

# Mode List vs Activity List Representation for a Sustainable Multi-mode Resource-Constrained Project Scheduling Problem

Paper number 131 for fast track

## Abstract

The efficient and energetically sustainable planning of industrial processes is an outstanding challenge for both academic and the business community. This paper addresses an energy-based extension of the Multi-mode Resource-Constrained Project Scheduling Problems (MRCPSP) called MRCPSP-ENERGY. In this extension, the activities require a consumption of renewable resources and an amount of energy for their execution. Different energy consumptions result in different execution modes of the activities and therefore in a different scheduling and energy usage of a project. The objective is to maximize the efficiency of the project, which minimizes both *makespan* and energy consumption. Since the problem is NP-hard, the metaheuristic approaches are the most suitable techniques to tackle this problem. This paper shows that the activity list representation, commonly used in metaheuristics, can lead to a significant number of redundant solutions, that is, solutions that have different representations, but they are in fact the same. This point is a severe disadvantage for a search procedure. For this reason, a new Two-Phase Genetic Algorithm for solving the MRCPSP-ENERGY is proposed, trying to avoid the redundant solutions by focusing the search on execution modes. The proposed GA is evaluated and compared with instances of the PSPLIB-ENERGY library.

## 1 Introduction

The energy consumption in the industrial sector is growing by leaps and bounds. Based on the U.S. Energy Information Administration report in 2016, the industrial sector, including manufacturing, consumed approximately a third of the total delivered energy in the world (U.S. Energy Information Administration (EIA) 2016). The environmental implications of the industrial process are gaining more and more importance. Therefore, the energy consumption reduction in resource-allocation projects is a critical aspect of the industry sector (Zhang et al. 2016).

The Multi-mode Resource Constrained Project Scheduling Problem (MRCPSP) is one of the most studied scheduling problems because many problems can be modeled as

variants of it. One extension of this problem, which incorporates energy consumption in activities, is called the MRCPSP-ENERGY (Morillo Torres, Barber, and Salido 2017). It includes an additional optimization criterion based on energy consumption that gives rise to different execution modes of activities, and the objective is to maximize the efficiency of the project. This efficiency criterion is managed by combining the *makespan* and the energy consumption criteria into a new combined objective. The objective function of the MRCPSP-ENERGY is more sensitive than that of the traditional MRCPSP. This is because the traditional MRCPSP does not distinguish between solutions with different modes of execution if they do not affect the *makespan*, conversely, such solutions may lead to a different energy consumption in the MRCPSP-ENERGY. This paper addresses the MRCPSP-ENERGY for two main reasons: first, the broad interest of reaching efficient solutions in scheduling processes and second, because redundant solutions are easier to analyze under the presence of a highly sensitive objective function.

Solving the MRCPSP-ENERGY instances has an NP-Hard complexity, so existing exact methods can only find the solution to small-size problems in a reasonable time. Therefore, metaheuristic methods have become more relevant because they can find near-optimal solutions in a short time. Most metaheuristic methods use a solution representation based on the activity list and apply movement rules based on changes of order on this activity list to explore the neighborhood of a solution. This paper shows that this widespread representation can produce a large number of redundant solutions, which has a negative impact on the search effort. To avoid this disadvantage, a new Genetic Algorithm (GA) to solve the MRCPSP-ENERGY is proposed, which works with a list-based representation (activity and mode list) and includes two optimization phases. The first one is an optimization phase over the mode list. The second one is an optimization phase over the activity list. In each phase, the genetic operators are implemented on the corresponding list, leaving the other constant. The proposed algorithm is evaluated using instances in the PSPLIB-ENERGY library (Morillo Torres, Barber, and Salido 2017).

The main contribution is to show that searching through the mode list is a novel way to explore the solution space, achieving better solutions compared to the search through

the activity list. Besides, both search procedures can be combined to achieve even better solutions. On the other hand, this document emphatically encourages researchers to develop new solution methods for the MRCPSP-ENERGY taking advantage of this search alternative.

The paper is organized as follows. Section 2 presents the problem description. Section 3 briefly describes the main methodologies applied for solving the MRCPSP. Section 4 shows some examples of redundant solutions of the activity list-based representation. In Section 5, the new Genetic Algorithm is described, and the computational experiments and results are analyzed in Section 6. Finally, section 7 summarizes some concluding remarks.

## 2 Problem description

The MRCPSP-ENERGY (Morillo Torres, Barber, and Salido 2017) is an extension of the well-known Resource-Constrained Project Scheduling Problem (RCPSP). In the MRCPSP-ENERGY, the activities have different execution modes, associated with different energy consumptions. Additionally, activities require an amount of renewable resources for their execution. These resources are used by activities, and once an activity ends, the resources are again available for another. The goal is to maximize the relative efficiency of the project, which minimizes both the energy consumption and the *makespan*. Formally, the problem can be described as a project that consists of a set of  $n$  non-preemptive activities  $I = \{0, \dots, i, \dots, n\}$ , a set  $B$  of  $K^\rho$  shared renewable resources  $B = \{1, \dots, b, \dots, K^\rho\}$ , and an available amount  $R_b^\rho$  of every renewable resource. Each activity  $i$  has  $M_i = \{1, \dots, M\}$  execution modes, where each mode requires an execution time  $d_{im}$ , a total of  $r_{ib}^\rho$  renewable resource of type  $b$ , and an amount of energy  $e_{im}$  where  $m \in M_i$  for its realization. Activities are subject to precedence constraints, which indicate that each activity cannot be started before all its predecessor activities are completed. The different execution modes for each activity gives rise to different energy consumption and different execution times.

Figure 1 shows an example of an MRCPSP-ENERGY instance. It has 11 activities, the first and the last activity are dummy activities that represent the beginning and the end of the project, respectively. There are 3 renewable resources with a maximum availability of 4 units for each of them. The execution time and energy consumption are at top of each node, and the resource usage at the bottom. The arrows show the precedence relations between activities.

The following notation is considered. The total energy consumption of a project ( $TECP$ ) is the sum of the energy consumption  $e_{im}$  for each activity  $i \in I$ ,  $LB0_{\min}$  is the critical path with the shortest execution time,  $e_{\min}$  is the sum of energy consumption with lower consumption value,  $P_i$  is the set of immediate predecessor activities of activity  $i$ ,  $C_{\max}$  is the *makespan* of the project and  $(es_i)$  and  $(ls_i)$  are the earliest start time and the latest start time of activity  $i$ , respectively, both calculated by applying the forward and backward pass method, given an upper bound  $T$  for  $C_{\max}$ .

Also, let  $\xi_{imt}$  be a binary decision variable that takes value 1 when the activity  $i$  is executed in mode  $m \in M_i$  and

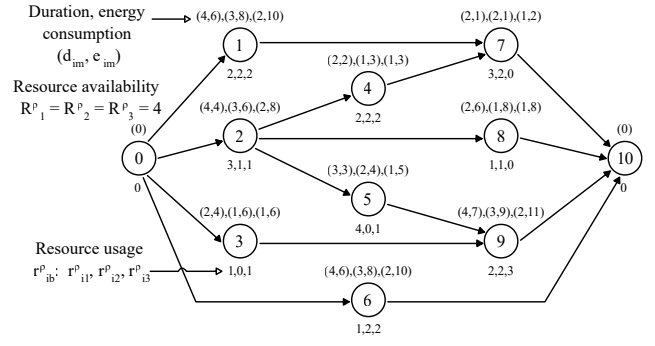


Figure 1: An MRCPSP-ENERGY example.

starts at time  $t$ , and 0 otherwise. The MRCPSP-ENERGY can be formulated as follows:

$$Max : \eta(C_{\max}, TECP) \quad (1)$$

Subject to:

$$\sum_{m=1}^{M_i} \sum_{t=es_i}^{ls_i} \xi_{imt} = 1 \quad \forall i \in I \quad (2)$$

$$\sum_{m=1}^{M_j} \sum_{t=es_j}^{ls_j} t * \xi_{jmt} \geq \sum_{m=1}^{M_i} \sum_{t=es_i}^{ls_i} (t + d_{im}) * \xi_{imt} \quad (3)$$

$$\forall i \in P_j, \forall j \in I$$

$$\sum_{i=0}^n r_{ib}^\rho * \sum_{m=1}^{M_i} \sum_{s=\max\{t-d_{im}, es_i\}}^{\min\{t-1, ls_i\}} \xi_{ims} \leq R_b^\rho \quad (4)$$

$$\forall b \in B, t = 1, \dots, T$$

$$\xi_{imt} \in \{0, 1\} \quad (5)$$

Where:

$$\eta(C_{\max}, TECP) = \frac{1}{UBPP} * \frac{1/C_{\max}}{TECP} \quad (6)$$

$$UBPP = \frac{1/LB0_{\min}}{e_{\min}} \quad (7)$$

Expression (2) ensures that each activity starts only once. Expression (3) represents the precedence constraints. Expression (4) ensures that resources availability is not exceeded. Finally, Expression (1) shows the optimization function where the objective is to maximize the relative efficiency  $\eta(C_{\max}, TECP)$ , which has been defined by Expressions (6) and (7). This criterion considers both the energy consumption and *makespan*, simultaneously. Overall, the relative efficiency is interpreted as the efficiency of the project regarding an upper bound of the project performance ( $UBPP$ ) (Expression (7)).

The MRCPSP-ENERGY is a strongly NP-Hard problem because it is a generalization of the standard RCPSP which is well-known NP-Hard.

### 3 Literature review

Most of the related academic literature is dedicated to MRCPSP with both renewable and non-renewable resources. The main difference between solution methods is the use of a penalty function as part of the objective function to avoid unfeasible solutions when the problem addresses non-renewable resources. Following the academic literature, solution methods can be classified into two groups: exact approaches and metaheuristic approaches.

Elmaghraby was the first one who considered different execution modes for activities in the project scheduling problem (Elmaghraby 1977) and later, several exact methods were proposed (Talbot 1982; Patterson et al. 1989; Sprecher and Drexl 1998; Zhu, Bard, and Yu 2006). Most exact methods are based on the branch-and-bound or branch-and-cut algorithms, enumeration schemes or lower boundaries. Unfortunately, the results show that exact methods can only solve small-sized problems. Indeed, there are MRCPSP instances in the PSPLIB library with only 30 activities for which optimal solutions have not been found.

Metaheuristic approaches are search algorithms that include escaping strategies from a local optimum, with the aim to explore and find a competitive approximation to a global optimum. Following Van Peteghem and Vanhoucke (Van Peteghem and Vanhoucke 2014), the metaheuristic procedures for solving the MRCPSP have four elements: Schedule and Mode Representations, Schedule Generation Schemes (SGS), metaheuristic algorithms, and local search procedures.

**Schedule Representations.** The representation stands for how to code a complete solution (the execution mode and the start time of each activity). The coding consists of one schedule representation and one mode representation. Kolisch and Hartmann distinguished five different schedule representations (Kolisch and Hartmann 1999), but currently, the activity list representation is the most used. To represent the execution modes, usually, a mode list is used.

**Schedule Generation Schemes.** To decode a schedule and mode representation in a complete solution, a Schedule Generation Scheme (SGS) is used. There are mainly two SGS, the serial and the parallel scheme (Kolisch and Hartmann 1999). Both schemes produce a feasible solution. In the serial scheme, the solution building is done through a single set of eligible activities that is updated at each step. In the parallel scheme, there are many sets of eligible activities that depend on the time when resources are available. The serial scheme produces a set of schedules that always contain the optimum, while the parallel scheme produces a set of schedules that might exclude the optimal solution. In spite of this fact, both schemes are used in the literature, because the parallel scheme usually builds more compact solutions than the serial scheme (Kim 2009).

**Metaheuristic algorithms.** There are several metaheuristic methods for solving the MRCPSP. One of the first ones, proposed by (Kolisch and Drexl 1997), consists of a local search that tries to find a feasible solution and then performs a single neighborhood search. In (Hartmann 2001) is proposed a GA, which uses a precedence-feasible activity list representation and the serial SGS as decoding rule. The al-

gorithm includes a new operator called inheritance, which does not significantly improve the solutions. In (Józefowska et al. 2001), a simulated annealing (SA) with and without a penalty function was proposed, the latter achieves the best results. They also use the activity list representation and the serial SGS. In (Bouleimen and Lecocq 2003), it is presented another implementation of SA, where the neighbor solutions are generated by using two phases. First, a mode-feasible solution is searched and then improved by random shifts of activities. The first tabu search was proposed by (Nonobe and Baraki 2002). It operates on the activity list representation, and neighbor solutions are generated either by a change on the mode list or by shifting some activities on the activity list. The authors propose a new solution building procedure, but the optimal solutions might not be reached by it. In (Alcaraz, Maroto, and Ruiz 2003), a GA with same schedule and mode representation was proposed, but they include an additional element: the forward/backward gene, which indicates the direction of serial SGS to generate a schedule. The results show that this algorithm outperforms the SA approach proposed by (Józefowska et al. 2001), but does not outperform the GA proposed by (Hartmann 2001). Alternatively, (Zhang, Tam, and Li 2006) proposed a particle swarm optimization (PSO), which uses two particles as solution representation. The first particle contains a vector with priority values of each activity (like the Random Key representation), and the second particle includes the information about the activity execution modes. Based on their results, the PSO achieves competitive results, though the GA outperforms the PSO.

More recently, in (Li and Zhang 2013), it is proposed an ant colony optimization algorithm (ACO) which uses two levels of pheromones, the first level is used to make the selection of an activity and insert it into a sequence. The second level is used to perform the execution mode assignment. In (Tseng and Chen 2009), a genetic local search with two phases was proposed. In the first one, an initial population is generated, and the best solutions are grouped into an elite set. In the second phase, a deep search is carried out in regions defined by the elite set. In (Lova et al. 2009), it is developed a hybrid GA (MM-HGA), which uses, in addition to the activity list representation, two additional genes: forward/backward gene and a serial/parallel gene. The first gene is related to the decoding direction and the second gene is related to the decoding algorithm. They also proposed a new normalized fitness function that relates the *makespan* and the units of resources from the excess of non-renewable resources. The reported computation results show that MM-HGA outperforms the other heuristics. Later, a bi-population version of GA was proposed by (Peteghem and Vanhoucke 2010). The main difference consists of using two different populations: one of them contains schedules only right-justified, and the other contains schedules only left-justified. The algorithm includes specific genetic operators and an adapted serial SGS with a local search for improving execution modes. The obtained results show that this algorithm achieves one of the best solutions in the literature.

To summarize, there are several proposed metaheuristic procedures to solve the MRCPSP and, based on the reported

computational experiments, the population-based, as well as hybrid metaheuristics, are those that achieve the best results. For an extensive study of these methodologies, we refer the reader to (Węglarz et al. 2011; Kolisch and Hartmann 2006).

**Local search procedures.** The purpose of local search procedures is to improve the current solution iteratively. They often focus on achieving feasibility or improving the *makespan*. One of most relevant local search is the Forward-Backward Improvement (FBI) proposed by (Tormos and Lova 2001) and adapted to be used in the MRCPSPP proposed by (Lova et al. 2009). The FBI consists of a backward and forward pass. In the first one, the activities are considered from right to left and are scheduled at the latest feasible time. Then, in the forward pass, the activities are considered from left to right and are scheduled at the earliest feasible time. In this way, the FBI can often achieve more compact schedules.

#### 4 Redundant solutions on activity list-based representations

It can be deduced from the previous section that the activity list-based representation has a fundamental role in metaheuristic procedures to solve MRCPSPP instances. Formally, the activity list representation is an array of  $n$  elements that represents activities. This list is feasible with respect to precedence constraints. The position of each activity in the array represents the priority of the activity to be scheduled by using an SGS. On the other hand, the mode list is also an array of  $n$  elements but the elements represent the execution mode of the activities in ascending order, i.e. the element  $i$  represents the execution mode of the activity  $i$ .

To decode a complete solution, the GA needs both an activity list and a mode list. The decoding can be carried out in several ways: by using the serial or parallel SGS and the forward or backward direction. In the backward direction, the predecessor and successor activities are swapped. In Figure 2, an example of an activity list representation and a mode list representation of a feasible solution for the instance represented in Figure 1 is shown.

Activity list:	10	7	6	1	4	9	8	5	2	3	0
Activities:	0	1	2	3	4	5	6	7	8	9	10
Mode list:	0	2	2	3	2	1	2	1	3	1	0

Figure 2: Example of an activity list and a mode list.

Metaheuristic procedures use this representation scheme for coding the solutions and then apply movement rules to make the search procedure in the neighborhood. Most procedures make the search by modifying the activity list. However, it can be easily deduced that different activity lists can obtain equal solutions when they are decoded, which are called redundant solutions. For example, Figure 3 shows two different activity lists (activity list 1 and activity list 2) and one mode list of feasible solutions for the instance presented in Figure 1. The two activity lists were decoded by using the

serial SGS in the forward direction. Although the two activity lists are different (see the underlined activities in Figure 3), the solutions obtained are exactly the same. It is worth noting that the differences between the activity lists are not just a simple activity swap.

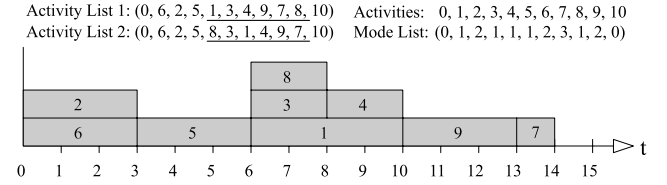


Figure 3: Example of redundant solutions using the activity list representation.

Redundancy in solutions can occur at any search procedure when permutations are generated over the activity list. In this way, we can see that the solution space (i.e., the set of permutations on a list of activities) is composed of three sets of solutions: unfeasible solutions, redundant solutions, and unique solutions. When the activity list is precedence-feasible, the search is only performed on redundant and unique sets of solutions. Estimating the number of redundant solutions of the MRCPSPP is not possible without, first, generating, decoding and checking all permutations. This is because redundant solutions depend on the sequence of scheduled activities and their use of resources at each time point for each solution. Nevertheless, based on the results of Section 6, the number of redundant solutions is much larger than the number of unique solutions. This fact is one of the first conclusions of these experiments and it is of relevance in the efficiency of the search process.

On the other hand, only two conditions must be fulfilled so that any change on the mode list guarantees a different solution (scheduling), while the list of activities remains unaltered. The first condition is that the duration of the execution modes of a selected activity must be different, and the second condition is that there must be other activities that consume the released resources or activities that have to be relocated due to the lack of them. As a change of mode causes a change in the selected activity duration, the serial or parallel decoding algorithms may or may not have available resources at different times of the scheduling. Furthermore, the successor activities will also be available at different time points. This will depend on the finishing time of the selected activity. As a result, decoding algorithms can build new combinations of activities that can be scheduled in this way: if the new duration the selected activity is lower, then the changes in the scheduling start from the new end time of such activity. If the new duration is higher, then the changes start from its former end time. As all possible combinations of modes are feasible, when selecting any activity and both conditions are satisfied, the space of feasible solutions and the space of unique solutions match.

The solution space for representations based only on a list of  $n$  activities is bounded by the number of permutations ( $n!$ ). Similarly, the solution space for representations based

on a list of  $m$  execution modes is bounded by the number of combinations ( $m^n$ ). Of course, the permutation space is rather larger than the combination space, as the number of activities increases. Figure 4 shows an illustration of the solution space of the activity list representation and the mode list representation. In this figure, the solution space of the mode list can be seen as solutions related to the combination of the execution modes on a list of activities.

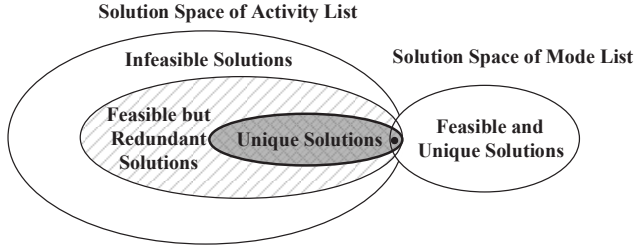


Figure 4: The solution space related to the activity list and the mode list representation.

Therefore, the problem lies in the fact that the computational effort of a search based only on permutations of the activity list could be wasted on redundant solutions. Taking into account that most of the metaheuristics are mainly focused on permutations of the activity list, this paper proposes an in-depth study of searching over the mode list, instead.

## 5 A Mode and activity list-based Genetic Algorithm

In this section, a new genetic algorithm (GA) to solve MRCSP-ENERGY that uses two optimization phases, based on the activity list and the mode list, is described. First, we briefly describe the codification scheme, the fitness function, the selection criteria, and the replacement process. Following, we will focus on the description of specific genetic crossing and mutation operators related to the two proposed optimization phases.

**Solutions encoding and fitness function.** The activity list and mode list plus two genes are used as the codification of solutions for the MRCSP-ENERGY. The two additional genes are the SGS gene and the direction gene. The SGS gene represents the method to be used to decode the solution: its value will be 0 when the serial scheme is used or 1 when the parallel scheme is used. The direction gene can be forward or backward (with a value equal to 0 or 1, respectively).

The fitness function is the relative efficiency of the project, as it was described in section 2. The objective is to maximize this value.

**Initial population.** To generate the initial population, the Regret Based Biased Random Sampling (RBBS) with the Latest Start Time (LST) as a priority rule is applied. The values of the SGS gene, direction gene and execute mode are randomly chosen.

**Population size.** There is not a standard method to estimate the best population size in this kind of problems. Most

authors carry out computational experiments to estimate this parameter. Based on our experiments, was found that the population size should be related to the number of activities and to the number of available iterations. Let  $Pop$  be the population size,  $NoI$  be the number of iterations and  $NoG$  be the number of generations. Such parameters are defined in Expression 8.

$$Pop = \left( \frac{1}{n} * NoI \right) + 15 \quad \text{and} \quad NoG = \frac{NoI}{Pop} \quad (8)$$

**Selection.** Stochastic sampling with replacement is used, depending on the fitness value of the solutions. Therefore, every individual in the population has a probability of being chosen according to its fitness value. When an individual is chosen, a replica of that individual is included in the next selection.

**Replacement.** It refers to how to create the next population  $P3$  from the parents' population  $P1$  and the offspring population  $P2$ . First,  $P1$  and  $P2$  are sorted based on their fitness value. Then,  $P3$  is built with 50% of the best  $P1$  individuals and 50% of the best  $P2$  individuals.

**Local improvement.** Two simple local improvements are used in the proposed GA. The first one is the well-known forward-backward improvement (FBI) described in Section 3. It is applied to the initial population. The second local improvement consists of reviewing all activities checking if they can be executed with less energy consumption (longer execution time) without breaking precedence and resource constraints as long as the *makespan* remains unchanged. It is applied to the best final solution found.

### 5.1 Optimization phases

As it was pointed out, the proposed GA divides the search process into two optimization phases: the first one uses crossover and mutation operators over the mode list to expand the search. This phase plays the most important role in our proposed GA algorithm. The second optimization phase uses crossover and mutation operators over the activity list, and it is done at the end of the algorithm to intensify the search. Each phase has specific genetic operators (crossover and mutation), which are detailed below.

#### Genetic operators for the optimization on the mode list

**Crossover.** It allows building new solutions from two selected parents. These solutions must share features of both parents. We use a two-point crossover only on the mode list. Therefore, two random integers  $q_1$  and  $q_2$  with  $0 < q_1 < q_2 < n$  are generated. The first genes from 0 to  $q_1$  are taken from parent 1, the next genes from  $q_1$  to  $q_2$  are taken from parent 2 and the genes remaining are taken from the parent 1 again. The activity list is randomly inherited only from one of the two parents. The SGS gene and direction gene are inherited when they are equal in both parents, otherwise, they are randomly generated.

**Mutation.** Although mutation does not generally have a main role in genetics operators, it introduces new genetic material which encourages exploration. Particularly, a slight perturbation over the mode list would cause great changes

over the solution. The mutation consists of randomly selecting a solution, then taking randomly an activity and changing its execution mode.

#### Genetic operators on the activity list

**Crossover.** A modified two-point crossover is used over the activity list. Thus, two random integers  $q_1$  and  $q_2$  with  $0 < q_1 < q_2 < n$  are generated. Thus, the first genes from 0 to  $q_1$  are taken from parent 1, next genes from  $q_1$  to  $q_2$  are taken from parent 2. Here, these genes inherited ( $q_1$  to  $q_2$ ) might not be those who are in the positions  $q_1$  to  $q_2$  of parent 2, such that they must be searched from the beginning of the activity list of parent 2, and then the first activities that are not repeated in the child are inherited. In the same way, the remaining ones ( $q_2$  to  $n$ ) are taken from the parent 1 again, without repeating genes from both parent 1 and parent 2. The modes of activities are inherited from their corresponding parents. For SGS and direction genes, the gene value is inherited when it is the same in both parents and randomly generated, otherwise.

**Mutation.** A multi-insertion mutation operator is proposed to minimize the probability to produce the same solution. It consists of randomly selecting an activity  $i$  and then, inserting it in another randomly chosen position, this new position must be feasible by predecessors. This process is repeated a fixed number of times for different activities. The mutation is implemented by chromosome (solution) instead of by gene (activity).

Finally, the GA parameters were fixed through experimentation on the PSPLIB-ENERGY library. The set of tests carried out indicates that the number of insertions for the mutation in the second optimization phase must be constant with a value of three insertions. Besides, the number of iterations to change from the first to the second optimization phase was determined as  $2/3$  of the total number of iterations. This value represents the importance of the optimization phase on the mode list with regard to the optimization phase on the activity list. Likewise, the experimental results show that a high mutation probability achieves better solutions than a low value. Thus, the probability of mutation for both optimization phases was fixed at 90%.

Figure 5 shows an example of how the offspring are generated by using both the activity list optimization and the mode list optimization. In this example, the serial SGS and the forward direction are used. The schedule  $A$  is the Parent 1 (genes with overscore) and the schedule  $B$  is the Parent 2 (genes with underscore), which were used to generate the schedules  $C$  and  $D$ . The schedule  $C$  is created by the optimization on the activity list. A two-point crossover is used. Thus, the first 5 activities are inherited from Parent 1, the following 3 activities are searched from the beginning of the activity list of Parent 2, and only those that are not repeated in the activity list of schedule  $C$  (activities 1, 4, and 7) are inherited. Finally, the 3 remaining activities are searched from Parent 1. As can be seen in figure 5, despite using a two-point crossover from different parents, schedule  $C$  has the same activity list as Parent  $A$ . The only reason for solutions to be different is that the inherited activity modes from Parent  $B$  have different durations to the those of Parent  $A$ . The schedule  $D$  is created by the optimization on the mode list;

the schedule inherits the activity list from Parent 2 and the mode list is created by a two-point crossover, where the fifth and sixth position of the mode list are inherited from Parent 2. Despite the activity list of Parent 2 and the activity list of schedule  $D$  are exactly the same, the corresponding solutions are very different, due to the different mode lists. In fact, the schedule  $D$  has the best fitness value among the four schedules.

Usually, it is assumed that modifying only the list of modes, it only affects the execution modes of an already pre-established order of activities (determined from the list of activities) to be scheduled. However that pre-established order also changes and therefore the solution, as shown in the Figure 5.

## 6 Empirical assessment and results

The performed empirical assessment of the proposed GA and the obtained computational results are divided into two groups: the first group is focused on the impact of the redundant solutions of the activity list-based representation. The second group is focused on the performance assessment of the proposed GA by using the PSPLIB-ENERGY library.

The PSPLIB-ENERGY library (Morillo Torres, Barber, and Salido 2017) is a set of MRCPSP-ENERGY instances. It consists of four sets of problems  $j30$ ,  $j60$ ,  $j90$ , and  $j120$ , each of them with 480 instances, except the last one that has 600 instances. Each set has 30, 60, 90, and 120 jobs, respectively. All problems are composed of activities with three execution modes.

### 6.1 Impact of redundant solutions in the activity list-based representation

To carry out this evaluation, an exhaustive search was carried out through all the permutations of the activity list and all the combinations of the mode list. Since the high computational cost of this search, a set of 480 instances, each with 10 activities, was adapted from set  $j30$  of the PSPLIB-ENERGY library. For the complete set of permutations generated by the activity list (Figure 6) and the full set of combinations generated by the mode list (Figure 7), the following measures are shown: (i) the total number of solutions (total), (ii) the number of feasible solutions including redundant ones (feasible), and (iii) the number of unique solutions (unique).

In Figure 6, the total number of permutations ( $10! = 3\,628\,800$ ) is the same for all instances. Due to the huge number of permutations, Figure 6 is shown in logarithmic scale. As it can be observed, the number of redundant solutions were rather higher than the unique ones. On average of the 480 instances, 99, 63% of the feasible solutions were redundant ones. Even there were some instances with only one unique solution.

In Figure 7, the total number of combinations for each activity list ( $3^{10} = 59\,049$ ) is the same for all instances, and all combinations are feasible. There are some instances with a number of unique solutions different from the number of feasible solutions. This is because some activities have equal execution modes. For instance, in the instance shown in Figure 1, the activity 3 has the mode 2, that corresponds

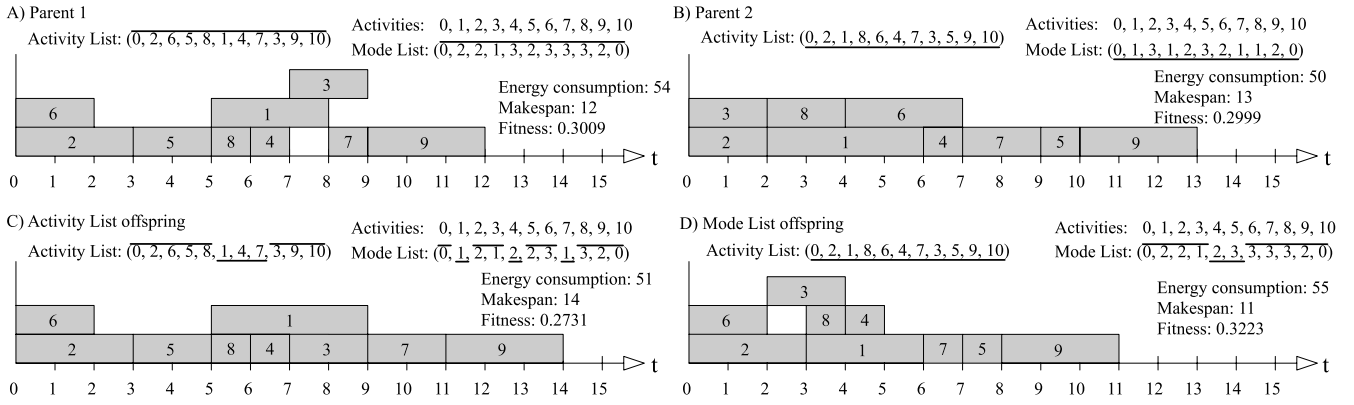


Figure 5: An example of offsprings, by using the optimization phases on the activity list (Offspring C) and on the mode list (Offspring D).

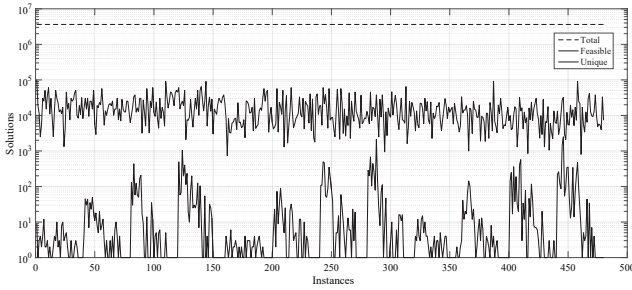


Figure 6: Activity list permutations for set j10.

to a duration of 1 time unit and an energy consumption of 6 units. This activity cannot be executed in lower time than 1, and thus the duration of mode 3 is equal than mode 2.

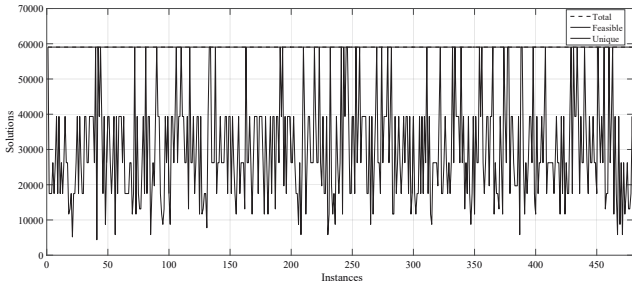


Figure 7: Mode list combinations for set j10.

Therefore, although the solution space of the activity list representation is higher than the solution space of the mode list representation, most of its solutions are redundant, which makes the search unsuccessful. In contrast, the solution space of the mode list representation is composed of feasible and unique solutions. In many cases, these sets are the same, allowing the search always to explore different solutions. There are only redundant solutions when there are

activities with duplicated durations of execution modes or when there are no activities that are affected by the change of modes in a partial solution.

## 6.2 Assessment of the proposed genetic algorithm

At this point, we assess the performance of the proposed GA by using the PSPLIB-ENERGY library. As it is usual in the evaluation of the classical RCPSP, sets  $j30$ ,  $j60$  and  $j120$  are assessed. To the best of our knowledge, (Morillo Torres, Barber, and Salido 2017) is the only work that reports results for the instances in this library. Therefore, the proposed GA was compared with their algorithm (called in this paper Basic-GA). On the other hand, one of the most used stopping criteria in literature for the RCPSP and MRCPSP is the maximum number of iterations. An iteration is defined as a complete scheduling (Kolisch and Hartmann 2006). In this paper, the same stopping criterion is used.

Table 1 shows the relative efficiency of the solutions obtained by different algorithms. Test cases consist of three sets of instances of the PSPLIB-ENERGY for which 1 000, 5 000, and 50 000 iterations were allowed. To evaluate the impact of the different optimization phases, we propose three variants of our proposed algorithm: the GA with the optimization only on the activity list (AL-GA), the GA with the optimization only on the mode list (ML-GA) and the GA with the two optimization phases (TP-GA). Table 1 is sorted by the 50 000 column.

As it can be seen from Table 1, the ML-GA outperformed the AL-GA (note that the objective is to maximize the relative efficiency of the project). It is interesting to remark that in the ML-GA, the activity lists of all instances remained constant once the initial population was created, and then only the execution modes have changed. On the other hand, in AL-GA, the execution mode of all activities remained constant, and the activities are always inherited with the execution mode corresponding to its parent. This indicates that searching through the mode lists can achieve high-quality solutions, as compared to searching through the activity lists.

Due to redundant solutions with different activity lists, an optimal solution can be represented by a mode list and sev-

Table 1:  $\bar{\eta}$  obtained by using the proposed genetic algorithm for solving PSPLIB-ENERGY library.

j#	Algorithm	Iterations/ $\bar{\eta}$		
		1 000	5 000	50 000
j30	<b>TP-GA</b>	<b>0,6400</b>	<b>0,6526</b>	<b>0,6575</b>
	ML-GA	0, 6386	0, 6507	0, 6572
	AL-GA	0, 6344	0, 6499	0, 6568
	Basic-GA	0, 5966	0, 6091	0, 6293
j60	<b>TP-GA</b>	0, 6558	<b>0,6863</b>	<b>0,6927</b>
	ML-GA	<b>0,6573</b>	0, 6791	0, 6908
	AL-GA	0, 6398	0, 6672	0, 6905
	Basic-GA	0, 6029	0, 6182	0, 6424
j120	<b>TP-GA</b>	0, 5185	0,5376	<b>0,5595</b>
	ML-GA	<b>0,5206</b>	<b>0,5392</b>	0, 5588
	AL-GA	0, 5065	0, 5203	0, 5476
	Basic-GA	0, 4760	0, 4875	0, 5032

eral activity lists. Thus, by fixing a list and performing the search on the other one, optimal solutions can be excluded. However, the proposed TP-GA searches on the mode list first and, in the final phase, on the activity list. Both search procedures are combined to avoid excluding optimal solutions and to achieve better results. In fact, Table 1 shows that the TP-GA outperformed all other algorithms.

To measure the impact of redundant solutions on the search based on the activity list, we calculate the average number of redundant offspring generated by the AL-GA, where both parents were different from each other. The offspring from equal parents were not counted. From the results, the average number of redundant solutions that does not contribute with new information to GA was at least 15% from the total number of solutions created.

On the other hand, since few papers have reported results of the PSPLIB-ENERGY library, we carried out a comparison of the proposed algorithm with the well-known ILOG CPLEX CP Optimizer 12.7.1 for solving the MRCPSP-ENERGY instances. We fixed, for both methods, a timeout of 10 seconds per instance. Table 2 shows a summary of the results. For the  $j30$  set, the CP-optimizer method obtained a higher relative efficiency value ( $\bar{\eta}$ ) than the TP-GA, with a difference of 0.0471%. This difference was reduced in set  $j60$  with a value of 0.0216%, and finally, the TP-GA outperformed CP-optimizer in set  $j120$  with a 0.1825% difference. In all instances, the number of optimal solutions were quite similar. These results show that the proposed algorithm is highly competitive and able to find efficient solutions in a short time.

## 7 Conclusions

In this paper, the Multi-mode Resource-Constrained Project Scheduling Problem, and particularly the MRCPSP-ENERGY, have been addressed to analyze the impact of redundant solutions generated by the activity list represen-

Table 2: Comparison of the proposed GA and an CP-Optimizer.

Label	CP-Optimizer	TP-GA
j30		
<b>Objective function (<math>\bar{\eta}</math>)</b>	<b>0.65897</b>	<b>0.65836</b>
Optima	166	160
Instances	480	480
j60		
<b>Objective function (<math>\bar{\eta}</math>)</b>	<b>0.69443</b>	<b>0.69428</b>
Optima	20	17
Instances	480	480
j120		
<b>Objective function (<math>\bar{\eta}</math>)</b>	<b>0.55366</b>	<b>0.55467</b>
Optima	0	0
Instances	600	600

tation and to propose an undervalued search alternative to reduce them. To this end, a new genetic algorithm has been proposed, focusing the search on mode lists instead of activity lists. This proposal is based on the fact that, although genetic operators can modify the activity list, the resulting solution (complete scheduling) can be exactly the same, even if they come from different activity lists. Therefore, the computational effort of a search might be wasted on redundant solutions.

The results show that when the search is done on the activity list, a large number of redundant solutions is generated. Conversely, a change of mode in a selected activity on the mode list guarantees a different programming if and only if two conditions are met. The first is that such activity must have execution modes with different durations and the second is that there must exist at least another activity that can be relocated for the excess or lack of resources caused by the change of mode of the selected activity. The computational results also show that the proposed genetic algorithm achieved the best results regarding previous algorithms for solving the PSPLIB-ENERGY library in all problem sets. Besides, the results reveal that the search by modifying only mode lists can achieve better results than the search by modifying only activity lists. Finally, the results compared with a well-known tool (CP-Optimizer) showed that the proposed algorithm is highly competitive and it is able to find near-optimal solutions in a short time, outperforming CP-Optimizer for high complexity instances.

Based on these results and the fact that the literature about scheduling problems with different execution modes is focused on the optimization of the activity list, we think there exists a valuable field of research focused not only on the optimization of the activity list but also on the optimization of the mode list. In fact, the latter became the most successful search procedure when compared with the former.

## Acknowledgements

The paper has been partially supported by the research projects XXXX and YYYY.

## References

- Alcaraz, J.; Maroto, C.; and Ruiz, R. 2003. Solving the Multi-Mode Resource-Constrained Project Scheduling Problem with Genetic Algorithms. *The Journal of the Operational Research Society* 54(6):614–626.
- Bouleimen, K., and Lecocq, H. 2003. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research* 149(2):268–281.
- Elmaghraby, S. E. 1977. *Activity Networks: Project Planning and Control by Network Models*. John Wiley & Sons Inc.
- Hartmann, S. 2001. Project Scheduling with Multiple Modes: A Genetic Algorithm. *Annals of Operations Research* 102(1-4):111–135.
- Józefowska, J.; Mika, M.; Różycki, R.; Waligóra, G.; and Węglarz, J. 2001. Simulated Annealing for Multi-Mode Resource-Constrained Project Scheduling. *Annals of Operations Research* 102(1):137–155.
- Kim, J.-L. 2009. Proposed methodology for comparing schedule generation schemes in construction resource scheduling. *Proceedings of the 2009 Winter Simulation Conference (WSC)* 2745–2750.
- Kolisch, R., and Drexl, A. 1997. Local search for nonpreemptive multi-mode resource-constrained project scheduling. *IIE Transactions* 29(11):987–999.
- Kolisch, R., and Hartmann, S. 1999. Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis. In Węglarz, J., ed., *Project Scheduling*. Springer US. chapter 7, 147–178.
- Kolisch, R., and Hartmann, S. 2006. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research* 174(1):23–37.
- Li, H., and Zhang, H. 2013. Ant colony optimization-based multi-mode scheduling under renewable and nonrenewable resource constraints. *Automation in Construction* 35:431–438.
- Lova, A.; Tormos, P.; Cervantes, M.; and Barber, F. 2009. An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. *International Journal of Production Economics* 117(2):302–316.
- Morillo Torres, D.; Barber, F.; and Salido, M. A. 2017. A new model and metaheuristic approach for the energy-based resource-constrained scheduling problem. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 095440541771173.
- Nonobe, K., and Baraki, T. 2002. Formulation and Tabu Search Algorithm for the Resource Constrained Project Scheduling Problem. In *Essays and Surveys in Metaheuristics*. Springer US. 557–588.
- Patterson, J. H.; Slowinski, R.; Talbot, F. B.; and Węglarz, J. 1989. An algorithm for a general class of precedence and resource constrained scheduling problems. *Advances in Project Scheduling* 3–28.
- Peteghem, V. V., and Vanhoucke, M. 2010. A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research* 201(2):409–418.
- 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(2):431–450.
- Talbot, F. B. 1982. Resource-Constrained Project Scheduling with Time-Resource Tradeoffs: The Nonpreemptive Case. *Management Science* 28(10):1197–1210.
- Tormos, P., and Lova, A. 2001. A Competitive Heuristic Solution Technique for Resource-Constrained Project Scheduling. *Annals of Operations Research* 102(1-4):65–81.
- Tseng, L.-Y., and Chen, S.-C. 2009. Two-Phase Genetic Local Search Algorithm for the Multimode Resource-Constrained Project Scheduling Problem. *IEEE Transactions on Evolutionary Computation* 13(4):848–857.
- U.S. Energy Information Administration (EIA). 2016. Monthly Energy Review - October 2016. Technical report, Office of Energy Statistics, Washington.
- Van Peteghem, V., and Vanhoucke, M. 2014. An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. *European Journal of Operational Research* 235(1):62–72.
- Węglarz, J.; Józefowska, J.; Mika, M.; and Waligóra, G. 2011. Project scheduling with finite or infinite number of activity processing modes - A survey. *European Journal of Operational Research* 208(3):177–205.
- Zhang, Z.; Tang, R.; Peng, T.; Tao, L.; and Jia, S. 2016. A method for minimizing the energy consumption of machining system: integration of process planning and scheduling. *Journal of Cleaner Production* 137(20):1647–1662.
- Zhang, H.; Tam, C. M.; and Li, H. 2006. Multimode Project Scheduling Based on Particle Swarm Optimization. *Computer-Aided Civil and Infrastructure Engineering* 21(2):93–103.
- Zhu, G.; Bard, J. F.; and Yu, G. 2006. A Branch-and-Cut Procedure for the Multimode Resource-Constrained Project Scheduling Problem. *INFORMS Journal on Computing* 18(3):377–390.