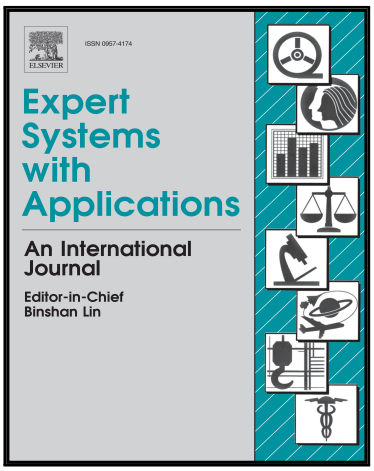
A multi-objective intelligent water drop algorithm to minimise cost Of goods sold and time to market in logistics networks

Moncayo–Martínez, L. A., Mastrocinque, E.

Original citation & hyperlink:
Moncayo–Martínez, LA & Mastrocinque, E 2016, 'A multi-objective intelligent water drop algorithm to minimise cost Of goods sold and time to market in logistics networks' Expert Systems with Applications, vol 64, pp. 455-466. DOI: 10.1016/j.eswa.2016.08.003
https://dx.doi.org/10.1016/j.eswa.2016.08.003

DOI 10.1016/j.eswa.2016.08.003
ISSN 0957-4174

Publisher: Elsevier



A Multi–objective Intelligent Water Drop Algorithm to Minimise Cost Of Goods Sold and Time to Market in Logistics Networks

Luis A. Moncayo–Martínez, Ernesto Mastrocinque

PII: S0957-4174(16)30397-9
DOI: 10.1016/j.eswa.2016.08.003
Reference: ESWA 10795

To appear in: *Expert Systems With Applications*

Received date: 28 December 2015
Revised date: 25 May 2016
Accepted date: 1 August 2016

Please cite this article as: Luis A. Moncayo–Martínez, Ernesto Mastrocinque, A Multi–objective Intelligent Water Drop Algorithm to Minimise Cost Of Goods Sold and Time to Market in Logistics Networks, *Expert Systems With Applications* (2016), doi: 10.1016/j.eswa.2016.08.003

Highlights

- This is the first attempt to solve the SC Design problem using IWD Metaheuristic.
- We modify the single-objective IWD Meta-heuristic to solve a bi-objective SC Design problem.
- We compare our results to the ones computed by Ant Colony Optimisation (ACO).
- We solve several instances to show the performance of our hybrid algorithm
- Our results outperform the ones computed by ACO.

A Multi–objective Intelligent Water Drop Algorithm to Minimise Cost Of Goods Sold and Time to Market in Logistics Networks

Luis A. Moncayo–Martínez[a,*], Ernesto Mastrocinque[b]

[a] *Department of Industrial Engineering and Operations, Instituto Tecnológico Autónomo de México (ITAM), Rio Hondo #1, Col. Progreso Tizapan, C.P. 01080, Mexico City, Mexico*
[b] *School of Management, Royal Holloway, University of London, Egham Hill, TW20 0EX, Egham, UK*

Abstract

The Intelligent Water Drop (IWD) algorithm is inspired by the movement of natural water drops (WD) in a river. A stream can find an optimum path considering the conditions of its surroundings to reach its ultimate goal, which is often a sea. In the process of reaching such destination, the WD and the environment interact with each other as the WD moves through the river bed. Similarly, the supply chain problem can be modelled as a flow of stages that must be completed and optimised to obtain a finished product that is delivered to the end user. Every stage may have one or more options to be satisfied such as suppliers, manufacturing or delivery options. Each option is characterised by its time and cost. Within this context, multi–objective optimisation approaches are particularly well suited to provide optimal solutions. This problem has been classified as NP hard; thus, this paper proposes an approach aiming to solve the logistics network problem using a modified multi–objective extension of the IWD which returns a pareto set.

Artificial WD, flowing through the supply chain, will simultaneously minimise the cost of goods sold and the lead time of every product involved by using the concept of Pareto optimality. The proposed approach has been tested over instances widely used in literature yielding promising results which are supported by the performance measurements taken by comparison to the Ant Colony Meta-heuristic as well as the true fronts obtained by exhaustive enumeration. The pareto set returned by IWD is computed in 4 seconds and the Generational Distance, Spacing, and Hyper–area metrics are very close to those computed by exhaustive enumeration. Therefore, our main contribution is the design of a new algorithm that overcome the algorithm proposed by Moncayo-Martínez and Zhang (2011).

This paper contributes to enhance the current body of knowledge of expert and

*Corresponding author. Tel.: +52(55)5490 4618; fax: +52(55)5490 4611
Email addresses: luis.moncayo@itam.mx (Luis A. Moncayo–Martínez), Ernesto.Mastrocinque@rhul.ac.uk (Ernesto Mastrocinque)

intelligent systems by providing a new, effective and efficient IWD-based optimisation method for the design and configuration of supply chain and logistics networks taking into account multiple objectives simultaneously.

Keywords: Logistics Networks, Bill of Materials, Water Drop Intelligence, Pareto Optimality, Swarm Intelligence, Bi–objective Optimisation

1. Introduction

Increasing competition in today's global market has forced enterprises to configure and evaluate their supply chain (SC) and many logistics providers have recognised that an optimal SC Design (SCD) is a paramount part for any business strategy. When the SC is designed, one of the most important objectives is to deliver products to customers in due time at the lowest possible cost (Simchi-Levi et al., 2008). This is important because an optimal SCD results in cost reduction by 10% and decrements in service time by 40% (Harrison, 2001).

The design process is not easy due to several factors, e.g. market expansion, wide range of suppliers, customers' waiting time, and competitors. Although those factors are important, the Cost of Goods Sold (CoGS) and the Lead Time (LT) (or time to market) have been recognised as the most important objectives to optimise (Aslam and Ng, 2010; Ho et al., 2010).

Traditionally, the SC is modelled as a network in which the nodes represent facilities such as suppliers, manufacturing plants, warehouses, retailers, and customers. The SCD problem has been limited to select the number of facilities and determine the amount of units to flow among them. Moreover, it is assumed that the suppliers, plants, warehouses, and retailers have been selected. This severely reduces the opportunity to optimise the overall SC cost because the selected option may increase both CoGS and LT, see Chandra and Grabis (2007); Shapiro (2007); Goetschalckx (2011) to check a comprehensive list of these models.

On the other hand, the SC can be represented as a Bill of Materials (BOM) in which each node represents a supplying, a manufacturing, or a delivery stage. Each stage could be performed by one or more options, e.g. a component, represented by a supplying stage, could be supplied by one or more suppliers; a sub or final assembly, represented by a manufacturing stage, could be assembled in one or more manufacturing plants or production lines; and a customer, represented by a delivery stage, could be the transportation mode used to deliver the product. Therefore, the problem is to determine: from which supplier should each component be obtained?; where will each product be assembled?; and how should each product be delivered to the customer? The complexity of the problem increases by the fact that the selected options must minimise both the CoGS and the LT for a family of products.

Those objectives are conflicting with each other since a reduction in time increases the cost, e.g. suppose two options which can perform a stage, if the cost of option one is greater than the cost of option two, then the time of the option one is less than the

time of the second option (Cheshmehgaz et al., 2013).

When the SCD problem is modelled as a BOM, the resulting problem is a combinatorial optimisation problem (COP) in which the solution is not based on a sequence but on the selection of variables that "best" perform the objective functions, i.e. the solution of this problem is to select the subset of options (or variables) that minimise the CoGS and LT. This kind of COP has been categorised as NP-hard, thus to find exact solutions in polynomial time is difficult (Garey and Johnson, 1979).

Exhaustive enumerations could be used to find the exact solutions but to compute all the possible combinations is not practical. More efficient methods should be used to find the "best" combination.

Metaheuristics have been widely used to find near–optimal solutions for hard COP in short periods of time (Talbi, 2009). Graves and Willems (2005), Huang et al. (2005), and Wang and Shu (2007) solved the problem minimising only the CoGS using Dynamic Programming, Genetic Algorithm, and Fuzzy Sets, respectively. Moncayo-Martínez and Zhang (2011) minimised CoGS and LT, simultaneously, and Moncayo-Martínez and Zhang (2013) minimised the cost of safety stock using Ant Colony Optimisation (ACO), nevertheless their results are not compared to any other optimisation method to prove the efficiency of the ACO-based algorithm and solved only one instance . Hence, a metaheuristic called Intelligent Water Drop (IWD) that is inspired by the flow of rivers is proposed to solve the CoGS and LT in assembly SC.

This natural behaviour has been applied successfully to a number of theoretical problems such as the travelling salesman problem and multiple-knapsack problem (Shah-Hosseini, 2007, 2008, 2009; Alijla et al., 2014). Industrial applications include job-shop scheduling (Niu et al., 2012), vehicle routing problem (Kamkar et al., 2010; Booyavi et al., 2014), trajectory planning in aerial vehicles (Duan et al., 2009), design of irrigation systems (Hendrawan and Murase, 2011), real-life wast collection problem (Islam and Rahman, 2013), economical load dispatch (Rayapudi, 2011), parallel processor scheduling Mokhtari (2015), and capacitated vehicle routing problem which is solved by a novel IWD and Cuckoo Search Algorithm (Teymourian et al., 2016).

The proposed IWD-based algorithm minimises two objectives and the pareto optimality criterion is used to evaluate them. Computing a pareto set to compare the performance of two algorithms is a standard method in multi-objective optimisation (Coello et al., 2006; Helbig and Engelbrecht, 2013).

This paper contributes in two aspects: a) as nowadays the focus in research is problem-oriented rather than promoting certain algorithm (Blum et al., 2011; Blum and Roli, 2003), an IWD-based algorithm is proposed to solve the bi-objective SCD problem which outperforms the multi-objective metrics reported when ACO is used; and b) the original IWD algorithm is modified to solve a bi-objective problem.

In the last decade researchers have contributed to the body of knowledge of expert and intelligent systems by focusing on developing and applying meta-heuristics and swarm-based algorithms for complex supply chain configuration and logistics problems. In such a context, our paper provides an efficient methodology based on the IWD algorithm for the complex multi-objective optimisation of logistics networks, making

an analogy between the methodology and the particular application.

This paper is organised as follows. Relevant literature is reviewed in section 2. Theory of the IWD is provided in section 3, as well as the problem representation and the proposed solution algorithm. Seven instances are solved in Section 4 and the results are reported in section 5. Finally, conclusions are drawn in section 6.

2. Literature Review

The scope of SCD problem has been limited to minimise both the cost of opening a number of facilities (e.g. suppliers, manufacturing plants, and warehouses) and the cost of transporting the number of components, sub and final assemblies that flow among the facilities (Mula et al., 2010; Melo et al., 2009). One important drawback is that it is assumed that the suppliers, manufacturing plants and warehouses have been selected, i.e. there is only one option to perform the operations of supplying, manufacturing, and delivery.

Some examples of those models are proposed by Amiri (2006), Tsiakis and Papageorgiou (2008), and Ko and Evans (2007). Examples of multi-objective optimization models include: Guillen et al. (2005) maximised the net present value, minimised demand dissatisfaction, and the financial risk; Franca et al. (2010) evaluated the tradeoff between profit and quality; Cardona-Valdes et al. (2011) minimised the total cost and service levels; Yeh and Chuang (2011) optimised the transport cost, lead time, quality and green factor when supplier are selected; and Che and Chiang (2010) considered three evaluation criteria. The above approaches developed a mixed-integer programming (MIP) model and computed a pareto set which includes a set of solutions or SCDs.

Recently, the SC has been modelled as a BOM. Every element of BOM, represents a stage that can be performed by one or more options. Graves and Willems S. (2001); Graves and Willems (2005) modelled the SC as a BOM to minimise the total SC cost which includes the CoGS and the on-hand and in-transit inventory. They developed a dynamic programming (DP) algorithm to solve the single-objective MIP model and solve a widely used notebook SC.

To cope with the disadvantage of high computational effort of DP, Huang et al. (2005) proposed a genetic algorithm (GA) to minimise the total SC cost. The results equal the ones returned by DP when the notebook SC is solved. Li and Womer (2008) added resource constraints, i.e. the possible options which can perform a stage have limited resources (e.g. suppliers have limited capacity). They implemented an approach based on constraint programming (CP).

Wang and Shu (2007) included uncertainty in the options cost and time using fuzzy sets (FS). Their model was solved by a GA. You and Grossmann (2010) focused on safety placement under demand uncertainty in the chemical industry, Lagrangean relaxation and piecewise linear approximation are proposed to solve the problems. Huang Zhang (2005) focused on the effect of quantity discounts and demand variability when a generic BOM is used. Huang et al. (2011) studied the effect of coordinating the price and order quantity when the SC is modelled as a BOM. All these approaches solved

the notebook SC proposed in Graves and Willems (2005)

The described approaches minimised the total SC cost by selecting one option per stage that "best" performs the objective. However, the lead time is as important as the costs of both CoGS and inventory when the SC is designed (see section 1).

Nepal et al. (2011) added a second objective (compatibility firm) to the single-objective model proposed by Graves and Willems (2005). The compatibility firm measures the SC alliance, i.e. the compatibility of the players (selected options) in the SC. Therefore, an option to perform a stage is selected not only based on the cost related objectives but also on its compatibility index which encompasses: structural factors. e.g. cultural alignment, information sharing, and cooperation; managerial factors. e.g. compatibility in strategic goals and conflict management technique; and financial factors, e.g. profit margin and return on assets. Each option selected to perform a stage has a compatibility index which is computed by fuzzy logic. The total compatibility index is computed as the sum of the individuals indexes. The bi-objective model is reduced to a single-objective one by weighted sum method and the model is solved by standard GA.

Although, weighted sum method is straightforward implemented, it has difficulty in finding solutions uniformly distributed over the solution space (Konak et al., 2006). Moreover, the model does not account for time to market metrics, i.e. the proposed solution could minimise both cost and compatibility firm but there could be a solution that minimises the lead time as well.

Moncayo-Martínez and Zhang (2011) minimised both CoGS and LT using ACO but they solved one instance and their results were not compared with any other approach. They tuned the ACO parameters to generate a pareto set with solutions with low CoGS and short LT. Yuce et al. (2014) solved the same bi-objective problem by means of the Bees Algorithm and its modified version, obtaining better performance compared to the ACO. However they solved only one instance.

Moncayo-Martínez et al. (2015) used the IWD algorithm to optimise inventory levels in an assembly supply chain showing promising results compared to the ACO algorithm although the authors tested the IWD algorithm using only one case study. Our approach does not attempt to minimise inventory cost but CoGs Moreover, they use a hybrid approach based on dynamic programming and IWD to solve the bi–objective problems. Thus, medium size instances could be solved. In order to confirm and extend the results, in this paper we propose a multi-objective IWD-based algorithm to optimise a SCD problem when the SC is modelled as a BOM, using seven case studies and comparing the results to those returned by ACO and exhaustive enumeration..

On the other hand, there is no published attempt to solve the bi-objective SCD problem by means of IWD as shown in recent surveys specialised in SC design problem and published by Chandra and Grabis (2007); van der Vaart and van Donk (2008); Mula et al. (2010).

In relation to the multi-objective optimisation using IWD in theoretical or practical problems, Niu et al. (2013) solved the job shop problem minimising the makespan, tardiness, and mean flow time of the schedules. The objective of their research is to find

the best pareto non-dominated set. Moreover, Booyavi et al. (2014) and Teymourian et al. (2016) have applied the IWD algorithm and proposed an enhanced version with the cuckoo search algorithm for solving the vehicle routing problem. However they performed a single-objective optimisation.

3. Intelligent Water Drop and Pareto Criterion

[INSERT NOMENCLATURE HERE]

The IWD metaheuristic is a novel swarm-based algorithm which imitates the natural process between the water drops and the river bed. This metaheuristic was first proposed to solve the Travelling Salesman Problem with promising results since it converges quickly to optimal solutions (Shah-Hosseini, 2007). The basic idea embedded in IWD metaheuristic is that water drops follow an ideal straight line from their origin to their destination (e.g. lake, a sea, or a bigger river) because of the gravitational force. In reality, this is not possible since there are obstacles and barriers that force drops to look for an unblocked path.

In the IWD algorithm: a) every water drop has two properties: *velocity* and *soil* that change during its trip to the destination; b) the environment or river is represented by a set of paths that are full of soil; and c) a water drop is supposed to flow in discrete steps, i.e. the problem is represented by a graph $G = \{V, E\}$ where V is the set of vertices and E is the set of edges, thus the water drop travels from vertex to vertex until it has found the destination.

While a water drop travels from a vertex to another, its velocity is increased by an amount that is non–linear proportional to the inverse of the soil over the edge that links the two vertices. Additionally, an amount of soil from the edge (path), joining the two vertices, is removed and the water drop gathers the removed soil. The amount of soil is non-linearly proportional to the inverse of the time needed for the water drop to pass from a vertex to another. The time is proportional to the velocity of the water drop and inversely proportional to the distance between the two vertices.

A water drop chooses the following vertex to go based on a probability decision rule. This rule states that the probability of a water drop to select a vertex is inversely proportional to the soil of the edge that links two vertices, thus edges with lower soil have higher chances to be selected by the water drop.

Once a water drop has selected the next vertex, the soil over the vertex and the soil from the water drop are updated using an updating parameter which is a small positive number less than one. The water drop stops until it reaches the termination condition.

In this way, a water drop builds a sequence of vertices that returns the total objective value (e.g. cost, distance, etc.). Using the objective value of all the water drops, the iteration-best solution is found and the soil from the edges that form the solution is updated by a global updating parameter which is chosen between $[0, 1]$.

[INSERT ALGORITHM 1 HERE]

Algorithm 1 outlines the IWD Meta–heuristic. Basically, it is divided into two parts: a) solution construction in which every water drop visits all the vertices and its velocity and soil are updated by the local updating parameter, then some amount of soil is removed from the edges as well (lines 6–15); and b) total best solution in which the solutions generated by all the water drops are compared to find the iteration best solution. If it is "better" than the total best solution, then it is replaced by the iteration best solution. Some amount of soil from the edges in the total best solution is updated using the global updating parameter (lines 16–19).

As our proposed algorithm minimises two objectives, CoGS and LT, the concept of Pareto Optimality Criterion is applied to determine which solutions are "better" than others. Those solutions build the solution set and are called non–dominated solutions. A solution of this kind is one in which any improvement in one objective can only take place if at least one of the other objectives worsens. Hence, a solution $\mathbf{s} = \{\mathbf{s}_1, \ldots, \mathbf{s}_k\}$ (in our case $\mathbf{s}_1 = CoGS$ and $\mathbf{s}_2 = LT$) dominates another solution $\mathbf{s}' = \{\mathbf{s}'_1, \ldots, \mathbf{s}'_k\}$, represented by $\mathbf{s} \preceq \mathbf{s}'$, if and only if $\forall\, l \in \{1, \ldots, k\}, \mathbf{s}_l \leq \mathbf{s}'_l \wedge \exists\, l \in \{1, \ldots, k\} : \mathbf{s}_l < \mathbf{s}'_l$. The non-dominated solutions form the solution set represented by $SS := \{\mathbf{s} \in \Omega \mid \neg\exists\, \mathbf{s}' \in \Omega\ \mathbf{s}' \preceq \mathbf{s}\}$, where Ω is the feasible solution space.

3.1. Mathematical representation

In order to mathematically represent the bi-objective optimisation problem, we add a second objective function to the single-objective model proposed by Graves and Willems S. (2001); Graves and Willems (2005). They represented the SC as a graph $G = \{V, E\}$ where the set of vertices represents the supplying, manufacturing, and delivery stages (i), thus $V = \{1, \ldots, i, \ldots, I\}$ and I is the total number of stages. The set of edges represents the relationships between two stages. These relationships could be between: a supplying and a manufacturing stage, two manufacturing stages, or a manufacturing and a delivery stage, thus $E = \{(1, 2), (1, i), \ldots, (i, i')\}$. The subset of delivery stages is defined as $DS \subseteq V$. This is important since the products' demand is generated in those stages.

Every stage i has different options $j = \{1, \ldots, J_i\}$ which can perform the stage. The cost and time of those options are c_{ij} and t_{ij}, respectively. In order to select an option to perform a stage, a binary variable is used. This variable is defined as $y_{ij} = 1$, if the option j performs stage i. Otherwise, $y_{ij} = 0$.

The $CoGS$ is modelled by Eq. 1. Graves and Willems S. (2001); Graves and Willems (2005) defined it as the value of goods sold during the company's interval time of interest ξ.

$$CoGS = \xi \sum_{i=1}^{I} \mu_i C_i \tag{1}$$

$\mu_i = \sum_{i':(i,i')\in E} \mu_{i'}$ is the demand at stage i, and C_i is the cost of the selected option at stage i.

The lead time at stage i (LT_i) is defined as the time of the selected option to perform the stage (T_i) plus the maximum lead time of its preceding stages i'. Formally, the lead time is defined as follows $LT_i = T_i + \max_{i':(i',i)\in E}\{LT_{i'}\}$. The lead time at the delivering stages is known as the time to market, see Eq. 2. We add it to the single objective model.

$$LT = \max_{i\in DS}\{LT_i\} \tag{2}$$

We minimise Eq. 1 and Eq. 2 subject to Eq. 3 to 7. We use the common formulation used in literature when the SC is represented as a BOM.

The SC is designed when Eq. 3, 4 and 6 are solved, i.e. when the values of all the binary variables (y_{ij}) are known as well as when the time (T_i) and cost (C_i) of the stages are set. Eq. 5 computes the lead time for all the stages and Eq. 7 warranties y_{ij} can only take the values of 0 or 1 .

$$\sum_{j=1}^{J_i} c_{ij}y_{ij} - C_i = 0, \quad for\ i = 1, ..., I \tag{3}$$

$$\sum_{j=1}^{J_i} t_{ij}y_{ij} - T_i = 0, \quad for\ i = 1, ..., I \tag{4}$$

$$T_i + \max_{i':(i',i)\in E}\{LT_{i'}\} - LT_i = 0, \quad for\ i = 1, ..., I \tag{5}$$

$$\sum_{j=1}^{J_i} y_{ij} = 1, \quad for\ i = 1, ..., I \tag{6}$$

$$y_{ij} = \{0, 1\}, \quad for\ i = 1, ..., I, \quad j = 1, ..., J_i \tag{7}$$

3.2. Proposed IWD-based algorithm

In order to solve the SCD problem, the IWD algorithm creates R rivers (representing the number of iterations), $r = \{1, \ldots, R\}$, each one with D water drops, $d = \{1, \ldots, D\}$. A water drop solution is a subset S_d of options which perform the stages. $s_d = (LT, CoGS)$ stands for the value of Eq. 1 and 2 generated by subset of options selected by the water drop d. In every iteration (one per river), each river r creates a solution set $SS_r = \{s_1, \ldots, s_d, \ldots\}$ which contains all the non-dominated solutions. It is said that $s_d = (LT, CoGS)$ dominates $s'_d = (LT', CoGS')$, if $(LT \leq LT') \wedge (CoGS \leq CoGS')$ and $(LT < LT') \vee (CoGS < CoGS')$. In order to add s_d to the SS_r, the last condition, called pareto optimality criterion, must be proved for every s_d generated by river r. The final output of the algorithm is the last solution set $SS = SS_R$. Table 1 is an analogy between the elements of the IWD algorithm and those of the SCD problem.

[INSERT TABLE 1 HERE]

In the first part of the proposed algorithm, every water drop creates a s_d. To do so, d is placed in a stage i and then the probability that d selects $j : j \in i$ is computed by Eq. 8, where θ is a very small constant to avoid zero division. The value of p_{ij} depends on the amount of soil of j (ϕ_{ij}), thus the larger the value of ϕ_{ij}, the less the chance j is selected.

$$p_{ij} = \frac{\frac{1}{\theta + g_{ij}}}{\sum_{j':j' \in i} \frac{1}{\theta + g_{ij'}}}; \text{ where } g_{ij} = \begin{cases} \phi_{ij}, & if \ \min_{j \in i} \{\phi_{ij}\} \geq 0 \\ \phi_{ij} - \min_{j \in i} \{\phi_{ij}\}, & \text{otherwise} \end{cases} \tag{8}$$

The value of ϕ_{ij} could be equal or greater than zero. On the other hand, when $\phi_{ij} < 0$, we compute $\phi_{ij} - \min_{j \in i} \{\phi_{ij}\}$ to get all the values of $\phi_{ij} \geqslant 0$. Once every p_{ij} has been computed, the selection of the option j to perform stage i is based on a probabilistic decision rule. This rule is used to allow water drops to look for new paths (or trying new options) and avoid stagnation as well as explore more options. Thus, the option j with the highest probability p_{ij} is not always selected. As general rule, the larger the value of p_{ij} is, the greater the chance to select j to perform i while the options with lower p_{ij} still have possibilities to be selected and explore new solutions.

The value of the binary variable y_{ij} of the selected option j is set to 1 and it is stored in the selected options list $S_d = \{j \ldots\}$. After that, the water drop velocity is updated using Eq. 9; the increments of the option soil are updated by Eq. 10; the soil from the option j (ϕ_{ij}) is updated as well as the water drop soil ϕ_d by Eq. 11 where ρ_n is the local updating parameter.

$$v_d = v_d + \frac{a_v}{b_v + c_v(\phi_{ij})^2} \tag{9}$$

$$\Delta\phi_{ij} = \frac{a_s}{b_s + c_s(\tau_{ij})^2}, \text{ where } \tau_{ij} = \frac{e^{1/t_{ij}} + e^{1/c_{ij}}}{v_d} \tag{10}$$

$$\phi_{ij} = (1 - \rho_n)\phi_{ij} - \rho_n \Delta\phi_{ij}, \text{ and } \phi_d = \phi_d + \Delta\phi_{ij} \tag{11}$$

Notice that the time, a water drop spends in option j, is $\tau_{ij} = \frac{HV}{v_d}$ (Eq. 10), $HV = e^{1/t_{ij}} + e^{1/c_{ij}}$ to weigh the time and cost of the option.

When a water drop d visits all the stages i, the values of Eq. 1 and 2 are computed based on selected options S_d, i.e. $s_d = (LT, CoGS)$ is known.

In the second part of the algorithm, the river r creates a solution set SS_r applying the pareto optimality criterion to all $s_d = (LT, CoGS)$, thus the non-dominated s_d are added to the $SS_r = \{s_d, s_{d'}, \ldots\}$. Finally, the soil of the options j that belong to a non-dominated s_d is updated using a global updating parameter (ρ_w), Eq. 12.

$$\phi_{ij} = (1 - \rho_w)\phi_{ij} - \rho_w \left(\frac{1}{I - 1}\right) \phi_d, \ j \in s_d, \quad s_d \in SS_r \tag{12}$$

Eq. 8–12 are the common equation used by IWD, e.g. Shah-Hosseini (2009), Niu et al. (2012), and Islam and Rahman (2013).

Algorithm 2 is a summary of the proposed IWD algorithm. The first part, in which every water drop selects an option per stage, is carried out by lines 6–25, thus $S_d = \{j, \ldots\}$ is created. So as to create a solution every drop d visits every stage i to select an option j. While the drop d is on a stage i, d computes p_{ij} for every j (lines 10–12). Before going to the next stage the probabilistic decision rule is applied to select j to perform i, i.e. the variable y_{ij} is set to 1; the increments of soil and velocity are computed and updated (lines 13–16); as well as the selected option j is stored in drop's solution S_d (lines 17–18). Once the drop has selected an option to every stage, the cost (C_i) and time (T_i) per stage is computed; then the lead time (LT_i) can be computed as well (lines 20–23). Finally, the drop's solution can be calculated $s_d = LT, CoGS$.

Once, each water drop has computed $s_d = (LT, CoGS)$, the solution set of the river r (SS_r) is built by the non-dominated s_d, this is carried out by lines 26–31. The output of the algorithm is the solution set generated by the last river (line 33).

[INSERT ALGORITHM 2 HERE]

4. Experimental Applications

To test the algorithm, seven instances are solved, shown in Fig. 1. Instances 1, 2, and 4 are taken from Graves and Willems (2005). They represent a notebook SC in which two notebooks are sent to two different markets. The notebooks share the main assembly (called notebook assembly) which is assembled using a circuit board assembly and several components. The only difference between the two computers is the cover colour that could be either gray or blue. The blue notebook, instance 1, is sold at the US market. The gray notebook, instance 2, is sent to the US and Export markets. In instance 4, the two notebooks are represented.

[INSERT FIGURE 1 HERE]

Instance 3 appears in Graves and Willems (2000). It describes a digital capture device (the final assembly) which mainly consists of a charger coupled device, a circuit board assembly, and local accessories. The final assembly is sent to a Central Distribution centre and then supplied to US and export demand.

A bulldozer SC is depicted in instance 5 (Graves and Willems, 2003). The main assembly encompasses: a) a common subassembly in which the transmission, drivetrain and the brake system are assembled; b) a chassis; and c) the dressed-out engine. The final assembly is assembled by the main assembly, the track roller frame, and the suspension group. This instance has no delivery stages and the stages have two options that could perform it.

Instances 6 and 7 represent a tractor SC with three products Wheel Loader (WHL), Track Loader (TRL), and Track-Type Tractor (TTT). The three products share the

main assembly which consists of a transmission, an engine and a chassis. When a shovel and a suspension with wheels are attached to the main assembly, WHL is produced. when a blade, a track roller frame, and the suspension is attached to the main assembly a TTT is produced. The TRL is similar to the TTT but instead of a blade a shovel is attached to the main assembly.

The reader is encouraged to check the references in Fig. 1 to know the values of the time (t_{ij}) and cost (c_{ij}) for all the options in every stage. We provide a summary of the characteristics of the instances in Table 2.

The values of the static parameters are tuned through different combinations and are based on theoretical studies and similar reported studies like Shah-Hosseini (2009, 2008). A multi–objective approach to solve a job–shop problem set the values of $(a_v, b_v, c_v) = (a_s, b_s, c_s) = (1, 0.01, 1)$, $\rho_n = 0.1$, $\rho_w = 0.9$ to obtain optimum solution (Niu et al., 2013).

The variable parameters are user selected and they should be tuned experimentally (Shah-Hosseini, 2009, 2008). In our case, we set to $\phi_{ij} = 10,000$, $\phi_d = 10,000$, and $v_d = 4$ according to the value of the $CoGsS$ and LT in each instance. The heuristic parameter HV is adjust according to the value of time and cost in each stage. Therefore, the values of these parameters are adjust according to the value of CoGS and LT of the instance.

[INSERT TABLE 2 HERE]

The algorithm runs using 10 rivers ($R = 10$) with 450 water drops ($D = 450$) each one. $R = 30$ and $D = 450$ are set to compare the outputs to the results generated by the ACO-based algorithm, proposed by Moncayo-Martínez and Zhang (2011), in which 30 ant colonies are created.

The algorithm was programmed using Java and a Lenovo T530 computer with a Core i7 processor at 2.90GHz and 4GB of RAM memory.

5. Results and Discussions

5.1. Results

Every instance is solved running the algorithm thirty times using $R = 10$ and $D = 450$, then it is run again thirty times setting $R = 30$ and $D = 450$. The average CPU time is computed as well as the average value of five multi-objective optimisation metrics (Coello et al., 2006).

a) *Spacing (S)*. It measures the distribution of solutions throughout the non-dominated solutions, i.e. it measures how well the solutions in the solution set are spread. A value of zero means all the solutions within the solution set are equidistantly spaced.

b) *Generational Difference (GD)*. This is a way to test the distance (i.e. how far) the true pareto set is from the solution set SS. It is clear that the smaller the GD value, the closer the SS is from true Pareto Set.

c) *Hyperarea (H)*. It relates the covered area by the SS to area covered by the true Pareto Set. In our bi-objective problem, H is the summation of all the rectangle areas bounded by the reference point $(LT, CoGS) = (0, 0)$.

d) *Hyperarea Ratio (HR).* It measures the ration of the area of coverage of SS to the true Pareto Set (tPS) objective space, thus the optimum value of HR is one.

[INSERT TABLE 3 and 4 HERE]

The average value of those metrics is shown in Table 3 when the IWD is used and in Table 4 when ACO is run. The objective is to prove that IWD algorithm outperforms the ACO algorithm.

Firstly, the values of the metrics have been compared, when Exhaustive Enumeration (EE) is used to compute the tPS for instances 1, 2, 3, and 4, to these values generated by our proposed algorithm. As shown in Table 3, the largest CPU time is obtained when EE computes the pareto sets. The CPU time when $R = 30$ is larger than the time when $R = 10$. However, the values of the metrics are better using $R = 30$: the value of the GD is smaller and the values of S are closer to those computed by EE. The values of H, when we set $R = 10$ and $R = 30$, are very similar to the values computed by EE. The values of HR are very close to 1 using either $R = 10$ or $R = 30$, it means that the area covered by the solutions sets is similar to the area covered by the tPS.

The same analysis can be carried out when ACO solves the same instance (see Table 4). The CPU time is longer when $P = 30$ than the time when $P = 10$. In relation to the S, the outputs are mixed, the values of S when $P = 10$ are closer to those generated by EE for instances 2 and 3. On the other hand, the values of H are similar to the ones computed by EE for $R = 10$ and $R = 30$.

Secondly, for instances 5, 6, and 7, the values of S and H have been compared when $R = 10$ and $R = 30$ (see Table 3). The values of S are very similar. On the contrary, the values of H are smaller when $R = 30$ than those when $R = 10$, i.e. when setting $R = 30$, the area covered by the solution set is smaller. This is good since our reference point is (0,0) to compute H. The values of those metrics when ACO is used (see Table 4) are mixed. The values of S for $P = 30$ are better for instances 5 and 6 and the values of H are smaller in instances 5 and 7.

Finally, the results of IWD and ACO algorithm have been compared when $R = 30$ and $P = 30$, respectively. For instances 1, 2, 3, and 4, the solutions set generated by our algorithm are closer to the tPS, according to GD (see Table 3 and 4) and Fig. 2(a), 2(b), 2(c), and 2(d).

For instance 5, the values of S and H are smaller using ACO, thus it looks like ACO outperforms IWD. Nevertheless, the number of solutions in the SS using ACO is smaller than the number of solutions using IWD, 9 and 15 respectively, see Fig. 2(e). Moreover, the LT of solutions computed by ACO is between 9 and 12 days, while the LT of solutions in IWD is between 16 and 52, thus the solutions in SS are spread across the solution space. Hence, the IWD generates a SS which covers most of the solution space.

[INSERT FIGURE 2 HERE]

The analysis carried out for instances 6 and 7 is similar to the one performed for instance 5. Notice that for instances 5, 6, and 7 the solutions of SS using ACO algorithm are not spread evenly on the solution space, as shown in Fig. 2(e), 2(f), and 2(g). Therefore, the IWD algorithm generates SS which covers a larger area of the solution space that the sets generated by ACO.

[INSERT FIGURE 3 HERE]

So as to define the variability of our algorithm, box plots are drawn using the results of the thirty runs (Talbi, 2009). As shown in Fig. 3(a), our proposed algorithm solves the seven instances in less time (40% average) than time spend by the ACO algorithm.

The box plots of the GD are plotted in Fig. 3(b). As shown in it, 100% of the 30 results are below the box plot generated using ACO algorithm for the instances 1,2,3, and 4. (notice that for the instances in which the true PS is computed, the GD can be computed). As shown in Fig. 2, the IWD algorithm finds more solutions or SCs than ACO algorithm; thus the decision maker can choose one among more available solutions. The values of spacing are plotted in Fig. 3(c) and the ones of hyperarea are plotted in Fig. 3(d).

Based on the values of the CPU time, GD, and S in Table 3 and 4 for instance 1,2,3, and 4, our IWD-based algorithm solves those instances faster than the ACO approach proposed by Moncayo-Martínez and Zhang (2011). Moreover, the SS returned by our approach are closer to the tPS and the solution are more evenly spaced (see Fig. 3).

In case of instance 5, 6, and 7, our approach returns SS evenly spaced (see Fig. 2); thus, the values of S are bigger than those of ACO (see Table 3 and 4).

According to the descriptive analysis in Fig. 3 and the values of the metrics computed in Table 3 and 4, it seams that the ACO approach by Moncayo-Martínez and Zhang (2011) returns similar results for instance 1, 2, 3, and 4 but for bigger instances (5, 6, and 7) the IWD–based algorithm returns solutions in SS that are evenly spread over the feasible region; thus, the hyperarea is greater. This suggests that our IWD algorithm overcome the ACO–based algorithm.

5.2. Discussion

The original IWD algorithm is successfully customised to solve the bi–objective SCD problems when the $CoGS$ and LT are minimised simultaneously. Our modified algorithm overcome the ACO algorithm proposed by Moncayo-Martínez and Zhang (2011) according to results in Fig. 2 and 3.

Our problem is different from Moncayo-Martínez et al. (2015) and Moncayo-Martínez and Zhang (2013) in that we do not include the safety stock cost which is computed by Dynamic Programming; thus, mid–size instances could be solved. Although they minimised the safety stock and lead time, our approach minimises by IWD–based Bi–objective Algorithm the $CoGS$ and time to market. This work can inspired further application of IWD algorithm to the Logistics Network problems by adding the safety

stock cost, capacity constrains, or sustainability issues such as environmental and social impacts..

On the other hand, we modified the original IWD algorithm to compute a pareto set; thus, our algorithm returns a set of Logistics Networks and the decision maker could select the best that suits s/he.

The proposed methodology has proven to be flexible by providing excellent results on seven different supply chain configuration instances, characterised by different number of stages and options. Moreover, not only the algorithm has proven to give solutions equal or very close to the true Pareto fronts for less complex instances, but it has also shown to be capable of finding many more Pareto solutions covering a larger area of the solution space on the more complex instances, compared to the sets generated by ACO. Therefore the proposed method shows all its potential especially when applied for solving complex multi-objective supply chain and logistics optimisation problems.

6. Conclusions

This paper presents a modified multi-objective optimisation model for minimising the $CoGS$ and LT in a manufacturing supply chain, simultaneously. The model enhances SC efficiency by jointly considering the cost and time during the selection of an option to perform a stage.

We modified the original IWD algorithm to return a pareto set that overcome the pareto sets generated by Moncayo-Martínez and Zhang (2011) when the second objective is to minimise the CoG instead of the the safety stock.

The proposed multi-objective IWD algorithm is applied to seven instances widely used in literature and the results were compared to an ACO-based algorithm as well as exhaustive enumeration when it is possible to compute the true Pareto Set.

According to the results, we conclude that our proposed algorithm outperforms the performance of the ACO algorithm. When the values of S and H are computed for the true Pareto Set, those values are very close to those ones computed by our algorithms. The values of the HR are very close for both algorithms. According to Tables 3 and 4, our algorithm returns more spaced solutions sets, i.e. our algorithm covers a wider area of the solution space.

The solution sets are plotted to show that our algorithm generates solutions with lower cost and shorter lead time than those computed by ACO.

Future search directions may on one hand, compare the proposed approach with other metaheuristic algorithms and, on the other hand, propose enhanced or hybrid versions of the IWD algorithm. Furthermore, the proposed IWD algorithm might be used to solve more complex supply chain design problems taking into account safety stock or sustainability issues.

Acknowledgements

The authors would like to acknowledge the many valuable suggestions made by anonymous reviewers.

The completion of this article was supported by the Asociación Mexicana de Cultura A.C. and the Mexico's National Council of Science and Technology (CONACyT).

References

Alijla, B., Wong, L., Lim, C., Khader, A., Al-Betar, M., 2014. A modified Intelligent Water Drops algorithm and its application to optimization problems. Expert Systems with Applications 41 (15), 6555–6569.

Amiri, A., 2006. Designing a distribution network in a supply chain system: Formulation and efficient solution procedure. European Journal of Operational Research 171 (2), 567–576.

Aslam, T., Ng, A. H. C., 2010. Multi-objective optimization for supply chain management: a literature review and new development. In: 2010 International Conference on Supply Chain Management and Information Systems. IEEE, pp. 1–8.

Blum, C., Puchinger, J., Raidl, G. R., Roli, A., 2011. Hybrid metaheuristics in combinatorial optimization: A survey. Applied Soft Computing 11 (6), 4135–4151.

Blum, C., Roli, A., 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys 35 (3), 268–308.

Booyavi, Z., Teymourian, E., Komaki, G., Sheikh, S., 2014. An improved optimization method based on the intelligent water drops algorithm for the vehicle routing problem. In: 2014 IEEE Symposium on Computational Intelligence in Production and Logistics Systems (CIPLS). pp. 59–66.

Cardona-Valdes, Y., Alvarez, A., Ozdemir, D., 2011. A bi-objective supply chain design problem with uncertainty. Transportation Research Part C: Emerging Technologies 19 (5), 821–832.

Chandra, C., Grabis, J., 2007. Supply chain configuration. Concepts, solutions and applications. Springer.

Che, Z., Chiang, C., 2010. A modified Pareto genetic algorithm for multi-objective build-to-order supply chain planning with product assembly. Advances in Engineering Software 41 (7-8), 1011–1022.

Cheshmehgaz, H. R., Desa, M. I., Wibowo, A., 2013. A flexible three-level logistic network design considering cost and time criteria with a multi-objective evolutionary algorithm. Journal of Intelligent Manufacturing 24 (2), 277–293.

Coello, C. A. C., Lamont, G. B., Veldhuizen, D. A. V., 2006. Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation), 2nd Edition. Springer-Verlag New York, Inc.

Duan, H., Liu, S., Wu, J., 2009. Novel intelligent water drops optimization approach to single UCAV smooth trajectory planning. Aerospace Science and Technology 13 (8), 442–449.

Franca, R. B., Jones, E. C., Richards, C. N., Carlson, J. P., 2010. Multi-objective stochastic supply chain modeling to evaluate tradeoffs between profit and quality. International Journal of Production Economics 127 (2), 292–299.

Garey, M., Johnson, D., 1979. Computers and intractability: A guide to the theory of NP-completeness. W. H. Freeman & Co.

Goetschalckx, M., 2011. Supply Chain Engineering. Springer.

Graves, S., Willems, S., 2000. Optimizing the supply-chain configuration for new products. In: Proceedings of the 2000 MSOM Conference.

Graves, S., Willems, S., 2005. Optimizing the supply chain configuration for new products. Management Science 51 (8), 1165–1180.

Graves, S., Willems S., 2001. Optimizing the Supply Chain Configuration for New Products. MIT working paper, 1–36.

Graves, S. C., Willems, S. P., 2003. Supply chain design: safety stock placement and supply chain configuration. In: de Kok, A. G., Graves, S. C. (Eds.), Handbooks in Operations Research and Management Science. Vol. 11. Elsevier, pp. 95–132.

Guillen, G., Mele, F., Bagajewicz, M., Espuna, A., Puigjaner, L., 2005. Multiobjective supply chain design under uncertainty. Chemical Engineering Science 60 (6), 1535–1553.

Harrison, T., 2001. Global supply chain design. Information Systems Frontiers 3 (4), 413–416.

Helbig, M., Engelbrecht, A. P., 2013. Performance measures for dynamic multi-objective optimisation algorithms. Information Sciences 250, 61–81.

Hendrawan, Y., Murase, H., 2011. Neural-Intelligent Water Drops algorithm to select relevant textural features for developing precision irrigation system using machine vision. Computers and Electronics in Agriculture 77 (2), 214–228.

Ho, W., Xu, X., Dey, P. K., 2010. Multi-criteria decision making approaches for supplier evaluation and selection: A literature review. European Journal of Operational Research 202 (1), 16–24.

Huang, G., Zhang, X., Liang, L., 2005. Towards integrated optimal configuration of platform products, manufacturing processes, and supply chains. Journal of Operations Management 23 (3-4), 267–290.

Huang, Y., Huang, G. Q., Newman, S. T., 2011. Coordinating pricing and inventory decisions in a multi-level supply chain: A game-theoretic approach. Transportation Research Part E: Logistics and Transportation Review 47 (2), 115–129.

Huang Zhang, L., 2005. Optimal supply chain configuration for platform products: impacts of commonality, demand variability and quantity discount. International Journal of Mass Customisation 1 (1), 107–133.

Islam, M., Rahman, M., 2013. An Improved Intelligent Water Drop Algorithm for a Real-Life Waste Collection Problem. In: Tan, Y., Shi, Y., Mo, H. (Eds.), Advances in Swarm Intelligence. Vol. 7929. Springer Berlin Heidelberg, pp. 472–479.

Kamkar, I., Akbarzadeh-T, M. R., Yaghoobi, M., 2010. Intelligent water drops a new optimization algorithm for solving the Vehicle Routing Problem. In: 2010 IEEE International Conference on Systems Man and Cybernetics. pp. 4142–4146.

Ko, H. J., Evans, G. W., 2007. A genetic algorithm-based heuristic for the dynamic integrated forward/reverse logistics network for 3PLs. Computers & Operations Research 34 (2), 346–366.

Konak, A., Coit, D. W., Smith, A. E., 2006. Multi-objective optimization using genetic algorithms: A tutorial. Reliability Engineering & System Safety 91 (9), 992–1007.

Li, H., Womer, K., 2008. Modeling the supply chain configuration problem with resource constraints. International Journal of Project Management 26 (6), 646–654.

Melo, M. T., Nickel, S., Saldanha-da Gama, F., 2009. Facility location and supply chain management - A review. European Journal of Operational Research 196 (2), 401–412.

Mokhtari, H., 2015. A nature inspired intelligent water drops evolutionary algorithm for parallel processor scheduling with rejection. Applied Soft Computing 26, 166–179.

Moncayo-Martínez, L. A., Ramírez-López, A., Recio, G., 2015. Managing inventory levels and time to market in assembly supply chains by swarm intelligence algorithms. The International Journal of Advanced Manufacturing Technology 82 (1-4), 419–433.

Moncayo-Martínez, L. A., Zhang, D. Z., 2011. Multi-objective ant colony optimisation: A meta-heuristic approach to supply chain design. International Journal of Production Economics 131 (1), 407–420.

Moncayo-Martínez, L. A., Zhang, D. Z., 2013. Optimising safety stock placement and lead time in an assembly supply chain using bi-objective MAX–MIN ant system. International Journal of Production Economics 145 (1), 18–28.

Mula, J., Peidro, D., Díaz-Madroñero, M., Vicens, E., aug 2010. Mathematical programming models for supply chain production and transport planning. European Journal of Operational Research 204 (3), 377–390.

Nepal, B., Monplaisir, L., Famuyiwa, O., 2011. A multi-objective supply chain configuration model for new products. International Journal of Production Research 49 (23), 7107–7134.

Niu, S. H., Ong, S. K., Nee, A. Y. C., 2012. An improved Intelligent Water Drops algorithm for achieving optimal job-shop scheduling solutions. International Journal of Production Research 50 (15), 4192–4205.

Niu, S. H., Ong, S. K., Nee, A. Y. C., 2013. An improved intelligent water drops algorithm for solving multi-objective job shop scheduling. Engineering Applications of Artificial Intelligence.

Rayapudi, S. R., 2011. An Intelligent Water Drop Algorithm for Solving Economic Load Dispatch Problem. International Journal of Electrical and Electronics Engineering 5 (1), 43–49.

Shah-Hosseini, H., 2007. Problem solving by intelligent water drops. In: IEEE Congress on Evolutionary Computation. pp. 3226–3231.

Shah-Hosseini, H., 2008. Intelligent water drops algorithm: A new optimization method for solving the multiple knapsack problem. International Journal of Intelligent Computing and Cybernetics 1 (2), 193–212.

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.

Shapiro, J. F., 2007. Modeling the supply chain, 2nd Edition. Duxbury.

Simchi-Levi, D., Kaminsky, P., Simchi-Levi, E., 2008. Designing and managing the supply chain: concepts, strategies, and case studies, 3rd Edition. McGraw Hill Professional.

Talbi, E. G., 2009. Metaheuristics: from design to implementation. Wiley Series on Parallel and Distributed Computing. John Wiley & Sons.

Teymourian, E., Kayvanfar, V., Komaki, G., Zandieh, M., 2016. Enhanced intelligent water drops and cuckoo search algorithms for solving the capacitated vehicle routing problem. Information Sciences 334-335, 354–378.

Tsiakis, P., Papageorgiou, L., 2008. Optimal production allocation and distribution supply chain networks. International Journal of Production Economics 111 (2), 468–483.

van der Vaart, T., van Donk, D., 2008. A critical review of survey-based research in supply chain integration. International Journal of Production Economics 111 (1), 42–55.

Wang, J., Shu, Y., 2007. A possibilistic decision model for new product supply chain design. European Journal of Operational Research 177 (2), 1044–1061.

Yeh, W.-C., Chuang, M.-C., apr 2011. Using multi-objective genetic algorithm for partner selection in green supply chain problems. Expert Systems with Applications 38 (4), 4244–4253.

You, F., Grossmann, I. E., 2010. Integrated multi-echelon supply chain design with inventories under uncertainty: MINLP models, computational strategies. AIChE Journal 56 (2), 419–440.

Yuce, B., Mastrocinque, E., Lambiase, A., Packianather, M. S., Pham, D. T., oct 2014. A multi-objective supply chain optimisation using enhanced Bees Algorithm with adaptive neighbourhood search and site abandonment strategy. Swarm and Evolutionary Computation 18, 71–82.

Nomenclature

i	a stage	r	river
I	total number of stages	d	water drop
j	an option	R	total number of rivers
J_i	total number of options j than can perform stage i	D	total number of water drops
V	set of stages i	S_d	subset of options j selected by d, $S_d = \{j, j', \ldots\}$
E	set of edges representing the stages relationship (i, i')	s_d	a solutions created by d using S_d, $s_d = (LT, CoGS)$
$DS \subseteq V$	subset of delivering stages	SS_r	solution set computed by r, $SS_r = \{s_a, \ldots, s_d \ldots\}$
$CoGS$	cost of goods sold	p_{ij}	probability of choosing j to perform i
LT	lead time or time to market	ϕ_{ij}	amount of soil of $j \in i$
c_{ij}, t_{ij}	cost and time of the option j to perform stage i	ϕ_d	amount of soil of d
C_i, T_i	cost and time of the selected option to perform stage i	v_d	velocity of d
y_{ij}	binary variable equals 1 if j performs i. Otherwise, it equals 0	θ	small constant to avoid zero division
ξ	company's interval time of interest	$\Delta\phi_{ij}$	change of soil in j
μ_i	demand at stage i	ρ_n	local updating factor
a_v, b_v, c_v	velocity updating parameters	ρ_w	global updating factor
a_s, b_s, c_s	soil updating parameters	ρ_w	global updating factor
HV	heuristic value	τ_{ij}	time spent by a water drop to cross option j

Data: a graph $G = \{V, E\}$
Result: total best solution
initialisation of static parameters: # of water drops, total best solution, # of iterations, local and global updating parameter;
initialisation of dynamic parameters: initial soil from edges and water drop as well as their initial velocity, list of visited vertices;
spread water drop randomly on the vertices;
update the list of visited vertices;
while *iteration* $\leqslant$ *# of iterations* **do**
    for *every water drop* **do**
        while *there are unvisited vertices* **do**
            select the next vertex based on the probability decision rule;
            update the list of visited vertices;
            update the water drop velocity;
            compute the amount of soil gathered by the water drop;
            update the soil of the edge which links the selected vertex;
            update the soil of the water drop;
        end
    end
    find the iteration best solution;
    update the soil on the edges that form the iteration best solution;
    update the total best solution by the iteration best solution;
    iteration = *iteration* + 1;
end
The algorithm stops here with the total best solution

Algorithm 1: single-objective IWD metaheuristic

Element	IWD	SCD
(i, j)	an edge that connects vertex i to vertex j.	i is a stage and j is an option which can perform the stage ($j \in i$).
Number of Rivers	$R = 1$	$R > 1$
Water Drop (d)	d visits all vertices and has to decide which vertex j to go while d has just visited vertex i.	d visits all stages and selects the option j to perform stage i.
Soil – ϕ_{ij}	amount of soil of the edge (i, j), thus soil is deposited in edges.	amount of soil of the option j which can perform the stage i.
Soil – ϕ_d	amount of soil of the water drop d.	
Velocity – v_d	velocity of the water drop d.	
Neighbourhood	set of nodes i that have not been visited yet.	set of options j which can perform stage i, ($j \in i$).
Water Drop Solution (S_d)	a sequence of visited node $S_d = \{i, \ldots\}$.	a set of selected options to perform the stages $S_d = \{j, \ldots\}$.
Algorithm Solution (SS)	total best solution built by all the water drops of the river.	a solution set (pareto front) built by every the rivers.
Probability (p_{ij})	probability that d goes to the vertex j if it is in vertex i.	probability of selecting option j to perform stage i.
Probabilistic Decision Rule	The larger the value of p_{ij}, the greater the chances to select j.	
Heuristic Value (HV_{ij})	customised according to the problem.	we proposed $e^{1/t_{ij}} + e^{1/c_{ij}}$.

Table 1: Analogy between elements of IWD and SCD

Data: $V = \{1, \ldots, i, \ldots, I\}$; $j = \{1, \ldots, J_i\}\ \forall\ i$; $E = \{(1, i), \ldots, (i, i') \ldots\}$
Result: Solution set SS

initialise static parameters: D, R, ρ_n, ρ_w, a_v, b_v, c_v, a_s, b_s, c_s;
initialise variable parameters: $\phi_{ij}\ \forall\ i, j$, v_d;
set $r = 1$;
while $r \leqslant R$ **do**
    set $SS_r = \{\}$;
    for $d = 1$ **to** $d = D$ **do**
        set $S_d = \{\}$;
        while $V \neq \{\}$ **do**
            select an stage i from V;
            for $j = 1$ **to** $j = J_i$ **do**
                compute p_{ij} (Eq. 8)
            end
            select j to perform i based on the probabilistic decision rule, i.e. set $y_{ij} = 1$ (Eq. 6 is solved);
            update v_d (Eq. 9);
            compute soil increments $\Delta\phi_{ij}$ (Eq. 10);
            update the soil of ϕ_{ij} and ϕ_d (Eq. 11);
            store the selected option, $S_d \leftarrow j$;
            delete i from V;
        end
        for $i = 1$ **to** $i = I$ **do**
            compute C_i and T_i using S_d (Eq. 3 and 4);
            compute LT_i (Eq. 5);
        end
        set $s_d = (LT, CoGS)$ (Eq. 1 and 2);
    end
    foreach s_d **do**
        verify s_d is dominated by other s'_d, i.e. $s'_d \preceq s_d$;
        if $\neg\exists\ s'_d \mid s'_d \preceq s_d$ **then**
            $SS_r \leftarrow s_d$
        end
    end
    if $r = R$ **then**
        stop $SS = SS_R$;
    else
        update every option $j \mid j \in s_d$ and $s_d \in SS_r$ (Eq. 12);
        $r = r + 1$;
    end
end

Algorithm 2: Proposed algorithm to solve the SCD problem

		stages i (V)			options	edges			demand		
instance	products[♯]	sup	man	del[♭]	(j)	(E)	solutions[§]	ξ	product	customer[†]	μ
1	1	9	3	1	26	12	3,072	250	1	1	125
2	1	9	3	2	28	13	6,144	250	1	1	200
									1	2	75
3	1	8	7	2	32	16	12,288	250	1	1	15
									1	2	4
4	2	10	4	3	33	16	24,576	250	1	1	200
									1	2	75
									2	1	125
5	1	14	8	0	44	21	4,194,304	250	1	0	5
6	3	18	11	9	76	37	2.74×10^{11}	250	1	1	20
									1	3	12
									1	4	23
									2	2	10
									2	4	32
									3	1	21
									3	2	9
									3	3	17
									3	4	6
7	3	18	11	9	105	37	1.28×10^{16}	250	the same as 6		

♯ the products represent the final assembly, i.e. a manufacturing stage
♭ the subset of delivery stages is represented by $D \subset V$
§ solutions $= \prod_i J_i$, where J_i is the number of options which can perform the stage i
† this are the delivering stages, a customer could ask for more than one product

Table 2: Summary of the seven instances

Table 3: Results of the IWD-based algorithm and Exhaustive Enumeration

				IWD-based algorithm									
	Exhaustive Enumeration†			$R = 10$ $D = 450$					$R = 30$ $D = 450$				
Instance	CPU Time (ms)	S	H ($\times 10^6$)	CPU time	$GD^\sharp$	S	H ($\times 10^6$)	$HR^\flat$	CPU time	$GD^\sharp$	S	H ($\times 10^6$)	$HR^\flat$
1	1,105	125,328	4,561	85	11,067	111,484	4,556	0.9987	458	10,298	117,594	4,563	1.0208
2	1,236	247,613	11,277	98	44,117	307,666	11,290	1.0012	733	30,747	266,074	11,290	1.0083
3	4,456	142,298	2,317	138	3,922	142,852	2,318	1.0006	1,532	4,708	142,526	2,318	1.0010
4	15,327	379,380	16,380	143	203,919	671,619	16,464	1.0051	2,522	159,347	391,203	16,443	1.0039
5	–	–	–	296	–	70,400	4,754	–	2,680	–	70,449	13,272	–
6	–	–	–	1,045	–	163,811	6,962	–	4,456	–	163,811	6,962	–
7	–	–	–	2,230	–	185,345	8,404	–	4,516	–	185,345	6,966	–

† The true Pareto Set was able to compute using exhaustive enumeration for instance 1, 2, 3, and 4

♯ The Generational Difference is computed using the true Pareto Set and the solution set SS generated by IWD-based algorithm

♭ This ratio is computed dividing H of the IWD algorithm by H of the true Pareto Set

Table 4: Results of the ACO-based algorithm and Exhaustive Enumeration

				ACO-based algorithm									
	Exhaustive Enumeration†			$P = 10$ $Q = 450$					$P = 30$ $Q = 450$				
Instance	CPU Time(ms)	S	H ($\times 10^6$)	CPU time	$GD^\sharp$	S	H ($\times 10^6$)	$HR^\flat$	CPU time	$GD^\sharp$	S	H ($\times 10^6$)	$HR^\flat$
1	1,105	125,328	4,561	665	61,031	138,909	4,580	1.0041	1,585	61,040	141,825	4,578	1.0041
2	1,236	247,613	11,277	853	121,126	294,564	11,318	1.0036	2,030	112,790	259,131	11,298	1.0019
3	4,456	142,298	2,317	1,982	8,307	150,383	2,198	0.9487	2,400	8,148	148,703	2,261	0.9758
4	15,327	379,380	16,380	1,208	213,554	381,615	16,449	1.0042	3,740	220,943	398,750	16,429	1.0030
5	–	–	–	527	–	6,252	2,536	–	6,205	–	3,126	1,987	–
6	–	–	–	6,886	–	90,674	5,819	–	20,141	–	84,828	5,824	–
7	–	–	–	6,820	–	96,640	5,834	–	21,565	–	110,957	5,778	–

† The true Pareto Set was able to compute using exhaustive enumeration for instance 1, 2, 3, and 4

♯ The Generational Difference is computed using the true Pareto Set and the solution set SS generated by ACO-based algorithm

♭ This ratio is computed dividing H of the ACO algorithm by H of the true Pareto Set

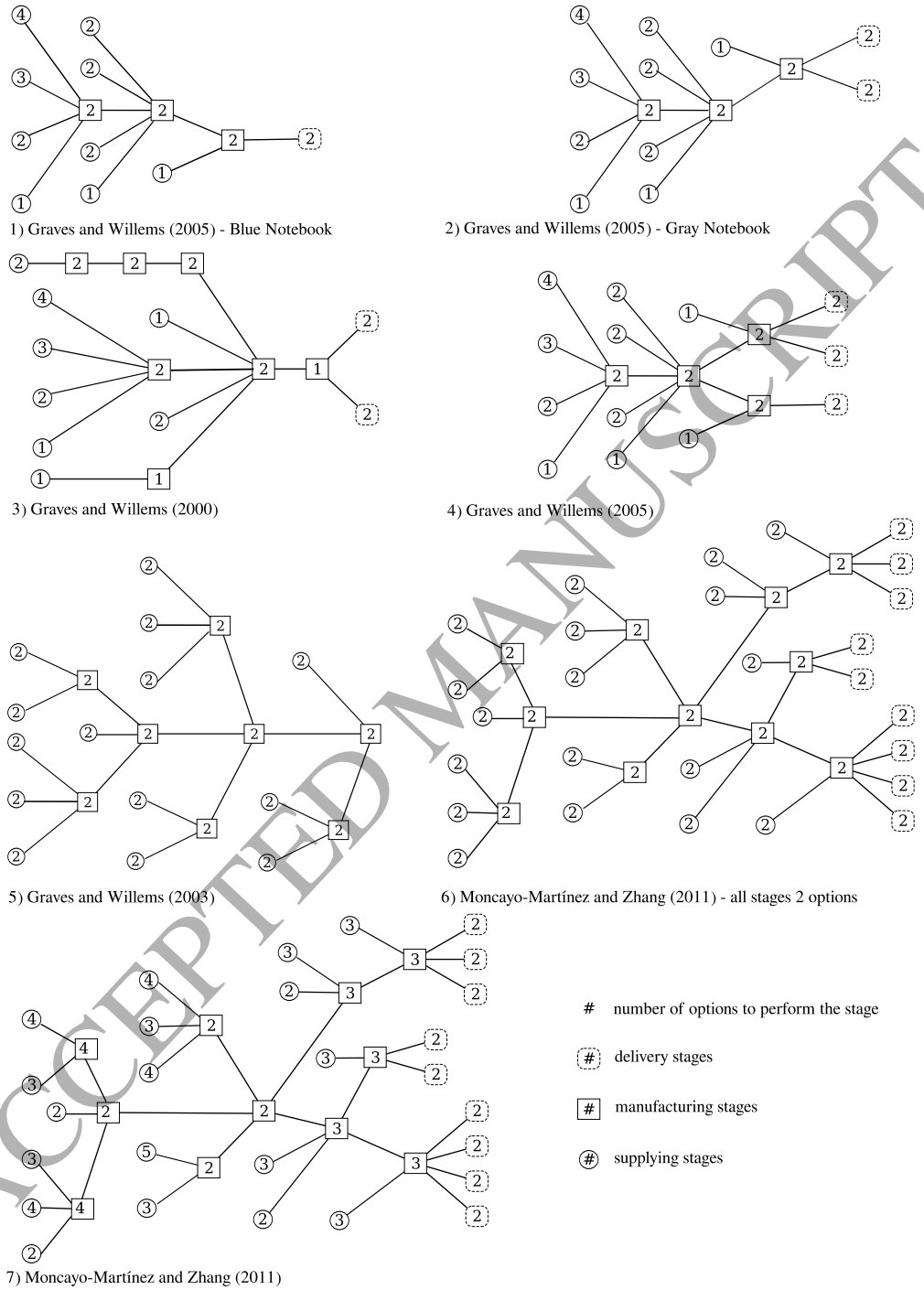
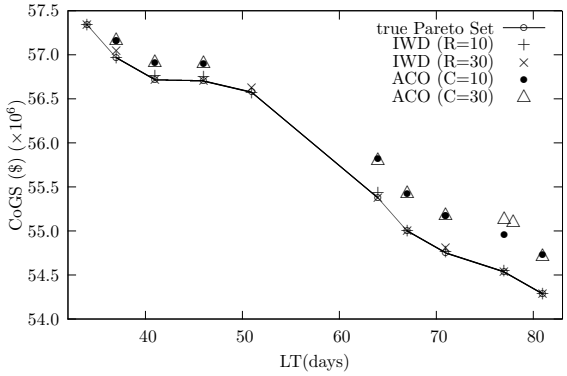
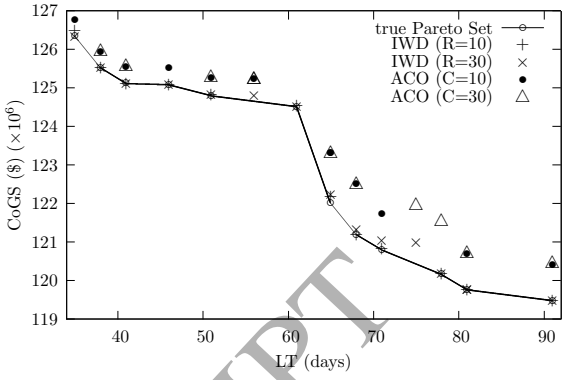


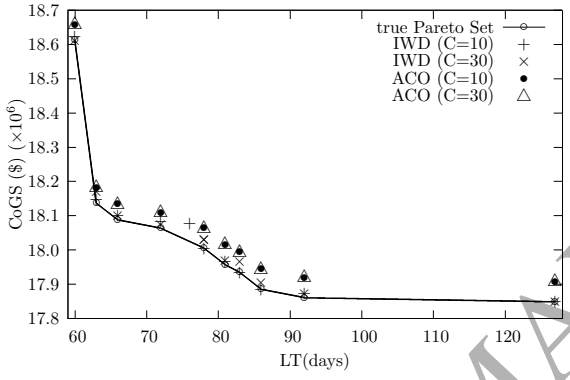
Figure 1: Instances



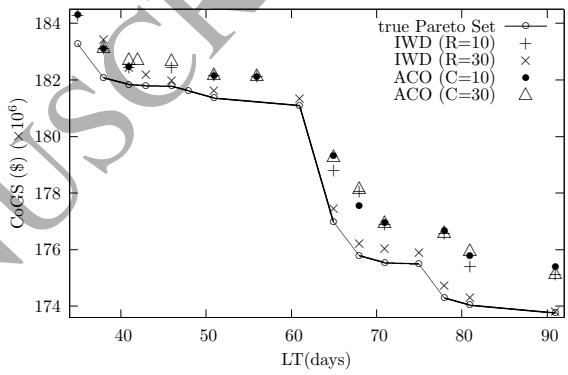
(a) Instance 1



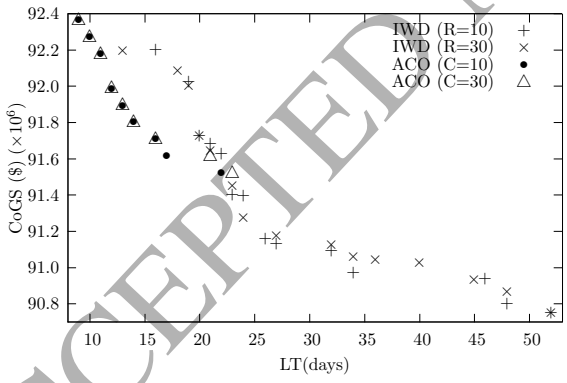
(b) Instance 2



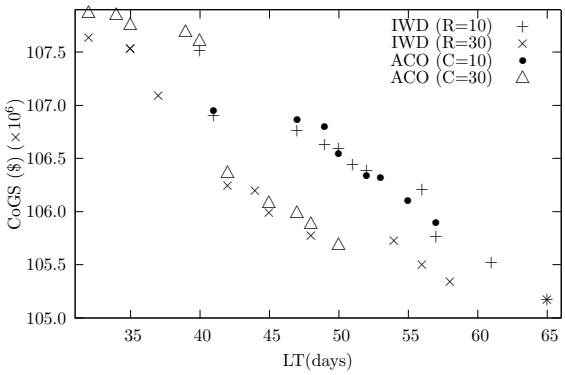
(c) Instance 3



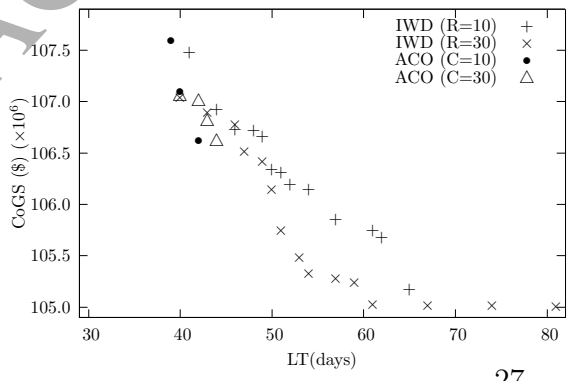
(d) Instance 4



(e) Instance 5

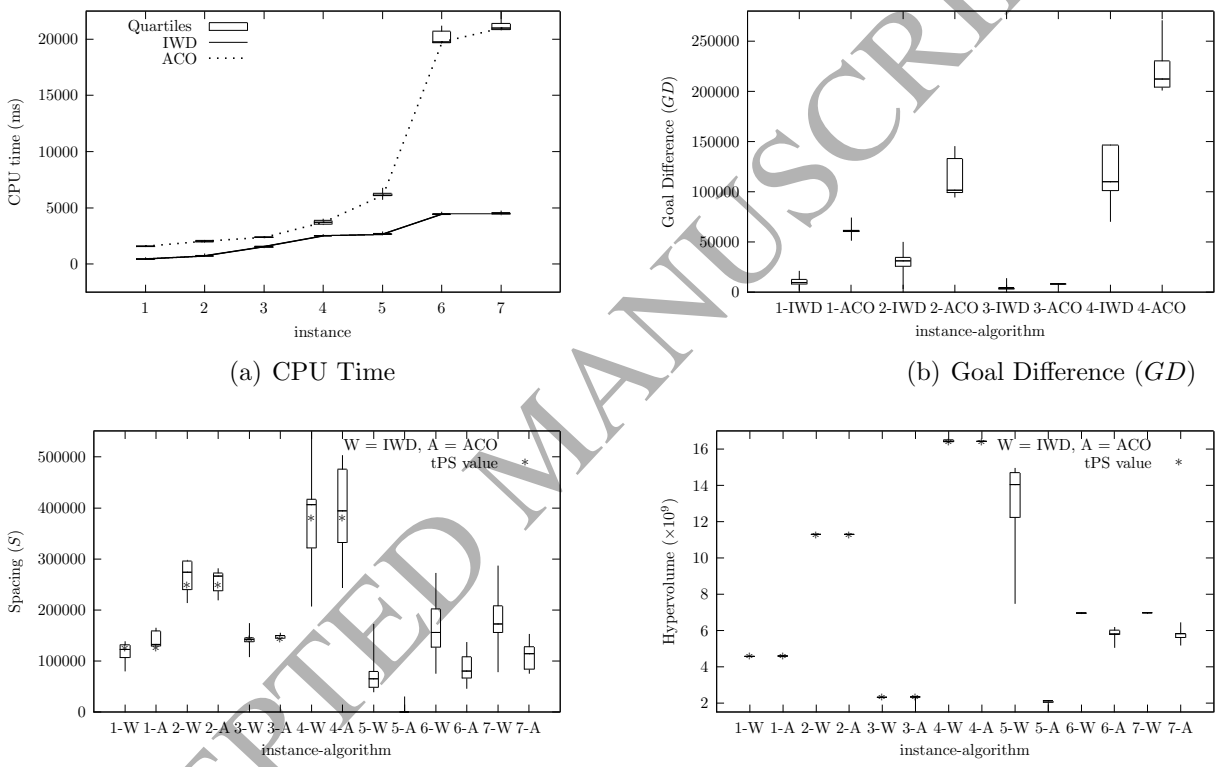


(f) Instance 6



(g) Instance 7

Figure 2: Solution Sets (SS) and true Pareto Sets



(a) CPU Time

(b) Goal Difference (GD)

(c) Spacing (S)

(d) Hypervolume (H)

Figure 3: Box plots of metrics