A Genetic Algorithm for the Resource-Constrained Project Scheduling Problem

Erdem Ozleyen
Aegis Project Controls, USA

Abstract

The resource-constrained project scheduling problem (RCPSP) aims to find a schedule of minimum makespan by starting each activity such that resource constraints and precedence constraints are respected. However, as the problem is NP-hard in the strong sense, the performance of exact procedures is limited and can only solve small-sized project networks. In this study a genetic algorithm is proposed for the RCPSP. The proposed genetic algorithm (GA) aims to find near-optimal solutions and also overcomes the poor performance of the exact procedures for large-sized project networks. Experiment results show that the suggested algorithm outperforms the well known commercial software packages.

Keywords: Project management, resource constrained, scheduling, genetic algorithm

1. Introduction

Construction scheduling involves the definition of activities, the estimation of durations for individual activities, establishment of the relations between activities, and the required resources for undertaking activities. Accordingly, project management must decide which resources are going to be used for the execution of a project, must decide on the capacity of the various resource types and must estimate the resource requirements for the project activities. Apparently, if these resources are adequate, then the project could be executed to attain expected project duration. On the other hand, if these resources are limited, then more likely there will be a delay in the project completion time. As a matter of fact, a sufficient schedule, which integrates resources properly, provides competitive benefit to the company during project period.

Scheduling problems involve many types of constraints. Resource constrained project scheduling problem (RCPSP) emerges when there are limits on the availability of resources. Obtaining an adequate solution for the RCPSP is crucial for scheduling and planning of construction projects. Ineffective allocation of resources because of inappropriate scheduling of resource constrained projects (RCP) will increase the project duration and cost considerably. The solution of RCPSP addresses a schedule of minimum duration by assigning a start date to each task such a way that the precedence relations and resource constraints are satisfied.

The RCPSP is one of the most difficult optimization problems. Indeed, RCPSP belongs to the class of the NP-hard problems (Blazewicz et al., 1983) which expresses that

solution time for achieving the optimal solution by using exact methods can be considerable long. Some exact methods exist only for the small projects and also they take more than polynomial time when the project grows or extra resource constraints are added. Therefore, most research studies have been devoted to improve heuristic and meta-heuristic procedures to obtain near-optimal solutions within a polynomial time.

The objective of this study is to present a genetic algorithm which solves RCPSP to produce near-optimal solutions within an acceptable computation time. The proposed algorithm is built by using C++ programming language and proved to conveniently operate on different problem sets.

2. Literature Review

The Resource Constrained Project Scheduling Problem (RCPSP) has been widely studied in the field of scheduling, resulting in a comprehensive variety of optimization techniques. Numerous research studies have focused on development of meta-heuristic procedures to produce near-optimal solutions within a reasonable computation time. The GA maintains a population of solutions that are coded as strings called chromosomes and a measure of their fitness is computed by an agent. Starting from an existing population, each iteration generates new chromosomes by applying operators (like crossover and mutation) to two chosen parents (Montoye-Torres et al., 2010). Genetic algorithms are very important meta-heuristic optimization algorithms but they have very limited hill climbing ability for escaping from local minima and often suffer from premature convergence (Kolisch & Hartmann, 2006).

Several previous studies (Debels & Vanhoucke, 2005; Valls et al., 2005; Mendes et al., 2005; Kim & Ellis, 2008; Seda et al., 2009) provided detailed descriptions of different heuristic methodologies employed to solve the RCPSP. In recent years, combination of different GA mechanisms to improve the performance of algorithm has become popular and several meta-heuristics approaches have been proposed in the literature: scatter search (SS) and electromagnetism theory (Debels et al., 2004), path re-linking strategy and tabu search (Kochetov & Stolyar, 2003), particle swarm optimization (Zhang et al., 2006), hybrid neural approach (Colak et al., 2006). Also a hybrid meta-heuristic procedure has been presented by (Tseng & Chen, 2006) named ANGEL, integrating ant colony optimization (ACO), genetic algorithm (GA) and local search strategy. The hybrid genetic algorithm proposed by (Valls et al., 2008) presents several modifications in the GA concept. These modifications are: a crossover operator particular for the RCPSP, a local enhancement operator that is employed to all generated schedules, a new way to choose the parents to be coupled, and a two-step strategy by which the second step re-starts the evolution from a neighbor’s population of the best schedule found in the first step. An alternative representation of the chromosomes using a multi-array object-oriented model has been proposed by (Montoye-Torres et al., 2010) which takes advantage of programming features in most common languages for the design of decision support systems.

In this paper a unique modified GA is presented. Contrarily to a traditional GA, the proposed algorithm employs two independent populations, modified two-point crossover operator, enhanced mutation operator and unique parent selection mechanism. Contrary to existing studies, the experiment results are compared with Primavera Project Planner (P6 version 7.0). Especially, until recent years, problem sets with 300 activities have not been tested extensively. Also in this study, the proposed algorithm and Primavera Project Planner are tested and compared by problem sets with 300 activities in addition to problem sets with 30, 60, and 120 activities.

3. Research Methodology

Resource constrained project scheduling problem can be formed conceptually in the following way:

$$\textbf{min } f_n$$

Subject to

$$f_i \leq f_j - d_j \qquad \textbf{for all } (i,j) \in A$$

$$f_1 = 0$$

$$\sum_{i \in S_t} r_{ik} \leq a_k \qquad \textbf{for } k=1,\ldots,m \textbf{ and } t=1,\ldots,f_n$$

In this formulation, dummy start (f_1) and finish activities has been considered. The variables f_i mean the finish times of the tasks, while the d_i express the duration of the tasks, a_k denote the k^{th} resource's availability, r_{ik} the resource demand of the activity *i* for resource *k,* S_t denotes the group of activities that are proceeding at time *t*, *m* is the number of resource types in the project, and lastly, the *j* is the next activity in the chain.

Strategy

In this genetic algorithm, contrary to traditional genetic algorithms, the initial population consists of 2 parts, left population that involves left-justified (forward) schedules and right population that contains right-justified (backward) schedules. Basically, the GA is based on application of fitness calculation process, selection process, crossover operator, mutation operator, and replacement process to initial populations.

The termination criterion in this study is the number of schedules created during the algorithm, such as the first created left-justified and right-justified schedules and the produced schedules after crossover and mutation. Generally, 1000 schedules and in some problems 5000 schedules have been generated in this study.

Throughout the algorithm, at the end of every GA cycle the right population is updated by using left-justified schedules and left population is updated by using right-justified schedules. Therefore, right (left)-justified schedules have been converted to left (right)-justified schedules. By doing so, advantages of repetitive forward and backward scheduling have been used. The pseudo-code is given in the following **Figure 1**.

```
Procedure of Genetic Algorithm

    Step 1    Build an initial right and left population
    Step 2    Revise  number of schedules
    Step 3    Calculate fitness of individuals
    Step 4    Find the best individual

For [i=1,number of schedules]

    Updating right-population

    Step 5    Select 1st Parent from left-population
    Step 6    Find the best partner for 1st Parent
    Step 7    Apply crossover operator
    Step 8    Replace right-population by children
              Revise  number of schedules
    Step 9    Apply mutation operator
              Revise  number of schedules
    Step 10   Find the best individual

    Updating left-population

    Step 5    Select 1st Parent from right-population
    Step 6    Find the best partner for 1st Parent
    Step 7    Apply crossover operator
    Step 8    Replace left-population by children
              Revise  number of schedules
    Step 9    Apply mutation operator
              Revise  number of schedules
    Step 10   Find the best individual

Check if number of schedules has been exceeded
```

Figure 1: Pseudo-code of proposed GA

Building the Initial Population

Firstly, before starting to create first population the number of successors and successors are converted to number of predecessors and predecessors in order to follow the left population creation procedure.

Secondly, left (right) population that is created by using forward (backward) scheduling. The left (right) population is produced randomly where the activities with a 0 number of predecessors (successors) are put to the selection pool and one activity is selected. After this, the numbers of predecessors (successors) of other activities are recalculated and again the activities with a 0 number of predecessors (successors) are put to the selection pool

and one activity is selected. This cycle is repeated until the finish (start) mile stone has been started and finally a feasible starting order of activities is found.

After creating the starting order, the activities' start time and finish time can be calculated. The activities are started by following starting order at first feasible time according to their resource usage and the resource constraints in the project. During assignment of activities start time and finish time, the precedence relations and resource usages should be considered. The activity should start immediately after its precedence is finished if there is an enough resource throughout the activity's duration. On the other hand, if the activity has no predecessor but the resource is inadequate or the resource is available only for a few days, the activity can be started first resource available day.

Parent Selection and Crossover Procedure

After producing 50 left individuals and 50 right individuals, the number of schedules is revised and the next step is calculation of individuals' fitness values to guarantee that the algorithm finds appropriate matches. The fitness values are calculated with following basic formula.

Fitness Value = 1 / individual's makespan

So, the bigger makespan means lower fitness value and that shows the individual (schedule) is not good in terms of project makespan because the objective of the problem is minimizing the project makespan. After calculating the fitness values of individuals, the population is sorted according to their fitness values and the bigger fitness value is positioned at the top of the population as shown in **Figure 2**.

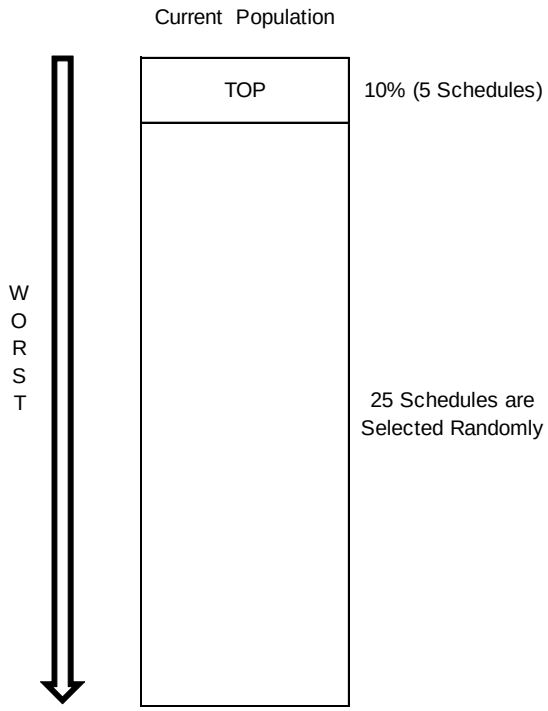

Figure 2: Sorting mechanism of the algorithm

The top part of the population (5 best schedules) and the randomly selected 25 schedules from the non-top part of the population are put in a parent selection pool in **Figure 3**.

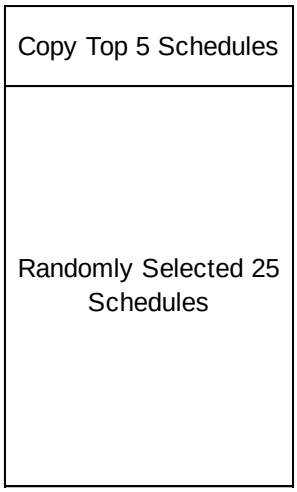

Figure 3: Parent selection pool

The first parent (called father in GA) is selected randomly from this parent selection pool. After, the best match (called father in GA) for that father is found for crossover by calculating the resource utilization ratio (RUR) and total resource utilization (TRU). Resource utilization ratio (RUR), specifies the resource usage at time *t* and calculated as:

$$\mathbf{RUR\ (t,\ S) = (1/K)\ *} \sum_{\mathbf{j\ \epsilon\ active\ (t,\ S)}} \sum_{\mathbf{k=1}}^{\mathbf{K}} \mathbf{r_{jk} / a_k}$$

In the above formulation, the *active (t, S)* expresses the set of activities in schedule *S* at time *t*. *K* represents the number of resources, *k* represents the resource type, the r_{jk} represents the activity *j*'s resource requirement of resource type *k*, and lastly a_k is the availability of resource type *k*.

After calculating RUR, the intervals where the resource usages are high and the intervals where the resource utilization is low will be exposed. In this thesis, t_1 and t_2 are identified as the crossover points where the TRU is maximal between these points. To that aim, the length of the peak, *l is* chosen randomly between (1/4) of makespan and (3/4) of makespan and the total resource utilization of an interval with a start time *t* and length *l* is calculated as:

$$\mathbf{TRU\ (t,\ l,\ S)\ =} \sum_{\mathbf{time\ =\ t}}^{\mathbf{T+l-1}} \mathbf{RUR\ (time,\ S)}$$

The crossover point t_1 is set to *t* where $t \in [0, \text{makespan-}l]$ for which *TRU (t, l, S)* is maximal and the second point t_2 is set to t_1+l. For the rest of the intervals the average RUR will be low.

After defining the crossover points according to father, for the remaining intervals where the RUR is low, the best mother which has a high TRU in intervals [0, t_1] and [t_2, makespan] will be found through the parent selection pool.

Then, two-point crossover operator is applied by using random key (RK) values of activities. The random key values are used to define the priority list based on activity information, such as start time or finish time. For left-justified schedules, the RK takes the value of the finish time of the activity and on the other hand, in right-justified schedules, the RK takes the value of start time of the activity and for the child there are 3 cases; in case 1, if *mother's RK* < t_1, the *child's RK* is *mother's RK*–200, in case 2, $t_1 \leq$ *mother's RK* $\leq t_2$, the *child's RK* is *father's RK*, and in case 3, if *mother's RK* > t_2, the *child's RK* is *mother's RK*+200.To prevent the ambiguous in priority structure the large constant 200 is used. Thus, the usage of mother's best part in terms of resource usage will be guaranteed.

A simple resource-constrained construction of a two-span bridge problem is used to demonstrate the application of proposed algorithm. The network diagram and resource usages, example schedules of father and mother, the RUR and TRU profiles, crossover calculations, and the child schedule are illustrated at **Figure 4, 5, 6, 7**, **Table 1**, and **Figure 8**, respectively.

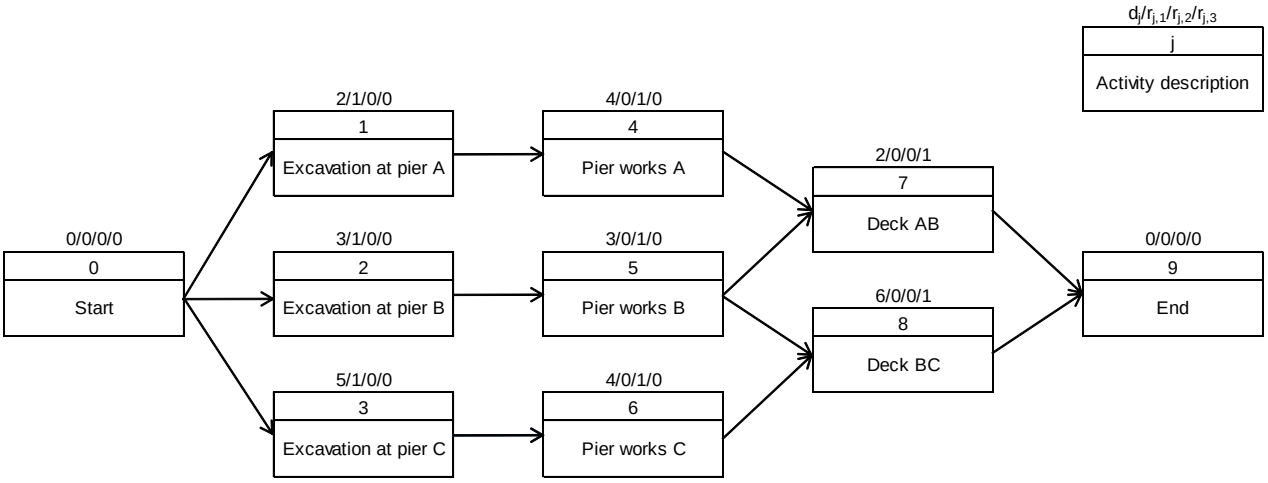

Figure 4 – Network Diagram and Resource Usages

The resource availability for each resource is 1 for this example project. The colored activities express the activities which belong to case 2, and hence RK values of father are same with RK value of child. In this example, the large constant is taken 200.

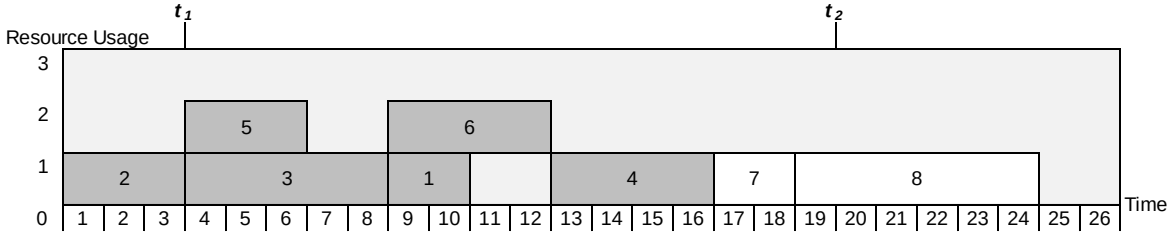

Figure 5 – The schedule of father for crossover

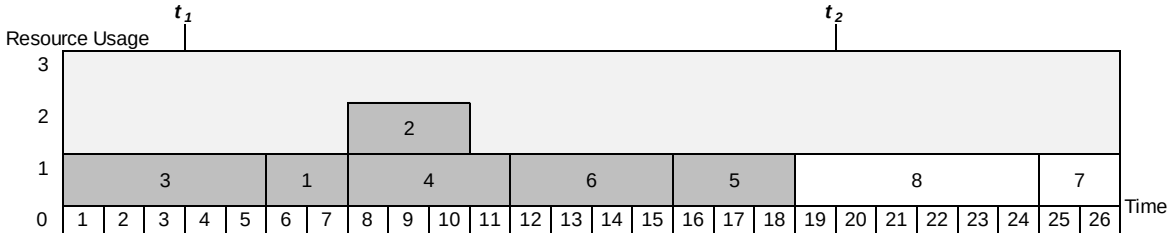

Figure 6 – The schedule of mother for crossover

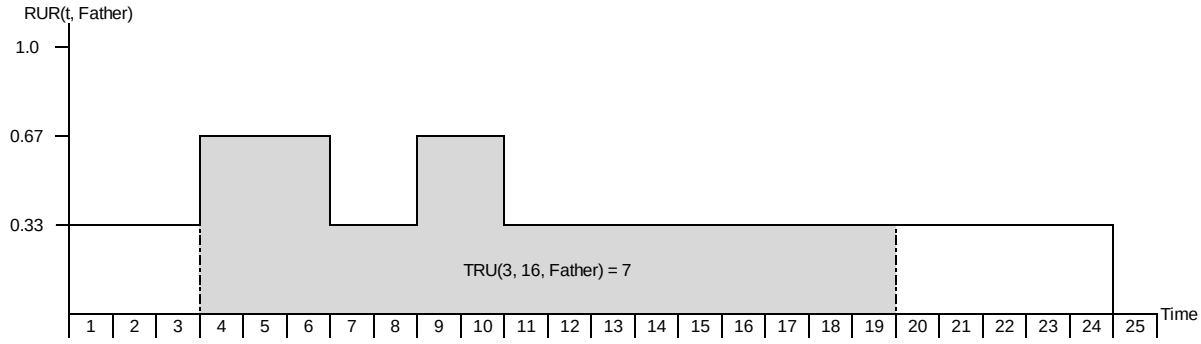

Figure 7 – RUR profile of father

Table 1 – Random key values of the crossover operator

Activity	0	1	2	3	4	5	6	7	8	9
RK of Father	0	10	3	8	16	6	12	18	24	24
RK of Mother	0	7	10	5	11	18	18	26	24	26
RK of Child	-200	10	3	8	16	6	12	226	224	226
Child Start Time	0	12	0	3	14	5	8	18	12	20

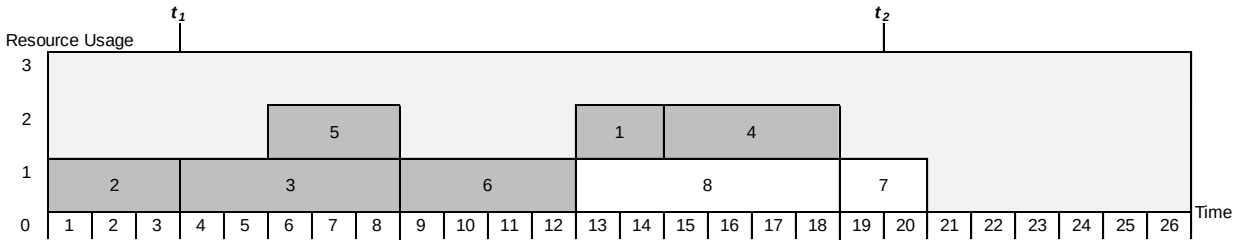

Figure 8 – The schedule of child

As shown in **Table 1**, the random key values are activities' finish time because father and mother schedules are left-justified schedules. In addition, in this example, the length of the peak, l is chosen 16, which should be between (1/4) of makespan which is 6 and (3/4) of makespan which is 18. Moreover, the maximum TRU, which is 7, is found between 3 and 19 as seen on **Figure 7**.

Finally, the child schedule is produced. Father and mother schedules were left-justified schedules, so that the child schedule is right-justified schedule and its random key values are start time of activities. After crossover operator, the crossover applied schedules are replaced with randomly selected individuals from non-top part of the population.

Mutation Procedure

After crossover operator, the mutation procedure takes place, in proposed algorithm the mutation rate is %2 which means, just one individual is chosen from the population. First, one of the schedules is selected randomly from entire population. Second, one of the activities is selected randomly from the schedule except start and finish milestones because they are not allowed to modify. Third, the selected activity's predecessors' and successors' positions are detected to guarantee that there will be no improper precedence relationship between activities in the schedule. After, without changing the relations, the randomly selected activity is shifted with another randomly selected activity. Then, the makespan of the new schedule is found and the mutation is accepted if the new makespan is better that previous makespan of the schedule. The accepted schedule is replaced with the previous schedule. Finally, the schedule is accepted or not the number of schedules is revised. The previous example will be used to illustrate the mutation operator in **Figure 9**.

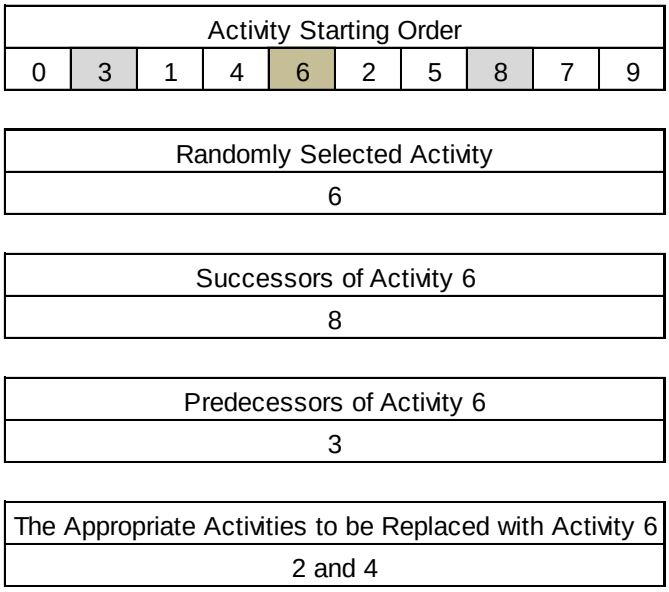

Figure 9 – Application of mutation operator

In the example schedule, the randomly selected activity is 6 and it has a successor of 8 and the predecessor of 3. In terms of activity 6's precedence constraints, in starting order list, the activities between 3 and 8 (1, 4, 2, and 5) are appropriate to replaced with. On the other hand, in terms of activity 1's precedence constraints, activity 4 is the successor of activity 1, if this replacement occurs the activity 1 will be successor of the activity 4 which is not possible. In addition, in terms of activity 5's precedence constraints, activity 2 is the predecessor of activity 5, if this replacement occurs the activity 5 will be predecessor of the activity 2 which is not possible too. So, there are only two activities appropriate to replace, activity 2 and activity 4.

4. Computational Experiments

In this section the results of the computational experiments are presented. Randomly selected 40 problems from the project scheduling problem library (PSPLIB); 10 problems from each problem sets with 30, 60, 120, and 300 activities have been tested by the algorithm. The

selected problem sets have been also tested by Primavera Project Planner (P6 version 7.0) for comparison. The problems' upper bounds (best known solutions) have been known.

The activities are entered to the software with their resource requirements, durations, and the relations with other activities and scheduled according to five different algorithms; Activity ID, Total Float, Late Finish, Early Start, and Free Float. The algorithms are used in both ascending and descending order, therefore in total 10 algorithms are compared with final algorithm in **Table 2** and **Table 3**.

According to Primavera results, the Avg. Dev. U.B. has been calculated for comparison in **Table 4**, **Table 5**. The final algorithm has obviously outperformed the Primavera Project Planner software. The MS Project software uses the same algorithms with Primavera's Total Float (Ascending) and Activity ID algorithms. Hence, the MS Project 2010 software has not been tested.

Table 2 – Comparison of makespans (30 and 60 activities)

Activity Number	Set	CPM L. B.	U.B.	Act. ID (Asc.)	Act. ID (Des.)	TF (Asc.)	TF (Des.)	LF (Asc.)	LF (Des.)	ES (Asc.)	ES (Des.)	FF (Asc.)	FF (Des.)	Proposed Algorithm
30	1	51	71	84	92	83	101	86	95	92	85	89	86	72
	2	70	129	148	158	148	159	147	164	142	162	152	158	129
	3	65	103	107	119	116	126	111	126	111	111	117	118	103
	4	34	58	73	87	70	88	66	83	73	75	71	76	61
	5	45	76	93	102	108	110	87	110	93	104	110	93	81
	6	43	67	95	82	79	106	81	106	81	81	98	94	70
	7	44	64	93	83	73	88	74	87	83	78	80	91	66
	8	44	76	106	88	85	123	92	105	97	99	88	105	78
	9	62	85	109	111	102	110	102	110	107	103	102	108	87
	10	50	80	113	119	95	126	91	124	99	129	101	117	85
60	1	69	112	145	155	145	165	142	170	143	144	142	157	119
	2	65	72	108	86	73	150	82	150	83	105	94	111	72
	3	85	110	155	142	138	176	132	174	148	155	144	152	112
	4	80	144	187	205	183	199	179	195	169	218	165	195	153
	5	74	109	157	137	135	178	137	178	136	183	143	144	118
	6	75	112	139	160	152	168	139	174	143	151	160	136	119
	7	67	115	162	161	160	181	142	171	139	164	156	170	121
	8	78	144	174	197	197	186	175	193	172	186	178	181	156
	9	79	129	174	173	160	191	151	189	160	169	164	178	137
	10	73	123	158	165	151	161	146	170	152	161	152	158	133

Table 3 – Comparison of makespans (120 and 300 activities)

Activity Number	Set	CPM L. B.	U.B.	Act. ID (Asc.)	Act. ID (Des.)	TF (Asc.)	TF (Des.)	LF (Asc.)	LF (Des.)	ES (Asc.)	ES (Des.)	FF (Asc.)	FF (Des.)	Proposed Algorithm
120	1	98	213	286	287	269	332	259	330	263	287	269	304	237
	2	97	234	311	327	297	349	293	342	293	317	308	321	261
	3	99	289	396	387	377	449	356	447	346	416	375	405	324
	4	103	147	209	229	207	266	196	259	200	233	217	218	168
	5	104	215	285	337	285	327	286	352	281	305	306	302	245
	6	91	144	194	194	197	230	181	220	195	194	197	198	161
	7	95	237	311	315	289	340	292	351	294	341	309	310	264
	8	117	280	370	407	370	399	356	405	352	401	347	376	310
	9	120	200	275	277	260	295	230	284	244	274	271	273	217
	10	121	167	228	230	215	265	197	267	213	240	217	237	182
300	1	41	188	216	228	203	220	204	240	208	212	204	219	194
	2	42	831	982	964	909	1065	888	976	902	988	908	1015	861
	3	41	832	1003	1008	943	1069	911	1004	929	1006	960	1017	879
	4	40	1512	1570	1603	1571	1563	1529	1605	1558	1582	1578	1585	1531
	5	61	758	948	901	862	1110	862	1110	856	928	878	1001	803
	6	1142	1412	1457	1496	1468	1510	1455	1503	1458	1464	1475	1513	1438
	7	1069	1162	1203	1247	1200	1260	1195	1257	1201	1220	1206	1260	1167
	8	1111	1576	1577	1576	1576	1579	1577	1577	1577	1577	1576	1579	1576
	9	59	414	437	477	445	457	428	471	425	469	463	457	420
	10	64	1526	1592	1617	1602	1622	1563	1611	1600	1616	1578	1629	1544

Table 4 – Comparison of deviations from U.B. (30 and 60 activities)

Dev. U.B. (%)														
Activity Number	Set	CPM L. B.	U.B.	Act. ID (Asc.)	Act. ID (Des.)	TF (Asc.)	TF (Des.)	LF (Asc.)	LF (Des.)	ES (Asc.)	ES (Des.)	FF (Asc.)	FF (Des.)	Proposed Algorithm
30	1	51	71	18.31	29.58	16.90	42.25	21.13	33.80	29.58	19.72	25.35	21.13	1.41
	2	70	129	14.73	22.48	14.73	23.26	13.95	27.13	10.08	25.58	17.83	22.48	0.00
	3	65	103	3.88	15.53	12.62	22.33	7.77	22.33	7.77	7.77	13.59	14.56	0.00
	4	34	58	25.86	50.00	20.69	51.72	13.79	43.10	25.86	29.31	22.41	31.03	5.17
	5	45	76	22.37	34.21	42.11	44.74	14.47	44.74	22.37	36.84	44.74	22.37	6.58
	6	43	67	41.79	22.39	17.91	58.21	20.90	58.21	20.90	20.90	46.27	40.30	4.48
	7	44	64	45.31	29.69	14.06	37.50	15.63	35.94	29.69	21.88	25.00	42.19	3.13
	8	44	76	39.47	15.79	11.84	61.84	21.05	38.16	27.63	30.26	15.79	38.16	2.63
	9	62	85	28.24	30.59	20.00	29.41	20.00	29.41	25.88	21.18	20.00	27.06	2.35
	10	50	80	41.25	48.75	18.75	57.50	13.75	55.00	23.75	61.25	26.25	46.25	6.25
Avg. Dev. U.B.				28.12	29.90	18.96	42.88	16.24	38.78	22.35	27.47	25.72	30.55	3.20
60	1	69	112	29.46	38.39	29.46	47.32	26.79	51.79	27.68	28.57	26.79	40.18	6.25
	2	65	72	50.00	19.44	1.39	108.33	13.89	108.33	15.28	45.83	30.56	54.17	0.00
	3	85	110	40.91	29.09	25.45	60.00	20.00	58.18	34.55	40.91	30.91	38.18	1.82
	4	80	144	29.86	42.36	27.08	38.19	24.31	35.42	17.36	51.39	14.58	35.42	6.25
	5	74	109	44.04	25.69	23.85	63.30	25.69	63.30	24.77	67.89	31.19	32.11	8.26
	6	75	112	24.11	42.86	35.71	50.00	24.11	55.36	27.68	34.82	42.86	21.43	6.25
	7	67	115	40.87	40.00	39.13	57.39	23.48	48.70	20.87	42.61	35.65	47.83	5.22
	8	78	144	20.83	36.81	36.81	29.17	21.53	34.03	19.44	29.17	23.61	25.69	8.33
	9	79	129	34.88	34.11	24.03	48.06	17.05	46.51	24.03	31.01	27.13	37.98	6.20
	10	73	123	28.46	34.15	22.76	30.89	18.70	38.21	23.58	30.89	23.58	28.46	8.13
Avg. Dev. U.B.				34.34	34.29	26.57	53.27	21.55	53.98	23.52	40.31	28.69	36.14	5.67

Table 5 – Comparison of deviations from U.B. (120 and 300 activities)

Dev. U.B. (%)														
Activity Number	Set	CPM L. B.	U.B.	Act. ID (Asc.)	Act. ID (Des.)	TF (Asc.)	TF (Des.)	LF (Asc.)	LF (Des.)	ES (Asc.)	ES (Des.)	FF (Asc.)	FF (Des.)	Proposed Algorithm
120	1	98	213	34.27	34.74	26.29	55.87	21.60	54.93	23.47	34.74	26.29	42.72	11.27
	2	97	234	32.91	39.74	26.92	49.15	25.21	46.15	25.21	35.47	31.62	37.18	11.54
	3	99	289	37.02	33.91	30.45	55.36	23.18	54.67	19.72	43.94	29.76	40.14	12.11
	4	103	147	42.18	55.78	40.82	80.95	33.33	76.19	36.05	58.50	47.62	48.30	14.29
	5	104	215	32.56	56.74	32.56	52.09	33.02	63.72	30.70	41.86	42.33	40.47	13.95
	6	91	144	34.72	34.72	36.81	59.72	25.69	52.78	35.42	34.72	36.81	37.50	11.81
	7	95	237	31.22	32.91	21.94	43.46	23.21	48.10	24.05	43.88	30.38	30.80	11.39
	8	117	280	32.14	45.36	32.14	42.50	27.14	44.64	25.71	43.21	23.93	34.29	10.71
	9	120	200	37.50	38.50	30.00	47.50	15.00	42.00	22.00	37.00	35.50	36.50	8.50
	10	121	167	36.53	37.72	28.74	58.68	17.96	59.88	27.54	43.71	29.94	41.92	8.98
Avg. Dev. U.B.				35.11	41.01	30.67	54.53	24.54	54.31	26.99	41.71	33.42	38.98	11.46
300	1	41	188	14.89	21.28	7.98	17.02	8.51	27.66	10.64	12.77	8.51	16.49	3.19
	2	42	831	18.17	16.00	9.39	28.16	6.86	17.45	8.54	18.89	9.27	22.14	3.61
	3	41	832	20.55	21.15	13.34	28.49	9.50	20.67	11.66	20.91	15.38	22.24	5.65
	4	40	1512	3.84	6.02	3.90	3.37	1.12	6.15	3.04	4.63	4.37	4.83	1.26
	5	61	758	25.07	18.87	13.72	46.44	13.72	46.44	12.93	22.43	15.83	32.06	5.94
	6	1142	1412	3.19	5.95	3.97	6.94	3.05	6.44	3.26	3.68	4.46	7.15	1.84
	7	1069	1162	3.53	7.31	3.27	8.43	2.84	8.18	3.36	4.99	3.79	8.43	0.43
	8	1111	1576	0.06	0.00	0.00	0.19	0.06	0.06	0.06	0.06	0.00	0.19	0.00
	9	59	414	5.56	15.22	7.49	10.39	3.38	13.77	2.66	13.29	11.84	10.39	1.45
	10	64	1526	4.33	5.96	4.98	6.29	2.42	5.57	4.85	5.90	3.41	6.75	1.18
Avg. Dev. U.B.				9.92	11.78	6.80	15.57	5.15	15.24	6.10	10.75	7.68	13.07	2.45

5. Conclusions

In this study, a genetic algorithm for solving resource-constrained project scheduling problem is proposed. The presented algorithm is capable of solving the RCPSPs with 30, 60, 120, and 300 activities with 4 different resources. The algorithm considers constrained resources and activity relationships at the same time.

In contrast to traditional genetic algorithms, two different initial population are used, the left population that is produced by forward scheduling and the right population that is produced by backward scheduling. In doing so, maintaining the diversity, which is essential part of evolutionary algorithms, has been achieved. In addition, the probability of crossing over similar population has been disappeared and the advantages of both forward and backward scheduling have been harmonized.

Furthermore, modified crossover operator, which is suggested for the first time, has been used. Advantages of iterative forward/backward scheduling have been exploited by feeding the left (right)-justified schedules with right (left)-justified schedules. During this feeding, before crossing over, the population has been sequenced according to their fitness values and best %10 of the population has been directly passed to the next generation. In crossover, the parent has been selected randomly from the parent selection pool and the best partner for the selected parent has been found by using resource utilization ratio and total resource utilization parameter. The partner schedule is selected according to its resource utilization. There are two crossover points, so the project makespan has been divided into three parts. The maximum resource utilization part of the father has been found and this schedule has been matched with the mother with the maximum resource utilization in terms of other two parts of the schedule. Therefore, the father has been paired with the best fit mother in terms of resource utilization.

After crossing over, lastly, the mutation operator has been used to add entirely new individual to the population. By accepting the only individuals which have been improved after mutation, the population is prevented from being worse population. These processes; crossing over, mutation, and updating the population have been applied until the maximum number of schedules has been achieved.

Efficiency of the developed algorithm is validated, by comparing the Primavera results with the results of the final algorithm. 10 problems from each problem sets with 30, 60, 120, and 300 activities have been solved by both Primavera and final algorithm. Notwithstanding the number of the activities, the final algorithm is better than Primavera in the solution of RSPSPs. In addition, when the Primavera's solutions are compared with the final algorithm, the performance of Primavera is observed unsatisfactory to solve resource-constrained project scheduling problems.

The best five Primavera algorithms for solving the RCPSPs with 30 activities are; Late Finish (Ascending), Total Float (Ascending), Early Start (Ascending), Free Float (Ascending), and Early Start (Descending), respectively. The best five Primavera algorithms

for solving the RCPSPs with 60 activities are; Late Finish (Ascending), Early Start (Ascending), Total Float (Ascending), Free Float (Ascending), and Activity ID (Ascending), respectively. The best five Primavera algorithms for solving the RCPSPs with 120 activities are; Late Finish (Ascending), Early Start (Ascending), Total Float (Ascending), Free Float (Ascending), and Activity ID (Ascending), respectively. The best five Primavera algorithms for solving the RCPSPs with 300 activities are; Late Finish (Ascending), Early Start (Ascending), Total Float (Ascending), Free Float (Ascending), and Activity ID (Ascending), respectively.

According to the results, the best algorithm is same for all type of problems with 30, 60, 120, and 300 activities. Late Finish (Ascending) algorithm outperforms all other algorithms in Primavera. In addition, among the other heuristic algorithms in Primavera, the Early Start (Ascending) and Total Float (Ascending) algorithms perform relatively well.

Within the environment of this study, two essential contributions to the existing researches are made. Firstly, a unique crossover operator has been presented and used. With the new crossover operator, especially in large projects, the performance improvement has been observed when compared to conventional GAs. Secondly, the Primavera software has been tested on small and large projects with number of activities of 30, 60, 120, and 300 and compared with the final algorithm. Thus, the performance of developed algorithm has been clearly observed. In addition to these, the best Primavera algorithm for the RCPSP has been observed as Late Finish (Ascending) algorithm.

As computational outcomes show, to find better solutions, more CPU time is required, especially in large-sized projects. Being iterative algorithm, the better solutions can be reached by using more processing units. Therefore, the computational performance can be easily increased by using supercomputers, parallel computing, or graphic based processing units. In addition to the using new technologies, changing code architecture or combining the genetic algorithm with other meta-heuristic methods can be increase the performance of existing study.

5. References

Abeyasinghe M. Chelaka L., Greenwood David J., and Johansen D. Eric, (2001). "An Efficient Method for Scheduling Construction Projects with Resource Constraints", International Journal of Project Management, 19:29-45.

Alcaraz J. and Maroto C., (2001). "A Robust Genetic Algorithm for Resource Allocation in Project Scheduling", Annals of Operations Research, 102, 83–109.

Alcaraz J., Maroto C., and Ruiz R., (2004). "Improving the performance of genetic algorithms for the RCPS problem", Proceedings of the Ninth International Workshop on Project Management and Scheduling, 40–43.

Anagnostopoulos K. and Koulinas G., (2011). "Resource-Constrained Critical Path Scheduling by a GRASP Based Hyperheuristic", Journal of Computing in Civil Engineering, March.

Artigues C., Demassey S., and Néron E., (2008). "Resource-Constrained Project Scheduling: Models, algorithms, extensions and applications", Wiley, Hoboken.

Bent J.A. and Thumann A., (1994). "Project Management for Engineering and Construction", The Fairmont Press, Lilburn.

Bettemir Ö.H., (2009). "Optimization of Time-Cost-Resource Trade-off Problems in Project Scheduling Using Meta-heuristic Algorithms", Middle East Technical University, PhD. Dissertation.

Blazewicz L., Lenstra J., and Kan A.R., (1983). "Scheduling subject to resource constraints: classification and complexity", Discrete Applied Mathematics, 5, 11-24.

Bouleimen K. and Lecocq H., (2003). "A New Efficient Simulated Annealing Algorithm for the Resource-Constrained Project Scheduling Problem and Its Multiple Mode Version", European Journal of Operational Research, 149:268–281.

Chen W., Shi Y-J, Teng H-F, Lan X-P, and Hu L-C, (2010). "An Efficient Hybrid Algorithm for Resource-Constrained Project Scheduling", Information Sciences, 180:1031–1039.

Christodoulou S., (2010). "Scheduling Resource-Constrained Projects with Ant Colony Optimization Artificial Agents", Journal of Computing in Civil Engineering, January-February, 45-55.

Colak S., Agarwal A., and Erenguc S., (2006). "Resource-Constrained Project Scheduling Problem: A Hybrid Neural Approach", Perspectives in Modern Project Scheduling (Jan Weglarz and Joanna Jozefowska, eds), 297-318.

Debels D., Reyck B., Leus R., and Vanhoucke M., (2004). "A Hybrid Scatter Search Electromagnetism Meta-Heuristic for Project Scheduling", European Journal of Operational Research, 169:638–653.

Debels D. and Vanhoucke M., (2005). "A Bi-Population Based Genetic Algorithm for the Resource-Constrained Project Scheduling Problem", Vlerick Leuven Gent Working Paper Series, 6482/08.

Debels D. and Vanhoucke M., (2006). "Meta-Heuristic Resource-Constrained Project Scheduling: Solution space restrictions and neighbourhood extensions", Faculteit Economie en Bedrijfskunde, University Gent, May, 387.

Debels D., Vanhoucke M., (2007). "A Decomposition-Based Genetic Algorithm for the Resource-Constrained Project-Scheduling Problem", Operations Research, Vol. 55, No. 3, May–June, 457–469.

Demeulemeester E.L. and Herroelen W.S., (2002). "Project Scheduling: A Research Handbook", Kluwer Academic Publishers, Boston.

Franco E.G., Zurita F.T., and Delgadillo G.M., (2007). "A Genetic Algorithm for the Resource Constrained Project Scheduling Problem (RSPSP)", Bolivia Research and Development, *Vol. 7*, 41 – 52.

Hartmann S., (1998). "A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling", Naval Research Logistics, 45:733-750.

Hong W., Tongling L., and Dan L., (2010). "Efficient Genetic Algorithm for Resource-Constrained Project Scheduling Problem", Transactions of Tianjin University, 16: 376-382.

Kim K., (2003). "A Resource-constrained CPM (RCPM) Scheduling and Control Technique with Multiple Calendars", Virginia Polytechnic Institute and State University, PhD. Dissertation.

Kim J-L, and Ellis R.D., (2008). "Permutation-Based Elitist Genetic Algorithm for Optimization of Large-Sized Resource-Constrained Project Scheduling", Journal of Construction Engineering and Management, November, 904-913.

Kochetov Yu.A. and Stolyar A.A., (2003). "Evolutionary Local Search with Variable Neighborhood for the Resource Constrained Project Scheduling Problem", Workshop on Computer Science and Information Technologies (CSIT), Ufa, Russia.

Kolisch R. and Hartmann S., (2006). "Experimental Investigation of Heuristics for Resource-Constrained Project Scheduling an Update", European Journal of Operational Research, 174:23–37.

Leu S-S, Chen A-T, and Yang C-H, (1999). "Fuzzy Optimal Model for Resource-Constrained Construction Scheduling", Journal of Computing in Civil Engineering, July, pp. 207-216.

Mendes J.J.M., Goncalves J.F., and Resende M.G.C., (2005). "A Random Key Based Genetic Algorithm For the Resource Constrained Project Scheduling Problem", June, AT&T Labs Research Technical Report.

Mobini M.D.M., Rabbani M., Amalnik M.S., Razmi J., and Rahimi-Vahed A.R., (2009). "Using an Enhanced Scatter Search Algorithm for a Resource-Constrained Project Scheduling Problem", Soft Comput, 13:597–610.

Mobini M., Mobini Z., and Rabbani M., (2010). "An Artificial Immune Algorithm for the Project Scheduling Problem Under Resource Constraints", Applied Soft Computing Journal.

Mutlu M.Ç., (2010). "A Branch and Bound Algorithm for Resource Leveling Problem", Middle East Technical University, M.Sc. Thesis.

Ranjbar M., (2008). "Solving the Resource-Constrained Project Scheduling Problem Using Filter-and-Fan Approach", Applied Mathematics and Computation, 201:313–318.

Seda M., Matousek R., Osmera P., Pivonka P. and Sandera C., (2009). "A Flexible Heuristic Algorithm for Resource-Constrained Project Scheduling", Proceedings of the World Congress on Engineering and Computer Science, October, Vol 2.

Thomas P.R. and Salhi S., (1998). "A Tabu Search Approach for the Resource Constrained Project Scheduling Problem", Journal of Heuristics, 4: 123–139.

Toklu Y. Cengiz, (2002). "Application of Genetic Algorithms to Construction Scheduling with or without Resource Constraints", Canadian Journal of Civil Engineering, 29: 421–429.

Montoya-Torres J.R, Franco E.G., and Mayorga C.P., (2010). "Project Scheduling with Limited Resources Using a Genetic Algorithm", International Journal of Project Management, 28:619–628.

Tseng L-Y and Chen S-C, (2006). "A Hybrid Metaheuristic for the Resource-Constrained Project Scheduling Problem", European Journal of Operational Research 175:707–721.

Valls V., Ballestin F., and Quintanilla S., (2003). "A hybrid genetic algorithm for the RCPSP", Technical report, Department of Statistics and Operations Research, University of Valencia.

Valls V., Ballestin F., and Quintanilla S., (2004). "A population-based approach to the resource-constrained project scheduling problem", Annals of Operations Research, 131:305–324.

Valls V., Ballestin F., and Quintanilla S., (2005). "Justification and RCPSP-A Technique That Pays", European Journal of Operational Research, 165:375–386.

Valls V., Ballestin F., and Quintanilla S., (2008). "A Hybrid Genetic Algorithm for the Resource-Constrained Project Scheduling Problem", European Journal of Operational Research, 185:495–508.

Wall M.B., (1996). "A Genetic Algorithm for Resource-Constrained Scheduling", Massachusetts Institute of Technology, PhD. Dissertation.

Xiaoguang Y., Dechen Z., Lanshun N., and Xiaofei X., (2009). "A Novel Genetic Simulated Annealing Algorithm for the Resource Constrained Project Scheduling Problem", Institute of Electrical and Electronics Engineers, May, 978-1-4244-3894.

Zhang H., Li H., and Tam C.M., (2006). "Particle Swarm Optimization for Resource-Constrained Project Scheduling", International Journal of Project Management, 24:83–92.

Zhang H., Li H., and Tam C.M., (2006). "Permutation-Based Particle Swarm Optimization for Resource-Constrained Project Scheduling", Journal of Computing in Civil Engineering, March-April, 141-149.