An adaptive crossover genetic algorithm for multi mode resource constrained project scheduling with discounted cash flows.

Vijay S Bilolikar (bilolilkar@frcrce.ac.in) Fr C R college of Engg. Mumbai

Karuna Jain (nitie.director@gmail.com) NITIE Mumbai

Abstract

This paper presents an adaptive genetic algorithm for solving a multimode resource-constrained project scheduling problem with discounted cash flows for minimizing costs. The genetic algorithm operates on two crossovers adaptively. A mathematical model is developed and detailed computational experiments are performed on a standard problem set to evaluate the performance.

Keywords: Project Management, Resource constrained Project scheduling, Genetic algorithm.

Introduction

Resource constrained project scheduling (RCPSP) is concerned with scheduling of activities in a project under resource and precedence constraints. RCPSP is traditionally considered an NP hard optimization problem (Blazewicz et al., 1983). In this paper, an adaptive genetic algorithm (AGA) is used to address MRCPSP with discounted cash flows (MRCPSPDCF) with cost minimization as objective. The solution procedure assigns mode and start time to all activities in a project such that the project cost is minimized. The 0–1 integer programming mathematical formulation based on Talbot (1982) is presented for the problem.

Literature review

This problem was introduced by (Mohring 1984) as the resource investment problem and an exact procedure based on graph algorithms is presented. Other exact procedures are proposed by (Demeulemeester 1995) and (Rodrigues and Yamashita 2010). (Demeulemeester 1995) developed an exact algorithm for the RACP. It is based on iterative solutions for the RCPSP which is based on the branch-and-bound algorithm of (Demeulemeester and Herroelen 1992). The algorithm tries to schedule based on the cheapest efficient point by iteratively altering the resources corresponding to deadline. (Rodrigues and Yamashita 2010) modified the algorithm developed by Demeulemeester (1995), by combining heuristic rules for initial solution, so that solution space is reduced. He presented new bounds for the branching scheme that reduced the number of sub problem. Langrangean relaxation and column generation technique for this problem and developed two lower bounds for the resource availability cost problem (RACP) (Drexl and Kimms 2001). RIP with MRCPSP is solved using due date constraints (activity slack) and resource usage (resource investment) and are used to select and schedule tasks (Hsu and Kim 2005). A multi-start heuristic based on the scatter search methodology which searches the solution space using solutions generated by different heuristics is proposed by (Yamashita et al.

2006). A genetic algorithm for solving the resource investment problem with tardiness as penalty is studied by (Shahdrok and Kianfer 2007)

(Ranjbar et al. 2008) introduced the resource trade-off problem with multiple resources. The authors present a new hybrid meta heuristic algorithm based on scatter search and path re-linking methods. In the SS algorithm, they use path re-linking concepts to generate children from parent solutions, in the form of a new combination method. They also incorporate new strategies for diversification and intensification to enhance the search, in the form of local search and forward–backward scheduling, based on so-called reverse schedules, with the activity dependencies reversed. (Peng and Wang 2009) solved multi-mode resource-constrained DTCTP model (MRC-DTCTP) using GA in which activity crashing is used to reduce the cost. The bounds for the activity cost considered. (Van Peteghem and Vanhoucke 2011) presented an invasive weed optimization algorithm.

Model development

The MRCPSPDCF can be formulated as follows. We consider a project consisting of precedence-related set $A = \{1, 2, j\ldots.J\}$ of J activities. We consider additional activities $j = 0$ representing the only source and $j = J + 1$ representing unique sink activity of the network; activities are topologically labelled such that the predecessors of the activity *j* will always be numbered less than the activity number j. We define the set of immediate predecessor and successor activities for an activity j as $\mathcal{P}_\text{j}$ and $\mathcal{S}_\text{j}$ respectively. Precedence relations among activities require that activity *j* cannot be started unless activity $i \in \mathcal{P}_\text{j}$ is not over. All activities except source and sink activity need resources for processing. Activities may use renewable and or non-renewable resources. The horizon of the project is sum of durations of all the activities, with longest mode, in the project.

The set of renewable resources is given by $\mathcal{R}_k^r$ where $k = \{1,2,\ldots.K\ \}$ and the set of non-renewable resources is given by R_l^n where $l = \{1,2,3\ldots L\ \}$. Per-period availability of renewable resource, of type *k*, is given by$\mathcal{R}_k^{ar}$. The total availability of non-renewable resource of type *l* for the entire project duration is given by$\mathcal{R}_l^{an}$.

Depending on the amount of resources consumed, an activity *j* may be processed in more than one way (each way referred to as a mode) and a set of all such modes of execution for an activity *j* is denoted by $M_j=\{1,2,\ldots m_j,\ldots M_j\ \}$. Activity *j*, performed in mode $m \in M_j$, has duration d_{jm}.

Activity performed in mode m requires $\mathcal{R}_{jmk}^r$units of *k*-type renewable resources per unit time and $\mathcal{R}_{jml}^n$units of *l*-type non-renewable resources.

Notations

C_{r_k}	Per period cost of using one unit of renewable resource $\mathcal{R}_k^r$of type *k*.
C_{nr_l}	Per unit cost of non-renewable resource $\mathcal{R}_l^n$of type *l*.
D	The dead line of the project and is taken as two third of the horizon of the project.
x	Overhead cost per day.
γ	The cost of the capital.
Y	Bonus rate per day.
Z	Penalty cost per day.
es_j	The earliest start time of the activity *j*
ls_j	The latest start time of the activity *j*
s_j	The start time of the activity *j*

f_j	The finish time of the activity *j*
c_j^f	Fixed cost associated with activity *j*
C_j^v	Variable cost associated with activity *j*

Considering the time value of money, the present value of future cost of activity execution can be calculated by multiplying it with the discounting factor given by:

Discounting factor = $\frac{1}{(1+\gamma)^{s_j}}$ where s_j is the start time of the activity *j*.

The cost associated with an activity *j* can be stated as:

$$c_j^f \;=\; x * d_{jm}$$
$$C_j^v = \left(\sum_{k=1}^{K}(\mathcal{R}_{jmk}^{r})(d_{jm})(C_{r_k}) + \sum_{l=1}^{L}(\mathcal{R}_{jml}^{n} * C_{nr_l})\right)$$

The discounted value of total cost of execution of activity *j* is given by

$$c_j^T = \frac{c_j^f + c_j^v}{(1+\gamma)^{s_j}}$$

The penalty and bonus arises, depending on the deadline overrun of the project. If f_J is the finish time of the terminal activity of the project, the bonus and penalty is estimated as

- The bonus amount B is given by
  $$B = \frac{(D-f_j)*y}{(1+\gamma)^{f_j}} \;\; if\; D > f_J$$
- The penalty amount is given by
  $$P = \frac{(f_J-D)*Z}{(1+\gamma)^{f_J}} \;\; if\; f_J > D$$

Mathematically, this model can be formulated by defining 0–1 variablesx_{jmt}.

$$minimize\left\{\sum_{j=1}^{J}\sum_{t\,=\,es_j}^{ls_j} c_j^T\, x_{jmt} \;\mp\; B(P)\right\} \tag{1}$$

Subject to:

$$\sum_{m=1}^{M_j}\sum_{t=es_j}^{ls_j} x_{jmt} = 1 \quad j = 1,2,\ldots\ldots,J \tag{2}$$

$$\sum_{m=1}^{M_j}\sum\nolimits_{t=es_j}^{ls_j}(t + d_{jm})x_{jmt} \leq \sum_{m=1}^{M}\sum\nolimits_{t=es_i}^{ls_i} t x_{imt} \qquad j \in\; p_i \tag{3}$$

$$\sum_{j=1}^{J}\sum_{m=1}^{M_j}\mathcal{R}_{jmk}^{r}\;\sum\nolimits_{s=max\{t-d_{jm},es_j\}}^{\min\{t-1,ls_j\}} x_{jmt} \leq\; \mathcal{R}_k^{ar} \quad k = 1,2,\ldots,K; t = 1,2,\ldots.,D. \tag{4}$$

$$\sum_{j=1}^{J}\sum_{m=1}^{M_j}\mathcal{R}_{jml}^{n}\;\sum\nolimits_{s=es_j}^{ls_j} x_{jmt} \leq\; R_l^n \quad l = 1,2,\ldots,L; \tag{5}$$

$$f_J \;\leq D\;; \tag{6}$$

$$x_{jmt} = \{0,1\} \quad j = 1,2,\ldots\ldots,J; m = 1,2,\ldots,M_j; t =\; es_j,\ldots,ls_j. \tag{7}$$

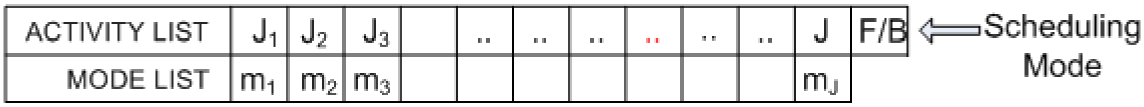
We need to determine the execution mode and its starting time. The decision variable of the problem is as follows:

$$x_{jmt} = \begin{cases} 1, & if\ activity\ j\ is\ executed\ in\ mode\ m\ and\ started\ at\ time\ t \\ 0, & otherwise \end{cases}$$

The objective function (1) is to minimize the cost of the project. It is assumed that the dummy start node and dummy end node can only be processed in a single mode with duration equal to zero. Equations (2) to (5) represent the constraints of the problem. Equation (2) assures that each activity is assigned exactly one mode and exactly one start time. Equation (3) represents the precedence constraints, i.e., the start time of the *j* is always greater than or equal to the finish time of its predecessor activity *i*, which belongs to predecessor set $\mathcal{P}_j$ of *j*. Equation (4) checks the per-period renewable resource violation by the activity that is in progress at time t. Equation (5) represents the constraints on the non-renewable resources. It ensures that total requirement of non-renewable resources by all the activities is less than or equal to the available resources. Equation (6) sets the deadline of the completion of the project. Finally, Equation (7) imposes binary values on the decision variables.

Genetic algorithm

The problem data is pre-processed using data pre-processing procedure suggested by Sprecher et al (1997). The pre-processing procedure removes inefficient, non-executable modes of activities and redundant non-renewable resources from the problem input data. GA works on this processed data. The project information is presented in the form of solution representation and genetic operators work on these representations. These representations are decoded back using decoding procedure i.e. schedule generation schemes (SGS). We used solution representation as shown in the figure 3.



ACTIVITY LIST	J_1	J_2	J_3		..	..	..	..	..	..	J	F/B
MODE LIST	m_1	m_2	m_3								m_J	

Figure 3 Solution representation

Initial population

The initial population is generated using two heuristic rules viz. latest finish time (LFT) and shortest processing time (SPT).The solutions in the population are decoded to schedules using serial schedule generation scheme (SSGS). We used the rank selection strategy for selecting individuals for the crossover operation. In rank selection methods, the individual solutions receive their rank from the fitness values.

Fitness computation

Since the objective is cost minimization, the algorithm is driven by the cost of the schedule. Therefore, the schedule cost is used as a fitness score of every chromosome in the population. By relaxing non-renewable resource constraints, solutions with infeasibility with respect to non-renewable resources are allowed to exist in the population so that high-quality genes from such schedules are captured during the crossover and mutation process. Such infeasible individuals are penalized in fitness function. The penalty is in terms of the cost associated with violation of non-renewable resource. The fitness function for evaluation of fitness score with penalty is given by equation (8). Fitness is computed for each individual depending on whether an individual solution is feasible or infeasible. The penalty cost is not discounted. Thus, the fitness value of a feasible individual is always less than that of infeasible individuals.

$$\boldsymbol{f(x)} = \begin{cases} \{\sum_{j=1}^{J}(TC_j)\} & if\ feasible \\ \{\sum_{j=1}^{J}(TC_j)\} + \sum_{l\in NR}^{L} \max\left\{0, \left(\sum_{j=1}^{J} r_{jml}^{nr} - \mathcal{ANR}_1\right) * c_{r_l^{nr}}\right\} & otherwise \end{cases} \qquad (\mathbf{8})$$

Cross over

Crossover is reproduction process in which two chromosomes combine together to produce two offspring and transmit the genetic characteristics. The crossover design should achieve balance of high mean fitness and diversity of population, which is often difficult with a single crossover operator. Therefore we used two crossover operators viz three point and multi point forward backward crossover operator. These operators operate adaptively on same population. Three point crossover operator introduces diversity while multi point forward backward operator raises average fitness of the population.

Three point crossover:

In three point crossover method, we generate three integer random numbers, say p, q and r, from [1, J] such that 1≤ p < J/3, J/3≤ q < 2J/3 and 2J/3≤ r < J. where J is equal to number of activities in project. They are used as crossover points for three point crossover method. The fractional values are rounded to next higher integer values. The generation of son and daughter is explained in the pseudo algorithm shown in fig 2. Let $J_i^s,\ J_i^d,\ J_x^f, J_y^m$ represent the activity from the son, daughter, father and mother schedule respectively. $\mathrm{m}_{\mathrm{im_k}}^{\mathrm{s}},\ \mathrm{m}_{\mathrm{im_k}}^{\mathrm{d}},\ \mathrm{m}_{\mathrm{xm_k}}^{\mathrm{f}},\ \mathrm{m}_{\mathrm{ym_k}}^{\mathrm{m}}$ represent the modes of the activity belonging to son, daughter, father and mother schedule respectively. The generation of son and daughter is explained in the pseudo-algorithm shown in Fig 2.

$\boldsymbol{Generation\ of\ son}$	$\boldsymbol{Generation\ of\ Daughter}$
	-
$\boldsymbol{for\ i = 1\ \ to\ p\ do}$	$\boldsymbol{for\ i = 1\ \ to\ p\ do}$
$\boldsymbol{J_i^s = J_x^f}$	$\boldsymbol{J_i^d = J_y^m}$
$\mathbf{m_{im_k}^s = m_{xm_k}^f}$	$\mathbf{m_{im_k}^d = m_{ym_k}^m}$
$\boldsymbol{for\ i = p+1\ \ to\ q\ do}$	$\boldsymbol{for\ i = p+1\ \ to\ q\ do}$
$\boldsymbol{y\ = lowest\ index \mid 1\ \le\ y\ \le J\ j_y^m \notin \{j_1^s, \ldots\ldots, j_p^s\ \}}$	$\boldsymbol{x\ = lowest\ index\ \lvert 1\ \le\ x\ \le J\ j_x^f \notin \{j_1^d, \ldots\ldots, j_p^d\ \}}$
$\boldsymbol{J_i^s = J_y^m}$	$\boldsymbol{J_i^d = J_x^f}$
$\mathbf{m_{im_k}^s = m_{ym_k}^m}$	$\mathbf{m_{im_k}^d = m_{xm_k}^f}$
$\boldsymbol{for\ i = q+1\ \ to\ r\ do}$	$\boldsymbol{for\ i = q+1\ \ to\ r\ do}$
$\boldsymbol{x\ = lowest\ index \mid 1\ \le\ x\ \le J\ \ j_x^f \notin \{j_1^s, \ldots\ldots, j_q^s\ \}}$	$\boldsymbol{y\ = lowest\ index \mid 1\ \le\ y\ \le J\ j_y^m \notin \{j_1^d, \ldots\ldots, j_q^d\ \}}$
$\boldsymbol{J_i^s = J_x^f}$	$\boldsymbol{J_i^d = J_y^m}$
$\mathbf{m_{im_k}^s = m_{xm_k}^f}$	$\mathbf{m_{im_k}^s = m_{ym_k}^m}$
$\boldsymbol{for\ i = r+1\ \ to J\ do}$	$\boldsymbol{for\ i = r+1\ \ to J\ do}$
$\boldsymbol{y\ = lowest\ index \mid 1\ \le\ y\ \le J\ j_y^m \notin \{j_1^s, \ldots\ldots, j_r^s\ \}}$	$\boldsymbol{x\ = lowest\ index \mid 1\ \le\ x\ \le J\ j_x^f \notin \{j_1^d, \ldots\ldots, j_r^d\ \}}$
$\boldsymbol{J_i^s = J_y^m}$	$\boldsymbol{J_i^d = J_x^f}$
$\mathbf{m_{im_k}^s = m_{ym_k}^m}$	$\mathbf{m_{im_k}^d = m_{xm_k}^f}$
$\boldsymbol{J_{j+1}^s\ \ = j_{j+1}^f}$	$\boldsymbol{J_{j+1}^d = j_{j+1}^m}$
$\boldsymbol{scheduling\ mode\ of\ the\ son\ is}$ $\boldsymbol{copied\ from\ father}$	$\boldsymbol{scheduling\ mode\ of\ the\ daughter\ is}$ $\boldsymbol{copied\ from\ mother}$

Figure 1Pseudo code for Three point crossover

Multi point forward-backward crossover

The generation of son and daughter is explained in the pseudo-algorithm shown in Fig 4. The procedure ensures precedence relationship among the activities. Fig 5 describes the procedure for generation of son and daughter for the example network shown in Fig 1.

$\boldsymbol{Generation\ of\ son}$	$\boldsymbol{Generation\ of\ daughter}$
$\boldsymbol{for\ i = 1\ \ to\ n\ do}$ $J_i^s = J_x^f$ $\mathbf{m}_{\mathrm{im_k}}^{\mathrm{s}} = \mathbf{m}_{\mathrm{xm_k}}^{\mathrm{f}}$ *such that* $\mathrm{x} = \boldsymbol{lowest\ index}\ \|1\ \le\ x\ \le J\ j_x^f \notin \{j_1^s, \ldots, j_n^s\ \}$ $y = (n - (i - 1))$ $J_{n-(i-1)}^s = J_y^m$ $\mathbf{m}_{(n-(i-1))\mathrm{m_k}}^{\mathrm{s}} = \mathbf{m}_{y\mathrm{m_k}}^{\mathrm{m}}$ *such that* $\mathrm{y} = \boldsymbol{highest\ index}\ \|1 \le y \le J\ j_y^m \notin\ \{j_1^s,, j_n^s\ \}$ $J_{j+1}^s = j_{j+1}^f$ $\boldsymbol{scheduling\ mode\ of\ the\ son\ is\ copied\ from}$ $\boldsymbol{father}$	$\boldsymbol{for\ i = 1\ \ to\ n\ do}$ $J_i^d = J_y^m$ $\mathbf{m}_{\mathrm{im_k}}^{\mathrm{d}} = \mathbf{m}_{\mathrm{ym_k}}^{\mathrm{m}}$ *such that* $y\ = \boldsymbol{lowest\ index}\|\ 1 \le y \le J\ j_y^m \notin\ \{j_1^d, .., j_n^d\ \}$ $x = (n - (i - 1))$ $J_{n-(i-1)}^d = J_x^f$ $\mathbf{m}_{(n-(i-1))\mathrm{m_k}}^{\mathrm{d}} = \mathbf{m}_{x\mathrm{m_k}}^{\mathrm{f}}$ *such that* $x = \boldsymbol{highest\ index}\|1 \le x \le J\ \ j_x^f \notin \{j_1^d,, j_n^d\ \}$ $J_{j+1}^d = j_{j+1}^m$ $\boldsymbol{scheduling\ mode\ of\ the\ daughter\ is\ copied\ from}$ $\boldsymbol{mother}$

Figure **Error! No text of specified style in document.**Pseudo code for Multi point forward backward crossover

For the RCPSP the cross over probability is in the range 0.7 to 0.9 depending on other genetic operators. Present study considered the fixed crossover probability as 0.9.

Mutation

We introduced entirely new genetic material in the population by generating new solutions, called mutant, using convex combinations of priority rules viz minimum latest finish time (LFT) and minimum slack (SLK). The scheduling mode gene is randomly assigned to mutant solutions. The number of mutant solutions is kept about three to five percent of population. These mutant solutions will replace the weakest solutions in the population.

Termination criteria

The benchmark for the problems on cost minimization is not available in the literature and they are tested on different benchmark datasets using different stop criteria, a fair comparison between each of these procedures is difficult (Van Peteghem and Vanhoucke (2013)). Therefore, we used termination criteria as 50,000 schedules for each problem. We calculate the cost associated with a schedule based on a critical path method (CPM) calculation with non-critical path activities, scheduled as late as possible. The performance evaluation criterion compares the percentage deviation in cost of a solution from the cost of CPM path based solution of the problem.

$$\%\text{ Average deviation in cost} = \frac{(cost\ \ of\ the\ CPM\ path\ based\ solution - cost\ of\ solution\ )}{(cost\ of\ the\ CPM\ path\ based\ solution)} * 100$$

Computational experiments

The experiments detailed herein were performed on Intel Pentium desktop machine with frequency of 2.60 GHz and 512 MB RAM. The GA was coded in C++, compiled with Microsoft Visual C++ v.6.0 compiler and tested in Linux. We used a set of standard test problems available at www.psplib.com. We have two renewable and two non-renewable resources.

For each instance from the PSPLIB library, we generated cost figures for the renewable and non-renewable resources from the interval (0; 1000] with uniform distribution. For renewable resources, we calculated per-period cost, and for non-renewable resources, we calculated per-unit cost such that $C_{r_1} < C_{r_2} < C_{nr_1} < C_{nr_2}$. The cost of the capital employed 'γ' is taken as 0.05 % per period. Through exhaustive simulation on J10 dataset, optimum GA configuration was determined for numerical investigation. The computational investigation revealed that the algorithm performance was superior with α value as 0.75, initial population size as 200. All other the parameters being held constant, the performance of the algorithm was superior at crossover and mutation probabilities as 0.7 and 0.03 respectively.

Results and discussion

We tested the performance of the algorithm for standard datasets available at http://www.om-db.wi.tum.de/psplib/main.html (i.e., *j*10, *j*12, *j*14, *j*16, *j*18, *j*20, *j*30).

Table 1 lists the number of instances in a dataset for which feasible solutions could be found using the algorithm, with cost minimization as the objective. The results show that the algorithm provided feasible solutions for almost all the problem instances across datasets.

Table 1. Number of feasible solutions found by the algorithm

Data Set	Number of Instances	Number of instances with feasible solutions found
J10	536	536
J12	547	545
J14	551	545
J16	550	541
J18	552	544
J20	554	542
J30	552	524

The algorithm failed to find feasible solutions in some instances, especially for projects with a high number of activities. This suggests that adaptation of the algorithm helped in directing the search trajectory toward the feasible regions of the multi-modal solution space of MRCPSP.

The experiments were carried out to study the performance of the algorithm. The performance is evaluated by comparing deviations between the cost objective function of the schedule and the CPM based schedule cost objective function. Higher deviations indicate that lower cost is achieved. The average and standard deviations in cost based on the proposed method for all the datasets are given in Table 2.

Results of experimentation, as given in Table 2, show that the algorithm yielded better solutions (lower cost) than the CPM-based schedule. Similarly the standard deviations are very low for the algorithm. This indicates that the algorithm was able to come out of the local optima and reached closer to the optimal solutions. Results show that the average deviations are higher and become significantly higher as the project size grows. Similarly standard deviations are lower as project size grows. An increase in project size also increases the complexity of the problem. Therefore, the proposed algorithm can help in solving scheduling programs for bigger projects, which have more complexity

Table.2 The deviations of COST values

Data Set	Average deviation	Standard deviation
J10	7.072	13.00
J12	7.435	9.49
J14	6.623	10.47
J16	6.720	10.88
J18	5.856	10.76
J20	5.521	10.12
J30	3.615	8.87

.

Conclusion and future research direction

In this paper, we discussed the MRCPSP with discounted cash flows. Our goal was to minimize the cost of the project. We presented an integer 0-1 programming formulation for the problem and considered only negative cash flows. The performance of the algorithm was compared with CPM path-based solutions. The costs associated with the resources were generated randomly with uniform distribution. The results of the computational experiment confirm that the algorithm performed well. In the future, we plan to explore different payment models and develop a branch-and-bound procedure for the accurate evaluation of the proposed algorithm.

References

1. A. Sprecher, S. Hartmann and A. Drexl, “An exact algorithm for project scheduling with multiple modes”, OR Spektrum 19, 195–203, 1997.
2. A. Drexl, A. Kimms, “Optimization guided lower and upper bounds for the resource investment problem”. Journal of Operations Research Society 52:340–351, 2001.
3. C-C Hsu, D Kim, “A new heuristic for the multi-mode resource investment problem”. Journal of Operations Research Society 56:406–41, 2005.3
4. D Yamashita, V Armentano, M Laguna, “Scatter search for project scheduling with resource availability cost”. European Journal of Operations Research 169:623–637, 2006.
5. Erik Demeulemeester., “Minimizing resource availability costs in time-limited project networks”. Management Science 10(41):1590–1598, 1995.
6. E. Demeulemeester, W. Herroelen., “A branch-and-bound procedure for the multiple resource- constrained project scheduling problem”. Management Science 38:1803–1818, 1992.
7. F. B.Talbot, “Resource-constrained project scheduling with time-resource trade-offs: the nonpreemptive case”. Management Science 28 (10), 1197–1210. 1982.

8. S. Hartmann, A. Drexl, “Project scheduling with multiple modes: a comparison of exact algorithms”. Networks 32 (4), 283–297. 1998.
9. Since Hartmann,.” Project scheduling with multiple modes: a genetic algorithm”. Annals of Operations Research 102 (1–4), 111–135.2001
10. J. Blazewicz,, J.K.Lenstra, , A.H.G. Rinnooy Kan, “Scheduling projects subject to resource constraints: Classification and complexity”. Discrete Applied Mathematics 5, 11-24. 1983.
11. M Ranjbar, F Kianfar, S Shadrokh, “Solving the resource availability cost problem in project scheduling by path re-linking and genetic algorithm”. Applied Mathematics and Computation 196:879–888, 2008.
12. N. Metropolis, A.W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller, “Equation of State Calculation by Fast Computing Machines”. Journal of Chem. Phys., 21, 1087-1091. 1953
13. R. Kolisch,, A. Sprecher, 1997. "PSPLIB – A project scheduling problem library”. European Journal of Operational Research 96 (1), 205–216.
14. R Kolisch, S. Hartmann, “Experimental investigation of heuristics for resource-constrained project scheduling: an update”. European Journal of Operations Research 174:23–37, 2006.
15. R Mohring, “Minimizing costs of resource requirements in project networks subject to a fixed completion time”. Operation Research 32(1):89–120, 1984.
16. S. Rodrigues, D Yamashita, “An exact algorithm for minimizing resource availability costs in project scheduling”. European Journal of Operations Research 206:562–568, 2010.
17. S Shadrokh, F Kianfar, “A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty”. European Journal of Operations Research 181:86–101, 2007.
18. S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, “Optimization by Simulated Annealing”. Science, vol 220, No. 4598, 671-680, 1983
19. V Van Peteghem, M Vanhoucke, “An artificial immune system for the multi-mode resource-constrained project scheduling problems”. Flex Serv Manuf Journal 25:122–144, 2013
20. W. L. Peng, C.G. Wang, “A multi-mode resource-constrained discrete time-cost trade-off problem and its genetic algorithm based solution”. International Journal of Project Management 27 (6), 600–609, 2008.