	34104.68
10×100	19834.04	21589.25	350.261	391.281	20619.52	21310.81
15×100	17333.61	17151.32	131.6165	305.6165	17504.25	20701.55

5.2. Computational Results

Firstly, the proposed mathematical model was coded in GAMS 24.8 software to compare the optimal solution of the problem with the solutions obtained from two proposed algorithms. This comparison has been executed for small size problems where optimal solutions can be found in a reasonable time (less than 3 hours). As mentioned in test problem sub section, for this purpose, the number of jobs vary from 10 to 15. Table 1 shows the error percentage of two proposed algorithms in comparison with the exact solution obtained for the problem. For the purpose of evaluating the effectiveness of the proposed SA algorithm and comparing its performance with the proposed GA, computational experiments are conducted. The algorithms were coded in MATLAB 2016 software and ran on an Intel Corei5 2.5 GHZ computer. Table 2 shows the average make span criterion for 10 different instances in each problem class obtained by running each algorithm. Parameters of the instances were generated as described in the dataset section. Table 3 shows the average, max and standard deviation of CPU time for each of these three algorithms. As shown by the results given in the above tables, the SA-GA algorithm yields improved solutions in most cases whereas the GA algorithm alone yields worse solutions in less CPU time.

Table 2
Average, max and standard deviation of CPU Time of the proposed algorithms

M×J	**Average**		**Standard Deviation**		**Max**	
	SA_GA	GA	SA-GA	GA	SA-GA	GA
5×20	91.348	83.22	32.51	32.259	104.9	97.41
10×20	176.25	160.65	58.775	55.8975	221.35	202.215
15×20	246.391	227.75	80.964	75.8676	296.8	270.12
5×50	205.876	178. 84	68.991	65.0919	257.35	234.615
10×50	420.78	392.702	130.09	120.081	492.8	446.52
15×50	598.237	531.43	186.288	170.6592	711.12	643.008
5×100	436.891	402.29	146.652	134.9868	523.7	474.33
10×100	863.594	780.23	301.791	274.6119	1054.49	952.041
15×100	1267.56	1145.804	405.089	367.5801	1487.105	1341.395

6. Conclusion and Future Work

An integrated production and distribution problem in a two-stage supply chain has been studied in this paper. The problem consisted of *m* manufacturers with different locations and production rates, and a distributer that transported the ordered products to customers in different locations. In the present problem, the products were seasonal and perishable and had to be delivered to end-users before a specified time. A mixed integer programming model has been proposed for the problem for which a Low-Level Co-evolutionary Hybrid (LCH) algorithm was presented. Simulated Annealing (SA) and Genetic Algorithm (GA) were the two algorithms used in the hybridization process. Here, concepts of GA were used in the local search process of SA. The initial solution for this algorithm was generated by a proposed heuristic method. To comply with the constraints, a mixed repair and penalize strategy was introduced and finally, the results obtained were compared with those of the GA as the current gold standard for solving similar problems in the literature. The results of the current study have demonstrated improved performance of the SA-GA algorithm in yielding improved solutions in most cases whereas the GA can yield worse solutions in less CPU time. Extending the model to *k* distributers, considering other objective functions such as total tardiness and developing other meta-heuristic methods such as hybrid PSO are recommended as candidate directions for future studies.

References

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.

Chan, F. T. S., Chung, S. H., & Chan, P. L. Y. (2005). An adaptive genetic algorithm with dominated genes for distributed scheduling problems. *Expert Systems with Applications, 29*(2), 364–371.

Delavar, M. R., Hajiaghaei-Keshteli, M., & Molla-Alizadeh-Zavardehi, S. (2010). Genetic algorithms for coordinated scheduling of production and air transportation. *Expert Systems with Applications*, *37*(12), 8255-8266.

Chang, Y. C., & Lee, C. Y. (2004). Machine scheduling with job delivery coordination. *European Journal of Operational Research*, *158*(2), 470-487.

Chang, Y. C., Chang, K. H., & Kang, T. C. (2015). Applied variable neighborhood search-based approach to solve two-stage supply chain scheduling problems. *Journal of Testing and Evaluation*, *44*(3), 1337-1349.

Fahimnia, B., Luong, L., & Marian, R. (2012). Genetic algorithm optimisation of an integrated aggregate production–distribution plan in supply chains. *International Journal of Production Research*, *50*(1), 81-96.

Guo, Z., Zhang, D., Leung, S. Y. S., & Shi, L. (2016). A bi-level evolutionary optimization approach for integrated production and transportation scheduling. *Applied Soft Computing*, *42*, 215-228.

Hall, N. G., & Potts, C. N. (2003). Supply chain scheduling: Batching and delivery. *Operations Research*, *51*(4), 566-584.

Hamidinia, A., Khakabimamaghani, S., Mazdeh, M. M., & Jafari, M. (2012). A genetic algorithm for minimizing total tardiness/earliness of weighted jobs in a batched delivery system. *Computers & Industrial Engineering*, *62*(1), 29-38.

Hao, J., Cao, L., & Jiang, D. (2015). Integrated production-distribution scheduling problem with multiple independent manufacturers. *Mathematical Problems in Engineering*, Article ID 579893, 5 pages

Jia, Z. H., Zhuo, X. X., Leung, J. Y., & Li, K. (2019). Integrated production and transportation on parallel batch machines to minimize total weighted delivery time. *Computers & Operations Research*, *102*, 39-51.

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.

Hu, Y. (2004). *Performance analysis and optimization for directional residual based fault isolation* (Doctoral dissertation, George Mason University).

Lejeune, M. A. (2006). A variable neighborhood decomposition search method for supply chain management planning problems. *European Journal of Operational Research*, *175*(2), 959-976.

Lee, C. Y., & Chen, Z. L. (2001). Machine scheduling with transportation considerations. *Journal of Scheduling*, *4*(1), 3-24.

Lee, Y. H., Kim, S. H., & Moon, C. (2002). Production-distribution planning in supply chain using a hybrid approach. *Production Planning & Control*, *13*(1), 35-46.

Low, C., Chang, C. M., Li, R. K., & Huang, C. L. (2014). Coordination of production scheduling and delivery problems with heterogeneous fleet. *International Journal of Production Economics*, *153*, 139-148.

Mazdeh M. M., Sarhadi M., & Hindi K.S. (2007). A branch-and-bound algorithm for single-machine scheduling with batch delivery minimizing flow times and delivery costs. *European Journal of Operational Research, 183*(1), 74–86.

Mazdeh, M. M., Sarhadi, M., & Hindi, K. S. (2008). A branch-and-bound algorithm for single-machine scheduling with batch delivery and job release times. *Computers & Operations Research*, *35*(4), 1099-1111.

Mazdeh, M. M., Shashaani, S., Ashouri, A., & Hindi, K. S. (2011). Single-machine batch scheduling minimizing weighted flow times and delivery costs. *Applied Mathematical Modelling*, *35*(1), 563-570.

Moon, C., Kim, J., & Hur, S. (2002). Integrated process planning and scheduling with minimizing total tardiness in multi-plants supply chain. *Computers & Industrial Engineering*, *43*(1-2), 331-349.

Su, C. S., Pan, J. C. H., & Hsu, T. S. (2009). A new heuristic algorithm for the machine scheduling problem with job delivery coordination. *Theoretical Computer Science*, *410*(27-29), 2581-2591.

Taheri, S. M. R., & Beheshtinia, M. A. (2019). A Genetic Algorithm Developed for a Supply Chain Scheduling Problem. *Iranian Journal of Management Studies*, *12*(2), 107-132.

Yin, Y., Cheng, T. C. E., Hsu, C. J., & Wu, C. C. (2013). Single-machine batch delivery scheduling with an assignable common due window. *Omega*, *41*(2), 216-225.

Zandieh, M., & Molla-Alizadeh-Zavardehi, S. (2009). Synchronizing production and air transportation scheduling using mathematical programming models. *Journal of computational and Applied mathematics*, *230*(2), 546-558.

Zegordi, S. H., Abadi, I. K., & Nia, M. B. (2010). A novel genetic algorithm for solving production and transportation scheduling in a two-stage supply chain. *Computers & Industrial Engineering*, *58*(3), 373-381.

Zhong, W., Dósa, G., & Tan, Z. (2007). On the machine scheduling problem with job delivery coordination. *European Journal of Operational Research*, *182*(3), 1057-1072.