Journal of Industrial Engineering International *Islamic Azad University, South Tehran Branch*
March 2006, Vol. 2, No. 1, 53 - 64

Solving a generalized aggregate production planning problem by genetic algorithms

R. Tavakkoli-Moghaddam*

Associate Professor, Department of Industrial Engineering , Faculty of Engineering, University of Tehran, Iran

N. Safaei

Research Scholar, Department of Industrial Engineering Iran University Science and Technology, Tehran, Iran

Abstract

This paper presents a genetic algorithm (GA) for solving a generalized model of single-item resource-constrained aggregate production planning (APP) with linear cost functions. APP belongs to a class of production planning problems in which there is a single production variable representing the total production of all products. We linearize a linear mixed-integer model of APP subject to hiring/firing of workforce, available regular/over time, and inventory/shortage/subcontracting allowable level where the total demand must fully be satisfied at end of the horizon planning. Due to NP-hard class of APP, the real-world sized problems cannot optimality be solved within a reasonable time. In this paper, we develop the proposed genetic algorithm with effective operators for solving the proposed model with an integer representation. This model is optimally solved and validated in small-sized problems by an optimization software package, in which the obtained results are compared with GA results. The results imply the efficiency of the proposed GA achieving to near optimal solutions within a reasonably computational time.

Keywords: Aggregate production planning; Linear mix-integer programming; Genetic algorithm

1. Introduction

Aggregate production planning (APP) belongs to a class of production planning problems in which there is a single production variable representing the total production of all products [9]. This means that there must be some units for measuring the aggregate output that is called *"aggregate unit of production*" such as tons for a steel mill, cases for a bottling plant, barrels for a refinery, machine-hours for a job shop, or man-hours for a maintenance department. APP is a medium term capacity planning that determines the minimum cost workforce and production plans to meet fluctuating demand requirements over a planning horizon. In general, its aim is to determine the production quantity and inventory level in an aggregate term in such a way that the expected demand is met by utilizing the resources of an organization efficiently and effectively [23]. In fact, APP plans and controls the process of different aspects of the entire production activities in order to satisfy customers' demands. In other words, APP is the best use of potentials and capacities when changes in demands occur. In APP, the management must decide upon a production rate in a mix-term plan horizon. The final output of an APP is the periodically production rate of the factory for a given horizon planning with being involved in production details. The solution to an aggregate model establishes the production capacity and the aggregate production level for each period.

Some of researchers have advocated a hierarchical approach for modeling the production planning problem [1-5]. With this approach, commonly known as hierarchical production planning (HPP), the problem is usually broken down into a series of decision levels. The highest decision level is generally known as aggregate production planning. At the aggregate level, the number of variables is reduced to a manageable quantity by grouping similar resources into resource centers and similar finished products into

* *Corresponding author. E-mail: tavakoli@ut.ac.ir*

common product types as well as subdividing the planning horizon into large discrete time units, say in months. Then, an APP model is often developed and employed to construct an aggregate plan. Eventually, this aggregate plan is passed to a lower decision level considering a much shorter planning horizon and disaggregates it.

APP usually covers a time period ranging from 12 to 24 months. In the aggregate plan, data are usually based on monthly or quarterly data. The most important input data for an APP problem are demand forecast because the main aim of APP is to respond to demand fluctuations in a proper manner. Three principal types of resources involved in an APP problem are manpower, time, and subcontract ones. The main limitations of such problem can include machine capacity, storage area, inventory level, safety stock level, manpower adjustment, regular/over time of production, and subcontract level. In addition, the corresponding costs also consist of inventory holding, overtime, subcontracts, shortage, loss sales, and hiring/firing/training of labor [11].

According to Saad [27], all traditional models of APP problems may be classified into four categories as follows: (1) linear programming (LP) [8,29], (2) linear decision rule (LDR) [20], (3) transportation method [6], and (4) management coefficient approach [7]. Also, some of heuristic APP strategies to satisfy the demand are discussed in details in the literature [10,28]. Some of these techniques yield the optimum solution, while others give only acceptable ones. In addition, some required models that are easy to be formulated, while others require complicated models. The principal disadvantage of APP problems is the computational difficulty resulting from the size of the model or type of cost functions (linear, nonlinear or both of them). Vollman [31] pointed to the disadvantage of the use of the traditional APP techniques in industry. He put forward the following reasons for such a failure:

1. Uniform rates have been assumed for different products. This assumption does not reflect reality in organizations producing various products.
2. It is so difficult for the general managers to grasp and understand the mathematical methods used in available techniques. Hence, the lack of interest on their parts in utilizing those techniques.
3. Models used in the above techniques require deterministic and certain data that the collection and quantification of these data is difficult requiring extra costs such as for employment and training of new staffs, etc. Gilgeous [16] has also put forward the following points:

A. Methods are rooted in their own specific qualifications. Hence, they are appropriate for a definite rage of variables.

B. The real world problems face a definite range of planning variables. Therefore, none of the available techniques can produce optimal or near-optimal solutions of plans. Other limitations of the current techniques are as follow:

- Functions used by the current techniques do not appropriately reflect real cost functions in organizations.
- Models of a case cannot directly be used in other cases.

On the other hand, the consideration of the all realistic parameters in an APP model makes the model difficult and non-optimally solvable. Therefore, it is necessary for a trade-off between the selection of a non-exact model with an optimal solution and an exact model with a near-optimal solution. Obviously, an exact model with a near-optimal solution is preferred. Nowadays, meta-heuristic methods are widely used as near-optimal approaches for solving NP-hard problems such as a generalized APP and the like. Due to the above reason, we examine the efficiency of a well-known and effective metaheuristic method based on a genetic algorithm for solving a generalized APP problem with semi-realistic conditions in which most of real parameters are considered.

Numerous models have been proposed to model APP [13,18,22,25]. Among the mathematical programming approaches, linear programming (LP) has been the most widely accepted method as APP problems with large numbers of variables and constraints solved efficiently. Also, the parametric programming or post optimality analysis is used to investigate the effects of changing certain constants in a linear program over a specified range [12]. The linear decision rule can be considered as an important contribution for long-term strategic APP decisions [21]. This analytical rule is determined to minimize the quadratic cost functions subject to inventory and workforce balance equations. As a result, it provides an optimal smoothing solution for aggregate inventory, production, and workforce levels.

Wang and Fang [32] presented a genetics-based approach to imitate the human decision procedure for a classical product mix problem as an APP problem in a fuzzy environment. Their research is just one in context of the GA implementation for an APP problem. Tavakkoli-Moghaddam and Biyabani [30] proposed a special design of a GA to work out an APP in order to minimize production costs in a real-case

study of a car industry. There is no research in the literature using metaheuristics for an generalized APP problem with a long horizon planning.

The rest of the paper is organized as follows. Section 2 describes the proposed APP problem and model. Section 3 considers the proposed GA to solve the model. Some numerical results are included in section 4 and concluding remarks are made in section 5.

2. Problem formulation

We propose a nonlinear model of APP considering four main costs as production, adjusting work force, inventory carrying, and shortage ones. In this proposed model, all resources are also assumed to be finite and restricted by the decision maker and customer, available time for manpower in regular and over time, sub-contracting level, storage area, shortage level (customer satisfaction), and adjusting work force level. The ending inventory/shortage is not allowable. The demand is deterministic and dynamic under a multi-period horizon planning. The primary resources, beginning inventory, and initial work force level are known as a priori. We linearize the above nonlinear model to linear mixed-integer one. The mathematical presentation of the proposed model is descried below.

2.1. Mathematical formulation

Input parameters are defined as follows:

T : Number of periods in horizon planning.
d_t : Forecasted demand in period t.
I_0 : Beginning inventory.
W_0 : Initial work force level.
$I_{\max}$: Minimum inventory level available in period t (units).
$B_{\max}$: Maximum backorder (shortage) level available in period t (units).
$H_{\max}$: Maximum allowable hiring in each period.
$F_{\max}$: Maximum allowable firing in each period.
$R_{\max}$: Maximum production volume in regular time in each period (units).
$O_{\max}$: Maximum production volume in overtime time available in each period (units).
$S_{\max}$: Maximum production volume in subcontracted available in each period (units).
r_t : Regular time production cost per unit in period t (\$/unit).
o_t : Overtime production cost per unit in period t (\$/unit).
s_t : Subcontracting cost per unit in period t (\$/unit).
h_t : Hiring cost per one worker in period t (\$/man).
f_t : Firing cost per one worker in period t (\$/man).
h^+_t : Inventory carrying cost per unit in period t (\$/unit).
h^-_t : Backorder cost per unit in period t (\$/unit).
k : Number of workers required per unit product.

Decision variables are defined as follows:

P_t : Total aggregate production in period t.
R_t : Regular time production volume in period t (units).
O_t : Overtime production volume in period t (units).
S_t : Subcontracting production volume in period t (units).
H_t : Worker hired in period t.
F_t : Worker fired in period t.
I_t : Inventory/backorder level in period t (negative inventory ≡ shortage) .

By defining the above notations, the proposed mathematical model of APP can be written below:

$$\text{Min } Z = \sum_{t=1}^{T} (r_t R_t + o_t O_t + s_t S_t + h_t H_t + f_t F_t + h_t^+ \max\{I_t, 0\} + h_t^- \max\{-I_t, 0\}) \quad (1)$$

Subject to:

$$I_t = I_{t-1} + P_t - d_t \quad \forall t \quad (2)$$

$$P_t = R_t + O_t + S_t \quad \forall t \quad (3)$$

$$k \times P_1 = W_0 + H_1 - F_1 \quad (4)$$

$$k \times P_t = k \times P_{t-1} + H_t - F_t \quad \forall t > 1 \quad (5)$$

$$R_t \le R_{\max} \quad \forall t \quad (6)$$

$$O_t \le O_{\max} \quad \forall t \quad (7)$$

$$S_t \le S_{\max} \quad \forall t \quad (8)$$

$$\max\{I_t, 0\} \le I_{\max} \quad \forall t \quad (9)$$

$$\max\{-I_t, 0\} \le B_{\max} \qquad \forall t < T \tag{10}$$

$$H_t \le H_{\max} \qquad \forall t \tag{11}$$

$$F_t \le F_{\max} \qquad \forall t \tag{12}$$

$$\max\{-I_T, 0\} = 0 \tag{13}$$

$$P_t, R_t, O_{t,}, S_t, H_t, F_t \in \mathrm{N}^+ \text{ and } I_t \in \mathrm{N}. \tag{14}$$

The nonlinear objective function (1) is equal to the total cost of production, adjusting work force, inventory carrying, and shortage in planning horizon. Equation (2) indicates the balanced inventory constraint between periods. Equation (3) determines the total production level at each period. Equations (4) and (5) calculate the number of workers hired/fired at each period with respect to the total production level. Inequalities (6), (7), and (8) correspond to the maximum available volume of production in regular time, overtime, and subcontracting at each period respectively. Inequalities (9) and (10) correspond to the maximum allowable level of inventory and shortage (backorder) at each period respectively. Likewise, inequalities (11) and (12) correspond to the maximum allowable level of hiring and firing at each period respectively. Finally, Equation (13) indicates the lack of shortage at the end of horizon planning. The proposed model includes $10T$ constraints and $7T$ integer variables, where T is the number of periods in horizon planning.

2.2. Model linearization

The above proposed model is a nonlinear one because the existence of negative inventory or shortage in the system resulting the **max{}** term in the objective function (1). To transform the proposed model into a linear one, the non-negative variables I^+_t, I^-_t are introduced where $I^+_t = \max\{ I_t , 0\}$, $I^-_t = \max\{-I_t , 0\}$.

Then, I^+_t and I^-_t are put in the objective function and the following inequalities are introduced to the proposed model as constraints.

$$I^+_t \ge I_t \qquad \forall t , \tag{15}$$

$$I^-_t \ge -I_t , I^-_T = 0 \qquad \forall t . \tag{16}$$

Thus, constraints (10) and (11) must be changed as $I^+_t \le I_{\max}$ and $I^-_t \le S_{\max}$ respectively.

2.3. An illustrative example

To evaluate the performance of the developed model, a small-sized instance is generated at random [26] and optimally solved by a branch-and-bound (B&B) method. This example consists of 12 periods with the total demand $\sum D_t$=32190, in which its original data are shown in Table 1. The optimal solution obtained by the B&B method with CPU time less than one second is shown in Table 2. The graph associated with the demand rate versus the total production is shown in Figure 1. As depicted, the demand has a chaotic pattern; contrariwise, production rate has a nearly smooth pattern because of high hiring and firing costs with respect to other costs effecting on constraint (5).

3. Genetic algorithm implementation

Genetic algorithms (GAs) [14,15,17,19,24] attempted to mimic the biological evolution process for discovering good solutions. They are based on a direct analogy to the Darwinian natural selection and mutations in the biological reproduction. They belong to a category of heuristics known as stochastic search methods employing randomized choice of operators in their search strategy and they do not depend on priori knowledge of the features of the domain completely. These operators have been conceived through

Table 1. A typical Example.

period	1	2	3	4	5	6	7	8	9	10	11	12
d_t	1897	3602	3214	2068	3773	2628	1483	3944	2799	1654	3358	2970
r_t	14	13	16	16	15	14	13	16	15	14	13	17
o_t	21	28	26	23	29	26	22	20	27	23	30	28
s_t	40	34	31	35	39	33	37	34	38	32	36	33
h_t	105	117	108	121	133	146	108	149	112	124	137	128
f_t	238	208	214	234	205	225	245	202	222	242	212	219
h^+_t	1	9	2	10	8	7	5	8	7	5	4	6
h^-_t	20	18	14	12	11	19	18	13	12	10	18	14

$R_{\max}$=2400, $O_{\max}$=400, $S_{\max}$=200, $H_{\max}$=100, $F_{\max}$=50, I_0=1200, W_0=400, k=0.2

Table 2. Optimal solutions.

	R_t	O_t	S_t	I_t	F_t	H_t	W_t	Z_t
1	1895	1895	0	0	0	21	1198	0
2	2395	2395	0	0	100	0	0	9
3	2880	2400	400	80	97	0	0	343
4	2880	2400	400	80	0	0	469	0
5	2880	2400	400	80	0	0	0	424
6	2880	2400	400	80	0	0	0	172
7	2800	2400	400	0	0	16	1145	0
8	2800	2400	400	0	0	0	1	0
9	2695	2400	295	0	0	21	0	103
10	2695	2400	295	0	0	0	938	0
11	2695	2400	295	0	0	0	275	0
12	2695	2400	295	0	0	0	0	0
Total	32190	28290	3580	320	197	58	4026	1051

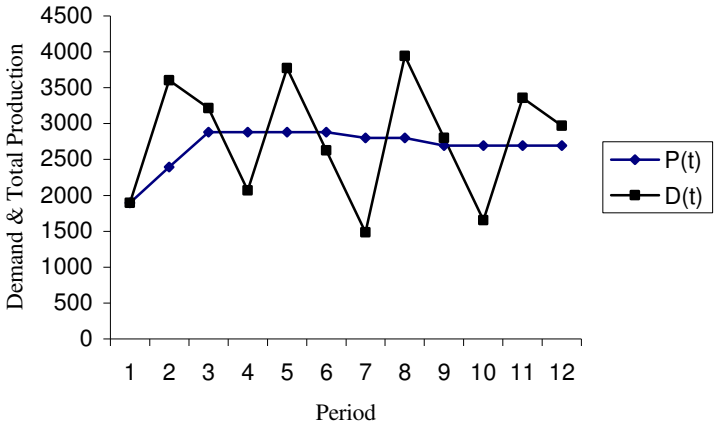

Figure 1. Demand rate versus optimal production rate in each period.

abstractions of natural genetic mechanisms such as crossover and mutation and they have been cast into algorithmic forms. Repetitive executions of these heuristics need not yield the same solution. A genetic algorithm maintains a collection or population of solutions throughout the search. It initializes the population with a pool of potential solutions to the problem and seeks to produce better solutions (individuals) by combining the better of the existing ones through the use of one or more genetic operators. Individuals are chosen at each iteration with a bias towards those with the best objective or fitness values. With various mapping techniques and an appropriate measure of fitness of individuals (i.e., objective function value), a genetic algorithm can be tailored to evolve a solution for many types of optimization problems.

3.1. Chromosome representation

The first and important step of GA is to design and represent a proper chromosome for the solution structure coding. This structure severely depends on the nature of the model's decision variables and constraints. The binary representation scheme of genes is one of the most commonly used ones. However, the "real/integer number" presentation scheme has been very successful for a function optimization in recently years [33], because it needs no transformation of number systems. According to our proposed model, each solution is represented by a chromosome formed as an integer vector with T genes as shown in Figure 2, where T is the number of periods. The value of t^{th} gene, i.e, P_t, indicates the total production in period t that is bounded in interval $[0, R_{max}+O_{max}+S_{max}]$. Con-

straint (12) forces $\sum P_t \geq \sum D_t$. Also, constraint (9) forces $\sum P_t \leq \sum D_t + I_{max}$, thus for each chromosome we have $\sum D_t \leq \sum P_t \leq \sum D_t + I_{max}$. Other decision variables, i.e, R_t, O_t, S_t, I_t, H_t, and F_t must be updated with respect to the current solution and the model's constraints by Equations (17) to (22).

1	2	3	…	T
P_1	P_2	P_3	…	P_T

Figure 2. Chromosome representation.

$$R_t = \min\{R_{max}, P_t\} \quad \forall t \tag{17}$$

$$O_t = \min\{O_{max}, \max\{P_t - R_{max}, 0\}\} \quad \forall t \tag{18}$$

$$S_t = \min\{S_{max}, \max\{P_t - R_{max} - O_{max}, 0\}\} \quad \forall t \tag{19}$$

$$I_t = I_{t-1} + P_t - D_t \quad \forall t \tag{20}$$

$$H_t = \max\{0, k\,(P_t - P_{t-1})\} \quad \forall t \tag{21}$$

$$F_t = \max\{0, k\,(P_{t-1} - P_t)\} \quad \forall t\,, \tag{22}$$

where $P_0=w_0/k$ and I_0 are known as a priori. Equations (17), (18), and (19) allot proportionally the total production between regular time, overtime, and subcontract level respectively at each period. If one of the constraints (9) to (13) is violated by on hand solution, then the degree of violation is added to the objective function (see Equation 23) by the penalty coefficient λ, where is a large positive number. Equation (23) indicates the fitness function that chromosomes are evaluated by that.

$$\begin{aligned} Z = &\sum_{t=1}^{T}(r_t R_t + o_t O_t + s_t S_t + h_t H_t + f_t F_t + h_t^+ w_t + h_t^- z_t) \\ &+ \lambda \sum_{t=1}^{T}(\max\{0, I_t^+ - I_{max}\} + \max\{0, I_t^- - S_{max}\} \\ &+ \max\{0, H_t - H_{max}\} + \max\{0, F_t - F_{max}\})\,. \end{aligned} \tag{23}$$

3.2. Genetic operators

To explore solution space, we use an extended single point crossover and a new version of crossover so-called "Arithmetic Crossover". Also, to exploit neighbor solutions, three types of mutations are introduced as so-called "Exchange Mutation", "Arithmetic Mutation", and "Inversion". The introduced operators are discussed in detail as follows:

- **Single point crossover (SPC).** It is an extended version of classical crossover in which two selective parents are recombined by a single cross point (*cp*). As depicted earlier, the cross point must be determined in such a way that the inequality $\sum D_t \leq \sum P_t \leq \sum D_t + I_{max}$ is observed in new offspring. Thus, according to Figure 3, it is necessary that $\sum_{t=1}^{T} D_t \leq \sum_{t=1}^{cp} P_t + \sum_{t=cp+1}^{T} P_t' = \sum_{t=1}^{T} D_t + I_{max}$ and $\sum_{t=1}^{T} D_t \leq \sum_{t=1}^{cp} P_t' + \sum_{t=cp+1}^{T} P_t = \sum_{t=1}^{T} D_t + I_{max}$ are observed in selective parents. For more description, suppose that T=5, $\sum D_t$ =50, I_{max}=10 and two selective parents be {8, 12, 9, 11, 12} and {15, 9, 7, 6, 16}, then *cp*=3 is a true cross point because 50 ≤ 8+12+9+**6+16**=51 ≤ 50+10 and 50 ≤ 15+9+7+**11+12**= 54 ≤ 50+10. Thus, two new offspring are as {8, 12, 9, **6, 16**} and {13, 9, 7, **11, 12**}. In general, it is possible that such cross point is never found.

- **Arithmetic crossover (AC).** Arithmetic crossover products a single offspring by linear combining two selective parents as shown in Figure 4. The principle of this operator is based on the following fact:

$$\sum_{t=1}^{T} D_t \leq \sum_{t=1}^{T}[\lambda P_t + (1-\lambda)P_t] \leq \sum_{t=1}^{T} D_t + I_{max}, \lambda \in (0,1).$$

- **Exchange mutation (EM).** Exchange mutation swaps the value of the two random selected genes together. As shown in Figure 5, the value of genes 2 and 5 are exchanged together.

- **Inversion-mutation (IM).** Inversion mutation inverses a selective substring of the current solution. As shown in Figure 6, the substring selected between periods 2 to 6 is inversed in the new solution.

- **Arithmetic mutation (AM).** Arithmetic mutation reduces the production level for a selective period (gene) by the amount of Δ and then it is added to other selective period. This causes that the total production level is remained constant through horizon planning. As shown in Figure 7, the product level in period 2 is reduced by the amount of Δ and then it is added to one in period 5.

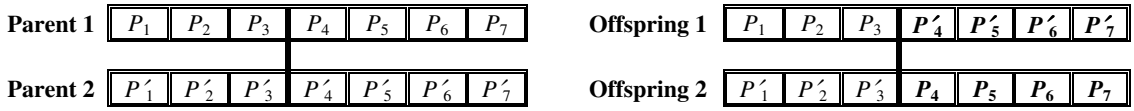

Figure 3. Single point crossover implementation.

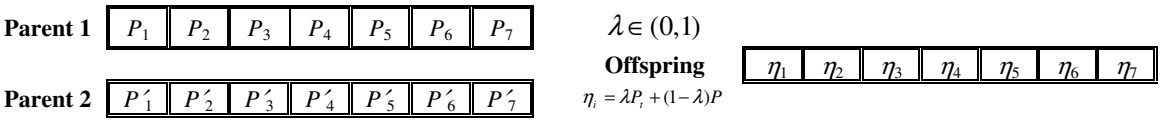

Figure 4. Arithmetic mutation implementation.

Figure 5. Exchange mutation implementation.

Figure 6. Inversion mutation implementation.

Figure 7. Arithmetic mutation implementation.

Each operator must be selected with a certain rate or probability that is known in advance by GA parameter settings. The operator selection strategy is determined by a proper rate for each operator (that called "tuning") and how to select them leading to the convergence of GA to the global optimum neighborhood in a reasonable time. We use a two-level strategy for the operator selection which the type of operator (i.e., "crossover", "mutation" or "reproduction") is determined in level I (type level) and then the mode of operator selected in level I is determined in level II (mode level). A schema of the applied strategy is shown in Figure 8. Term P_Ω dedicates the rate or selection probability of operator mode/type Ω, where $P_C + P_M + P_R = 1$, $P_{SPC}, + P_{AC} = 1$, and $P_{EM}+P_{IM}+P_{AM} = 1$. For instance, the probability of calling arithmetic crossover at each iteration is equal to $P_C \times P_{AC}$. In general, a classical tuning for the type level can be as $P_C \cong 0.7$, $P_R = (1/Pop\_Size)$, and $P_M = 1-(P_C + P_R)$. The term of $P_R = (1/Pop\_Size)$ causes only one elite at each calling immigrates to the new generation.

As shown in Figure 8, in addition to the maintained operators, we used "reproduction" operator for maintaining the elites at each generation. For this work, chromosomes in the current generation are first sorted as descending order with respect to their fitness (i.e., objective function value), then $\rho = \min\{P_R \times Pop\_Size, Pop\_Size\text{-}K\}$ percent of the best chromosomes are immigrated to the next generation, where P_R is the reproduction rate, *Pop_Size* is the number of population at each generation and *K* is the number of required offspring for creating the new generation in the current time.

```
operator_Type = RANDOM_SELECTION (P_C, P_M, P_R)
SELECT CASE Operator_Type
 CASE crossover :
  cross_mode = RANDOM_SELECTION (P_SPC, P_AC)
  SELECT CASE cross_mode
   CASE SPC:  CALL Single_Point_Crossover( )
   CASE DPC: CALL Double_Point_Crossover( )
   CASE AC:   CALL Arithmetic_Crossover( )
  END SELECT

 CASE mutation :
  mut_mode = RANDOM_SELECTION (P_EM, P_IM, P_AM)
  SELECT CASE mut_mode
   CASE EM:   CALL Echelon_Mutation( )
   CASE DPC: CALL Inversion( )
   CASE AC:   CALL Arithmetic_Mutation( )
  END SELECT
 CASE reproduction : CALL Reproduction( )
END SELECT
```

Figure 8. Two-level procedure for the operator selection.

3.3. Parent selection strategy

The parent selection strategy means how to choose the chromosome in the current population that will create offspring for the next generation. In general, it is better that the best solutions in the current generation have more chance to be selected as parents to create offspring. The most common method for the selection mechanism is the "roulette wheel" sampling, in which each chromosome is assigned a slice of a circular roulette wheel and the size of the slice is proportional to the chromosome's fitness. The wheel is spun *Pop_Size* times. On each spin, the chromosome under the wheel's marker is selected to be in the pool of parents for the next generation. This method can be implemented as follows:

1. Let F be the sum of the fitness values of all solutions in the current population as follows:

$$F = \sum_{i=1}^{Pop\_Size} (f_i), \tag{24}$$

where, f_i is the fitness values of solution i and *Pop_Size* is equal to the number of chromosomes in the current population.

2. Let P_i be the relative probability related to chromosome i as follows:

$$P_i = \frac{f_i}{F}; \quad k = 1, 2, ..., Pop\_Size. \tag{25}$$

3. Let q_i be the cumulative probability related to chromosome i as follows:

$$q_k = \sum_{j=1}^{k} P_j, \quad k = 1, 2, ..., Pop\_Size. \tag{26}$$

4. Generate a random number, say r, in the range of [0, 1]. If $r < q_1$, then the first chromosome is selected. Otherwise, the i^{th} chromosome is selected where,

$$q_{i-1} < r < q_i, \quad (2 \le i \le Pop\_Size). \tag{27}$$

The fitness of each solution is obtained by Equation (23). The initial population is randomly created in terms of a continuous uniform distribution. To maintain the diversity in each population, we pass each lately created offspring through a similarity check filter. This filter verifies the similarity between the new offspring and each other chromosomes in the new generation. According to the chromosome structure shown in Figure 2, the similarity between two chromosomes i, j is equal to $s_{ij}=e/(3\times T)$ where $0\le s_{ij} \le 1$, e is the number of genes with same allele (i.e., value) and locus (i.e., position) in both chromosomes. If s_{ij} is greater than an arbitrary value $0 < \theta \le 1$, then two chromosomes i and j are the same.

3.4. Stoppage rules

Two criteria are used as stoppage rules: 1) maximum number of elapsed generation (G) that is a common criterion and 2) deviation-generation. If the deviation of the current generation decreases below an arbitrary constant (ε), then the algorithm is stopped. The deviation of each generation g can be calculated as follow:

$$\sigma_g = \left[\frac{1}{Pop\_Size} \sum_{k=1}^{Pop\_Size} \left(F_g^k - \bar{F}_g \right)^2 \right]^{\frac{1}{2}}, \tag{28}$$

where F_g^k is the fitness of k^{th} chromosome in generation g. $\bar{F}_g$ is average fitness of all chromosomes in generation g that is calculated as follows:

$$\bar{F}_g = \frac{1}{Pop\_Size} \sum_{k=1}^{Pop\_Size} F_g^k. \tag{29}$$

Thus, if $g > G$ or $\sigma^2_g \le \varepsilon$, then the algorithm is stopped.

4. Experimental results

In this section, we implement the proposed GA for the proposed APP problem by solving several instances in medium and large sizes in addition to the a small-sized instance presented in Table 1 (12 period). Then, we present the results of computational experiments associated with the instances generated at random according to a uniform distribution in certain intervals as shown in Table 3. Intervals are adopted from the presented data in the literature [27]. The instances are optimally solved by the Lingo 8.0 software package by using a branch-and-bound (B&B) method on an intel® Celeron® mobile 1.3 GHz Personal Computer with 512 Mb RAM. The best obtained GA parameter setting is shown in Table 4. The experimental results show that the extended single-point crossover, exchange mutation and inversion are more efficient than other introduced operators. Also, the best of population size is obtained *Pop_Size* =150. The compare resulting from GA and optimal solutions with respect to the objective function value (OFV) and CPU time related to five instances with 12, 30, 40, 45, and 50 periods is presented in Table 5. As depicted in Table 5, the proposed model is very hard in terms of computational time for instances with more than 50 periods and its associated CPU time exceeds more than one hour. Though, the proposed GA does not present a reasonable CPU time in small-sized instances with respect to the optimal one. However by increasing the dimension of instances, the results obtained by GA become reasonable and acceptable. As shown in Table 5, the OFV related to the problems 8 and 9 cannot be obtained in a reasonable time (say, more than 1 hour). Thus, we consider an interval for the OFV for large-sized instances that constructed by the IP bound and IP best values (see LINGO software documents for more details). The mean gap of the OFV between GA and optimal methods is reported as 0.36 % that is a very satisfactory and promising result. This gap is calculated as $[(OFV_{GA}-OFV_{OPT})/OFV_{OPT}]\times 100$.

A typical convergence of the proposed GA with respect to the deviation of generations according to Equation (29) is shown in Figure 9 (Problem 5 from Table 5). Figure 9 shows that this method becomes convergent to a small neighborhood of optimum solution within 50 generations. Also, the capability of the proposed GA approaching to feasible space within 40 generations is shown in Figure 10. This figure indicates the continuing reduction of infeasibility because of designing good operators and choosing a proper value of the penalty coefficient. The comparison between optimal and GA production rate related to problem 5 given in Table 5 is periodically shown in Figure 11. As shown in this figure, the GA production rate approximately conforms to optimal production rate.

Table 3. Intervals to random generation of instances.

Parameter	Interval
d_t	U(P_{max}-δ, P_{max} +δ)
r_t	U(10, 20)
o_t	U(20, 30)
s_t	U(30, 40)
h_t	U(100, 150)
f_t	U(200, 250)
H^+_t	U(1, 10)
h^-_t	U(10, 20)

* $P_{max}=R_{max}+O_{max}+S_{max}$
* $\delta = P_{max}/2, (3P_{max})/4$

Table 4. GA parameter settings.

				$P_C = 0.7$		$P_M = 0.295$			$P_R = 0.005$
G	ε	λ	*Pop_Size*	P_{SPC}	P_{AC}	P_{EM}	P_{IM}	P_{AM}	
200	2	10000	150	0.7	0.3	0.34	0.34	0.32	

Table 5. Comparison between GA and optimal solutions.

No.	Period No.	Optimal Solution		GA Solution			GAP (%)
		OFV	CPU Time (Sec.)	*Pop_Size*	OFV	CPU Time (Sec.)	
1	12	583864	1	100	583921	110	0.009763
2	12			150	583864	136	0.000000
3	12			200	584157	162	0.050183
4	30	1586764	10	100	1595944	154	0.578536
5	30			150	1591023	151	0.268408
6	30			200	1595800	335	0.569461
7	40	2093220	60	150	2107304	226	0.672839
8	45	[2306340, 2306690]	1200	150	2323030	319	0.723657
9	50	[2630010,2630090]	> 3600	150	2640735	497	0.407793
						Average:	0.364516

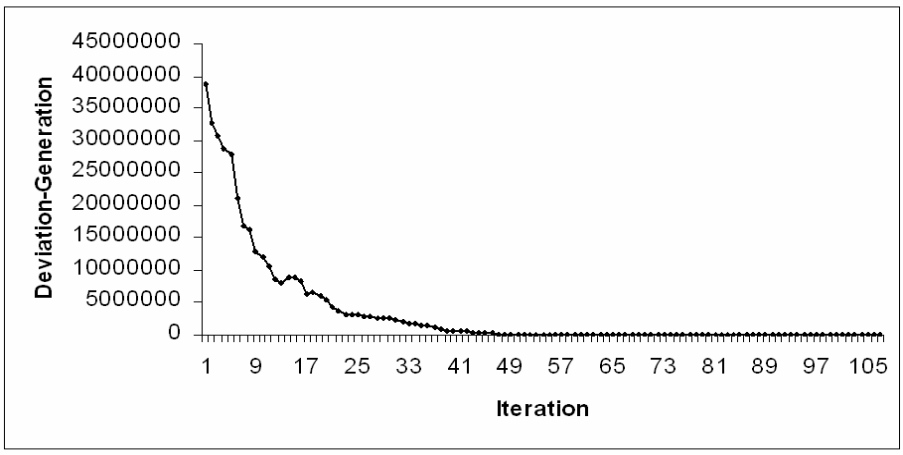

Figure 9. Convergence of GA with respect to the deviation of generation (Problem 5 from Table 5).

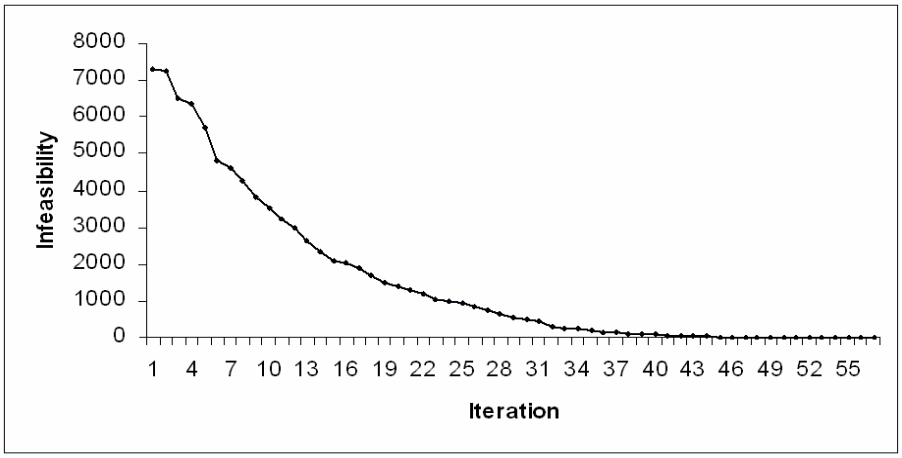

Figure 10. Infeasibility reducing rate (Problem 5 from Table 5).

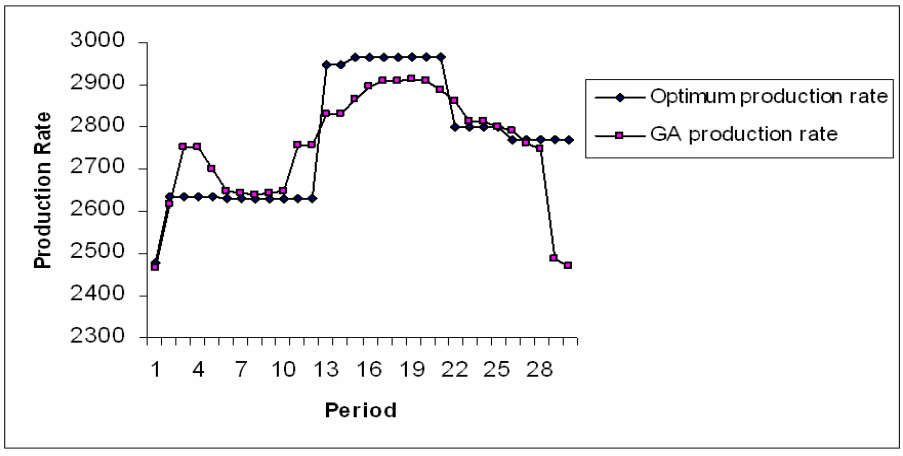

Figure 11. Comparison between optimal and GA production rate (Problem 5 from Table 5).

5. Conclusion

In this paper, we have formulated a single-item resource-constrained aggregate production planning (APP) problem with the most of realistic costs. To solve such a NP-hard problem, we propose an efficient metaheuristic method based on genetic algorithms (GAs) with novel operators and selection strategy. Due to the lack of shortage at the end of horizon planning, we design an extended version of single point crossover, arithmetic crossover, and arithmetic mutation operators required in the proposed GA. The obtained results show the efficiency of the proposed GA and designed operators with respect to the objective function value and CPU time. The efficiency of the proposed GA will be compared with other evolutionary algorithms such as simulated annealing, tabu search, scatter search, ant colony and particle swarm optimization, and memetic algorithm for future researches. In addition, a multi-item resource-constrained APP can be considered for the further research.

References

[1] Ari, E. A. and Axsater, S., 1988, Disaggregation under uncertainty in hierarchical production planning. *European Journal of Operational Research*, 35, 182-186.

[2] Axsater, S., 1986, On the feasibility of aggregate production plans. *Operations Research*, 34, 796-800.

[3] Axsater, S. and Jonsson, H., 1984, Aggregation and disaggregation in hierarchical production planning. *European Journal of Operational Research*, 17, 338-350.

[4] Bitran, G. R., Haas, E. A. and Hax, A. C., 1981, Hierarchical production planning: a single stage system. *Operations Research*, 29, 717-743.

[5] Bitran, G. R. and Hax, A. C. 1971, On the design of hierarchical production planning systems. *Decision Science*, 8, 28-55.

[6] Bowman, E. H., 1956, Production scheduling by the transportation method of linear programming. *Operations Research*, 4, 100–103.

[7] Bowman, E. H., 1963, Consistency and optimality in managerial decision making. *Management Science*, 9, 310–321.

[8] Charnes, A. and Cooper, W. W., 1961, *Management Models and Industrial Applications of Linear Programming*. John Wiley & Sons, NY.

[9] Dilworth, J. B., 1993, *Production and Operations Management: Manufacturing and Services.* 5th Edition, McGraw-Hill Co.

[10] Fogarty, Blackstone, and Hoffamnn, 1991, *Production and Inventory Management*. 2nd Edition, South Western Publishing.

[11] Gaither, N., 1996, *Production and Operations Management*. 7th Edition, Buxbury Press.

[12] Gal, T., 1979, *Postoptimal Analysis, Parametric Programming and Related Topics*. McGraw-Hill Co., NY.

[13] Gelders, L. F. and Van Wassenhove, L. N., 1981, Production planning: a review. *European Journal of Operational Research*, 7, 101-110.

[14] Gen, M. and Cheng, R., 1997, *Genetic algorithms and engineering design*. John Wiley & Sons, NY.

[15] Gen, M. and Cheng, R., 2000, *Genetic Algorithms and Engineering Optimization*. John Wiley & Sons, NY.

[16] Gilgeous, V., 1989, Modeling realism in aggregate planning: a goal search approach. *International Journal of Production Research*, 27, 1179-1193.

[17] Goldberg, D. E., 1989, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing.

[18] Hax, A. C., 1973, Aggregate capacity planning: a review. *Technical Report No. 85* (AD 711 298), Springfield, VA, National Technical Information Service.

[19] Holland, J. H., 1992, *Adaptation in Natural and Artificial Systems: An Introductory Aanalysis with Applications to Biology, Control, and Artificial Intelligence*. 2nd Edition, MIT Press, Cambridge.

[20] Holt, C. C., Modigliani, F. and Simon, H. A., 1955, Linear decision rule for production and employment scheduling. *Management Science*, 2, 1–30.

[21] Holt, C. C., Modigliani, F., Muth, I. F. and Simon, H. A., 1960, *Planning Production, Inventory and Work Force*. Prentice-Hall Co., NJ.

[22] Johnson, L. A. and Montgomery, D. C., 1974, *Operations Research in Production Planning, Scheduling, and Inventory Control*, John Wiley & Sons, NY.

[23] Kleiner, P. and Kleiner, B. H., 1995, Aggregate planning today. *Work Study*, 44 (3), 4-7.

[24] Michalewicz, Z., 1996, *Genetic Algorithms + Data Structures = Evolution Programs*. 3rd Edition, Springer Publishing.

[25] Moder, J. and Elmaghraby, S. E., 1978, Aggregate production planning. In: *Handbook of Operations Research*, Van Nostrand Reinhold, NK.

[26] Pradenas, L. and Penailillo, F., 2004, Aggregate production planning problem, a new algorithm. *Electronic Notes in Discrete Mathematics*, 18, 193–199.

[27] Saad, G., 1982, An overview of production planning model: structure classification and empirical assessment. *International Journal of Production Research*, 20, 105–114.

[28] Silver, E. A., Pyke, D. F. and Peterson, R., 1998, *Inventory Management and Production planning and Scheduling*. John Wiley & Sons, NY.

[29] Singhal, K. and Adlakha, V., 1989, Cost and shortage trade-offs in aggregate production planning. *Decision Sciences*, 20, 158–165.

[30] Tavakkoli-Moghaddam, R. and Biyabani, H., 2004, *The Use of Genetic Algorithms for Aggregate Production Planning*. Proceedings of the 4th International Symposium on Intelligent Manufacturing Systems, Sakarya, Turkey, September 6-8, 683-692.

[31] Vollman, T. E., 1992, *Management Planning and Control System*. 3rd Edition, IRWIN Publication.

[32] Wang, D. and Fang S., 1997, A genetics-based approach for aggregated production planning in a fuzzy environment. *IEEE Transactions On Systems, Man, and Cybernetics*, 27 (5), 636-645.

[33] Wright, A. H., 1991, *Genetic Algorithms for Real Parameter Optimization.* In: Foundations of Genetic Algorithms, G. J. E. Rawlins (ed.), San Mateo, CA: Morgan-Kaufmann, 205–218.