DOI: https://doi.org/10.1007/s00170-015-7313-x

Managing inventory levels and time to market in assembly supply chains by swarm intelligence algorithms

Luis A. Moncayo–Martínez[1] · Adán Ramírez–López[1] · Gustavo Recio[2]

Abstract The proposed work addresses the problem of placing safety stock under the guaranteed-service model when a set of supplying, manufacturing and delivery stages model the production system. Every stage has a set of options that can perform the stage and every option has an associated cost and time. Hence, the problem is to select an option per stage that minimises the safety stock and lead time at the same time. We proposed solving the problem using two swarm intelligent meta-heuristics, Ant Colony and Intelligent Water Drop, because of their results in solving NP-hard problems such as the safety stock problem. In our proposed algorithm, swarms are created and each one selects an option per stage with its safety stock and lead time. After that, the Pareto Optimality Criterion is applied to all the configurations to compute a Pareto front. A real-life logistic network of the automotive industry is solved using our proposed algorithm. Finally, we provided some multi-objective performance metrics to assess the performance of our approach and carried out a statistical analysis to support our conclusions.

Keywords Safety stock · Swarm intelligent · Automotive industry

✉ Luis A. Moncayo–Martínez
luis.moncayo@itam.mx

Adán Ramírez–López
adan.ramirez@itam.mx

Gustavo Recio
grecio@inf.uc3m.es

[1] Department of Industrial Engineering and Operations, Instituto Tecnológico Autónomo de México (ITAM), Mexico City, Mexico

[2] Computer Science and Engineering Department, Carlos III University of Madrid, Madrid, Spain

1 Introduction

A production system could be modelled as a set of stages that must be carried out in order to assemble the final product.

In this work, we divided the production system into supplying, manufacturing and delivery stages that can be conducted by one or more options, i.e. every stage has different ways to accomplish its task. The supplying stages are components required to produce both subassemblies and the final products, and every option could be a supplier or sourcing point. In manufacturing stages, the subassemblies and final products are assembled. Thus, an option to achieve the assembly task could be a single workstation, manufacturing cell or an assembly line. The delivery stages are the customers who ask for a final product, and the different options are the ways to deliver the product to the final customer, e.g. standard or expedited delivery.

Every option that can perform a stage has a time and cost associated, e.g. in a supplying stage, every supplier who can provide the part quotes a cost and lead time. Deciding what options to select at each stage and the amount of safety stock to place to deliver the products to the customer at the guaranteed-service time is called supply chain (SC) configuration [13]. This decision is not trivial since the decision maker selects the set of options that minimise both the safety stock cost and lead time, simultaneously. According to the concept of cost-responsiveness efficient frontier, which refers to the ability to meet short lead times at the lower possible costs, the cost is inversely proportional to the lead time [6]. Safety stock cost represents the second largest element of logistics costs in companies (the first one is transportation costs) and short lead time is needed to reduce both inventory levels and time to market.

Once the problem of choosing an option per stage is solved, the amount of safety stock (placed in every stage, to meet the guaranteed-service time) is computed. This time is the time in which a stage must serve its successive stages to allow them meeting their service time, thus in delivery stages it is the time in which customers must receive their products. The problem of placing safety stock under guaranteed-service time model has been proved to be NP-hard [16].

The contribution of this work is to develop a two-part algorithm to solve the problem of both selecting an option per stage and placing safety stock to minimise inventory cost and lead time, simultaneously. The two problems are NP hard. Thus, the problem of selecting is solved using either of these meta-heuristics: ant colony optimisation (ACO) and intelligent water drop (IWD). Due to their results in efficiently solving combinatorial problems [9, 28, 33], an algorithm based on dynamic programming [13] solves the safety stock placing problem once the meta-heuristic has selected one option per stage. Unlike the existing literature, this work includes the minimisation of both safety stock costs and lead time.

The paper is organised as follows: Section 2 presents a literature review related to safety stock under guaranteed service time inventory models and supply chain configuration. Section 3 outlines theory about swarm intelligence, specifically about ACO and IWD. Section 4 contains details about the proposed algorithm, and Section 5 describes a real life logistics network that is used to test the proposed algorithm. The results are described in Section 6. Finally, Section 7 provides some conclusions and future work.

2 Literature review

The network design configuration is one of the most important strategic decisions in industry due to its long-lasting effect on companies. Traditionally, this problem is reduced to minimising costs all over the three levels of decision making. Those costs include opening/closing facilities, operating, locating, pricing, fulfilment, and transportation [11]. Other factors encompass risk management [12], tax issues [32], and environmental concerns [30]. The techniques used to solve those models range from linear deterministic models to nonlinear stochastic ones and the most used solution techniques include traditional operational research methods and genetic algorithms [5].

Traditionally, the logistic chain is represented by a set of nodes that represent the facilities, and the links are the flow of components from one node to others [5, 11, 12, 30, 32]. In this work, the nodes are supplying, manufacturing and delivery stages, and the links are the relationships among stages, e.g. in a delivering stage, all its linked preceding stages must have been carried out to achieve the delivering task.

In relation to inventory management, in the literature, there are two approaches to set inventory levels over multistages. In the first one, called stochastic service, backorders are allowed because of the uncertain nature of the time in which the supplier provides components. Although the stochastic inventory problem is difficult, researchers and practitioners have studied it widely [2]. On the other hand, in the second approach called guaranteed-service time (GST), every stage must serve all its succeeding stages just in the promised time. Therefore, the problem is not to generate backorders but to set the amount of safety stock.

One of the first works to apply the staged representation was used to configure an SC for new products [13, 14] with just one option per stage and one cost objective. This objective includes safety and in-transit inventory costs, under GST model, as well as the Cost of Goods Sold (CoGS). A dynamic programming (DP) algorithm is used to solve the problems. A genetic algorithm solves both the selection and safety stock problems to minimise inventory cost and CoGS [15].

An extension of the problem includes resource constraints, thus every option that can perform a stage has a limited capacity [17]. The solution technique used to solve the problem is project scheduling. Another proposed extension includes variability in demand and stages' lead time; in this case, fuzzy sets [31] and Bender's decomposition [23] are the solution tools. When demand and standard deviation are unknown, an algorithm based on bass model assesses the impact of demand dynamics during new product diffusion on the configuration of SC [1].

Other approaches minimise more than one objective. In [22], the costs of safety stock (under GST) and CoGS are minimised as well as subjective objectives such as alignment of business practices and financial objectives. This approach does not minimise the lead time (LT) or time to market. On the other hand, in [20, 21], an ant colony algorithm is proposed to minimise the CoGS and LT, simultaneously. In [19], the safety stock cost and LT are minimised. They solved one theoretical instance but there is no statistical evidence of the performance of the proposed approach.

In this work, the algorithm in [19] is modified and another is proposed to compare the solution sets. Moreover, a statistical analysis is carried out to prove the efficiency of both meta-heuristics in minimising both the safety inventory cost and LT, simultaneously; subject to select one option per stage and placing safety stock under GST models.

3 Problem formulation

3.1 The supply chain configuration problem

The graph $G = \{N, L\}$ models the SC configuration problem, where the set of nodes $N = \{1, \ldots, i, \ldots, I\}$ represent the set of supplying, manufacturing and delivering stages where I is the total number of them. The set of links represents the relationships between two stages $L = \{(1, i), (i, i'), ..., (i, I)\}$. Each stage i has a set of options (O_i) that can carry out every stage, thus $O_i = \{1, \ldots, j, \ldots, J_i\}$ where J_i is the total number of options at stage i, see Fig. 1.

In the SC configuration problem, every option $j \in O_i$ has a time t_{ij} and cost c_{ij} associated. For any option j and j' than can perform a stage i, either of these situations could happen: $t_{ij} < t_{ij'}$ if $c_{ij} \geq c_{ij'}$ or $t_{ij} \geq t_{ij'}$ if $c_{ij} < c_{ij'}$. In words, when the option j can perform a stage faster than option j', the cost of j is bigger than the cost of j'. Alternatively, when the option j can perform a stage cheaper than option j', the time of j is longer than the time of j'.

Therefore, the problem of SC configuration is to select an option j per stage i given that the safety stock and lead time at the delivering stages are minimised, simultaneously. The time and cost of the selected option to perform a stage are T_i and C_i, respectively.

The lead time (LT_i) at each stage is the time of the selected option T_i plus the maximum lead time $LT_{i'}$ of stages i' that preceded i, as shown below:

$$LT_i = T_i + \max_{i'|(i',i)} \{LT_{i'}\} \tag{1}$$

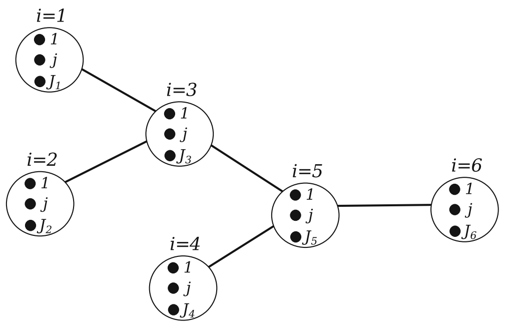

Fig. 1 Representation of the Supply Chain Configuration Problem

3.2 Safety stock problem

In the problem of safety stock, the aim is to set the amount of inventory in every stage i given that customers receive products just in a promised or guaranteed service time Ω. In this problem, it is assumed that demand is bounded to guarantee 100 % service. The total demand in any ι period is $d(\iota) = \mu\iota + z\sigma\sqrt{\iota}$ if demand is normally distributed [27]. A base stock inventory policy is assumed, thus the base stock level B is set to $B = d(\iota)$. If the demand for a period of time is $\mu\iota$, the resulting safety stock is $B - \mu\iota = z\sigma\sqrt{\iota}$, i.e. demand in ι consecutive periods is no more that z standard deviations above its mean.

Each stage i quotes a guaranteed-service time π_i in which stage i must serve its successor stages $i' \mid (i, i')$. In delivering stages, it is the time Ω in which customers must receive the products, i.e. $\pi_i = \Omega \; \forall \; i \in D$. On the other hand, preceding stages $i' \mid (i', i)$ must serve i just in the inbound service time λ_i (see Fig. 2).

$\lambda_i + T_i$ is the maximum time in which the stage can serve its successor stages, i.e. a stage can supply other stages after it has waited to be served λ_i and the stage has finished performing its task T_i. Nonetheless, the stage must serve other stages just in π_i, thus the net time to serve is $\lambda_i + T_i - \pi_i$ [13]. Therefore, the consecutive period ι of the base stock policy is $\iota = \lambda_i + T_i - \pi_i$. In this problem, ι is called the days of inventory required to accomplish the guaranteed-service time. Hence, the safety stock cost (SSC) for each stage is:

$$SSC = \sum_{i=1}^{I} \omega K_i z \sigma_i \sqrt{\lambda_i + T_i - \pi_i} \tag{2}$$

where ωK_i is the holding cost per unit per time period (ω = inventory holding cost percent per year and K_i = cumulative unit cost at stage i) and z is the number of times the standard deviation σ is above the mean, i.e. z is the percentage of time that safety stock covers demand variation [13]. For example, if the percentage is 0.95, z = 1.645.

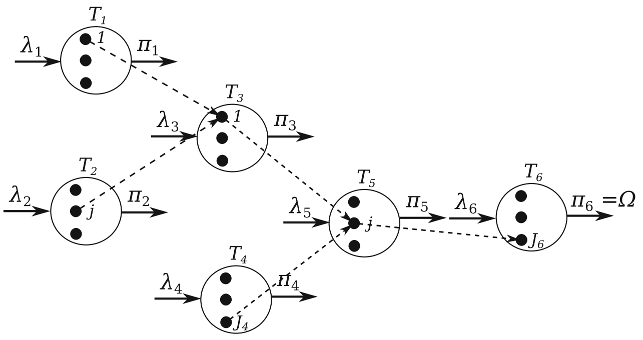

Fig. 2 Inventory levels under guaranteed-service time

So, as to model the problem, a binary variable y_{ij} is used to select an option per stage. Therefore,

$$y_{\mathrm{ij}} = \begin{cases} 1, & \text{if stage } i \text{ performs stage } i \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Formally, the problem is modelled as follows

$$\text{Min} \quad SSC = \sum_{i=1}^{I} \omega K_{\mathrm{i}} z \sigma_{\mathrm{i}} \sqrt{\lambda_{\mathrm{i}} + T_{\mathrm{i}} - \pi_{\mathrm{i}}} \tag{4}$$

$$\text{Min} \quad LT = \max_{\mathrm{i} \in \mathrm{D}} \{LT_{\mathrm{i}}\} \tag{5}$$

subject to

$$\sum_{j=1}^{J_{\mathrm{i}}} y_{\mathrm{ij}} = 1, \forall \; i \in N \tag{6}$$

$$\sum_{j=1}^{J_{\mathrm{i}}} c_{\mathrm{ij}} y_{\mathrm{ij}} - C_{\mathrm{i}} = 0, \forall \; i \in N \tag{7}$$

$$\sum_{j=1}^{J_{\mathrm{i}}} t_{\mathrm{ij}} y_{\mathrm{ij}} - T_{\mathrm{i}} = 0, \forall \; i \in N \tag{8}$$

$$T_{\mathrm{i}} + \max_{\mathrm{i}' | (\mathrm{i}', \mathrm{i})} \{LT_{\mathrm{i}'}\} = LT_{\mathrm{i}}, \forall \; i \in N \tag{9}$$

$$C_{\mathrm{i}} + \sum_{\mathrm{i}' | (\mathrm{i}', \mathrm{i})} K_{\mathrm{i}'} = K_{\mathrm{i}}, \forall \; (i', i) \in L \tag{10}$$

$$\lambda_{\mathrm{i}} + T_{\mathrm{i}} - \pi_{\mathrm{i}} \geq 0, \forall \; i \in N \tag{11}$$

$$\lambda_{\mathrm{i}'} - \pi_{\mathrm{i}} \geq 0, \forall \; i \mid (i, i') \in L \tag{12}$$

$$\pi_{\mathrm{i}} \leq \Omega, \forall \; i \in D \tag{13}$$

$$\lambda_{\mathrm{i}}, \pi_{\mathrm{i}} \geq 0 \text{ and integer}, \forall \; i \in N \tag{14}$$

The first objective (4) is the safety stock cost (SSC) and the second one (5) is the maximum lead time at each delivery stage. This time stands for the maximum lead time (LT) in which products are delivered to customers [21], thus Eq. 5 minimises the longest time taken by the network to deliver products to customers.

Equation 6 guarantees that one option per stage is selected. Equations 7 and 8 set the cost and time of the selected option to perform the stage. Therefore, Eqs. 6, 7 and 8 configure the SC.

Equation 9 sets the lead time for every stage i and Eq. 10 is the cumulative cost (K_{i}) at stage i. K_{i} is the time of the selected option T_{i} plus the cumulative cost of preceding stages i' of stage i.

Equation 11 guarantees that the days of inventory are equal or greater than zero. For two stages, linked by (i, i'), the guaranteed service time (π_{i}) of stage i must be greater than or equal to inbound service time ($\lambda_{i'}$) of stage i' (12), i.e. $\pi_{\mathrm{i}} \leq \lambda_{\mathrm{i}'}$. At delivering stages, the guaranteed-service time π_{i} $\forall \; i \in D$ is greater than or equal to guaranteed-service time Ω (or promised service time) quoted to customers. Equations 11, 12 and 13 set the amount of safety stock per stage to deliver products to customers at the promised service time Ω.

4 Swarm intelligence

Swarm behaviour is one of the main characteristics of many species in nature, e.g. flocks of birds, schools of fish, and colonies of ants. Every element in the swarm follows simple rules but collectively, the swarm is capable of achieving complex tasks, such as finding the shortest distance between nest and food in case of ant colonies or building sophisticated nests as wasps. A characteristic of a swarm is its ability to adapt itself to environmental changes that allow swarms to be robust enough and to keep alive despite changes in their environment. The strength of the swarm lies in the cooperation of every individual element by means of a communication's system, such as the pheromones in ant colonies. This communication's system among elements in the swarm forms the "collective intelligence" that is known as *swarm intelligence* (SI) [4].

Researchers in optimisation theory have developed many techniques based on the decentralised communication system that simulates the behaviour of swarms. One of the most successful examples is Ant Colony Optimisation (ACO) that was introduced to solve combinatorial problems, early in 1990s. The advantages of SI-based techniques over traditional ones are their robustness and flexibility that are required to deal with increasingly complex problems in practice and theory [3]; e.g. travelling salesman problem, scheduling, vehicle routing and nurse rostering.

4.1 Ant colony optimisation metaheuristic

The foraging behaviour of ant colonies is the main element of the ACO meta-heuristic. The aim of the colony is its survival (by bringing food from the forage area to the nest) rather than the individual survival of ants. Ants achieve their goal by finding the shortest path between the nest and food area. They communicate to each other using a chemical substance called *pheromone* that could be reinforced by other ants or evaporated by the environment.

When an ant leaves the nest, it looks for a food source randomly. Ants can smell pheromones deposited by other ants while looking for the food area. In nature, an ant both tends to choose the path that has a strong smell of pheromones and reinforces the amount of pheromones of the selected path according to the quantity and quality of food [3]. On the other hand, if ants do not reinforce pheromones in a path for some time, pheromones evaporate. As ants take longer to travel in long paths, pheromones are not reinforced at the same rate as in short paths. Hence, pheromones are highly concentrated in short paths. In this way, the pheromones guide the path selection process of ants that will look for food.

Dorigo and colleagues [8] proposed ACO meta-heuristic based on the behaviour of real ant colonies. A graph $G = \{N, L\}$ represents the problems. N is the set of nodes i and L is the set of links (i, i'). A solution (S) of the problem is a set of solution components that can be either nodes or links; e.g. if the solution to the problem is a subset of links that form a Hamiltonian path (as in the travelling salesman problem), a solution to the problem is a sequence $S = \langle \ldots, (1, i), (i, i'), \ldots \rangle$. So, as to simulate the communication system, a pheromone matrix T is used. It is usually associated with the solution components, e.g. $T = [\ldots, \tau_{(1,i)}, \tau_{(i,i')}, \ldots]$ means that ants deposit pheromones (τ) over the links. In the beginning, the solution is empty $S = \langle\rangle$ and the pheromone matrix T takes a given initial value.

An artificial ant (A) constructs a solution by adding a solution component at a time. The selection of it, at each step, is performed probabilistically based on the amount of pheromones. The transaction probability $p_{i,i'}$ is defined as the probability that ant A chooses to go to node i, if A is currently at node i.

$$p_{i,i'} = \frac{\left[\tau_{(i,i')}\right]^{\alpha} \left[\eta_{(i,i')}\right]^{\beta}}{\sum_{(i,l) \in N_i} \left[\tau_{(i,l)}\right]^{\alpha} \left[\eta_{(i,l)}\right]^{\beta}}, \tag{15}$$

where $\eta_{(i,i')}$ is the heuristic information provided by a different source from ants, e.g. either cost or time to go from i to i'; α and β are parameters that weight the relative importance of the pheromones and heuristic value, respectively, and N_{i} is the neighbourhood of node i that is defined as $N_{\mathrm{i}} = \{(i, i') \mid (i, i') \in L\}$. Notice that the solution s is feasible, thus constraints are not violated every time a node is added to the solution. When an ant finishes building a solution, the value of the objective function $f(s)$ is computed.

The following step in ACO is to update the amount of pheromones according to $f(S)$. This part depends on the version of ACO but the Ant System [8] implements the most general pheromone update rule. Firstly, the pheromone evaporation process takes place. The objective is to allow ants to exploit new paths (or sequences) to avoid the fast convergence of the algorithm to sub-optimal solutions. Therefore, pheromone evaporation is a useful way to avoid stagnation by enabling the algorithm to forget "bad" decisions. In ACO, an update evaporation rule (16) is used to evaporate pheromones. This rule depends on the evaporation factor $\rho \in (0, 1]$, which is the desired number of pheromones to evaporate.

$$\tau_{(\mathrm{i,i'})} \leftarrow (1 - \rho)\, \tau_{(\mathrm{i,i'})}, \quad \forall\, (i, i') \in L \tag{16}$$

Secondly, each ant A deposits pheromones over the selected solution components. The number of pheromones depends on the quality of the solution generated $f(S)$.

$$\tau_{(i,i')} \leftarrow \quad \tau_{(i,i')} + \sum_{A} \Delta\tau_{(i,i')}, \ \forall\, (i, i') \in L \tag{17}$$

where $\Delta\tau_{(i,i')}$ is the amount of pheromones ant A deposits. It is defined as follows:

$$\Delta\tau_{(i,i')} = \begin{cases} \frac{1}{f(S_{\mathrm{A}})}, & \text{if } i \in S_{\mathrm{A}} \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

According to Eq. 18, the shorter the path (i.e. the lower the value of $f(S_{\mathrm{A}})$), the more pheromones the links $(i, i') \in S_{\mathrm{A}}$ receive. As a general rule, the links that are used frequently and belong to short paths, receive more pheromones. Hence, those links are more likely to be selected in future iterations of the algorithm.

4.2 Intelligent water drop

The intelligent water drop (IWD) algorithm simulates the movement of natural water drops that flow into rivers, lakes and seas [24, 26]. The main idea behind IWD is that streams find optimum paths to reach the goal of travelling from a source of water to the sea, given that the environment does not allow the free movement of water drops. Based on the nature, the gravitational force pulls each water drop to the earth's centre in a straight line but a river is full of twists and turns. Hence, with no obstacle, the water drop must follow a straight path from the water source to the destination. Each water drop flowing in a river has two characteristics: *velocity* and *capacity* of carrying an amount of soil. Moreover, the water drop can transfer soil from the river bed because of its velocity. Faster water drops can collect and transfer more soil than slower ones.

The velocity and the soil of a water drop increase when it flows from one point to another. At the same time, the soil of the river bed decreases between the two points. The velocity increments depend on the amount of soil in the path; thus, the velocity increases more in a path with low soil. As a general rule, water drops prefer crossing paths with low soil because they allow water drops to move faster and to attain a higher velocity; thus , water drops collect more soil from that path.

In IWD algorithm, every artificial water drop (W) travels from the water source to destination in discrete steps. Therefore, a graph $G = \{N, L\}$ can represent the problem. A node is the water source and another is the destination.

When a drop moves from node i to node i', the drop's velocity increases $(\Delta v_{i,i'})$ nonlinearly proportional to the inverse of the soil $(s_{i,i'})$ between i to i', as follows.

$$\Delta v_{i,i'} = \frac{a_v}{b_v + c_v \left(s_{i,i'}\right)^{2\alpha}} \tag{19}$$

where a_{v}, b_{v}, c_{v} and α are positive parameters set by the user. At the same time, the soil of the drop is increased by removing some soil of the path linking i and i'. This increment $(\Delta s_{\mathrm{i,i'}})$ is nonlinearly and inversely proportional to the time $(t_{\mathrm{i,i'}})$ a water drop spent to pass from i and i'.

$$\Delta s_{\mathrm{i,i'}} = \frac{a_{\mathrm{s}}}{b_{\mathrm{s}} + c_{\mathrm{s}} \left(t_{\mathrm{i,i'}}\right)^{2\beta}} \tag{20}$$

where a_{s}, b_{s}, c_{s} and β are positive parameters set by the user. The time $t_{\mathrm{i,i'}}$ is equal to $t_{\mathrm{i,i'}} = \frac{\eta_{\mathrm{i,i'}}}{v_{\mathrm{w}}}$, where $\eta_{\mathrm{i,i'}}$ is the local heuristic function that measures the undesirability of a water drop to pass from i and i' and v_{w} is the water drop velocity.

The amount of soil removed from the path $s_{\mathrm{i,i'}}$ is:

$$s_{\mathrm{i,i'}} \leftarrow \rho_o s_{i,i'} - \rho_n \Delta s_{i,i'} \tag{21}$$

where ρ_{o} and ρ_{n} are positive numbers between zero and one. The amount of soil gathered by the water drop s_{w} and its velocity (v_{w}) are shown below:

$$\begin{aligned} s_{\mathrm{w}} &\leftarrow s_{\mathrm{w}} + \Delta s_{\mathrm{i,i'}} \\ v_{\mathrm{w}} &\leftarrow v_{\mathrm{w}} + \Delta v_{\mathrm{i,i'}} \end{aligned} \tag{22}$$

The process of selecting one node i' while the drop is at i is based on a probabilistic function $(p_{\mathrm{i,i'}})$.

$$p_{\mathrm{i,i'}} = \frac{\frac{1}{\epsilon + g_{\mathrm{i,i'}}}}{\sum_{(\mathrm{i,l}) \in \mathrm{N_i}} \frac{1}{\epsilon + g_{\mathrm{i,l}}}} \tag{23}$$

where ϵ is a small positive number to prevent a possible division by zero, N_{i} is neighbourhood of node i defined as $N_{\mathrm{i}} = \left\{(i,i') \mid \left(i,i'\right) \in L\right\}$, and $g_{\mathrm{i,i'}}$ is used to shift the soil $s_{\mathrm{i,i'}}$ of the path joining nodes i and i' toward positive values [25], as follows:

$$g_{\mathrm{i,i'}} = \begin{cases} s_{\mathrm{i,i'}}, & \text{if } \min_{(\mathrm{i,l}) \in \mathrm{N_i}} \left\{s_{\mathrm{i,l}}\right\} \geq 0 \\ s_{\mathrm{i,i'}} - \min_{(\mathrm{i,l}) \in \mathrm{N_i}} \left\{s_{\mathrm{i,l}}\right\}, & \text{otherwise} \end{cases} \tag{24}$$

As a general rule, this function prefers a path (i,i') with less soil than with more soil. The IWD algorithm builds solutions using a parametrised probabilistic model and the parameters are updated after each iteration in order to compute high quality solutions.

5 Proposed swarm optimisation based algorithm

The proposed algorithm is divided into four parts. Firstly, the chosen swarm intelligence based algorithm selects one option to perform a stage, i.e. every agent (ant or IWD) "visits" each node i and chooses the option j that carries out the task represented by the node. Secondly, the dynamic programming algorithm is used to place the amount of safety stock that enables the SC to deliver products just in the guaranteed service time. Thirdly, the algorithm both computes the values of the two objectives functions (Eqs. 4 and 5) and applies the Pareto Optimality Criterion (POC) to all the solutions to determine the set of non-dominated solutions P. Finally, the swarm-based algorithm modify the environment according to the solutions that belong to the set of non-dominated solutions.

Algorithm 1 outlines our swarm-based approach. The number of iterations X and agents W is set; notice that an

Algorithm 1 Proposed Swarm-based Algorithm

```
Require: $G = \{N, L\}$, $O_i$, $\Omega$
Ensure: $P$
 1: set the number of iterations $X$
 2: set the number of agents $W$
 3: initialise all parameters
 4: select ACO or IWD based algorithm
 5: for do $x = 1$ to $x = X$ do
 6:     for do $w = 1$ to $w = W$ do
 7:         For all $i \in N$ do
 8:         select an option $j \in O_{\mathrm{i}}$ to carry out stage $i$
            (Sect. 5.1), insert $j$ to $S_{\mathrm{w}}$
 9:         if IWD true then
10:             local update (Sect. 5.4)
11:         end if
12:     end for
13:     place safety stock to guarantee the service time
        (Sect. 5.2)
14:     compute $SSC$ and $LT$, i.e. $S_w(LT, SSC) =$
        $\langle \ldots, j, \ldots \rangle$
15: end for
16: compute the set of non-dominated solutions $\mathbf{ND}_x$ (Sect.
    5.3)
17: if $x = X$ then
18:     return $\mathbf{ND}_X$ (this is the algorithm solution)
19: else
20:     global update (Sect. 5.4)
21: end if
22: end for
```

agent can be either an ant in ACO or a drop in IWD. The parameters that must be initialised are ω, z, T, ρ, $s_{\mathrm{i,j}}$, s_{w}, v_{w}, ρ_{o}, ρ_{n}, α, β and $S = \langle\rangle$.

In the case of ACO, the pheromones lie on the options j that can perform the stage i, thus $\tau_{(\mathrm{i,j})}$ stands for the amount of pheromones over j that can perform i. On the other hand, in the case of IWD, $s_{(\mathrm{i,j})}$ stands for the amount of soil in the option j that can perform stage i.

5.1 Selection algorithm

The aim of the selection algorithm, shown in Algorithm 2, is to select one option j to perform the stage i. When this algorithm selects one option, an agent (either an ant or a drop) performs the selection task. In line 1, Algorithm 2 computes the stage's neighbourhood O_{i}. It is the set of options that can perform stage i.

If the ACO-based algorithm carries out the task of selecting an option, the Algorithm 2 (lines 2–7) gets the pheromone matrix M. Each element $\tau_{(\mathrm{i,j})}$ of M is the amount of pheromones deposited in the option $j \mid j \in O_{\mathrm{i}}$. For all options in O_{i}, Eq. 15 computes the probability ($p_{\mathrm{i,j}}$) that the agent selects option j to perform stage i. The heuristic value, used in $p_{\mathrm{i,j}}$, is defined as $\eta_{(\mathrm{i,j})} = e^{\frac{1}{t_{\mathrm{ij}}}+\frac{1}{c_{\mathrm{ij}}}}$, i.e. the time and cost of the option j.

Algorithm 2 Selection Algorithm

```
Require: $i$, $O_i = \{1, \ldots, j, \ldots, J_i\}$
Ensure: $j$
 1: compute the stage neighbourhood, $O_i = \{1, \ldots, j, \ldots, J_i\}$
 2: if ACO true then
 3:     get pheromone matrix $M = [\ldots, \tau_{(i,j)}, \ldots]$
 4:     for all $j \in N_i$ do
 5:         compute $p_{i,j}$ (Eq. 15)
 6:     end for
 7: else
 8:     get $s_{ij}$ $\forall$ $i, j$
 9:     for all $j \in N_i$ do
10:         compute $p_{i,j}$ (Eq. 23, 24)
11:     end for
12: end if
13: set a random number, $rndNumber \in (0, 1)$
14: set $j = 1$
15: while $rndNumber > 0$ do
16:     $rndNumber -= p_{i,j}$
17:     $j++$
18: end while
19: return option $j$ to perform stage $i$ (Eq. 6 is solved).
```

If the agent is a water drop, Eqs. 23 and 24 compute the option to perform the stage (lines 7–12 in Algorithm 2). Hence, s_{ij} is the amount of soil in option $j \mid j \in O_{\mathrm{i}}$.

One criterion that ants may use in order to choose one option is to select the option with the highest probability $p_{\mathrm{i,j}}$, but this could lead to stagnation. In order to cope with this problem, a *random–based proportional rule* (lines 13–18 in Algorithm 2), based on $p_{\mathrm{i,j}}$, is used by the agents to select an option. This rule states that the higher the value of the probability $p_{\mathrm{i,j}}$, the more attractive an option j is selected by the agent to perform the stage. Finally, the algorithm solves Eq. 6 (line 19).

Once an option has been selected for every stage (lines 7–13 in Algorithm 1), the cost (C_{i}) and time (T_{i}) of the selected options are known, thus Eqs. 7 and 8 are solved as well as the cumulative cost K_{i} (9) and the lead time LT_{i} (10).

5.2 Place safety stock

Graves and Willems [13] introduced the dynamic programming (DP) algorithm to place safety stock; thus, Eqs. 11, 12 and 13 are solved at the end of it. A detailed explanation of this algorithm appears on [18] and [27].

So, as to set the safety stock levels to deliver product on the guaranteed-service time, each stage is labelled as k_{i}. It is an index that states the position of i in the set of labelled stages $U = \{1_{\mathrm{i'}}, \ldots, k_{\mathrm{i}}, \ldots, I_{\mathrm{i''}}\}$. In order to label a stage, it must have just one upstream or downstream stage, i.e. stage i has the position k_{i} if there is just one $(i', i) \in L$ or one $(i, i') \in L$, see Algorithm 3 (lines 1–7).

Algorithm 3 requires that each stage has exactly one adjacent stage with a higher index because a stage could have more than one upstream and/or downstream stages.

If the stage i' is downstream from stage i (i.e. $(i, i') \in L$), the decision to be made is the guaranteed-service time π_{i} of stage i. The expected cost is denoted $\theta^{\pi}_{\mathrm{i}}(\pi)$. On the other hand, if $(i', i) \in L$, the decision to be made is the inbound service time λ_{i} of stage i, thus $\theta^{\lambda}_{\mathrm{i}}(\lambda)$ stands for the expected cost.

To know the value of $\theta^{\pi}_{\mathrm{i}}(\pi)$ and $\theta^{\lambda}_{\mathrm{i}}(\lambda)$, Eq. 25 computes the expected holding cost at stage i, see Algorithm 3 (lines 8–16).

$$\begin{aligned}\Theta_{\mathrm{i}}(\pi, \lambda) = \;& \omega K_{\mathrm{i}} z \sigma_{\mathrm{i}} \sqrt{\lambda + T_{\mathrm{i}} - \pi} \\ &+ \sum_{(\mathrm{i',i}) \in \mathrm{L} \mid \mathrm{k_{i'}} < \mathrm{k_i}} \min_{0 \le \mathrm{q} \le \lambda} \left\{\theta^{\pi}_{\mathrm{i'}}(q)\right\} \\ &+ \sum_{(\mathrm{i,i'}) \in \mathrm{L} \mid \mathrm{k_{i'}} < \mathrm{k_i}} \min_{\pi \le \mathrm{p} \le \Gamma_{\mathrm{i'}} - \mathrm{T_{i'}}} \left\{\theta^{\lambda}_{\mathrm{i'}}(p)\right\} \qquad (25)\end{aligned}$$

The first term is the holding cost at stage i. So, as to compute it, the days of inventory must be equal or greater than zero, thus constraint (11) guarantees it.

The second term is the minimum holding cost of all the upstream stages i' from i, given that the index $k_{i'}$ of upstream stages i' is less than the index k_i of stage i, (Algorithm 3, lines 10–12). Notice that the value of the guaranteed service time of the upstream stages $\pi_{i'}$ is known, thus the inbound service time λ_i of stage i is $\pi_{i'} = \lambda_i$ according to constraint (12). Hence, the optimum value of λ_i is the value of $\pi_{i'}$ where the holding cost is minimised, see line 11 in Algorithm 3.

The third term is the minimum holding cost of the downstream stages i' from i, given that the index $k_{i'}$ of upstream stages i' is less than the index k_i of stage i, (Algorithm 3, lines 13–15). In this case, the inbound service time $\lambda_{i'}$ is known, thus the optimum value of π_i is equal to $\lambda_{i'}$ (see constraint 12), where the holding cost of downstream stages is minimised.

Finally, the minimum value of the safety stock placement cost is known when $\theta_I^{\lambda}(\lambda)$ is evaluated for $k_i = I_i$ (see line 17 in Algorithm 3). The optimum value is the minimum value of $\theta_I^{\lambda}(\lambda)$ found in line 18 Algorithm 3.

The optimal values of λ and π for all stages is found by "backtracking" similar to Wagner–Whitin algorithm [7].

Algorithm 3 Safety stock placement

```
Require: $\omega_i, K_i, z_i, \sigma_i, T_i$
Ensure: minimum safety stock cost ($\theta_I^{\lambda}(\lambda)$), $\lambda_i, \pi_i$
 1: set $k \leftarrow 1$ and set $U = \{\}$
 2: while $N \neq \{\}$ do
 3:     select $i \in N$ such that there is just one $(i', i) \in L$ or
        just one $(i, i') \in L$
 4:     label $i$ with index $k_i$ and insert $k_i$ to $U$
 5:     remove $i$ from $N$
 6:     set $k \leftarrow k + 1$
 7: end while
 8: for $k = 1$ to $k = I - 1$ do
 9:     get stage $i$ at position $k$, i.e. $k_i$
10:     for all$(i', i) \in L \mid k_{i'} < k_i$ do
11:         evaluate $\theta_i^{\lambda}(\lambda) = \min_{\pi}\{\Theta_i(\pi, \lambda)\}$, for $\lambda =$
            $0, 1, \ldots, \Gamma_i - T_i$
12:     end for
13:     for all $(i, i') \in L \mid k_{i'} < k_i$ do
14:         evaluate $\theta_i^{\pi}(\pi) = \min_{\lambda}\{\Theta_i(\pi, \lambda)\}$, for $\pi =$
            $0, 1, \ldots, \Gamma_i$
15:     end for
16: end for
17: evaluate $\theta_I^{\lambda}(\lambda) = \min_{\pi}\{\Theta_i(\pi, \lambda)\}$ for $\lambda =$
    $0, 1, \ldots, \Gamma_I - T_I$
18: choose the $\lambda$ that minimises $\theta_I^{\lambda}(\lambda)$, $\lambda = 0, 1, \ldots, \Gamma_I$
    $- T_I$
```

.

5.3 Compute the non-dominated solutions

In this part of the proposed algorithm, there are W solutions or logistics network designs. Each solution $S_w(LT, SSC) = \langle \ldots, j, \ldots \rangle$ has an LT (5) and an SCC (4) associated as well as a set of selected options to perform stages. As the proposed algorithm minimises two objectives, we applied the Pareto Optimality Criterion to determine which solutions are "better" than others. As a result, Algorithm 4 returns a set of non-dominated solutions $\mathbf{ND} = \{\ldots, S_w, \ldots\}$. A non-dominated solution is a solution that cannot be improved in any of the objectives. Formally, a solution $\mathbf{s} = \{\mathbf{s}_1, \ldots, \mathbf{s}_H\}$ dominates another solution $\mathbf{s}' = \{\mathbf{s}'_1, \ldots, \mathbf{s}'_H\}$, represented for $\mathbf{s} \preceq \mathbf{s}'$, if and only if $\mathbf{s}$ is partially less than $\mathbf{s}'$, i.e. $\forall\ h \in \{1, \ldots, H\}, \mathbf{s}_h \leq \mathbf{s}'_h \wedge \exists\ h \in \{1, \ldots, H\} : \mathbf{s}_h < \mathbf{s}'_h$ [29]. Hence, a solution $\mathbf{s}$ belongs to the set of non-dominated solutions if there is no solution $\mathbf{s}'$ that dominates $\mathbf{s}$, i.e. $\mathbf{ND} := \{\mathbf{s} \mid \neg\exists\, \mathbf{s}', \mathbf{s}' \preceq \mathbf{s}\}$.

As the proposed algorithm minimises two objectives, the solution $S_w(LT, SCC)$ dominates the solution $S_{w'}(LT', SCC')$ if and only if $(LT \leq LT') \wedge (SCC \leq SCC')$ and $(LT < LT') \vee (SCC < SCC')$. So, as to include a solution to the non-dominated solutions set, the solution must not be dominated by other solutions, see Algorithm 4.

5.4 Update environment

5.4.1 IWD-based algorithm

In the basic IWD algorithm, the link between two nodes contains a certain amount of soil. However, in our algorithm, the options than can perform a stage contain a large amount

Algorithm 4 Non-dominated Solutions Set

```
Require: $\forall\ S_w(LT, SSC) = \langle \ldots, j, \ldots \rangle$
Ensure: ND $= \{\ldots, S_w, \ldots\}$
 1: for $w = 1$ to $w = W$ do
 2:     get $S_w(LT, SSC)$
 3:     for $w' = 1$ to $w' = W$ do
 4:         get $S_{w'}(LT', SSC')$
 5:         if $(LT \leq LT') \wedge (SCC \leq SCC')$ and $(LT <$
            $LT') \vee (SCC < SCC')$ then
 6:             $S_w \preceq S_{w'}$
 7:         end if
 8:     end for
 9:     if $\neg\exists\ S_{w'} \mid S_{w'} \preceq S_w$ then
10:         insert $S_w$ in ND
11:     end if
12: end for
```

of soil, i.e. the soil increments $\Delta s_{i,j}$ (20) stand for the amount of soil in option j that can perform stage i and $s_{i,j}$ (21) is the amount of soil deposited in option j. In the same way, the $\Delta v_{i,j}$ (19) stands for the drop's velocity increments when the drop chooses the option j to perform stage i.

In the case of IWD, there are two types of environmental updates. The first one is a local update procedure that takes place every time a drop selects an option to perform a stage. In this procedure, the amount of soil in both the options and the drop decreases as well as the drop's velocity increments. The second one, called global update, carries out the process of updating the soil of the options that form a non-dominated solution, thus, the set of non-dominated solutions must be computed before running the global update procedure.

Algorithm 5 (lines 1–5) describes the local update procedure in detail, see Algorithm 1 lines 7–12.

So, as to run the global update procedure, the non-dominated solution set $ND_x = \{\ldots, S_w, \ldots\}$ has been computed; thus, it is an input to update environment procedure (Algorithm 5 line 7).

The global update procedure, Algorithm 5, lines 8–10, removes an amount of soil of the selected option in a non-dominated solution using Eq. 26.

$$s_{i,j} = (1-\rho_n)s_{i,j} + \rho_n \frac{2 s_w}{|N|\,(|N|-1)}, \quad j \in S_w, S_w \in ND_x \tag{26}$$

where $|N|$ is the number of stages.

5.4.2 ACO-based algorithm

In ACO meta-heuristic, there is just a global update process taking place once all ants have built a solution. Our proposed algorithm evaporates an amount of pheromones of the selected options in a non-dominated solution.

The set of non-dominated solutions is input to this part of the algorithm (Algorithm 5 line 7). Firstly, the algorithm gets the pheromone matrix which represents the amount of pheromones in all the options for all the stages. Secondly, an amount of pheromones evaporates from all the options using Eq. 16, so as to avoid stagnation. Finally, the algorithm deposits an amount of pheromones in options that belong to a non-dominated solution, see Algorithm 5 lines 11–13.

Algorithm 5 Update Enviroment

```
Require: the selected option $j$ to perform stage $i$
Ensure: updated environment
 1: if localUpdate = true then
 2:     get $s_{i,j}$, $s_w$, and $v_w$
 3:     compute the increments $\Delta v_{i,j}$ (Eq. (19)) and $\Delta s_{i,j}$ (Eq. (20))
 4:     remove an amount of soil from the option $j$ to perform stage $i$, $s_{i,j}$ (Eq. (21))
 5:     update water drop' soil ($s_w$) and velocity ($v_w$) (Eq. (22))
 6: else
 7:     get the set of non-dominated solutions, $\mathbf{ND}_x = \{\ldots, S_w, \ldots\}$
 8:     If IWD = true then
 9:         update the soil of the selected options $j \mid j \in S_w, S_w \in ND_x$ (Eq. (26))
10:     else
11:         get the pheromone matrix $T = [\ldots, \tau_{(i,j)}, \ldots]$
12:         evaporate pheromones of all options (Eq. (16))
13:         deposit pheromones of the selected options $j \mid j \in S_w, S_w \in ND_x$ (Eq. (17))
14: end if
15: end if
```

6 Experimental application

So, as to prove our proposed algorithm, we carried out a study in a company that assembles fixed brakes and clutch pedals modules. The company is located in the business automotive cluster in Northeast Mexico and manufactures twenty different modules for customers such as Nissan, Ford, GM, Mazda, Fiat, etc.

Figure 3 depicts the actual logistic network for one of the best selling-products of the company. Table 1 shows the data related to the cost and time per stage. Although the logistics network is the actual one, we modified the information in Table 1 as the company requested.

Most of the process encompasses stages of cutting, folding and welding components obtained from external suppliers. The final product ($i = 27$) is produced when the main

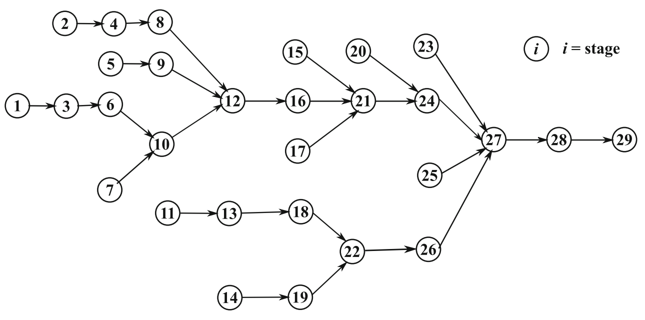

Fig. 3 Real-life application

Table 1 Real-life application data

Stage (i)	Name	Option (j)	Time (t_{ij})	Cost (c_{ij})	Stage (i)	Name	option (j)	time (t_{ij})	cost (c_{ij})
1	84135	1	10	20.25	13	Cutting	1	0	0.39
		2	5	13.34			2	1	0.24
		3	7	18.04	14	99661	1	5	2.17
		4	6	10.23			2	7	2.07
2	99661	1	5	2.17			3	3	2.58
		2	10	0.98	15	285534	1	40	0.77
		3	2	6.45			2	30	2.33
		4	1	8.08			3	35	2.58
3	Cutting	1	0	0.45	16	Painting	1	0	5.96
		2	1	0.30			2	1	3.87
		3	5	0.28	17	285579	1	30	7.92
4	Cutting	1	0	0.55			2	35	8.98
		2	1	0.23			3	25	6.56
5	99662	1	15	4.04	18	Folding	1	0	1.86
		2	10	6.34	19	Cutting	1	0	0.25
		3	20	2.35	20	293483	1	60	4.94
6	Folding	1	0	0.74			2	30	8.45
		2	1	0.80			3	20	10.34
		3	2	0.50	21	Bushing Insertion	1	0	0.66
7	285699	1	100	4.55	22	Welding	1	0	4.05
		2	80	5.55	23	285552	1	80	0.73
		3	60	6.23			2	60	0.24
		4	75	6.23	24	Pedal Pad Insertion	1	0	1.16
8	Welding	1	0	4.46	25	285573	1	50	2.96
		2	1	4.00			2	55	2.50
		3	2	3.45			3	45	2.05
9	Cutting	1	0	0.32	26	Main Bracket	1	0	32.71
		2	1	0.25			2	1	25.54
10	Joining	1	0	2.04			3	2	10.56
		2	1	2.00	27	Pivot Bolt Insertion	1	0	1.76
		3	2	1.86			2	1	1.05
		4	3	0.90			3	2	0.98
11	99675	1	10	23.61	28	Functional Tests	1	0	0.83
		2	8	25.45			2	1	0.45
		3	12	13.24	29	Shipping	1	1	0.75
12	Arm Yokes Union	1	0	4.31			2	3	3.45
		2	1	4.01					
		3	2	3.98					

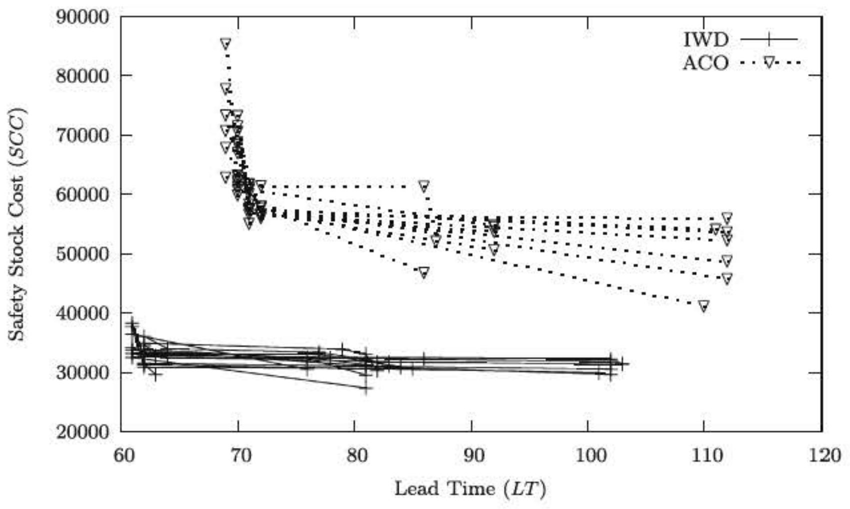

Fig. 4 Non-dominated sets of the 15 runs

bracket ($i = 26$) and the pedal pad ($i = 24$), as well as two other components ($i = 23, 25$), are joined. After the final product is inspected ($i = 28$), it is sent to an overseas customer ($i = 29$).

A worker in a manufacturing cell assemblies a yoke-arm ($i = 12$) by cutting, folding and welding a set of components and sub-assemblies. After the yoke arm is painted ($i = 16$), another worker inserts a clevis bolt in it ($i = 21$) and two other components to finally assemble the pedal pad ($i = 24$). In a separate manufacturing cell, two folding and welding sub-assemblies form a main bucket ($i = 26$).

The experiments were carried out in a Linux-based Lenovo T530 computer with 4 GB RAM memory and an Intel Core i7 (2.90 GHz) processor. Each algorithm is run 15 times and the statistical analysis is presented in the following section.

The parameters to compute the safety stock costs are inventory holding cost of 45 % ($\omega = 0.45$) and the percentage of times that safety stocks cover demand is 0.98, i.e. $z = 2.06$. The demand of this particular SKU is 145 and the standard deviation is 80.

The parameters to run the ACO-based algorithm are set as follows: the number of colonies is 20 each one with 100 ants, the relative importance of pheromones is set to $\alpha = 1$, the value of the relative importance of the heuristic value is $\beta = 1$, the evaporation factor is $\rho = 0.5$. Using those values, the ACO algorithm obtains better results as reported in [8, 20, 21].

The IWD-based algorithm is run using 20 rivers each one with 100 drops. The constant to compute the velocity increments are $a_v = 100$, $b_v = 1$ and $c_v = 1$. The parameters of the local and global updating parameters are $\rho_o = 0.05$ and $\rho_n = 0.05$, see [10].

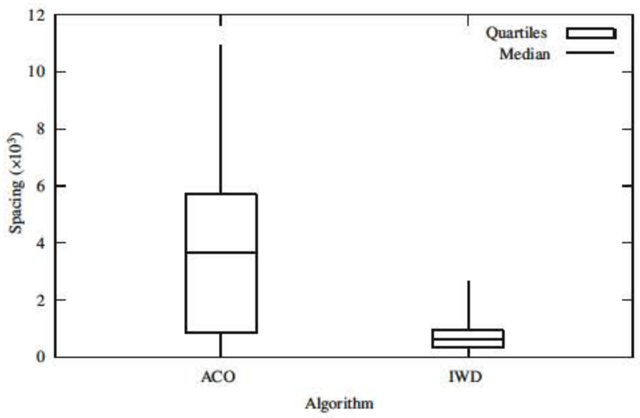

Fig. 5 Spacing metric (S)

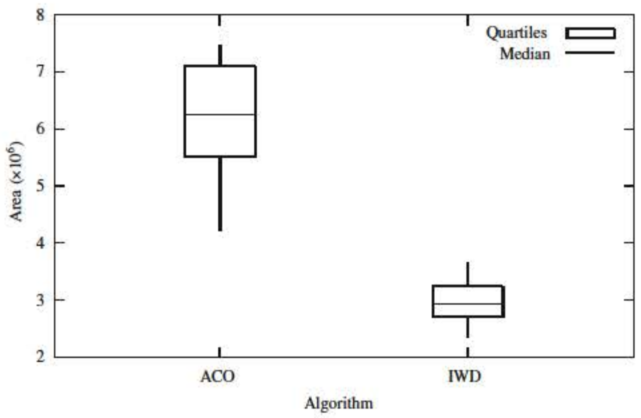

Fig. 6 Hyper-area metric (H)

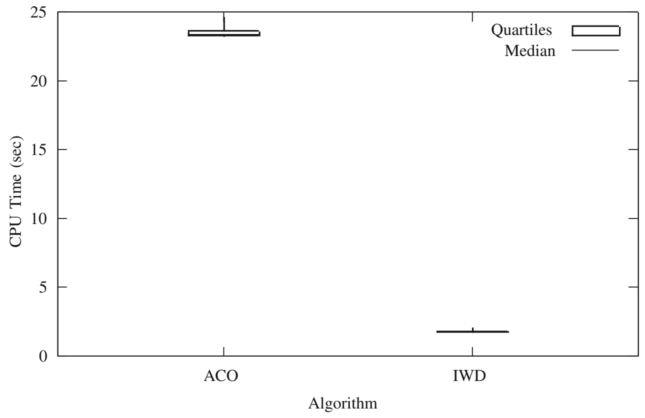

Fig. 7 CPU time

7 Results

We run 15 times the IWD-based algorithm and 15 times the ACO-based algorithm to test the generated non-dominated sets. In Fig. 4, the 30 sets are plotted. According to it, the IWD algorithm generates non-dominated sets with solutions (i.e. SC configurations) that strongly dominates the solutions generated by ACO-based algorithm.

In order to analytically show the performance of the proposed algorithm, four metrics used in multi-objective optimisation are computed. Although there is no a standard set of metrics in multi-objective optimisation, some metrics have been proposed to measure the convergence and diversity of two solutions sets.

The diversity-based indicators measure the distribution (or the spread) of the solutions over the solution space. On the other hand, the convergence-based indicators evaluate the closeness between two solutions sets, usually between the true pareto set and others. The reader is encouraged to see [29] for an extensive explanation on these metrics.

One diversity indicator is the spacing (S), which measures the distance variance of neighbouring solutions of a non-dominated set. The optimum value of Spacing is zero ($S = 0$). It means that all the solutions are spaced evenly apart. As shown in Fig. 6, the fifteen non-dominated sets computed when using the IWD algorithm return sets with a value of space (S) close to zero. On the other hand, the solution sets computed using ACO are not evenly apart as shown in Fig. 4.

A metric that measures both diversity and convergence is the hyper-area (H) indicator. It computes the covered area of a non-dominated set with respect to the objective space.

This metric represents the summation of all the rectangular areas bounded by a given reference point. In our case, it is set in (0,0) in Fig. 4; therefore, the optimum value of the hyper area is zero ($H = 0$) which means that the two objectives take the hypothetical value of zero. As shown in Fig. 6, each non-dominated set generated using IWD-based algorithm returns a value of H lower than the value of H returned when ACO-based algorithm is utilised.

Two important issues in meta-heuristics is the CPU time to solve an instance and the number of non -dominated solutions in the solution set. As shown in Fig. 7, the CPU time at which the IWD-based algorithm solves the instance in Fig. 3 is much less than the time spent by the ACO-based algorithm.

Regarding the number of non-dominated solutions in a set (see Fig. 8), the maximum number of solutions in a set is four when using the ACO algorithm. On the other hand, the maximum number of solutions when IWD is utilised is six. Therefore, it seems that when using IWD algorithm the non-dominated solution sets have more solutions than the ones computed by the ACO algorithm.

In Table 2, a solution set generated by IWD algorithm is shown. According to Table 1, just two stages ($i = 7, 29$) require safety stock to guarantee instant delivery to customers, i.e. $\Omega = \pi_{29} = 0$. Stage $i = 7$ is a supplying stage that represents a component supplied from China; thus, the lead time provided by supplier 1 is 100 days (see Table 1). The stage $i = 29$ is a delivery stage, that requires most of the safety stock. According to the records of the OEM, the investment in safety stock is reduced about 10 %.

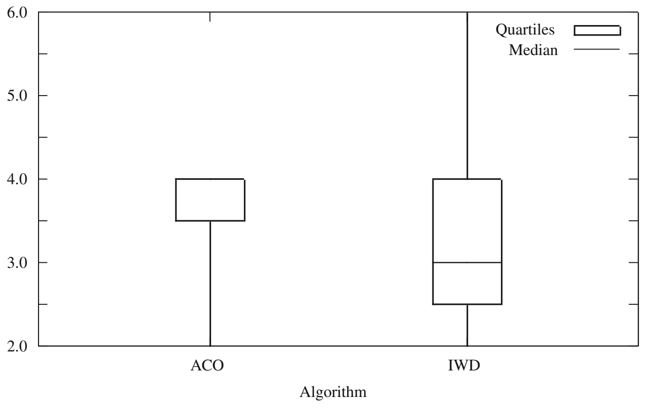

Fig. 8 Number of non-dominates solutions

Table 2 Non-dominated solution set by IWD-based algorithm

Stage	S_1				S_2				S_3				S_4				S_5				S_6			
(i)	j	λ_i	π_i	ι_i^*	j	λ_i	π_i	ι_i^*	j	λ_i	π_i	ι_i^*	j	λ_i	π_i	ι_i^*	j	λ_i	π_i	ι_i^*	j	λ_i	π_i	ι_i^*
1	4	0	6	0	4	0	6	0	3	0	7	0	4	0	6	0	2	0	5	0	4	0	6	0
2	3	0	2	0	3	0	2	0	1	0	5	0	3	0	2	0	3	0	2	0	4	0	1	0
3	2	10	11	0	2	10	11	0	1	5	5	0	1	10	10	0	2	10	11	0	2	10	11	0
4	1	10	10	0	2	2	3	0	1	5	5	0	1	10	10	0	1	2	2	0	1	10	10	0
5	3	0	20	0	3	0	20	0	1	0	15	0	3	0	20	0	1	0	15	0	2	0	10	0
6	3	11	13	0	1	10	10	0	1	10	10	0	3	12	14	0	1	11	11	0	3	10	12	0
7	3	11	13	58	3	10	10	60	4	10	10	75	3	12	14	58	2	11	11	80	2	10	12	78
8	3	10	12	0	1	1	1	0	2	3	4	0	3	1	3	0	1	5	5	0	2	5	6	0
9	1	20	20	0	2	10	11	0	2	10	11	0	1	15	15	0	1	10	10	0	2	10	11	0
10	1	80	80	0	4	75	78	0	1	100	100	0	1	100	100	0	2	60	61	0	3	60	62	0
11	1	0	10	0	1	0	10	0	3	0	12	0	3	0	12	0	3	0	12	0	3	0	12	0
12	1	102	102	0	1	60	60	0	1	61	61	0	2	100	101	0	1	76	76	0	1	81	81	0
13	2	8	9	0	2	10	11	0	1	8	8	0	1	8	8	0	1	8	8	0	2	12	13	0
14	1	0	5	0	1	0	5	0	3	0	3	0	1	0	5	0	3	0	3	0	3	0	3	0
15	1	0	40	0	2	0	30	0	3	0	35	0	1	0	40	0	3	0	35	0	1	0	40	0
16	1	102	102	0	1	79	79	0	2	84	85	0	1	82	82	0	1	63	63	0	1	62	62	0
17	3	0	25	0	3	0	25	0	3	0	25	0	1	0	30	0	3	0	25	0	3	0	25	0
18	1	8	8	0	1	10	10	0	1	8	8	0	1	8	8	0	1	8	8	0	1	10	10	0
19	1	3	3	0	1	5	5	0	1	7	7	0	1	5	5	0	1	5	5	0	1	7	7	0
20	3	0	20	0	3	0	20	0	1	0	60	0	1	0	60	0	1	0	60	0	2	0	30	0
21	1	102	102	0	1	79	79	0	1	65	65	0	1	82	82	0	1	63	63	0	1	62	62	0
22	1	8	8	0	1	10	10	0	1	8	8	0	1	8	8	0	1	8	8	0	1	10	10	0
23	2	0	60	0	2	0	60	0	2	0	60	0	1	0	80	0	1	0	80	0	2	0	60	0
24	1	102	102	0	1	79	79	0	1	65	65	0	1	82	82	0	1	63	63	0	1	62	62	0
25	2	0	55	0	2	0	55	0	1	0	50	0	2	0	55	0	2	0	55	0	2	0	55	0
26	3	8	10	0	1	12	12	0	2	13	14	0	3	10	12	0	3	12	14	0	1	8	8	0
27	2	102	103	0	1	80	80	0	1	65	65	0	1	83	83	0	1	80	80	0	1	101	101	0
28	1	103	103	0	1	82	82	0	1	65	65	0	1	83	83	0	1	80	80	0	1	82	82	0
29	1	103	0	104	1	82	0	83	1	65	0	66	1	83	0	84	1	80	0	81	1	82	0	83
	$SSC = 37'009$				$SSC = 33'937$				$SSC = 33'665$				$SSC = 32'153$				$SSC = 32'018$				$SSC = 32'008$			
	$LT = 62$				$LT = 64$				$LT = 77$				$LT = 81$				$LT = 82$				$LT = 83$			

$*$ days of inventory

8 Conclusions

In this work, a hybrid swarm intelligence algorithm is presented to solve the problem of solving the problem of minimising the cost of placing safety stock under guaranteed-service time and the time to market.

The proposed hybrid algorithm has two parts. In the first one, an option is selected to perform a stage and in the second part, a dynamic programming algorithm is used to set the amount of safety stock based on the selected options. The first part of the algorithm (i.e. selection algorithm) is carried out by one of the following two swarm intelligence algorithms: Intelligent water drop (IWD) or Ant Colony Optimisation (ACO).

So, as to test the proposed algorithm, a real life application is solved. We ran 15 times each algorithm and showed that the IWD-based algorithm returns solutions sets that dominate the ones computed using ACO-based algorithm. Based on two metrics in multi-objective optimisation, the solutions sets generated by IWD-based algorithm generate an hyper-area (H) smaller than the area generated by ACO-based algorithm. Therefore, when IWD is used, the minimum value of every objective is reached given that the reference point is set in (0,0). On the other hand, the solutions in every solution set computed by IWD are better distributed over the solution frontier since the value of the Spacing metric is close to zero. According to the value of these two metrics, it seems that the IWD-based algorithm performs better than the other swarm intelligence algorithm when the bi-objective problem of placing safety stock and time to market is solved.

Acknowledgements Asociación Mexicana Cultura, A.C. As part of the National Research Network "Sistemas de Transporte y Logística", the authors acknowledge all the support provided by the National Council of Science and Technology of Mexico (CONACYT) through the research program "Redes Temáticas de Investigación"

References

1. Amini M, Li H (2011) Supply chain configuration for diffusion of new products: an integrated optimization approach. Omega 39(3):313–322
2. Bakker M, Riezebos J, Teunter RH (2012) Review of inventory systems with deterioration since 2001. Eur J Oper Res 221(2):275–284
3. Blum C, Merkle D (2008) Swarm intelligence: Introduction and applications. Springer
4. Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm intelligence. Oxford
5. Chandra C., Grabis J. (2007) Supply chain configuration. Concepts, solutions and applications. Springer
6. Chopra S, Meindl P (2012) Supply chain management: Strategy, planning and operations, 5edn. Pearson Prentice Hall
7. Crowston WB, Wagner MH (1973) Dynamic lot size models for multi-stage assembly systems. Manag Sci 20(1):14–21
8. Dorigo M, Stützel T (2004) Ant Colony Optimization. MIT Press
9. Dorigo M, Stützle T (2010) Ant colony optimization: Overview and recent advances. In: Gendreau M, Potvin JY (eds) Handbook of Metaheuristics, International Series in Operations Research & Management Science, vol 146. Springer, pp 227–263
10. Duan H, Liu S, Wu J (2009) Novel intelligent water drops optimization approach to single UCAV smooth trajectory planning. Aerosp Sci Technol 13(8):442–449
11. Farahani RZ, Rezapour S, Drezner T, Fallah S (2014) Competitive supply chain network design: an overview of classifications, models, solution techniques and applications. Omega 45(0): 92–118
12. Goh M, Lim JYS, Meng F (2007) A stochastic model for risk management in global supply chain networks. Eur J Oper Res 182(1):164–173
13. Graves S, Willems S (2000) Optimizing strategic safety stock placement in supply chains. Manuf Serv Oper Manag 2(1): 68–83
14. Graves S, Willems S (2005) Optimizing the supply chain configuration for new products. Manag Sci 51(8):1165–1180
15. Huang G, Zhang X, Liang L (2005) Towards integrated optimal configuration of platform products, manufacturing processes, and supply chains. J Oper Manag 23(3-4):267–290
16. Lesnaia E, Vasilescu I, Graves S (2005) The Complexity of Safety Stock Placement in General-Network Supply Chains, Innovation in Manufacturing Systems and Technology (IMST)
17. Li H, Womer K (2008) Modeling the supply chain configuration problem with resource constraints. Int J Proj Manag 26(6): 646–654
18. Moncayo-Martínez L, Resendiz-Flores E, Mercado D, Sanchez-Ramirez C (2014) Placing safety stock in logistic networks under guaranteed-service time inventory models: An application to the automotive industry. J Appl Res Technol 12(3): 538–550
19. Moncayo-Martínez L, Zhang D (2013) Optimising safety stock placement and lead time in an assembly supply chain using bi-objective MAX–MIN ant system. Int J Prod Econ 141(1): 18–28
20. Moncayo-Martínez LA, Recio G (2014) Bi-criterion optimisation for configuring an assembly supply chain using pareto ant colony meta-heuristic. J Manuf Syst 33(1):188–195
21. Moncayo-Martínez LA, Zhang DZ (2011) Multi-objective ant colony optimisation: a meta-heuristic approach to supply chain design. Int J Prod Econ 131(1):407–420
22. Nepal B, Monplaisir L, Famuyiwa O (2011) A multi-objective supply chain configuration model for new products. Int J Prod Res 49(23):7107–7134
23. Osman H, Demirli K (2012) Integrated safety stock optimization for multiple sourced stockpoints facing variable demand and lead time. Int J Prod Econ 135(1):299–307
24. Shah-Hosseini H (2007) Problem solving by intelligent water drops. In: IEEE congress on evolutionary computation, pp 3226–3231
25. Shah-Hosseini H (2008) Intelligent water drops algorithm: a new optimization method for solving the multiple knapsack problem. Int J Intell Comput Cybern 1(2):193–212
26. Shah-Hosseini H (2009) The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. Int J Bio-Inspired Comput 1(1/2):71–79
27. Snyder LV, Shen ZJM (2011) Fundamentals of supply chain theory. Wiley
28. Straub J (2013) Characterization of extended and simplified intelligent water drop (SIWD) approaches and their comparison to the intelligent water drop (IWD) approach. In: Proceedings of the 25th international conference on tools with artificial intelligence

29. Talbi EG (2009) Metaheuristics: From design to implementation, Wiley series on parallel and distributed computing. John Wiley & Sons
30. Wang F, Lai X, Shi N (2011) A multi-objective optimization for green supply chain network design. Decis Support Syst 51(2):262–269
31. Wang J, Shu Y (2007) A possibilistic decision model for new product supply chain design. Eur J Oper Res 177(2):1044–1061
32. Wilhelm W, Liang D, Rao B, Warrier D, Zhu X, Bulusu S (2005) Design of international assembly systems and their supply chains under NAFTA. Transp Res E: Logistics and Transportation Review 41(6): 467–493
33. Xing B, Gao WJ (2014) Intelligent water drops algorithm. In: Innovative computational intelligence: a rough guide to 134 clever algorithms. Springer, pp 365–373