A Binary Particle Swarm Optimization-based algorithm to Design a Reverse Logistics Network

Ernesto Del R. Santibanez-Gonzalez¹, and Henrique Pacca Luna²

¹Department of Computer Science, Federal University of Ouro Preto, Ouro Preto, MG, Brazil
²Computer Science Institute, Federal University of Alagoas, Maceió, AL, Brazil

Abstract - As recognized by several authors, the design of a reverse logistics network is a complex problem and still relatively unexplored and underdeveloped. We propose a binary particle swarm optimization (BPSO)-based scheme for solving a NP-hard remanufacturing network design problem. The algorithm combines a traditional stochastic search with an optimal solution method for solving to optimality a relaxed LP problem. We divide the swarm into two elementary groups. The first swarm group guides the search for the best location of remanufacturing facilities, while the second group defines the optimal flows between the facilities. We solve to optimality a relaxed LP problem obtained from the original problem and then we project the solution into the swarm space. The algorithm was coded in GAMS and we report computational results for 10 network instances generated randomly with up to 350 sourcing facilities, 100 candidate sites for locating reprocessing facilities and 40 remanufacturing facilities. Computational results regarding gap and computing times are promising.

Keywords: Evolutionary algorithm; particle swarm optimization; sustainable supply chain problem; reverse logistics; integer programming

1 Introduction

The management of product return flows has received increasing attention in the last decade. The efficient stewardship of waste and product return flows is concerned with the final destination of products and their components, and what is their impact on the pollution of the air, water and earth besides the costs of treating the disposal in landfill. There are specific rules in some regions like the European Union, where The European Waste Electrical and Electronic Equipment Directive (WEEE) and the End of Life Vehicles Directive, (ELV) are encouraging companies in the automotive industry to collaborate with other businesses and organizations in the supply chain to ensure that products can be disassembled and reused, remanufactured, recycled or disposed of safely at the end of their life. In addition to stronger legal environmental restrictions, there are several reasons why an increasing number of companies will be interested in becoming engaged in sustainable initiatives such as the management of reverse flows within their supply chain [1,2]. Such as it was noted by some authors [3], huge monetary values ‘can be gained by redesigning the reverse supply chain to be faster and reduce costly time delays’. According to the same authors, design strategies for reverse supply chains are relatively unexplored and underdeveloped. Returned products are remanufactured if this strategy is judged to be cost-effective. Some firms could treat all of the product returns as defective. Some returned products may be new and never used; then these products must be returned to the forward flow. Some products that are not reused or remanufactured are sold for scrap or recycling. Remanufactured products are sold in secondary markets for additional revenue, often to a marketing segment that is unwilling or unable to purchase a new product. In this context, remanufacturing activities are recognized as a main option of recovery in terms of their feasibility and benefits. In this paper is addressed the problem of designing a remanufacturing supply chain network and it is proposed a binary particle swarm optimization-based algorithm for solving it. We study this NP-hard combinatorial optimization problem as part of reverse logistics problems.

In the last decades, evolutionary algorithms (EVO) have been widely used as robust techniques for solving a number of hard combinatorial optimization (CO) problems. An EVO is directed by the evolution of a population in the search for an optimum solution to the CO problems. Particle Swarm Optimization (PSO) is an evolutionary algorithm that has been applied with success in many areas and appears to be a suitable approach for several optimization problems. Since it has been indicated by some authors, in spite of this technique to have been used by success in many continuous problems, in the discrete or binary version there are still some difficulties [4]. In this paper, for test instances of the problem that were generated randomly, we analyze the performance of the proposed algorithm in term of computing times and quality of the solution obtained.

This paper makes two primary contributions. First, we propose a PSO-based scheme, combining an optimization method inside of a PSO algorithm scheme for solving a reverse supply chain problem. Second, a computational study of the proposed method is performed for problem instances of up to 350 sourcing facilities, 100 candidate sites for locating reprocessing facilities and 40 remanufacturing facilities (350x100x40), to provide conclusions regarding the quality of the solution obtained and the computing times. These are the largest instances problems tested so far with evolutionary algorithms.
The rest of the paper is organized as follows: In the second section, we provide a literature review with a brief introduction to sustainable and reverse supply chain and with a special emphasis on the remanufacturing case in connection with reverse logistics. In the third section, we formulate the mathematical model for the problem. In section 4 is proposed the PSO algorithm for solving it. In the fifth section, we present experimental results for large sets of networks generated randomly. The last section contains our conclusions.

2 Literature review

The rise of environmental and sustainable concern implies a number of changes for companies from the strategy level to the operational viewpoint, affecting their people and impacting their business processes and technology. In this regard, network design and facility location models are among the important strategic decisions for companies and organizations facing sustainable issues. Supply Chain Management - SCM, and more specifically Sustainable Supply Chain Management – SSCM provide a good framework for addressing sustainable issues. SSCM involves (a) many organizations, (b) many business processes across and within these organizations, and (c) with social, environmental and economic objectives shared by each organization and the entire Supply Chain. Reverse logistics is part of the SSCM and comprises a series of activities to treat returned products until they are properly recovered or disposed of [1]. These activities include collection, cleaning, disassembly, test and sorting, storage, transport, and recovery operations. Regarding recovery operations, we can find a combination of several main recovery options, like reuse, repair, refurbishing, remanufacturing, cannibalization and recycling [5].

There is another type of important logistics network design problem for the specific case of closed-loop supply chains. Typically, the problem of logistics network design is addressed for ‘forward’ supply chains or reverse supply chains, as in the case that we mentioned previously. When we integrate the forward and reverse supply chains, we obtain a closed system and then there is a new class of problems encompassed by the term closed-loop supply chain management (and design).

In this paper, we focus on the remanufacturing network problem and then on reverse supply chain. Here, remanufacturing is defined as one of the recovery methods by which worn-out products or parts are recovered to produce a unit that is equivalent in quality and performance to the original new product and that can be resold as new products or parts. Because remanufacturing activities are often implemented by the original producer such a network is likely to be a closed-loop system. Remanufacturing activities are recognized as the main option for recovery in terms of feasibility and benefits. It provides firms with a way to master the disposal of their used products, to reduce effectively the costs of production and to save raw materials.

Facility Location Models for Reverse and Closed-loop Supply Chain

The facility location is one of the strategic problems being part of a planning process for managing and designing the supply chain network. The problem of locating facilities and allocating customers is not new to the operations research community and covers the key aspects of supply chain design [6]. This problem is one of ‘the most comprehensive strategic decision problems that need to be optimized for long-term efficient operation of the whole supply chain’ [7]. As it was observed by [8], some small changes to classical facility location models turn these problems quite hard to solve.

In the last few years, mathematical modeling and solution methods for the efficient management of return flows (and/or integrated with forward flows) has been studied in the context of reverse logistics, closed-loop supply chain and sustainable supply chain.

In [9] the research on reverse logistics (and closed-loop supply chain) was classified into three functional areas: Distribution, Inventory and Production and, Supply Chain Scope. Traditional distribution decisions involve the design of network and the location of forward and reverse facilities for the distribution of products and for collecting and reprocessing returned products. In this work we focus on quantitative models for designing closed-loop supply chains, and it concerns the decisions regarding the topological structure of the network, the number of facilities to locate, their locations and capacities and the allocation of product flows between the facilities.

For a review of research on quantitative models for reverse logistics and closed-loop supply chains before 2000, we refer to [9], [10] did a compilation of the research published on reverse logistics within the period 1995-2005. They studied the topic based on the classification proposed by [9].

As argued by numerous authors, traditional approaches for solving closed-loop (reverse) supply chains network design problems usually formulate large-scale mixed-integer linear programs (MILP) that model potential facility location as binary and product flows as positive variables. Typically, these problems belong to the class of NP-hard combinatorial optimization problems. The integration between forward and reverse flows into closed-loop supply chains networks introduce some modifications to the traditional facility location models and also give rise to some additional complexities.

In chronological order, reverse logistics models are discussed recently, for example, by [11-14]. Almost all the authors proposed MILP models. The majority of solution methods are based on standard commercial packages.

Closed-loop supply chain models are taking into account in [15-19]. Stochastic models in combination with multiobjective function were presented by [20].

Regarding evolutionary algorithms for solving related problems have been studied by: Simulated Annealing [21]; Genetic Algorithm [18], [22], [23]; Memetic Algorithm [24] and Tabu Search [21]. Notice that in [23] were solved problems of up to 15 returning centers, 10 disassembly centers.
and 14 processing centers. They did not report computational results regarding time or the quality of the gap obtained. In [21] were solved problems with network of up to 100x40x30 with a maximum computing time (s) of 2.939 and a maximum gap (compared to a lower bound) of 15%.

2.2 Discrete Particle Swarm Optimization Algorithm

PSO is a metaheuristics based on the social behavior and communication of birds’ flock and shoal of fishes [25]. PSO can be considered as an evolutionary algorithm because its way of exploration via neighborhood of solutions (particles) across a population (swarm) and exploiting the generational information gained. But, it has some divergences from other evolutionary algorithms in such a way that it has no evolutionary operators such as crossover and mutation of genetic methods. PSO has the advantage that is easy of use with fewer parameters to adjust. In PSO, the potential solutions (particles), move around in a multidimensional search space with a velocity, which is constantly updated by a combination of the particle’s own experience, the experience of the particle’s neighbors and the experience of the entire swarm. PSO has been successfully applied to a wide range of applications [26]. Since PSO is developed for continuous optimization problem initially, most existing PSO applications are resorting to continuous function value optimization [26, 27]. Recently, a few researches applied PSO for solving discrete combinatorial optimization problems (for example: [28, 29]).

3 Mathematical Model for designing a remanufacturing supply chain network (RSCP)

In this section we present the MILP model for the problem of designing a sustainable supply chain network. This problem can be categorized as a single product, static, three-echelon, capacitated location model with known demand. The remanufacturing supply chain network consists of three types of members: sourcing facilities (origination sites like a retail store), collection sites and remanufacturing facilities. At the customer levels, there are product demands and used products ready to be recovered, for example cell phones. We suppose that customers return products to origination sites like a retail store. At the second layer of the supply chain network, there are reprocessing centers (collection sites) used only in the reverse channel and they are responsible for activities, such as cleaning, disassembly, checking and sorting, before the returned products are sent back to remanufacturing facilities. At the third layer, remanufacturing facilities accept the checked returns from intermediate facilities and they are responsible for the process of remanufacturing. In this paper we address the backward flow of returns coming from sourcing facilities and going to remanufacturing facilities through reprocessing facilities properly located at pre-defined sites. In such a supply chain network, the reverse flow, from customers through collection sites to remanufacturing facilities is formed by used products, while the other (“forward” flow) from remanufacturing facilities directly to point of sales consists of “new” products.

3.1 RSCP Model

In our model is assumed that the product demands (new ones) and available quantities of used products at the customers are known and deterministic. We introduce the following inputs and sets:

- \( I \) = the set of sourcing facilities at the first layer, indexed by \( i \)
- \( J \) = the set of remanufacturing nodes at the third layer indexed by \( j \)
- \( K \) = the set of candidate reprocessing facility locations at the mid layer, indexed by \( k \)
- \( a_i \) = supply quantity at source location \( i \in I \)
- \( b_j \) = demand quantity at remanufacturing location \( j \in J \)
- \( f_j \) = fixed cost of locating a mid layer reprocessing facility at candidate site \( k \in K \)
- \( g_k \) = management cost at a mid layer reprocessing facility at candidate site \( k \in K \)
- \( c_{ij} \) = is the unit cost of delivering products at \( k \in K \) from a source facility located in \( i \in I \)
- \( d_{kj} \) = is the unit cost of supplying demand \( j \in J \) from a mid layer facility located in \( k \in K \)
- \( u_k \) = capacity at reprocessing facility location \( k \in K \)

We consider the following decision variables:

- \( x_{ik} \) = flow from source facility \( i \in I \) to reprocessing facility located at \( k \in K \)
- \( y_{kj} \) = flow from remanufacturing facility located at \( k \in K \) to facility \( j \in J \)

Following the model proposed by [30], the remanufacturing supply chain design problem (RSCP) is defined by:

\[
\text{Min } \sum_{k \in K} f_k w_k + \sum_{k \in K} c_{ik} x_{ik} \sum_{k \in K} \sum_{i \in I} g_k x_{ik} + \sum_{k \in K} \sum_{j \in J} d_{kj} y_{kj} \tag{1}
\]

Subject to

\[
\sum_{i \in I} x_{ik} \leq u_k w_k \quad \forall k \in K \tag{2}
\]
\[
\sum_{k \in K} x_{ik} \leq a_i \quad \forall i \in I \tag{3}
\]
\[
\sum_{j \in J} y_{kj} \geq b_j \quad \forall j \in J \tag{4}
\]
\[
\sum_{i \in I} x_{ik} = \sum_{j \in J} y_{kj} \quad \forall k \in K \tag{5}
\]
\[
x_{ik}, y_{kj} \geq 0, \quad \forall i \in I, j \in J, k \in K \tag{6}
\]
\[
w_k \in \{0,1\} \quad \forall k \in K \tag{7}
\]

The objective function (1) minimizes the sum of the installation reprocessing facility costs plus the delivering costs.
from sourcing facilities to reprocessing facilities and from these to remanufacturing facilities. Constraint (2) warrants that supplying at facility \( k \in K \) is delivered to a mid layer reprocessing facility already opened. Constraint (3) warrants that all the return products from I is going backward to facility \( k \in K \). Constraint (4) warrants that the demand at facility \( j \in K \) must be satisfied by reprocessing facilities. Constraint (5) ensures that all the return products arriving to facility \( k \) are also delivered to remanufacturing facilities. Constraint (6) is a standard positive constraint. Constraint (7) is a standard binary constraint. This model has \( O(n^2) \) continuous variables where \( n=\max(|I|, |J|, |K|) \) and \(|K|\) binary variables. The number of constraints is \( O(n) \).

Notice that following [31], models such as the RSCP forget the origin of the products arriving to remanufacturing facilities then we lost the trace of the products. To overcome this point, we can get another formulation introducing triply subscripted variables \( x_{ijkl} \) to manage the origin to destination product flows and some other changes in parameters, variables and constraints. But this is not the objective of this paper.

4 Evolutionary Scheme

Particle Swarm Optimization (PSO) is an evolutionary computation method to solve continuous optimization problems. This is an optimization algorithm based on swarm theory where the main idea of a classical PSO is to model the flocking of birds flying around a peak in a landscape. In PSO the birds are substituted by artificial beings so-called particles and the peak in the landscape is the peak of an objective (fitness) function. The particles of the swarm are flying through the search solution space with a velocity forming flocks around peaks of fitness functions. In a continuous PSO, an individual particle’s status \( i \) on the search solution space \( D \) is characterized by two factors: its position \( u \) and velocity \( v \). The position \( u \) and velocity \( v \) of the \( ith \) particle in the \( d \)-dimensional search solution space can be represented as:

\[
u_{id}^{t+1} = c_1 v_{id}^{t} + c_2 r_1(p_{id} - u_{id}^{t}) + c_3 r_2(p_{gd} - u_{id}^{t}) \tag{8}
\]

\[
u_{id}^{t+1} = u_{id}^{t} + v_{id}^{t+1} \tag{9}
\]

Where \( c_1 \) is called the inertia weight factor, \( c_2 \) and \( c_3 \) are constants called acceleration coefficients, \( r_1 \) and \( r_2 \) are two independent random numbers uniformly distributed in the range of \([0, 1]\). \( p_{id} \) corresponding to the personal best objective value of particle \( i \) obtained so far at time \( t \). \( p_{gd} \) represents the best particle found so far at time \( t \). Equation (8) stands for calculating the new velocity of each particle \( i \) at time \((t+1)\). Equation (9) stands for updating the position of particle \( I \) at time \((t+1)\). Each \( v_{id}^{t} \in [-v_{max}, v_{max}] \) and \( u_{id}^{t} \in [-x_{max}, x_{max}] \), with \( v_{max} \) and \( x_{max} \) set by users to control excessive roaming of particles outside the search solution space. Particles fly toward a new position according to (9). The process is repeated until a user-defined stopping criterion is reached.

4.1 The Binary PSO Algorithm

In order to manage discrete optimization problems, in [32] is proposed a binary PSO (BPSO) algorithm. The difference between a BPSO algorithm and traditional PSO algorithm is that (9) has been replaced by the following expression:

\[
u_{id}^{t+1} = \begin{cases} 1 & \text{if rand}(\cdot) < S(u_{id}^{t+1}) \\ 0 & \text{if rand}(\cdot) > S(u_{id}^{t+1}) \end{cases} \tag{10}
\]

Where \( S(\cdot) \) is the Sigmoid function and \( \text{rand}(\cdot) \) is a random number uniformly distributed in the range \([0,1]\). We use this method in this paper.

Typical PSO algorithm encodes the whole solution for the RSCP model as a particle, and then applies the traditional algorithm for solving the target problem. We follow a different approach in this paper, taking advantage of the structure of the RSCP model.

We can set to 0 or 1 the value of variables \( w_k \) \( \forall k \in K \), let call them \( w_k^*\). Once you set variables \( w_k^* \) we can re-write the RSCP model as follow:

\[
RSCP_k \min \sum_{k \in K} f_k w_k^* + \sum_{k \in K} \sum_{i \in K} c_{ik} x_{ik} + \sum_{k \in K} \sum_{j \in J} d_{kj} y_{kj} \tag{1a}
\]

Subject to:

\[
\sum_{i \in I} x_{ik} \leq u_i w_k^* \forall k \in K^* \tag{2a} - (6)
\]

Where, the set \( K^* \) contains the facilities already opened, and the variables \( w_k^* \) are now coefficients with known values. Constraint (2a) continues to ensure that all the sourcing facilities deliver return products to a facility already opened. The rest of constraints remain the same as the original RSCP model. Moreover, this associated problem is a linear programming problem.

In our evolutionary scheme, we use a BPSO algorithm for guiding the whole process of seeking an optimal solution for the RSCP problem. However, part of the particle is obtained by solving to optimality the RSCP model. In this way, we use a decomposed BPSO algorithm, based into two groups of swarms, one of them guide the overall search for an optimal solution.

4.2 The particle coding method

To find a good coding method corresponding to the optimization problem is the most critical problem [27]. In this paper, the particle’s d-dimensional space is divided into two sets \( u^1 \) and \( u^2 \). The length of each set (vector) respectively corresponds to the number of candidate sites to locate reprocessing facilities \(|K|\), the second part represents the solution to the relaxed LP associated.

In the first group of particles, every component of each vector can take only 1 or 0. If the kth component of \( u^1 \) is equal to 1, then the reprocessing facility at candidate site \( k \) must be open, 0 otherwise. The second group of particles represents the
flows of return products between the sourcing facility and the reprocessing facilities and between the reprocessing facilities and the remanufacturing facilities. As described earlier in this paper, this part is obtained solving to optimality the RSCP_R problem.

4.3 The feasibility procedure

The standard BPSO algorithm could generate some particles of the first swarm group representing unfeasible solutions. To overcome this problem, for each particle of the first group we apply a simple feasible solution procedure. This procedure consists in two steps. In the first step we check if the number of facilities already opened is enough to satisfy the entire demand. In the second step, for those particles that do not satisfy the demand, we get a feasible solution to RSCP based on the initial particle. This procedure is rather simple; it starts open facilities till the demand is satisfied. The swarm is composed of just feasible solutions. The fitness of a particle is calculated by solving simultaneously the RSCP_R problem and using the objective function (1a).

4.4 Other considerations

We initialize our algorithm with a size swarm of 20 particles. The maximum number of iterations was also set to 10. We used these setting parameters because initial testing provided good results besides the complexity of the problem addressed in this article. The acceleration coefficient c_2 is set to 2 and c_3 to 1. The inertia coefficient starts with a value of 0.9 and it decreases till get 0.4 depending on the number of iterations performed. The initial swarm is composed of just feasible solutions. Each particle of the first group is generated randomly using a uniform distribution in the range [0,1]. Then is applied the binary procedure and after that is applied the feasibility procedure. Regarding the second group of particles, for each particle we solve to optimality the LP problem described earlier in this article. The velocity vector v^i_{da} is generated randomly in the range [-4,4] as in the continuous method. At every iteration, the algorithm uses the method of coding described earlier and updates the position vector u^i_{da} according to (9). The velocity vector v^f_{da} is updated at each iteration according (8). The fitness function is like (1). It is straightforward to calculate its value from u^i_{da}. At every iteration we check for updating the best position of a particle and the global best position of the swarm. Notice that we do not explore swarm neighborhood structures, i.e. the way information is distributed among its members.

5 Numerical Experiments

In this section we discuss and compare the computational results obtained by the proposed evolutionary algorithm. Our propose is to analyze the performance of the proposed algorithm regarding computational time to get the solution and the quality of the upper bound (solution) obtained.

We implemented the algorithm in GAMS and we used CPLEX as a subroutine (called from inside GAMS) for solving to optimality the RSCP_R problem. All the experiments were developed on a PC with 4Gb RAM and 2.3GHz. In the literature, there are not large data sets available for our problem. We generated randomly 10 test problems following similar methodologies used for well known related supply chain problems (for example: [15]). These test problems are data sets corresponding to networks of up to 350 originations sites, 100 candidate sites for locating reprocessing facilities and 40 remanufacturing facilities. The data set for the test problems are given in Table 1 and they are available with the authors. All the transportation costs were generated randomly using a uniform distribution with parameters [1,40].

Management costs were set to 30 for all the instance problems except for the No.1. Instances 7 and 8 are the same than instances 4 and 5 respectively except for the value of fixed costs that were obtained multiplying by 10 the fixed costs of instances 4 and 5. Instances 9 and 10 are the same than instances 7 and 8 respectively but the capacity of each location was increased in 50%. Sourcing units (a_i), capacity of reprocessing facilities (m_j) and capacity of remanufacturing facilities (b_l) are shown in Table 1.

Table 1: Data set

<table>
<thead>
<tr>
<th>#</th>
<th>Instance Problems</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>f_k</th>
<th>a_j</th>
<th>m_k</th>
<th>b_l</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>400x20x15</td>
<td>40</td>
<td>20</td>
<td>15</td>
<td>300</td>
<td>150</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>2</td>
<td>1000x40x20</td>
<td>100</td>
<td>40</td>
<td>20</td>
<td>500</td>
<td>150</td>
<td>400</td>
<td>750</td>
</tr>
<tr>
<td>3</td>
<td>1500x40x20</td>
<td>150</td>
<td>40</td>
<td>20</td>
<td>1000</td>
<td>200</td>
<td>800</td>
<td>1500</td>
</tr>
<tr>
<td>4</td>
<td>2000x80x20</td>
<td>200</td>
<td>80</td>
<td>20</td>
<td>1000</td>
<td>300</td>
<td>800</td>
<td>3000</td>
</tr>
<tr>
<td>5</td>
<td>3000x80x40</td>
<td>300</td>
<td>80</td>
<td>40</td>
<td>2000</td>
<td>200</td>
<td>800</td>
<td>1500</td>
</tr>
<tr>
<td>6</td>
<td>3500x100x40</td>
<td>350</td>
<td>100</td>
<td>40</td>
<td>2000</td>
<td>200</td>
<td>800</td>
<td>1750</td>
</tr>
<tr>
<td>7</td>
<td>4000x100x20</td>
<td>400</td>
<td>100</td>
<td>20</td>
<td>10000</td>
<td>3000</td>
<td>800</td>
<td>3000</td>
</tr>
<tr>
<td>8</td>
<td>5000x80x40</td>
<td>500</td>
<td>80</td>
<td>40</td>
<td>20000</td>
<td>200</td>
<td>800</td>
<td>1500</td>
</tr>
<tr>
<td>9</td>
<td>6000x80x20</td>
<td>600</td>
<td>80</td>
<td>20</td>
<td>10000</td>
<td>3000</td>
<td>1200</td>
<td>3000</td>
</tr>
<tr>
<td>10</td>
<td>7000x80x40</td>
<td>700</td>
<td>80</td>
<td>40</td>
<td>20000</td>
<td>200</td>
<td>1200</td>
<td>1500</td>
</tr>
</tbody>
</table>

We run 5 trials for each test instances. Table 2 and 3 show the results obtained using the algorithm. For each instance problem and trial, the tables show the solution provided by the initial random phase of the algorithm (z_{ran}), the solution provided by the BPSO2 algorithm (z_p), the computing times (seconds) and the gap (100[z^{*} - z_{p}]/z^{*}) obtained comparing the z^{*} with the optimal integer value of the objective function (z^{*}) obtained by GAMS. All the GAP values were rounded to two decimals. The table also shows the minimum, maximum and average value for each column and each test instance. Observe that, for all the test instances, the maximum average gap is 1.58% (instance #9) and the minimum average gap is 0.04% (instance #2). Regarding computing times, notice that all test instances were solved in less than 58 seconds.

6 Conclusions

In this paper, we proposed a Binary PSO-based scheme for solving a reverse supply chain network design problem. This is a NP-hard problem and was formulated as a mixed integer 0-1 linear programming problem (MIP). The algorithm guides the search for an optimal solution to the RSCP problem combining a random solution search with optimal solutions generated by solving to optimality an associated problem. The algorithm generates two groups of swarms. First is generated the particles representing whether the facility is opened or closed. The second part of the chromosome is obtained solving to optimality the RSCP_R problem associated to the
original RSCP problem. The proposed BPSO algorithm was coded in GAMS and tested using 10 test instances generated randomly. Computational results are regarding gap and computing times are promising.

Table 2: Random phase solution value ($z_{ran}$), GA solution value ($z_{GA}$), computing times (secs) and gap(%)

<table>
<thead>
<tr>
<th># instance</th>
<th>trial</th>
<th>$z_{ran}$</th>
<th>$z_{GA}$</th>
<th>secs</th>
<th>Gap(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>144500</td>
<td>146900</td>
<td>13,40</td>
<td>0,00</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>149400</td>
<td>148550</td>
<td>13,46</td>
<td>1,12</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>142920</td>
<td>149000</td>
<td>13,87</td>
<td>1,43</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>149400</td>
<td>147650</td>
<td>13,51</td>
<td>0,51</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>150050</td>
<td>147900</td>
<td>13,71</td>
<td>0,68</td>
</tr>
</tbody>
</table>

Minimum: 148500,0, 149400,0, 13,40, 0,00
Maximum: 150050,0, 149000,0, 13,87, 1,43
Average: 149310,0, 148000,0, 13,59, 0,75

Table 3: Continuation - random phase solution value ($z_{ran}$), GA solution value ($z_{GA}$), computing times (secs) and Gap(%)

<table>
<thead>
<tr>
<th># instance</th>
<th>trial</th>
<th>$z_{ran}$</th>
<th>$z_{GA}$</th>
<th>secs</th>
<th>Gap(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1</td>
<td>2765500</td>
<td>2765500</td>
<td>40,01</td>
<td>0,28</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2768700</td>
<td>2765600</td>
<td>36,20</td>
<td>0,29</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2766700</td>
<td>2765600</td>
<td>35,81</td>
<td>0,29</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2765600</td>
<td>2764800</td>
<td>35,75</td>
<td>0,26</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2765500</td>
<td>2764100</td>
<td>35,78</td>
<td>0,23</td>
</tr>
</tbody>
</table>

Minimum: 2765500,0, 2766700,0, 35,81, 0,29
Maximum: 2768700,0, 2765600,0, 36,20, 0,29
Average: 2766400,0, 2765120, 36,71, 0,27

8
<table>
<thead>
<tr>
<th># instance</th>
<th>trial</th>
<th>$z_{ran}$</th>
<th>$z_{GA}$</th>
<th>secs</th>
<th>Gap(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3481700</td>
<td>3480800</td>
<td>55,85</td>
<td>0,23</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3483000</td>
<td>3481400</td>
<td>54,33</td>
<td>0,25</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3481300</td>
<td>3480700</td>
<td>54,68</td>
<td>0,23</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3482600</td>
<td>3481400</td>
<td>55,05</td>
<td>0,25</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3482900</td>
<td>3480200</td>
<td>56,52</td>
<td>0,22</td>
</tr>
</tbody>
</table>

Minimum: 3481300,0, 3482600,0, 55,05, 0,25
Maximum: 3483000,0, 3481400,0, 56,52, 0,22
Average: 3482300,0, 3480900, 55,28, 0,24

9
<table>
<thead>
<tr>
<th># instance</th>
<th>trial</th>
<th>$z_{ran}$</th>
<th>$z_{GA}$</th>
<th>secs</th>
<th>Gap(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2727700</td>
<td>2521400</td>
<td>35,46</td>
<td>1,69</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2523500</td>
<td>2513600</td>
<td>36,06</td>
<td>1,38</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2531000</td>
<td>2517200</td>
<td>36,55</td>
<td>1,52</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2531000</td>
<td>2519900</td>
<td>36,48</td>
<td>1,63</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2535500</td>
<td>2521100</td>
<td>36,79</td>
<td>1,68</td>
</tr>
</tbody>
</table>

Minimum: 2523500,0, 2531000,0, 36,55, 1,52
Maximum: 2535500,0, 2521100, 36,79, 1,68
Average: 2570520,0, 2518640, 36,36, 1,58

10
<table>
<thead>
<tr>
<th># instance</th>
<th>trial</th>
<th>$z_{ran}$</th>
<th>$z_{GA}$</th>
<th>secs</th>
<th>Gap(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2996600</td>
<td>2992000</td>
<td>38,97</td>
<td>1,09</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2999900</td>
<td>2995600</td>
<td>46,73</td>
<td>1,21</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2995400</td>
<td>2999000</td>
<td>48,01</td>
<td>1,05</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3003400</td>
<td>2996000</td>
<td>47,72</td>
<td>1,35</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3000000</td>
<td>2994000</td>
<td>46,95</td>
<td>1,16</td>
</tr>
</tbody>
</table>

Minimum: 2995400,0, 2999000,0, 48,01, 1,05
Maximum: 3003400,0, 2996000,0, 47,72, 1,35
Average: 2998800,0, 2994420, 45,68, 1,17

References


Acknowledgments

This research was partially supported by the Fundação de Amparo á Pesquisa do Estado de Minas Gerais – FAPEMIG, VALE e CNPq - Brazil.


