

Optimization and Simulation of Resource Constrained Scheduling Problem Using Genetic Algorithm

Jiancheng Wang

Department of Equipment Command, Equipment Academy, Beijing, China

Email address:

wjianch@sohu.com

To cite this article:

Jiancheng Wang. Optimization and Simulation of Resource Constrained Scheduling Problem Using Genetic Algorithm. *Science Journal of Business and Management*. Vol. 4, No. 6, 2016, pp. 229-237. doi: 10.11648/j.sjbm.20160406.18

Received: October 30, 2016; **Accepted:** November 22, 2016; **Published:** January 7, 2017

Abstract: Due to the development of management idea and the scarcity of some resources, the lean management has become the necessary request to implement effective control of resource constrained project. Resource constrained project scheduling is the significant guarantee to attain the lean management. The resource constrained project scheduling problem (RCPSP), with the objective of minimizing project duration and with the precedence relations described by an activity-on-arrow (AOA) network, is formulated as a combination optimization problem and solved using the priority-based genetic algorithm (GA). The activity priorities are represented by chromosome and serial scheduling scheme (SSS) and parallel scheduling scheme (PSS) are developed and utilized to transform chromosome-represented priorities to an active schedule subject to the logic and resource constraints so that project duration corresponding to each chromosome can be evaluated. The overall framework of the GA for the RCPSP is developed and the basic components of the algorithm are designed. Simulation is provided so as to investigate the performance of the priority-based GA with SSS and PSS as decoding method, respectively. The optimal solution to a small-sized resource constrained benchmark instance is scheduled to find the shortest project duration. Comparative simulation results demonstrate not only the effectiveness and efficiency of GA with SSS or PSS as decoding methods in solution to RCPSP with precedence relation of activities diagrammed as an AOA network but also the effect of different evolution parameter settings on solution quality of the problem.

Keywords: Resource Constrained, Project Duration Optimization, Equipment Support, Genetic Algorithm, Network Planning, Activity-on-Arc Network, Serial Scheduling Scheme, Parallel Scheduling Scheme

1. Introduction

Due to the development of management idea and the scarcity of resources, the importance of lean management has been gradually recognized. Lean management requests that limited resources should be considered when project scheduling is made. Resource-constrained scheduling is the effective approach to lean management, and it has been widely applied in various management fields such as construction engineering, equipment support, manufacturing industry, software development, and so on.

On account of the limited resource units that can be assigned to activities, the project completion time will most likely be greater than the duration that critical path method (CPM) has originally calculated, without taking into account the resource usage limit. It is obvious that a project scheduling with inadequate consideration of the resource

usability will result in an unexpected ending. For project managers, activities of a project must be scheduled under resource constrained conditions such that the project will be completed within the scheduled task duration, which is advantageous in organizing and carrying out of the project in line with the predetermined arrangement. How to properly schedule the activities in a project under constraint of limited resources belongs to the resource constrained project schedule problem (RCPSP) [1-14]. The objective of RCPSP is to properly schedule dependent activities over time such that the task duration is minimized while the precedence and resource constraints have to be met.

Since its advent, RCPSP has been widely studied over the past few decades. The suggested algorithms can be classified into two categories: exact solution methods [15-20] and heuristic algorithms.

Mingozi et al. [15] and Brucker et al. [16] developed

branch and bound algorithms, and Gavish and Pirkul [2] used dynamic programming. Blazewicz et al. proved that RCPSP is nondeterministic polynomial (NP)-hard in the strong sense such that the computation time for obtaining the optimal solution using exact algorithms can be extremely long for project even of medium scale. As a result, many procedures have been proposed in the past to overcome the combinatorial explosion problem. The approaches are grouped into priority rule-based methods, classical metaheuristics, non-standard metaheuristics, and other heuristics. Of the classical meta-heuristics, there are genetic algorithms (GAs), particle swarm optimization (PSO), tabu search (TS), simulated annealing (SA), and ant systems (AS), scatter search (SS), filter-and-fan approach [21-38]. For a detailed description of basic components of heuristics such as schedule generation schemes, priority rules, and representations, the reader is referred to [27-30].

This work is focused on a genetic algorithm based meta-heuristic approach with both serial scheduling scheme (SSS) and parallel scheduling scheme (PSS) as schedule generation scheme (SGS) to assist the resource constrained project scheduling. To the best of the author's knowledge, the SGS is well studied for RCPSP with precedence relationship expressed in an activity-on-node (AON) format, but little efforts are given to extend the algorithms to RCPSP with precedence relationship expressed in an activity-on-arc (AOA) format. Simulation results show that none of the schemes is dominant. Generally speaking, for RCPSP with precedence relationship expressed in AOA format, serial scheduling scheme is suitable for relatively small instances while parallel scheduling scheme is suitable for relatively large ones, which is consistent with those for RCPSP with precedence relationship expressed in AON format. Simulation results demonstrate the solution quality and efficiency of RCPSP with moderate size is rather satisfactory. The deficiencies in GA performance such as premature convergence and slow convergence have been identified. The quality and efficiency of solution to RCPSP with moderate or large size may be further improved by bidirectional planning [39] and other strategies [40-52].

The rest of this paper is organized as follows: in Section 2 RCPSP is formulated as a combination optimization problem, in Section 3 the framework and basic components of genetic algorithm are developed and designed. Simulation results of a benchmark instance are given in Section 4. In the end, the work is concluded in Section 5.

2. Formulation

The relationship among activities of a project can be described either as an AOA network or an AON network. The latter one is usually utilized in most research papers, but the former one is especially adopted here for purpose of comparison study. The AOA network is a directed acyclic diagram (DAG) $G(V, A)$ where V is the set of nodes and A is the set of activities between which a finish-start precedence relationship with time lag 0 exists. Each activity can be

distinguished either by number index or by node pair (p, q) . In normal case, the number index expression of an activity in AOA network is adopted for simplicity and ease of description. The node pair expression of an activity is used where necessary. Sets of all immediate predecessor and successor activities of an activity j are denoted by P_j and S_j , respectively.

The Classical resource-constrained project scheduling problem (RCPSP) can be stated as follows: a project consists of N ($|A|=N$) activities in all including the dummy ones. The first activity 1 and the last activity N are, by convention, defined as dummy activities, i.e. activities that require zero execution time. The activities are interrelated by two types of constraints. The first one is a precedence relation between j and $i \in P_j$ which forces activity j not to be started before any one of its immediate predecessor activities, $i \in P_j$, has been finished. This type of constraint can be determined by the corresponding AOA network of the project and the activity durations. The constraint of the second type is related to the resource requirements. The resources can be classified either as renewable or non-renewable. Non-renewable resources are consumed by the activities while renewable resources are not consumed and keep the same amount along the entire lifetime of the project. The renewable resources are considered here. The set of renewable resource types is K . While being processed, activity j requires r_j^k units of resource of type $k \in K$, during each time period of its non-preemptable duration d_j . Resource type k has a limited capacity of R_k , constant through the project execution, which cannot be violated at any time period, i.e., the sum of resource usage of all ongoing activities A_t in time instance t should not exceed R_k units of resource type $k \in K$. The parameters d_j , r_j^k , and R_k are assumed to be nonnegative, deterministic, and integer.

The objective for RCPSP is to find a schedule of the project such that the project duration is minimized subject to precedence and resource constraints. The problem is formulated as an optimization problem as in (1), (2), (3) and (4). The decision variable is the actual start time of activity j , t_j^{AS} .

$$\min T_p = \max \{t_j^{\text{AS}} + d_j \mid j \in A\} \quad (1)$$

s.t.

$$t_j^{\text{AS}} \geq t_i^{\text{AS}} + d_i, j \in A \setminus \{1\}, i \in P_j \quad (2)$$

$$\sum_{j \in A_t} r_j^k(t) \leq \pi R_k^k, t = 1, \dots, T, k \in K \quad (3)$$

$$t_i^{\text{AS}} \geq 0, j \in A \quad (4)$$

In which, A_t is the set of ongoing activities in time instant t , as defined in (5).

$$A_t = \{j \mid j \in A, d_j \neq 0, t_j^{\text{AS}} \leq t \leq t_j^{\text{AS}} + d_j - 1\} \quad (5)$$

An upper bound on the project's makespan is denoted as

$T = \sum_{j=1}^N d_j$, and πR_i^k is the as-yet-unused resource units of type k at time instant t .

The objective function (1) minimizes the makespan of the project. Constraints (2) take into consideration the precedence relations between activities i and j , where i immediately precedes j . Finally, constraint set (3) limits the total resource usage within each time period to the maximum available amount.

3. Solution Using Genetic Algorithm

GA is essentially an iteration procedure that operates on chromosomes, which resulting in solution to the decision variables.

3.1. Overall Framework

The overall framework of genetic algorithm is developed as in Algorithm 1. It is actually constructed as an iteration process. Initialization is done first, and then enters the outermost loop. Within the loop, the population is evaluated, and the best individual of current generation and the best individual so far are updated. Generation gen is increased by 1. Terminating condition is now checked. If the terminating condition is not satisfied, selection, crossover and mutation operators as well as the elitist strategy are executed to generate the next population, which is the beginning of the next iteration; otherwise, exit outermost loop and necessary quantities as well as the elapsed time are output.

Algorithm 1. Framework of Genetic Algorithm

```

/* initialization */
set  $aPopulation.chrom = rand(popSize, N)$ 
/* main loop */
set  $gen = 1$ 
repeat
  if  $gen > 1$  then
    /* generate next population */
    roulette wheel selection
    parametrized uniform crossover
    mutation
    /* perform evolution based on elitist strategy */
    /* replace chromosome of a individual by that of current
       best one */
     $ip = unidrnd(popSize)$ 
     $aPopulation.Chrom(ip, :) = currentBestIndividual.chrom$ 
  end if
  /* evaluation loop */
  for  $i = 1$  to  $popSize$  do
    decode  $aPopulation.Chrom(i, :)$  to get  $S_n$  and actual
      start time
    set  $aPopulation.serial = S_n$ 
    compute project duration  $T_p$  from actual start times of
      all activities
    set  $aPopulation.objValue(i) = T_p$ 
  end for
  /* find the best individual */
  set  $bestIndividual.chrom = aPopulation.chrom(1, :)$ 

```

```

set  $bestIndividual.serial = aPopulation.serial(1, :)$ 
set  $bestIndividual.objValue = aPopulation.objValue(1)$ 
set  $bestIndex = 1$ 
for  $i = 1$  to  $popSize$  do
  if  $aPopulation.objValue(i) < bestIndividual.objValue$  then
    set  $bestIndividual.chrom = aPopulation.chrom(i, :)$ 
    set  $bestIndividual.serial = aPopulation.serial(i, :)$ 
    set  $bestIndividual.objValue = aPopulation.objValue(i)$ 
    set  $bestIndex = i$ 
  end if
end for
/* find out the best individual so far */
if  $gen = 1$  then
   $currentBestIndividual.chrom = bestIndividual.chrom$ 
   $currentBestIndividual.serial = bestIndividual.serial$ 
   $currentBestIndividual.objValue = bestIndividual.objValue$ 
else
  if  $bestIndividual.objValue < currentBestIndividual.
    objValue$  then
     $currentBestIndividual.chrom = bestIndividual.chrom$ 
     $currentBestIndividual.serial = bestIndividual.serial$ 
     $currentBestIndividual.objValue = bestIndividual.
      objValue$ 
  end if
end if
set  $gen = gen + 1$ 
until  $currentBestIndividual.objValue$  is good enough or
 $gen > maxGen$ 
/* output results */
output  $currentBestIndividual$ 
output the scheduled actual start times of activities

```

3.2. Initialization

Initialized quantities includes P_j , d_j , and r_j^1 , $j \in A$, R_1 , crossover probability P_c , mutation probability P_m , crossover probability threshold T_{CP} , percentage of mutants in a population $Perc_m$, maximum evolution generation $maxGen$, population size $popSize$, chromosome length N , the number of AOA network nodes $numNode$.

Define variable $aPopulation$ to hold the whole population, which is a structure of chromosome $chrom$, precedence feasible serial $serial$, and objective value $objValue$. Define variable $bestIndividual$ to track the best individual of the whole population in a generation, and variable $bestIndex$ to track the index to the best individual in the generation. Define variable $currentBestIndividual$ to track and record the best individual so far. The initial chromosomes are randomly generated such that $aPopulation.chrom = rand(popSize, N)$.

3.3. Chromosome Expression

In general, there are two forms of chromosome representations: permutation-based representation and priority-based representation [53]. The latter is adopted here for its remark-able merit that the activity sequence decoded from chromo-some after crossover and mutation is still precedence feasible, and no repairing operator is needed to

resolve the precedence conflict, which would appear if the former expression of a chromosome is used. For priority-based representation, a chromosome is made of N genes:

$$\text{chromosome} = (\underbrace{\text{gene}_1, \dots, \text{gene}_N}_{\text{priorities}})$$

The j th gene of a chromosome stands for the priority of activity j , which will be used by the chromosome decoding. For activity j , it would compete with its priority among its competitors in current decision set to be scheduled first.

3.4. Schedule Generation Scheme

Generally, a priority rule based scheduling heuristic is made up of two main components, a schedule generation scheme and a priority rule [54]. Two different schemes can be distinguished: the so-called serial and the parallel method. Both generate a feasible schedule by extending a partial schedule (i.e. a schedule where only a subset of the activities has been assigned the start times) in a stage-wise fashion. In each stage the generation scheme forms the set of all schedulable activities, the so-called decision set. A specific priority rule is then employed in order to choose and schedule one or more activities from the decision set. The formal algorithm for serial scheduling scheme (SSS) and parallel scheduling scheme (PSS) are developed, as shown below, to transform the chromosome-represented priorities to an active schedule under the precedence constraints and limited resource constraints so that the chromosome can be evaluated during searching for an optimal solution.

Procedure 1. Serial Scheduling Scheme

```

begin
   $n \leftarrow 1$ 
   $S_n \leftarrow \varnothing$ 
   $t_j^{\text{AS}} \leftarrow T, j \in A$ 
   $\pi R_t^k \leftarrow R_k, t = 0, \dots, T-1, k \in K$ 
   $v = a\text{Population.chrom}$ 
  while  $|S_n| \leq N-1$  do
     $D_n \leftarrow \{j \mid j \notin S_n, P_j \subseteq S_n\}$ 
     $j^* \leftarrow \min_{j \in D_n} \{j \mid v(j) = \max_{i \in D_n} \{v(i)\}\}$ 
    if  $P_{j^*} = \varnothing$  then
       $t_{j^*}^{\text{ES}} \leftarrow 0$ 
    else
       $t_{j^*}^{\text{ES}} \leftarrow \max \{t_i^{\text{AS}} + d_i \mid i \in P_{j^*}\}$ 
    end if
    if  $d_{j^*} = 0$  then
       $t_{j^*}^{\text{AS}} \leftarrow t_{j^*}^{\text{ES}}$ 
    else

```

$$t_{j^*}^{\text{AS}} \leftarrow \min \{t \mid t \in \{t_{j^*}^{\text{ES}}, \dots, T-1\}, \pi R_t^k \geq r_{j^*}^k, \\ \tau = t, t+1, \dots, t+d_{j^*}-2, t+d_{j^*}-1, k \in K\}$$

end if

$$S_{n+1} \leftarrow S_n \cup \{j^*\}$$

for $t=0$ **to** $T-1$ **do**

$$A_t \leftarrow \{j \mid j \in A, d_j \neq 0, t_j^{\text{AS}} \leq t \leq t_j^{\text{AS}} + d_j - 1\}$$

if $A_t = \varnothing$ **then**

$$\pi R_t^k \leftarrow R_k, k \in K$$

else

$$\pi R_t^k \leftarrow R_k - \sum_{j \in A_t} r_j^k, k \in K$$

end if

end for

$$n = n+1$$

end while

end

Procedure 2. Parallel Scheduling Scheme

begin

$$n \leftarrow 1$$

$$t_n \leftarrow 0$$

$$D_n \leftarrow \{1\}$$

$$A_n \leftarrow \varnothing$$

$$C_n \leftarrow \varnothing$$

$$v = a\text{Population.chrom}$$

while $|A_n \cup C_n| \leq N-1$ **do**

if $n > 1$ **then**

$$t_n \leftarrow \min \{t_j^{\text{AS}} + d_j \mid j \in A_{n-1}\}$$

$$A_n \leftarrow A_{n-1} \setminus \{j \mid j \in A_{n-1}, t_j^{\text{AS}} + d_j = t_n\}$$

$$C_n \leftarrow C_{n-1} \cup \{j \mid j \in A_{n-1}, t_j^{\text{AS}} + d_j = t_n\}$$

if $A_n = \varnothing$ **then**

$$\pi R_{t_n}^k \leftarrow R_k, k \in K$$

else

$$\pi R_{t_n}^k \leftarrow R_k - \sum_{j \in A_n} r_j^k, k \in K$$

end if

$$D_n \leftarrow \{j \mid j \notin \{C_n \cup A_n\}, P_j \subseteq C_n,$$

$$r_j^k \leq \pi R_{t_n}^k, k \in K\}$$

end if

while $D_n \neq \varnothing$ **do**

$$j^* \leftarrow \min_{j \in D_n} \{j \mid v(j) = \max_{i \in D_n} \{v(i)\}\}$$

$$t_{j^*}^{\text{AS}} \leftarrow t_n$$

$$A_n \leftarrow A_n \cup j^*$$

$$\pi R_{t_n}^k \leftarrow R_k - \sum_{j \in A_n} r_j^k, k \in K$$

$$D_n \leftarrow \{j \mid j \notin \{C_n \cup A_n\}, P_j \subseteq C_n,$$

$$r_j^k \leq \pi R_{t_n}^k, k \in K\}$$

end while

$$n = n+1$$

end while

$S_n \leftarrow C_n \cup A_n$
end

3.5. Evaluation

During each generation, the individuals in the current population are evaluated. To evaluate an individual, the chromosome is decoded first. By decoding, the actual start time of activity j , t_j^{AS} , is obtained, and the objective value can be further computed based on the actual start times of all activities. Closely after evaluation, the best individual of the generation, and the best individual so far are picked out.

3.6. Fitness Function

Since RCPSp is formulated as a minimization problem, and the roulette wheel selection is a fitness-proportional selection, a transformation is utilized to map the natural objective value into a fitness value to ensure the individual of lower objective value has bigger selection probability.

The transformation of objective value T_p corresponding to chromosome c into a fitness value is implemented by an absolute fitness function as shown in (6).

$$f(c) = T - T_p(c) \quad (6)$$

3.7. Next Population Generation

Based on the fitness value of chromosomes in the current generation, the roulette wheel selection mechanism is used to probabilistically copy N chromosomes into the next generation. After pair making, parametrized uniform crossover [55] is utilized to inherit the genes of same loci into the chromosome of children either from father or from mother depending on the comparison between a real random number and the crossover probability threshold T_{CP} . For each locus, a real random number in the interval $[0, 1]$ is generated. If the random number obtained is smaller than T_{CP} , then the allele of father is inherited by the offspring. Otherwise, the inherited allele is that of the mother. Mutation operator should prevent premature convergence of the population to local minima. To this end, instead of performing gene-by-gene mutation at each generation with very small probability P_m , a few randomly selected individuals, say $Perc_m$ percent of $popSize$, are replaced with randomly generated new ones. These new individuals (called mutants) are randomly generated with the same distribution as the original population. Therefore, no genetic material of the current population is brought into the mutants, and the mutation operator can therefore prevent premature convergence of the population and lead to a simple statement of convergence, i.e., if a sufficiently large number of generations are carried out, then the entire solution space will be sampled. After mutation, the elitist strategy [56] is performed to replace the chromosome of an arbitrary individual in current generation by that of the current best one so far. The main advantage of this strategy is that the best solution is monotonically improving from

one generation to the next. At this point, the process of next generation is completed.

3.8. Output

Outputted are the best chromosome $chrom_{best}$, the best activity serial S_n corresponding to $chrom_{best}$, the actual start time $t_{S_n}^{AS}$ corresponding to S_n and the shortest project duration T_p . The consumed CPU time of a running of the algorithm is also stored.

4. Simulation Instance

In this section, numerical simulations is implemented in Matlab Language to evaluate the performance of the GA proposed in Section 3 and to compare the efficiency of serial scheduling scheme and parallel scheduling scheme. The results of revised version a small-sized benchmark instance [29] are reported, in which the precedence relation of activities in the project is diagramed as an activity on arrow network.

The AOA network for the instance with 12 nodes and 15 activities including 4 dummy ones is shown in Figure 1, and the parameters P_j , d_j , and r_j^1 , are given in Table 1. $R_1=5$. The evolutionary environment of the problem is that $popSize=50$, $N=15$, $P_c=0.80$, $Perc_m=5$, $T_{CP}=0.7$, $maxGen=20$. The GA program is randomly run 10 times with PSS as decoding method, and all solutions have attained the exact one of the problem, which shows the capability of the GA with PSS as decoding method and with the given parameter settings. The scheduled results of the precedence feasible activity sequence S_n and the actual start time $t_{S_n}^{AS}$ for each run are listed in Table 2 (index to GA running is indicated by IX). The resource allocation profile with PSS as decoding method is shown in Figure 2. The evolution process of the problem is shown in Figure 3. The optimum project duration 20 is obtained at the early stage of the evolution process, which is the same as that of the AON network [29]. It demonstrates that the proposed algorithm can approach the known exact solution rapidly.

For comparison, with the same parameter settings as used for the first 10 runs, the program is randomly run another 10 times for the solution of the instance problem with SSS as decoding method, and 9 out of 10 solutions (but not given here) have attained the exact one of the problem, which also shows the capability of the GA with SSS as decoding method and with the given parameter settings.

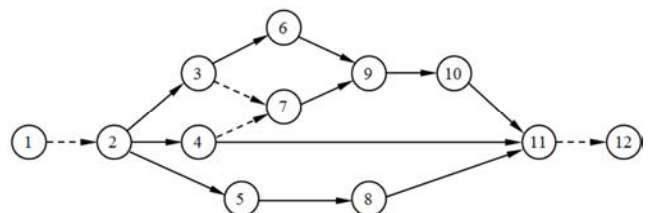


Figure 1. AOA network for the instance example with 12 nodes and 15 activities including 4 dummy ones.

It can be drawn from the above two groups of solutions that the schedule generation scheme, PSS or SSS, has little effect on the running time rate of obtaining the same optimal solutions as the exact one to the problem. But if the size of the resource constrained project schedule problem is decreased or increased, the effect of schedule generation scheme on the rate may be different. In addition to the adoption of schedule generation scheme, the evolution parameter settings will also affect the solution quality of the genetic algorithm. For example, the GA program is randomly run the third 10 times with all the conditions being kept the same as the second 10 runs except for the percentage of mutants in a population, $Perc_m$, is increased from 5 to 15. Under this circumstance, all solutions have attained the exact one of the problem, as contrasted to the 9 out of 10 in the second 10 runs.

Table 1. Parameters of all activities.

j	(p, q)	P_j	d_j	r_j^1
1	(1, 2)	-	0	0
2	(2, 3)	1	4	2
3	(2, 4)	1	2	3
4	(2, 5)	1	4	2
5	(3, 6)	2	3	5
6	(3, 7)	2	0	0
7	(4, 7)	3	0	0
8	(4, 11)	3	3	3
9	(5, 8)	4	4	3
10	(6, 9)	5	4	2
11	(7, 9)	6, 7	3	2
12	(8, 11)	9	2	2
13	(9, 10)	10, 11	2	5
14	(10, 11)	13	3	2
15	(11, 12)	8, 12, 14	0	0

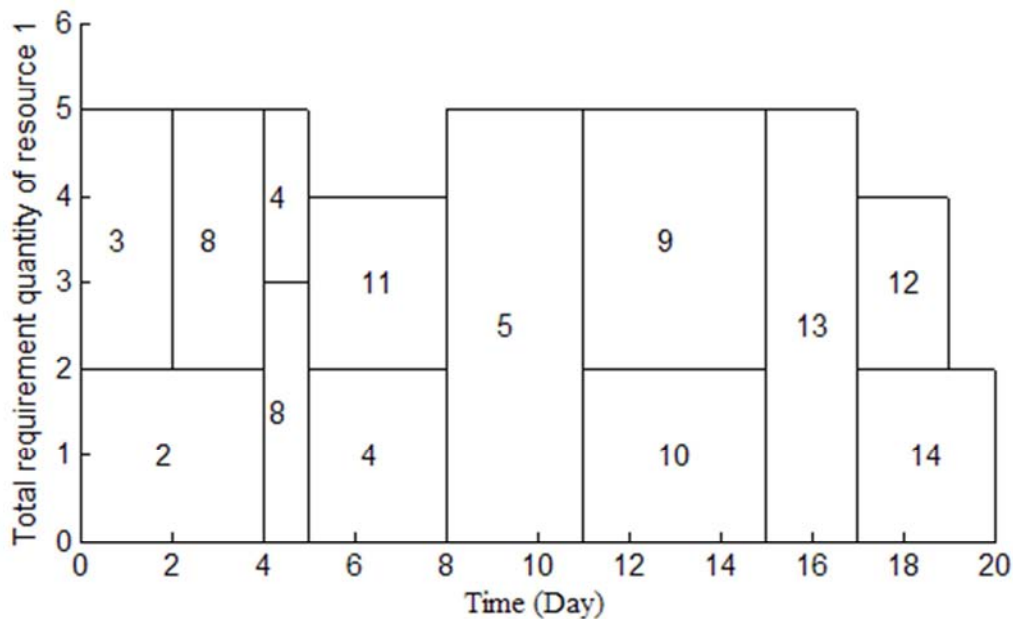


Figure 2. Allocation profile of resource 1.

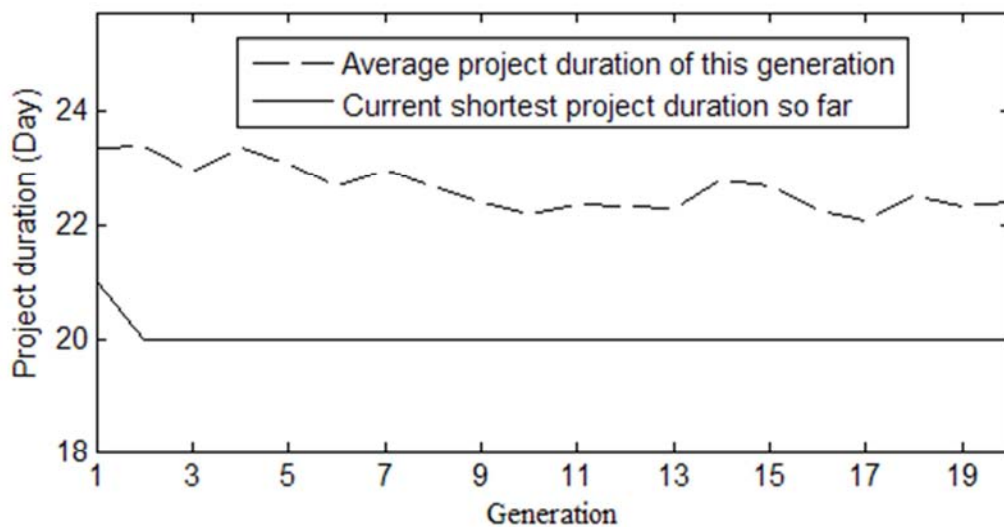


Figure 3. Evolution process of the genetic algorithm.

Table 2. 10 Results of Precedence Feasible Activity Sequence and Actual Start Time Scheduled with PSS.

IX	Index to activity positions in precedence and resource feasible activity serial S_n															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	S_n	1	2	3	8	7	6	4	11	5	10	9	13	12	14	15
	$t_{S_n}^{AS}$	0	0	0	2	2	4	4	5	8	11	11	15	17	17	20
2	S_n	1	2	3	7	8	6	4	11	5	9	10	13	14	12	15
	$t_{S_n}^{AS}$	0	0	0	2	2	4	4	5	8	11	11	15	17	17	20
3	S_n	1	2	3	7	8	6	4	11	5	10	9	13	14	12	15
	$t_{S_n}^{AS}$	0	0	0	2	2	4	4	5	8	11	11	15	17	17	20
4	S_n	1	3	2	8	7	6	4	11	5	10	9	13	12	14	15
	$t_{S_n}^{AS}$	0	0	0	2	2	4	4	5	8	11	11	15	17	17	20
5	S_n	1	2	3	8	7	6	4	11	5	10	9	13	14	12	15
	$t_{S_n}^{AS}$	0	0	0	2	2	4	4	5	8	11	11	15	17	17	20
6	S_n	1	3	2	7	8	6	4	11	5	9	10	13	14	12	15
	$t_{S_n}^{AS}$	0	0	0	2	2	4	4	5	8	11	11	15	17	17	20
7	S_n	1	2	3	7	8	6	4	11	5	10	9	13	14	12	15
	$t_{S_n}^{AS}$	0	0	0	2	2	4	4	5	8	11	11	15	17	17	20
8	S_n	1	3	2	8	7	6	4	11	5	9	10	13	14	12	15
	$t_{S_n}^{AS}$	0	0	0	2	2	4	4	5	8	11	11	15	17	17	20
9	S_n	1	3	2	8	7	4	6	11	5	10	9	13	14	12	15
	$t_{S_n}^{AS}$	0	0	0	2	2	4	4	5	8	11	11	15	17	17	20
10	S_n	1	2	3	7	8	6	4	11	5	10	9	13	14	12	15
	$t_{S_n}^{AS}$	0	0	0	2	2	4	4	5	8	11	11	15	17	17	20

5. Conclusions

Priority-based representation is used to express chromosome in GA, which is decoded by serial scheduling scheme and parallel scheduling scheme. A resource constrained instance project is scheduled using genetic algorithm to find the shortest project duration. Simulation results show the capability of the GA with PSS or SSS as decoding method, as well as the effect of different evolution parameter settings on solution quality of the small-sized instance problem. Future efforts will be focused on the research regarding how the decoding method and evolution parameter settings affect the convergence and the efficiency of the proposed algorithm.

References

- [1] E. W. DAVIS and J. H. Patterson, "A comparison of heuristic and optimum solutions in resource-constrained project scheduling," *Management Science*, vol. 21, pp. 944–955, April 1975.
- [2] B. Gavish and H. Pirkul, "Algorithms for multi-resource generalized assignment problem," *Management Science*, vol. 37, pp. 695–713, June 1991.
- [3] R. Klein, "Project scheduling with time-varying resource constraints," *International Journal of Production Research*, vol. 38, pp. 3937–3952, November 2000.
- [4] J. Buddhakulsomsiri and D. S. Kim, "Properties of multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting," *Eur. J. Oper. Res.*, vol. 175, pp. 279–295, November 2006.
- [5] R. Kolisch and S. Hartmann, "Experimental investigation of heuristics for resource-constrained project scheduling: an update," *Eur. J. Oper. Res.*, vol. 174, pp. 23–37, October 2006.
- [6] J. Buddhakulsomsiri and D. S. Kim, "Priority rule-based heuristic for multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting," *Eur. J. Oper. Res.*, vol. 178, pp. 374–390, April 2007.
- [7] P.-H. Chen and H. Weng, "A two-phase GA model for resource-constrained project scheduling," *Autom. Constr.*, vol. 18, pp. 485–498, July 2009.
- [8] D. Heon Jun and K. El-Rayes, "Multiobjective optimization of resource leveling and allocation during construction scheduling," *J. Constr. Eng. Manag.*, vol. 137, pp. 1080–1088, December 2011.
- [9] P. Ghoddousi, E. Eshtehardian, S. Jooybanpour, and A. Javanmardi, "Multi-mode resource-constrained discrete time-cost-resource optimization in project scheduling using non-dominated sorting genetic algorithm," *Autom. Constr.*, vol. 30, pp. 216–227, March 2013.
- [10] V. V. Peteghem and M. Vanhoucke, "An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances," *Eur. J. Oper. Res.*, vol. 235, pp. 62–72, May 2014.
- [11] C. Kellenbrink and S. Helber, "Scheduling resource-constrained projects with a flexible project structure," *Eur. J. Oper. Res.*, vol. 246, pp. 379–391, October 2015.

- [12] J. Cheng, J. Fowler, K. Kempf, and S. Mason, "Multi-mode resource-constrained project scheduling problems with non-preemptive activity splitting," *Computers & Oper. Research*, vol. 53, pp. 275–287, January 2015.
- [13] M. Vanhoucke and J. Coelho, "An approach using SAT solvers for the RCPSP with logical constraints," *Eur. J. Oper. Res.*, vol. 249, pp. 577–591, March 2016.
- [14] S. Kreter, J. Rieck, and J. Zimmermann, "Models and solution procedures for the resource-constrained project scheduling problem with general temporal constraints and calendars," *Eur. J. Oper. Res.*, vol. 251, pp. 387–403, June 2016.
- [15] A. Mingozzi, V. Maniezzo, S. Ricciardelli, and L. Bianco, "An exact algorithm for the resource constrained project scheduling problem based on a new mathematical formulation," *Management Science*, vol. 44, pp. 714–729, May 1998.
- [16] P. Brucker, S. Knust, A. Schoo, and O. Thiele, "A branch-and-bound algorithm for the resource constrained project scheduling problem," *Eur. J. Oper. Res.*, vol. 107, pp. 272–288, June 1998.
- [17] R. Klein and A. Scholl, "Scattered branch and bound: an adaptive search strategy applied to resource-constrained project scheduling," *Cent. Eur. J. Oper. Res.*, vol. 7, pp. 177–201, January 1998.
- [18] A. Sprecher, "Scheduling resource-constrained projects competitively at modest memory requirements," *Manag. Sci.*, vol. 46, pp. 710–723, May 2000.
- [19] A. Naber and R. Kolisch, "MIP models for resource-constrained project scheduling with flexible resource profiles," *Eur. J. Oper. Res.*, vol. 239, pp. 335–348, December 2014.
- [20] L. James, "A branch-and-cut decomposition algorithm for solving chance-constrained mathematical programs with finite support," *Mathematical Programming*, vol. 146, pp. 219–244, August 2014.
- [21] S.-E. Sampson and E.-N. Weiss, "Local search techniques for the generalized resource-constrained project scheduling problem," *Naval Research Logistics*, vol. 40, pp. 665–675, August 1993.
- [22] E. Pinson, C. Prins, and F. Rullier, "Using tabu search for solving the resource-constrained project scheduling problem," in *Proceedings of the Fourth International Workshop on Project Management and Scheduling*, E. Demeulemeester and W. Herroelen, eds. Leuven: Research Center for Operations Management, 1994, pp. 102–106.
- [23] J.-K. Lee and Y.-D. Kim, "Search heuristics for resource constrained project scheduling," *Journal of the Operational Research Society*, vol. 47, pp. 678–689, May 1996.
- [24] S. Hartmann, "A competitive genetic algorithm for resource-constrained project scheduling," *Naval Research Logistics*, vol. 45, pp. 733–750, October 1998.
- [25] T. Baar, P. Brucker, and S. Knust, "Tabu-search algorithms and lower bounds for the resource-constrained project scheduling problem," in *Meta-heuristics: advances and trends in local search paradigms for optimization*, S. Voss, S. Martello, I. Osman, and C. Roucairol, Eds. Dordrecht: Kluwer Academic Publishers, 1998, pp. 1–8.
- [26] K. Bouleimen and H. Lecocq, "A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple modes version," *Eur. J. Oper. Res.*, vol. 149, pp. 268–281, September 2003.
- [27] R. Kolisch and S. Hartmann, "Heuristic algorithms for solving the resource-constrained project scheduling problem: classification and computational analysis," in *Project Scheduling: Recent Models, Algorithms and Applications*, J. Weglarz, Ed. Berlin: Kluwer Academic Publishers, 1999, pp. 147–178.
- [28] S. Hartmann and R. Kolisch, "Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem," *Eur. J. Oper. Res.*, vol. 127, pp. 394–407, December 2000.
- [29] H. Wang, T.-L. Li, and D. Lin, "Efficient genetic algorithm for resource-constrained project scheduling problem," *Trans. of Tianjin University*, vol. 16, pp. 376–382, October 2010.
- [30] F. Gargiulo and D. Quagliarella, "Genetic algorithms for the resource constrained project scheduling problem," in *Proc. the 13th IEEE International Symp. Computational Intelligence and Informatics (CINTI 2012)*. Budapest: IEEE Press, 2012, pp. 39–47.
- [31] A. Lim, H. Ma, and B. Rodrigues, "New meta-heuristics for the resource-constrained project scheduling problem," *Flexible Services and Manufacturing Journal*, vol. 25, pp. 48–73, June 2013.
- [32] M. Mori and C. C. Tseng, "A genetic algorithm for multi-mode resource constrained project scheduling problem," *Eur. J. Oper. Res.*, vol. 100, pp. 134–141, July 1997.
- [33] V. Valls, F. Ballestín, and S. Quintanilla, "A hybrid genetic algorithm for the resource-constrained project scheduling problem," *Eur. J. Oper. Res.*, vol. 185, pp. 495–508, March 2008.
- [34] R. Zamani, "A competitive magnet-based genetic algorithm for solving the resource-constrained project scheduling problem," *Eur. J. Oper. Res.*, vol. 229, pp. 552–559, September 2013.
- [35] H. Zhang, X. Li, H. Li, and F. Huang, "Particle swarm optimization-based schemes for resource-constrained project scheduling," *Autom. Constr.*, vol. 14, pp. 393–404, June 2005.
- [36] H. Zhang, H. Li, and C. Tam, "Particle swarm optimization for resource-constrained project scheduling," *Int. J. Proj. Manag.*, vol. 24, pp. 83–92, January 2006.
- [37] D. Merkle, M. Middendorf, and H. Schmeck, "Ant colony optimization for resource-constrained project scheduling," *IEEE Trans. Evol. Comput.*, vol. 6, pp. 333–346, August 2002.
- [38] J.-G. He, X.-D. Chen, and X. Chen, "A filter-and-fan approach with adaptive neighborhood switching for resource-constrained project scheduling," *Computers & Operations Research*, vol. 71, pp. 71–81, July 2016.
- [39] R. Klein, "Bidirectional planning: improving priority rule-based heuristics for scheduling resource-constrained projects," *Eur. J. Oper. Res.*, vol. 127, pp. 619–638, December 2000.
- [40] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Comput. Surv.*, vol. 31, pp. 264–323, September 1999.
- [41] A. El Imrani, A. Bouroumi, H. Zine El Abidine, M. Limouri, and A. Essaïd, "A fuzzy clustering-based niching approach to multimodal function optimization," *Cogn. Syst. Res.*, vol. 1, pp. 119–133, June 2000.

- [42] J. Alami, A. E. Imrani, and A. Bouroumi, "A multipopulation cultural algorithm using fuzzy clustering," *Appl. Soft Comput.*, vol. 7, pp. 506–519, March 2007.
- [43] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: the fuzzy c-means clustering algorithm," *Comput. Geosci.*, vol. 10, pp. 191–203, December 1984.
- [44] W. Kwedlo, "A clustering method combining differential evolution with the K-means algorithm," *Pattern Recogn. Lett.*, vol. 32, 1613–1621, September 2011.
- [45] Y.-J. Wang, J.-S. Zhang, and G.-Y. Zhang, "A dynamic clustering based differential evolution algorithm for global optimization," *Eur. J. Oper. Res.*, vol. 183, pp. 56–73, November 2007.
- [46] Z. Cai, W. Gong, C.-X. Ling, and H. Zhang, "A clustering-based differential evolution for global optimization," *Appl. Soft Comput.*, vol. 11, pp. 1363–1379, January 2011.
- [47] M.-Y. Cheng and K.-Y. Huang, "Genetic algorithm-based chaos clustering approach for nonlinear optimization," *J. Mar. Sci. Technol.*, vol. 18, pp. 435–441, June 2010.
- [48] R. Caponetto, L. Fortuna, S. Fazzino, and M. G. Xibilia, "Chaotic sequences to improve the performance of evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, pp. 289–304, June 2003.
- [49] B. Li and W.-S. Jiang, "Optimizing complex functions by chaos search," *Cybern. Syst.* vol. 29, pp. 409–419, January 1998.
- [50] A. Thammano and A. Phu-ang, "A hybrid evolutionary algorithm for the resource-constrained project scheduling problem," *Artificial Life and Robotics*, vol. 17, pp. 312–316, December 2012.
- [51] D. Debels and M. Vanhoucke, "A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem," *Operations Research*, vol. 55, pp. 457–469, May-June 2007.
- [52] M.-Y. Cheng, D.-H. Tran, and Y.-W. Wu, "Using a fuzzy clustering chaotic-based differential evolution with serial method to solve resource-constrained project scheduling problems," *Automation in Construction*, vol. 37, pp. 88–97, January 2014.
- [53] J. Gonçalves, M. Resende, and J. Mendes, "A biased random-key genetic algorithm with forward-backward improvement for the resource constrained project scheduling problem," *Journal of Heuristics*, vol. 17, pp. 467–486, October 2011.
- [54] R. Kolisch, "Serial and parallel resource-constrained project scheduling methods revisited: theory and computation," *Eur. J. Oper. Res.*, vol. 90, pp. 320–333, April 1996.
- [55] W. Spears and K. Dejong, "On the virtues of parameterized uniform crossover," in *Proceedings of the Fourth International Conference on Genetic Algorithms*, R. Belew and L. Booker Eds. San Diego: Morgan Kaufmann Publishers Inc., 1991, pp. 230–236.
- [56] D. Goldberg, *Genetic algorithms in search optimization and machine learning*. Reading: Addison-Wesley, 1989, p. 115.