An Artificial Immune Algorithm for Minimizing Total Cost of Resources in the Resource Constrained Project Scheduling Problem

Ramin Golestaneh
Azizollah Jafari
An Artificial Immune Algorithm for Minimizing Total Cost of Resources in the Resource Constrained Project Scheduling Problem

Ramin Golestaneh
Department of Industrial Engineering,
MSc. student, University of Science and Culture, Tehran
R.golestaneh@usc.ac.ir

Hossein Karimi
Department of Industrial Engineering,
PhD student K.N.Toosi University of Technology
hkarimi@mail.kntu.ac.ir

Azizollah Jafari
Department of Industrial Engineering,
Faculty of Engineering, University of Science and Culture,
jafari@usc.ac.ir

Mohammad Khalilzadeh
Department of Industrial Engineering,
Science and Research Branch, Islamic Azad University
(SRBIAU)

Abstract—In this article, an Artificial Immune Algorithm (AIA) for minimizing total costs of both renewable and non-renewable resources in the Resource-Constrained Project Scheduling Problem (RCPSP) is presented. We assume renewable resources that are limited in number, are restricted to very expensive equipment and machines, therefore they are rented and used in other projects, and are not available in all project periods. In other words, there is a predefined ready date as well as a due date for each renewable resource type, so that no resource is used before its ready date. However, resources are permitted to be used after their due date by paying penalty costs depending on the resource type. The objective is to minimize the total costs of both renewable and non-renewable resource usages. For this purpose, we present a metaheuristic algorithm namely Artificial Immune Algorithm (AIA) inspired by the vertebrate immune system to solve this problem.

In order to examine the performance of this algorithm, data derived from studied literature were used, and their answers were compared with those of the Simulated Annealing (SA) algorithm. Results show that in average, quality of AIA answers was better than those of the SA algorithm. Moreover, AIA was more sustainable.

Keywords- Resource Constrained Project Scheduling, Total Cost of Resources, Artificial Immune Algorithm

I. INTRODUCTION

The Resource-Constrained Project-Scheduling Problem (RCPSP) is one of the most important issues in the areas of project scheduling and combinatorial optimization. RCPSP includes a project that has a number of specific activities with certain durations. RCPSP includes two constraints, precedence constraints in which technical precedence relation between activities is finish to start, and the other, resource constraints. The objective here is to minimize project makespan. In RCPSP, While an activity is being executed, preemption is not permitted, i.e., once started, an activity should be continued until it is finished. It is shown in Blazewicz, Lenstra, and Rinnooy-Kan [1] that the RCPSP, as a job-shop generalization, is NP-hard in the strong sense. Several solution procedures have been presented in the literature. They can be classified into three categories: exact methods such as works of Demeulemeester and Herroelen [2], Mingozi et al. [3], and Patterson et al. [4], which mainly make use of various branch-and-bound procedures; heuristic methods based on the serial and the parallel schedule generation schemes of Boctor [5], Kolisch, and Drex [6]. Finally, metaheuristic methods based on taboo search from Baar et al. [7], Nonobe and Ibaraki [8], simulated annealing of Bouleimen and Lecocq [9], Cho and Kim [10], and genetic algorithms form Alcaraz and Maroto [11], Alcaraz et al. [12], Hartmann [13], Mendes et al. [14], and Valls et al. [15]. Surveys on several other solution procedures can be found in works of Demeulemeester and Herroelen [16]. The RCPSP under minimization of total costs of resources is an applicable problem and a modified version of the RCPSP in which all assumptions and constraints of the RCPSP are maintained, but the objective function is different. Moreover, several exact, heuristic and metaheuristic algorithms are proposed for scheduling problems with objective function related to tardiness. Nadjafi and Shadrokh [17] developed a B&B algorithm for the weighted earliness–tardiness project-scheduling problem using generalized precedence relations. Liaw, Lin, Cheng, and Chen [18] developed a B&B algorithm
for scheduling unrelated parallel machines for minimizing total weighted tardiness. Essafi, Yazid, and Dauzère-Pérès [19] presented a genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem. Bilge, Kıracı, Kurtulan, and Pekgün [20] developed a tabu search algorithm for the parallel machine total tardiness problem. In addition, Bilge, Kurtulan, and Kıracı [21] presented a tabu search algorithm for the single machine total weighted tardiness problem. Bianco, Dell’Olmo, and Speranza [22] referred to resources that can be assigned to only one activity at a time in dedicated form. However, in this article, we assume that there exist only a few renewable resources such as very expert human resources with high skill levels, particular types of cranes, and tunnel boring machines that have to be leased from third party providers. Considering that these limited renewable resources are employed in other projects, there is a dictated ready-date as well as a due-date for each of them, such that no resource can be accessible before its ready-date; however, these resources are allowed to be used after their due dates by paying penalty cost, depending on the resource type. In addition, we suppose that there are a few non-renewable resources like budget, materials, energy, or other resources consumed during the project. Ranjbar et al. [23] studied this problem with single mode for each activity, and availability of one unit for each type of renewable resource, without considering non-renewable resources. The problem we studied here is a generalization of the problem introduced by Ranjbar et al., with the difference that we introduce a metaheuristic-based AIA algorithm by considering both renewable and non-renewable resources cost. In addition, the assumption that only one unit of each resource type is available, and no activity needs more than one resource for execution has been removed.

II. PROBLEM MODELING AND FORMULATIONS

In this article, we introduce a resource-constrained project-scheduling problem with finish-to-start precedence relations among project activities, considering renewable and nonrenewable resource costs. For each project, n activities and R renewable resources, and NR non-renewable resources are given. Rk is the availability of each renewable resource. The duration of an activity i is given as di. Activity j requires rjk units of renewable resource k, and njk units of non-renewable resource k. Our model is presented using an activity-on-node (AON) network. Thus, there are two dummy activities, first activity and the last activity (0 and n+1), which denote start and end of the project, respectively. The dummy start and end activities have zero duration and zero resource consumption. It should be noted that for each renewable resource K, ρk, δk, and pk show the ready date, due date, and tardiness penalty cost of this renewable resource, respectively. In order to embed the resource release dates in the network, one dummy node corresponding to each resource k, k=1,...,R, is added to the project network. This node displays an activity with duration ρk with no resource requirements, which is a direct successor of the start dummy activity and direct predecessor of every activity i∈Nj, where Nj is a set of activities that need a number of renewable resources of KεR type for execution. Each type of limited renewable resource is rented for a fixed period, starting from its ready time, and ending with its due-date, and is not available before its ready time, but can be used after its due-date provided a tardiness penalty cost is paid. Non-renewable resources are unlimited. All activities are ready at the beginning of the project, and no preemption is permitted. We define the problem parameters as follows:

n: Number of project activities
NR: Number of non-renewable resources
R: Number of renewable resources
cj: Unit cost of non-renewable resource k
Rk: Renewable resource k availability
ρk: Ready time of renewable resource k
δk: Due date of renewable resource k
Pk: Tardiness penalty cost of renewable resource k for each period
Prj: Set of predecessors of activity j
dj: Duration of activity j
rk: Renewable resource k requirement for executing activity j
nrk: Non-renewable resource k requirement for executing activity j
Tk: Is the renewable resource k tardiness, determined by Tk=max(0,CPk−δk), where CPk is the release time of resource k in the project and equal to CPk=max i∈Nj δk. (Earliest) finish time that is shown with fi, (fi=si+dj), where si is an integer and shows the start time of activity i.

The mixed integer-programming model for this problem can be formulated as follows:

\[
\min \quad z = \sum_{k=1}^{NR} \sum_{j=1}^{R} C_k (\sum_{k=1}^{R} \sum_{k=1}^{R} \sum_{j=1}^{R} P_k T_k)
\]

\[\text{s.t.}\]

\[CP_k \geq S_i + d_i, \quad i \in N_k, \quad k=1,2,...,R\] (2)

\[T_k \geq CP_k - \delta_k, \quad k=1,2,...,R\] (3)

\[T_k \geq 0, \quad k=1,2,...,R\] (4)

\[S_j - S_i \geq d_i, \quad j=1,2,...,n, \quad i \in Prj\] (5)
\[ \sum_{j \in A(t)} \tau_{jk} \leq R_k \quad k = 1,2,\ldots,R \]  
\[ S_i \geq \rho_k \quad i \in N_k \quad k=1,2,\ldots,R \]  
\[ S_v \ CP_v \ T_k \in N^+ \quad \text{for } i=0,1,\ldots,n+1 \]  

In the above model, objective function (1) is the project cost minimization, in which the first and second terms are total costs of using non-renewable resources, and total penalty costs of renewable resources tardiness, respectively. It should be noted that as the cost of renting for each renewable resource is fixed, it does not need incorporation in the objective function. Constraint (2) shows that the release time of each resource is not less than the finish time of each activity, which requires that resource. Constraint sets (3) and (4) ensure that \( T_j \) is equal to max \( \{CP_j-\delta_j, 0\} \). Constraint (5) is the precedence constraint implying that start time of activity \( j \) must be after all its predecessors are finished. Constraint (6) is the renewable resource constraint, where \( A(t) \) is the set containing in-progress activities at time \( t \). Constraint (7) makes the starting times of all activities greater than or equal to the release dates of their corresponding resources. Constraint (8) ensures that variables \( S_i \), \( CP_i \) and \( T_k \) are non-negative integers.

### III. ARTIFICIAL IMMUNE ALGORITHM

An Artificial Immune System (AIS) is a computational algorithm proposed by [24] and inspired by theories and components of the vertebrate immune system. The efficient mechanisms of an immune system make artificial immune systems useful for scheduling problems. AIS is used for solving job-shop [25] [26], flow-shop [27] and resource-constrained project scheduling problems [28]. The biological immune system of vertebrates includes molecules, cells and organs, perform the task of defending the living body against foreign invading substances called pathogens. In order to understand the AIS, some preliminary biological definitions should be characterized:

The vertebrate immune system is able to identify and eliminate disease-causing elements, called antigens, by the use of immune cells, of which B-cells are the most common, and are randomly distributed. These immune cells have receptor molecules on their surfaces namely antibodies, whose aim is to recognize and bind to pattern-specific antigens. The main function of the B-cells is production and secretion of antibodies as a response to exogenous proteins such as bacteria and viruses. Each B-cell is programmed to produce a specific antibody. These antibodies interact with antigens and superior antibodies with higher affinity with antigens are proliferated with higher rates, so that they have more progeny in the next generation of antibodies. The next generation of antibodies is composed by a cell division scheme called clonal proliferation. Antibodies with higher affinity are under concentration in the proliferation phase. Reactions of organism immune systems to pathogens are based on two main concepts namely clonal selection, and affinity maturation. Clonal selection states that by pathogen invasion, a number of immune cells that recognize these pathogens will proliferate. During cellular reproduction, the cells suffer high rates of somatic mutations together with a selective force. Generally, cells with low affinity receptors are mutated at a higher rate and vice versa. The whole process of somatic mutation plus selection is known as affinity maturation. In this section, a problem-solving technique for minimizing the total cost of resources in the RCPSP based on the principles of the vertebrate immune system is presented. Here, antigen refers to the objective function, and antibodies refer to candidate solutions for the Problem.

The most important and key sections of this algorithm are explained in the following subsections.

#### A. Initial Population

Several authors have used random techniques to initiate population. They argue that random initial starting solutions are more diverse, and use less computational effort than heuristic procedures in producing initial solutions [29]. Any solution of this problem (instance) is a vector \( \vec{S} = (S_1, S_2,\ldots,S_n) \), where \( S_i \) is an integer and represents the start time of activity \( i \). The precedence and resource constraint is considered while generating random activity lists.

#### B. Clonal Selection Process

A population of POP solution vectors is generated and for each solution vector, the objective function is determined. Only the best \( P_{\text{cloud}} \%) \) solution vectors will be available for proliferation. For these solution vectors, the corresponding affinity value is determined as follows:

\[ \text{Affinity}(v) = \frac{1}{\text{OFV}(v)} \]

where \( \text{OFV}(V) \) refers to the objective function value of solution vector \( V \). The number of clones of a solution vector \( V \) in the population is given by the affinity of the solution vector \( V \) over the sum of affinities of all population solution vectors multiplied by the number of population elements. Since cloning of the antibodies is done directly proportional to their affinity value, it can be noticed that solution vectors with higher objective function value will appear less frequently compared to solution vectors with low objective function values. The size of the antibody population is fixed and infeasible solution vectors cannot be proliferated.

#### C. Affinity Maturation

Affinity maturation is performed after proliferation. This process is applied in two phases: first, a hyper-mutation...
procedure is applied on each solution vector of the population, and afterwards, the receptor editing mechanism is used.

D. Hyper-mutation

Since a solution vector contains an activity list, a mutation process is applied on it. For the mutation on the activity list, the number of mutations is calculated. The hyper-mutation number for solution vector $V$ can be formulated as follows:

$$\text{numbMut}(v) = \left[ \frac{n \times (OFV(v) - BOFY)}{BOFY} \right]$$

This function is applied for only 15% of $V$ vectors. The lower objective function value, the lower the number of mutations will be. By applying this mutation process, the algorithm can explore the neighborhood of the solution. That neighborhood will expand when the hyper-mutation number increases.

A mutation is defined as follows: in the current activity list, an activity is chosen and is moved randomly to a new position. In order to respect the precedent constraints, the new position of the activity lies between the position of its latest predecessor, and the position of its earliest successor.

E. Receptor Editing

After the cloning and mutation process, a new population of antibodies is generated. Only the best $P_{\text{du}}$% antibodies are preserved in the population. In that way, schedules with a low objective function value stay incorporated in the antibody population, and antibodies evolving to inferior search regions are eliminated. The newly generated population is the start population for a new generation process. This process continues until the stop condition is met.

Note that extensive testing revealed that the optimal values for different algorithmic parameters are $P_{\text{du}}=80\%$, $P_{\text{ed}}=5\%$, and POP=60. Moreover, the algorithm runs in 100 iterations.

IV. COMPUTATIONAL STUDY

The aim of this section is to evaluate the performance of AIA approach by comparing it with SA. The performance is appraised according to the objective function value quality. The solver applied in the study was MATLAB (2009), run with a machine equipped with Windows 7, Intel (R) Core (TM) 2 with 2.53 GHz processor, and 3 GB of RAM.

A. Test problem

We used the sample problems library of PSPLIB [30] and selected two sets of project scheduling problems (i.e. j30 and j60). These data do not include some of our model parameters such as costs of non-renewable resources, penalty cost of renewable resources, and ready time of renewable resources etc. hence, in this article, we have used discrete uniform distribution to select the parameters. Unit costs of non-renewable resources were randomly selected from discrete uniform distribution $(2, 6)$. The unit penalty costs of renewable resource tardiness were randomly chosen from discrete uniform distribution $(10, 30)$. The ready times of renewable resources were randomly generated from discrete uniform distribution $(0, 15)$, and the renewable resource due dates were randomly picked from discrete uniform distribution:

$$\sum_{j=1}^{n} s_j / 5, \sum_{j=1}^{n} e_j / 3$$

Finally, the required non-renewable resources for executing activities were selected from discrete uniform distribution $(0, 10)$. We supposed that the number of non-renewable resources is the same as the number of renewables.

B. Performance of the Proposed Algorithm

In order to evaluate performance of the AIA algorithm, the algorithm was implemented for the introduced test data. Obtained results were compared with results of the SA algorithm. Comparison results are given in tables (1) and (2), which are presented for j30 and j60 data, respectively. The first column of the table shows the Data Name. The ending section of the Data Name, which is in number form, represents the number of random parameter productions that do not constitute the data collection, and were randomly produced according to the previous section method. In fact, for each problem, parameters are randomly produced three-times. Columns 2 and 3 show the name of the metaheuristic algorithm. Each of these columns comprises four other columns, in which the first column shows the minimum result obtained from five repetitions. Columns 2, 3, and 4 represent average, maximum, and standard deviation, respectively. The last row of this table displays result averages for simpler and better comparison.

A criteria namely gap was defined to observe the performance difference between the two algorithms. This criterion is defined as $\text{gap} = 100(\text{Avg}_{\text{AIA}} - \text{Avg}_{\text{SA}}) / \text{Avg}_{\text{SA}}$. In fact, this value shows the amount of AIA improvement compared to SA. Fig. (1) represents this issue for the utilized data. As can be seen, the vertical axis is %gap, and the horizontal axis is data. In the next section, we will provide descriptions of obtained results.
TABLE I. COMPARISON BETWEEN PERFORMANCE OF SA AND AIA FOR J30

<table>
<thead>
<tr>
<th>Name</th>
<th>SA</th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
<th>SDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>j301_1.1</td>
<td>3011</td>
<td>3138.2</td>
<td>3311</td>
<td>157,9152</td>
<td></td>
</tr>
<tr>
<td>j301_1.2</td>
<td>2694</td>
<td>2734.2</td>
<td>2895</td>
<td>89,88993</td>
<td></td>
</tr>
<tr>
<td>j301_1.3</td>
<td>3092</td>
<td>3166.4</td>
<td>3464</td>
<td>166,3635</td>
<td></td>
</tr>
<tr>
<td>j305_2.1</td>
<td>3969</td>
<td>3969</td>
<td>3969</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>j305_2.2</td>
<td>3364</td>
<td>3390.8</td>
<td>3431</td>
<td>36,69741</td>
<td></td>
</tr>
<tr>
<td>j305_2.3</td>
<td>4022</td>
<td>4022</td>
<td>4022</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>j310_1.1</td>
<td>2935</td>
<td>2935</td>
<td>2935</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>j310_1.2</td>
<td>2962</td>
<td>2962</td>
<td>2962</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>j310_1.3</td>
<td>2999</td>
<td>2999</td>
<td>2999</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>j3015_1.1</td>
<td>3311</td>
<td>3311</td>
<td>3311</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>j3015_1.2</td>
<td>3230</td>
<td>3230</td>
<td>3230</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>j3015_1.3</td>
<td>3371</td>
<td>3371</td>
<td>3371</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>3246.67</td>
<td>3269.05</td>
<td>3225</td>
<td>37.57</td>
<td></td>
</tr>
</tbody>
</table>

TABLE II. COMPARISON BETWEEN PERFORMANCE OF SA AND AIA FOR J60

<table>
<thead>
<tr>
<th>Name</th>
<th>SA</th>
<th>Min</th>
<th>Avg</th>
<th>Max</th>
<th>SDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>j601_1.1</td>
<td>4657</td>
<td>4657</td>
<td>4657</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>j601_1.2</td>
<td>5711</td>
<td>5711</td>
<td>5711</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>j601_1.3</td>
<td>5274</td>
<td>5274</td>
<td>5274</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>j605_1.1</td>
<td>6035</td>
<td>6053.8</td>
<td>6129</td>
<td>42.03808</td>
<td></td>
</tr>
<tr>
<td>j605_1.2</td>
<td>5959</td>
<td>5981.8</td>
<td>6037</td>
<td>34.57166</td>
<td></td>
</tr>
<tr>
<td>j605_1.3</td>
<td>5965</td>
<td>6020.8</td>
<td>6058</td>
<td>50.9382</td>
<td></td>
</tr>
<tr>
<td>j6010_1.1</td>
<td>6050</td>
<td>6050</td>
<td>6050</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>j6010_1.2</td>
<td>6193</td>
<td>6193</td>
<td>6193</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>j6010_1.3</td>
<td>6213</td>
<td>6213</td>
<td>6213</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>j6015_1.1</td>
<td>6411</td>
<td>6411</td>
<td>6411</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>j6015_1.2</td>
<td>6154</td>
<td>6154</td>
<td>6154</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>j6015_1.3</td>
<td>6120</td>
<td>6120</td>
<td>6120</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>5933.08</td>
<td>5941.20</td>
<td>5955.17</td>
<td>10.63</td>
<td></td>
</tr>
</tbody>
</table>

C. Discussion

Table 1 shows that the average mean of the results obtained using the artificial immune algorithm for j30 equals 3248.17, which is better than the obtained result by simulated annealing methods, which is equal to 3269.05, and its improvement value equals 20.88. This result also holds true for the average minimum and average maximum. Table 2 shows that the average mean of the results obtained by using the artificial immune algorithm for j60, equals 5941.20, which is better than the obtained result by simulated annealing method, which is equal to 5977.63, and its improvement value equals 36.44. This result also holds true for the average minimum and average maximum. In addition, although by increasing the size of the problem and number of activities, the solution strength worsens than the state in which the number of activities was less. However, the artificial immune algorithm in this case gives better answers than the simulated annealing algorithm.

Another important criterion for performance evaluation is standard deviation. As shown in table 1, the standard deviation average for the resulting answer from the simulated annealing method for j30 is 35.57. However, the standard deviation average for the resulting answer from our artificial immune method equals zero. This result also holds true for j60. Therefore, the resulting answers of the immune algorithm is less scattered than the resulting answers of simulated annealing, which is sign of the better accuracy of the later. As seen in Figure 1, it is clear that the artificial immune algorithm is more efficient than the simulated annealing method algorithm. In this figure, as the gap between these two methods gets larger, indicates that the immune algorithm is more reliable than the simulated annealing method. Hence, points having positive values for the gap indicate the appropriateness of the artificial immune algorithm over the other.
V. CONCLUSION

In this paper, we studied the problem of minimizing total costs of both renewable and non-renewable resources in the resource constrained project-scheduling problem. We formulated and mathematically modeled this problem as mixed integer programming model; then we developed a metaheuristic algorithm called Artificial Immune Algorithm for the proposed

![Figure 1. %gap between selected algorithms](image)

Project-scheduling problem. In the AIA, candidate solutions were considered as immune cells of the organism, and the objective function was considered as the invading instance, namely antigen. The clonal selection and the affinity maturation are considered the main concepts of immune systems. In order to confirm performance of the proposed algorithm, the algorithm was applied to various test problems available in the literature, and reliability was compared with the simulated annealing algorithm. Computational results showed that the proposed algorithm provided competitive results in comparison with the SA algorithm.

REFERENCES

(eds.) Artificial Neural Networks in Pattern Recognition, University of Paisley, pp. 67–84, 2002.


