A new bi-objective evolutionary algorithm using clustering heuristics to solve the Multi-mode Resource-Constrained Project Scheduling Problem

Sonda Elloumi (a), Philippe Fortemps (b), Jacques Teghem (b), Taïcir Loukil (a)

(a) GIAD, University of Economic and Management Sciences, Rue de l'aerodrome Km 4 Sfax, Tunisia
(b) MATHRO, Faculté Polytechnique de Mons, 9 rue de Houdain, B-7000, Mons, Belgium

Abstract

In this paper, the non preemptive multi-mode resource-constrained project scheduling is considered. We use a bi-objective approach to solve the mono-objective problem of makespan minimization. Our main focus is to propose a high-performance algorithm that solves the problem at hand. A new evolutionary algorithm is developed. The encoding procedure is based on feasible activity list and a mode assignment. We define two new fitness functions dealing with clustering tools, namely k-means and hierarchical clustering algorithms. Results of the implementation are emphasized to decide on the best algorithm configuration. The latter is used to compare our algorithm performance to other recent heuristics and metaheuristics. Standard instances sets included in PSPLIB are tested. The results testify the good performance of our new hybrid evolutionary algorithm.

Introduction

The Multi-mode Resource-Constrained Project Scheduling Problem (MRCPSP) is one of the most studied generalizations of the well known Resource-Constrained Project Scheduling Problem (RCPSP). It is closer to real problems since each activity may be performed in one out of several modes and it deals with more than one kind of resources.

Noticeably, the MRCPSP is more complex than the RCPSP, which is itself NP-hard. Sprecher and Drexl (1998) prove that highly resource-constrained projects with at least 20 activities and 3 modes per activity can not be solved by exact optimization procedures within a reasonable computation time. Furthermore, Kolisch and Drexl (1997) demonstrate that, if at least 2 nonrenewable resources are taken, even the problem of whether finding a feasible solution for the MRCPSP is NP-complete.

Consequently, one can deduce how crucial the resort to heuristic and metaheuristic algorithms is. These methods provide near-optimal solutions for large-sized projects within relatively little time. The literature about the MRCPSP is exceptionally rich with these algorithms.

According to Boctor (1996a) and (1996b), the pioneer studies of the MRCPSP were carried out by Elmaghraby (1977), Slowinski and Weglarz (1978) and Weglarz (1980). These authors studied the preemptive case. Then, Talbot (1982) tackled the nonpreemptive case to give a mathematical formulation (on which many post-studies are based, like Sprecher (1994), Sprecher et al. (1997) and Hartmann (2001), etc.). This formulation enables one to handle renewable, nonrenewable and doubly constrained resources. It also permits either the minimization of the project duration or cost, or the maximization of its net present value.

Talbot (1982) proposes a deterministic enumeration scheme. The method provides optimal solutions for small-sized problems, and is able to find heuristic solutions for large-sized ones. Other exact procedures have been proposed, we quote Hartmann and Drexl (1998), Sprecher and Drexl (1998) and Sprecher et al. (1997). These are different variants of Branch-and-Bound method.

Heuristic methods have been conceived to solve the MRCPSP. Talbot (1982) proposes his exact procedure to heuristically solve large-sized problems. Boctor (1993) presented a comparison of 21 heuristic scheduling rules. The same author (1996a) proposes a new efficient heuristic algorithm. The latter is a particular heuristic because it allows scheduling more than one activity at a time. Bianco et al. (1998) treat the MRCPSP, but they restrict the non renewable resource constraints to the unique budgetary constraint. Drexl and Gruenewald (1993) propose a stochastic scheduling method.

Several metaheuristic procedures have been devoted in the literature. Józefowska et al. (2001) present two versions of simulated annealing approach: with and without penalty function. Likewise, Bouleimen and Lecocq (2003) conceive a new simulated annealing algorithm for the RCPSP and its multi-mode version. Özdamar (1999) conceive a new hybrid genetic algorithm for the MRCPSP; the main particularity of the algorithm appears in the solutions encoding by the mode assignment and the priority rules. Hartmann (2001) proposes a genetic algorithm which is, in the best of our knowledge, the most efficient to solve the MRCPSP; the algorithm is especially rich with two local search procedures already proposed by Sprecher et al. (1997). Alcaraz et al. (2003) and Elloumi et al. (2006) apply a similar framework; they tried to overcome Hartmann's GA drawbacks and to improve it.

Analogously, in this paper we propose a new Evolutionary Algorithm (EA) approach dealing with MRCPSP where the objective is the minimization of the project duration. Tasks can not be interrupted once begun, and various kinds of

resources are considered. Our main focus is to propose a high-performance algorithm that solves the outlined problem.
The remainder of this paper is organized as follows: the first section is devoted to the presentation of the problem. The second section deals with the new EA steps and its features. The third section is oriented towards the analysis of the algorithm computational results. Hence, we present the experimental framework. Then, we determine the best configuration of our proposed Evolutionary Algorithm. In the end of this section, we exhibit results drawn from the comparison of our Evolutionary Algorithm with other metaheuristics proposed in the literature of MRCPSP.
In the remaining, we mean by MRCPSP the non-preemptive Multi-mode Resource-Constrained Project Scheduling Problem with the objective of minimizing the project duration, unless contrarily mentioned.

Problem description

We consider a project consisting of J activities labeled j = 1…J. These jobs are partially ordered, i.e. there are precedence relations between some of them. The precedence relations are expressed by sets Pj of immediate predecessors of job j. We assume that the activities are numerically labeled, that is an activity j has always a higher number than all its predecessors.
For simplification sake, the precedence relations can be depicted by an acyclic activity-on-node network. We set additional activities j = 0 and j = J + 1 which represent respectively unique dummy source and sink nodes.
Three most used categories of resources are considered:

- Renewable resources (whose set is referred to by $\mathcal{K}^{\rho}$): in our study, we are interested in constant per-period availability. This availability of a renewable resource k is denoted by $\mathcal{R}_k^{\rho}$;
- Nonrenewable resources (whose set is referred to by $\mathcal{K}^{\upsilon}$): for each resource $k \in \mathcal{K}^{\upsilon}$, this availability is designated by $\mathcal{R}_k^{\upsilon}$;
- Doubly constrained resources: recall (as mentioned in the first section) that we deal with this type of resources by representing it in both kinds of constraints: the renewable and nonrenewable resources.

Each activity may be executed in one out of several modes. A mode is a scenario of performing a job. It reflects, for the activity in question, first the consumption on each resource k belonging to either category of resources, and second the related duration. Once started, an activity may not be interrupted, and its mode may not be altered. For each activity $j = 1…J$, we distinguish a set of different modes; the latter is denoted by $\mathcal{M}_j = \{1,..., M_j\}$. The job j performed on a mode $m \in \mathcal{M}_j$ has a processing time referred to by p_{jm}, it requires r^{ρ}_{jmk} units of each renewable resource $k \in \mathcal{K}^{\rho}$ during its process, and r^{υ}_{jmk} units of each nonrenewable resource $k \in \mathcal{K}^{\upsilon}$.
The dummy activities represent a special case, in the sense that each of them is supposed to have a unique mode characterized by no request on any resource, and duration equal to zero. Let us set T an upper bound for the project makespan. T may be obtained for example by adding the maximum durations of all the activities.
The objective of this study is to find a mode and a finish time for each activity so that the schedule is feasible and the makespan of the project is minimal with respect to the precedence and resource constraints.

Proposed Evolutionary Algorithm

In this section, we propose a new evolutionary algorithm to solve the MRCPS problem. To personalize an EA, one or more components may be altered. The variation may deal with (a) the solution encoding: different kinds of representation are available in the literature of project scheduling; a brief survey is given by Alcaraz and Maroto (2001); (b) operators: there is a large spectrum of crossover, mutation and selection operators in the literature so choosing one or other alternatives from the range of these operators leads to a special EA; (c) Evaluation of the chromosomes: this component reflects the ability of chromosomes to survive and contribute offspring in the next generation; to measure this ability, a fitness function must be conceived for each chromosome. Besides, the introduction of new components leads to a personalized (even hybrid) EA.
In the following, we define the proposed algorithm stages: encoding, crossover, mutation and selection in addition to further proposed procedures particularizing the proposed EA.

Basic scheme

Generate initial population consisting of *POP* chromosomes;
Apply a local search procedure;
Initialize the current generation number to 0;
WHILE generation number < GEN **AND** run-time < CPU time limit **DO**
BEGIN
Increment the generation number;
Select *POP* individuals for crossover;
Produce *POP* children ($\mathcal{CHI}$) from pairs of the selected individuals by **crossover**;
Apply mutation to $\mathcal{CHI}$ with a probability $pmut$;
Compute fitness values;
Reproduce population := selected individuals from the last population $\cup$ $\mathcal{CHI}$.
END

Figure 1: Proposed EA

First of all, a preprocessing procedure is applied in project data to reduce the search space efficiently. After that, the EA starts with generating an initial population by means of a suitable encoding; the number of individuals in a population is constant through generations, say *POP*; we

assume *POP* to be an even integer. Then, for each individual the makespan and a penalty values are computed; a local search procedure is applied to ameliorate the generated solution by trying to reduce penalties. In our case, the constraints to be respected are the renewable resource and precedence constraints only. The violation of nonrenewable resource constraints is penalized by a certain measure that will be defined later. *POP*/2 pairs of individuals are randomly selected. Resulting children undergo mutation with a probability p_{mut}. Once again, the makespan and a penalty values are computed for the newly produced individuals and the fitness values are determined. Here intervenes the reproduction operator to maintain *POP* individuals ready to contribute in the next generation. This procedure is repeated for a determined number of generations, say GEN, or until the CPU time limit is reached. Figure 1 depicts the proposed EA scheme.

Preprocessing

This step is introduced before the beginning of the EA in order to reduce the search space. The reduction procedure was first developed by Sprecher et al. (1997) to accelerate their branch-and-bound algorithm for the MRCPSP. The idea behind this procedure is to exclude modes and/or renewable as well as nonrenewable resources from the input data. Because of the interdependency between the elimination of these contents, the latter are dealt with as follows:

Step 1: Remove all non-executable modes from the project data.

Step 2: Delete the redundant nonrenewable resources.

Step 3: Eliminate all inefficient modes.

Step 4: if any mode has been erased within Step 3, go to Step 2.

For more details of the components interdependency and the merit of the procedure, we refer the reader to Sprecher et al. (1997) and Hartmann (2001).

Definition of individuals

The encoding chosen for individuals is activity list representation. Since we are dealing with the multi-mode version of the standard RCPSP, individuals must reveal at the same time activities order and mode assignment; hence, the finish times of activities and, therefore, a makespan of the project can be deduced. Obviously, activities finish times depend at the same time on the two above-mentioned individual elements. So, our EA deals simultaneously with both sequencing and mode assignment problems.

The representation opted for is a pair $I = (\lambda, m)$ of two vectors. The first vector $\lambda = (j_1,\ldots,j_J)$ denotes a precedence feasible activity list. The second vector m symbolizes the modes relating to the corresponding jobs in the list. Accordingly, the notation which will be used is the following:

$$I = \begin{pmatrix} j_1 \cdots j_J \\ m(j_1) \ldots m(j_J) \end{pmatrix}.$$

Individuals are obtained by fixing the activities modes randomly from the remaining set of modes after the preprocessing procedure. Then, the activity list is constructed as if we deal with a traditional single mode RCPSP with respect to the resulting mode assignment; that is, we start with the source dummy activity at time 0, and we successively repeat the following: we consider the set of eligible jobs and, after that, we randomly allocate an activity from the set at its earliest start time regarding precedence and renewable resources constraints.

Observe that the ensuing schedule is precedence and renewable resource feasible, but it is not necessarily nonrenewable resource feasible. Recall that, as proved by Kolisch and Drexl (1997), "already finding a feasible schedule is an NP-complete problem if at least two nonrenewable resources are given". So, to overcome this difficulty, we find it practical to introduce schedules which violate the nonrenewable resource constraints into the search space.

Fitness computation

Recall that Kolisch and Drexl (1997) demonstrate that, if at least 2 nonrenewable resources are taken, even the problem of whether finding a feasible solution for the MRCPSP is NP-complete. So, in our algorithm, we allow the introduction of nonrenewable resource infeasible individuals. Note that all individuals are necessarily precedence and renewable resource feasible.

Although we are dealing with a single performance measure, our fitness function behaves by considering an additional performance measure to be minimized. The latter takes its favor from introducing solutions which are infeasible with regard to nonrenewable resource constraints; it consists of a penalty value attributed to these solutions. The fitness of a given individual $I = (\lambda, m)$ is computed in the following manner:

We set T an upper bound for the project makespan. We choose T the sum of the maximum processing times of the jobs. Formally,

$$T = \sum_{j=1}^{J} \left(\underset{m(j)\in\mathcal{M}_j}{Max}\, p_{jm(j)} \right);$$

Like Hartmann (2001), let us assign for each nonrenewable resource $k \in \mathcal{K}^{\upsilon}$ a penalty measure:

$$L_k(m) = R_k^{\upsilon} - \sum_{j=1}^{J} r_{jm(j)k}^{\upsilon};$$

$L_k(m)$ designates the leftover capacity of the nonrenewable resource $k \in \mathcal{K}^{\upsilon}$ relating to the mode assignment m. A negative value of $L_k(m)$ implies that the availability of the nonrenewable resource k is exceeded and, consequently, the mode assignment m is infeasible with respect to the nonrenewable resource on hand. Then, we propose an aggregation measure L(m) given by:

$$L(m) = \sum_{\substack{k \in K^{\upsilon} \\ L_k(m) < 0}} \left\lfloor \frac{|L_k(m)| \times T}{R_k^{\upsilon}} \right\rfloor ;$$

That is, L(m) considers the negative $L_k(m)$; in other words, it takes into account only nonrenewable resources whose total requirement exceeds the capacity. L(m) is the sum of the absolute values of adjusted negative $L_k(m)$. These are adjusted by dividing each of them by the related resource availability, the purpose being to remove the unit effect; then, the result is multiplied by T to convert all into time measure.

Clearly, L(m) > 0 if there is any excess of an availability of nonrenewable resource;
= 0 otherwise.

The penalty is considered as a second criterion to be minimized in addition to the makespan. Hence, the problem becomes bi-objective. To solve it, we consider at each generation the whole set of parents and children distributed regarding the makespan and the penalty value criteria. Then, to compute the fitness value, we firstly adopt the rank-based fitness assignment method for Multi Objective Genetic Algorithms (MOGAs) conceived by Fonseca and Fleming (1993); secondly, we propose different density computation methods borrowed from well known clustering heuristic procedures.

Rank-based fitness assignment method. The method consists in assigning a rank value to each individual depending on its position within the population, and considering the criteria to be optimized. A given individual is better ranked as it is little dominated.

Suppose that, at a given generation t, an individual I is dominated by a number p(t) of individuals in the considered population. Its rank will be determined as follows:

$$\text{Rank}(I, t) = 1 + p(t).$$

Remark that all non dominated solutions have a rank equal to 1. Note also that, for a given individual, this metric may varies through generations because of the population distribution changes.

Clustering heuristics for density computation. To avoid a premature convergence of the algorithm, we introduce density evaluation into the fitness computation. The metric will penalize individuals grouping within the population. To identify the potential groups of neighbor solutions, we propose using k-means or hierarchical clustering methods.

- K-means algorithm: The algorithm was first conceived by MacQueen (1967). With a predefined fixed number of clusters, say K. in our case we choose K equal to the worst rank value (the largest), the algorithm comes down to the next steps:
  Step 1: From the population, choose randomly K points that represent the "centroids".
  Step 2: Assign individuals to the cluster that have the closest centroid. With this intension, we choose to apply the Euclidian distance.
  Step 3: After assigning all individuals, recalculate the positions of the new centroids by computing, for each criterion, the average value of individuals within the same cluster.
  Step 4: repeat Steps 2 and 3 until the centroids become steady or after performing a maximum number of iterations.
  Finally, the density D of an individual amounts to the total number of individuals within its cluster.
- Hierarchical method: In the hierarchical clustering, a limit distance, say dlimit, is fixed instead of deciding the number of clusters. Here, we choose to apply the Manhattan distance. dlimit can be the mean value of the distance matrix, it can be also the median.
  The hierarchical clustering algorithm is performed as follows:
  Step 1: identify the minimum value, say dmin, within the matrix distance, and join the two corresponding individuals into the same cluster.
  Step 2: compute the centroid of the new cluster.
  Step 3: Update the distance matrix by introducing the new centroid considered as a new individual and removing the two old ones.
  Step 4: Repeat steps 1-3 until dmin $\geq$ dlimit.
  Similarly, the density D of an individual is the total number of individuals within its cluster.

Fitness function. A fitness function of an individual I is given by:

$f(I,t) = \dfrac{1}{\text{rank}(I, t) \times D(I, t)}$ when the density is computed

and,

$f(I,t) = \dfrac{1}{\text{rank}(I, t)}$ when the density is not computed, which comes to fix the density to 1 for all individuals within generations.

Note that the fittest individual that may occur is a not dominated solution which is alone within its cluster. Consequently, this individual would have a rank and a density equals to one. Therefore, the fitness value will be, in its turn, equal to one. Other individuals who have greater rank and/or have a higher density would be assigned a fitness value less than one.

Initial population

The initial generation is obtained by repeating the next steps POP times.

Firstly, a mode assignment is generated by randomly selecting $m(j) \in Mj$ for activities j = 1…J.

Secondly, the resulting mode assignment is checked for nonrenewable resource feasibility. If L(m) > 0, a local search procedure is applied trying to improve the current mode assignment. The process consists in randomly selecting a job which has more than one mode alternative, and its current mode is changed by randomly selecting $m'(j) \in Mj \setminus \{m(j)\}$. The result is a new mode assignment m'. If there is improvement, that is if L(m') < L(m), then

m':=m, namely m(j) is replaced by m'(j). This process is repeated until J consecutive unsuccessful trials to improve the mode assignment have been made or, in the best case, until L(m)=0 that is until the individuals become nonrenewable resource feasible.
Thirdly, we adopt the mode assignment and construct a precedence feasible schedule by randomly choosing activities from the eligible set at each stage.
Finally, activities finish times are computed and the makespan of the project is derived from them.

Crossover

In this phase, we repeat successively the following selection until we get POP/2 pairs of individuals ready for crossover. We randomly select two individuals from the current population. Individuals should not be selected twice within one generation. For our algorithm, we opt for the one point crossover for both components of the chromosome: the activity list and the mode assignment.
Let us consider the adapted one point crossover approach used by Hartmann (2001). We select two parents $I^{\mathcal{M}}$ and $I^{\mathcal{F}}$ embodied as follow:

$$I^{\mathcal{M}} = (\ j_1^M \ \ldots\ldots\ j_J^M\ ) \text{ and } I^{\mathcal{F}} = (\ j_1^F \ \ldots\ldots\ j_J^F\ ).$$

A random number q_{cross1} is generated randomly, such that $1 < q_{cross1} < J$.
A daughter $I^{\mathcal{D}}$ and a son $I^{\mathcal{S}}$ are produced. $I^{\mathcal{D}}$ is defined as follows: the q_{cross1} first jobs are taken from the mother; so, for the positions i = 1,…, q_{cross1} we set $j_i^D := j_i^M$.
For the remained positions (i.e. i = q_{cross1} +1,…,J), activities are taken from the father from the set of activities not already scheduled. That is

$j_i^D := j_k^F$, where k is the lowest index such that $j_k^F \notin \{\ j_h^D\ / \text{h} = 1\ldots\text{i-1}\}$.

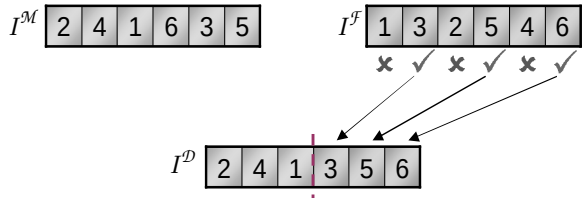

Figure 2: One-point crossover for activity list representation, producing the daughter; q_{cross1} =3

Figure 2 illustrates the operation by an example; Set q_{cross1} =3. The first three activities of the daughter are taken from the mother; the remaining is taken from the father: we begin by the first job (1); it is already scheduled. The second job (3) is not yet scheduled, so we put it, etc.
We proceed similarly for the son by replacing the mother by the father and the father by the mother.
After applying the operation on the activity list, q_{cross2} is drown randomly such that $1 \leq q_{cross2} \leq$ J-1; suppose q_{cross2} = 4. Figure 3 illustrates the process.
The first q_{cross2} modes in the daughter are taken from the mother; however the remained modes are taken from the father. Similarly, the son takes the q_{cross2} first modes from the father and the remaining from the mother. More explicitly, modes of activities in the daughter I^D are defined as follows:

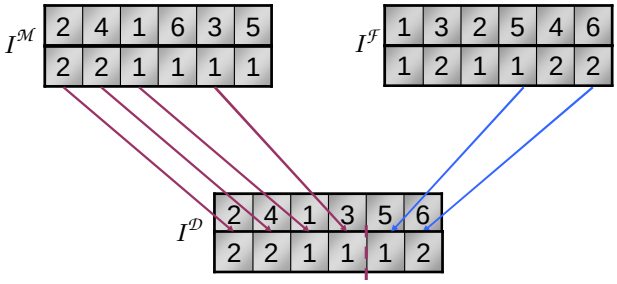

Figure 3: One-point crossover applied on mode assignments, producing the daughter mode. q_{cross2} =4

$m^D(j_i^D) := m^M(j_i^D)$ i = 1… q_{cross2};

$m^D(j_i^D) := m^F(j_i^D)$ i = q_{cross2}+1…J.

In the example above, the first 4 activities in daughter I^D are defined from the mother, even if the fourth job (job 3) is taken from the father. The fifth and sixth jobs have their modes from the father.

Mutation

The mutation operator is applied on newly generated individuals with a probability of mutation p_{mut}. This operator is applied, first, on activity list string and, second, on the mode assignment one.

In the first sub-stage, we choose a position q_{mut} such that $1 \leq q_{mut} \leq$ J-1; then we check whether jobs in the positions q_{mut} and q_{mut}+1 can be permuted; that is if the job in position q_{mut}+1 is not an immediate successor of the job in position q_{mut}, we can permute the jobs; otherwise, we choose another position q'_{mut}. We repeat the procedure until two jobs are permuted or until J unsuccessful attempts are made. Note that at this stage, the mode assignment is not affected; that is permuted jobs keep their initially assigned modes.

Afterwards, we randomly select a job which has more than one mode alternative. From the set of modes $\mathcal{M}_j$, we randomly assign a mode m'(j) $\neq$ m(j).

Selection

Now, the selection operator intervenes. We apply the ranking and the roulette wheel methods. The processes consist in considering both the current and the newly generated populations. Then, POP individuals are selected. Note that with roulette wheel method, an individual can be chosen more than once.

Left shift procedure

The multi-mode left shift is introduced by Sprecher et al. (1997) in their exact algorithm. It is also applied successfully in Hartmann (2001) genetic algorithm. The

heuristic considers at the same time modes and start times of activities. It is applied on feasible individuals only. Each job is considered lone. Then, without disturbing the remainder of the schedule, the change of the job's mode as well as its start time are checked simultaneously. The new job position must evidently respect the precedence relations as well as the per period renewable resource consumption. The purpose is to obtain an improved makespan feasible schedule. Hence, each mode is checked in such a way that the new schedule mode assignment remains nonrenewable resource feasible. Then, feasible positions are checked. The first makespan improving feasible alternative is applied, and the procedure is repeated with next jobs on the resultant schedule.

Computational results

Test problems

In this section, we present and analyze experimental results, the outcome of carrying out the EA presented in the last chapter. The experiments have been performed on a Pentium4-based IBM-compatible personal computer with 3.00 GHz clock-pulse and 512 Mo RAM. The GA has been compiled in with the Microsoft Visual C++ 6.0 compiler and tested under Windows XP professional.

We ran the GA program on standard test instances developed by Kolisch et al. (1995), by means of the project generator ProGen. These problems as well as their best found solutions are available in the project scheduling problem library PSPLIB; for more details, we refer the reader to the article of Kolisch and Sprecher (1996).

In our study, we have used MRCPSP instances sets; they contain instances with 10, 12, 14, 16, 18, 20 and 30 non-dummy activities, named respectively J10, J12, J14, J16, J18, J20 and J30. J10-20 are sets entirely solved to optimality, whereas no optimal solutions are found to J30 instances. The performance of our algorithm is essentially measured in terms of average and maximum percentage deviation from the optimum, as well as the number of instances the optimal solution is found.

Configuration of the algorithm

This subsection reports the best configuration of our algorithm through a numerical investigation. The purpose is to decide on the procedures adopted and the best numerical pattern. Following Hartmann (2001), we choose to make this investigation on J20 instances since the results of smaller projects are not very divergent.

Impact of the preprocessing procedure. These steps are already applied by Hartmann (2001). The experiments he conducts on the same instances set show that, with preprocessing, 4.4% of the modes are removed. Besides, 29 % of the instances have their both nonrenewable resources redundant. Moreover, our experiments show that the procedure is not time consuming.

Impact of the local search procedure. Next, the first local search procedure improving the mode assignment succeeds in considerably improving the number of feasible individuals within the initial population. Indeed, without this procedure only 53% of the individuals are nonrenewable resource feasible, whereas with local search this portion passes to 94%. Besides, the procedure improves, in average, penalties of 99.5% of unfeasible individuals within the population. Hence, we decide to apply the local search on nonrenewable resource unfeasible mode assignments.

Impact of the left shift procedure. The procedure is tested on the algorithm with both preprocessing and local search procedures, but without density computation. That is the fitness function is based on the rank values only. Since it consists in computing new individuals, the procedure requires a considerable additional computation time. The population size is also a determinant and time consuming criterion. Hence, we experiment, like Hartmann (2001), simultaneously the impacts of the procedure and the population size.

	Population size			
	30	60	90	120
Without left shift	1.83%	1.24%	1.02%	0.88%
With left shift	1.57%	1.25%	1.16%	1.18%

Table 1: Impact of left shift schedules improvement. 1 second, J=20.

Table 1 summarizes the results. Visibly, the best alternative until now is a population size of 120 without improvement by left shift heuristic. The investigation on mutation probability leads to best results with p_{mut} = 90%. To our mind, the unusual high probability of mutation is due to the nature of the selected mutation variant. In fact, the latter leads to taking another individual, but not wholly different from the first. The second solution belongs to the neighborhood of the initial one.

Impact of the clustering alternatives. The clustering tools used in computing the density are explored with different stopping criteria. We cote for instance a limiting distance which is function of the average distance or the median distance. These measures are computed from the distance matrix between the individuals of the whole population and children. The stopping criteria may also consist in a limiting number of clusters. In the latter case, the number of different ranks and different functions of the maximum rank value are tested

	Maximum number of clusters	
	maximum rank value	1.5 maximum rank value
K-means clustering	8.04%	0.91%
Hierarchical clustering	4.08%	1.01%

Table 2: Impact of clustering tools. 1 second, J=20.

Table 2 shows results of some of these experiments. The population size is fixed to 120. The left shift procedure is not applied. Obviously, the best average deviation is

obtained with the k-means clustering when the stopping criterion is equal to 1.5×the maximum rank value. This configuration will be used in the comparison.

Comparison with other algorithms. In this subsection, we compare our EA with the GA of Hartmann (2001), the EA of Elloumi et al. (2006), the simulated annealing of Bouleimen and Lecocq (2003), and finally with the truncated branch-and-bound (B&B) of Sprecher and Drexl (1998).

The computational time stopping criterion supposes that all algorithms are tested using similar computer frameworks, this is not the case. Analogously, the number of computed schedules criterion presumes that schedules require the same computational effort in all algorithms; this is not the case either. Hence, results are displayed using these two stopping criteria types.

	J	Av. Dev.	Max. dev.	Feasible	Optimal
New EA (a)	10	0.09	13.04	100.0	98.6
	12	0.13	8.69	100.0	97.3
	14	0.43	12.13	100.0	90.0
	16	0.46	13.79	100.0	88.9
	18	0.67	13.24	100.0	84.1
	20	0.91	21.53	100.0	78.52
	30	1.89	15.65	86.17	-
Hartmann's GA (b)	10	0.06	6.3	100.0	98.7
	12	0.14	9.1	100.0	97.3
	14	0.44	10.3	100.0	89.8
	16	0.59	10.5	100.0	87.8
	18	0.99	13.3	100.0	78.3
	20	1.21	14.2	100.0	73.3
	30	16.93	151.9	86.3	-
Elloumi et al.'s EA (a)	10	0.17	18.2	100.0	97.6
	12	0.34	10.0	100.0	93.2
	14	0.93	16.7	100.0	82.0
	16	1.28	21.7	100.0	76.0
	18	1.35	13.6	100.0	72.2
	20	2.18	18.2	100.0	62.5
Sprecher and Drexl's Truncated B&B (c)	10	0.00	0.0	100.0	100.0
	12	0.12	17.9	100.0	98.2
	14	1.46	33.3	99.6	85.7
	16	3.81	52.4	99.5	69.5
	18	7.48	77.4	98.0	57.4
	20	11.51	78.6	96.4	47.3
	30	57.22	244.0	55.8	-

Table 3: Comparison with other heuristics. (In %).

(a) Pentium 3.00 GHz, time limit 1 sec.
(b) Pentium 133 MHz, time limit 1 sec.
(c) Pentium 100 MHz, time limit 5 times the instance size (in seconds).

	J	Av. Dev.	Max. dev.	Optimal
New EA	10	0.09	13.04	98.6
	12	0.13	8.69	97.3
	14	0.43	12.13	90.0
	16	0.46	13.79	88.9
	18	0.67	13.24	84.1
	20	0.91	21.53	78.52
Bouleimen and Lecocq's SA	10	0.21	7.8	96.3
	12	0.19	6.3	91.2
	14	0.92	10.6	82.6
	16	1.43	12.9	72.8
	18	1.85	11.7	69.4
	20	2.10	13.2	66.9

Table 4: Comparison with the Simulated Annealing algorithm of Bouleimen and Lecocq (2003). 1 second, (in %).

Table 3 and 4 display a comparison between different algorithms in terms of the average and the maximum deviations, the percentages of instances to which a feasible and an optimal solutions are found. The deviations are computed from optimal solutions for the sets J10-20, and from lower bounds for J30 since no optima solutions are known to these instances.

(a) Set J0, 6000 schedules	Av. Dev.	Optimal
Hartmann (2001)	0.10	98.1
New EA (POP=90, GEN=66)	0.11	97.8
Alcaraz et al. (2003)	0.19	96.5
Kolish and Drexl (1997)	0.50	91.8
Özdamar (1999)	0.86	88.1

(b) Av. Dev (%), 5000 schedules	J10	J12	J14	J16	J18	J20
New EA (POP=60, GEN=83)	0.21	0.29	0.77	0.91	1.30	1.62
New EA (POP=90, GEN=55)	0.14	0.24	0.80	1.14	1.53	2.09
Alcaraz et al. (2003)	0.24	0.73	1.00	1.12	1.43	1.91
Józefowska et al. (2001)	1.16	1.73	2.6	4.07	5.52	6.74

Table 5: Comparison with other heuristics. Fitness including k-means stopped at a quantity of clusters equal to ten number of different ranks (In %).

Table 5 displays a comparison on the basis of the number of computed schedules stopping criterion. The best configuration of our algorithm depends on the set explored in addition to the stopping criterion. Remark that with a population size of 90, the algorithm succeeds to find better deviations than with 60, and this for the sets J10 and 12; this is not the case for J14-20. The new EA gives better results than the other algorithms except that of Hartmann though the difference is not important.

Conclusion

In this paper, we present a new Evolutionary Algorithm (EA) to solve the multi-mode resource constrained project scheduling problem (MRCPSP) with the objective of minimizing the project makespan. Since the problem of whether finding a feasible solution for the MRCPSP with at least two non renewable resources is NP-complete, a penalty function is introduced. The penalty is dealt as a criterion to be minimized; hence, the problem becomes bi-objective. Clustering heuristics are introduced into the rank-based fitness. The comparison with other algorithms recently proposed in the literature of MRCPSP proves the outperforming of our EA. Further research may be conducted by improving this algorithm, especially in order to reduce its computational efforts, or by applying it on new project scheduling contexts such as scheduling in uncertain environment.

References

Alcaraz, J., and Maroto, C. 2001. *A Robust genetic algorithm for resource allocation in Project scheduling*. Annals of Operations Research. 102 p. 83-109.

Alcaraz, J., Maroto, C. and Ruiz, R. 2003. *Solving the Multi-Mode Resource-Constrained Project Scheduling Problem with genetic algorithms.* Journal of Operation Research Society. 54 p. 614-626.

Bianco, L., Dell'Olmo, P. and Speranza, M.G. 1998. *Heuristics for multimode scheduling problems with dedicated resources*. European Journal of Operational Research. 107 p. 260-271.

Boctor, F.F. 1993. *Heuristics for scheduling projects with resource restrictions and several resource-duration modes.* International Journal of Production Research. 31 p. 2547-2558.

Boctor, F.F. 1996a. *A new and efficient heuristic for scheduling projects with resource restrictions and multiple execution modes.* European Journal of Operational Research, 90 p. 349-361.

Boctor, F.F. 1996b. *Resource-Constrained Project Scheduling by simulated annealing*. International Journal of Production Research. 34 p. 2335-2351.

Bouleimen, K., and Lecocq., H. 2003. *A new efficient simulated annealing algorithm for resource-constrained project scheduling problem and its multiple mode version*. European Journal of Operational Research, 149 p. 268-281.

Drexl, A. and Grüenewald., J. 1993. *Nonpreemptive multi-mode resource-constrained project scheduling*. IIE Transactions. 25 p. 74-81.

Elloumi, S., Loukil, T., and Teghem, J. 2006. *Ordonnancement de projets multi-mode sous contraintes de ressources: Un algorithme évolutionnaire basé sur l'information de l'efficacité.* ValenSciences N°5, Presses Universitaires de Valenciennes. P. 237-252.

Elmaghraby, S.E. 1977. *Activity networks: project planning and control by network models.* New York: Wiley.

Fonseca, C.M., and Fleming, P.J. 1993. *Genetic algorithms for multi-objective optimization: Formulation, discussion and generalization*. In Genetic Algorithms: Proceedings of the Fifth International Conference, (S. Forrest, ed.), San Mateo, CA : Morgan Kaufmann, July 1993.

Hartmann, S. 2001. *Project Scheduling with Multiple Modes: a genetic algorithm*. Annals of Operations Research. 102 p. 111-135.

Hartmann, S., and Drexl, A. 1998. *Project Scheduling with Multiple Modes: a comparison of exact algorithms.* Networks. 32 p. 283-297.

Józefowska, J., Mika, M., Różycki, R., Waligóra, G. and Weglarz, J. 2001. *Simulated annealing for Multi-Mode Resource-Constrained Project Scheduling*. Annals of Operations Research. 102 p. 137-155.

Kolisch, R., and Sprecher, A. 1996. *PSPLIB – A project scheduling problem library.* European Journal of Operational Research. 96 p. 205-216.

Kolisch, R., Sprecher, A., and Drexl, A. 1995. *Characterization and generation of a general class of resource-constrained project scheduling problems.* Management Science. 41 p. 1693–1703.

Kolisch, R., and Drexl, A. 1997. *Local search for nonpreemptive multi-mode resource-constrained project scheduling*. IIIE Transactions, 29 p. 987-999.

MacQueen, J. 1967. *Some methods for classification and analysis of multivariate observations*. In L.M. Le Cam and J. Neyman, editors, Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, volume 1, p. 281-297.

Özdamar, L. 1999. *A genetic algorithm approach to a general category project scheduling problem*. IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and reviews. 29 p. 44-59.

Słowiński, R., and Węglarz, J. 1978. *Solving the general project scheduling problem with multiple constrained resources by mathematical programming*. In: J. Stoer (ed.), Optimization Techniques. Lectures Notes in Control and Information Sciences 7, part 2, Springer-Verlag, Berlin-Heidelberg-New York, 278-288.

Sprecher, A. 1994. *Resource-Constrained Project Scheduling: exact methods for the multi-mode case.* Lecture Notes in Economics and Mathematical Systems 409 (Berlin: Springer.).

Sprecher, A., and Drexl, A. 1998. *Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm*. European Journal of Operational Research, 107 p. 431–450.

Sprecher, A., Hartmann, S., and Drexl, A. 1997. *An exact algorithm for project scheduling with multiple modes.* OR Spektrum, 19 p. 195–203.

Talbot, F.B. 1982. *Resource-constrained project scheduling with time–resource tradeoffs: The nonpreemptive case.* Management Science, 28 p. 1197–1210.

Weglarz, J. 1980. *Control in resource allocation systems.* Foundation of Control Engineering, 5 p. 159-180.