An Application of Genetic Algorithms for General Dynamic Lotsizing Problems

Xie Jinxing

Tsinghua University, China

Abstract: This paper presents an application of genetic algorithms for dynamic lotsizing problems, including the implementation methodology and the testing results of the algorithms. Currently, most of the existing studies for dynamic lotsizing problems concentrate on heuristic lot-sizing techniques which only consider some simple production structures and/or simple external demands strctures. In this paper, the general dynamic lot-sizing problems are considered, which are characterized by the fact that each stage may have several predecessor and/or successor stages, all the items can have independent requirements, and/or all the cost parameters can be time-varying. A genetic algorithm for the problems is introduced, which attempt to heuristically optimize under all the conditions simultaneously. As to my knowledge, this genetic algorithm is the first one capable of solving such general dynamic lotsizing problems. In order to apply genetic algorithm, a coding scheme for lotsize plan/schedule is given and a feasibility routine is presented. In computational experiments, this genetic algorithm performed extremely well. It is concluded that the genetic algorithm is efficient and effective for dynamic lotsizing problems.

Keywords: *Production planning/scheduling, lotsizing problems, genetic algorithms*

1. Introduction

The wide-spread and popular use of material requirements planning (MRP) systems in industry has resulted in increased interest in the topic of decision-making in multi-stage production systems. As firms have incorporated MRP concepts into their production planning and distribution system, the multi-item multi-stage dynamic lot-sizing problem has become a problem of prime importance, because the lot-sizing procedures used by currently available MRP systems are quite limited in their ability to coordinate production plans of various stages of the manufacturing process and various patterns of external demands cost parameters for items.

Many researchers have studied the problem and a lot of lotsizing procedures have been presented. For example, Gupta and Yeung [7], Coleman [5], Aggarwal and Park [4], Lambrechet et al. [9], Rosling [12], Afentakis [1], Afentakis et al. [3], Heinrish and Schn. [8], Afentakis and Gavish[2] are some review papers or recent advances. Since most of the problems are NP-hard (Maes et al.[10]), most of the existing algorithms use heuristic techniques to solve the problem approximately. Currently, most of the existing studies for dynamic lotsizing problems concentrate on heuristic lot-sizing techniques which only consider some simple production structures and/or simple external demands strctures. For example, the dynamic lotsizing problems in general production structures are seldom considered. This is an obstacle to the real application of these lotsizing techniques in production planning and scheduling.

Recently, genetic algorithms are deeply studied and widely used in combinatorial optimization problems and a lot of successful application instances and good results are reported (Goldberg[6], Reeves[11], etc.). But as to my knowledge, the application of genetic algorithms for multi-item multi-stage dynamic lotsizing problems is not suggested.

In this paper, the general lot-sizing problems are considered, which are characterized by the fact that each stage may have several predecessor and/or successor stages, all the items can have independent requirements, and/or all the parameters can be time-varying. A genetic algorithm for the problems is introduced, which attempt to heuristically

Genetic Algorithms in Engineering Systems: Innovations and Applications
12-14 September 1995, Conference Publication No. 414, © IEE, 1995

optimize under all the conditions simultaneously. In order to apply genetic algorithm, a coding scheme for lotsize plan/schedule is given and a feasibility routine is presented. In computational experiments, this genetic algorithm performed extremely well. It is concluded that the genetic algorithm is efficient and effective for dynamic lotsizing problems.

2. Mathematical formulation of the problem

In this paper, we consider the following multi-item multi-level dynamic lotsizing problem: Given the external demand for N items over a time horizon of T periods, find the solution which minimizes total setup, production and holding cost, satisfying the following conditions:

- The product structure can be presented as an acyclic directed network, where every node is a item and the arc illustrates the assembly or distribution relation between items, and the weight of an arc is the quantity relation between the two end nodes of the arc. In general production systems, each node can have more than one immediate predecessors and/or more than one immediate successors. (Usually we assume that all the nodes be numbered as satisfying the condition that the number of each predecessor is greater than that of each of its successes.)

- The production capacity is assumed enough, so no capacity constraints considered. Moreover, the production of items is instantaneous, despite of the production quantity.

- The backlogging of end items is not allowed.

- The lead times are assumed to be constant and, without loss of generality, are assumed to be zero.

Mathematically, this problem can be stated as follows :

Problem GDLP

$$Min\ COST(Y,X,I) = \sum_{i=1}^{N}\sum_{t=1}^{T}\{s_{it}Y_{it} + c_{it}X_{it} + h_{it}I_{it}\} \quad (1)$$

$$s.t. \quad I_{i,t-1} + X_{it} - I_{it} = d_{it} + \sum_{j\in S(i)} r_{ij}X_{jt},$$
$$i = 1,\cdots,N, t = 1,\cdots,T, \quad (2)$$

$$Y_{it} = \begin{cases} 0, & if \quad X_{it=} = 0, \\ 1, & if \quad X_{it=} > 0, \end{cases}$$
$$i = 1,\cdots,N, t = 1,\cdots,T, \quad (3)$$

$$I_{it}, X_{it} \geq 0,$$
$$i = 1,\cdots,N, t = 1,\cdots,T, \quad (4)$$

$$Y_{it} \in \{0,1\},$$
$$i = 1,\cdots,N, t = 1,\cdots,T, \quad (5)$$

where the known parameters are

N = the number of items,

T = the number of time periods,

d_{it} = the external demand for item i in period t ,

$S(i)$ = the set of immediate successors of item i , ($S(i) = \varnothing$ if i is an end item),

r_{ij} = the number of units of item i required to produce one unit of j ,

s_{it} = the setup cost item i in period t ,

c_{it} = the production cost for unit item i in period t ,

h_{it} = the holding cost for unit end-of-period inventory of item i in period t ,

and the decision variables are

X_{it} = the amount of item i produced in period t (lotsize),

Y_{it} = a binary variable indicating where production is allowed for item i in period t ,

I_{it} = the inventory of item i at the end of period t .

Most of the concepts above mentioned can be find in many papers on multi-item multi-level dynamic lotsizing problems. For example, the mathematical formulation given by Maes et al. [10] is very similar to GDLP except that the overtimes of resources are not included in it.

3. Genetic algorithm of the problem

A genetic algorithm of a optimization problem is an iterative procedure to heuristically search the optimal solution of the problem. Three basic genetic operators included in the

procedure are known as reproduction, mutation and crossover (Goldberg[6], Reeves[11], etc.). All these operators work on the decision variables. During design a genetic algorithm to solve a problem, we first must give a coding scheme of the decision space of the problem.

The decision variables in GDLP are X_{it}, Y_{it} and I_{it} among which Y_{it} is a 0-1 integer variable and the others are positive real number variables. But to code a real number is too complicated to be accepted in designing a genetic algorithm of GDLP. In order to design a computationally efficient genetic algorithm, we will only consider the setup pattern variables Y_{it} as decision variables. The other real number variables will be considered as being dependent on Y_{it} and so they can be computed from Y_{it} and known parameters of the problem. Then the most important problem in designing the genetic algorithm is how these variables are computed from Y_{it} and known parameters of the problem. Making use of a useful property of dynamic lotsizing problems, we can build up the relationships between real number variables and Y_{it} and/or known parameters of the problem.

Denote population size as $MAXPOP$ (a even number) and maximum iteration times (i.e. maximum generations) as $MAXGEN$. The j th individual (i.e. decision variable) in g th generation (i.e. iteration) is coded as following:

$$Y^{g,j} = (Y_{11}^{g,j}, Y_{12}^{g,j}, \cdots Y_{1T}^{g,j}, Y_{21}^{g,j}, Y_{22}^{g,j}, \cdots, Y_{2T}^{g,j}, \vdots \; Y_{N1}^{g,j}, Y_{N2}^{g,j}, \cdots, Y_{NT}^{g,j})$$

$j = 1,2,\cdots, MAXPOP$; $g = 1,2,\cdots, MAXGEN$.

There is following property (usually named " zero-switch " property) for dynamic lotsizing problem (Afentakis and Gavish[2]):

Proposition. There is an optimal solution to GDLP in which $x_{it} I_{i,t-1} = 0$.

Given production sequence

$$\{Y_{it}, \quad i = 1,\cdots, N, \quad t = 1,\cdots, T\},$$

we can determine the production lotsizes as following according to " zero-switch " property:

(i) If $Y_{i\tau} = 0$, then $X_{i\tau} = 0$;

(ii) If $Y_{i\tau_1} = 1$, $Y_{i\tau_2} = 1$, $\tau_1 < \tau_2 \le T$ and $Y_{i\tau} = 0$ for $\tau_1 < \tau < \tau_2$, (or $Y_{i\tau} = 0$ for $\tau_1 < \tau \le T$, let $Y_{i,T+1} = 1$, $\tau_2 = T+1$), then $X_{i\tau_1} = \sum_{\tau=\tau_1}^{\tau_2-1} (d_{i\tau} + \sum_{j \in S(i)} r_{ij} X_{j\tau})$.

In order to assure feasibility (i.e. no backordering is allowed), we change the objective function to including penalty items. That's to say, before computing the fitness value of each individual, we compute the objective values of all individuals according to the following objective function:

$$COSTP(Y, X, I) = \sum_{i=1}^{N} \sum_{t=1}^{T} \{s_{it} Y_{it} + c_{it} X_{it} + h_{it} I_{it}\} + \beta \sum_{i=1}^{N} \sum_{t=1}^{T} [\max\{0, -I_{it}\}]^2 \qquad (6)$$

where β is penalty coefficient (a big enough positive number).

Now we can describe the genetic algorithm for dynamic lotsizing problems as following:

Algorithm AG

Step1. g = 0. Randomly initialize $oldpop = \{Y^{0,j}, j = 1,2,\cdots, MAXPOP\}$, the population set in g th generation.

Step2. If g = $MAXGEN$, print the solution and stop.

Step3. For each $Y^{0,j} \in oldpop, j = 1,2,\cdots, MAXPOP$, compute its objective function value as following:

3.1. Determine X^j according to $Y^{0,j}$ (compute from item 1 to N):

If

$$Y_{it_1}^{0,j} = 1, Y_{it_2}^{0,j} = 1 (1 \le t_1 < t_2 \le T+1)$$

and

$Y_{i\tau} = 0$ for $\tau_1 < \tau < \tau_2$, (assume $Y^{0,j}_{i,T+1} = 1$), then

$$X^j_{it_1} = \sum_{\tau=t_1}^{t_2-1}(d_{i\tau} + \sum_{m\in S(i)} r_{im} X^j_{m\tau}),$$

$$X^j_{i\tau} = 0 \qquad (1 \le t_1 < \tau < t_2 \le T+1).$$

3.2. Determine I^j ($I^j_{i,0}(\forall i,j)$ are known parameters) according to X^j (compute from period 1 to T for each item):

$$I^j_{i,t} = I^j_{i,t-1} + X^j_{i,t} - \sum_{m\in S(i)} X^j_{mt}, \qquad (1 \le t \le T).$$

3.3. Compute the corresponding objective function value (including penalty items) according to (6) from $Y^{0,j}, X^j, I^j$.

Step4. Produce

$newpop = \{Y^{1,j}, j = 1,2,\cdots, MAXPOP\}$, the population set in $(g+1)$th generation:

4.1. Compute the fitness value $fit(Y^{0,j})$ for each individual $Y^{0,j}$ according to their objective function values obtained in step3.3:

$$fit(Y^{0,j}) = \underset{i=1}{\overset{MAXPOP}{Max}} COSTP(Y^{0,i}) + \varepsilon - COSTP(Y^{0,j}),$$

$$j = 1,2,\cdots, MAXPOP,$$

where ε is a positive constant and $COSTP(Y) = COSTP(Y, X, I)$.

4.2. (Reproduction/Selection) Select Y^{0,j_1}, Y^{0,j_2} from the set *oldpop* according to fitness values. The probability to select $Y^{0,j}$ is

$$pr(Y^{0,j}) = fit(Y^{0,j}) \Big/ \sum_{i=1}^{MAXPOP} fit(Y^{0,i}),$$

$$j = 1,2,\cdots, MAXPOP.$$

4.3. (Mutation) If the mutation probability is p_m, then after mutation,

$$Y'^{0,j}_{it} = \begin{cases} Y^{0,j}_{it}, & in\ probability\ of\ \ 1-p_m, \\ \overline{Y}^{0,j}_{it}, & in\ probability\ of\ \ p_m, \end{cases}$$

$$j = j_1, j_2.$$

Here

$$\overline{Y}^{0,j}_{it} = \begin{cases} 0, & if\ \ Y^{0,j}_{it} = 1, \\ 1, & if\ \ Y^{0,j}_{it} = 0. \end{cases}$$

4.4. (Crossover) If the crossover probability is p_c, then after crossover,

$$Y^{1,j} = \begin{cases} Y'^{0,j}, & in\ probability\ of\ \ 1-p_c, \\ Y''^{0,j}, & in\ probability\ of\ \ p_m, \end{cases}$$

$$j = j_1, j_2.$$

Here randomly select a crossover position $s(1 \le s \le NT)$ and then

$$Y''^{0,j_1} = (Y'^{0,j_1}_1, Y'^{0,j_1}_2, \cdots, Y'^{0,j_1}_s, Y'^{0,j_2}_{s+1}, Y'^{0,j_2}_{s+2}, \cdots, Y'^{0,j_2}_{NT})$$

$$Y''^{0,j_2} = (Y'^{0,j_2}_1, Y'^{0,j_2}_2, \cdots, Y'^{0,j_2}_s, Y'^{0,j_1}_{s+1}, Y'^{0,j_1}_{s+2}, \cdots, Y'^{0,j_1}_{NT})$$

4.5. Add new individuals Y^{1,j_1}, Y^{1,j_2} to the set *newpop*.

4.6. If all $MAXPOP$ individuals are produced, go to step5; otherwise go to step4.2.

Step5. $g = g+1, oldpop = newpop$, i.e. $Y^{0,j}_{it} = Y^{1,j}_{it}, \forall i,t,j$, and go to step2.

4. Computational Experiments

The performance of the algorithm described in the previous section will be evaluated on a set of testing problems. The algorithm are programmed with Borland C++ 3.1 running under MS-DOS 6.2 on a compatible PC486DX2-50. The experiments reveal that this algorithm obtain very good approximation solution of GDLP in reasonable computation time. Following are some of the examples of the experiments.

We use the following control parameters for testing this genetic algorithm:

- Maximum generations *MAXGEN*=100;
- Population size *MAXPOP*=30;
- Mutation Probability p_m=0.033;
- Crossover Probability p_c=0.6.

Example 1. (Single Item) This example uses the famous classical test data given by Wagner and Whitin[13], N=1, T=12 (Table 1).

Table 1. Testing data for example 1

t	1	2	3	4	5	6	7	8	9	10	11	12
d_t	69	29	36	61	61	26	34	67	45	67	79	56
s_t	85	102	102	101	98	114	105	86	119	110	98	114
h_t	1	1	1	1	1	1	1	1	1	1	1	1

We have run algorithm GA for 20 times for this problem, and algorithm GA always find the optimal value 864 in our experiments. Following are a typical output:

gen= 0 min= 946 max= 2433 avg= 1171
gen= 1 min= 946 max= 1197 avg= 1105
gen= 2 min= 888 max= 1397 avg= 976
gen= 3 min= 888 max= 1397 avg= 976
gen= 4 min= 888 max= 1339 avg= 930
gen= 5 min= 864 max= 1339 avg= 916

Here "gen" denotes the iterative times (generations); "min", "max" and "avg" denotes, respectively, the minimum, maximum and average cost found in this generation.

Example 2. (General System) Production structure is shown as Fig.1, N=4, T=5. Testing data is shown in table 2.

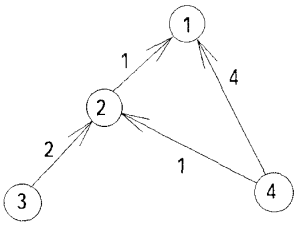

Fig. 1 Product Structure for Example 2

Table 2. Testing data for example 2

t	1	2	3	4	5	6
d_{1t}	2	5	1	3	10	4
d_{2t}	0	2	5	0	0	10
d_{3t}	1	0	2	0	0	1
d_{4t}	0	0	3	0	1	0
s_{1t}	44	44	44	44	44	44
s_{2t}	51	51	51	51	51	51
s_{3t}	24	24	24	24	24	24
s_{4t}	33	33	33	33	33	33
h_{1t}	10	10	10	10	10	10
h_{2t}	5	5	5	5	5	5
h_{3t}	1	1	1	1	1	1
h_{4t}	1	1	1	1	1	1

Using a complete enumeration method, we can find the optimal value for this problem is 625. For this problem, algorithm GA also always find the optimal solution in our experiments. Following are a typical output:

gen= 0 min= 755 max= 1365 avg= 1002
gen= 1 min= 755 max= 1048 avg= 954
gen= 2 min= 723 max= 1047 avg= 800
gen= 3 min= 723 max= 1047 avg= 771
gen= 4 min= 676 max= 1015 avg= 739
gen= 5 min= 671 max= 1026 avg= 754
gen= 6 min= 645 max= 905 avg= 696
gen= 7 min= 645 max= 1161 avg= 739
gen= 8 min= 645 max= 1161 avg= 699
gen= 9 min= 625 max= 1161 avg= 676

These examples reveals that the genetic algorithm GA is an effect and effective algorithm for solving general dynamic lotsizing problems.

5. Summary

We developed a genetic algorithm for general multi-item multi-leval dynamic lot-sizing problems. Product structure can be any type, each item in the system can have external demands and the cost parameters (setup cost, holding cost, production cost) can be time-varying. Experiments show that this method is effective and efficient.

This implementation methodology can also used in designing genetic algorithms for capacitated lotsizing problems (Xie et. al.[14]). For the application of genetic algorithms for lotsizing problems is just at its beginning, more theoretical analysis and computation experiments should be made to this kind of genetic algorithms.

Acknowledgments

This research was partly supported by the CIMS office of 863 plan in China. The author is grateful to professor Han Jiye, Jiang Qiyuan, Xing Wenxun and Ren Shouju for their encouragement and insightful comments.

References

[1] Afentakis P., A Parallel Heuristic Algorithm for Lot Sizing in Multistage Production Systems, IIE Transactions, 19/1(1987), 34-42.

[2] Afentakis P., B. Gavish, Optimal Lot-Sizing Algorithms for Complex Product Structures, Operations Research, 34/2(1986), 237-249.

[3] Afentakis P., B. Gavish, V. Karmarkar, Optimal Solutions to the Lot-Sizing Problem in Multi-Stage Assembly

Systems, Management Science, 30/3(1984), 222-239.

[4] Aggarwal A., J. K. Park, Improved Algorithms for Economic Lot Size Problems, Operations Research, 41/3(1993), 549-571.

[5] Coleman B. J., A Further Analysis of Variable Demand Lot-Sizing Techniques, Production and Inventory Management Journal, 33/3 (1992), 19-24.

[6] Goldberg D. E., Genetic Algorithms in Search, Optimization, and Machine Learning, 1989, Addison-Wesley, Reading, Mass., USA.

[7] Gupta Y. P., Y. Keung, A Review of Multi-Stage Lot-Sizing Models, International Journal of Operations and Production Management, 10/9(1990), 57-73.

[8] Heinrich C. E., Ch. Schneeweiss, Multi-Stage Lot-Sizing for General Production Systems, in: S. Axsater, Ch. Schneeweiss, E. Silver(eds.), Multistage Production Planning and Inventory Control, Lecture Notes in Economics and Mathematical Systems 266, 1986, Springer, Berlin, 150-181.

[9] Lambrecht M. R., J. VanderEechen, H. Vanderveken, Review of Optimal and Heuristic Models for a Class of Facilities in Series Dynamic Lot Size Problems, in: L. B. Schwarz(ed.), Multi-level Production/Inventory Control Systems: theory and Practice, TIMS Studies in the Management Sciences 16, 1981, Amsterdam, North-Holland, 69-94.

[10] Maes J., J. O. McClain, L. N. Van Wassenhove, Multilevel Capacitated Lotsizing Complexity and LP-Based Heuristics, European Journal of Operational Research, 53/2(1991), 131-148.

[11] Reeves R. C.(ed.), Modern Heuristic Techniques for Combinatorial Problems, 1993, Blackwell Scientific, Oxford.

[12] Rosling K., Optimal Lot Sizing for Dynamic Assembly Systems, in: S. Axsater, Ch. Schneeweiss, E. Silver(eds.), Multistage Production Planning and Inventory Control, Lecture Notes in Economics and Mathematical Systems 266, 1986, Springer, Berlin, 119-131.

[13] Wagner H. M., T. M. Whitin, Dynamic Version of the Economic Lot Size Model, Management Science, 5/1(1958), 89-96.

[14] Xie J., Q. Jiang, W. Xing, A Genetic Algorithms for General Capacitated Lotsizing Problems, Technical Report, Tsinghua University, Beijing, 1994.