Indian Journal of Fundamental and Applied Life Sciences ISSN: 2231– 6345 (Online)
An Open Access, Online International Journal Available at www.cibtech.org/sp.ed/jls/2014/04/jls.htm
2014 Vol. 4 (S4), pp. 2004-2014/Shadrokh and Najafi
Research Article

MODELING A MODIFIED GENETIC ALGORITHM FOR PROJECT SCHEDULING SUBJECT TO PERISHABLE, NONRENEWABLE RESOURCES

Shahram Shadrokh and *Hadi Najafi S.M.
Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran
**Author for Correspondence*

ABSTRACT

This paper attempts to provide a solution for project scheduling problems subject to perishable, nonrenewable resources. The objective is to schedule activities, procurement and ordering so as to minimize the costs. We designed and employed an approach based on genetic algorithm (GA) to solve the problem. Chromosome structure is defined in indirect form with a size considerably smaller than all decision variables of the model. With a few genes, one can equate one point of the question space. In this algorithm we have offered an innovative method to generate initial population. Also, crossover and mutation operators use a combination of several methods with consideration to chromosome structure. Analysis of the results indicates that the proposed algorithm not only has a good convergence and consistency but also has a good efficiency in medium-size problems and finds optimal solution within a short time**.**

Keywords**:** *Resource-Constrained Project Scheduling, Perishable Nonrenewable Resources, Procurement Scheduling, Modified Genetic Algorithm, Ordering Costs*

INTRODUCTION

Nonrenewable resources are those resources which are accessible at any time during a project and are limited to the entire duration, not to a certain period of the project. A good example of nonrenewable resources is money. At any time during a project, we have access to the entire money allocated to the project. In other words, money is limited to the entire duration of project. Raw materials and energy are other examples of such resources (Demeulemeester, 2002). One of the nonrenewable resources is perishable resources. Perishable resources are those items which are subject to decay and destruction within a short time. Scientifically, perishable product is defined as an item which loses its value and efficiency if not properly maintained, stored or transported or if not used within a specified period of time (Business Dictionary, 2013). Perishable items are the main source of income in food industry. According to an international study on supermarkets in 2005, perishable items constituted more than 54% of total sales and approximately 57% of inventories (Kouki, 2010). Therefore, efficient management of perishable items results in the achievement of competitive advantage. In 2005, Roberti reported that approximately 10% of perishable items spoil and decay before consumption (Roberti and Scheer, 2005). This is not limited to foodstuff. In 2006, approximately 4.6% of collected blood in the United States has decayed before injection to applicants.

The U.S Federal Government keeps huge amounts of drug as Strategic National Stockpile (SNS) to protect public health in emergency occasions. Efficient management and control of these perishable resources not only reduces SNS costs but also considerably enhances public safety (Shen *et al.*, 2010). The resources of a project may be perishable too. Efficient management and timely supply of these resources not only reduces project costs but also speeds up the project. In other words, when some resources of a project spoil and decay, not only some part of the budget has to be spent for repurchasing them but also the activities using these resources are interrupted. The project slows down if the lost resources are not procured in the right time or if the activities involved are critical to the project. This research consists of five parts. Part 2 presents the review of literature. Part 3 provides the definition of subject and modeling. Part 4 presents and approaches to solve the problems and finally Part 5 provides a comparison and conclusion and gives some recommendations for future studies.

Literature Review

Project scheduling subject to perishable resources is a new subject. So far, there has been no study on perishable resources in the literature of project scheduling. Research on perishable items has been limited to inventory management and control of perishable products and there is no source of study on project scheduling subject to perishable resources. The only point addressed in the literature is that certain resources of a project may be perishable and should be used within a specified time (Brucker and Knust, 2012). Chande *et al.,* (2005) made a study on inventory management of perishable items and dynamic pricing. In highly price-sensitive markets, simultaneous combination of price promotions and appropriate replacement of goods greatly helps to control market establishment costs. In a supply chain in which perishable items are transported, inventory management is a complicated issue and management of a product based on dynamic pricing is even more complicated. Radio Frequency Identification (RFID) has made it easier to manage perishable items. Chande *et al.,* (2005) provided an algorithm for inventory management of perishable items based on dynamic pricing. Their objective was to determine optimal times for proposing research and for ordering and filling inventory (Chande *et al.,* 2005).

Deniz *et al.,* (2004) made a study on the effect of replacement of goods in the management of perishable items. To do so, they considered a discrete supply chain and increased the inventory of a certain item using two distinct policies: one based on inventory level and the other based on the need to new goods. In this research they compared upward and downward methods in the replacement of goods, concluding that goods remain fresher in downward method (Deniz *et al.,* 2004). Among other studies concerning inventory management of perishable items we can mention (Deniz, 2007; Kouki, 2010; Naso *et al.,* 2007; Shen *et al.,* 2010). In addition to those mentioned earlier, there are some other works in the literature. Ballestin (2007) presented a genetic approach to solve the problem of resource renting in RCPSP (Ballestin, 2007). Trautmann and Schwindt (2005) carried out a research on short-term planning for production and presented an algorithm to solve the problem (Trautmann and Schwindt, 2005). Also, Cesta *et al.,* (2002) studied RCPSP and presented an approach to create a limitation-based time schedule (Cesta and Smith). Leon and Balakrishnan (1995) studied the adaptability of RCPSP using neighborhood method (Leon and Balakrishnan, 1995). Blazewicz *et al.,* (1996) made a study on scheduling in computer production systems (Blazewicz *et al.,* 1996). Silva *et al.,* (2008) presented an effective algorithm to solve DRCPSP (Silva *et al.,* 2008). Chaleshtari *et al.,* (2011) developed a branch and bound algorithm for RCPSP-Cu which resources of problem are cumulative (Chaleshtari and Shadrokh, 2011). RCPSP has also been studied by De Reyck *et al.,* (1999), Ginis (2002), Kuster *et al.,* (2010), Laborie (2003) and Neumann and Schwindt (2003).

Subject Modeling

This study is going to address perishable resources (1, …, PR). It should be said again that perishable resources are some kind of nonrenewable resources which decay and spoil within a specified period of time. The network of activities is in the form of activity in node or AoN and two activities of 0 and n+l have been considered as start and end of the project. We assume that the period in which each perishable item decays is a specified and fixed value and each perishable item is procured once it decays or runs out of stock. An objective of this study is to determine optimal ordering amount for each perishable item during different periods with minimum cost.

Each perishable item has an ordering cost consisting of a fixed ordering cost and a variable cost for purchase. Maintenance cost has also been taken into account. Our objective is to minimize ordering and maintenance costs of perishable resources. We assume that delivery takes no time and the ordered item is available once it is ordered. We also assume that perishable resources are consumed in an activity just in the amount determined for that activity and that surplus and decayed resources are accumulated and eradicated at the end of project. Therefore, the project has maintenance cost for the surplus items and eradication cost has been considered to be zero.

To deal with cost and time at the same time, we incorporated cost minimization into target function and total project duration into constraints. Ordering amount of each item is another variable which we have considered to be continuous.

Parameters and Variables of the Model: Before describing the model, let's introduce parameters and decision variables used in modeling:

- **T:** Maximum duration of project
- **M_j:** Large number (Big-M) relating to each perishable item j
- **t**: Time (Period)
- **d_j:** Activity j execution time
- **P_k:** Set of prerequisite activities of activity k
- **PR:** Number of perishable resources
- **A_j:** Fixed ordering cost of perishable item j
- **C_j:** Variable ordering cost of each unit of perishable item j
- **h_j:** Maintenance cost of each unit of perishable item j in each time unit
- **P_{ij}:** Amount of activity i use of perishable item j which is consumed in the first period of activity execution.
- **q_{jt}:** Amount of ordering perishable item j in time t
- **EX_j:** Expiry period of perishable item j
- **X_{it}:** If activity i starts in time t, 1, otherwise 0.
- **Y_{jt}**: If perishable item j is ordered in time t, 1, otherwise 0.
- **Z_{ijt}**: If activity i uses perishable item j ordered in time t, 1, otherwise 0.

Mathematical Model of the Subject: Our objective is to minimize project costs which include ordering and maintenance costs of perishable resources. Ordering cost itself consists of two parts: fixed cost of ordering and variable cost of ordering and purchasing each unit of perishable resources. Project scheduling subject to perishable, non-renewable resources is modeled as follows:

$$Min\left\{\sum_{j=1}^{PR}\sum_{t=0}^{T}\left[y_{jt}.A_j+q_{jt}.C_j+\left[\sum_{\tau=0}^{t}q_{j\tau}-\sum_{i=0}^{n+1}\sum_{\tau=0}^{t}x_{i\tau}.p_{ij}\right].h_j\right]\right\} \tag{1}$$

$$\sum_{t=0}^{T}x_{it}=1 \quad ;\forall i \tag{2}$$

This equation shows that each activity starts only once.

$$\sum_{t=0}^{T}(t+d_i).x_{it}\le\sum_{t=0}^{T}t.x_{jt} \quad ;\forall j \quad , \quad i\in p_j \tag{3}$$

This constraint is used for compliance with prerequisites and post requisites of activities according to the network of project activities.

$$\sum_{t=0}^{T}t.x_{n+1,t}\le T \tag{4}$$

While this constraint exists in the model, total duration of project has also been taken into consideration. Although the objective is not to minimize project duration, we will not let project duration to become more than expected. T value shows the maximum.

$$\sum_{i=0}^{n+1}z_{ijt}.p_{ij}\le q_{jt} \quad ;\forall j,t \tag{5}$$

This equation shows that the ordered amount of each perishable item must always be bigger than or equal to the amount that project activities use that item in that time. This constraint must exist in each moment of project duration and for each perishable item.

$$x_{it}\le\sum_{\tau=t-EX_j+1}^{t}z_{ij\tau} \quad ;\forall i,j,t \tag{6}$$

This constraint explains the relationship between activities and perishable resources and determines which resources must be used by which activities.

$$q_{jt} \leq M_j . y_{jt} \qquad ;\forall j,t \tag{7}$$

This constraint establishes the relation between variables 0 and 1 of ordering perishable resources and the ordered amount of the perishable item.

$$x_{it}, y_{jt}, z_{ijt} \in \{0,1\}; q_{jt} \geq 0 \qquad ;\forall i,j,t \tag{8}$$

This shows the variables, all of which are 0 or 1. The variable of ordered amount is a non-negative continuous variable.

As you can see, modeling has used large number. While theoretically any large number seems to be justifiable, in practice and in calculations, not any large number would be appropriate because there is a limitation in the number of decimal places and bit lengths of a number, whether when we are accurately solving the problem by software or when we are solving it by designed algorithm via computer. In other words, that a number is larger does not mean that the model is better and reaches the answer. According to what was said above, the smaller Big-M relating to perishable resources results to the smaller probability of producing an inappropriate answer. On the other hand, minimizing the number may cause the issue to become unjustifiable. So we have to find the smallest large number relating to each perishable item and put it in the model. For this purpose we consider the large number relating to each perishable item to be total use of that item in different activities. It is calculated through the following equation:

$$M_j = \sum_{i=0}^{n+1} p_{ij} \tag{9}$$

Proposed Algorithm

The proposed approach is based on genetic algorithm. We have attempted to use problem structure to find better answers. Every effort has been made to design chromosome based on problem structure with minimum number of genes.

Chromosome Structure: In the designed algorithm, the number of genes of a chromosome is calculated by the following formula:

$$L_c = nActivity + nPR \times T \tag{10}$$

In other words, the number of genes of a chromosome equals the number of activities plus the product of the number of perishable resources in project completion time. The values of the first part of chromosome, which equals the number of activities, indicate start time of each activity.The genes of the second part indicate the ordering amount of each perishable item in each period of time. So, we will have genes in the number of periods existing in project (T) for each perishable item.

Solution Algorithm: Before explaining the proposed algorithm, it's essential to introduce the algorithm parameters that are used in this paper. **Children Percentage** equals to the number of children in each iteration, the population of which is expressed as a percentage of the initial population. **Mutation Percentage** is the number of chromosome selected from the population to perform the mutation operation on them which is expressed as a percentage of the initial population and the population of children. **Mutation probability** is the probability of mutation of selected gene.

In the proposed algorithm, we first generate an initial population in a random manner. This population consists of 200 chromosomes. We randomly generate some numbers between 1 and T for the first part and some numbers between 1 and M_j for the second part of each chromosome. After generating the initial population, we calculate the fitness of each chromosome. The children percentage is considered to be 40%, mutation percentage is considered to be 10% and mutation probability is considered to be 6%.

Development of Initial Population: We make a two-dimensional array with dimension of 2 × number of activities and put $T - d_i$ value as the latest start time of activity i in the first line and in the element relating to the activity with no post requisite. In this equation, $d_i$is duration of activity i. Then we put minimum of $x_{if} - d_i$ in the elements of the first line relating to activity i which has post requisite. x_{if} is the latest start time of post requisite activity i. This way the latest start time of each activity is determined. For the activity with no post requisite, we put value 1 as the earliest start time of activity in the element of second line relating to that activity. Then we put maximum of x_{pi} + d_p for activity i which has post requisite. In this equation, $x_{pi}$and d_p are respectively earliest start time and duration of post requisite activity i. This way the earliest start time of each activity is determined. So we have two numbers as earliest and latest start times for each activity. To generate initial population from the above array, a random number between the earliest and latest start times of an activity is selected and put in the related gene in order for answer to be closer to feasible space.

Fitness Function: Fitness amount of an answer equals total target function of the main model plus total fine of that answer's violation of three constraint categories of equations 3, 5 and 6.

$$f_i = z_i + \sum_{j=1}^{m} \alpha_j \cdot v_{ij} = z_i + \sum_{j=1}^{m} \left[\alpha_j \cdot Max\left(\frac{g_{ij}}{g_{0j}} - 1, 0 \right) \right] \qquad (11)$$

In this equation, g_{ij} is constraint j amount for answer *i* and g_{oj} is right side amount of this constraint. By multiplying violation amount of this constraint by violation value, the share of this constraint in fitness amount of answer *i* is determined. Therefore, if the model has m constraints, fitness of answer *i* is calculated by the equation.

Roulette Wheel Selection: The method used in the proposed algorithm is roulette wheel selection. In roulette wheel selection; a random number between 0 and 1 is generated to choose a chromosome. In any interval that the number is located, a chromosome corresponding to that number is selected. This is exactly similar to what roulette wheel does. We pick some pieces equal to fitness of chromosomes and then rotate the wheel. When the wheel stops, we select the chromosome which corresponds to that section of the circle in which roulette wheel indicator is located. This selection is used to select parents for crossover function.

Multipoint Crossover: One of the most important parts of GA is crossover. The method used in the proposed algorithm to combine genes of parents and produce child is relocation or multipoint crossover. As described in chromosome structure, we have two categories of genes. For the first part, a random gene between 1and T is selected (E.g. x is selected) and for first child we select genes from 1 to x genes of first parent and rest of genes will select from second parent. For the second part of chromosome, we choose genes relate to each perishable resources from first parent and next one from another parent.

Mutation: As chromosome structure has two parts, mutation is also divided into two parts. One part of mutation relates to genes of the first part of chromosome and start time of activities and the other part relates to genes of the second part and the ordering amounts of perishable resources in each time. For each repetition of mutation operator, one of the chromosomes of the present population (population of new generation children plus the best ones of previous generation) is randomly selected and then its genes are mutated.

For both parts of chromosome we used some methods and some effective methods are as below. For random selected gene(s), one of these methods with probability of 6% (Mutation probability) will be performed. Mutation in first part of chromosome:

1. Selected gene's value replaces with a random number between 1 and T.
2. If selected gene's value is greater than 1, its value will minus 1 unit.
3. If selected gene's value is less than T, its value will add by 1 unit.
4. Two random genes will select and their values will replace.
5. Two random genes will select, if first one is not related to a prerequisite activity of second gene, then, the second one value will be equal to first gene. Otherwise if first gene is prerequisite activity of second gene, then, the second one value will be equal to first gene value plus its duration (d_i).

Mutation in second part of chromosome:

1. For each perishable resource j a random gene is selected and a random number between 0 and M_j will produce and replace in that gene.

2. For each perishable resource a random gene is selected and if its value is greater than 0, next genes till expiry date of that resource will be replace with 0.
3. For each perishable resource a random gene is selected and if its value is greater than 1, it value will minus 1 unit.
4. For each perishable resource j a random gene is selected and if its value is less than M_j, its value will add by 1 unit.

It's necessary to say that in all iteration of algorithm the probability of selection of above choices is not equal and as algorithm moves ahead, in first part probability of method 4 will decrease and method 5 will increase and in second part, probability of selection of method 3 and 4 will increase and in all iterations this probabilities depend on mutation probability.

RESULTS AND DISCUSSION

Results

Algorithm Parameters: In genetic algorithm, many parameters affect the process of solution and search in answer space. Among these parameters we can mention the number of initial population, percentage of children, percentage of mutation, mutation probability and penalties of violation from constraints in calculation of chromosome fitness. During the tuning of algorithm we execute the algorithm 50 times for each value. Table1 illustrates the summary of results and the average time of reaching an optimal solution for each parameter.

Table 1: Algorithm Parameters Tuning

Initial population size	Time(sec)	Children percentage	Time (sec)	Mutation percentage	Time (sec)	Mutation probability	Time (sec)
500	127.21	60	103.73	60	76.32	0.15	325.44
400	85.97	50	84.18	50	69.67	0.1	191.28
300	63.59	45	83.05	40	97.84	0.09	133.13
200	50.45	40	51.02	30	95.25	0.08	184.99
100	90.10	35	72.27	25	51.39	0.07	313.54
50	102.59	30	68.41	20	76.73	0.06	37.91
10	124.22	20	80.30	15	49.53	0.055	160.54
				10	38.14	0.05	41.77
				5	154.34	0.04	90.26
						0.03	124.25
						0.02	82.70
						0.01	153.19

The best value for the parameter of initial population size is 200.The best value for the parameter of children percentage is 40%. The best value for the parameter of mutation percentage is 10%.The best value found for mutation probability is 6%.

Fitness Function: The best combination for fitness function is penalty of $10^7$for the constraint of equation 3and penalty of $10^5$for the constraint of equation 5 and 6.

Crossover: In subject of algorithm tuning, we proposed four types of crossover and we execute the algorithm 50 times for each method. Table 2 shows the summary of average time of reaching an optimal solution for each method. The results indicate that there is not a considerable difference between the crossover methods.

Table 2: Solution Time Changes of Genetic Algorithm with Change of Crossover Method

Crossover Method	Time (sec)
Two-point method in the form of two random points of the genes of first and second parts of the chromosome	51.23
Multipoint method in the form of selection of random points relating to each item in the second part of the chromosome	39.21
Multipoint method in the form of selection of a random point of the first part and the points relating to each perishable item in the second part of chromosome	38.47
Multipoint method in the form of selection of a number of random points in the entire chromosome	43.44

Comparison of Results: To develop a case of project scheduling subject to perishable, nonrenewable resources, we need many parameters. Among parameters of the model, three parameters of number of activities, number of perishable resources and maximum completion time of project have a fundamental role (E.g. A sample problem p_1 is shown as p_1(nActivity, nPR, T)). The number of other parameters depends on these three parameters. In order to evaluate our algorithm, we generated 20 small-size, 40 medium-size and 20 big-size cases and solved them using CPLEX 12.2. Algorithm has been coded and executed on C++ platform on a PC with Core 2 Duo 2.20 GHz CPU and 2.00 GBs RAM.
To develop the cases, sample problems of the project scheduling library (PSPLIB) (Kolisch and Sprecher, 1996) have been used. As RCPSP problem instances of the library are only subject to the renewable resources, MRCPSP problem instances have been used in order to have data of the nonrenewable resources as well. Using the following method, each MRCPSP instance has been transformed to a perishable RCPSP, the subject of this paper, instances:

- For each activity one mode is randomly selected among the modes of that activity.
- The activity network and precedence relations and job durations are same.
- The horizon of project (T) is randomly created due to feasibility of instance.
- Other characteristic of perishable item (A_j, C_j, h_j, EX_j and p_{ij}) is randomly generated.

We use three sets of multimode project scheduling problems from the PSPLIB, j10, j20 and j30 for experiments and convert them to perishable instances using the above procedure.

Table 3: Some examples of final answers and solution time using CPLEX 12.2 for generated instances

Problem	Answer	Time	Problem	Answer	Time
2(10,4,18)	4755	00:00:00.64	8(20,8,32)	17320	08:17:02.31
61(10,6,23)	6950	00:00:01.42	10(20,8,32)	19232	08:12:42.36
58(10,8,26)	8179	00:00:02.78	12(20,8,50)	13470	*
3(20,4,29)	15482	00:06:25.09	27(30,8,60)	34502	00:46:41.62
5(20,6,30)	16702	01:13:05.07	28(30,8,61)	39493	07:13:29.08
6(20,6,32)	18945	01:36:43.68	32(30,8,80)	13568	*
63(20,8,30)	16974	00:52:45.32	53(30,10,100)	12823	*

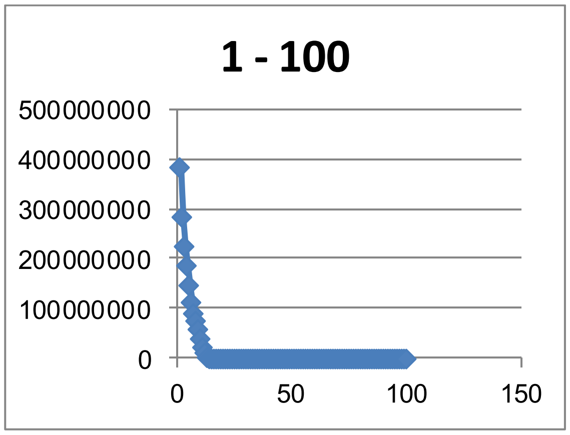
In table 3 mark * means that CPLEX has failed to reach an optimal answer within 12 hours. For such cases, the value shown in answer column is the best answer that the software has reached after 12 hours. In other cases the value in answer column is the optimal solution of that instance.
After developing and accurately solving introduced cases, we put our algorithm to test. For this purpose we solved 80 cases, consisting of small and medium and big-sized ones, by the algorithm and repeated each case for 50 times.
In execution of the proposed algorithm we have two conditions for termination. First condition is reaching a predetermined answer. It's useful in instances that we have its best answer. The second

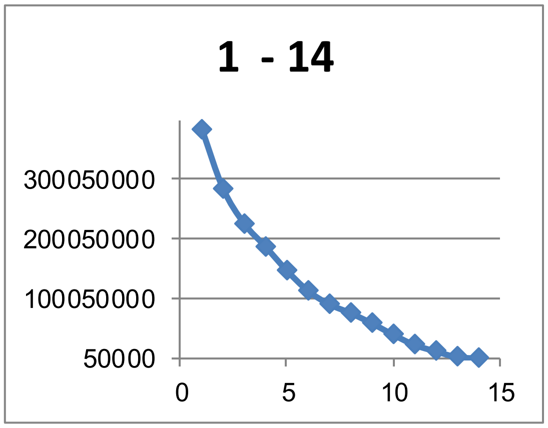
condition is change in less than 0.1 percent in average fitness of best population in ten consecutive iterations.
After executing the algorithm 50 times for each case, we reached the following results. The fourth and fifth columns of the table.4 show the average of best answer and the average of stoppage time of algorithm for each case.

Table 4: Comparison of a number of final answers and solution time of the cases by the proposed algorithm and using CPLEX 12.2 that has been solved and we have optimal solutions of them

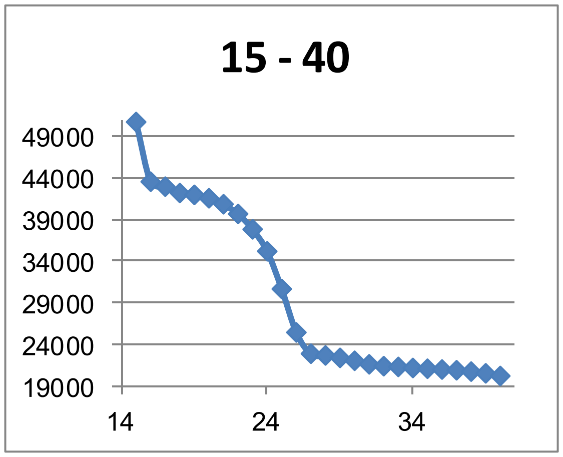
Problem	Optimal solution	CPLEX solution time (sec)	Genetic solution	Genetic time (sec)	Status	Percentage of deviation from optimal solution
2(10,4,18)	4755	0.64	4755	1.59	*	-
61(10,6,23)	6950	1.42	6950	1.84	*	-
3(20,4,29)	15482	385.09	15482	37.18	*	-
6(20,6,32)	18945	5803.68	18945	87.26	*	-
63(20,8,30)	16974	3165.32	16974	71.05	*	-
8(20,8,32)	17320	29822.31	17632	286.57		1.80
10(20,8,32)	19232	29562.36	19840	354.08		3.16
9(20,8,32)	21150	27985.49	21310	290.48		0.76
27(30,8,60)	34502	2801.62	34502	111.39	*	-
28(30,8,61)	39493	26009.08	42678	295.35		8.06
31(30,8,61)	38969	26772.33	40472	310.39		3.86

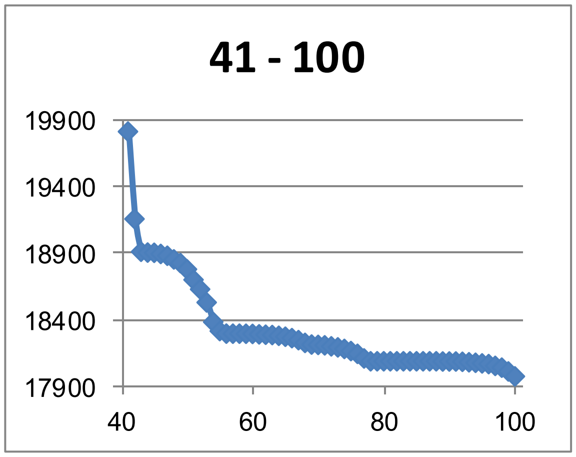
In table 4, mark * in status column means that the proposed genetic algorithm has reached an optimal solution. In some cases, algorithm has not reached an optimal solution and there is a little difference between the best solution found and optimal solution. In cases whose final solution deviates from optimal answer, average deviation of the best solution is 4.989%. In all cases, both optimal and non-optimal, the average deviation of final solution of genetic algorithm from optimal solution is 2.32%.
In cases which have reached an optimal solution, there is 97% improvement in solution time. The above calculations and results are based on average of the best solution produced in the last repetition of algorithm. The improvement of solution time and the percentage of deviation of final solution from optimal solution demonstrate the consistency of the proposed algorithm.
Another factor to be investigated is convergence of algorithm. To do so, we calculated average fitness of the chromosomes created in each generation in 50 repetitions of the algorithm for each case. The average fitness of the first generation was equal to the average fitness of initial population. Then we calculated the average fitness of the second generation, and so on. We performed this for the first 100 generations of each repetition of algorithm.Figure.1 illustrates this behavior of algorithm.





(a) **(b)**





(c) **(d)**

Figure 1: Average fitness of best population in generation 1 to 100. (a) 1- 100, (b) 1-14, (c) 15- 40, (d) 41-100

For each repetition, we separated the best chromosomes in the amount of initial population from the last generation. We saw that in cases which had reached an optimal solution, the entire chromosomes were in feasible space in all repetitions. Also, in cases which had not reached an optimal solution in genetic algorithm, more than 92.2% of chromosomes were in feasible space. Furthermore, in cases which had reached an optimal solution, approximately 98% of chromosomes had less than 5% deviation from the optimal solution and in cases which had not reached an optimal solution, 54.7% of solutions had less than 5% deviation from the optimal solution.

Table 5: Comparing the performance of algorithm with respect of problem's size

	j10	j20	j30
Deviation from optimal solution	0	1.4	5.1
Improvement in solution time	-78.7	96.2	98.1
Cases reached optimal solution	100	87.5	60

Table.5 shows the results group by problem's size. In this table all values are in form of percentage and results are a comparison between CPLEX 12.2 solution and proposed algorithm. In j10 problems however proposed algorithm solved all cases bat its solution time is greater than CPLEX because of size of problems that exact software will solve it less than 2 seconds. In j20 problems, proposed algorithm solved 87.5% of cases and reached optimal solution with a significant improvement in solution time. Finally in j30 problems however improvement in time is significant but 40% of cases did not reached the optimal solution.

Conclusion

Comparison of the results demonstrates the efficiency of the proposed algorithm in medium size problems (j20). It is not rational to solve very small problems by the proposed algorithm because precises of tware can find their optimal solutions in a fraction of a second (although we solved such problems by the proposed algorithm and found optimal solution in a longer time). Therefore, the proposed algorithm is more efficient in medium size problems than in very small and very big problems. Comparison of the results also demonstrates that the proposed algorithm has an acceptable convergence and consistency. Genetic algorithm is based on a number of parameters and operators. When we were changing these parameters and operators to improve the algorithm, we found out that each has a significant role in the process of reaching a final solution. Therefore, improvement of each parameter will result in improvement of the algorithm. The effect of these improvements can be easily seen when problem size increases.

The only effect of parameter T on the project is that it makes the problem feasible or infeasible. However it does not have any effect on optimal solution but has a great effect on number of variables of model. To include the effect of maximum completion time of project and make the model more realistic, we have to incorporate time value of money and interest rate in the modeling. That is why we add the following parameter to the model as suggestion for future studies.

Effective Interest Rate: To incorporate time value of money, we should take effective interest rate into consideration. This requires that the relationship between t and year be identified. The effective interest rate of each time unit of the project is calculated by the following equation:

$$r_e = \left(1 + \frac{r}{n}\right)^n - 1 \tag{12}$$

In the equation.12, n represents the relationship between time unit of the project and time unit of year. In the other words, if time unit of the project is day, n is 365. Now we have to change Mathematical Model of the subject, so that interest rate is taken into consideration.

Since we have included time value of money, completion time of project (T) automatically affects objective function of the problem. Considering the type of objective function, purchase scheduling in final solution is expected to be in such a manner that resources are ordered as late as possible. It is recommended to analyze project scheduling model subject to perishable resources and interest rate and to compare its optimal scheduling with Mathematical Model of this paper which has not taken interest rate into consideration.

Finding Appropriate Lower Bound: In NP-hard and big size problems, it is not possible to find an optimal solution in an appropriate time. In the other words, when solving such problems with a heuristic or meta-heuristic approach, we cannot evaluate the performance of our proposed approach if we do not have an optimal solution of the problem. This was obviously seen in the previous section and in problems in which we did not have an optimal solution. If an appropriate lower bound is generated for such problems, we can evaluate the performance of heuristic or meta-heuristic approaches. In our study, we used lagrangian relaxation method to generate lower bound and the obtained lower bound had a very bad quality with 56% deviation from optimal solution. It is recommended that an appropriate method be developed to generate lower bound with good quality for project scheduling subject to perishable resources.

REFERENCES

Blazewicz J, Ecker KH, Schmidt G and Weglarz J (1996). *Scheduling in Computer and Manufacturing Systems* (Elsevier).

Brucker P and Knust S (2012). *Complex Scheduling* (Springer).
Business Dictionary (2013). Perishable. BusinessDictionary.com. WebFinance, Inc. Available: http://www.businessdictionary.com/definition/perishable.html (accessed: November 24, 2013).
Cesta A, Oddi A and Smith SF (2002). A constraint-based method for project scheduling with time windows. *Journal of Heuristics* **8**(1) 109-136.
Chaleshtarti AS and Shadrokh S (2011). Branch and Bound Algorithms for Resource Constrained Project Scheduling Problem Subject to Cumulative Resources. Paper presented at the Information Management, Innovation Management and Industrial Engineering (ICIII), 2011 International Conference on.
Chande A, Dhekane S, Hemachandra N and Rangaraj N (2005). Perishable inventory management and dynamic pricing using RFID technology. *Sadhana* **30**(2-3) 445-462.
Demeulemeester EL (2002). *Project Scheduling: a Research Handbook* (Springer) **102**.
Deniz B (2007). *Essays on perishable inventory management*: ProQuest.
Deniz B, Scheller-Wolf A and Karaesmen I (2004). *Managing inventories of perishable goods: the effect of substitution*: GSIA Working Paper# 2004-E55, Carnegie-Mellon University, Pittsburgh, PA.
Ginis R (2002). *Automating Resource Management for Distributed Business Processes*. California Institute of Technology.
Herbots J, Herroelen W and Leus R (2007). Dynamic order acceptance and capacity planning on a single bottleneck resource. *Naval Research Logistics (NRL)* **54**(8) 874-889.
Herbots J, Herroelen W and Leus R (2010). Single-pass and approximate dynamic-programming algorithms for order acceptance and capacity planning. *Journal of Heuristics* **16**(2) 189-209.
Kouki C (2010). *Perishable Items Inventory Management and the Use of Time Temperature Integrators Technology*.
Kuster J, Jannach D and Friedrich G (2010). Applying local rescheduling in response to schedule disruptions. *Annals of Operations Research* **180**(1) 265-282.
Kolisch R and Sprecher A (1996). PSPLIB – A project scheduling problem library. *European Journal of Operational Research* **96** 205–216.
Laborie P (2003). Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. *Artificial Intelligence* **143**(2) 151-188.
Leon VJ and Balakrishnan R (1995). Strength and adaptability of problem-space based neighborhoods for resource-constrained scheduling. *Operations-Research-Spektrum* **17**(2-3) 173-182.
Naso D, Surico M and Turchiano B (2007). Reactive Scheduling of a Distributed Network for the Supply of Perishable Products. *IEEE Transactions on Automation Science and Engineering* **4**(3) 407-423.
Neumann K and Schwindt C (2003). Project scheduling with inventory constraints. *Mathematical Methods of Operations Research* **56**(3) 513-533.
Orji IM and Wei S (2013). Project Scheduling Under Resource Constraints: A Recent Survey. *International Journal of Engineering* **2**(2).
Roberti M and Scheer F (2005). RFID will help keep perishables fresh. *RFID Journal*.
Shen Z, Dessouky M and Ordóñez F (2010). Perishable inventory management system with a minimum volume constraint. *Journal of the Operational Research Society* **62**(12) 2063-2082.
Silva A, Ochi LS and Santos HG (2008). *New effective algorithm for dynamic resource constrained project scheduling problem. Paper presented at the Proceedings of International Conference on Engineering Optimism (ENGOPT),* Rio de Janeiro, Brazil (June 2008).
Trautmann N and Schwindt C (2005). A MINLP/RCPSP decomposition approach for the short-term planning of batch production. *Computer Aided Chemical Engineering* **20** 1309-1314.