CHAPTER 5

GENETIC ALGORITHM APPROACH FOR RESOURCE CONSTRAINED PROJECT SCHEDULING PROBLEM

5.1 Introduction to Genetic Algorithms

Genetic algorithm (GA) is a stochastic heuristic search method whose mechanisms are based upon the principal of biological evolution processes discover by Charles Drawn in 1859. GA was first developed by J Holland (1975) that was based on the mechanics of natural selection in biological systems. It is an outstanding method to solve the complex optimization problems. The major benefit of this algorithm is that it provides a robust search in a complex space in compare with other optimization solution (Holland 1987). GA’s are good for both local and global search of the search space because it operate on more than one solution space at once. Holland (1987) explained some of the characteristics of genetic algorithm compared to normal optimization search procedures are: (1) Smart search; (2) Progressive optimization; (3) Global optimal solution. (4) Black-box structure; (5) Good versatility; (6) Parallel type algorithm; (7) Intrinsic learning; (8) Stability.

The general strengths of genetic algorithms lie in their ability to explore the search space efficiently through parallel evaluation of fitness and mixing of partial solutions through crossover and solve multi-criterion optimization problems (Goldberg, 2002). A more specific advantage of GAs is their ability to represent rule-based, permutation-based, and constructive solutions to many pattern recognition and machine learning problems.

Several studies have been done to solve the RCPSP using GA (Hartmann 1998, Alcaraz and Maroto 2001, Hindi et al. 2002, Toklu 2002). GA has also been used successfully to solve construction management problems, including resources scheduling with a small number of activities (Chan et al. 1996, Leu and Yang 1999). A permutation-based GA proposed by Hartmann (1998) makes use of activity list representation. The study also proposed additional two encodings, which include priority value based GA similar to the work of Lee and Kim
(1996). From their computational results, the permutation-based encoding GA outperformed two other encoding algorithms.

GA uses a direct analogy of natural behavior. GA performed random search in large multimodal search space in order to provide the optimal or a near optimal solution. GA starts with an initial population representing a set of random solutions. Each member of this population is called a chromosome which is coded as a binary string of finite length. A collection of chromosomes or solution strings is called a population. It works with a population of individual each representing a possible solution to a given problem. During each generation, the selection of the fittest chromosome is the current generation (g) in the population to be represented in the next generation (g+1). The selection depends on their merit over the fitness if the chromosomes to optimize the objective function. The selection operator selects a chromosome from the current population (p) and the probability of selection of the i-th chromosome is $p(X_i)/\sum F(X_i)$ where $F(X_i) =$ fitness of the i-th chromosome. This kind of selection scheme is called proportionate selection or the roulette wheel selection strategy.

The highly fit individual is given opportunity to reproduce by crossover with other individuals in the population. By crossover, the two selected matches from the parent population are intermixed and generate newer chromosome patterns called offspring. If chromosomes are of fixed length $L$, randomly selecting the crossover point to be the $r$-th position in the gene patterns, where $1 \leq r \leq L$, is called single point cross-over. This is done with a certain crossover probability $P_c$, the value of $P_c$ is selected from the range 0-1. Gene size is number of bits to represents a gene position in the chromosome.

Mutation is another operation in GA which is exchange two neighboring genes without violating precedence relationship in order to create an individual that could not have been produced by the crossover operator. In natural evolution, mutation is a random process where one allele of a gene is replaced by another to produce a new genetic structure. It is needed as a fine tuning operator. The mutation operator was operated as follows: for each individual from a generation, the operator generates a real random number and then swaps an activity after pivot point with activity at pivot point if a random number is equal to or less than mutation probability. Generally the mutation probability $P_m$ is kept very low, in the range of 0-0.5. The
mutation probability is gene based. The mutation operator may operate in different ways. In the binary representation for chromosomes, it generally toggles one or more bit position(s). For other representations, it works differently but its role is to make minor changes to chromosome patterns. In case of real representation of chromosomes, generally Gaussian distribution function is used to update a gene. In GA, the mutation operator acts as a background operator and is typically used to recover lost patterns. Thus using the above three operators, a new population \( P(g+1) \) from the current population \( P(g) \). This process is repeated till some stopping criteria are satisfied.

Most of the genetic algorithms have three main operators 1) Selection 2) Crossover and 3) Mutation. The Genetic Algorithm is performed in the following steps:

Step 1: Population initialization
Step 2: Calculate the fitness value of each individual
Step 3: Selection
Step 4: Crossover
Step 5: Mutation
Step 6: Analyze the stop condition, if meet stop condition, go to step 7, else go to step 2
Step 7: Output the individual with best fitness value.

The Algorithm of GA:

Begin

\[ G=0 \]

Initialize \( P(g) \)

Evaluate \( P(g) \)

Termination_condition = false

While termination_condition = false do

begin

\[ g = g + 1 \]

Selects parents from \( P(g) \)
Crossover
Mutation
Evaluate P(g + 1)
end
end

5.2 Operators of Genetic Algorithm

Genetic algorithms (GA) are the most popular technique in evolutionary computation research. In genetic algorithms, the representation used is a fixed length bit string. Each position in the string is assumed to represent a particular feature of an individual and the value stored in that position represents how that feature is expressed in the solution. The string is evaluated as a collection of structural features of a solution. The analogy may be drawn directly to genes in biological organisms. Each gene represents an entity that is structurally independent of other genes.

5.2.1 Initial Population

Population generation can be executed randomly or by seeding. Seeding is generating the initial population by high quality solution obtained from another heuristic technique. Seeding can help GA finding better solutions rather more quickly than it can do from a random start. However, there is a possible disadvantage in that the chance of premature convergence may be increased.

5.2.2 Selection

The selection phase plays an important role in driving the search towards better individuals and in maintaining a high genotypic in the population. The selection operation mimics the survival of the fittest concept of natural genetic systems. The chromosome that process better fitness value will have higher probability of appearing in the next generation.
The roulette wheel strategy is used for selection for individual which ensures that the probability of selection of the fittest candidate will be the maximum. If we rotate the wheel, the probability that the wheel comes to a stable state with the arrow head pointing to any marked region on the wheel is determined by the percentage of area covered by the region. Similarly in the selection scheme if the sum of fitness of all the strings be represented by the total area of the wheel, where the marked areas represent the fitness of individual strings, then the probability of their selection depends on their fitness value.

5.2.3 Crossover

The most important genetic recombination operator is called crossover. Crossover is the necessary operation for the genetic reproduction. New genes are reproduced from randomly selected genes. Couples, namely the parents, are determined by randomly generated numbers and new two genes are reproduced from parents by crossover operation. The location of the crossover is also determined by generating a random number. After the crossover new two genes are generated by the existing gene combination of the population.

A commonly used method, called one point crossover, multipoint crossover which exchange portions of their representation. Eshelman et al. (1989), worked on multipoint crossover that examined the biasing effect of traditional one point crossover and considered a range of alternatives. Eshelman et al. (1989) concluded that simple crossover has considerable positional bias and the bias may be against the production of good solutions. In multiple point type of crossover operations, the gene is divided into equal intervals, which is same with the crossover point number and for each interval simple crossover operator is executed.

Parent 1

0 1 0 1 0 1 0 0

Crossover Point

Parent 2

0 1 0 1 0 0 1 0
After crossover

Offspring 1

\[
\begin{array}{cccccc}
0 & 1 & 0 & 1 & 0 & 0 \\
\end{array}
\]

Offspring 2

\[
\begin{array}{cccccc}
0 & 1 & 0 & 1 & 0 & 1 \\
\end{array}
\]

Figure 9: One point crossover operator

5.2.4 Mutation

Mutation prevents domination of a certain gene which has high probability of survival. Crossover can produce good fit genes from existing genes, but it cannot generate a new gene for a specific portion which does not exist in the population. Therefore, mutation operator has significant importance as it can produce new gene combinations, which is not generated at the initialization of the population.

Mutation will alter the value at any arbitrary gene position and every gene position will have the equal probability of undergoing this kind of changes. A new string is generated from the old one by the mutation process. The mutation process introduces some extra variability into the population even if it is performed in a very low probability. It has an important role in the generation process. Any accidental loss of information due to crossover may be recovered by this process.

Before Mutation

\[
\begin{array}{cccccc}
0 & 1 & 1 & 0 & 0 & 0 \\
\end{array}
\]

After mutation

\[
\begin{array}{cccccc}
0 & 1 & 1 & 0 & 1 & 0 \\
\end{array}
\]

Figure 10: Mutation operation
5.3 Genetic Algorithm for Resource Constrained Project Scheduling Problem

5.3.1 Background Study

The resource constrained project scheduling problems (RCPSP) is one of the most challenging and particle applications in product development, product planning, construction planning and scheduling, software projects etc. For over the decades, RCPSP has received attention of researchers for solving these problems with various exact methods i.e. mathematical programming, dynamic programming, zero-one programming and branch and bound method using mathematical models but the disadvantage is that it could not solve the bigger and more complicated problems in practice.

Researcher attempts use heuristic rules to overcome these problems. The heuristic used priority rules as the thumb rule to determine priority among activities completing for resource availability. It combines one or more priority rules and schedule generation schemes to generate one or more schedules. These heuristic procedures generally produce solutions to the RCPSP in a reasonable amount of time and depend upon size of the project network and type of resources used for the project. However, studies by Copper (1976) indicate that it is not possible to determine a priority of best set heuristic rules for a particular problem. Hegazy (1999) has explained inconsistent with the quality of results produced on the project network. The use of heuristic rules is not guaranteed optimal solution but significantly it produces very good feasible solutions.

Finally, various meta-heuristic methods, such as genetic algorithm, simulated annealing, tabu search and ant colony have been applied to the RCPSP to overcome the limitation of exact optimal methods and priority rule based heuristics. Most of the meta-heuristics are activities listed representation in the RCPSP make use of the serial scheduling scheme as a decoding procedure (Boctor 1996, Hartmann 1998, Kim and Ellis 2008). A few meta-heuristic used parallel generation schemes as a decoding procedure (Lee and Kim 1996). It is recognized that the selection of serial scheme for all optimal solutions from the search space may not include
when applying in parallel scheme, whereas the search space of the serial scheme always has an optimal solution (Kim 2009).

Genetic algorithm (GA) has been recognized as a powerful and application optimization method for RCPSP. The GA is meta-heuristic with biological concept of survival of fittest. This technique has been found more effective at obtaining an optimal or sub-optimal solution than the analytical techniques in RCPSP (Holland 1975, Goldberg 1989). The GA model scheduling starts the activities in a single project using a serial scheme of allocation with the objective of minimizing the different between resource availability and utilization.

The most important objective is to minimize the project duration under limited resource constraints. In the view of minimizing the deviation of the required from the availability of resources, it intends to develop a more effective elitist algorithm that can find optimal or near optimal schedule with multiple resource constraints. This research proposed a permutation based elitist genetic algorithm to search for optimal and/or suboptimal solution to the RCPSP.

5.3.2 Problem Definition and Formulation

RCPSP consists on executing a group of activities limited by constraints. Each activity in a project has a corresponding duration and also needs certain amount of resources such as labor or materials to execute. The precedence relationship is force to some activities to begin after the finalization of others. In addition, processing every activity requires a predefined amount of resources, which are available in limited quantities in every time unit. The aim of the limited resources the shortest duration is to make the daily resources requirement no more than daily supply, to make full use of limited resources and make the total duration as short as possible. The assumptions of these problems are that the availability of resources is constrained to some maximum value that the project has to complete using the given resources. The RCPSP is characterized by a factorial growth in the amount of computation required to consider all possible solution as the problem size increases.

The objective function is formulated for exploring a solution to the RCPSP. The objective function are minimizing project duration, minimizing total project cost, maximizing net present value etc. Minimizing project duration is used in this study as an objective function RCPSP.
The GA has been applied successfully to scheduling problems in the area of manufacturing and construction: Chan et al. 1996; Hegazy 1999; Leu and Yang 1999; Toklu 2002; Zhang et al. 2006. Hegazy (1999) solved resource allocation problems using GA where the concept of minimizing total slack was incorporate in the decision variable. Leu and Yang (1999) developed a GA-based multi-criteria optimal model for construction scheduling, which incorporated resource allocation, resource leveling, and time–cost trade-off problems into a unified system. Jin-Lee Kim et al. (2008) used of both permutation-based encoding and decoding procedures and the elitist strategy for optimization of large sized project networks in RCPSP. This research proposed an elitist genetic algorithm to search for optimal and/or suboptimal solutions to the RCPSP of large sized project networks.

The objective of resource constrained project scheduling is to minimize the duration time of a project without exceeding the resource limits. Resource constrained allocation is one type of sequencing problem. Using GA to solve, a string of population represents a possible sequence of activities and each characteristic in the string stand for an activity ID (or name) in the lower position has a higher priority for allocation of resources.

RCPSP is describe as a project having a set of n + 2 activities that need to processed j = 1, 2, ...n, n + 1, n + 2. The activity 1 and n + 2 are consider dummy activity marked start and end activity. These activities are not considered in resource allocation and having a duration of 0 time unit. The structure of the project is represented by an activity-on-node network G = (V, E) in which V denotes the set of vertices (node) representing activities and E is the set of edges (arcs) representing the finish-start precedence relationships with zero time-lag.

Activities are related with two types of constraints. First, precedence relations are finish-start (FS) type that is, j activity may just be started when i activity is concluded. For each activity j there is a predecessor activity P_j are all less than j, that is, for P_i ∈ P_j, it needs to meet the condition i < j. A second type of constraint is related to resources. Activities consume resources that are provided in limited quantities. Suppose the completion of the activity i, j ∈ (1, 2 … n + 2) needs kth resources. There are a total of k renewable resources available during whole duration of project. Resources are in the set k = 1, 2 ... m. Resource is limited with its capacity R_k in each
time period. The duration of the activity denoted as $d_j$ and the amount of resource $k$ used by the activity $j$ in each time unit is denoted as $r_{t,k}$ units of $k$, $(k = 1, 2, ..., m)$ resource type during each period of duration $d_j$ (there is only one way of processing the activities). The activities cannot be interrupted once they are started.

The $d_j$, $r_{j,k}$, $R_k$ parameters are supposed as known, determined and integer; also, $d_1 = d_n = 0$ and $r_{1,k} = r_{n,k} = 0$ for every $k$ $(k = 1, 2, ..., m)$. The RCPSP problem instance is solved if the starting times (or ending times) for each activity satisfying the precedence and resources constraints, in a way that the duration of the project is minimized. The objective of RCPSP is to find minimum project duration for each all resources constraints are met. The optimization model for the minimum project duration with resource allocation as follows:

Minimize $f(i) = \text{Max} \ (t_i + d_i | i = 1, 2, ..., n)$ \hspace{1cm} (1)

Subject to

$t_j - t_i \geq d_i$ for all $S_j$ \hspace{1cm} (2)

$\sum r_{t,k} \leq b_k$, $j \in A(t)$, $(k = 1, 2, ..., m)$, $t \geq 0$ \hspace{1cm} (3)

$f(i) \geq 0$ \hspace{1cm} (4)

Where $f(i)$ is the fitness function of an individual, which means the finish time of activity $i$; $t_i$ is the starting date of activities $i$, $d_i$ is the duration of activity $i$, $S_i$ is the set of successors of activity $i$, $r_{t,k}$ is the resource demand of $k^{th}$ of activity $i$ at the duration of $t$, $b_k$ is the resource limit of the $k^{th}$ resource, $m$ is the total number of resources type and $n$ is the total numbers of activities.

Equation (1) expresses the computation of the project duration, which is the objective function of the RCPSP. Equation (2) takes the precedence relations into consideration between each pair of activities $(i, j)$, where $i$ immediately precedes $j$. This constraint is the difference between the starting times of successor activities and those of predecessor activities should be equal to or greater than the duration of the connecting activity. Equation (3) limits the total resource usage within each period to the available amount. This constraint express that the sum of all available resources allocated to different activities on the same day should not exceed the total number of units available on that day.
Finally, Equation (4) states that the fitness value, which is the finish time of an activity \( i \) of an individual schedule, is always equal to or greater than zero. RCPSP, however, became a solvable problem in a polynomial time if resource constraints are eliminated. The starting times for each activity are calculated as follows:

\[
    s_1 = 0; \text{ and } s_i = \max ( s_i + d_i ), \text{ } i = 1, 2, \ldots, n.
\]

This method generates a sequence of activity as early as it can be, taking into account the precedence relationships. \( ES_i \) is the starting time for \( i \) activity in ES sequence and \( EF \) to \( ES_i + d_i \) that is, the earliest ending time for \( j \) activity.

### 5.4 Proposed Genetic Algorithm for Resource Constrained Project Scheduling

GAs is a robust general purpose search program based on the mechanism of natural selection and natural genetics. In this research, selection presents an elitist GA to search for an optimal solution to the resource constrained project scheduling of large project networks by implementing serial scheduling generation scheme as a decoding procedure. The elitist GA is proposed using following operators: definition phase, evaluation, elitist selection, one point crossover, mutation, fitness calculation and termination. GAs typically works with the collection or population solutions rather than with a single solution. The initial population of possible solutions to the RCPSP is prepared at the very beginning of the operation. GAs, potential solution to the problem of RCPSP is represented as a population of chromosomes and each chromosomes stands for a possible solution. Each individual solution is represented by a single string like entity called a chromosome. A chromosome typically consists of a number of genes.

Two attributes are associated with each gene: its position and its contents which code for a solution. The operators and their functions generate random number for producing an initial population. The chromosomes in the population generate through successive iteration, which are called generation. During each generation, the chromosomes are evaluated using the serial scheduling scheme for calculating a fitness value of each individual. The fitness of each solution is determined by evaluating its performance with respect to objective function. In natural
survival-of-fittest process, best chromosomes which are the potential solutions exchange information to produce offspring.

These offspring chromosomes are created by merging two parent chromosomes using a crossover operator or modifying a chromosome using a mutation operator. A new generation is formed by selecting some of the parents and offspring according to the fitness values and rejecting the others so as to keep the population size constant. During each generation, the chromosomes are evaluated on their performances with respect to the fitness functions (objective functions). Fitter chromosomes have higher survival probabilities. This process is continued for a large number of offspring generations in which the population is evolving (better solution replace unfit solution) until a terminating criteria is met. The final chromosomes hopefully represent the optimal or near-optimal solution to the problem.

### 5.5 Permutation Based Encoding and Decoding

The permutation encoding is used to represent the problem. Hartmann (1998) proposed three encodings: (i) a permutation-based GA (ii) a priority based GA (iii) a priority rule-based GA. Table 22 shows three different types of individual for encoding. The permutation based encoding is called activity-list representation which is proved appropriate for RCPSP.

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Individual I</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permutation-based encoding</td>
<td>( I = (I_1^1, \ldots, I_J^1) )</td>
<td>( I = (3, 5, 2, 1, 6, 4) )</td>
</tr>
<tr>
<td>Priority value-based encoding</td>
<td>( I = (pv_1^1, \ldots, pv_J^1) )</td>
<td>( I = (0.28, 0.42, 0.63, 0.12, 0.78, 0.56) )</td>
</tr>
<tr>
<td>Priority rule-based encoding</td>
<td>( I = (pr_1^1, \ldots, pr_J^1) )</td>
<td>( I = (LST, MINSLK, SIO, LFT, LPT, EST) )</td>
</tr>
</tbody>
</table>

To encode the problem, a schedule representation is a prime structure among the activities. A schedule generation scheme needs to apply for decoding the schedule representation into a schedule. A solution for the RCPSP is represented in chromosome where each gene in the
chromosome represents an activity and its position represents the sequence of that activity to be scheduled. The permutation based representation indicates the sequence to start the activities. An activity in the permutation must appear in a location after all its predecessors. An activity cannot come after the position of one of its successors (predecessors) in the list used for generation of the individual.

Precedence feasibility individuals are generated using the random number generation (Kim 2006). The random number generation creates an initial population for possible solution of RCPSP in which the feasibility for the precedence relationships among the activities in the individual chromosome is tested. The fitness value for the project is obtained from the maximum value out of all fitness values of every activity to be scheduled in a project. The procedure of GA is followed as:

Serial/parallel generation scheme
Selection of elitism chromosome
One point crossover operation
Mutation operation
Termination criteria
Algorithms steps
Test results
Conclusion

5.6 Priority Scheduling with Genetic Algorithms
In RCPSP the greatest problem with genetic algorithms is finding a suitable chromosome representation. The simplest way to generation the population is randomly. Population size (number of chromosomes) is an important factor that affects the solution. Larger population size increases the likelihood of obtaining a global optimum; however it will increase process time. Once the population is generated the fitness of each chromosome in the population is evaluated and calculated. Different approaches have been taken for solving this type of problem with genetic algorithms, including priority based encoding and sequence scheduling and rule based scheduling.
During each generation, the chromosomes are evaluated on their performances with respect to the fitness functions (objective functions). Fitter chromosomes have higher survival probabilities. After several generations, chromosomes in the new generation may be closely identical, or certain termination conditions are met. The final chromosomes hopefully represent the optimal or near optimal solutions to a problem.

GAs generates possible chromosomes by using crossover and mutation operators. Crossover explained in this research is a one-cut-point (1-point) method, which randomly selects one cut-point at parent strings and exchanges the right parts of two parent strings to generate offspring strings. The mutation method is alters one or more genes in a chromosome within a specified range, depending on a predefined mutation rate. Figure 11 shows definition phage of the project, activity sequencing, evaluation phase, Elitist chromosome selection, Roulette wheel selection, one point crossover and mutation operations of GA.
# Serial Generation Scheme

## Definition phase

<table>
<thead>
<tr>
<th>Define activity no</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define predecessor activity</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>Define resource required</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

### Define duration of activity

| 4 | 6 | 2 | 8 | 4 | 10 | 16 | 8 | 6 | 6 | 10 |

## Activity Sequencing

### Activity sequence generation 0

| 2 | 6 | 3 | 8 | 1 | 4 | 7 | 9 | 5 | 11 | 10 |

### Activity sequence generation 1

| 1 | 4 | 3 | 2 | 9 | 7 | 6 | 5 | 8 | 10 | 11 |

### Activity sequence generation n

| 3 | 2 | 1 | 7 | 4 | 6 | 5 | 9 | 10 | 8 | 11 |
**Evaluation Phase (Based on the definition phase)**

<table>
<thead>
<tr>
<th>Activity no</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Date</td>
<td>7</td>
<td>13</td>
<td>9</td>
<td>21</td>
<td>25</td>
<td>31</td>
<td>37</td>
<td>37</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finish date</td>
<td>8</td>
<td>12</td>
<td>20</td>
<td>36</td>
<td>30</td>
<td>34</td>
<td>46</td>
<td>42</td>
<td>42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Objective function value</td>
<td>16</td>
<td>16</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>46</td>
<td>46</td>
<td></td>
</tr>
</tbody>
</table>

**Elitist Chromosome Selection (minimum of maximum)**

<table>
<thead>
<tr>
<th>Selection 1</th>
<th>2</th>
<th>6</th>
<th>3</th>
<th>8</th>
<th>1</th>
<th>4</th>
<th>7</th>
<th>9</th>
<th>5</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection 2</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>9</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>8</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>Selection n</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>7</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>9</td>
<td>10</td>
<td>8</td>
<td>11</td>
</tr>
</tbody>
</table>
**Crossover (One-point crossover Operation)**

<table>
<thead>
<tr>
<th>Parent 1</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>7</th>
<th>6</th>
<th>9</th>
<th>11</th>
<th>4</th>
<th>8</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
</table>

**Elitist Chromosome selection**

<table>
<thead>
<tr>
<th>Parent 2</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>6</th>
<th>7</th>
<th>4</th>
<th>9</th>
<th>5</th>
<th>11</th>
<th>8</th>
<th>10</th>
</tr>
</thead>
</table>

**Roulette wheel selection**

**One point Crossover point**

<table>
<thead>
<tr>
<th>Offspring 1</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>7</th>
<th>6</th>
<th>4</th>
<th>9</th>
<th>5</th>
<th>11</th>
<th>8</th>
<th>10</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Offspring 2</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>6</th>
<th>7</th>
<th>9</th>
<th>11</th>
<th>4</th>
<th>8</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
</table>
Figure 11: Operation of the proposed algorithm

1. Uniform Mutation Operation
2. Evaluate Fitness
3. Evaluate Objective function value
4. Reach termination condition?
5. Yes
   - Get Final Fitness Value
   - Stop
6. No
5.7 Permutation based Encoding and Decoding

5.7.1 Definition Phase

The proposed algorithm is a permutation based encoding (Hartmann 1998). This is called an activity list representation. A schedule representation is a representation of a priority structure among the activities. A solution for the RCPSP is represented in a chromosome that represents an activity sequence for the problem. A chromosome is also called an individual that is given by an activity sequence. Each gene in a chromosome stands for an activity number. An activity has a lower priority than all preceding activities in the sequence and a higher priority than all succeeding activities.

Each individual in the population is represented by an array of activities in the project. In figure 12, it has shown the activity list representation for a project with N activities. Activity j will be the ith activity chosen to be scheduled. It will be scheduled in its earliest feasible start time. When activity j is chosen to be scheduled, all its predecessors, which will appear in some position 1, 2 . . . i - 1 will have already been scheduled. In this way, the related schedule will always be a feasible schedule. In the definition phase, we have defined the resource required, the duration to finish each individual activity, predecessor relationship of each activity. Table 20 shows example of a project definition phase.

![Figure 12: Activity list representation](image)

Table 23: Definition phase of the project I

<table>
<thead>
<tr>
<th>Definition Phase</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define activity no</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>Define predecessor activity</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>Define resource required</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Define duration of activity</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>8</td>
<td>4</td>
<td>10</td>
<td>16</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>10</td>
</tr>
</tbody>
</table>
We have developed a random number generator to generate precedence feasible permutation in visual C++ program coding. The random number generator creates an initial population for possible solutions to the RCPSP. For example, an individual (3,1,4,2,5,6,8,9,7,11,10) for 11 activities. The fitness value for the project is obtained from the maximum value out of all fitness values of every activity to be scheduled in a project. The project duration is then obtained by comparing the finish time of the last activity and the fitness value of the activity.

5.7.2 Serial Scheduling Generation Scheme

The serial schedule generation scheme was developed by Kelly (1963). In RCPSP, a serial schedule generation scheme is the mechanism to compute the fitness value as well as the start and finish times of the activities in an individual. The purpose of applying the schedule generation scheme to the individuals in a population is to obtain schedules that showed the resource profile and the project duration. Figure 13 shows project duration with the resource profile. It consists of the n stages, which is the same as the number of activities to be scheduled. A set of activities to be scheduled can be classified as two disjointed activity sets: scheduled set and decision set. In scheduling set $S_n$ the activities that are already scheduled are belong to the partial schedule. The decision set $D_n$ contains the unscheduled activities with every predecessor being in the scheduled set.

For example of an individual of 11 activities which is randomly generated. Table 24 shows the randomly generated activities. As per the randomly selected activities in each individual, the first activity selected based on the sequence. Activity 2 is selected in stage 1 first because it is first position of all activities in the individual, I. It has zero predecessor relationship so it can start from zero. Activity 2 is scheduled at time 0 for the duration 6 days as it required resource is not exceeding the availability of resources. Figure 13 shows the serial scheduling generation scheme for all stages.

Table 24: Example of randomly generated activities

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>6</th>
<th>8</th>
<th>3</th>
<th>1</th>
<th>4</th>
<th>7</th>
<th>9</th>
<th>5</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
</table>
The activity 2 required 6 resources hence the resource left is 2 as the maximum available resource is 8. In stage 2, activity 6 is next selected as per the sequence list of individual. Activity 2 is the immediate predecessor of activity 6. The activity 2 is completed hence the activity 6 carry forward for the activity schedule. If the predecessor activity does not completed in the scheduling process then it is selected next activity in sequence list of individual and continue till it found predecessor activity exit in the sequence.

In stage 2, the resource requirement for the activity 6 is 2 units and duration to complete is 10 days. The resource required for the activity 6 is less than available resource. So the schedule is between time 7 and time 16. The resource left after the activity 6 schedules is 6 units in this span of time frame.

In stage 3, activity 3 is selected as per the sequence list of individual. Activity 3 can be scheduled at the earliest possible time because it has no precedence relationship with any other activities. Activity 3 cannot be scheduled at time 0 due to the limitation of available resources. Activity 3 requires 4 resources. Therefore, activity 3 is scheduled at time 7 as its resource requirements 4 do not exceed its resource availability throughout the period being scheduled. The activity 3 schedules for 2 days from time 7. The serial scheme continues in the same way up to the last activity in the individual is scheduled.
5.7.3 Elitism Selection

Selection is the procedure by which better than average solutions are determined for recombination. The selection criterion is used to choose parent chromosomes for crossover operations. The proposed algorithm is select the elitist chromosomes with minimum of maximum strategy that find the best chromosome to the new population, whereas the rest of the population remains as normal. It then selects parent chromosomes using roulette wheel selection method. The roulette wheel selection operator is then employed, as used in many studies (Hartmann 1998); Leu and Yang (1999). The concept of the selection is to determine selection probability for each individual proportional to the fitness value. The selection of elitism chromosomes enables the proposed algorithm to increase the performance of the genetic algorithm. It preserves the best individual generated to the previous generation $t$ into the current generation $t+1$, if the fitness value of an individual in the current population is larger than that of every individual in the current population. Among all the individuals from previous generations, the best individual, elite, is selected for the new population.

5.7.4 One Point Crossover

The crossover operation corresponds of concept of mating. It combines pieces of information coming from different individuals in a population. It is the principal mechanism by which GA arranges for good schemas present on different chromosomes to a single individual. The variation will be less, if the starting times for schemata are fixed for the genotype because at time 0 the number of activities can be start is limited with the precedence relationships. For example, schema of the genotypes $(1, 3, 2, *, *, *, *, *, *, *)$ or $(1, 2, 3, *, *, *, *, *, *, *, *)$ will be likely to schedule at the same starting time if they do not violate the resource constraints (Kim 2006).

In RCPSP, mostly two different types of crossover operators are used for cross over operations. The union crossover operator $3$, UX3, Leu and Yang (1999) and the one point crossover Hartmann (1998) are identified as good methods for the permutation-based encoding for the solutions to the RCPSP. In our algorithm we have explained about one point crossover operation.
The one point crossover operator is capable of preserving schemata in a more effective manner because it keeps the first half of both parents intact and random.

In this proposed algorithm, we have selected one point cross over operator for permutation based encoding for RCPSP. First, the parent 1 individual is selected by elite strategy that copying the best chromosome based on the objective function. Second, parent 2 individual is selected based on the roulette wheel selection method.

5.7.5 Proposed One Point Crossover Algorithm

The proposed crossover point selection algorithm is enables the program to select the best crossover point. This will enable program to select more random offspring. This randomization process will increase the search space of the fitness function. By this cross over algorithm, the increased randomization enable the program to search the fitness value in more wide space and this will increase the convergence between number of generation and standard deviation with larger space.

To find out the crossover point in both the individuals the activity number simultaneously has to keep adding. In every step the cumulative sum has to be compare. If the value found to be equal then the respective sequence position has been stored. The process carry forward till the penultimate activity in the sequence and in each case the respective sequence number has been noted. The cross over point is the closest to the center of the sequence. It has been considered to select the sequence point closest to the center point of sequence to balance the randomization and the predetermined sequence structure determined based on the definition and predecessor relation. In figure 14, we will consider the first case for more randomization of activities and both is equal distance from the center.
### 5.7.6 Mutation Operation

The mutation is changing two neighboring genes without violating the precedence relationship. Mutation is performed at the gene level but not every gene should be mutated. In this algorithm, for each individual generation, the operator generates a real random number and then swaps an activity after pivot point with activity if a random number is equal to or less than mutation probability. A mutation on an individual does not necessarily change the related schedule because interchanging two activities that have the same start time in the activity sequence.
5.7.7 Termination Conditions

The proposed algorithm terminates to produce the best solution that contains the starting and finishing time of all the activities in the individual, resource fitness values, the project duration, sequencing date. The termination conditions include the number of generations which can be set up at the beginning of the algorithm.

5.7.8 Proposed Algorithm for Computing the Fitness Value of Each Individual

1. Define the project activity number, predecessor activity, resource requirement and activity duration.
2. Get the random sequence of activity number by random number generator operator in C++.
3. Select each activity one by one as per the sequence generated randomly.
4. For each selected activity get the predecessor activity, if the predecessor activity is zero it implies that the activity can start at a specific time.
5. Based on the predecessor activity number and the binary function it has to check whether the predecessor activity is actually exist or not. If the binary operator is 0 the predecessor activity does not exist and in this case the selection will carried forward for next activity to repeat the same, else if it is 1 then that activity sequence has been selected.
6. The start time has been calculated, it is the time greater than the finish date of the predecessor activity.
7. Base on the resource availability and available time span (which has to be fit to the activity duration defined in definition phase) the start time and finish time has been calculated.
8. The finish time has been calculated for each activity.
9. The fitness value is the maximum value of each activity finish date.
10. The available resource based on the different activity start and finish interval has been calculated using visual C++ programming. The sequenced data is the finish date of some activity and duration is the time span from its previous duration, the available resource is the resource available for the next sequenced activity to perform based on the random sequencing.
11. On activating each activity this available resource, time span and sequenced date adjusted based on the start and finishing time of each activity.

12. The final fitness value is the maximum fitness value of each population of sequenced activity string.

13. The fitness value will be calculated based on each population fitness value.

14. The minimum value of each population fitness after cross over and mutation will be the objective function value.

5.7.9 Experimental Results and Analysis

The permutation based elitist GA has been programmed using C++ programming language and tested on Pentium core to duo CPU 2.53 GHz processor under the Windows 7 operating system. The computational experiments are explained to demonstrate the performance of the proposed algorithm. The parameters for the proposed algorithm include population size, crossover probability and mutation probability for global search.

The population size, crossover, and mutation rates are taken, respectively, 200, 0.5, and 0.05 for the solution. Increasing populations size generally result in better solution, but it required more computer memory. The algorithm terminated with the number of generation of 200. Table 23 shows computational result of randomly generated individual I and the schedule. It is tested using a single resource constraint with one point crossover and mutation. The result shows that the fitness value is 46 days. When no crossover is occurred, the computation result cannot obtain the optimal value 46 days. So the one point crossover operation worked well in this algorithm. We have taken two example of randomly generated individual. After running GA the fitness value obtained is 46 days in both the cases. Figure 15, 16 and 17 are shows the resource allocation with one type of resource constrained and the fitness value of individual project I, II and III respectively.

Table 25: Randomly generated activities individual I

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>9</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>8</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>
Table 26: Calculation of objective function of each population string of project I

Table 27: Definition phase for project I

Figure 15: Serial schedule generation scheme project I
Table 28: Randomly generated activities for project II

| Activity | 2 | 6 | 3 | 1 | 4 | 7 | 9 | 5 | 11 | 10 |

Table 29: Definition phase of project II

<table>
<thead>
<tr>
<th>Definition Phase</th>
<th>Activity</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predecessor activity</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Required resource</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Duration of activity</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>8</td>
<td>4</td>
<td>10</td>
<td>16</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

Table 30: Calculation of objective function of each population string of project II

<table>
<thead>
<tr>
<th>Sequencing Phase to Calculate Objective Function</th>
<th>Activity sequence generated by random no generator</th>
<th>2</th>
<th>6</th>
<th>3</th>
<th>8</th>
<th>1</th>
<th>4</th>
<th>7</th>
<th>9</th>
<th>5</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary operator</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Start Date</td>
<td>1</td>
<td>7</td>
<td>7</td>
<td>17</td>
<td>9</td>
<td>13</td>
<td>21</td>
<td>25</td>
<td>31</td>
<td>37</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>Finish date</td>
<td>6</td>
<td>16</td>
<td>8</td>
<td>24</td>
<td>12</td>
<td>20</td>
<td>36</td>
<td>30</td>
<td>34</td>
<td>46</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>Objective function value</td>
<td>6</td>
<td>16</td>
<td>16</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>46</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>Sequenced date</td>
<td>6</td>
<td>8</td>
<td>16</td>
<td>16</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>Time span</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Available resource</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

For example of an individual of 11 activities which is randomly generated. Table 25 shows the randomly generated activities. As per the randomly selected activities in each individual, the first activity selected based on the sequence. Activity 2 is selected in stage 1 first because it is first position of all activities in the individual, I. It has zero predecessor relationship so it can start from zero. Activity 2 is scheduled at time 0 for the duration 6 days because it required resource is not exceeding the availability of resources. Figure 15 shows the serial scheduling generation scheme for all stages.
The resource requirement is 6 hence the resource left is 2 as the maximum available resource is 8. In stage 2, activity 6 is next selected as per the sequence list of individual. Activity 2 is the immediate predecessor of activity 6. The activity 2 is completed hence the activity 2 carry forward for the activity schedule. If the predecessor activity does not completed in the scheduling process then it is selected next activity in sequence list of individual and continue till it found predecessor activity exit in the sequence. In stage 2, the resource requirement for the activity 6 is 2 units and duration to complete is 10 days. So as the resource required is 2 is less than available resource 8 we will schedule it in between time 7 to time 16 and resource left is 6 unit in this span of time frame.

In stage 3, activity 3 is selected as per the sequence list of individual. Activity 3 can be scheduled at the earliest possible time because it has no precedence relationship with any other activities. Activity 3 cannot be scheduled at time 0 due to the limitation of available resources. Activity 3 requires 4 resources. Therefore, Activity 3 is scheduled at time 7 because its resource requirements 4 do not exceed its resource availability throughout the period being scheduled is 6. The activity 3 schedules for 2 days from time 7. The serial scheme continues in the same way up to the last activity in the individual is scheduled. Table 25 shows project profile of individual III and the serial scheduling generation scheme is explained in figure17.
Table 31: Randomly generated activities for project III

| 1 | 3 | 9 | 2 | 4 | 6 | 7 | 14 | 8 | 5 | 12 | 10 | 13 | 11 | 15 |

Table 32: Profile of the elite individual for project III

<table>
<thead>
<tr>
<th>Activity</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predecessor activity</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>14</td>
<td>11</td>
<td>6</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Required resource</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Duration of activity</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>16</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>10</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 33: Calculation of objective function of each population string of project III

<table>
<thead>
<tr>
<th>Activity sequence generated by random no generator</th>
<th>1</th>
<th>3</th>
<th>9</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>7</th>
<th>14</th>
<th>8</th>
<th>5</th>
<th>12</th>
<th>10</th>
<th>13</th>
<th>11</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary operator</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Start Date</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>13</td>
<td>18</td>
<td>25</td>
<td>29</td>
<td>33</td>
<td>39</td>
<td>29</td>
<td>39</td>
</tr>
<tr>
<td>Finish date</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>3</td>
<td>12</td>
<td>9</td>
<td>24</td>
<td>17</td>
<td>25</td>
<td>28</td>
<td>32</td>
<td>38</td>
<td>44</td>
<td>38</td>
<td>42</td>
</tr>
<tr>
<td>Objective function value</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>12</td>
<td>12</td>
<td>24</td>
<td>24</td>
<td>25</td>
<td>28</td>
<td>32</td>
<td>38</td>
<td>44</td>
<td>38</td>
<td>44</td>
</tr>
</tbody>
</table>

**Maximum resource level**

<table>
<thead>
<tr>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
5.8 Comparison with Other GA Method

The comparison of convergence of standard deviation is evaluated for the fitness search process for the algorithm. Figure 18 shows the convergence behavior of standard deviation. We compare the results obtained from permutation based elitist GA (Kim et al. 2006). Compare with other heuristic method, the proposed genetic algorithm elitist process find the best individual solution for the next generation so that improve solutions are obtained. The algorithm prevents premature convergence. The proposed one-point crossover operator and mutation operator and a single resource are used with a fixed resource availability to make comparison with the result. The overall performances of the algorithm find the same fitness value which can be consider an optimal and/or near optimal solution to the RCPSP.
5.9 Conclusions
A scheduling generation scheme is most important for the heuristic scheduling process. This research addressed a permutation based elitist genetic algorithm for searching for the optimal solutions to the large-sized multiple resource constrained project scheduling problems. The proposed algorithm produces a suboptimal and/or optimal solution to the large-sized multiple RCPSP.

The proposed algorithm is searched competitive solution as compare to the heuristic method. The proposed algorithm is a good combination of activity-list representation and elitist strategy to search optimal and/or near optimal solution to the RCPSP. The performances of the GA operators used in this research shows that the algorithm demonstrated its performance and accuracy through the comparisons with optimal and lower bound solutions. Hence the proposed genetic algorithm is a powerful tool for scheduling single and multiple resource constrained projects.

In this research, a genetic algorithm procedure has been proposed to produce an optimal and/or near optimal solutions. The permutation based elitist GA shows the performance of search for the optimal and/or suboptimal solution close to the lower bound solutions of RCPSP. The proposed one point crossover selection algorithm makes combination of the individual more randomly to produce best individual for next generation. This randomization process will increase the search space of the fitness function.

The proposed algorithm is searched the fitness value in more wide space and converges between number of generation and standard deviation with a large space. Compare with the other GA algorithm, the proposed algorithm preserved the best solutions to the next generation so as to improve solutions are obtained.