An Evolutionary Scheme for Solving a Reverse Supply Chain Design Problem

Ernesto Del R. Santibanez-Gonzalez\(^1\), and Henrique Pacca Luna\(^2\)

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

Abstract - According to some authors, design strategies for reverse Supply Chain (SC) are relatively unexplored and underdeveloped. We propose a genetic algorithm (GA) that guides the search for an optimal solution to an NP-hard remanufacturing SC problem. It combines a random solution search with an optimal solution method for solving to optimality an associated LP problem. In this GA, each chromosome is generated into two steps. First is generated the part of the chromosome representing whether the facility is opened or closed. The second part of the chromosome is obtained solving to optimality a LP problem associated to the original problem. The proposed genetic algorithm was coded in GAMS. We report computational results for 11 network instances generated randomly with up to 350 sourcing facilities, 100 candidate sites for locating reprocessing facilities and 40 remanufacturing facilities \((350\times100\times40)\). Computational results regarding GAP and computing times are promising.

Keywords: Evolutionary algorithm, sustainable supply chain problem; remanufacturing location problems; integer programming

1 Introduction

The management of product return flows has received increasing attention in the last decade. The efficient waste and product return flows stewardship 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. Notice that by 2013 the total e-waste will reach 73 million metric tons. These products not only need to get back to the supplier, re-use or recycled, some disposal also remain in the earth producing several kind of damages. Several regions and nations set up tighter environmental objectives in a collaborative action to mitigate the potential damage for the economic and also the health of the people. There are specific rules in some regions like UE, where The European Waste Electrical and Electronic Equipment Directive (WEEE) and 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 environment restrictions, there are several reasons because an increasing number of companies will be interested into get engaged in sustainable initiatives like the management of reverse flows, going backwards from customers to recovery centers, within their supply chain \([1, 2]\). Some authors noted that huge monetary values “can be gained by redesigning the reverse supply chain to be faster and reduce costly time delays” \([3]\). According to the same authors, design strategies for reverse supply chains are relatively unexplored and underdeveloped. Returned products are remanufactured if judged cost-effective. Some firms may treat all product returns as defective. Some returned products may be new and never used; then these products must be returned to the forward flow. In this context, remanufacturing activities are recognized as a main option of recovery in terms of its feasibility and benefits. We study this problem as part of reverse logistics problems. In this paper is addressed the problem of designing a remanufacturing supply chain network and it is proposed a type of genetic algorithm for solving it. The problem is a NP-hard combinatorial optimization problem. For randomly generated test instances of the problem, we analyze the performance of the algorithm in term of computing times and quality of the solution obtained.

This paper makes two primary contributions. First it proposes an evolutionary scheme, combining an optimization method inside of a genetic algorithm scheme for solving a reverse distribution problem. Second, a computational study of the method is performed for problem instances of up to 350 sourcing facilities, 100 candidate sites for locating reprocessing facilities and 40 remanufacturing facilities \((350\times100\times40)\), concluding regarding the quality of the solution obtained and the computing times. These are the largest instances problems tested so far with evolutionary algorithms.

In the second section, we provide a literature review with a brief introduction to sustainable and reverse supply chain and particularly to the remanufacturing case in connection with reverse logistics. In the third section, we formulate the mathematical model for the problem and propose the genetic algorithm for solving it. In the fourth section, we present experimental results for large sets of networks generated randomly. The last section contains our conclusions.
2 Literature review

Adopting a friendly sustainable management implies a number of changes for companies from the strategy level up to the operational point of view, affecting their people and impacting their business processes and technology. In this regard, in [4] is noted, “the strategic level deals with decisions that have a long-lasting effect on the firm. These include decisions regarding the number, location and capacities of warehouses and manufacturing plants, or the flow of material through the logistics network”. They established a clear link between facility location models and strategic decisions of supply chain management (SCM). Supply Chain Management - SCM, and in particular Sustainable Supply Chain Management – SSCM gives a good framework to address sustainable issues. Green supply chain management (GSCM) was emerging in the last few years. This concept covers every stage in manufacturing from the first to the last stage of product life cycle, i.e. from product design to recycle. [5] made a carefully literature review and he shows that a broad frame of reference for Green Supply Chain Management (GSCM) is not adequately developed. As a consequence, the author identifies the need for a “succinct classification to help academicians, researchers and practitioners in understanding integrated GSCM from a wider perspective”. We focus on the remanufacturing network (RMN) 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 equivalent in quality and performance to the original new product and then can be resold as new products or parts.

2.1 Research on Facility Location Problems for Reverse Supply Chain Design

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]. Such as it was observed by [8], some small changes to classical facility location models are 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. Several authors have studied different aspects of closed-loop supply chain problems, see for example [9-14]. In [2] are discussed the new issues that arise in the context of reverse logistics and reviewed the mathematical models proposed in the literature. In [15] is proposed a generic recovery network model based on the elementary characteristics of return networks identified in [2]. In [16] is proposed a generic mixed integer model for the design of a reverse distribution network including repairing and remanufacturing options simultaneously.

Regarding reverse logistics networks in connection with location problems, in [17] is presented a two-level distribution and waste disposal problem, in which demand for products is met by plants while the waste generated by production is correctly disposed of at waste disposal units. In [18] is described a network for recycling sand from construction waste and proposed a two-level location model to solve the location problem of two types of intermediate facility. Regarding remanufacturing location models, in [19] is described a small reverse logistics network for the returns, processing, and recovery of discarded copiers. They presented a mixed integer linear programming (MILP) model based on a multi-level uncapacitated warehouse location model. The model was used to determine the locations and capacities of the recovery facilities as well as the transportation links connecting various locations. In [20] is proposed a 0-1 MILP model for a product recall distribution problem. They analyzed the particular case in which the customer returns the product to a retail store and the product is sent to a refurbishing site which will rework the product or dispose it properly. The reverse supply chain is composed of origination, collection and refurbishing sites. With the objective to minimize fixed and distribution costs, the model has to decide which collection sites and which refurbishing sites to open, subject to a limit on the number of collection sites and refurbishing sites that can be opened. In [21] are presented three generic facility location MILP models for the integrated decision making in the design of forward and reverse logistics networks. The formulations are based on the well-known uncapacitated fixed-charge location model and they include the location of used product collection centers and the assignment of product return flows to these centers. In [22] is presented a two-level location problem with three types of facility to be located in a reverse logistics system. They proposed a 0–1 MILP model which simultaneously considers “forward” and “reverse” flows and their mutual interactions. The model has to decide the number and locations of three different types of facilities: producers, remanufacturing centers and intermediate centers.

3 Modeling a remanufacturing supply chain network design problem (RSCP)

In this section we present the MIP model for our 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 call 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. All returned products (used products) are first shipped back to collection facilities where some of them will be disposed of for various reasons, like poor quality. The checked return-products will then be sent back to remanufacturing facilities, where some of them may still be disposed of. 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 \( j \in J \)
- \( f_k \) = 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 \( k \in K \)
- \( c_{ik} \) = 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:

- \( w_k \) = 1 if we locate a reprocessing facility at candidate site \( k \in K, 0 \) otherwise
- \( x_{ik} \) = 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 in [23], the remanufacturing supply chain design problem (RSCP) is defined by:

\[
M \ln \sum_{k \in K} f_k w_k + \sum_{k \in K} \sum_{i \in I} c_{ik} x_{ik} \sum_{k \in K} \sum_{i \in I} g_k x_{ik} + \sum_{k \in K \ j \in J} d_{kj} y_{kj} \quad (1)
\]

Subject to

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

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) warranties that supplying at facility \( k \in K \) is delivered to a mid layer reprocessing facility already opened. Constraint (3) warranties that all the return products from I is going back to facility \( k \in K \). Constraint (4) warranties that the demand at facility \( j \in J \) 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 \( O(n^3) \) continuous variables where \( n = \max{|I|, |J|, |K|} \) and \( |K| \) binary variables. The number of constraints is \( O(n) \).

Notice that following [24], models like 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 \( x_{ikj} \) 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.

3.2 An Evolutionary Scheme

Genetic Algorithms (GA) are a type of evolutionary algorithms (EVA) used to solve a number of combinatorial optimization problems. See for further details the paper by [25]. An EVA is composed by five basic components: (a) a genetic representation of solutions to a problem; (b) a way to create an initial population of solutions; (c) an evaluation function; (d) genetic operators that alter the genetic composition of children during reproduction and (e) values for the parameters [26]. In this section we briefly describe these components and the GA implementation for solving the RSCP problem.

Typical GA encodes the whole solution for the RSCP model as a chromosome, and then applies the above scheme for solving the target problem. We follow a little different approach in this paper.

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:

\[
RSCP_k
\]
In our evolutionary scheme, we use a genetic algorithm for programming problems.

The evolutionary scheme is as follows:

1. **Generate** an initial population for the head of the chromosome:
   a. Apply the feasibility procedure to satisfy capacity constraints.
   b. Solve to optimality the RSCP$_R$ problem.

2. **While** not Finish do
   a. Solve to optimality the RSCP$_R$ problem;
   b. Apply selection process;
   c. Do a partial crossover to the head of the individuals;
   d. Generate individuals using the mutation operation;
   e. Apply the feasibility procedure to satisfy capacity constraints.

3. **Endwhile.**

Observe that, in step (1) we generate each chromosome into two steps. In step (1a) is generated the head of the chromosome. This part of the chromosome represents whether the facility is open or closed as described below. The second part of the chromosome (1b), we call tale, is obtained solving the RSCP$_R$ problem to optimality. When it is solved the RSCP$_R$ problem, it is also simultaneously computed the fitness of each chromosome. In step (2) the cross-over procedure is applied to the head part of the chromosome, and again, the tale part of the chromosome is obtained solving to optimality the RSCP$_R$ problem. The rest of the steps in the above scheme are similar to a classical GA.

### 3.3 Representation, Initial Population and Feasible Solutions

In our implementation, each solution (individual) to the problem is coded as a chromosome. In the first part of the chromosome, called head, each gene corresponds to a facility location decision variable, taking value $1$ if a facility is open at the mid layer, and zero otherwise. We use a $|K|$ dimensional string to represent facilities located. The rest of the chromosome, called the tale, 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, the chromosome’s tale is obtained solving to optimality the RSCP$_R$ problem.

An initial population of 20 individuals is generated randomly. Each gene of the head of the chromosome is generated by a 0-1 uniform probability distribution. For each chromosome of the population 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 chromosomes that do not satisfy the demand, we get a feasible solution to RSCP based on the initial chromosome. This procedure is rather simple; it starts open facilities till the demand is satisfied. The population is composed of just feasible solutions. The fitness of a chromosome is calculated by solving simultaneously the RSCP$_R$ problem and using the objective function (1a).

### 3.4 Genetic Operators

We use the standard genetic operators. The crossover generates one new individual (chromosome) exchanging the genetic material of two (parental) individuals expecting that “good” solutions can generate “better” ones. One of these individuals is selected randomly from a set of 20 individuals, as described earlier. The other individual is the best one selected from the previous population. We do not limit the number of new chromosomes generated by crossover. In this work crossover probability ($\text{cross}_p$) is set to 0.7 (70%) and we perform one-point crossover. In the crossover procedure we generate a random value, if the $\text{cross}_p$ value is greater than the random value then we pick one individual from the set. We generate another random value between one and the number of potential facility sites for the corresponding layer, i.e., a cut point dividing each individual (parent) into two segments. The child is created by combining the first segment from the first parent and the second segment from the second parent. The mutation operator changes the value of a chromosome with some small probability. In our case, we do not practice mutation, we set this probability to 0.0 (0%) and remain constant through generations of the GA. After the crossover procedure, we check for infeasible solutions in the population. In the next step we correct this.

### 3.5 Additional Aspects about the Genetic Algorithm

The selection operator is based on elitist selection, favoring individuals of better fitness value to reproduce more often than the worse ones when generating the new population. In every iteration (generation) the whole population (20 individuals) is ranked in a non-decreasing order of the objective function value, and we select always the best individual. As we described earlier, feasibility of
constraints (2) is verified to every individual at every iteration. In case unfeasibility, a feasible routine is applied to get a 100% of feasible solutions in the population. Considering the optimal procedure we applied inside the genetic algorithm, we work with a reduced population size (20) and with 5 generations (iterations). Nevertheless, the performance in term of quality of the solutions obtained and computing times are promising as described hereafter.

4 Numerical Experiments

In this section we discuss and compare the computational results obtained for the proposed evolutionary algorithm. We analyze the performance of the proposed algorithm regarding computational time to get the solution and the quality of the upper bound obtained.

We implemented the algorithm in GAMS and we used CPLEX as a subroutine (called from inside GAMS) for solving to optimality the RCSPlp problem. All the experiments were developed on a PC with 2Gb RAM and 2.3GHz. In the literature, there are not large data sets available for our problem. We generated randomly 11 test problems following similar methodologies used for well known related supply chain problems [22, 23]. These test problems are data sets corresponding to networks of up to 350 origination 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 7-9 are the same than instances 1-6 except for the value of fixed costs that were obtained multiplying by 10 the fixed costs of instances 1-6. Instances 10 and 11 are the same than instances 8 and 9 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/\_k) are shown in Table 1. To illustrate the size problems, problem instances 5 and 6 involve 27,200 positive variables, 80 binary variables and 500 constraints; and 39,000 positive variables, 100 binary variables and 590 constraints, respectively.

In order to compare the performance of the proposed algorithm, we tested it with the test instances generated randomly. We run 5 trials for each test instance. Table 2 shows the results obtained using the algorithm. For each instance problem and trial, the tables show the solution provided by the random phase of the algorithm (z\_GA), the solution provided by the GA (z\_GA), the computing times (seconds) and the gap (100*(z\_GA-z\_opt)/z\_opt) obtained comparing the z\_GA with the optimal integer value of the objective function (z\_opt) obtained by GAMS. All the gap values were rounded to two decimals. The tables also show the minimum, maximum and average value for each column and each test instance. To illustrate, for problem instance 5 and trial 2, the solution value of the random phase (z\_GA), the GA solution value (z\_GA), the computing times and the gap are 2,131,500.0, 2,129,200.0, 74.39 seconds and 0.31% respectively.

The same problem instance, the minimum, maximum and average gap are 0.31%, 0.48% and 0.4% respectively. Observe that, for all the test instances, the maximum average gap is 3.65% (#1 instance) and the minimum average gap is 0.17% (#2 instance). Regarding computing times, notice that all the test instances were solved in less than 106 seconds.

Table 1: Data set

<table>
<thead>
<tr>
<th>#</th>
<th>Instance</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>f/_k</th>
<th>a/_i</th>
<th>m/_j</th>
<th>b/_k</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40x20x15</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>100x40x20</td>
<td>100</td>
<td>40</td>
<td>20</td>
<td>500</td>
<td>200</td>
<td>750</td>
<td>750</td>
</tr>
<tr>
<td>3</td>
<td>150x40x20</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>200x80x20</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>300x80x40</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>350x100x40</td>
<td>350</td>
<td>100</td>
<td>40</td>
<td>2000</td>
<td>200</td>
<td>800</td>
<td>1500</td>
</tr>
<tr>
<td>7</td>
<td>40x20x15</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>8</td>
<td>200x80x20</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>9</td>
<td>300x80x40</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>10</td>
<td>200x20x80</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>11</td>
<td>300x80x40</td>
<td>300</td>
<td>80</td>
<td>40</td>
<td>2000</td>
<td>200</td>
<td>800</td>
<td>1500</td>
</tr>
</tbody>
</table>

Table 2: Random phase solution value (z\_GA), GA solution value (z\_GA), computing times (secs) and gap (%)
The problem was formulated as a mixed integer 0-1 linear programming problem (MIP) to locate reprocessing facilities in order to minimize the flow cost from the origination facilities. At the last level there are remanufacturing facilities. We need to locate reprocessing facilities in order to minimize the flow cost from the origination to the remanufacturing facilities. The problem was formulated as a mixed integer 0-1 linear programming problem (MIP). The genetic algorithm guides the search for an optimal solution to the RSCP problem and it combines a random solution search with an optimal solution method for solving to optimality an associated problem. In this genetic algorithm, each chromosome is generated into two steps. First is generated the head of the chromosome, 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. Notice that, when the RSCP$^R$ problem is solved, it is also simultaneously computed the fitness of each chromosome. The proposed genetic algorithm was coded in GAMS and tested using 11 test instances generated randomly. Considering the few number of chromosomes and the total number of generations we used, computational results regarding GAP and computing times are promising.

Acknowledgments

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

References


