KES Transactions on Sustainable Design and Manufacturing I
Sustainable Design and Manufacturing 2014 : pp.524-535 : Paper sdm14-083

A Binary Particle Swarm Optimization (BPSO) algorithm to solve a reverse logistics supply chain problem

Ernesto Del R. Santibanez-Gonzalez[1*], Jairo R. Montoya-Torres[2], Luisa Huaccho Huatuco[3]

[1]Department of Computer Science, Federal University of Ouro Preto, Ouro Preto, MG, Brazil, * corresponding author: santibanez.ernesto@gmail.com

[2]Escuela Internacional de Ciencias Económicas y Administrativas, Universidad de La Sabana, Km 7 autopista norte de Bogotá, D.C., Chía (Cundinamarca), Colombia

[3] Leeds University Business School, University of Leeds, Woodhouse Lane, Leeds, LS2 9JT, UK

Abstract The design of a reverse logistics supply chain is a complex problem and is still relatively unexplored and underdeveloped. We propose a binary particle swarm optimization (BPSO)-based scheme for solving a NP-hard reverse logistics supply chain design problem. The algorithm combines a traditional stochastic search with an optimal solution method for solving to optimality a relaxed linear programming (LP) model. 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 swarm group defines the optimal flows between the facilities. We solve to optimality, a relaxed linear problem is obtained from the original problem and then we project the solution into the swarm space. This approach was validated using different data sets (instances) randomly generated with up to 350 sourcing facilities, 100 candidate sites for locating reprocessing facilities and 40 remanufacturing facilities. The results regarding experimental approximability of proposed heuristic versus the optimal solution are very promising, as well as their computation times.

Keywords sustainable supply chain; reverse logistics; remanufacturing; particle swarm optimization; integer programming.

1. Introduction

In this paper, we focus on the remanufacturing supply chain design problem and then on reverse logistics 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. As a consequence, these remanufactured products or parts can be resold as new products or parts. Because remanufacturing activities are often implemented by the original producer, such a supply chain is likely to be a closed-loop system. Remanufacturing activities are recognized as a main option of recovery in terms of its feasibility and benefits. It provides firms a way to master the disposal of their used products, to reduce effectively the costs of production and save raw materials. When designing a reverse logistics supply chain for remanufactured products, several well-known supply chain design problems arise. For instance, Facility location is one of the strategic problems being part of the supply chain planning process. 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 [5]. 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” [2]. As observed by Farahani and Hekmatfar [9], some small changes to classical facility location models turn these problems quite hard to solve.

Particle Swarm Optimization (PSO) is one of the meta-heuristic algorithms that have been successfully applied in many areas and is a suitable approach for several optimization problems. As pointed out by some authors, this technique has succeeded in many continuous problems, however, in discrete or binary version there are still some difficulties [8]. In this paper, we adapt the binary PSO algorithm to solve the reverse logistics supply chain design problem and analyze its performance, in terms of computational times and quality of the solution obtained, over a set of randomly generated instances (data sets).

This paper makes two primary contributions. Firstly, it proposes a PSO-based scheme, combining an optimization method inside of a PSO algorithm scheme for solving a reverse distribution problem. Secondly, in contrast with traditionally studied approaches that focus only on small sized problems, the current paper presents a set of computational experiments carried out on medium to large sized data sets (instances).

2. Literature Review

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 management. Dekker et al. [6] classified the research on reverse logistics (and closed-loop supply chain) into three functional areas: Distribution, Inventory and Production and Supply Chain Scope. Traditional distribution decisions involve the design of supply chain and the location of forward and reverse facilities for the distribution of products and for collecting and reprocessing returned products. These authors focused on quantitative models for designing closed-loop supply chains, and were concerned with the decisions regarding the topological structure of the supply chain, the number of facilities to locate, their locations and capacities and the allocation of product flows between the facilities. Rubio et al. [23] carried out a compilation of research published on reverse logistics within the period 1995-2005. They studied the topic based on the classification proposed by Dekker et al. [6].

Reverse logistics models have been discussed previously (e.g. [7],[10],[12],[21],[24],[25],[27]). Almost all the authors proposed MILP models. The majority of solution methods are based on standard commercial packages. Closed-loop supply chain models have also been taken into account (e.g. [1],[3],[4],[13],[18],[20],[28]. Stochastic models in combination with multi-objective functions were presented by Pishvaee and Torabi [22], while the work of Govindan et al. [11] presented a multi-criteria decision making approach based on the Analytical Network Process (ANP) to solve the problem of selecting a reverse logistics provider.

Because of the computational complexity of this problem, its solution using meta-heuristic algorithms has also been of interest by some researchers. Procedures such as Simulated Annealing [15],[21], Genetic Algorithms [16],[19],[28], Memetic Algorithms [22] and Tabu Search [15] have been proposed. Notice that Lee et al. [16] 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 solution obtained, measured using the gap (approximation) to the optimal solution. Lee and Dong [15] solved problems with supply chains of up to 100x40x30 with a maximum computing time of 2,939 seconds and a maximum gap (compared to a lower bound) of 15%.

3. Mathematical model for designing a remanufacturing supply chain

In this section, we present a mixed-integer linear programming (MILP) model for the problem of designing a remanufacturing supply chain. This problem can be categorized as a single product, static, three-echelon, capacitated location model with known demand. The remanufacturing supply chain consists of three types of members: sourcing facilities (e.g. a retail store), collection sites and remanufacturing facilities. At the customer level, there are product demands and used products ready to be recovered such as, 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, 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, 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. Figure 1 shows a reverse logistics supply chain with six sourcing facilities, eight candidate sites for locating reprocessing facilities (three out of eight facilities are opened) and two remanufacturing facilities

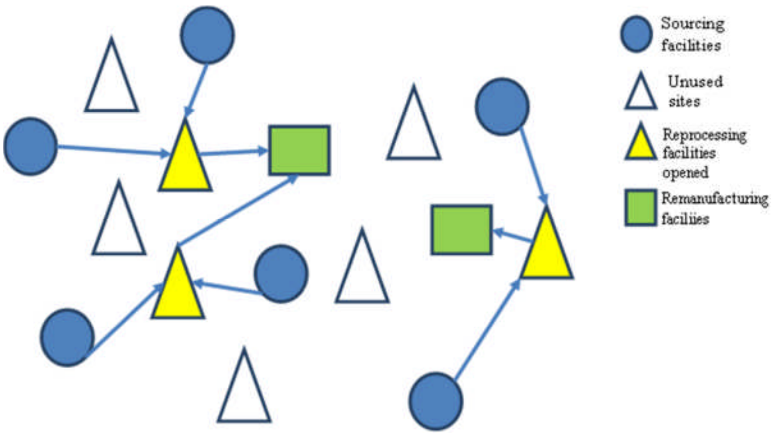

Figure 1. *Schematic representation of the reverse logistics supply chain under study*

Remanufacturing supply chain design problem (RSCP) Model

Our model assumes that 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_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 by Li [17], the remanufacturing supply chain design problem (RSCP) is defined by:

$$Minimize \sum_{k \in K} f_k w_k + \sum_{i \in I} \sum_{k \in K} c_{ik} x_{ik} + \sum_{k \in K} \sum_{i \in I} g_k x_{ik} + \sum_{k \in K} \sum_{j \in J} d_{kj} y_{kj} \qquad (1)$$

$$\sum_{i \in I} x_{ik} \le u_k w_k \qquad \forall\, k \in K \qquad (2)$$

$$\sum_{k \in K} x_{ik} \le a_i \qquad \forall\, i \in I \qquad (3)$$

$$\sum_{k \in K} y_{kj} \ge b_j \qquad \forall\, j \in J \qquad (4)$$

$$\sum_{i \in I} x_{ik} = \sum_{j \in J} y_{kj} \qquad \forall\, k \in K \qquad (5)$$

$$x_{ik}, y_{kj} \ge 0, \qquad \forall\, i \in I, \forall\, j \in J, \forall\, k \in K \qquad (6)$$

$$w_k \in \{0,1\} \qquad \forall\, k \in K \qquad (7)$$

The objective function (1) minimizes the sum of the installation reprocessing facility costs plus delivering costs from sourcing facilities to reprocessing facilities and from these to remanufacturing facilities. Constraint (2) ensures that supplying at facility $k \in K$ is delivered to a mid-layer reprocessing facility already opened. Constraint (3) ensures that all return products from $i \in I$ go backward to facility $k \in$ K. Constraint (4) ensures that demand at remanufacturing facility $j \in J$ is satisfied by reprocessing facilities $k \in K$. Constraint (5) ensures that all return products arriving to facility *k* are also delivered to remanufacturing facilities. Constraints (6) and (7) are respectively standard positive and binary constraints.

4. Solution approach based on PSO

In our solution approach, 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 $RSCP_R$ problem. 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.

Let L be the size of the swarm, each particle i has the position $u_i \in \{0,1\}^n$ in the search space and a velocity $\boldsymbol{v}_i \in \mathrm{R}^n$. Let p_i be the best known position for particle i and p_g the best known position for the swarm. The fitness function *f(.)* is the objective function value for a particle. The pseudo-code of our proposed Binary PSO Algorithm is shown in Table 1.

Table 1. *Pseudo-code of proposed BPSO algorithm*

1. **Generate** the swarm
    a. **For** each particle in the swarm $i = 1, ...,L$ **do**:
    b. **Initialize** the position of particle i using a random uniform distribution: $u_i \sim U(0,1)$;
    c. **Analyze** feasibility for each particle i. Whenever necessary apply the feasibility procedure;
    d. **Solve** to optimality the $RSCP_R$ problem and complete the dimension of particle i;
    e. **Initialize** the best position of particle i as its initial position: $\mathrm{p_i} \leftarrow u_i$;
    f. **If** ($f(p_i) < f(p_g)$) **update** the best swarm position: $p_g \leftarrow p_i$;
    g. **Initialize** velocity of particle: $\mathrm{v_i^t} \sim U(-4, 4)$
2. **While** stopping criteria **do**:
    a. **Update** value of t;
        i. **For** each particle $i= 1, ...,L$ **do**:
            1. **For** each dimension $d = 1, ..., n$ **do**:
                a. Select random numbers: $r_1, r_2 \sim U(0,1)$
            2. **Update** velocity of particle using (8)
            3. **Update** position of particle using (10)
            4. **Solve** to optimality the RSCPR problem and complete the dimension of particle;
            5. **If** ($f(\boldsymbol{x}_i) < f(p_i)$) **do**:
                a. **update** the best position of particle: $p_i \leftarrow \boldsymbol{x}_i$
                b. **If** ($f(p_i) < f(p_g)$) update the best swarm position: $p_g \leftarrow p_i$
    b. p_g contains the best solution found so far.
    c. **EndWhile**

4.2 The particle coding method

To find a good coding method corresponding to the optimization problem is the most critical problem [14]. 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 *k*-th 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 Generating the initial swarm

For step (1) of the algorithm, the initial swarm of size *L*, with the position of each particle *i* of the swarm is generated as follows:

Steps (1b) and (1c)

For particles belonging to the first swarm group, the position is generated randomly. First, a uniform distribution between [0,1] is used to generate a value *rand()*. For each velocity vector v_{id}^t , t=0, the position *d* is generated randomly within the interval [-4,4] as is usually done in the continuous PSO algorithm. Equation (10) is used to obtain the initial position of a particle *i*. Alternatively, a binary uniform distribution can be used to generate the initial position u_i. A *k*-dimensional vector is used to represent the initial position u_i of each particle, where |*K*| is the number of candidate sites to locate a reprocessing facility. This procedure can generate unfeasible solutions to the problem.

Step (1d)

For each particle of the first group corresponds to one particle of the second group. The position of a particle in the second group is generated by solving the associated LP model, which in turn considers the position of a particle in the first group. For instance, let us suppose that for particle u_3 is solved the problem $RSCP_R$,. The position for particle u_3 of this group can be the following:

(100;0;150;0;0;150;0;100;250;0;0;250).

For this instance problem with 4 sourcing facilities, 2 reprocessing facilities and 2 remanufacturing facilities, the position of u_3 indicates that 100 units of products flow from the first source facility to the first reprocessing facility opened, 150 units of products flow from the second sourcing facility to the first reprocessing facility opened, 150 units of products flow from the third sourcing facility to the second reprocessing facility opened and 100 units of products flow from the fourth sourcing facility to the second reprocessing facility opened. Then, 250 units of products are transported from the first reprocessing facility opened to the first remanufacturing facility opened and the same volume is transported from the second reprocessing facility opened to the second remanufacturing facility opened.

5. Numerical experiments

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

We implemented the MILP models in GAMS (General Algebraic Modeling System) and used CPLEX solver as a subroutine 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 hence randomly generated 10 test problems following similar methodologies used for well-known related supply chain problems (e.g. [18],[26]). These test problems are data sets corresponding to supply chains of up to 350 origination sites, 100 candidate sites for locating reprocessing facilities and 40 remanufacturing facilities. The data sets for the test problems are given in Table 2.

Table 2. *Description of data sets*

#	Problem size	Origi-nation sites	Candi-dates reproce-ssing facilities	Remanu-facturing facilities	Fixed costs	a_j	m_k	b_l
I1	40x20x15	40	20	15	300	150	400	400
I2	100x40x20	100	40	20	500	150	400	750
I3	150x40x20	150	40	20	1000	200	800	1500
I4	200x80x20	200	80	20	1000	300	800	3000
I5	300x80x40	300	80	40	2000	200	800	1500
I6	350x100x40	350	100	40	2000	200	800	1750
I7	200x80x20	200	80	20	10000	300	800	3000
I8	300x80x40	300	80	40	20000	200	800	1500
I9	200x80x20	200	80	20	10000	300	1200	3000
I10	300x80x40	300	80	40	20000	200	1200	1500

We run 5 trials for each test instance. Table 3 presents the results obtained using the algorithm.

Table 3. *Comparison of solution values (z_{ran}: algorithm's random phase, z_P: final solution BPSO)*

# instance		z_{ran}	z_P	%dev (OPT)	%imp(z_{ran}*z_p)
	Minimum	148500	146900	0.00	1.08
I1	**Maximum**	150050	149000	1.43	0.70
	Average	149310	148000	0.75	0.88
	Minimum	520300	520250	0.03	0.01
I2	**Maximum**	521350	520300	0.04	0.20
	Average	520750	520290	0.04	0.09
I3	**Minimum**	1065400	1065200	0.23	0.02

	Maximum	1066600	1065800	0.28	0.08
	Average	1065960	1065440	0.25	0.05
	Minimum	2090500	2089000	0.30	0.07
I4	**Maximum**	2091400	2090500	0.37	0.04
	Average	2090740	2089800	0.34	0.04
	Minimum	2130800	2130800	0.38	0.00
I5	**Maximum**	2132400	2131100	0.40	0.06
	Average	2131780	2130920	0.39	0.04
	Minimum	2468750	2467350	0.35	0.06
I6	**Maximum**	2471900	2469550	0.44	0.10
	Average	2470920	2468400	0.40	0.10
	Minimum	2765500	2764100	0.23	0.05
I7	**Maximum**	2768700	2765600	0.29	0.11
	Average	2766400	2765120	0.27	0.05
	Minimum	3481300	3480200	0.22	0.03
I8	**Maximum**	3483000	3481400	0.25	0.05
	Average	3482300	3480900	0.24	0.04
	Minimum	2525300	2513600	1.38	0.46
I9	**Maximum**	2727700	2521400	1.69	7.56
	Average	2570520	2518640	1.58	2.02
	Minimum	2994500	2990900	1.05	0.12
I10	**Maximum**	3003400	2999600	1.35	0.13
	Average	2998880	2994420	1.17	0.15

For each instance problem, the table provides solution obtained at the initial random phase of the algorithm (z_{ran}), the final value obtained by BPSO algorithm (z_P). The last two columns of this table present respectively the percentage deviation, denoted as %*dev*(*OPT*), between the final solution *zp* against the optimal value (z^*) obtained using the MILP model, and the percentage of improvement of the objective function value (e.g., z_{ran} versus z_p). These two metrics are computed using Equations (11) and (12) respectively:

$$\%dev(OPT) = \frac{z^* - z_p}{z^*} \times 100\% \tag{11}$$

$$\%imp(z_{ran} * z_p) = \frac{z_{ran} - z_p}{z_{ran}} \times 100\% \tag{12}$$

All values of %*dev* and %*imp* were rounded to two decimals. Observe that, for all the test instances, the maximum average %*dev* is 1.58% (instance #9) and the minimum average %*dev* is 0.04% (instance #2). Note also that BPSO procedure obtained the optimal solution for instance I9. From the results presented in this table, we observe that the proposed BPSO performs very well since percentage deviations from optimal values are never higher than 1.70% (for the maximum values) and no greater than 0.54% in average. In addition, according to values obtained for metric %*imp*, we observe that the improvement of the initial solution is about 0.35%, in average, which means that the BPSO algorithm shows very fast convergence. It is to note that the highest value of this improvement is observed for instance #9, with maximum and averages values of 7.56% and 2.02%, respectively. This means that, in one of the trials, the initial random solution employed by BPSO had to be improved in a 7.56%. When regarding the maximum value of the final deviation of the BPSO solution against the optimum for this data set (instance #9), we notice that again BPSO algorithm perform very well because the final solution is only 1.69% higher than the optimal value given by MILP. Finally, looking at computational times, we notice that all test instances were solved in less than 58 seconds, as shown in Table 4.

Table 4. *Computational times (in seconds) of BPSO*

# instance	Minimum	Maximum	Average
I1	13.40	13.87	13.59
I2	15.73	15.98	15.88
I3	19.01	20.10	19.75
I4	32.40	34.85	33.45
I5	54.80	58.32	56.35
I6	53.82	54.70	54.29
I7	35.75	40.01	36.71
I8	54.33	56.52	55.28
I9	35.46	36.79	36.27
I10	38.97	48.01	45.68
Average	35.367	37.915	36.725

6. Conclusions

This paper considered the problem of designing a reverse logistics supply chain. In the first instance, a mixed-integer linear programming (MILP) model was proposed in order to formalize the relation between the different elements of the reverse logistics supply chain. However, as this problem is classified as NP-hard, we proposed a solution scheme based on Binary Particle Swarm Optimization (BPSO). The algorithm guides the search for a solution by combining, on the one hand, a random solution search with an optimal solution, on the other hand, obtained by solving an associated relaxed problem. The algorithm generates two groups of swarms: (1) particles representing whether the facility is opened or closed, and (2) a set of chromosome when solving to optimality the associated relaxed problem. Computational experiments were carried out using random generated date sets. The aim of these experiments was to analyze the performance to the proposed BPSO algorithm on large sized data sets. Since approximability of solutions obtained by BPSO was very close to the optimum values in very short computation time, the proposed procedure looks to be very promising for actual implementation in real-life for the design of reverse logistics supply chains.

In addition, several lines for further research can be proposed. In the first instance, the model under study in this paper considered economic metrics (costs), especially in the objective function. Other dimensions of sustainable supply chain management (e.g., social / societal issues) can be included in the model as part of the constraints or in the objective function. Another opportunity for further research may be the inclusion of stochastic issue into the model. Finally, a trend in recent years regarding the implementation of intelligent algorithms is the hybridization of traditional meta-heuristics with other procedures. Hence, although the proposed BPSO algorithm incorporates the use of mixed-integer linear programming during the resolution stages, it could be interesting to analyze the benefits that could be achieved by incorporating other intensification or diversification procedures as additional steps in the procedure.

Acknowledgements

The work of Jairo R. Montoya-Torres was supported by a Marie Curie International Incoming Fellowship within the 7th European Community Framework Programme (project “DISRUPT”, grant No. ESR-299255).

References

[1] Alfonso-Lizarazo EH, Montoya-Torres JR, Gutiérrez-Franco E (2013) Modelling reverse logistics process in the agro-industrial sector: The case of the palm oil supply chain, Applied Mathematical Modelling, 37(23), 9652-9664

[2] Altiparmak F, Gen M, Lin L, Paksoy T (2006) A genetic algorithm approach for multiobjective optimization of supply chain networks. Computers & Industrial Engineering, 51, 197–216.

[3] Amin SH, Zhang G (2012) A proposed mathematical model for closed-loop network configuration based on product life cycle. International Journal of Advanced Manufacturing Technology, 58, 791-801.

[4] Barbosa-Povoa AP, Gomes Salema MI, Novais AQ (2007) Design and Planning of Closed Loop Supply Chains. In : Supply Chain Optimization, L. Papageorgiou & M. Georgiadis eds. NY:Wiley, p 187-218.

[5] Daskin MS, Snyder LV, Berger RT (2005) Facility Location in Supply Chain Design. In A. Langevin and D. Riopel (Eds.), Logistics Systems: Design and Optimization (pp. 39-65). New York, Springer.

[6] Dekker R, Fleischmann M, Inderfurth K, van Wasenhove L (Eds.) (2004) Reverse logistics: quantitative models for close-loop supply chains. Berlin: Springer-Verlag.

[7] Demirel NO, Gökçen H (2008) A mixed integer programming model for remanufacturing in reverse logistics environment. International Journal of Advanced Manufacturing Technology, 39, 1197-1206.

[8] Engelbrecht AP (2005). Fundamentals of Computational Swarm Intelligence. Wiley.

[9] Farahani RZ, Hekmatfar M (Eds.). (2009). Facility location: concepts, models, algorithms and case studies. Berlin: Physica-Verlag HD.

[10] Gomes ML, Barbosa-Povoa AP, Novais AQ (2011) Modelling a recovery network for WEEE: A case study in Portugal. Waste Management, 31, 1645–1660.

[11] Govindan K, Sarkis J, Palaniappan M (2013 in press) An analytic network process-based multicriteria decision making model for a reverse supply chain. International Journal of Advanced Manufacturing Technology. DOI: 10.1007/s00170-013-4949-2.

[12] Jayaraman V, Patterson RA, Rolland E (2003) The design of reverse distribution networks: Models and solution procedures. European Journal of Operational Research, 150, 128–149.

[13] Kassem S, Chen M (2013 in press) Solving reverse logistics vehicle routing problems with time windows. International Journal of Advanced Manufacturing Technology. DOI: 10.1007/s00170-012-4708-9.

[14] Kennedy J, Eberhart RC (2001) Swarm Intelligence. San Francisco: Morgan Kaufmann Publishers.

[15] Lee DH, Dong M (2008) A heuristic approach to logistics network design for end-of-lease computer products recovery. Transportation Research Part E, 44, 455-474.

[16] Lee JE, Gen M, Rhee KG (2009) Network model and optimization of reverse logistics by hybrid genetic algorithm. Computers & Industrial Engineering, 56, 951–964.

[17] Li N (2011) Research on Location of Remanufacturing Factory Based on Particle Swarm Optimization. Proceedings of International Conference on Management Science and Industrial Engineering (MSIE). Piscataway, USA: IEEE Press, pp. 1016-1019.

[18] Lu Z, Bostel N (2007) A facility location model for logistics systems including reverse flows: The case of remanufacturing activities. Computers & Operations Research, 34(2), 299–323.

[19] Min H, Ko HJ, Ko CS (2006) A genetic algorithm approach to developing the multi-echelon reverse logistics network for product returns. Omega, 34, 56–69.

[20] Mutha A, Pokharel S (2009) Strategic network design for reverse logistics and remanufacturing using new and old product modules. Computers & Industrial Engineering, 56, 334–346.

[21] Pishvaee MS, Kianfar K, Karimi B (2010) Reverse logistics network design using simulated annealing. International Journal of Advanced Manufacturing Technology, 47, 269-281.

[22] Pishvaee MS, Torabi SA (2010). A possibilistic programming approach for closed-loop supply chain network design under uncertainty. Fuzzy Sets and Systems, 161, 2668 – 2683.

[23] Rubio S, Chamorro A, Miranda F (2008) Characteristics of the research on reverse logistics (1995-2005). International Journal of Production Research, 46(4), 1099-1120.

[24] Salema MIG, Barbosa-Povoa AP, Novais AQ (2010) Simultaneous design and planning of supply chains with reverse flows: a generic modelling framework. European Journal of Operational Research, 203(2), 336–349.

[25] Sasikumar P, Kannan G, Haq AN (2010) A multi-echelon reverse logistics network design for product recovery—a case of truck tire remanufacturing. International Journal of Advanced Manufacturing Technology, 49, 1223-1234.

[26] Santibanez-Gonzalez E, Pacca Luna H (2012) A Binary Particle Swarm Optimization-based algorithm to Design a Reverse Logistics Network. Proceedings of the 2012 International Conference on Artificial Intelligence. Las Vegas, Nevada, USA. July 16-19, 2012.

[27] Zhou Y, Wang S (2008) Generic Model of Reverse Logistics Network Design. Journal of Transportation Systems Engineering and Information Technology, 8(3), 71-78.

[28] Zarei M, Mansour S, Kashan AH, Karimi B (2010) Designing a Reverse Logistics Network for End-of-Life Vehicles Recovery. Mathematical Problems in Engineering, Article ID 649028, 1-16. DOI: 10.1155/2010/649028.