Solution of the Spare Parts Joint Replenishment Problem with Quantity Discounts Using a Discrete Particle Swarm Optimization Technique

Orlando DURÁN[1], Luis PÉREZ[2]

[1] Pontificia Universidad Católica de Valparaíso,
Valparaíso, Chile
orlando.duran@ucv.cl

[2] Universidad Técnica Federico Santa María,
Valparaíso, Chile.
luis.perez@usm.cl

Abstract. Joint Replenishment of spare parts is a common practice in several industries, mainly where logistic difficulties exist, such as mining, petroleum and military missions. In addition, quantity discounts have been considered in many operations and production scenarios, as a useful practice to promote substantial savings to the actors of a supply chain. The model presented corresponds to the Joint Replenishment Problem in a system operating with quantity discounts. This work presents the definition and the solution of the optimization model using techniques based on the Particle Swarm Optimization (PSO) and a Genetic Algorithm (GA). Extensive computational experiments were performed and several performance comparisons are included. Results clearly show that the PSO algorithm achieved better repeatability and the GA presented better performance in terms of minimization capability getting lower fitness values.

Keywords: joint replenishment problem, quantity discounts, metaheuristics, particle swarm, genetic algorithm, logistics.

1. Introduction

The inventory costs occupy an important share in total cost. Replenishment, shipment consolidation/coordination policies, different inventory allocation methods and effective utilization of information are among the most important supply chain management issues today. Optimizing inventory management can make the company agile in the market, maintaining high customer service. On the other hand, the adoption of Just in Time (JIT) has caused a change in the ordering practice (Also, JIT principles have led to supply networks with relatively few direct suppliers, with each of them jointly delivering several items.) Therefore, order processing costs have been reduced through the use of long-term supply contracts, electronic ordering and joint replenishment. Consequently, the transportation costs have been strongly influenced by the joint replenishment practice. Joint replenishment inventory can lower the average cost of the inventory, enlarging the ordering item amount, which make companies more efficient in transport operations. In industry, the spare parts management plays a critical role in searching for a long term efficiency, availability and customer service. Some authors have highlighted the importance of spare management and the impact that causes to logistics. Spudic et al. [1] addressed the supply management of spare parts in military vehicles. Propadalo [2] presented an application of reliability-based spare parts management in the airspace industry. This paper presents the definition and solution of the Joint Replenishment Problem in a Consumable Spare parts system operating with suppliers offering quantity discounts, using techniques based on the Particle Swarm Optimization (PSO) and a Genetic Algorithm (GA). Section 2 presents a literature review and section 3, presents the experimental design and the problem definition. Section 4 and 5 present details about the two metaheuristics used in optimization of the model, respectively. Section 6 shows numerical examples; performance comparisons are included. Finally, conclusions are reported in section 7.

2. Literature Review

The joint replenishment problem (JRP) deals with the problem of coordinating the replenishment of a group of consumable spare parts that may be jointly ordered from a single supplier. The JRP is an NP-hard problem, as Arkin et al. [3] proved in 1989 .Goyal [4] presented a heuristics to solve the Joint Replenishment Problem. After these two works, numerous articles have dealt with the problem using different approaches. A thorough review of the literature by late

eighties may be found in Goyal and Satir [5]. As it was mentioned before, JRP is a NP hard problem, and it is unlikely that it will be solved by polynomial-time algorithms, therefore, few studies deal with exact optimisation procedures. The main assumptions mentioned in the literature to approach the JRP involve the grouping type item being direct (Direct Grouping Strategy – DGS) and indirect (Indirect Grouping Strategy – IGS). Other variant forms of the JRP problem consider the demand behaviour; therefore there are solution efforts for the constant, stochastic and dynamic demand, respectively. One of the classical heuristics for the problem is known as RAND (Kaspi and Rosenblatt [6]). Johansen and Melchiors [7] consider a periodical replenishment policy and a "can-order" policy. Regarding dynamic demand, Boctor [8] shows a procedure combining different heuristics with excellent results. Other works for special JRP cases are presented by Klein and Ventura [9], who solved the replenishment problem with discrete time. Khouja et al. [10] approached the problem with continuous reduction of the unitary costs of the items. Chan et al. [11] developed a solution model for scheduling deliveries from a supplier to different customers using the JRP. Some works point at the JRP with certain restrictions: Hoque [12], for instance, has approached the JRP with budget restrictions, defining the storage and transportation capacity; Moon and Cha [13], on the other hand, have considered resource restrictions. Bayindir et al. [14] have included variable production costs to the JRP. Other authors incorporate obsolescence aspects to the JRP (Goyal and Giri, [15]). The general JRP model assumes that the unit cost is independent of the quantity ordered. However, frequently suppliers attempt to influence their customers to place larger orders by offering quantity discounts. So far, only one work has been published on the JRP with quantity discounts (JRP+QD).Cha and Moon [16] solved the JRP+QD using simple heuristics and a modified RAND algorithm. The algorithms efficiency was proven in 1,600 randomly developed problems. One difficulty is observed in the proposed solving strategies: the run-time increases dramatically with the problem size (number of items), and often only small or moderately sized problem can be practically solved to provable optimality. In this case, the only possibility for larger instances is to trade optimality for run-time, yielding heuristic algorithms. In other words, the guarantee of finding optimal solutions is sacrificed for the sake of getting good solutions in a limited time. Therefore, the heuristics and resolution methods proposed thus far are better suited tothose problems where a little number of items are considered; this situation is not sufficiently realistic, however. There are some attempts to apply evolutionary algorithms for the solution of the JRP. We can mention Hong and Kim [17], who developed a genetic algorithm to solve the JRP problem not taking into account the restriction that the replenishment cycle has to be a multiple of the replenish cycle with the highest frequency among all items. Leung et al. [18] have presented an extension of the classical multi-customer and multi-item JRP problem; they applied a hybrid algorithm (simulated annealing and genetic algorithm) called SAGA. Chan et al. [11]applied a Genetic Algorithm to an extension of the classical multi-customer and multi-item JRP. Khouja [10] compared the performance of a genetic algorithm with JRP with the RAND performance. Olsen [19] used a genetic algorithm for the JR problem, using the direct grouping concept, which exceeds the traditional algorithm (RAND) in situations where the quotient between the fixed cost of order and the variable cost of order is considered high. In 2008, Olsen [20] explored the application of Genetic Algorithms for the JRP, where minor ordering costs depend on other items to be jointly replenished (inter-dependant costs). Dye and Hsieh [21] tested the efficiency of using another evolutionary algorithm, the Particle Swarm Optimization in a JR problem. Publications on the application of Particle Swarm Optimization in JRP with quantity discounts have not yet been issued.

3. Experimental Design

The main hypothesis of this work is that metaheuristics are able to quickly find near optimal solutions for the joint replenishment problem, with quantity discounts, when the number of spare parts is large. Based on this hypothesis, an optimization model was defined. Through experimentation and, using randomly generated data, the model was optimized using Particle Swarm Optimization and a Genetic Algorithm to respond to the following investigation question: Is it possible to optimize the inventory cost in systems with joint

replenishment using optimization techniques based on metaheuristics, in situations where quantity discounts are applied by the supplier?

The main objective of this work is: To solve an inventory optimization problem considering the joint replenishment (JRP), with suppliers applying quantity discounts. In addition, as a second objective, two evolutionary algorithms are applied to solve the problem. Finally, the values of a set of performance metrics were computed considering the results of both techniques. Comparisons and comments are also provided.

Problem Definition

The problem consists of searching for the minimum cost in a system operating with joint replenishment and quantity discounts for a large quantity of n spare parts. The problem involves a set of decision variables showing the replenishment schedule for each item. First, a model for such situation was established and the following assumptions were defined:

- Known and constant purchase period.
- Known and constant demand rate for all spare parts.
- No delay in the delivery.
- Only one supplier for each item exists.
- Finite analysis horizon.
- An item is replenished as the stock goes to zero.
- Each item involves a specific order cost.

If k_i represents the replenishment schedule of item i. As one of the assumptions indicates, the inventory level of any item goes to zero every k_i*T period, just when it is replenished again. That means that the inventory of all spare parts will go simultaneously to zero after m * T periods, where m is the least common multiple of all k_i. Then, the decision variable corresponds to a vector (of dimension n)composed by the k_i values. Here, it is assumed that T is the time step that corresponds to a unit time. Once the optimization model for joint replenishment with quantity discounts has been defined, the metaheuristics will be able to find the values of k_i assuring the lowest global cost of inventory management.

As shown in Figure 1, the stock level of item "1" (blue line) lasts for 2 periods of time (2T); therefore, k_1 equals 2. The stock level of the item "2" (red line) lasts for 3 periods of time (3T), so $k_2$is equal to 3. The stock level of item "3" (green line) lasts for 4 periods of time (4T), thus k_3 equals 4.As was commented before, the inventory levels of all spare parts will go simultaneously to zero after m * T periods, where m is the least common multiple of all k_i. In this example, m corresponds to12 periods.

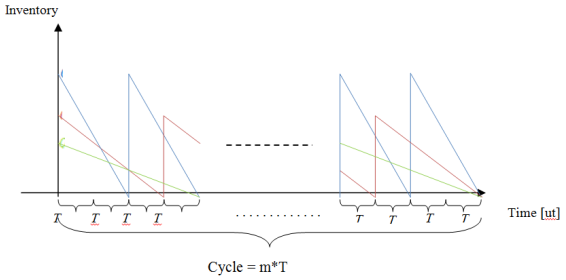

Figure 1. Purchase cycle and duration of each inventory

Optimization model

We introduce the following notations to discuss the JRP considering the quantity discounts:

C_i = Ordering cost of item i ($/order)
i = Index of item (i = 1, 2, 3…, n).
D_i = Demand rate of item i (unit/time)
P_i = Price of item i ($/unit).
Q_i = Quantity of item i(unit).
T =Basic cycle time (time).
k_i = Integer number that determines the replenishment schedule of item i (Decision Variable).
ε_i = Discount rate applied to Pi in function of Qi (−).
j = Index indicating the discount interval for Q_i (i= 1, 2, 3..., n).
m = Integer number representing the total duration of a purchasing cycle (−).
C_{fo} = Global ordering cost ($/order).
Q_{iy} = Quantity of item I triggering the y price break.
τ_i = Inventory holding cost of item I ($/unit/time).

According to the above assumptions and definitions, the total cost per unit time to be minimized is as follows:

$$TC = Cfo + \sum_{i=1}^{n} \left(P_i * D_i * m * T * \tau i * ki * T2 + 1 + Ci * mki \; um \right) \quad (1)$$

However, as such cost represents the total cost of a particular cycle (which duration depends on the values k_i) this cannot be compared to the cost of another cycle. For different k_i we obtain a

different m (representing the total duration – in periods T – of the cycle). Then, TC values cannot be compared. A way to eliminate the problem is to consider the unitary cost per period, dividing the total cost TC by the quantity of m periods that the cycle lasts, as follows:

$$tc = \frac{TC}{m}\left[\frac{um}{u\ period}\right] \quad (2)$$

$$tc = \frac{1}{m} * \left[Cfo * delta\_k + \sum_{i=1}^{n}\left(P_i * D_i * m * T * \tau i * ki * T2 + 1 + Ci * mki\right.\right. \quad (3)$$

Where P_i is the unit cost function of item i. This is a step function of T and k_i. For the all-units quantity discount, the corresponding price of the itemP_i is represented as follows:

$$P_i = P_i(1 - \varepsilon_i), for\ Q_i \leq D_i k_i T < q_{1(y+1)}$$

Where $D_i k_i T$ is the order quantity Q_i of item i, and y is the price break position. Therefore, without considering the values ofk_i, this cost can be compared and, thus, minimized. In the model, the factor, $delta\_k$, represents the fraction of non-void replenishments during the m periods. This factor may be calculated using:

$$delta\_k = \sum_{i=1}^{n}(-1)^{i+1} \sum_{\{\propto\subset\{1,\ldots,n\}:|\propto|=i\}} (lcm(k_{\propto 1}, \ldots, k_{\propto i}))^{-1}$$

$$= \sum_{i=1}^{n} 1/ki - \sum_{\{(i,j)\subset\{1,\ldots,n\}\}} 1/(lcm(k_i, k_j) + \sum_{\{(i,j,k)\subset\{1,\ldots,n\}\}} 1/ (lcm(ki,kj,kk) - \ldots + -1n+1(1(lcm(ki,..,kn)) \quad (4)$$

Where lcm represents the least common multiple of $(k_i, .., k_n)$

Figure 2 shows the total cost function for a case with two different spare parts. The horizontal axes show the variations of k_1 and k_2. It may be observed that this function has a large number of local minimum, thus representing a hard global optimization problem.

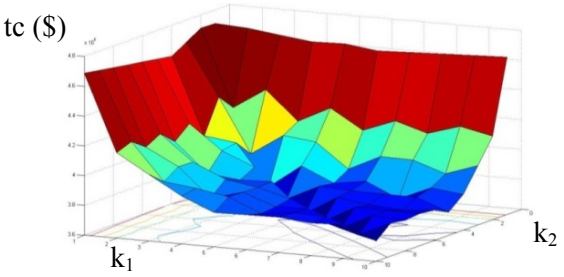

Figure 2. Total cost for two different spare parts.

4. Genetic Algorithms

Genetic Algorithms correspond to evolutionary methods that may be used to solve searching and optimization problems [22]. They are based on the genetic process of living organisms where, through generations, populations evolve according to the principles of natural selection and survival of the fittest. Through the imitation of this process, Genetic Algorithms are able to find solutions for real world problems. The power of Genetic Algorithms lies in their ability to exploit a complex, vast and, in some cases, discontinuous solution space, and their ability to successfully deal with a wide variety of problems from different areas, including those where other methods find difficulties. The evolution usually starts from a population of randomly generated individuals. Each individual in the population represents a solution to the problem and it is codified as a string of n genes. There are two basic types of representations: integer and binary. See Figure 3.

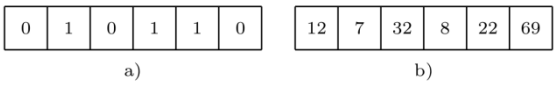

Figure 3. Chromosome representation: a) binary representation; b) integer representation.

In order to evolve to even more adapted populations, it is necessary to establish the aptitude of the individuals, i.e. it is possible to evaluate–through a fitness function- each individual and compare it with the rest of the population. Usually the fitness function represents the objective function of the optimization problem. Individuals who participate in the genetic process are previously chosen according to their fitness values. The population is processed and combined through a series of genetic operators. Through the use of some operators, it is possible to obtain new specimens which inherit some of the characteristics of their predecessors. As stated before, an important phase of genetic algorithms is the selection of the candidates who will pass to the reproduction stage. The selection process is usually probabilistic. Among the most common selection techniques are the selection by roulette, hierarchical, tournament, etc. The selection process itself does not introduce genetic variability to the population of the

subsequent generation, but increases the presence of better gifted individuals. The combination of genetic information is done by the crossover operator, while a random variation is made through the mutation operator. The selection process combined with the crossover operator allows for transmission of genetic information from one generation to the next. The algorithm is repeated until some termination criteria are met.

5. Swarm Optimization

The flocking birds have inspired the algorithm called Particle Swarm Optimization (PSO). PSO is considered an evolutionary computation (EC) method [23].It has been applied to many different areas, including manufacturing, logistics, maintenance, etc. PSO is initialized with a population of random solutions, where this initial population evolves over iterations to find optimal solutions. In PSO, each particle in population is represented by a position and has a velocity, which enables them to fly through the search space instead of dying or mutating. The modification of the particle position is carried out by using the information about its previous position and its current velocity. The algorithm has the information about the best position of each particle of the swarm (personal best) so far and the best position achieved in the swarm (group best) considering all the personal bests. These principles can be represented as follows:

$$v_i^{l+1} = wv_i^l + c_1 r_1\left(pbest_i^l - x_i^l\right) + c_2 r_2\left(gbest_g^l - x_i^l\right) \quad (5)$$

$$x_i^{l+1} = x_i^l + v_i^{l+1} \quad (6)$$

where:

- w is inertia factor;
- c_1 and c_2 are two positive values, known as cognitive and social parameters, respectively;
- $i = 1,2,\ldots,S$
- S is the swarm size;
- $r_1$andr_2 are random numbers uniformly distributed in [0, 1];
- $l = 1,2,\ldots; I$, iteration number;
- $I$is the maximum number of iterations.

The second term on the right hand side of Eq. (6) is the velocity that the particle had at the previous iteration. That velocity allows the particle to travel along the search space. The rest of the terms of the right side of the equation are used to change the particle velocity according to *pbest* and *gbest*. In general PSO executes the following series of phases: evaluation, comparison and imitation. Evaluation phase aims at measuring the potential of each particle in solving the optimization problem; the next phase, the comparison one, identifies the best particles; the following phase, that is called the imitation phase, generates new position and velocity vectors for each one of the particle in the swarm using the information of best particles found in the previous iteration. All the aforementioned phases are looped several times until meeting a given ending criterion. The main objective is finding a particle that constitutes the best solution of the target problem.

Modified PSO Algorithm

The main difference between the traditional PSO technique and the one used in this work is that the latter does not use the velocity vector in a traditional manner. Here, a mechanism adapted from Correa et al. [24] (proportional likelihood) has been especially designed to be used with discrete values in the PSO. As previously mentioned, the proposed algorithm deals with discrete values. Let*n* be the number of spare parts. Each position in the particle (vector) takes an integer value less or equal to *k* that represents the replenishment schedule for each item. *X(i,j)*is a vector that represents the position of the *i* particle. *B(i,j)*is the position in where the particle *i*had its best result so far the during the iteration process. The algorithm also records, in a vector called *G*, the best position ever attained by any particle of the swarm during the entire iteration process. The definition of the initial population is performed by generating random integer values within in the interval [l, k]. Possible solutions (particles) are encoded as fixed length discrete vectors, as is shown below:

X(i,j) = (x(i; 1); x(i; 2); ... ; x(i; n))

where x(i, j) ∈ 1, … , k;

i = 1, 2, ... ,S
j = 1, 2, ... ,n

For instance, let the maximum number of replenishment periods = 3 and the swarm size *S* = 4, a swarm could look like this:

$X(1,j)= (2\ 1\ 3\ 1\ 2\ 1)$

$X(2,j)= (1\ 1\ 2\ 1\ 2\ 3)$

$X(3,j)= (1\ 2\ 3\ 2\ 1\ 3)$

$X(4,j)= (1\ 2\ 2\ 3\ 1\ 3)$

Therefore, the first particle *X (1,j) = (2 1 3 1 2 1)*corresponds to a candidate where spare parts 1 and 5 will be replenished every 2periods, while spare parts 2,4 and 6 will be replenished every period. Finally, spare parts 5 will be replenished every two periods. After the initial generation of the population, the estimation of the fitness function is done (in this case, the estimation of the cost with each of the solutions suggested by the members of the swarm). During the imitation phasethe difference between two positions can be considered as the possible movement of a given particle. As it can be noted, the difference *(X*(i,j) - B(i,j)) represents the "distance" from the position of (*X(i,j)*) to the position of*B(i,j).*Likewise *(X(i,j)-G)*represents the "distance" from the position of (*X(i,j)*) to the position of the second term *G*.For instance, if we have the following values of *X(i,*j*) and B(i,*j*):*

$X(i,j)= (1\ 3\ 3\ 2\ 1\ 3)$

$B(i,j)= (1\ 1\ 3\ 3\ 2\ 3)$

The difference between *X(i,*j*)* and *B(i,*j*),* called *ΔXB*, represents the modifications that may be required to move the particle *X*into the position of particle *B*. The difference vector will have values equal to and different from zero. If *ΔXB≠0*, there is the possibility of changing the position with a potential enhancement in its fitness value. If the difference between a given element of *X(i,j)* and *B(i,j)* is not null, there is a potential of movement from the position *X(i,j)*to the position *B(i,j)* applied he operations described in the following. After the computation of*ΔXB*, a new vector is obtained, which records the positions where the elements *X(i,*j*)* and *B(i,*j*)* are not equal. Let*μ* the number of elements in *ΔXB* which are different from 0. And a random *generated* number*β(β∈[0,μ])* represents the number of alterations that may be applied to *X(i,j),*based on the difference between *X(i,j)* and *B(i,j).*Then, a vector called *ψ* with *β* binary numbers is generated. If the binary number is 1, the modification is made; otherwise, the change is not performed. Let us assume the following situation to illustrate this:

$X(i,j) - B(i,j) =$ | 1-1 | 3-1 | 3-3 | 2-3 | 1-2 | 3-3 |

$=$ | 0 | 2 | 0 | -1 | -1 | 0 |

A new vector *P(i,j)*= (2 4 5) is generated. Then *μ*= 3.If*β =2* and *ψ*= (0,1,1), positions 4 and 5 will be replaced in *X(i,j)* by the elements of *B(i,j)* (in the same positions). Position 2 will remain unaltered.

$X'(i,j)=$ | *1* | *3* | *3* | *3* | *2* | *3* |

This process will be repeated with the new position of *X'(i,j)* and computing the differences with respect to *G.* After that, it will be obtained the new position for each particle in the swarm. If one of the applied movements involves the same position with respect to *B(i,j)* and *G*, the modification with respect to the global best position, *G*, (the second operation in our algorithm), has the priority.

Perturbation Process

Despite the widely use of the basic PSO algorithm, it is recommended to use some sort of diversification strategy in order to escape of local minima [25]. Additionally, in order to add more exploratory capacity to the algorithm, the development of a sub-group of particle perturbation has been incorporated; this mechanism is inspired by [26]. The size of such sub-group of perturbations is approached as a fraction of the swarm size. This operates as a mutation mechanism preventing the algorithm from tending to a premature stabilization. The disturbed sub-group is chosen randomly from the complete swarm.

6. Numerical Examples

Three problems were randomly generated. The first involved 10 spare parts, the second 20 spare parts and the third problem involved 100 spare parts. Table 1 shows the data for the 10 item example. Table 2 presents data for the discount schedule for each of the 10 spare parts. In all cases, the discount schedules consider 4 intervals and four discount factors. In these examples, the population size was set to 100 particles or individuals. The crossover and mutation probabilities were set to 0,7 and 0,15, respectively for the Genetic Algorithm. The ending condition is the number of

generations or iterations set to 500. In the PSO algorithm, the size of the sub-group of perturbations was set to 5% of the population or swarm. Both the Genetic Algorithm and PSO algorithms were applied for each problem. Each technique was applied 30 times to each problem. Table 3 is shown for comparison purposes, where the following indicators or metrics are given: Amplitude ((max-min)/min); number of cases under 1% and 5% of dispersion and number of successes or the number of times that the best result was attained by the algorithms. Table 4 shows the cost values obtained for each experiment. Table 5 shows the percentage relative differences between the obtained values by both techniques. From those results it may be observed that both algorithms show an adequate repeatability, although the Amplitude obtained by the PSO algorithm is lower; the results from GA are more adequate in terms of the minimum values obtained. In addition, PSO showed a little better performance in terms of a lower dispersion. Nevertheless, both algorithms tend to present similar results when the size of the problem is growing. According to the latter, and considering the effort to obtain optimal values, we believe both algorithms are capable of obtaining good solutions in most cases. In order to illustrate the significance of the results and for comparative purposes, the costs for all the spare parts replenished in all periods, along with those obtained by the GA and PSO are shown in Figure 3.

Table 1. Parameters for the 10 items problem

item	D_i	P_i	C_i
1	27,9053	2176,61	1480,84
2	115,861	5431,84	4703,43
3	115,971	4059,53	3671,41
4	32,4078	2695,21	5050,96
5	118,014	1293,86	401,565
6	41,6613	1861,46	3195,69
7	159,83	2618,13	2358,68
8	113,195	2382,06	6013,94
9	37,5779	4155,32	2699,72
10	64,7659	2659,35	3892,24

It may be observed how sensitive the total cost is regarding the replenishment schedule. In addition, one can see how the application of this type of approach, based on metaheuristics, can help managers in deciding how and when each item should be replenished in order to ensure the minimum costs.

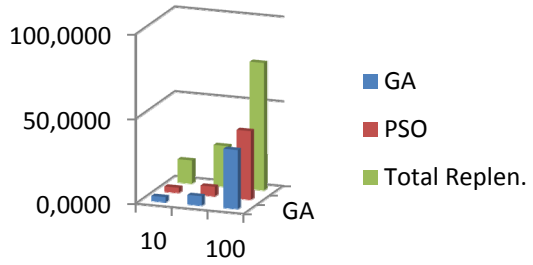

Figure 3. Comparison of the best results obtained by GA and PSO, along with the case where all the spare parts are replenished in every period

Table 2. Discount schedules

	□$_i$	Q_i		□$_i$	Q_i
item 1	0	Q_i<7	item 6	0	Q_i<1
	0,03	7<Q_i<14		0,025	2<Q_i<4
	0,05	14<Q_i<20		0,05	5<Q_i<12
	0,08	21<Q_i		0,08	12<Q_i
item 2	0	Q_i<100	item 7	0	Q_i<49
	0,03	100<Q_i<200		0,025	49<Qi<100
	0,05	200<Q_i<298		0,05	100<Q_i<147
	0,08	298<Q_i		0,08	148<Q_i
item 3	0	Q_i<97	item 8	0	Q_i<88
	0,03	97<Q_i<194		0,025	88<Q_i<178
	0,05	195<Q_i<290		0,05	178<Q_i<265
	0,08	290<Q_i		0,08	266<Q_i
item 4	0	Q_i<12	item 9	0	Q_i<35
	0,03	12<Q_i<25		0,025	35<Q_i<71
	0,05	25<Q_i<36		0,05	72<Q_i<105
	0,08	36<Q_i		0,08	105<Q_i
item 5	0	Q_i<47	item 10	0	Q_i<44
	0,03	47<Q_i<96		0,025	44<Q_i<88
	0,05	97<Q_i<142		0,05	88<Q_i<135
	0,08	142<Q_i		0,08	135<Q_i

Table3. Comparative results obtained
- both techniques -

Technique	PSO			GA		
Problems	10	20	100	10	20	100
Amplitude (%) (max-min)/min	2,01	1,59	8,72	4,85	4,25	8,62
% of cases < 1% of dispersion	87	46	3	87	50	6
% of cases < 5% of dispersion	100	100	43	97	100	93
Number of successes	8 of 30	1 of 30	1 of 30	18 of 30	3 of 30	1 of 30

Table 4. Costs obtained for both techniques ($)

	10 items		20 items		100 items	
	PSO	**GA**	**PSO**	**GA**	**PSO**	**GA**
MIN	3.098.400	3.042.517	5.610.813	5.571.515	38.493.000	34.019.000
AVERAGE	3.112.410	3.055.248	5.665.196	5.634.189	40.521.466	34.850.367
MAX	3.160.800	3.189.978	5.700.279	5.808.135	41.849.000	41.849.000

Table 5. Percentage relative differences obtained for both techniques (%)

	10 items		20 items		100 items	
	PSO	**GA**	**PSO**	**GA**	**PSO**	**GA**
MIN	1,84	0,00	0,71	0,00	13,15	0,00
AVERAGE	1,87	0,00	0,55	0,00	16,27	0,00
MAX	0,00	0,92	0,00	1,89	0,00	0,00

7. Conclusions

This work presented the application of two metaheuristics to optimize the Joint Replenishment Problem with Quantity Discounts. The Joint Replenishment Problem with Quantity Discounts combines the replenishment of more than one item each time that replenishment occurs. In addition, that situation was combined with the existence of discounts in the unitary costs according to the acquired quantity. There is only one work in the literature presenting a model for this situation, but it is limited to a small quantity of spare parts. In addition, a Genetic Algorithm and a Swarm Intelligence algorithm were applied to a set of simulated problems. Both techniques enabled feasible solutions of those complex problems. Specifically, the PSO algorithm outperformed the GA in terms of lower dispersion and better repeatability. On the other hand, GA got better cost values than PSO in all the studied cases. We believe that these techniques offer good and practical solutions to the challenges faced by managers that purchase spare parts from a series of suppliers which offer quantity discounts.

REFERENCES

1. SPUDIC, R., B. IVANKOVIC, V. KOVACEVIC, **Testing The Logistics Model Of Supplying Military Vehicles With Spare Parts**. PROMET-Traffic & Transportation, vol. 19(4), 2010, pp. 233-236.
2. PROPADALO, S., D. BEKAVAC, Z. JAKSIC, **Spare Modules Management Optimization of Airspace Surveillance System**. PROMET-Traffic & Transportation, vol. 24(4), 2012, pp. 233-236.
3. ARKIN, D. J., R. ROUNDY, **Computational Complexity of Uncapacitated Multi-echelon Production Planning Problems**, Operation Research Letters, vol. 8, 1989, pp. 61–66.
4. GOYAL, S., **Determination of Optimum Packaging Frequency of Items Jointly**

Replenished, Management Science, vol. 23, 1974, pp. 436-443.

5. GOYAL, S., A. SATIR, **Joint Replenishment Inventory Control: Deterministic and Stochastic Models**. European Journal of Operational Research, vol. 38, 1989, pp. 2-13.
6. KASPI, M., M. ROSENBLATT, **On the Economic Ordering Quantity for Jointly Replenished Items**. International Journal of Production Research, vol. 29, 1991, pp. 107-114.
7. JOHANSEN, S. G., P. MELCHIORS, **Can-order Policy for the Periodic Review Joint Replenishment Problem**, Journal of the Operational Research Society, vol. 54, 2003, pp. 283-290.
8. BOCTOR, F., G. LAPORTE, J. RENAUD, **Models and Algorithms for the Dynamic Demand Joint Replenishment Problem**, International Journal of Production Research, vol. 42, 2004, pp. 2667-2678.
9. KLEIN, C. M., J. A. VENTURA, **An Optimal Method for a Deterministic Joint Replenishment Inventory Policy in Discrete Time**, Journal of the Operational Research Society, vol. 46(5), 1995, pp. 649-657.
10. KHOUJA, M., Z. MICHALEWICZ, S. SATOSKAR, **A Comparison Between Genetic Algorithms and the RAND Method for Solving the Joint Replenishment Problem**. Production Planning & Control, vol. 11, 2000, pp. 556–564.
11. CHAN, C. K., B. K. S. CHEUNG, A. LANGEVIN, **Solving the Multi-buyer Joint Replenishment Problem with a Modified Genetic Algorithm**. Transportation Research Part B-Methodological, vol. 37(3), 2003, pp. 291-299.
12. HOQUE, M. A., **An Optimal Solution Technique for the Joint Replenishment Problem with Storage and Transport Capacities and Budget Constraints**, European Journal of Operational Research, vol. 175, 2006, pp. 1033-1042.
13. MOON, I. K., B. C. CHA, **The Joint Replenishment Problem with Resource Restrictions**, European Journal of Operational Research, vol. 173, 2006, pp. 190-198.
14. BAYINDIR, Z. P., S. I. BIRBIL, J. B. G. FRENK, **The Joint Replenishment Problem with Variable Production Costs**, European Journal of Operational Research, vol. 175, 2006, pp. 622-640.
15. GOYAL, S. K., B. C. GIRI, **Recent Trends in Modelling of Deteriorating Inventory**. European Journal of Operational Research, vol. 134(1), 2001, pp. 1-16.
16. CHA, B., I. MOON, **The Joint Replenishment Problem with Quantity Discounts**, OR Spectrum, vol. 27, 2005, pp. 569-581.
17. HONG, S. P., Y.-H. KIM. **A Genetic Algorithm for Joint Replenishment based on the Exact Inventory Cost**. Computers & Operations Research, vol. 36, 2009. pp. 167-175.
18. LEUNG, T. W., C. K. CHAN, M. D. TROUTT, **A Mixed Simulated Annealing-genetic Algorithm Approach to the Multi-buyer Multi-item Joint Replenishment Problem: Advantages of Meta-heuristic**, Journal of Industrial and Management Optimization, vol. 4(1), 2008, pp. 53-66.
19. OLSEN, A., **An Evolutionary Algorithm to Solve the Joint Replenishment Problem using Direct Grouping**. Computers & Industrial Engineering, vol. 48(2), 2005, pp. 223-235.
20. OLSEN, A., **Inventory Replenishment with Interdependent Ordering Costs: An Evolutionary Algorithm Solution**. International Journal of Production Economics, vol. 113(1), 2008, pp. 359-369.
21. DYE, C. Y., T. P. HSIEH. **A Particle Swarm Optimization for Solving Joint Pricing and Lot-sizing Problem with Fluctuating Demand and Unit Purchasing Cost**. Computers & Mathematics with Applications, vol. 60(7), 2010, pp. 1895-1907.
22. COELLO, C. **Introducción a la computación evolutiva**. Notes CINVESTAV – IPN, Departamento de Computación, México. 2008.

23. EBERHART, R. C., J. KENNEDY, **A New Optimizer using Particle Swarm Theory**, Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan. 1995, pp. 39-43.

24. CORREA, E. S., A. FREITAS, C. G. JOHNSON, **A New Discrete Particle Swarm Algorithm Applied to Attribute Selection in a Bioinformatics Data Set**. In Proceedings of the Genetic and Evolutionary Computation Conference - Seattle, WA, 2006, pp.35-42.

25. CABRERA, G., S. D. RONCAGLIOLO, J. P. RIQUELME, C. CUBILLOS, R. SOTO, **A Hybrid Particle Swarm Optimization - Simulated Annealing Algorithm for the Probabilistic Travelling Salesman Problem**, Studies in Informatics and Control, vol. 21(1), pp. 49-58, 2012.

26. LEE, S., H. PARK, M. JEON, **Binary Particle Swarm Optimization with Bit Change Mutation**. IEICE Transactions .on Fundamentals of Electronics, Communications and Computer Sciences, vol. 90(10), 2007, pp. 2253-2256.