International Journal of Hybrid Information Technology
Vol.8, No. 5 (2015), pp. 259-266
http://dx.doi.org/10.14257/ijhit.2015.8.5.28

Differential Evolution Algorithm for Constraint Joint Replenishment Problem with Indirect Grouping Strategy

Li Chengyan, Gao Jun, Zhang Tianwei and Wang Xiaotian

School of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China
E-mail:chengyan@hrbust.edu.cn

Abstract

The joint replenishment problem with deterministic resource restriction is considered. We present a differential evolution (DE) algorithm that uses indirect grouping strategy to solve constrained joint replenishment. The procedure and structure of the DE algorithm is proposed. Extensive computational experiments are performed to compare the performances of the DE algorithm with results of genetic algorithm (GA) and heuristic algorithm CRAND. The experimental results indicate that the DE algorithm performs relative to CRAND and superior to GA.

Keywords: *Inventory control, Joint replenishment problem, Differential evolution, Genetic algorithm*

1. Introduction

For the past few decades, the joint replenishment problem has received much attention since the early work of Shu [1]. Cost can be saved when replenishment of several items are coordinated in a multi-item inventory system. The constraint joint replenishment problem (which is abbreviated as CJRP for the rest of the paper) is the multi-item inventory problem of coordinating the replenishment of a group of items that may be jointly ordered from a single supplier under resource restrictions, for example, budget, storage, transportation capacity, etc. Replenishment of a group of item takes place after a fixed interval of time, called the basic cycle time. Time between two consecutive orders of an item is assumed to be an integer multiple of the basic cycle time. The objective of the JRP is to minimize the total costs incurred per unit time.

The total cost is composed of two parts:

- The ordering cost. This is the cost of preparing and receiving the order and the transportation cost.
- The holding cost. This is the cost of holding inventory which includes the cost of capital tied up in inventory, taxes and insurance.

In this situation, the ordering cost has two components-- a major common ordering cost S incurred whenever an order is placed and a minor ordering cost s_i incurred if item i is included in the order. It is assumed in the JRP that for each item i, its demand rate D_i is fixed, and each unit incurs a holding cost h_i per unit time. The approaches to the JRP can be generally classified into two types of strategies: a direct grouping strategy (DGS); and an indirect grouping strategy (IGS). Under DGS, items are partitioned into a predetermined number of sets and the items within each set are jointly replenished. Under IGS, a replenishment is made at regular time intervals (every T units of time) and each item has a replenishment quantity sufficient to last for exactly an integer multiple $k_i, i=1,2,\ldots,n$ of T. the decision variables in the IGS are T and k_i. The objective is to find a proper combination (T,k_i) so that total cost is as low as possible.

ISSN: 1738-9968 IJHIT

JRP has been proven to be non-deterministic polynomial hard (NP-hard) problem [2]. The literatures for JRP include the heuristics for the classic JRP under constant demand and special approaches for the JRP including meta-heuristics. Goyal [3] developed a heuristic algorithm using the Lagrangian multiplier for the JRP with one resource restriction. Van Eijs [4] has derived another algorithm that improves Goyal's algorithm. Kaspi and Rosenblatt [5] proposed RAND algorithm based on trying several values of basic cycle time between a minimum and a maximum value. Olsen [6] solved the JRP with direct grouping strategy using genetic algorithm (GA). Khouja *et. al.*, [7] applied GA approach to the JRP and compared the performance of GA algorithm to heuristic algorithm. Moon and Cha [8] introduced heuristic algorithm CRAND and the GA approach to the CJRP; they mentioned that a major advantage of the GA algorithm is its ability to handle constrained problems.

The Differential Evolution (DE) algorithm is one of the latest evolutionary optimization methods proposed by Storn and Price [9] for complex continuous non-linear functions. DE is a stochastic population-based optimization method. DE uses mutation, crossover, and selection operators at each generation to move its population toward the global optimum. DE was initially developed for solving the Chebyshev polynomial fitting problem because the problem was very difficult to solve by using other algorithms. Over the past ten years, DE has been successfully applied to resolve optimization problems, which exhibits remarkable performance in optimizing a wide variety of multi-dimensional and multi-modal objective functions in terms of final accuracy, convergence speed and robustness [10]. Many research indicated that the DE algorithm can solve the problem more effectively [11]. Kazemipoor et al. [12] considered the multiskilled project portfolio scheduling problem and presented an efficient metaheuristic algorithm based on DE, the comparison between the results of DE and Tabu search confirms the effectiveness of the DE algorithm. Wang et al. developed an approach based on DE to find a close to optimum for the basic JRP [13]. Das and Suganthan [14] surveyed the state-of-the-art of the differential evolution and its application. Lampinen [15] considered the constraint approach of DE algorithm. The DE algorithm is suitable for solving the CJRP because of its simple structure, easy implementation, robustness, and speediness [16].

The aim of this paper is to develop a practical DE algorithm for CJRP based on the literature and make comparison to other heuristic algorithm. The rest of this paper is organized as follows. Section 2 introduces the mathematical model of CJRP. Sections 3 develops the DE algorithm and shows that how the DE algorithm can be used to handle the CJRP. Section 4 illustrates the procedure of proposed algorithm with a numerical example. Section 5 gives the details computational experiments to compare the performance of DE, GA and CRAND algorithms. Section 6 summarize the conclusions of the present work and provide directions for future research.

2. Mathematical Model of Constraint Joint Replenishment Problem

Some common assumptions usually made for the constrained joint replenishment problem:

- The demand rate of each item is deterministic and constant.
- The unit holding cost of each item is known and constant.
- The major ordering cost incurred for an order is known and constant.
- The minor ordering cost incurred for a specific ordered item is known and constant.
- No quantity discount.
- No shortage is allowed.
- Stock replenishment is complete when it occurs.
- The budget constraint on the amount of an order is known and constant.

The following notation is defined:

I index of item, i=1,2,...,n
D_i demand rate of item i
S major ordering cost
s_i minor ordering cost of item i
h_i inventory holding cost of item i, per unit per unit time
b_i unit cost of item i
B limit on capital that can be invested
T basic cycle time (decision variable)
k_i integer number that decides the replenishment schedule of item i (decision variable)
Q_i ordering quantity of each replenishment cycle of item i
TC total annual holding and ordering cost for all items

Using the IGS approach, the cycle time for every product is an integer multiple k_i of a basic cycle T. Thus the cycle time for item i is:

$$T_i = k_i T \tag{1}$$

and the order quantity for product i is:

$$Q_i = TD_i = Tk_i D_i \tag{2}$$

Thus, the total holding cost of n items is:

$$C_h = \frac{1}{2}\sum_{i=1}^{n}(k_i T)D_i h_i \tag{3}$$

And the total ordering cost of n items is:

$$C_o = \frac{S + \sum_{i=1}^{n}(s_i / k_i)}{T} \tag{4}$$

The mathematical model for the JRP resource constraint as follows:

$$\min TC(T, k_i's) \tag{5}$$

s.t.

$$TC(T, k_i's) = C_o + C_h = \frac{S + \sum_{i=1}^{n}(s_i / k_i)}{T} + \frac{1}{2}\sum_{i=1}^{n}(k_i T)D_i h_i \tag{6}$$

$$\sum_{i=1}^{n} D_i k_i T b_i \le B \tag{7}$$

$$k_i \in Z^+, T \in R^+, i = 1, 2, \ldots n \tag{8}$$

The objective function, i.e., Eq. (6) contains the ordering costs (major ordering cost and minor ordering cost) and inventory holding cost. The inequalities in Eq. (7) require that the total value of items in a basic replenishment cycle must be less than the capital that can be invested. Thus, Eq. (7) is the capacity constraints for a feasible solution. Eq. (8) indicates that k_i's are positive integer number and T is positive real number.

3. Differential Evolution Algorithm

In this section, we present a differential evolution algorithm approach for the CJRP model. Introduced by many researches, DE algorithm is demonstrated to be an effective and robust method by applying in the optimization of some well-known non-linear, non-differentiable and non convex functions and it has been widely used in various areas for more than ten years. DE algorithm, differing from conventional evolutionary optimization methods, such as genetic algorithm (GA), relies on the mutation operations as the main operator. The DE algorithm introduces a novel mutation operation which is simple and effective. The mutation operation is based on the differences of randomly sampled pairs of solution in the population.

Furthermore, the fitness of an offspring is one-to-one competed with that of the corresponding parent in DE algorithm.

3.1. Representation and Initialization

In the CJRP model, the basic cycle T and n integer k_i's have to be decided for solving the problem. It is very important to represent a solution properly. Moon and Cha [8] have proven that the optimal basic cycle T is determined for a given $(k_1,k_2,\ldots,k_n)$ and the total relevant cost TC is computed for a given $(T,k_1,k_2,\ldots,k_n)$. So, in our study, we use n random number representation for n integer (k_i's), because it is very easy to decode our chromosome to a feasible solution.

The initial population is created by assigning random values of the decision variable. Each individual is generated by Eq. (9).

$$x_{ij} = rand(0,1), i = 1,2,\ldots,POPSIZE,\ j = 1,2,\ldots,D \tag{9}$$

Where $POPSIZE$ is the number of individuals; D is the dimension of each individuals; $rand(0,1)$ is a function which generates a uniform distribution rand number in range [0,1].

Our decoding process for each gene of our chromosome is as follows.

$$k_i = \text{int}[(k_i^{UB} - k_i^{LB} + 1)x_i] + k_i^{LB}, i = 1,2,\ldots,D \tag{10}$$

Where int[b] is the function which finds the integer number less than x; k_i^{LB} and k_i^{UB} is the lower and upper bound of k_i, respectively, and can be defined from the following equations.

$$k_i^{LB}(k_i^{LB} - 1) \le \frac{2s_i}{D_i h_i T_{\max}^2} \le k_i^{LB}(k_i^{LB} + 1) \tag{11}$$

$$k_i^{UB}(k_i^{UB} - 1) \le \frac{2s_i}{D_i h_i T_{\max}^2} \le k_i^{UB}(k_i^{UB} + 1) \tag{12}$$

.

$T_{\max}$ and $T_{\min}$ were defined by Eqs. (13) and (14).

$$T_{\max} = \sqrt{2(S + \sum_{i=1}^{n} s_i) / \sum_{i=1}^{n} D_i h_i} \tag{13}$$

$$T_{\min} = \min \sqrt{2s_i / D_i h_i} \text{ for } i \tag{14}$$

3.2. Mutation

For each target individual $X_i=\{x_{i1},x_{i2},\ldots,x_{iD}\}$, $i=1,2,\ldots,POPSIZE$, a mutant new individual V_i is generated according to

$$V_i = X_{r1} + F(X_{r2} - X_{r3}), r_1 \ne r_2 \ne r_3 \ne i \tag{15}$$

With randomly chosen integer indexes; $F \in [0,2]$ is a real number called mutation factor used to control the amplification of the differential variation.

3.3. Crossover

To complement the differential mutation search strategy, DE employs uniform crossover, which also known as binomial method, to enhance the potential diversity of the population. The crossover operator implements a discrete recombination of the trial individual V_i and the parent individual X_i to produce the offspring X_i^{new}.

The trial vector will be found using the following rules:

$$x_{ij}^{new} = \begin{cases} v_{ij} & \text{if } rand_j(0,1) \le CR \text{ or } j = rnb(i) \\ x_{ij} & \text{otherwise } j = 1,2,\ldots,D \end{cases} \tag{16}$$

Where D is the dimension of X_i; x_{ij} refers to the jth element of the individual; v_{ij} is similarly defined; $rand_j(0,1)$ is the jth evaluation of a uniform random number generator between [0,1]; $rnb(i)$ is a randomly chosen integer in the set $\{1,2,\ldots,D\}$ which ensures the trail vector gets at least one parameter from the mutated vector; $CR \in [0,1]$ is a crossover constant.

3.4. Selection

The selection operator is to determine whether the target (parent) or the new vector (offspring) survives to the next generation. If a new vector, $X_i^{new}, i=1,2,\ldots,POPSIZE$, has a smaller evaluation function value (total cost) than its garget vector, X_i , it is copied to the next generation; otherwise, it is the target vector that passes to the next generation. The selection process can be expressed as:

$$X_i = \begin{cases} X_i^{new} & \text{if } fitness(X_i^{new}) < fitness(X_i) \\ X_i & \text{otherwise} \end{cases} \quad (17)$$

Thus the next generation of population either gets better in terms of the fitness function or remains constants.

3.5. Elitist Procedure

The elitist procedure is to keep the best individual or replace the worst individual in population. In generation G of iteration, find the best individual, $X_{\text{best},G}$, and worst individual, $X_{\text{worst},G}$ in terms of the fitness function respectively. In the next generation, G+1, the elitist procedure is processed as follows:

$$\begin{aligned} &X_{best,G+1} = X_{best,G} \text{ if } fitness(X_{best,G}) < fitness(X_{best,G+1}) \\ &X_{worst,G+1} = X_{best,G} \text{ otherwise.} \end{aligned} \quad (18)$$

3.6. Stop Criterion

The termination condition is to stop if no improvement of fitness function is made in 50 generations.

4. Numerical Example

A numerical example will be employed to compare the proposed DE algorithm and GA algorithm; we use the numerical example of Moon and Cha [8]. The data for this example are given in table 1. We also assume S=200 and B=25000.

Table 1. Data for the Example

Item i	1	2	3	4	5	6
D_i	10000	5000	3000	1000	600	200
s_i	45	46	47	44	45	47
h_i	1	1	1	1	1	1
b_i	6.25	6.25	6.25	6.25	6.25	6.25

For this example, the population size of 10 is used and the CR=0.5 and F=0.5, respectively. Both representation and decoding solution of our best DE individual are shown in Table 2. The compared GA results made by Moon and Cha [8] are also shown.

Table 2. Representation and Decoding Solution of the Best GA Chromosome and DE Individual

Algorithm	Item	1	2	3	4	5	6
GA	Best GA chromosome	0.446	0.463	0.092	0.426	0.251	0.381
	k_i^{LB}	1	1	1	1	2	3
	k_i^{UB}	1	2	2	3	4	7
	Decoding solution(k_i)	1	1	1	2	2	4
	Optimal T	0.1818					
	TC	4168.4					
DE	Best DE individual	0.003	0.296	0.303	0.396	0.155	0.355
	k_i^{LB}	1	1	1	1	2	3
	k_i^{UB}	1	2	2	3	4	7
	Decoding solution(k_i)	1	1	1	2	2	4
	Optimal T	0.1818					
	TC	4168.4					

It shows in Table 2 that both GA and DE obtain the same optimal solution for this numerical example.

Figure 1 shows the structure of our differential evolution algorithm for the numerical example.

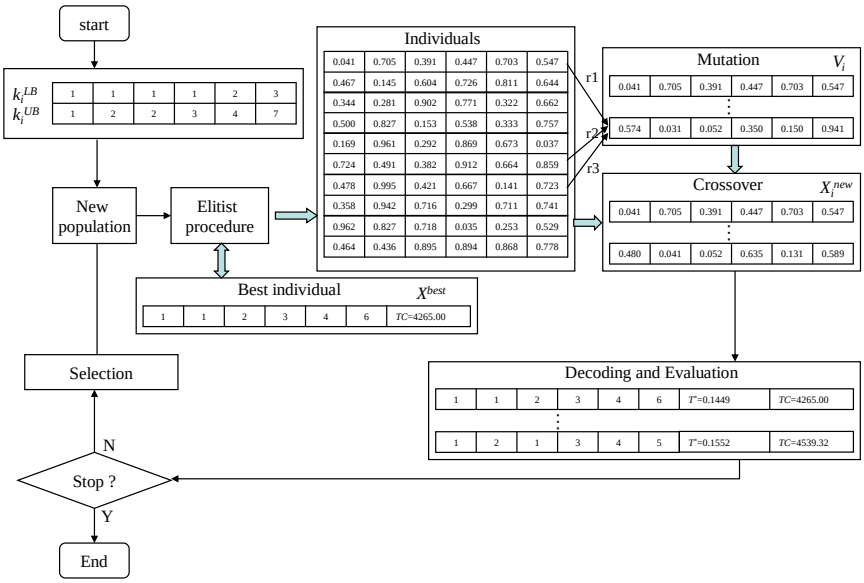

Figure 1. The Structure of the Proposed Differential Evolution Algorithm

5. Computational Experiments

In this section, we compare the performance of three algorithms, CRAND, GA and DE, for 1600 randomly generated CJRPs. The parameter values are all generated from a uniform distribution, D_i~U[100,100000], s_i~U[0.5,5.0] and h_i~U[0.2,3.0], respectively. Every b_i is considered as 1, and B is randomly generated, B~U[n×200, n×800]. Four different values of the number of items, n=10, 20, 30 and 50, and four different values of the major ordering cost, S=5, 10, 15 and 20 are considered. This results in 16 combinations of n and S, and for each combination, 100 problems with random parameter values are generated and solved. In the DE, the population size of 100 is used for solving the small size problems (n=10, 20 and 30). The population size of 200 is used for the large size problems (n=50). A summary of computational results is shown in Table 3.

Table 3. Comparison of Three Algorithms (% improvement)

n	S	Best solution problems			CRAND better than GA		CRAND better than DE	
		CRAND	GA	DE	Maximum	Average	Maximum	Average
10	5	100	99	98	0.0061	0.0001	0.0050	0.0001
	10	100	100	99	0.0000	0.0000	0.0001	0.0000
	15	100	99	100	0.0026	0.0000	0.0058	0.0000
	20	98	99	99	0.0176	-0.0011	0.0011	0.0000
20	5	100	86	92	1.5142	0.0567	1.1420	0.0028
	10	99	90	97	0.3290	0.0098	0.0023	0.0087
	15	100	83	99	0.6872	0.0156	0.0004	0.0000
	20	100	95	98	0.0997	0.0014	0.0017	0.0000
30	5	100	57	99	2.3667	0.1449	0.0006	0.0000
	10	99	70	97	1.0502	0.0298	0.0012	0.0000
	15	100	73	98	0.3539	0.0148	0.0020	0.0000
	20	100	82	99	0.4455	0.0066	0.0002	0.0000
50	5	96	51	93	1.2439	0.0921	2.1237	0.0065
	10	100	67	96	0.9690	0.0542	0.0490	0.0000
	15	95	66	90	1.0124	0.0430	0.0050	0.0000
	20	98	81	88	0.0982	0.0026	0.2301	0.0036
Maximum					2.3667		2.1237	
Average		99	81	96		0.0294		0.0014

As show in table 3, the DE performs very well relative to the CRAND. We can confirm that the DE is superior to GA, especially for large value of n (n=30 and 50). In addition, the DE can be easily expanded to the problems with several constraints like GA.

6. Conclusion

This paper focuses on the development of the efficient algorithm for solving the constrained joint replenishment problem. The DE algorithm is introduced to handle the constraints in JRP. Despite the DE is superior to the GA, the DE is more suitable for solving constrained JRP than CRAND for its extension ability. Future research could be the incorporation of uncertainty issue in constrained JRP, for example demand and budget.

Acknowledgements

This work was supported by grant No.12541142 from the Research Program of the Education Department of Heilongjiang Province, China.

References

[1] F. T. Shu, "Economic ordering frequency for two items jointly replenished", Manage Sci. vol. 6, no. 17, **(1971)**, pp. 406-410.
[2] E. Arkin, D. Joneja and R. Roundy, "Computational complexity of incapacitated multi-echelon production planning problems", Oper. Res Lett.vol. 2, no. 8, **(1989)**, pp. 61-66.
[3] S. Goyal, "Analysis of joint replenishment inventory systems with resource restriction", Oper Res Q, vol. 1, no. 26, **(1975)**, pp. 197-203.
[4] M. V. Eijs, "A note on the joint replenishment problem under constant demand", Oper Res Soc, 2, vol. 44, **(1993)**, pp. 185-191.
[5] M. Kaspi and M. Rosenblatt, "On the economic ordering quantity for jointly replenished items", Int. Prod Res, vol. 1, no. 29, **(1991)**. pp. 107-114.
[6] A. L. Olsen, "An evolutionary algorithm to solve the joint replenishment problem using direct grouping", Comput. Ind. Eng.vol. 2, no. 48, **(2005)**, pp. 223-235.
[7] M. Khouja, Z. Michaliwicz and S. Satoskar, "A comparison between genetic algorithms and the RAND method for solving the joint replenishment problem", Prod Plan Control.vol. 6, no. 11, **(2000)**, pp. 556-564.
[8] I. K. Moon and B. C. Cha, "The joint replenishment problem with resource restriction", Eur J Oper Re. vol. 1, no. 173, **(2006)**, pp. 190-198.
[9] R. Storn and K. Price, "Differential evolution- a simple and efficient heuristic for global optimization over continuous spaces", Glob Optim. vol. 4, no. 11, **(1997)**, pp. 341-359.
[10] A. Jha, K. Somani, M. K. Tiwari, F. T. S. Chan, K. J. Fernandes, "Minimizing transportation cost of a joint inventory location model using modified adaptive differential evolution algorithm", Int. Adv Manuf Technol., vol. 1, no. 60, **(2012)**, pp. 329–341.
[11] A. Musrrat, P. Millie and A. Ajith, "A simplex differential evolution algorithm", development and applications, T I Meas Control. vol. 6, no. 34, **(2012)**, pp. 691–704.
[12] H. Kazemipoor, R. T. Moghaddam, P. S. Shahrezaei and A. Azaron, "A differential evolution algorithm to solve multi-skilled project portfolio scheduling problems", Int J Adv Manuf Technol. vol. 5-8, no. 64, **(2013)**, pp. 1099-1111.
[13] L. Wang, J. He and Y. R. Zeng, "A differential evolution algorithm for joint replenishment problem using direct grouping and its application", Expert Syst. vol. 5, no. 29, **(2012)**, pp.429-441.
[14] S. Das and P. N. Suganthan, "Differential evolution", A survey of the state-of-the-art, IEEE T Evolut Comput. vol. 15, **(2011)**, pp. 4-31.
[15] J. Lampinen, "A constraint handling approach for the differential evolution algorithm", Proceedings of the 2002 Congress on Evolutionary Computation, **(2002)** May 12 -17, Honolulu, HI, pp.1468-1473.
[16] L. Wang, C. X. Dun, C. G. Lee, Q. L. Fu and Y. R. Zeng, "Model and algorithm for fuzzy joint replenishment and delivery scheduling without explicit membership function", Int J Adv Manuf Technol., vol. 9-12, no. 66, **(2013)**, pp. 1907-1920.

