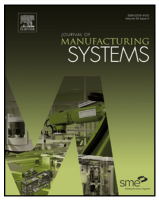
Journal of Manufacturing Systems xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

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



Bi-criterion optimisation for configuring an assembly supply chain using Pareto ant colony meta-heuristic

Luis A. Moncayo-Martínez[a,*], Gustavo Recio[b,1]

[a] Department of Industrial and Operations Engineering, Instituto Tecnológico Autónomo de México (ITAM), Río Hondo No. 1, Col. Progreso Tizapán, C.P. 01080 Mexico City, Mexico
[b] Department of Computer Science, Carlos III University of Madrid, Avda. de la Universidad No 30, Sabatini Building, 28911 Leganés, Madrid, Spain

ARTICLE INFO

Article history:
Received 24 October 2012
Received in revised form
25 September 2013
Accepted 2 December 2013
Available online xxx

Keywords:
Supply chain configuration
Multi-objective optimisation
Pareto set
Ant colony system

ABSTRACT

An assembly supply chain (SC) is composed of stages that provide the components, assemble both sub-assemblies and final products, and deliver products to the customer. The activities carried out in each stage could be performed by one or more options, thus the decision-maker must select the set of options that minimises the cost of goods sold (CoGS) and the lead time (LT), simultaneously. In this paper, an ant colony-based algorithm is proposed to generate a set of SC configurations using the concept of Pareto optimality. The pheromones are updated using an equation that is a function of the CoGS and LT. The algorithm is tested using a notebook SC problem, widely used in literature. The results show that the ratio between the size of the Pareto Front computed by the proposed algorithm and the size of the one computed by exhaustive enumeration is 90%. Other metrics regarding error ratio and generational distance are provided as well as the CPU time to measure the performance of the proposed algorithm.

1. Introduction

In recent years, the problem of supply chain (SC) configuration has attracted the attention of researchers in combinatorial optimisation and practitioners in SC management due to the effect on performance of an optimum design. Typically, optimum design decreases costs by 5–60% with 10% typical and coordinates and improves service time, decreasing it by 25–75% with 40% typical [1]. In addition, the SC configuration provides the basic structure for the SC operations, from the strategic to operational level, and it represents a competitive advantage for companies and a significant area of capital investment.

We model the SC by means of stages, which can be of three different kinds: supplying, manufacturing, and modes of delivery. Every stage is connected to one or more stages according to the products' bill of materials in which the sub- and final assemblies are represented by manufacturer stages, whereas components or raw materials are represented by the supplier stages. The modes of delivery are stages that include information about which customer asks for which product. They are connected to the manufacturers' stages that represent the final assembly. It is assumed that every stage could be performed by at least one option, thus a manufacturer stage could have one or more plants or production lines (options) in which an assembly could be assembled, a supplier stage could have at least one option that represents a supplier able to supply the component, and a delivery stage that represents different modes of delivery, e.g. normal or fast delivery. Every option that could perform a stage is associated with a certain time and cost, e.g. for a manufacturing stage, these correspond to the cost and time of producing either the final-assembly or sub-assembly.

Therefore, the problem of configuring the SC is about selecting an option for every stage given that the cost of goods sold (CoGS) and the products' lead time (LT) are minimised, simultaneously. This is not a trivial decision because the decision-maker could have many options, differentiated by their lead times and costs, that could perform the tasks of supplying a component, assembling a product, and delivering the final product to the customer. In order to select an option to perform a stage, the decision maker must take into account the trade-off between the time and cost added to the CoGS and LT, respectively.

So far, the approaches used to solve the SC configuration problem deal with optimising the CoGS and the most common techniques used to solve it are evolutionary computation and traditional operational research techniques. In this paper, we test a meta-heuristic called ant colony optimisation (ACO) to solve a bi-objective SC configuration problem. ACO has been proved to solve efficiently and effectively many real world and theoretical problems, specially hard combinatorial problems [2]. Moreover, ACO

* Corresponding author. Tel.: +52 55 5490 4618; fax: +52 55 5490 4611.
E-mail addresses: luis.moncayo@itam.mx (L.A. Moncayo-Martínez), grecio@inf.uc3m.es (G. Recio).
[1] Tel.: +34 91 624 8843.

0278-6125/$ – see front matter © 2013 The Society of Manufacturing Engineers. Published by Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jmsy.2013.12.003

can be used in dynamic environments due to its inherent parallelism, i.e. many ants look for a solution at the same time, and positive feedback accounts for rapid discovery of promising solutions.

Since the SC configuration problem is related to the selection of an option to perform a stage (i.e. a combinatorial problem) that minimises both CoGS and LT, we test the performance of ACO to solve the bi-objective problem applying the concept of Pareto optimality criterion to the solutions found by the ants.

We proposed not only applying the Pareto ACO (P-ACO) to the bi-objective SC configuration problem, but also comparing the results with the optimum Pareto set computed by exhaustive enumeration of a notebook SC widely used in literature.

One novelty in our paper is the comparison of the Pareto sets computed by exhaustive enumerations and the one returned by P-ACO to this problem. One of the latest surveys in the SC design problem conducted by Chandra and Grabis [3] shows that the most widely used meta-heuristic applied to SC design problem is the genetic algorithm (GA). Additionally, a survey published by Jones et al. [4] regarding the application of meta-heuristics to multi-objective problems shows that 70% of the articles utilise GA, 24% simulated annealing, and 6% tabu search, thus the application of ACO is relatively new in the SC configuration. A similar survey is published by Giagkiozis et al. [5].

A lot of techniques have been proposed to model and optimise the SC configuration (SCC) problem. We solve this problem using an algorithm based in ACO and modify it by minimising not only the CoGS but also the LT. As the aim of this research is to apply a new approach (Pareto ant colony optimisation) to solve the SCC problem. We identify mainly the mathematical, genetic-algorithm, and computational techniques.

When a mathematical approach is utilised, a mixed-integer programming (MIP) model must be built. The MIP models have two important disadvantages: (a) they provide a relatively simple and compact approximation of complex decision problems and (b) the computational complexity remains an important issue in their application [3]. In order to cope with problem complexity, only some entities of the SC are modelled and one objective is optimised.

When the MIP model has linear or quadratic objective functions or restrictions, it is solved using standard optimisation software as Tsiakis and Papageorgiou [6], Kouvelis et al. [7], Sadjady and Davoudpour [8], and Amin and Zhang [9]. Although the general-purpose optimisation software has high performance and is flexible, the SCC problem does not rely on some kind of linear or quadratic functions that oversimplify key issues such as demand uncertainty or cost and time of SC operations [10]. In order to reduce the computational complexity in mathematical approaches, other techniques such as meta-heuristics and simulation are utilised. The solutions generated by meta-heuristics could not be proved to be optimal but experiences during the past decades have shown that meta-heuristics find the solution or a very "good" solution rapidly and effectively [11].

Another technique used for solving the SC design problem is simulation. Although simulation is a useful tool for modelling the SC, it is not an optimisation technique in itself [12]. Therefore, attempts have been made to combine simulation and optimisation techniques in order to design the SC, see a complete survey by Terzi and Cavalieri [13].

There is no published attempt to solve the bi-objective SC configuration by means of ACO as shown in recent surveys published by Chandra and Grabis [3], van der Vaart and van Donk [14], Tako and Robinson [15], and Kleijnen [16], specialised in SC, and the surveys published by Dorigo and Stützle [2], Mohan and Baskaran [17], and Blum [18] about problems which have been solved by ACO. Some SC topics include order fulfilment by Silva et al. [19]; vehicle routing by Reimann et al. [20], and shop scheduling by Blum and Sampels [21].

2. Ant colony and Pareto optimisation

The ACO is a novel, nature-inspired meta-heuristic that mimics the real foraging behaviour of ant colonies and the concentration/evaporation of chemical substances called pheromones [2].

When ants begin looking for food, they explore the forage area randomly depositing a quantity of pheromones all along their trail. As soon as an ant finds food, it returns to the nest using the path built on its forward trip, reinforcing the pheromones deposited over it. While other ants are looking for food or when ants leave the nest, they tend to choose (in probability) trails that have a strong concentration of pheromones. As a general rule: the higher the quantity of pheromones over a trail, the higher the probability that ants follow it. The quantity of pheromones concentrates over a path because they are a function of time, i.e. if pheromones are not reinforced in short periods of time, they evaporate. It is clear that the shorter the distance from the nest to the food, the faster the ants complete the tour, thus the concentration of pheromones is higher in short trails than in long ones. As pheromones over long trails are not reinforced at the same rate, they evaporate.

As in real ants, the artificial ones (A), also called agents, cooperate to find solutions to hard combinatorial problems by reacting to changes in their environment using an indirect communication process based on artificial pheromones (τ), thus near- and optimum solutions emerge from agent's cooperative interaction. The problem to be solved is represented by a graph in which the pheromones are deposited over the set of either vertices or edges. An initial and a final condition represent the nest and the food, respectively. Ants build a solution by stepping from vertex to vertex based on: (a) a probabilistic decision rule that is a function of the quantity of τ over either vertices or edges and the heuristic information (η) which is a source of information that is not related to ants' parameters and gives A the opportunity of exploiting the specific knowledge of the problem; and (b) the set of problem constraints.

The ACO meta-heuristic has three basic procedures that can take place simultaneously or one at a time. The first procedure is the *construct ant solution* which allows A to build a solution, i.e. A travel around a graph which represents the problem. They know they have to stop travelling when they reach the final condition. The *apply local search* is the second sub-process. This is optional and aims to improve the ant's solution by applying a local search. Finally, in the third procedure called *update pheromones*, the pheromones trails are reinforced according to the solution's "quality" and are evaporated by an evaporation factor (ρ) to avoid stagnation due to an indiscriminate concentration of pheromones.

Multi-objective (MO) optimisation is a growing area of research, thus many techniques have been proposed over the years. Those techniques are broadly divided into preference-based methods and generating methods [22]. In the first ones, a solution is generated based on the preference of the decision maker who can provide this information before or while the solution process is run. On the other hand, generating methods are based on the idea of calculating a set of possible solutions from which the decision maker could select one according to his or her criteria.

In our case, we compute a set of solutions (thus we use the generating method) and then the concept of Pareto optimality is applied to it, thus the decision maker is able to select the "best" solution according to his or her criterion.

Angus [23] proposed an early taxonomy in MO-ACO that is based on the following common features: choice of pheromone model, solution construction process, solution evaluation, pheromone update, and treatment of Pareto optimal solutions.

As our proposed algorithm minimises two objectives, we applied the Pareto optimality criterion to the ant's solutions in order to determine which solutions are better than others, thus a set of non-dominated solutions are computed. A non-dominated solution is a solution that cannot be improved in any of the objectives without impairment in at least one of the other objective. Hence, a solution $\mathbf{s} = \{\mathbf{s}_1, \ldots, \mathbf{s}_k\}$ (in our case a solution $\mathbf{s}$ is a SC configuration) dominates another solution $\mathbf{s}' = \{\mathbf{s}'_1, \ldots, \mathbf{s}'_k\}$, represented by $\mathbf{s} \preceq \mathbf{s}'$, if and only if $\forall \ \ i \in \{1, \ldots, k\}, \mathbf{s} \leq \mathbf{s}' \ \ \wedge \exists \ \ i \in \{1, \ldots, k\} : \mathbf{s}_i < \mathbf{s}'_i$.

As we are minimising two objectives, then $k = 2$ (*LT* and *CoGS*). The solution $\mathbf{s} = (LT, CoGS)$ dominates $\mathbf{s}' = (LT', CoGS')$, if and only if $(LT \leq LT') \wedge (CoGS \leq CoGS')$ and $(LT < LT') \vee (CoGS < CoGS')$. The set of non-dominated solutions which belongs to the set of feasible solutions (Ω) of a problem is called the Pareto Optimal Set $(\mathbb{PS})$. Formally, the $\mathbb{PS}$ for a multi-objective problem $F(\mathbf{s})$ is defined in Eq. (1).

$$\mathbb{PS} := \{\mathbf{s} \in \Omega \mid \neg\exists \ \ \mathbf{s}' \in \Omega \ \ F(\mathbf{s}') \preceq F(\mathbf{s})\} \tag{1}$$

In Pareto ant colony (P-ACO), every colony p builds a $\mathbb{PS}_p$ and every ant q in the colony p generates a solution s_{pq} which must be proved to be non-dominated (Eq. (1)) to $\mathbf{s}_{pq} \in \mathbb{PS}_p$. Therefore, the $\mathbf{s}_{pq} \in \mathbb{PS}_p$ are allowed to modify the pheromones over the graph. In this way, the following ant colonies exploit the "knowledge" of the previous colonies. Hence the $\mathbb{PS}$ reported as a solution to the SC configuration problem is the one computed by the last ant colony.

3. Solution method to solve the SC configuration problem

3.1. Construction graph

The proposed graph that represents the SC configuration problem is divided into tasks or activities. Every node is represented by v_i^r, where r is the node type, i.e. $r = s$ is a supplying node, $r = a$ is an assembling node, $r = d$ is a delivering node, and i is the node number $i = \{1, \ldots, I\}$. Therefore, the set of nodes is $V = \{v_{i'}^s, \ldots, v_i^r, \ldots, v_{i''}^a, \ldots, v_{i'''}^{a'}, \ldots, v_I^d\}$, where a' represents a finished product.

Every v_i^r has $j = \{1, \ldots, J\}$ options that could perform the task r, thus v_{ij}^r represents the option j which can perform the task r. In supplying tasks, v_{ij}^s represents different suppliers that can supply the same component. In assembly tasks, v_{ij}^a stands for the different manufacturing plants or production lines in which a sub- or final assembly could be assembled. Meanwhile, v_{ij}^d represents the different ways of delivering the delivering tasks.

The set of edges represents the relationship among the tasks or activities. Formally, it is $E = \{e(v_i^s, v_i^a), e(v_i^a, v_i^a), e(v_i^a, v_i^{a'}), e(v_i^{a'}, v_i^d)\}$, i.e. there are relationships among supply and assembly activities, between two assembly tasks, between a sub-assembly task and a final assembly task, and between final assembly and delivery tasks.

3.2. Mathematical model of the SC configuration problem

Every v_{ij}^r has a binary variable associated x_{ij}^r which equals to 1 if the option j performs the task v_i^r and equals to 0 otherwise. The cost and time associated to each v_{ij}^r are c_{ij}^r and t_{ij}^r, respectively, and the cost and time of the selected option to perform the task v_i^r are C_i^r and T_i^r.

The problem of SC configuration is formulated as a bi-objective non-linear mixed-integer model.

$$\min \quad CoGS = \quad \xi \sum_r \sum_i \mu_i^r C_i^r \tag{2}$$

$$\min \qquad LT = \quad \max_{v_i^d \in V} \{LT_i^d\} \tag{3}$$

subject to

$$LT_i^r = T_i^r \quad + \max\left\{LT_{i'}^{r'} \mid \forall \ \ v_{i'}^{r'} : e\left(v_{i'}^{r'}, v_i^r\right) \in E\right\} \quad \forall \ \ v_i^r \tag{4}$$

$$\sum_j c_{ij}^r x_{ij}^r - C_i^r = 0 \quad \forall \ \ v_i^r \tag{5}$$

$$\sum_j t_{ij}^r x_{ij}^r - T_i^r = 0 \quad \forall \ \ v_i^r \tag{6}$$

$$\sum_j x_{ij}^r = 1 \quad \forall \ \ v_i^r \tag{7}$$

where ξ is the time period of interest and μ_i^r is the cumulative demand at the task v_i^r.

Eq. (2) is the *CoGS* for all the tasks. Eq. (3) is defined as the *time to market* which is computed by comparing the *LT* of the delivering tasks (v_i^d). In our case, we select the maximum value of the *LT* in v_i^d as an upper bound, so the time in which all products are delivered to the customers is not greater than the upper bound.

Eq. (4) is the time in which a task is finished. A task v_i^r is complete when all its preceding tasks $v_{i'}^{r'} : e\left(v_{i'}^{r'}, v_i^r\right)$ have been finished and then v_i^r performs its own task. Therefore, the LT_i^r for any task is the maximum time in which a task must wait to be served by all its preceding tasks plus the time in which a task is carried out.

Eqs. (5) and (6) are used to compute the cost and time of the selected option to carry out every task. Finally, Eq. (7) guarantees that just one option is selected to perform a task.

3.3. Pareto ant colony elements

In artificial ants, the pheromones must be deposited over the graph which represents the problem. In order to do this, a matrix called pheromones matrix $\mathbb{PM}$ is created. In our problem, the τ represents the desirability of selecting the option v_{ij}^r to perform the task v_i^r, thus the τ is deposited over the different options that can perform a task. Therefore, the proposed $\mathbb{PM}$ is defined as follows:

$$\mathbb{PM} = \left[\tau_{ij}^r\right] = \left[\tau_{ij}^s, \ldots, \tau_{i'j'}^a, \ldots, \tau_{i''j''}^{a'}, \ldots, \tau_{IJ}^d\right] \tag{8}$$

A solution of an ant colony represents the set of selected options to perform all the tasks, i.e. $\mathbf{s}_{pq} = < v_{ij}^s, \ldots, v_{i'j'}^a, \ldots, v_{i''j''}^{a'}, \ldots, v_{ij}^d >$. If the solution generated by the ant q is in the $\mathbb{P}$, then the selected options modify the values of the matrix, i.e. if $v_{ij}^r \in \mathbf{s}_{pq} \mid \mathbf{s}_{pq} \in \mathbb{P}$, then $v_{ij}^r \leftarrow v_{ij}^r + \Delta v_{ij}^r$. At the same time that the values of the selected options are reinforced, a certain amount of τ is evaporated in order to avoid stagnation (when the elements of the $\mathbb{PM}$ are multiplied by the evaporation factor $\rho \in (0, 1)$).

The process of reinforcing and evaporating pheromones over the $\mathbb{PM}$ is known as pheromones update, represented by Eq. (9).

$$\tau_{ij}^r \leftarrow \begin{cases} \tau_{ij}^r + \Delta\tau_{ij}^r + (1-\rho)\,\tau_{ij}^r, & \text{if option } v_{ij}^r \text{ performs task } v_i^r \\ (1-\rho)\,\tau_{ij}^r, & \text{otherwise} \end{cases} \tag{9}$$

In a multi-objective optimisation problem, the $\Delta\tau_{ij}^r$ must be an equation function of the objectives to optimise. Therefore, we propose incrementing the pheromones using Eq. (10).

$$\Delta\tau_{ij}^r = \frac{1}{Q}\left(e^{-\frac{LT}{\omega}} + e^{-\frac{CoGS}{\epsilon}}\right) \tag{10}$$

where Q is the total number of ants in a colony, and ω and ϵ are constants that regulate the value of the objectives.

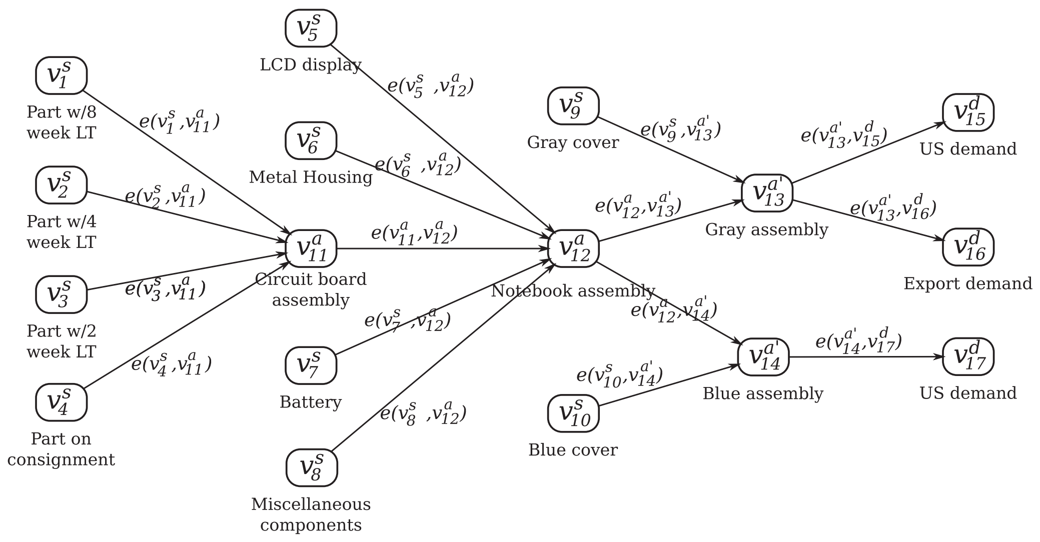


Fig. 1. SC notebook structure.

An important task of every, ant while it is over a node or a task, (v_i^r) is to create its neighbourhood ($\mathbb{N}_{v_i^r}$) which is defined as all the possible options that an ant can select to perform the task. Formally, Eq. (11) represents the node's neighbourhood.

$$\mathbb{N}_{v_i^r} = \{v_{ij}^r \mid v_{ij}^r \in v_i^r\} \tag{11}$$

Once the ant has created the node's neighbourhood, it has to compute the probability of selecting v_{ij}^r to perform v_i^r as in Eq. (12).

$$\mathbb{P}_{v_i^r} = \frac{\left[\tau_{ij}^r\right]^{\alpha}\left[\eta_{ij}^r\right]^{\beta}}{\sum_{\mathbb{N}_{v_i^r}}\left[\tau_{ij'}^r\right]^{\alpha}\left[\eta_{ij'}^r\right]^{\beta}} \tag{12}$$

where η is the heuristic value computed by $\eta_{ij}^r = (\omega/c_{ij}^r) + (\epsilon/t_{ij}^r)$, α is the ant's ability to select an option with high pheromone concentration, and β is the ants ability to select an option with the shortest time and lowest cost.

3.4. Solution algorithm

The proposed based Pareto-ACO algorithm to solve the SC configuration is shown in Algorithm 1. In order to run the algorithm the graph $G = \{V, E\}$ that models the SC has been set as well as the number of colonies ($p = 1, \ldots, P$) and the number of ants in every colony ($q = 1, \ldots, Q$). Basically, the algorithm is divided into the colony's job and the behaviour of every single ant.

In relation to the colony's job, each one generates a Pareto set $\mathbb{PS}_p$ by letting every ant q generate a solution $\mathbf{s}_{pq}$ (lines 5–20). Once every ant has selected one option per task, the Pareto optimality criterion is applied to every $\mathbf{s}_{pq}$ generated by the colony p as outlined in Section 2 (line 21). After that the pheromone matrix is updated by evaporating and reinforcing pheromones (lines 22–25). The options which belong to the Pareto set $\mathbf{s}_{pq} \in \mathbb{PS}_p$ increase the quantity of pheromones according to the "quality" of its solution using the proposed Eq. (10). At the same time, a quantity of pheromones is evaporated using the evaporation factor ρ in order to avoid stagnation and let ants find new solutions or SC configurations. Finally,

Table 1
Data for the SC notebook.

v_i^r		v_{ij}^r			v_i^r		v_{ij}^r		
r	i	j	t_{ij}^r	c_{ij}^r	r	i	j	t_{ij}^r	c_{ij}^r
s	1	1	40	130.00	s	9	1	40	5.0
		2	20	133.25			2	15	5.5
		3	10	134.91	s	10	1	40	5.0
		4	0	136.59			2	15	5.5
s	2	1	20	200.00	a	11	1	20	120.0
		2	10	202.50			2	5	150.0
		3	0	205.03	a	12	1	5	120.0
s	3	1	10	155.00			2	2	132.0
		2	0	156.93	a'	13	1	1	30.0
s	4	1	0	200.00	a'	14	1	1	30.0
s	5	1	60	300.00	d	15	1	5	12.0
		2	5	150.00			2	1	20.0
s	6	1	70	225.00	d	16	1	15	15.0
		2	30	240.00			2	2	30.0
s	7	1	60	40.00	d	17	1	5	12.0
		2	20	45.00			2	1	20.0
s	8	1	30	200.00					

the Pareto set reported as the solution is the one computed by the last ant colony $\mathbb{PS} = \mathbb{PS}_P$ (line 26).

In relation to the ant behaviour (lines 5–20), it has to visit every single node in the graph to select an option to perform it. In order to do this, the algorithm gets the set of nodes and sets an empty solution (lines 6 and 7). While the ant has not finished visiting all the nodes, it selects a node or task v_i^r (line 9) and creates the task neighbourhood $\mathbb{N}_{v_i^r}$ (line 10). Using Eq. (12), the probability of selecting the option v_{ij}^r is computed (line 11), then the *probability decision rule* is applied to every option (line 12). This rule states that the higher the value of $\mathbb{P}_{v_{ij}^r}$, the higher the probability of selecting v_{ij}^r to perform v_i^r. The selected option is stored in the ant solution and the visited node is deleted from the set of nodes (lines 13 and 14). In this way Eq. (7) is solved, thus the cost and time of the selected options is set using Eqs. (5) and (6) (lines 16 and 17) as well as the lead time for every node or task using Eq. (4) (line 18). Finally, the *LT* and the *CoGS* of the solution generated $\mathbf{s}_{pq}$ is calculated. This behaviour is repeated for all the ants in a colony.

Algorithm 1 *(Based Pareto-ACO algorithm to solve the SC configuration problem).*

Algorithm 1: Based Pareto-ACO algorithm to solve the SC configuration problem

```
   Data: $Q$ ant colonies with $P$ ants each one and a graph $G = \{V, E\}$
   Result: Pareto set $\mathbb{PS}$
1  begin
2      set $Q$ and $P$;
3      set the graph $G = \{V, E\}$ which models the SC configuration problem;
4      for $p = 1$ to $p = P$ do
5          for $q = 1$ to $q = Q$ do
6              get the set of nodes $V = \{v_{i'}^s, \ldots, v_i^r, \ldots, v_{i''}^a, \ldots, v_{i'''}^{a'}, \ldots, v_I^d\}$;
7              set an empty solution $\mathbf{s}_{pq} = \langle\ \rangle$;
8              while $V \neq \{\}$ do
9                  select any task $v_i^r$;
10                 set the neighbourhood $\mathbb{N}_{v_i^r}$ (equation 11);
11                 compute $\mathbb{P}_{v_{ij}^r}\ \forall\ v_{ij}^r \in \mathbb{N}_{v_i^r}$ (equation 12);
12                 select a $v_{ij}^r$ (equation 7) based on the *probability decision rule*;
13                 add the selected option to the solution, $\mathbf{s}_{pq} \leftarrow v_{ij}^r$;
14                 delete $v_{ij}^r$ from $V$
15             end
16             set the $C_i^r$ (equation 5) and $T_i^r$ (equation 6);
17             set the $LT_i^r\ \forall\ v_i^r \in \mathbf{s}_{pq}$ (equation 4);
18             compute the $CoGS$ (equation 2);
19             compute the $LT$ (equation 3);
20         end
21         compute the pareto set for the colony $p$, $\mathbb{PS}_p$ (equation 1);
22         forall $v_i^r \in s_{pq}$ do
23             modify pheromone matrix $\mathbb{PM}$ (equation 8);
24             increase and evaporate pheromones (equation 9 and 10);
25         end
26         return the pareto set $\mathbb{PS} = \mathbb{PS}_P$;
27     end
28 end
```

4. Experimental application

4.1. Notebook supply chain

A notebook SC example adapted from Graves and Willems [24] is used as a test case. The related data for solving the problem is shown in Table 1 and a graphical representation of this problem is depicted in Fig. 1.

The notebook SC has ten supplying tasks, $v^s_1, \ldots, v^s_{10}$; four assembling tasks, $v^a_{11}, v^a_{12}, v^{a'}_{13}, v^{a'}_{14}$; and three delivering stages, $v^d_{15}, v^d_{16}, v^d_{17}$. The total number of options is thirty-three (as shown in Table 1), e.g. the supplying stage v^s_1 has four options. Most of the tasks have two options such as the sub-assembling tasks or the delivering tasks, and both final-assembling tasks have one option each, so the experimental application has sixteen edges. According to this, the problem has $\prod_{i=1}^{I} J_i$ possible solutions where J_i is the number of options able to perform the stage v^r_i, thus the notebook SC has 24,576 possible solutions or SC chain configurations.

The pheromone matrix has a size of thirty-three according to the total number of options, thus it contains the quantity of pheromones over each sub-vertex such as:

$$\mathbb{PM} = \left[\tau^s_{1,1}, \ldots, \tau^a_{11,2}, \ldots, \tau^{a'}_{14,1}, \ldots, \tau^d_{17,2}\right]$$

As shown in Table 1, the final assembly tasks represent a grey notebook ($v^{a'}_{13}$) and a blue notebook ($v^{a'}_{14}$). The delivering tasks represent the US market which requires both grey notebooks (v^d_{15}) and blue ones ($v^{a'}_{17}$), and the export market requires grey notebooks (v^d_{16}). The company's time interval is 360 days ($\xi = 360$) and the demand at the delivering stages is: $\mu^d_{15} = 200$, $\mu^d_{16} = 125$, and $\mu^d_{15} = 75$ units per day.

4.2. Results

In order to test the proposed algorithm, we solved the notebook SC described in Section 4.1.

One important aspect in any meta-heuristic, is to tune the parameters of the algorithm in order to find the best results. In ACO, those parameters are the ant's ability to follow pheromones (α), the ant's ability to follow solutions with either low cost or short time (β), and the evaporation factor (ρ) which regulates the concentration and evaporation of pheromones. In [25], it is proposed to set the values of α to 1 and β from 2 to 5, as well as ρ to 0.5 to get promising results in most of the ant systems. According to this, we ran the algorithm setting the different parameters to these proposed values. We set $\alpha > \beta$ and $\alpha < \beta$ using three different levels of pheromone evaporation ($\rho = 0.1, 0.5, 0.9$).

To determine the values of the parameters α and β, we set the value of $\alpha = 1$ as proposed in [25]. The value of β is set to 3 given that we want the heuristic information to dominate the pheromone concentration (see Eq. (12)). $\beta = 3$ according to the value proposed in [25,26]. Other applications of bi-objective optimisation using Ant colony propose to set $\alpha = 1$ and $\beta = 3$ as default values with promises results, see [27]. On the other hand, we tested the case when $\alpha = 3$ and $\beta = 1$, i.e. the pheromone concentration dominates the heuristic information.

To determine the pheromone evaporation parameter ρ, the values 0.1, 0.5, and 0.9 were analysed. Those values are the most used values observed in literature, see [28].

As shown graphically in Fig. 2(a), when $\alpha = 3$, $\beta = 1$, and $\rho = 0.1$, the algorithm returns solutions or SC designs very close to the ones in the true Pareto set. Notice that there are 30 ant colonies, each one with 10,000 ants.

We repeated the experiment by changing the number of colonies and the number of ants per colony. Therefore, we set $P = 10$ and $Q = 1000$. As shown in Fig. 2(b), the Pareto set computed using $\alpha = 3$, $\beta = 1$, and $\rho = 0.1$ is the one with lowest CoGS in most of the solutions, as in Fig. 2(a). This suggests that the algorithm returns Pareto sets that are very close to the one computed by exhaustive enumeration when we set $\alpha = 3$, $\beta = 1$, and $\rho = 0.1$.

Another consideration in our proposed algorithm is the number of colonies and the number of ants in each one. Therefore, we compared the Pareto sets computed when $P = 30$ and $Q = 10{,}000$, $P = 20$ and $Q = 5000$, and $P = 10$ and $Q = 1000$. As shown in Fig 2(c), the greater the number of ant colonies, the closer the solution to the true Pareto set. However, the average CPU time increases from about 6 s when $P = 10$ and $Q = 1000$ to 238 s when $P = 30$ and $Q = 10{,}000$ as shown in Table 2.

Finally, a graphical proof of convergence is provided in Fig 2(d). The Pareto sets generated by colonies 1, 10, 20 and 30 are plotted using the following parameters $\alpha = 3$, $\beta = 1$, $\rho = 0.1$, $P = 30$, and $Q = 10{,}000$ in accordance with the previous results in which the proposed algorithm computes the closest Pareto set to the true one. As shown in Fig. 2(d), the first Pareto set computed by the colony 1 ($p = 1$) has solutions with the highest CoGS but the forthcoming colonies generates solutions with lower CoGS than their preceding ones.

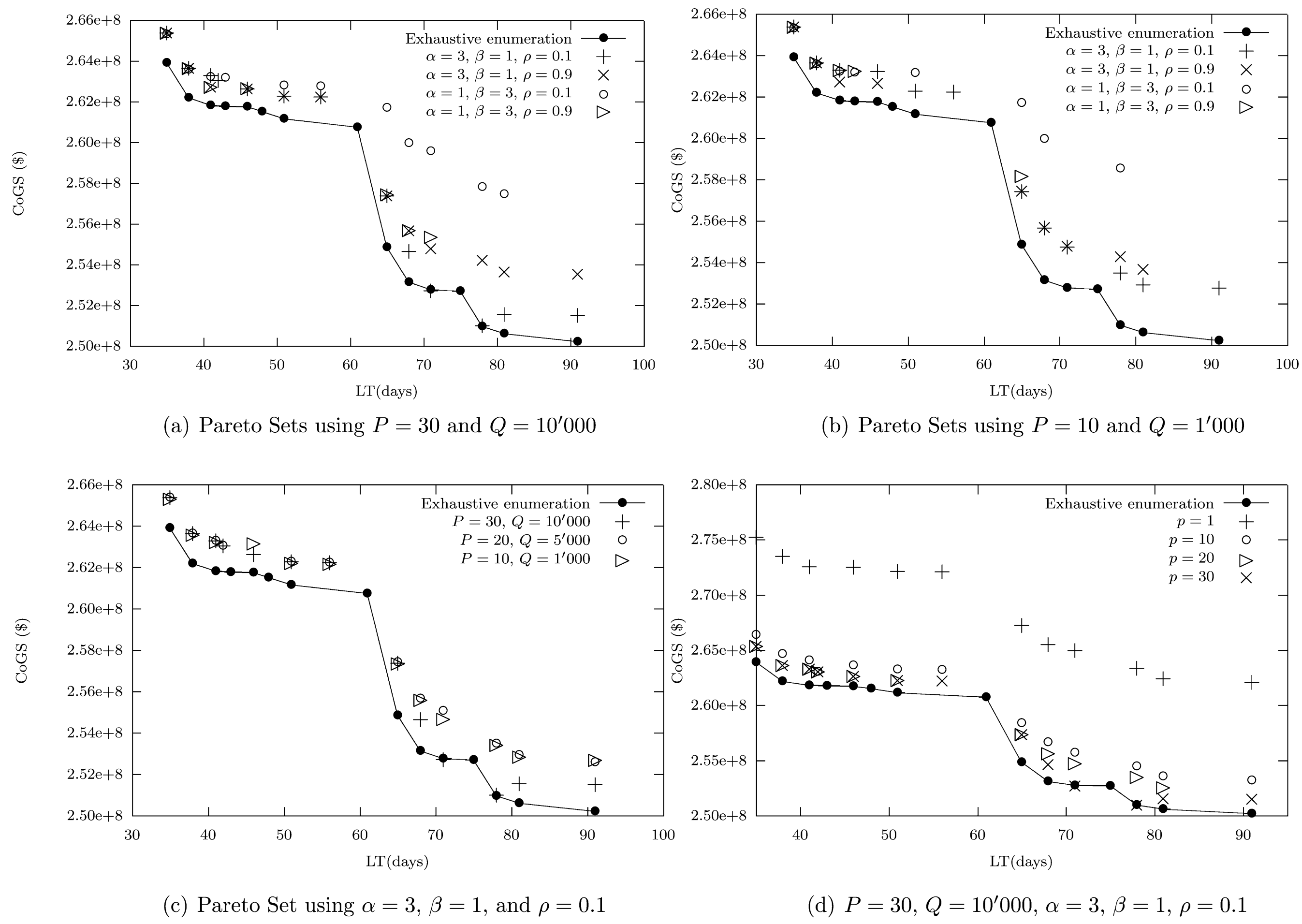
As shown graphically, colony 10 generates a Pareto set with solutions with lower CoGS than colony 1, and colony 20 generates a Pareto set better than the one computed by colony 10 in terms of CoGS and so on. Notice that from colony 1 to 10 the gap between the two Pareto sets is much bigger than the gap between colonies 10 and 20, or 10 and 30.

In order to test the algorithm efficiency analytically, some metrics related to Pareto optimisation are computed by changing the value of α, β, ρ, P, and Q, as shown in Table 2. Basically, the Pareto set computed by our algorithm (represented by $\mathbb{PS}_P$) is compared to the one computed by exhaustive enumeration (represented by $\mathbb{PS}_E$).

The first metric is the error ratio (ER) that measures the number of solutions or SC designs in $\mathbb{PS}_P$ that are not in $\mathbb{PS}_E$. Formally, the ER is computed by $\text{ER} = \sum_{q=1}^{|\mathbb{PS}_P|} e_q / |\mathbb{PS}_P|$, where $e_q = 1$ if $\mathbf{s}_{Pq} \in \mathbb{PS}_E$, $e_q = 0$ otherwise. Therefore, the best result of this metric is ER = 0, which means that $\mathbb{PS}_P = \mathbb{PS}_E$. As shown in Table 2, the smallest error ratio (ER = 0.8) is obtained when $\alpha = 3$, $\beta = 1$, $\rho = 0.1$, $P = 30$, and $Q = 10{,}000$ (run 1), as graphically shown in Fig 2(c), as well. This means that the best Pareto set computed by our algorithm detected 20% of the SC design in $\mathbb{PS}_E$. In the other runs, ER = 1 which indicates that none of the solutions in $\mathbb{PS}_P$ are in $\mathbb{PS}_E$.

An important issue in Pareto optimality is to measure the distance between the true Pareto set and the proposed one. The generational distance (GD) measures, on average, how far every solution $\mathbf{s}_{Pq} \in \mathbb{PS}_P$ is from the nearest solution $\mathbf{s}_n \in \mathbb{PS}_E$. As we are optimising two objectives, the Euclidean distance is computed by $d_{Pq} = \sqrt{(LT_{Pq} - LT_n)^2 + (CoGS_{Pq} - CoGS_n)^2}$, thus the generational distance is computed by $\text{GD} = \left(\sqrt{\sum_{q=1}^{|\mathbb{PS}_P|} d^2_{Pq}}\right) / (|\mathbb{PS}_P|)$. As shown in Table 2, the shortest GD is 360,207 that is reached in run 1. This suggests that the best Pareto set generated in this run is the closest to the $\mathbb{PS}_E$. Moreover, the generational average distance of runs using $P = 30$ and $Q = 10{,}000$ is 787,308, that is shorter than the average distance (948,574) of runs using $P = 10$ and $Q = 1000$. According to this, it seems that the greater the P and the Q, the shorter the GD. Notice that the longest GD is computed when $\alpha = 1$, $\beta = 3$, and $\rho = 0.1$, regardless the value of P and Q, i.e. the algorithm does not return a Pareto set near the $\mathbb{PS}_E$ when ants look for solutions with short time or low cost ($\alpha < \beta$) and 10% of the pheromones are evaporated ($\rho = 0.1$).

Another metric used in measuring the distance between the $\mathbb{PS}_P$ and $\mathbb{PS}_E$ is the maximum Pareto front error (ME). ME and



(a) Pareto Sets using $P = 30$ and $Q = 10'000$ (b) Pareto Sets using $P = 10$ and $Q = 1'000$

(c) Pareto Set using $\alpha = 3$, $\beta = 1$, and $\rho = 0.1$ (d) $P = 30$, $Q = 10'000$, $\alpha = 3$, $\beta = 1$, $\rho = 0.1$

Fig. 2. Results of the experimental application

GD complement each other, since both measure how far the two Pareto sets are from each other. The ME is computed by ME = $\max_q \left\{ d_{Pq} \mid \mathbf{s}_{Pq} \in \mathbb{PS}_P \right\}$. In words, the ME is the maximum Euclidean distance from the solutions in $\mathbf{s}_{Pq}$ to the nearest solution in the true Pareto set. As in the case of the GD, the maximum values of the ME are found in runs 3 and 8 (see Table 2). On the other hand, the minimum ME (2,527,200) is found in run 1, in which the GD is the minimum as well.

Another important metric is the number of non-dominated solutions in the $\mathbb{PS}_P$. This is measured by a metric called overall non-dominated vector generation (ONVG) which is defined as the size of the computed Pareto set, thus ONVG = $|\mathbb{PS}_P|$. This metric, expressed as a ratio of the total number of solutions in the true Pareto set, is known as the ONVG ratio (ONVG-R) which is computed by $ONVG - R = |\mathbb{PS}_P|/|\mathbb{PS}_E|$. In our experimental application the size of the true Pareto set is $|\mathbb{PS}_E| = 15$. As shown in Table 2, when P = 30 and Q = 10,000, the ONVG-R is greater than the runs in which P = 10 and Q = 1000. In words, it seems that the larger the number of ant colonies, the larger the size of the Pareto set found by the algorithm. According to the results in Table 2, the best values of ONVG and ONVG-R are computed in run 1, as in the other metrics.

According to the results of the metrics, our algorithm returns the closest Pareto set to the true one when the parameters are set to: $\alpha = 3$, $\beta = 1$, $\rho = 0.1$, $P = 30$, and Q =10,000, i.e. when those parameters are used: (a) 20% of the solutions in the true Pareto set are found (ER = 0.8), (b) the minimum average distance between the two Pareto sets is about 360,207 (GD) with a maximum Pareto front error (ME) of 2,527,200, and (c) the largest size of the computed Pareto set is 13 (ONVG) that represents 90% (ONVGR-R = 0.9) of the size of the true Pareto set. Notice that the algorithm is highly guided

Table 2
Metric performances of the proposed algorithm.

Run	α	β	ρ	P	Q	CPU time(s)	ER	GD	ONVG	ONVG-R	ME
1	3	1	0.1	30	10,000	260.9	0.80	360,207	13	0.9	2,527,200
2	3	1	0.9			239.3	1.00	619,316	12	0.8	3,319,200
3	1	3	0.1			234.2	1.00	1,442,953	11	0.7	6,919,200
4	1	3	0.9			218.5	1.00	726,756	7	0.5	2,599,200
5	3	1	0.1	20	5000	64.7	1.00	573,512	12	0.8	2,576,700
6	3	1	0.1	10	1000	5.8	1.00	561,909	12	0.8	2,576,700
7	3	1	0.9			5.9	1.00	733,335	9	0.6	3,319,200
8	1	3	0.1			4.6	1.00	1,614,849	8	0.5	7,639,200
9	1	3	0.9			4.6	1.00	884,203	5	0.3	3,319,200

by the amount of pheromones over every option that could perform a stage, since $\alpha > \beta$ and every time an ant colony computed a Pareto set 10% of the pheromones evaporate. On the other hand, when $\alpha < \beta$ and with a low evaporation rate ($\rho = 0.1$), the algorithm has its worst performance (see Table 2 and runs 3 and 8). In relation to the evaporation factor, the algorithm does not improve when the factor is increased, thus when we set $\rho = 0.9$, the algorithm does not improve the metrics (see runs 2, 4, 7 and 9).

5. Conclusions

The use of algorithms based on swarm intelligence for solving large-scale problems is very promising since those algorithms have proven to find high-quality solutions in a short period of time. In this piece of research we address the problem of designing an assembly supply chain given than it is divided into stages and every stage has at least one option that can perform the stage task.

In this paper, we solve the SC configuration problem by an ant colony-based algorithm using P ant colonies with Q ants each one. Every ant computes a solution or SC design to the problem, then the non-dominance criterion is applied to all the solutions and the non-dominated ones are allowed to modify the pheromone matrix.

In order to test our proposed algorithm, we solved a widely used notebook SC configuration problem and computed the true Pareto set by exhaustive enumeration.

We run the algorithm by changing the values of all its parameters. As shown in Fig. 2 and Table 2, the best Pareto set is computed by the algorithm to solve the SC configuration problem when $\alpha = 3$, $\beta = 1$, $\rho = 0.1$, $P = 30$, and $Q = 10{,}000$. As $\alpha > \beta$ and a low evaporation factor is used, the algorithm is ruled by the pheromones which are evaporated and concentrated efficiently. When the evaporation factor is increased to $\rho = 0.9$, the ants' choice about which option to select is taken mainly by the value of either α or β, but even when $\alpha > \beta$, the algorithm has a similar performance when $\rho = 0.1$.

The CPU time to compute the best Pareto set is about 235 s. It contains 20% of the solutions in the true Pareto set, the average distance between the two Pareto sets is about 360,207. The size of the true Pareto sets is 15 and the size of the computed one is 13, and the maximum Pareto front error is 2,527,200. According to these results, we concluded that our proposed algorithm is efficient.

Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful comments.

The completion of this article was supported by Asociación Mexicana de Cultura A.C.

This work was partially supported by the Spanish Ministerio de Ciencia e Innovación, under the project "Gestión de movilidad eficiente y sostenible, MOVES" with grant reference TIN2011-28336.

References

[1] Harrison T. Global supply chain design. Inf Syst Front 2001;3:413–6.
[2] Dorigo M, Stützle T. Ant colony optimization: overview and recent advances. In: Gendreau M, Potvin J, editors. Handbook of meta-heuristics. New York: Springer; 2010. p. 227–63.
[3] Chandra C, Grabis J. Supply chain configuration: concepts, solutions and applications. New York: Springer; 2007.
[4] Jones D, Mirrazavi S, Tamiz M. Multi-objective meta-heuristics: an overview of the current state-of-the-art. Eur J Oper Res 2002;137:1–9.
[5] Giagkiozis I, Purshouse R, Fleming P. An overview of population-based algorithms for multi-objective optimisation. Int J Syst Sci 2013;0:1–28.
[6] Tsiakis P, Papageorgiou L. Optimal production allocation and distribution supply chain networks. Int J Prod Econ 2008;111:468–83.
[7] Kouvelis P, Rosenblatt M, Munson C. A mathematical programming model for global plant location problems: analysis and insights. IIE Trans 2004;36:127–44.
[8] Sadjady H, Davoudpour H. Two-echelon, multi-commodity supply chain network design with mode selection, lead-times and inventory costs. Comput Oper Res 2012;39:1345–54.
[9] Amin SH, Zhang G. An integrated model for closed-loop supply chain configuration and supplier selection: multi-objective approach. Expert Syst Appl 2012;39:6782–91.
[10] Ding H, Benyoucef L, Xie X. A simulation-based multi-objective genetic algorithm approach for networked enterprises optimization. Eng Appl Artif Intel 2006;19:609–23.
[11] Poorzahedy H, Rouhani O. Hybrid meta-heuristic algorithms for solving network design problem. Eur J Oper Res 2007;182:323–34.
[12] Anderson N, Evans G, Biles W. Simulation optimization of logistics systems through the use of variance reduction techniques and criterion models. Eng Optimiz 2006;38:441–60.
[13] Terzi S, Cavalieri S. Simulation in the supply chain context: a survey. Comput Ind 2004;53:3–16.
[14] van der Vaart T, van Donk D. A critical review of survey-based research in supply chain integration. Int J Prod Econ 2008;111:42–55.
[15] Tako A, Robinson S. The application of discrete event simulation and system dynamics in the logistics and supply chain context. Decis Support Syst 2012;52:802–15.
[16] Kleijnen J. Supply chain simulation tools and techniques: a survey. Int J Simulat Process Model 2005;1:82–9.
[17] Mohan B, Baskaran R. A survey: ant colony optimization based recent research and implementation on several engineering domain. Expert Syst Appl 2012;39:4618–27.
[18] Blum C. Ant colony optimization: introduction and recent trends. Phys Life Rev 2005;2:353–73.
[19] Silva C, Sousa J, Runklerand T, Palm R. Soft computing optimization methods applied to logistic processes. Int J Approx Reason 2005;40:280–301.
[20] Reimann M, Doerner K, Hartl R. D-Ants: savings based ants divide and conquer the vehicle routing problem. Comput Oper Res 2004;31:563–91.
[21] Blum C, Sampels M. An ant colony optimization algorithm for shop scheduling problems: an ant colony optimization algorithm for shop scheduling problems. J Math Model Algorithms 2004;3:285–308.
[22] Diwekar U. Introduction to applied optimization. Cambridge: Springer-Verlag; 2008.
[23] Angus D, Woodward C. Multi objective ant colony optimisation. Swarm Intell 2008;3:69–85.
[24] Graves S, Willems S. Optimizing the supply chain configuration for new products. Manage Sci 2005;51:1165–80.
[25] Dorigo M, Stützle T. Ant colony optimization. Massachusetts: MIT Press; 2004.
[26] Neto RFT, Filho MG. An ant colony optimization approach to a permutational flowshop scheduling problem with outsourcing allowed. Comput Oper Res 2011;38:11286–93.
[27] Berrichi A, Yalaoui F, Amodeo L, Mezghiche M. Bi-objective optimization research on integrated fixed time interval preventive maintenance and production for scheduling flexible job-shop problem. Exp Syst Appl 2011;38:7169–78.
[28] Thiruvady D, Blum C, Meyer B, Ernst A. Hybridizing beam-ACO with constraint programming for single machine job scheduling. In: Lecture notes in computer science. 2009. p. 30–44.