Hybrid Local Search Methods in Solving Resource Constrained Project Scheduling Problem

Partha Pratim Das
The Heritage Academy, Kolkata, India
partha_pratim_das2004@yahoo.com

Sriyankar Acharyya
West Bengal University of Technology, Kolkata, India
srikalpa8@yahoo.co.in

***Abstract*—Now-a-days different meta-heuristic approaches, their variants and hybrids are being applied for solving Combinatorial Optimization Problems (COP). In this paper Resource Constrained Project Scheduling Problem (RCPSP) has been presented as a COP. This is a common problem for many construction projects. It is highly constrained and is categorized as a NP-hard problem. In our earlier work Simulated Annealing (SA_RCP) outperformed other meta-heuristics, like, Genetic Algorithm, Tabu Search, Particle Swarm Optimization and its variant in solving benchmark instances of this problem. Having been inspired by this result we have further developed new hybrids of Simulated Annealing and Tabu Search. In this work, we have proposed five more methods developed by combining Simulated Annealing and Tabu Search and applied them for solving a benchmark instance of this problem. The results show that Simulated Annealing incorporated with Tabu List, Greedy Selection Heuristic and aspiration criteria (GTSA_AC_RCP) outperforms other methods in getting optimal results with maximum hit and minimum fluctuations.**

***Index Terms*—Resource Constrained Project Scheduling, Local Search, Meta-heuristics, Simulated Annealing, Hybrid Methods**

I. Introduction

Many real-life industrial scheduling problems fall in the domain of Resource Constrained Project Scheduling Problem (RCPSP). A typical RCPSP instance considers a set of tasks. For each task the duration and resource requirements are fixed and known beforehand. Two types of constraints are always associated with a RCPSP. One corresponds to resources and the other is related to predecessor-successor relationships of the tasks. To obtain a feasible solution these two types of constraints must be satisfied. Our objective is to minimize the makespan of the schedule i.e. to schedule the tasks of the project in such a way that the project will be completed in minimum time. As the problems are NP-hard in complexity, deterministic methods are unable to produce acceptable solutions within reasonable time when problem size increases [1].

Initially, integer programming procedures and implicit enumeration (dynamic programming and branch-and-bound) were applied to solve RCPSP instances. During 60's and 70's Critical Path Method (CPM) and the Program Evaluation and Review Technique (PERT) were applied for solving RCPSP. In 1992 Demeulemeester and Herroelen proposed D&H algorithm to solve RCPSP [2]. D&H algorithm was based on a branch and bound approach. But this method becomes computationally intractable as the project size increases (e.g., more than 50 tasks).Work of Brucker et al. [3] and Kolisch and Hartmann [4] focus on heuristic algorithms to solve RCPSP problem. Fendley worked on comparison of several scheduling heuristics to solve RCPSP [5]. Mao [6] used an advanced branch and bound method making use of time increment method of event-driven to solve the exact solution of RCPSP. Advanced iterative algorithm was applied by Shou Yongyi [7] to optimize project scheduling. Cheng combined the heuristic algorithm and branch and bound method to solve this problem [8].

As the problem RCPSP is proved to be NP-hard [1], meta-heuristics are being used in solving this problem. Simulated Annealing (SA) was applied by Cho and Kim [9] to solve RCPSP. Lee and Kim used [10] Tabu Search (TS) to this problem. In 1990 Boctor [11] applied SA to solve this problem. In 2002 Genetic Algorithm (GA) was used by Toklu to solve project scheduling problem with or without resource constraints [12]. GA has also been applied to solve construction management problems, including resources scheduling with a small number of activities [13, 14, 15]. Some variations of GA like, a permutation-based GA [16] and self-adapting GA [17] were proposed by Hartmann in 1998 and 2002 respectively. He also proposed additional two encodings, which include priority value based GA similar to the work of Lee and Kim [10] and priority-rule based GA similar to the work of Dorndorf and Pesch [18].

In 2006 Zhang has shown that to solve RCPSP, like earlier methods heuristics also perform poorly compared

doi:10.4304/jcp.8.5.1157-1166

to meta-heuristic methods [19]. Recent studies reveal that meta-heuristics and their hybrids perform well for this kind of problems. In our earlier work [20] we applied suitable meta-heuristic techniques for solving this problem. The applied approaches are Simulated Annealing [21], Tabu Search [22], Genetic Algorithms [12, 13], Particle Swarm Optimization [19] and Elite Particle Swarm Optimization [23]. A benchmark instance of RCPSP [19] was selected to which the above mentioned techniques were applied and their performances were compared. The performance of Simulated Annealing was observed to be the best. The superiority of Simulated Annealing (SA_RCP) over other methods inspired us in developing several hybrids of SA and some other method and applying them on RCPSP instance.

In this paper we have applied five hybrid meta-heuristics on a typical RCPSP instance. Idea of using tabu list in local search has come from Tabu Search [22, 24]. Incorporation of tabu list always diversifies the search process and help escaping local minima. The basic SA_RCP incorporated with tabu list is a hybrid named here by TSA_RCP. Another hybrid is made by incorporating greedy selection heuristic for selecting a solution from the neighborhood with basic SA, namely, GSA_RCP. The third hybrid obtained from incorporating both tabu list and greedy selection heuristic in SA is named GTSA_RCP. Another hybrid TSA_AC_RCP is generated by including the concept of aspiration criteria in TSA_RCP. Aspiration criteria has been applied to Tabu Search by different researchers [25, 26]. Aspiration criteria has also been considered with GTSA_RCP in order to produce the new method GTSA_AC_RCP. These hybrids have been generated as a result of combining Simulated Annealing [21,27] with Tabu Search [22, 24, 28] and its variant that includes aspiration criteria [25, 26].

A commonly used aspiration criteria is to accept a tabu move only if the move takes us to a state that is better than the best state obtained so far. The comparison of the performance of the methods shows that GTSA_AC_RCP outperforms other methods in producing good quality solution in reasonable time.

We arrange the sections as follows. Section II describes the problem. Section III describes all six meta-heuristic methods. Section IV shows the experimental results and section V concludes the paper.

II. Problem Description

A typical Resource Constrained Project Scheduling Problem may be defined by a set of tasks V′ and a set of resource types E′ [19]. For execution each task x need some resources. The requirement of task x for a resource type r can be denoted by $e_{xr,}$ where e_{xr} indicates the number of instances of resource type r. For each resource type r, the total number of available instances E_r is fixed. The resources held by task x become available for other tasks when x finished its execution. Therefore, the resources are renewable. For each task x its duration is fixed and known. If a task x cannot start its execution until and unless another task y finishes its execution then the task y is called predecessor and the task x is called successor. The relationship between a predecessor and successor is called precedence relationship.

For the graphical representation of a typical RCPSP we assume that there are 0,1,…, N+1 tasks, where task 0 and N+1 are dummy tasks with duration zero. They represent the start time and the finish time of the project respectively. Nodes in the graph represent the tasks and the directed edges indicate the precedence relationships among the tasks. Fig.1 shows the particular RCPSP instance under consideration by the acyclic directed graph G = (V′, F), where V′ is the set of nodes and F is the set of edges. Our basic assumption for this problem instance is that the tasks under the project are non-preemptive, that is, cannot be interrupted in the middle of execution.

The constraints present in RCPSP are as follows:

1) Precedence Constraint: All predecessors of a task x must finish their execution before x starts.

2) Resource Constraint: For each resource type limited numbers of instances available.

A schedule may be defined as an arrangement of tasks. When a schedule satisfies both precedence and resource constraints then it is said to be a feasible schedule. Makespan of a schedule is defined as the time needed to complete the execution of all the tasks of a schedule. Our objective is to produce a feasible schedule with minimum makespan for the given set of tasks. RCPSP can be mathematically formulated as follows:

$$\min\{\max f_x \mid 1 \le x \le N\} \quad (1)$$

subject to:

$$f_x - f_y \ge d_x, \ \forall\ y \in P_x; \text{ where, } 1 \le x \le N. \quad (2)$$

$$\sum_{At'} e_{xr} \le E_r \text{ ; where } 1 \le r \le R; \ \ t' = s_1, s_2, \ldots, s_N \quad (3)$$

where N indicates the number of the tasks present in a project and f_y indicates the finish time of task y ($1 \le y \le N$); d_x stands for the duration of task x, P_x indicates the set of tasks that have been already scheduled (i.e., predecessors) before activity x can start its execution; E_r indicates the available number of instances of resource r ($1 \le r \le R$) and R is the number of the types of resources; e_{xr} is the amount of resource r required by task x, and $A_{t'}$ is the set of ongoing activities at t′ and s_x ($=f_x-d_x$) is the start time of task x. Equation (1) represents the objective, while (2) and (3), respectively represent precedence constraints and resource constraints.

Fig.1 shows all the tasks involved in the project, their duration, their resource requirements and the precedence relationships among them. The RCPSP instance under consideration has 25 tasks and two dummy tasks. Tasks are represented by the nodes in the graph. The resource requirements of each task are shown below the circle corresponding to the task. Each task has fixed duration which is indicated by an integer above the circle corresponding to the task. The project has three types of

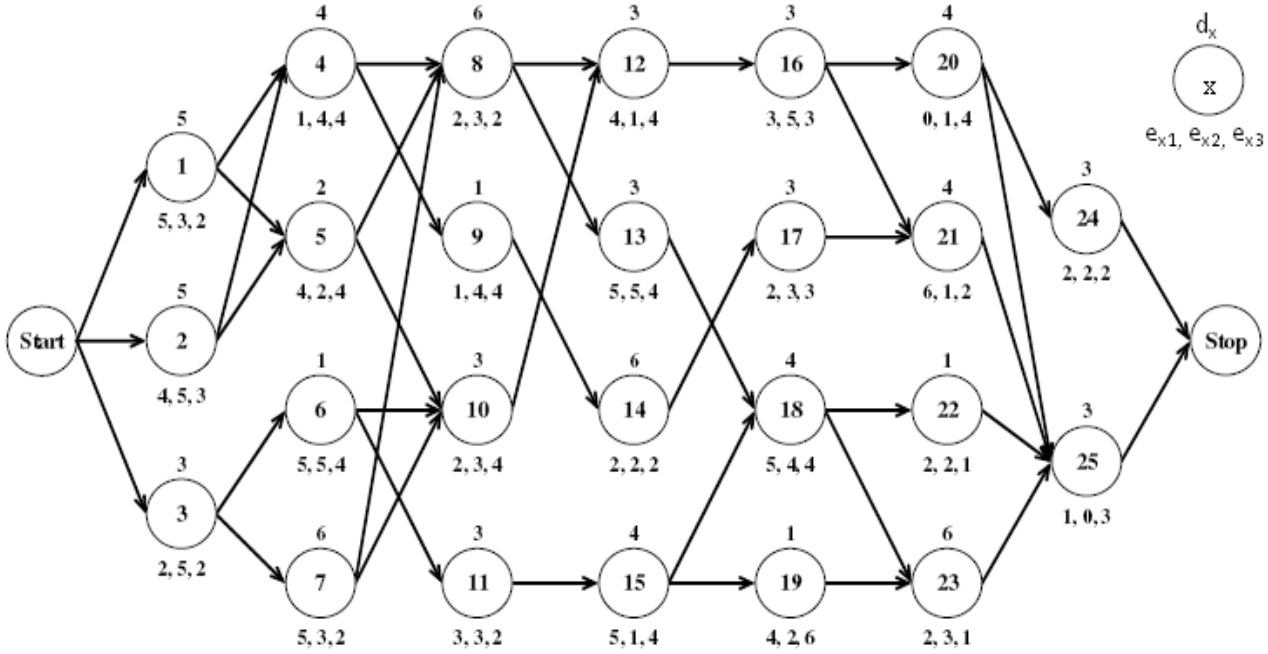

Figure 1. A typical example of RCPSP

renewable resources each of which has six instances. The precedence relationships among the tasks have been shown by the arrows in the figure [19].

III. Solution Methods

As meta-heuristic methods are more useful for solving larger RCPSP instances we have implemented one pure and five proposed hybrid meta-heuristic methods on the problem instance under consideration and compared the performance of those methods. As most of the methods used in this work are new, they are described elaborately in this section.

In our approach every solution is represented by a vector. The length of the vector equals to the number of tasks in the project under consideration. Each element of this vector represents the assigned priority for each task [19]. For example the i^{th} element of this vector represents the priority of i^{th} task. The tasks are then scheduled according to their priority. In our work we took the makespan of the schedule as the value of cost function for all the methods applied here. In meta-heuristic methods the stopping criteria of the algorithm may be defined by maximum number of iterations, by the number of iterations without an improvement of cost function or by a pre-specified threshold value of cost function. Sometimes a combination of aforesaid conditions may also be set as the stopping criteria. In this work for all the methods the stopping criteria is based on the number of iterations without an improvement of cost function. For all the methods the solution is represented by a twenty five element vector.

Here we have applied a parallel transformation scheme to produce a feasible schedule of the given RCPSP instance [19]. The parallel transformation scheme has a series of steps. At each step multiple tasks that satisfy both the precedence and resource constraints are scheduled i.e. more than one task can be executed simultaneously if both the constraints are met. The parallel transformation scheme obeys the following rules to produce a schedule from the priority vector.

1) Scheduling decision is taken at start and whenever one of the scheduled tasks finishes its execution and some tasks are still left to be scheduled.
2) A set of feasible tasks are chosen that satisfy the precedence constraints resource constraints.
3) From that set of feasible tasks, tasks are selected for execution one by one in descending order of their priorities and resources are assigned to them as required.
4) Tasks are scheduled to start execution at the newly determined scheduling time, and then the steps from 1) are continued again if any tasks have not been scheduled so far. Simulated Annealing and other five hybrid meta-heuristic methods are described below in details:

A. Simulated Annealing (SA_RCP)

Simulated Annealing is a well-known local search technique which can solve complex combinatorial problems. The general goal of local search techniques is to find good quality solutions in reasonable amount of time. SA_RCP is a probabilistic method. The idea of SA, was first introduced in 1983 by Kirkpatrick, Gelatt and Vecchi [21] and Cerny in 1985 [29]. It finds the global optima of a cost function that may have several local optima. It is inspired from the physical annealing process in metallurgy. Physical annealing refers to the heating

and controlled cooling of metal. This process brings the material structure from an arbitrary initial state to the arranged randomly. The slow cooling process helps the atoms in finding highly structured configurations having minimum internal energy.

When this concept is adopted for optimization techniques then the target is to minimize or maximize the value of objective function. Here value of the cost function is equivalent to the internal energy state. There we need a control parameter that imitates the temperature of the physical process. This control parameter controls the probability of accepting a solution that is worse than the current one [30, 31]. This probability is calculated by the value of $e^{-\Delta/T}$ where Δ indicates the difference between cost functions of current and neighborhood solution. First we start with a random initial solution and an arbitrary initial value of T. Then we generate a neighborhood solution. If the neighborhood solution is better than the current solution we always accept that and move to that solution. If it is worse, then we accept the neighborhood solution based on the probability calculated by the expression $e^{-\Delta/T}$. The value of control parameter T is gradually decreased throughout the process to increase the intensification as the search reaches the promising area.

In this work the neighbor of a solution (i.e. solution vector) is generated by randomly choosing a position in the vector and then updating it to a new random value [27]. The algorithm is given below:

Step1: Initialize
Step1.1:s ← GenerateInitialSolutionVector(), T ← T_0, s*←s
Step2: While termination conditions not met do
Step2.1: s´ ← PickAtRandom(N (s))
Step2.2: If f(s´) < f(s) then s ← s´,
Else accept s´ as new solution with probability p(T,s,s´)
Step2.3: If f(s´) < f(s*)then s*← s´
Step2.3: Update(T)
Step3: Return s*

Where s is current solution, s´ is neighborhood solution, N(s) is neighbourhood of s, neighbour generated by modifying single element, chosen randomly, of the current vector, f(s) is cost of s, T is Temperature i.e. the control parameter, T_0 is the initial value of T, s* holds best known solution and f(s´) holds cost of s´.

B. GTSA_RCP

For solving Resource Constrained Project Scheduling Problem here we proposed a method combining basic Simulated Annealing, greedy approach and tabu list named Greedy Tabu Simulated Annealing for RCPSP abbreviated as GTSA_RCP. It is a hybrid method. It combines some features of Tabu Search with Simulated Annealing algorithm. The algorithm is similar to Simulated Annealing algorithm except for two cases. We maintain a list L, conventionally known as tabu list that holds some attribute of last few moves [32, 33]. This list prevents the generation of a neighbor whose attribute matches with any element present in the list. So, tabu list minimum possible energy state. When heated the atoms in the metal are displaced from their current positions and actually prevents the generation of recently accepted solution. This algorithm also generates all or a set of neighborhood solutions (in case the number of all the neighbors is very high) of the current solution and then follows a greedy approach i.e. chooses the best among them [32, 33] and then moves to that solution according to Simulated Annealing neighbor acceptance criteria. This implies if the chosen solution is better than the current solution we always accept that and move to that solution. If it is worse, then we accept the neighborhood solution based on the probability calculated by the expression $e^{-\Delta/T}$. The length of the tabu list is a very sensitive parameter here. If the list is too short then the same type of solutions may be generated soon and consequently the search may be trapped in local optima. On the other hand if the list is too long then it may prevent some moves that are necessary to reach the global optima. So, length of the tabu list should be chosen carefully. Earlier in this paper we mentioned that to generate a neighbor we choose a position in the vector arbitrarily and change the priority value randomly. The index of that position has been used as the attribute to be kept in the tabu list. The algorithm is given below:

Step1: Initialize
Step1.1: s ← GenerateInitialSolutionVector(), T ← T_0, s*←s , L=ø
Step2: While termination conditions not met do
Step2.1: Select s´ in argmin[f(s´´)]; s´´∈ N´(s)
Step2.2: If f(s´) < f(s) then s ← s´,
Else accept s´ as new solution with probability p(T,s,s´) and record tabu i.e. index of modified element for the current move in L and delete the oldest entry if necessary
Step2.3: If f(s) < f(s*) then s*← s
Step2.4: Update (T)
Step3: Return s*

Where L is the tabu list, N´(s) is the non-tabu subset of neighbourhood of s and f(s´´) indicates the cost of s´´. Rest of the symbols carry the same meaning as before.

C. TSA_RCP

We proposed another method for RCPSP that incorporates tabu list in basic Simulated Annealing and named it Tabu Simulated Annealing for RCPSP abbreviated as TSA_RCP. This approach is similar to GTSA_RCP. It differs from GTSA_RCP in one aspect. Here instead of generating all or a set of neighbors, we just generate a single neighbor for the current solution and accept that according to basic Simulated Annealing algorithm criteria. Tabu list is used here in the same way and for the same purpose as it was used in GTSA_RCP. The algorithm is given below:

Step1: Initialize
Step1.1: s ← GenerateInitialSolutionVector(), T ← T_0, s*←s , L=ø
Step2: While termination conditions not met do

Step2.1: s′ ← PickAtRandom(N′(s))
Step2.2: If f(s′) < f(s) then s ← s′,
Else accept s′ as new solution with probability p(T,s,s′) and record tabu i.e. index of modified element for the current move in L and delete the oldest entry if necessary
Step2.3: If f(s) < f(s*) then s*← s
Step2.4: Update(T)
Step3: Return s*

The symbols carry the same meaning as before.

D. GSA_RCP

Next method we proposed is also similar to GTSA_RCP. It differs from GTSA_RCP in one aspect. No tabu list concept is present here. Only Greedy approach is used to select the best neighbor among the set of neighbors and the algorithm accepts the best neighbor according to Simulated Annealing neighbor acceptance criteria. This method has been given the name Greedy Simulated Annealing for RCPSP and abbreviated as GSA_RCP. The algorithm is given below:

Step1: Initialize
Step1.1: s ← GenerateInitialSolutionVector(), T ← T_0, s*←s
Step2: While termination conditions not met do
Step2.1: Select s′ in argmin[f(s′′)]; s′′ ∈ N(s)
Step2.2: If f(s′) < f(s) then s ← s′,
Else accept s′ as new solution with probability p(T,s,s′)
Step2.3: If f(s) < f(s*) then s*← s
Step2.4: Update (T)
Step3: Return s*

The Symbols carry the same meaning as before.

E. TSA_AC_RCP

This approach is similar to TSA_RCP. It differs from TSA_RCP in one aspect. Here we have included the concept of aspiration criteria. The concept of aspiration criteria can be defined as a condition that has to be satisfied to allow an otherwise tabu move to be accepted [26, 34]. In our work the aspiration function always accepts a solution which is better than the best solution obtained so far, even if the solution is prohibited by the tabu list if the basic Tabu Search Algorithm is followed. That means when the attributes of the next solution are already in the tabu list, then also that solution may be accepted by this method if it improves the current best solution. As this algorithm has been developed by incorporating the aspiration criteria into TSA_RCP method, we named it Tabu Simulated Annealing with Aspiration Criteria for RCPSP which has been abbreviated as TSA_AC_RCP. The algorithm is given below:

Step1: Initialize
Step1.1: s ← GenerateInitialSolutionVector(), T ← T_0, s*←s , L=ø
Step2: While termination conditions not met do
Step2.1: s′ ← PickAtRandom(N(s))
Step2.2: If f(s′) < f(s) then s ← s′,
Else if s′ has its attributes in the tabu list then s′ is rejected
Else accept s′ as new solution with probability p(T,s,s′) and record tabu i.e. index of modified element for the current move in L and delete the oldest entry if necessary
Step2.3: If f(s) < f(s*) then s*← s
Step2.4: Update (T)
Step3: Return s*

Symbols carry the same meaning as before.

F. GTSA_AC_RCP

This approach is similar to GTSA_RCP. The aspiration function is also included here. In this method also the aspiration function is applied in the same way as it was applied in TSA_AC_RCP method. We named this method Greedy Tabu Simulated Annealing with Aspiration Criteria for RCPSP that is abbreviated as GTSA_AC_RCP. The algorithm is given below:

Step1: Initialize
Step1.1: s ← GenerateInitialSolutionVector(), T ← T_0, s*←s , L=ø
Step2: While termination conditions not met do
Step2.1: Select s′ in argmin[f(s′′)]; s′′∈ N(s)
Step2.2: If f(s′) < f(s) then s ← s′,
Else if s′ has its attributes in the tabu list then s′ rejected
Else accept s′ as new solution with probability p(T,s,s′) and record tabu i.e. index of modified element for the current move in L and delete the oldest entry if necessary
Step2.3: If f(s) < f(s*) then s*← s
Step2.4: Update (T)
Step3: Return s*

Symbols carry the same meaning as before.

IV. EXPERIMENTAL RESULTS

We have applied all six algorithms on the given benchmark instance of RCPSP. The problem instance is standard and widely used [19]. The experiment in this paper is implemented by Microsoft Visual C++. Experiment environment is: CPU 2.6 GHz, RAM 2GB. Same computer has been used for all experiments.

Each of the methods has been applied 100 times or trials (with same parameter values but with different initial solutions generated randomly in order to include diversity in those methods) on the same problem instance under consideration. Table I shows the comparative performance of these methods. The 'Solved' column shows the number of times the problem is solved out of 100 trials, i.e., number of times optimal scheduling time (64) is reached. This criterion measures certainty of the algorithm. The 'Makespan r.m.s.' column denotes the

r.m.s. deviation of the obtained makespans w.r.t. optimal scheduling time. The 'Time' column shows the runtime averaged over solved instances. The 'Execution time r.m.s' denotes the r.m.s. deviations the execution time w.r.t. average execution time over solved instances. So, it can be used as a measure of stability. A lower value in this field indicates that the method has high stability. And the 'AES' column denotes the number of evaluations of the cost function to reach the optimal value, averaged over solved cases out of 100. The efficiency of optimization algorithms is measured on the basis of AES. AES stands for 'average number of evaluations on success'.

From Table I it can be easily observed that the average runtime of the methods do not vary widely. Those values are quite close to each other. It is clear from Table I that among all these four methods GTSA_AC_RCP was able obtained the lowest value for AES followed by GSA_RCP and GTSA_RCP in sequence. Remaining three methods have quite high values of AES compared to the three methods mentioned above. So, GTSA_AC_RCP has performed most efficiently on the problem instance under consideration.

Fig. 2 plots the execution time against each solved instance for each method. So, it can be viewed as the graphical representation of stability of each method. So, the conclusion that we have drawn by observing the values of 'Execution time r.m.s' column of Table I can be verified with this figure.

For a method, the number of iterations needed to reach the optimal value of cost function is different in different trials, where the total number of trials is 100. Depending on the number of iterations there are three cases. In the best case a method reaches its optimal value of cost function in minimum number of iterations. The same to produce the optimal makespan with maximum frequency ('Solved' = 98), where GTSA_RCP is just after it with 'Solved' = 97 and other methods are lagging behind. GTSA_AC_RCP also has obtained the minimum root mean square deviation (r.m.s. = 0.141) on makespan among all the methods. So, in this criterion also GTSA_AC_RCP has outperformed other five methods discussed here. Therefore, depending on the 'Solved' and 'Makespan r.m.s.' GTSA_AC_RCP outperforms all other methods. GTSA_RCP is the next best and other four methods are falling behind. The remaining four methods have reached the optimal makespan with same frequency and have the same r.m.s. deviation. From table I we can conclude that that when stability is concerned SA_RCP is the best method followed by TSA_AC_RCP and TSA_RCP as shown by the values under 'Execution time r.m.s' column. Table I shows that GTSA_AC_RCP has sense in meaning applies to average and worst cases, i.e., in worst case a method reaches the optimal value of cost function in maximum number of iterations. Fig. 3, Fig. 4 and Fig. 5 show how the value of cost function improves with the number of iterations for each method and finally reaches the optimal value for best, average and worst case respectively. Our observation is, in best and worst case GTSA_AC_RCP has reached the optimal value before others. And in the average case it is in third rank. So considering these three cases we can say that GTSA_AC_RCP has performed better than others. TSA_AC_RCP has performed well in average and worst cases. In both the cases its rank is two to reach the optimal value. Performance of GSA_RCP comes next to GTSA_AC_RCP and TSA_AC_RCP. GTSA_RCP performs well only in the best case. Rests of the methods are lagging behind in performance.

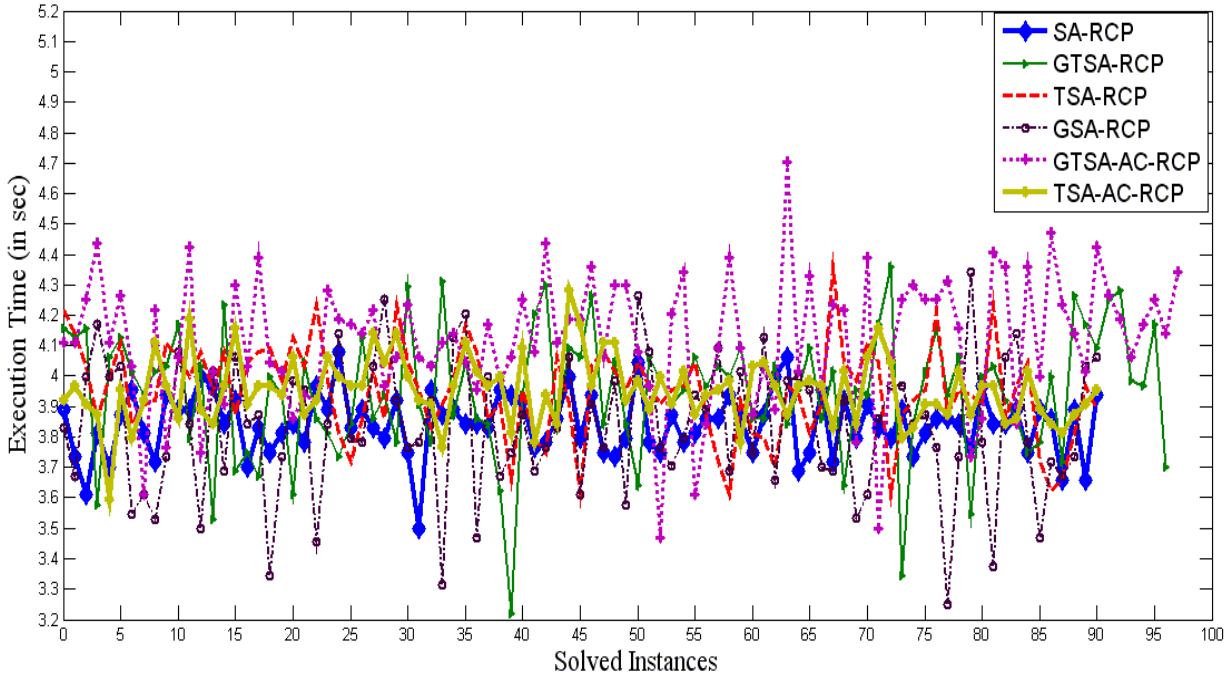

Figure 2. Stability Graph

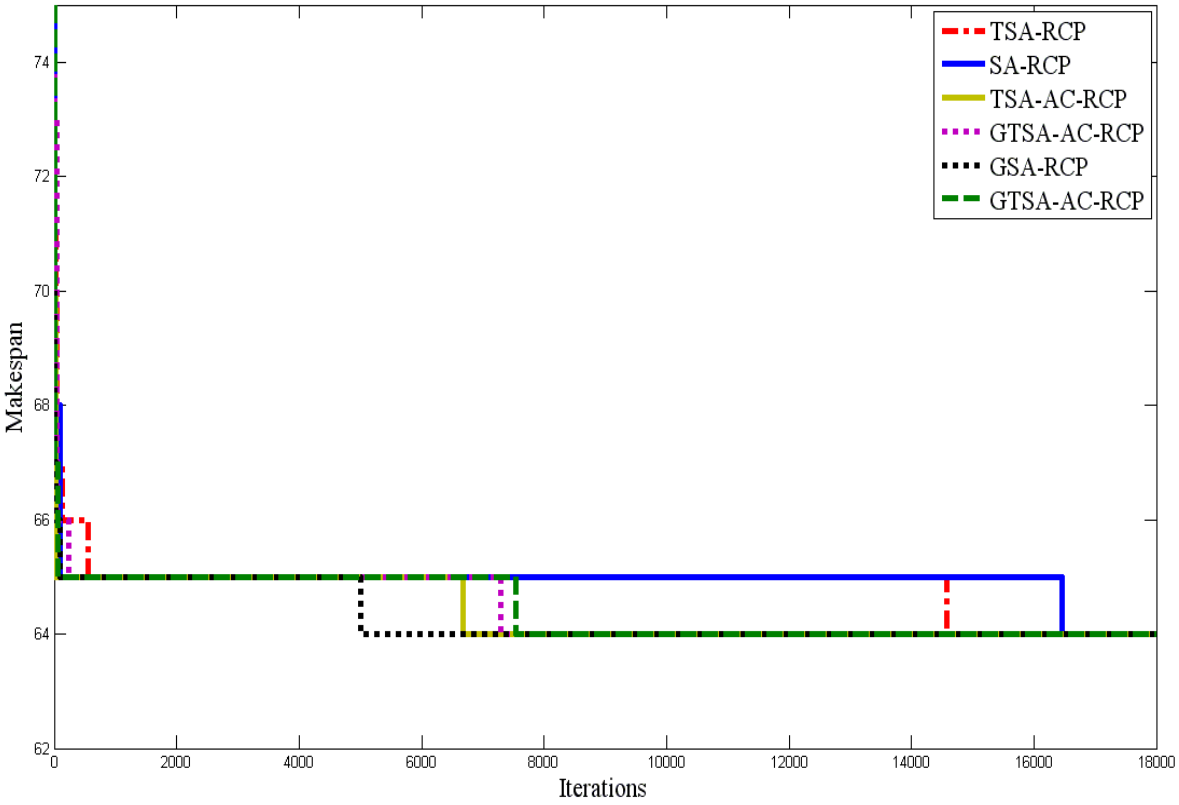

Figure 3. Cost vs. Iteration for best case

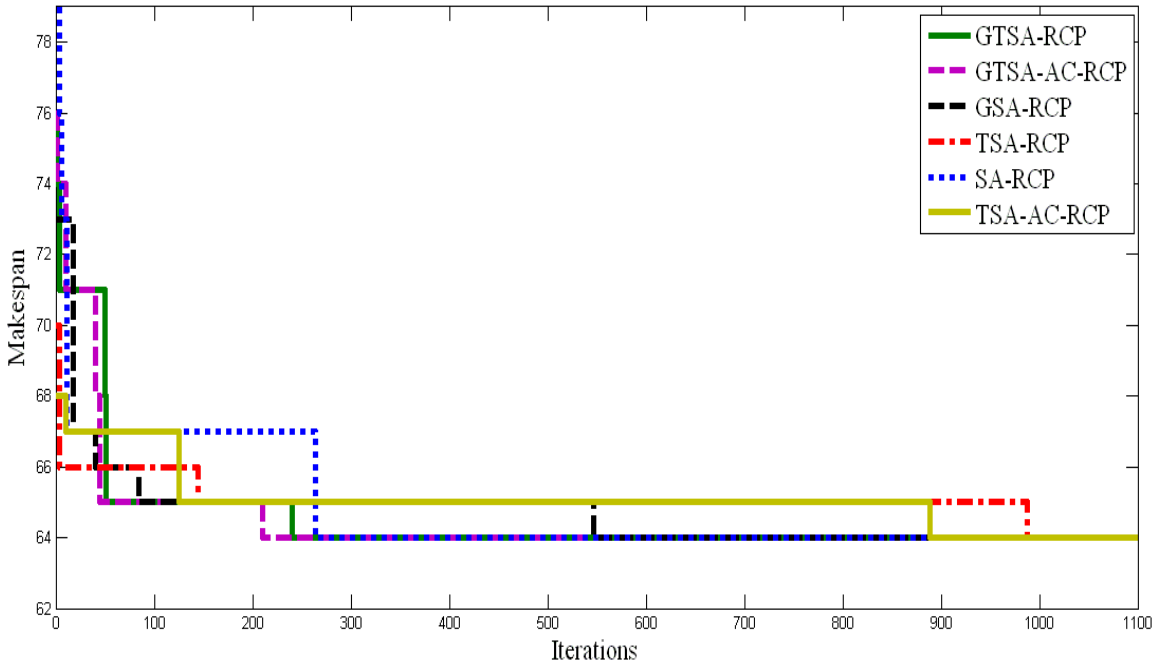

Figure 4. Cost vs. Iteration for average case

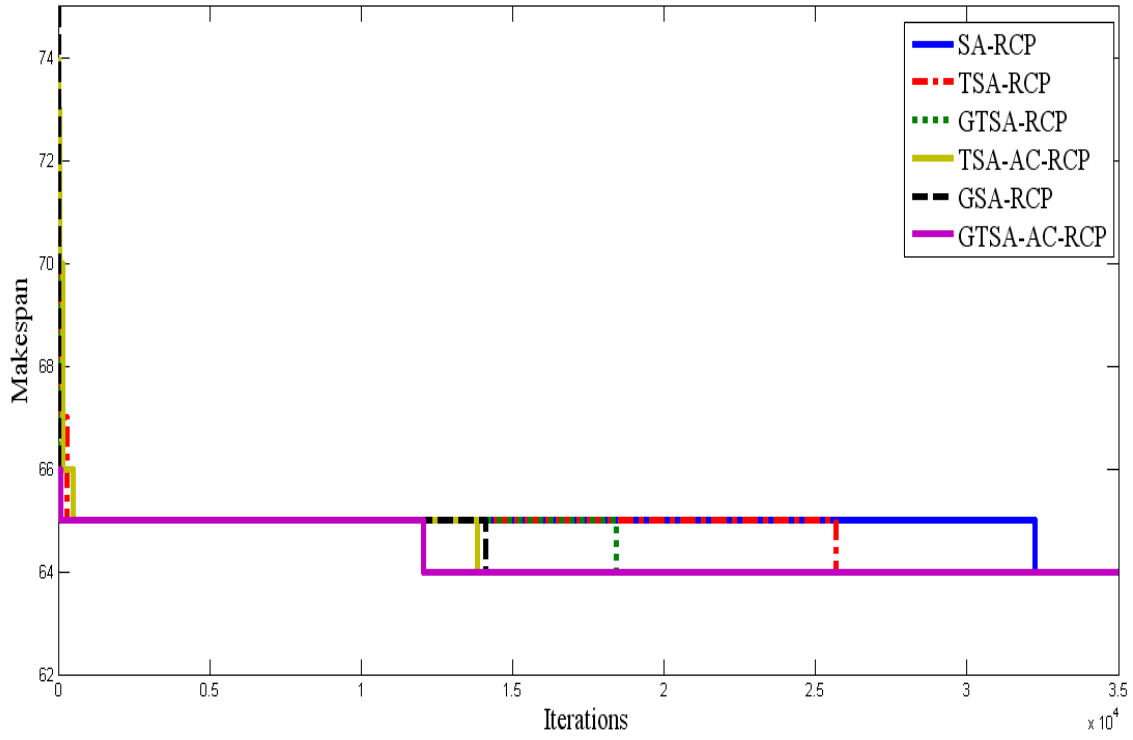

Figure 5. Cost vs. Iteration for worst case

TABLE I.
COMPARISON OF HYBRID LOCAL SEARCH METHODS ON RCPSP

Method	Instances	Solved	Makespan r.m.s.	Time (sec)	Execution time r.m.s	AES
SA_RCP	100	91	0.3	3.894	0.099	15178
GTSA_RCP	100	97	0.1732	4.003	0.208	10351
TSA_RCP	100	91	0.3	3.988	0.152	14279
GSA_RCP	100	91	0.3	3.896	0.222	9912
GTSA_AC_RCP	100	98	0.141	4.094	0.215	9444
TSA_AC_RCP	100	91	0.3	3.989	0.109	15498

V. CONCLUSION AND FUTURE WORK

The Resource Constraint Project Scheduling Problem (RCPSP) is a common problem in most of the construction engineering projects. Methods those are able to solve this problem effectively can significantly improve the productivity of construction industry.

In this work we have emphasized on the hybrid methods because in last few years many of the researchers have shown that for optimization problems the hybrid meta-heuristic methods often perform better than their pure parents. This inspired us to develop some hybrid methods by combining some features of Tabu Search with Simulated Annealing. Apart from basic SA (SA_RCP), five other hybrids are namely Tabu Simulated Annealing for RCPSP (TSA_RCP), Greedy Simulated Annealing for RCPSP (GSA_RCP), Greedy Tabu Simulated Annealing for RCPSP (GTSA_RCP), Tabu Simulated Annealing with Aspiration Criteria for RCPSP (TSA_AC_RCP) and Greedy Tabu Simulated Annealing with Aspiration Criteria for RCPSP (GTSA_AC_RCP). Their comparative performance has been observed and analysed. The optimum value of makespan found for this problem is 64.

Experimental results show that all six methods were able to find the optimum value for this problem. As their average runtime over solved instances are kept close to each other, it will be quite fair to compare their performance based on r.m.s. deviations on makespan and success rate. In consideration of those two criteria we see that GTSA_AC_RCP outperformed other methods in many criteria.

There are more complicated RCPSP with multiple objectives. Incorporation of uncertain activity duration and pre-emption need to be analysed in future. As optimal parameter setting plays a crucial role in producing good quality result that should also be focused and exercised.

REFERENCES

[1] J. Blazewicz, J.K. Lenstra, and A. H. G. Rinnooy Kaw, "Scheduling Subject to Resource Constraints: Classification and Complexity", Discrete Applied Mathematics, 5(1), 1983, pp.11-24.

[2] E. Demeulemeester and W. Herroelen, "A Branch and Bound Procedure for the Multiple Resource Constrained Project Scheduling Problem", Management Science, Vol. 38, No. 12, 1992, pp. 1803-1818.

[3] P. Brucker, S. Knust., A. Schoo and O. Thiele, ''A Branch and Bound Algorithm for the Resource Constrained Project Scheduling Problem'', European Journal of Operational Research, Vol. 107, No. 2, 1998, pp. 272-288.

[4] Rainer Kolisch, Sonke Hartmann, "Experimental Investigation of Heuristics for Resource-constrained Project Scheduling: An Update", European Journal of Operational Research, Vol. 174, No. 1, 2006, pp. 23-37.

[5] L. G. Fendley, "Toward the Development of a Complete Multiproduct Scheduling System", J. Industrial Engineering, Vol. 19, 1968, pp. 505-515.

[6] Mao N, Chen Q X and Chen X, "An Extension to the DH Branch and Bound Algorithm for MRCPSP", Control Theory and Applications, Vol. 18, 2001, pp. 119–126. (in Chinese)

[7] Shou Y Y, "Iterative Technique for Scheduling Resource Constrained Multiple Projects", Journal of Zhejiang University (Engineering Science), Vol. 38, 2004, pp. 1095–1099. (in Chinese)

[8] Cheng X and Wu C Q, "Hybrid Algorithm for Complex Project Scheduling", Computer Integrated Manufacturing Systems, Vol. 12, 2006, pp. 585–589. (in Chinese)

[9] J.H. Cho and Y.D. Kim, "A Simulated Annealing Algorithm for Resource Constrained Project Scheduling Problems", Journal of the Operational Research Society, Vol. 48, No. 7, 1997, pp. 736-744.

[10] J.K. Lee and Y. D. Kim, "Search Heuristics for Resource Constrained Project Scheduling", Journal of the Operational Research Society, Vol. 47, No. 5, 1996, pp. 678–689.

[11] F.F. Boctor, "Some Efficient Multi-heuristic Procedures for Resource Constrained Project Scheduling", European Journal of Operational Research, Vol. 49, 1990, pp. 3–13.

[12] Y. C. Toklu, "Application of Genetic Algorithms to Construction Scheduling with or without Resource Constraints", Canadian Journal of Civil Engineering, Vol. 29, 2002, pp. 421-429.

[13] W. Chan, D. K. H Chua and G. Kannan, "Construction Resource Scheduling with Genetic Algorithms", Journal of Construction Engineering and Management, ASCE, 122(2), 1996, pp.125-132.

[14] S. Leu, and C. Yang, "GA-Based Multicriteria Optimal Model for Construction Scheduling", Journal of Construction Engineering and Management, ASCE, 125(6), 1999, pp. 420-427.

[15] T. Hegazy and M. Kassab, "Resource Optimization Using Combined Simulation and Genetic Algorithms", Journal of Construction Engineering and Management, ASCE, 129(6), 2003, pp. 698-705.

[16] S Hartmann, "A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling", Naval Research Logistics, Vol. 45, 1998, pp. 733-750.

[17] S. Hartmann, "A self-adapting Genetic Algorithm for Project Scheduling under Resource Constraints", Naval Research Logistics, Vol. 49, No.5, 2002, pp. 433-448.

[18] U. Dorndorf and E. Pesch, "Evolution Based Learning in a Job Shop Scheduling Environment", Computers and Operations Research, Vol. 22, 1995, pp. 25-40.

[19] Hong Zhang, Heng Li and C.M. Tam, "Particle Swarm Optimization for Resource-constrained Project Scheduling", International Journal of Project Management, Vol. 24, 2006, pp. 83-92.

[20] P. P. Das and S. Acharyya, "Meta-heuristic Approaches for Solving Resource Constrained Project Scheduling Problem: A Comparative Study", Proc IEEE Int Conf on Computer Science and Automation Engineering (CSAE), Shanghai, China, 2011.

[21] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, "Optimization by Simulated Annealing", Science, Volume 220, Number 4598, 1983, pp.671-680.

[22] A. Hertz and D. Werra, "The Tabu Search Metaheuristic: How We Used It", Annals of Mathematics and Artificial Intelligence, Vol. 1, 1990, pp. 111-121

[23] J. Wei, L. Guangbin and L. Dong, "Elite Particle Swarm Optimization with Mutation", Proc Asia Simulation Conference — 7th Intl. Conf. on Sys. Simulation and Scientific Computing, 2008, pp. 800-803.

[24] D. Costa, "A Tabu Search Algorithm for Computing an Operational Time Table", European Journal of Operational Research, Vol. 76, 1994, pp. 98-110.

[25] A. Moilanen, "Parameterization of a Metapopulation Model: An Empifical Comparison of Several Different Genetic Algorithms, Simulated Annealing and Tabu Search", IEEE Int. Conf. on Evolutionary Computation, 1995, Vol. 2, pp. 551-556.

[26] J. Euchi and H. Chabchoub, "Tabu Search Metaheuristic Embedded in Adaptative Memory Procedure for the Profitable Arc Tour Problem", World Congress on Nature & Biologically Inspired Computing (NaBIC), 2009, pp. 204-209.

[27] P. J. M. V. Laarhoven, E. H. L. Aarts, and J. K. Lenstra, "Job Shop Scheduling by Simulated Annealing". Operations Research, 40(1), 1992, pp. 113-12

[28] A. Hertz, "Finding a Feasible Course Schedule Using Tabu Search", Discrete Applied Mathematics, Vol. 35, 1992, pp. 255-270.

[29] V. Cerny, "A Thermodynamical Approach to the Travelling Salesman Problem", Journal of Optimization Theory and Applications, Vol.15, 1985, pp. 41-51

[30] Z. N. Azimi, "Comparison of Meta-heuristic Algorithms for Examination Timetabling Problem",

J. Appl. Math. & Computing, Vol. 16, No. 1 – 2, 2004, pp. 337 – 354.
[31] Dai Chen, "A Comparative Study of Meta-heuristic Algorithms for the Fertilizer Optimization Problem", Thesis, University of Saskatchewan, Saskatoon, 2006.
[32] F. Glover, "Tabu Search, Part I", ORSA Journal on Computing, Vol. 1, 1989, pp. 190-206.
[33] Dell'Amico, M. and M. Trubian, "Applying Tabu Search to the Job-Shop Scheduling Problem", Annals of Operations Research, Vol. 41, 1993, pp.231-252.
[34] U. Ahmed and G. N. Khan, "Embedded System Partitioning with Flrxible Granularity by Using a Variant of Tabu Search", CCGEI, 2004, pp. 2073-2076.

Partha Pratim Das obtained B.Tech in Information Technology from University of Kalyani, West Bengal, India in 2004 and M.E. in Computer Science and Engineering from West Bengal University of Technology, West Bengal, India in 2010. His research interests include Combinatorial Optimization, Scheduling and Meta-heuristic Search.

Currently, he is an Assistant Professor at The Heritage cademy, Kolkata, West Bengal, India.

Sriyankar Acharyya obtained M. Tech. and Ph.D. in Computer Science and Engineering from Calcutta University. His research interests include Constraint Satisfaction Problems (CSP), Meta-heuristic Search, Combinatorial Optimization, Scheduling and Bio-informatics.

He worked in Bhabha Atomic Research Centre as Scientific Officer (C) from 1989 to 1995. Then, he was Reader in Physics at Vivekananda College and Reader in Computer & System Science at Visva Bharati University. At present he is Associate Professor in Computer Science and Engineering, West Bengal University of Technology (WBUT), West Bengal, India.