Hybridizing Constraint Programming and Meta-Heuristics for Multi-Mode Resource-Constrained Multiple Projects Scheduling Problem

Arben Ahmeti · Nysret Musliu

the date of receipt and acceptance should be inserted later

Abstract The Multi-Mode Resource-Constrained Multiple Projects Scheduling Problem (MMRCMPSP) is an important real-life problem. The aim is to schedule activities belonging to multiple project instances respecting different shared resources, precedence, and time constraints. To solve this problem we propose a new hybrid approach combining constraint programming (CP) and a meta-heuristic based algorithm. To this aim, we propose and evaluate a CP model that includes all constraints of MMRCMPSP. The hybrid approach takes advantage of the complementary features of CP and meta-heuristics. Our method outperforms state-of-the-art methods for this class of problems by generating new upper bounds for several instances. Moreover, we evaluate our method on the existing well-studied benchmark instances for multiple-mode resource constrained single project scheduling problems and provide new upper bounds for many instances.

Keywords Meta-Heuristics and Constraint Programming · Hybrid approach · Iterated Local Search · Project Scheduling · Min Conflicts

1 Introduction

Scheduling problems sum up a class of various combinatorial optimization problems of high interest for academics and industry. In particular, project scheduling is an important representative of this type of problems that usually has to deal with scheduling of activities of project/s under different types of

Arben Ahmeti
Institute of Logic and Computation, DBAI, TU Wien
E-mail: aahmeti@dbai.tuwien.ac.at

Nysret Musliu
Christian Doppler Laboratory for Artificial Intelligence and Optimization for Planning and Scheduling, Institute of Logic and Computation, DBAI, TU Wien
E-mail: musliu@dbai.tuwien.ac.at

constraints (resource constraints, precedence constraints, time horizons, etc.) and different types of objectives. One of the most studied problems of this class, Resource Constrained Project Scheduling Problem (RCPSP), have to deal with scheduling of activities of a project that require a certain amount of resources and are subject to precedence constraints among them. According to a review article [1], the RCPSP belongs to the group of NP-hard optimization problems. In many cases, the objective of this problem is the minimization of the project makespan. One of the most prominent benchmark libraries for RCPSP instances, PSPLIB, is provided by Kolisch and Sprecher [2]. Various enhanced versions of the RCPSP of academic and engineering interest have been introduced over time. A general variant (Multi-Mode Resource-Constrained Multiple Projects Scheduling Problem (MMRCMPSP)) of RCPSP is presented at the MISTA 2013 challenge.

In this paper, we focus on the MMRCMPSP problem. It extends RCPSP problem from two perspectives [3]: activities of a project may have more than one mode of execution (MRCPSP problem) and multiple instances of projects sharing scarce resources need to be scheduled simultaneously (RCMPSP problem). MMRCMPSP is a closer representation of the real-world scheduling problems. It consists on simultaneous scheduling of multiple project instances in an optimal schedule while their activities can be executed in more than one of modes and are subject to different types of resources, time, and precedence constraints. Moreover, the main objective in case of this problem is the minimization of the total project delay (TPD) and the total makespan (TMS) of projects serves as a tie-breaker.

MMRCMPSP as a more generalized form of project scheduling problem attracted the attention of many researchers. In the review article [4] a combination of Monte-Carlo Tree Search (MCTS) and hyper-heuristics were proposed. This was the winning approach of MISTA 2013 challenge. Geiger [5] introduced an iterated variable neighborhood search with four neighborhood structures. In [6] a multi-neighborhood, parallel local search approach was proposed. Another meta-heuristic approach based on iterated local search was introduced in [7]. [8] implemented a genetic algorithm for a multi-project environment with projects with assigned due dates and a resource dedication policy. A stochastic local search procedure with two neighborhoods was implemented by [9]. In [10] a genetic algorithm for mode assignment and a priority rules heuristic for job selection in MMRCMPSP was implemented. Other proposed methods for this problem may be found in [11], [12], [13] and [14]. Regardless of the contributions of many researchers to MMRCMPSP, optimal solutions of many existing instances are not known yet. In this paper, we introduce a new hybrid approach that combines a constraint programming approach with a meta-heuristic that extends the previous method proposed in [7]. Our approach outperforms the best existing algorithm [4] (to our best knowledge) for MMRCMPSP, by providing new upper bounds to many benchmark problems. Additionally, our approach provides new upper bounds for fifty benchmark instances (MMLIB library) for the MRCPSP problem [15]. The main contributions of this paper are:

- We provide a constraint programming model for MMRCMPSP and apply a state of the art CP solver to solve existing problem instances. Although constraint programming has been used previously for related project scheduling problems, to the best of our knowledge this is the first time that CP is applied for MMRCMPSP, which includes several extensions. Our model was tested with thirty benchmark instances from MISTA 2013 challenge. Long runs of algorithm resulted in new upper bounds for six instances. Execution of algorithm under time restriction conditions (5 minutes run per instance) gave the best results for three instances compared to other best solvers.
- The improvement of a local search based algorithm proposed in [7]. The improved version includes a new neighborhood operator that improved results of [7] for most benchmark instances. Additionally, it provides new upper bounds for five multiple-mode resource constrained single project scheduling problem instances.
- Proposal of a hybrid algorithm that combines the CP model and the improved meta-heuristic approach. The hybrid algorithm improved further the results and outperformed best state-of-the-art solver ([4]) for MMRCMPSP in many instances. We provide new upper bounds for almost half of the benchmark instances (fourteen new upper bounds and four equal upper bounds out of thirty) and fifty new upper bounds for multiple-mode resource constrained single project scheduling problem benchmark instances.

2 Problem Description

The MMRCMPSP problem [3] is comprised of a set of n projects: $P = \{1, 2, .., n\}$ and every project $i \in P$ is comprised of activities J_i that are executed in more than one of the modes taking into account different shared resources, time, and precedence constraints. Also, every project has a release date r_i, i.e. the earliest time when its activities could start. Every activity $j \in \{1, 2, .., |J_i|\}$ of every project has to be scheduled, i.e. its starting time s_{ij} has to be defined considering all constraints. The first and last activities of projects are dummy activities with only one execution mode, duration equal to zero and no resource requirements. There are sets of renewable and non-renewable resources $L_i \in \{1, ..., |L_i^\rho|, |L_i^\rho| + 1, ..., |L_i^\rho| + |L_i^v|\}$, where $L_i^\rho \in \{1, ..., |L_i^\rho|\}$ indicates renewable resources and $L_i^v \in \{|L_i^\rho| + 1, ..., |L_i^\rho| + |L_i^v|\}$ non-renewable resources. All non-renewable resources have fixed capacities for the whole project duration, for every project. Renewable resources have a fixed capacity per time unit. There are local renewable resources dedicated to a specific project only and global renewable resources (G^ρ) that are shared among all the projects. The availability of global renewable resources is limited by c_g^ρ, $g \in G^\rho$. There are no global non-renewable resources. Every activity has more than one available execution mode. An execution mode of an activity defines time duration required to complete the activity and its specific resource re-

quirements. Execution modes of activities are defined by $M_{ij} \in \{1, ..., |M_{ij}|\}$ and d_{ijm} (duration of activity $j \in J_i, i \in P$ in mode $m \in M_{ij}$). Moreover, r^{ρ}_{ijml}, r^{v}_{ijml} and r^{ρ}_{ijmg} determine the requirements for local renewable, non-renewable and global renewable resources, respectively, when activity $j \in J_i, i \in P$ is processed in mode $m \in M_{ij}$. Feasible projects schedules must always satisfy following hard constraints:

- For every local non-renewable resource $l \in L^v_i$ dedicated to every project $i \in P$, its total consumption cannot exceed its capacity $l \leq c_{il}$.
- For every local renewable resource $l \in L^{\rho}_i$ dedicated to every project $i \in P$, its total consumption at time unit t cannot exceed its capacity $l \leq c_{il}$.
- For every global renewable resource $g \in G^{\rho}$, its total resource consumption at time unit t cannot exceed its capacity $l \leq c^{\rho}_g$.
- Particular activities may require the completion of other activities before they start. In that case, feasible schedules must fulfill all precedence constraints between such activities, i.e. if activity $j \in J_i$, which is executed in mode m, must precede activity $j' \in J_i$.
- Release time of every project is respected, i.e. for each activity $j \in \{1, 2, .., |J_i|\}$ of every project $i \in P$, its start time $s_{ij} \geq 0$ and $s_{ij} \geq r_i$.

The objective is to find a feasible schedule with minimum total project delays (TPD) and projects total makespan (TMS). TPD is the primary objective and TMS is used as a tie-breaker. Project delay of a project i is defined as the difference between Critical Path Duration (CPD), a theoretical lower bound on the earliest finish time of the project, and the actual project duration (makespan):

$$PD_i = MS_i - CPD_i \tag{1}$$

MS_i - makespan of project i is calculated as difference:

$$MS_i = f_i - r_i \tag{2}$$

f_i - finish time of project i,
r_i - the release date of project i,
Total project delay is calculated as:

$$TPD = \sum_{i=1}^{n} PD_i \tag{3}$$

n - the number of projects.

Total makespan is the duration of the complete multi-project schedule:

$$TMS = \max_{i \in P} (f_i) - \min_{i \in P} (r_i) \tag{4}$$

Both soft constraints are combined into a single objective function as follows:

$$F = a * TPD + TMS \tag{5}$$

where value a in the MISTA 2013 challenge was $a = 100,000$.

3 The constraint programming model for MMRCMPSP

As stated earlier, MMRCMPSP generalizes other types of scheduling problems and is closer to real-world problems representations. Different models for different scheduling problems have been presented in the literature. We built our model using several features of the model for MRCPSP scheduling problems presented in [16].
Main extensions to our MMRCMPSP model are related to modeling of:

- Constraints related to a set of global renewable resources (G^{ρ}) that are shared among all the projects and their availability is limited by c^{ρ}_{g}, $g \in G^{\rho}$ (equation 18). There are no global non-renewable resources.
- Release dates constraints and dummy activities for every project (equations 7-9).
- Implementation of an objective function that consists on finding a feasible schedule that fulfills constraints while minimizing the total project delay (TPD) and the total makespan (TMS). Project delay is defined as the difference between Critical Path Duration (CPD) and the actual project duration (makespan). TPD is the primary objective and TMS as the duration of the whole multi-project schedule is used as a tie-breaker (equation 20).
- Multi-dimensional data structures to model multi-project instances execution modes, local renewable and non-renewable resources and add global resources.

We further describe the main components and constraints of the CP model. We have used the IBM CP Optimizer to implement it. According to the problem definition, every project $i \in P = \{1, 2, .., n\}$ is comprised of a set of non-preemptive activities or jobs J_i and has a release date r_i, i.e. the earliest time when the activities of the project i can start.
Activities are modelled as decision variables:

$$\text{interval}\, J_i \qquad \forall i \in P \tag{6}$$

and projects' release dates constraints and dummy activities:

$$\text{startOf}(s_i) = r_i \qquad \forall i \in P, s_i \in P_i \tag{7}$$

$$\text{startOf}(s_i) \leq \text{startOf}(j) \qquad \forall i \in P, j \in P_i \tag{8}$$

$$\text{endOf}(j) \leq \text{startOf}(e_i) \qquad \forall i \in P, e_i, j \in P_i \tag{9}$$

Every activity $j \in J_i$, of every project $i \in P$, has one or more available execution modes $m \in M_{ij}$. The execution mode of an activity determines duration d_{ijm} required to complete the activity and its specific resource requirements. Execution modes and processing time in every mode for every activity are modeled as decision variables in our model as well:

$$\text{interval}\, m_{ij}\, \text{optional} \qquad \forall i \in P, j \in J_i, m_{ij} \in M_{ij} \tag{10}$$

$$\text{interval}\, d_{ijm}\, \text{optional} \qquad \forall i \in P, j \in J_i, m \in M_{ij} \tag{11}$$

Since, every activity $j \in J_i$, can be executed in only one of its modes, then:

$$\text{alternative}(j, [m_{ij}]_{\in Mij}) \qquad \forall i \in P, j \in J_i \tag{12}$$

Resources are expressed as cumul-function expressions:

$$\text{cumulFunction}\, c_{il} \qquad \forall i \in P, l \in L_i^{\rho} \tag{13}$$

$$\text{cumulFunction}\, c_g^{\rho} \qquad \forall g \in G^{\rho} \tag{14}$$

$$\text{intExpr}\, h_{il} \qquad \forall i \in P, l \in L_i^{\nu} \tag{15}$$

Feasible schedules of projects must always fulfill following hard constraints:

- For each project $i \in P$ and each local non-renewable resource associated to that $l \in L_i^{\nu}$, total resource consumption does not exceed its capacity $l \leq h_{il}$. Since non-renewable resources have fixed capacities for the whole project duration we modeled them in our model as scalar expressions:

$$\sum_{j \in J_i} \sum_{m_{ij} \in M_{ij}} \text{presenceOf}(m_{ij})({r_{ijm}}_l^{\nu}) \leq h_{il} \quad \forall i \in P, l \in L_i^{\nu} \tag{16}$$

  In MMRCMPSP model every project has its own list of non-renewable resources.
- For each project $i \in P$ and each local renewable resource associated to that $l \in L_i^{\rho}$, total resource consumption does not exceed its capacity $l \leq c_{il}$. As the name implies local renewable resources are dedicated to a specific project and have a fixed capacity per time unit, meaning their capacity constraints have a temporal dimension. Therefore, they are modeled as cumulative functions:

$$\sum_{j \in J_i} \sum_{m_{ij} \in M_{ij}} \text{pulse}(m_{ij}, {r_{ijm}}_l^{\rho}) \leq c_{il} \quad \forall i \in P, l \in L_i^{\rho} \tag{17}$$

  In MMRCMPSP model every project has its own list of renewable resources.
- For each time unit t and each global renewable resource $g \in G^{\rho}$, total resource consumption at t does not exceed its capacity $l \leq c_g$. Global renewable resources are modeled similarly as local renewable resources as cumulative function. There is only one global resources list common to all projects:

$$\sum_{i \in P} \sum_{j \in J_i} \sum_{m_{ij} \in M_{ij}} \text{pulse}(m_{ij}, {r_{ijm}}_g^{\rho}) \leq c_g \qquad \forall g \in G^{\rho} \tag{18}$$

  There is no global non-renewable resources list.
- Feasible schedules must fulfill all precedence constraints between activities:

$$\text{endBeforeStart}(a, j) \qquad \forall i \in P, a, j \in P_i \tag{19}$$

According to definition the main objective function of MMRCMPSP problem consists of minimizing total project delay of projects and the projects total makespan as tie-breaker. It is defined as:

$$f = \min(\max(\text{endOf}(j) + \sum_{i \in P}(\alpha * \max(\text{endOf}(P_i))))), \forall j \in J_i \tag{20}$$

α – is a constant.

4 Description of extended meta-heuristic

Our main aim regarding the hybrid algorithms for MMRCMPSP is to combine two complementary search strategies. Various classifications and taxonomies for hybrid approaches can be found in the literature [17], such as combining meta-heuristics with constraint programming, combining meta-heuristics with exact methods from mathematical programming, combining meta-heuristics with other meta-heuristics or machine learning techniques. We opted for combination of an exact approach, a CP model, with a local search based algorithm that extends the algorithm in [7], which combines min conflicts and tabu search heuristics embedded in an iterated local search framework.

4.1 Solution representation

In our extended meta-heuristic solutions are represented as pair of vectors: $\overrightarrow{S} = \{\overrightarrow{\pi}, \overrightarrow{M}\}$, where $\overrightarrow{\pi}$ represents the vector of all activities from all projects and $\overrightarrow{M}$ is its corresponding modes vector:

$$\overrightarrow{\pi} = \{1, 2, .., |J_1|, |J_1| + 1, .., |J_1| + |J_2| + .. + |J_\text{n}|\} \tag{21}$$

Analogous representations were employed by [4], [6], [5] and [7]. Construction of a solution alternative is done by sequentially assigning activities into a schedule as early as possible.

Algorithm 1 SwapAndModeCh(s) neighborhood operator

```
repeat
  Improved ← false
  act_i ← random.select.from(J_i)
  while not IsSuccessor(s, act_{i+1}) do
    s' ← Swap(s, act_i, act_{i+1})
    for all modes_of_act_{i+1} do
      s'' ← OneModeChange(s', act_{i+1})
      if (eval(s'') < eval(s)) then
        s ← s''
        Improved ← true
      end if
    end for
  end while
until not Improved
```

First, we added a complex neighborhood operator, *SwapAndModeCh*, comprised of two existing simple ones *SwapActivity* and *OneModeChange* noted in [4], [6], [5]. As our CP solver (IBM CP optimizer) also includes a large neighborhood search, we excluded from the implementation of local search four-mode-change (*MinConFMC*) neighborhood operator due to the fact that it generates also large neighborhoods. *SwapAndModeCh* applies swap between an activity and its successor along the schedule until the precedence constraint is not violated. After each swap, activity is assigned each of its modes in turn and if there is an improvement the solution is accepted (algorithm 1).
Figure 1 illustrates the generated neighborhood by this operator assuming that $act_i = 3$, $SuccessorOf(act_i) = 5$ in a given input solution s. Activity 3 swaps with its descendants in the schedule all the way up to its successor, activity 5, with whom it has a precedence constraint.

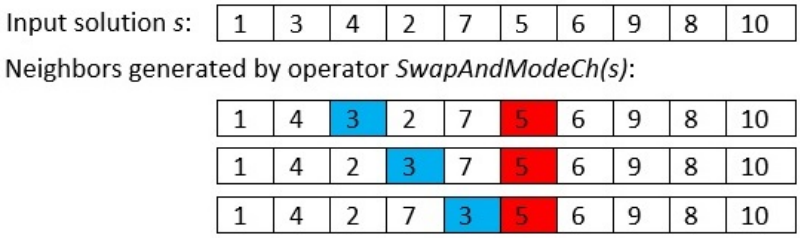

Fig. 1 SwapAndModeCh(s) neighborhood operator

The implementation of improved meta-heuristic is designed for multi-threaded execution environment (algorithm 2).

Algorithm 2 Improved algorithm ILS_Min_Con(s, Time)

for all $\overrightarrow{S}_i$ **do**
  $\overrightarrow{S}_i \leftarrow s$
end for
$\overrightarrow{S}_{best} \leftarrow \emptyset, \quad \overrightarrow{S}_{bestlocal} \leftarrow \emptyset$
$NoImprovement \leftarrow true, \quad LocalImprovement \leftarrow true$
repeat
  repeat
    for all $\overrightarrow{S}_i$ **do**
      $CloneProj(), CloneProjPart(), ComE(), ComF()$
    end for
    if $(BroadcastBestLocal())$ **then**
      $NoImprovement \leftarrow false$
    else
      $NoImprovement \leftarrow true$
    end if
  until NoImprovement
  $Improve \leftarrow true$
  while $Improve$ **do**
    for all $\overrightarrow{S}_i$ **do**
      $MinConOMC()$
    end for
    if $(BroadcastBestLocal())$ **then**
      $LocalImprovement \leftarrow true$
    else
      $Improve \leftarrow false$
    end if
  end while
  if $LocalImprovement$ **then**
    for all $\overrightarrow{S}_i$ **do**
      $PCom(), MinConTMC()$
    end for
  else
    for all $\overrightarrow{S}_i$ **do**
      $SwapAndModeCh(\overrightarrow{S}_i), INVS(), MinConSJL(), MinConSJR()$
    end for
    $BroadcastBestLocal()$
  end if
  if $(\overrightarrow{S}_{bestlocal} < \overrightarrow{S}_{best})$ **then**
    $\overrightarrow{S}_{best} = \overrightarrow{S}_{local}, \quad NoImprovement \leftarrow false$
  else
    $NoImprovement \leftarrow true$
  end if
  $LocalImprovement \leftarrow NoImprovement$
  if $(not \quad LocalImprovement)$ **then**
    $LocalImprovement \leftarrow not \quad LocalImprovement$
    $PerturbationSize \leftarrow PerturbationSize + 1$
  else
    $PerturbationSize \leftarrow 1$
  end if
  for all $\overrightarrow{S}_i$ **do**
    $Perturbate(\overrightarrow{S}_{bestlocal}, PerturbationSize), Reset(\overrightarrow{S}_{bestlocal})$
  end for
until Time

Other neighborhood operators depicted in algorithm 2, one mode change (MinConOMC), two-mode change (MinConTMC), shift an activity to its last predecessor (MinConSJL), shift an activity to its first successor (MinConSJR), invert subsequence of activities (INVS), compress project and move to the end (ComE), compress project and move to the front (ComF), clone a project (CloneProj), clone a project partially (CloneProjPart) and clone a sequence from a project (CloneSeq) are implemented similar to the implementation in [7]. Definition of BroadcastBestLocal() method is depicted in algorithm 3.

Algorithm 3 BroadcastBestLocal()

$minLocal \leftarrow \min_{i \in \{1,..,4\}} eval(\overrightarrow{S}_i)$
$s_{temp} \leftarrow \emptyset$
if $(eval(\overrightarrow{S}_{bestlocal}) > minLocal)$ **then**
  $minLocal = \min_{i \in \{1,..,4\}} eval(\overrightarrow{S}_i)$
  $s_{temp} \leftarrow \overrightarrow{S}_{arg \min_{i \in \{1,..,4\}} \overrightarrow{S}_i}$
  for all $\overrightarrow{S}_i$ **do**
    $\overrightarrow{S}_i \leftarrow s_{temp}$
  end for
  $\overrightarrow{S}_{bestlocal} \leftarrow s_{temp}$
  $return \quad true$
else
  $return \quad false$
end if

4.2 Acceptance criteria and perturbation

Similar to the implementation in [7], we accept only better solutions and implemented adaptive perturbation strategy in relation to instance size. The perturbation consists of changing modes of up to 10 % of randomly selected activities from the schedule.

4.3 Parameter tuning

In this algorithm, two adaptive tabu lists are implemented: one tabu list for operators that manipulate modes and one for operators that manipulate positions of activities in the schedule. The dimensions of these tabu lists are parameterized and experimentally determined as a percentage of the total number of activities in a given instance. Other parameters we fine tuned are: size of variable set for MinConOMC and MinConTMC operators and perturbation size threshold. Parameter values are fine tuned using the SMAC tool ([18], [19]) and obtained are depicted in Table 1.

Table 1 Parameters used for tests

Parameter	Value	Domain of values
ModeTBLength	30%	{10%, 15%, 20%, 25%, 30%, 35%, 40%}
SeqTBLength	20%	{10%, 15%, 20%, 25%, 30%, 35%, 40%}
VarSetSize	11	{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}
PertSizeThreshold	10%	{3%, 5%, 7%, 10%, 13%, 15%, 20%}

5 Hybrid Method

Our hybrid implementation consists of sequential execution of constraint programming model and extended meta-heuristic explained in the previous paragraph. According to [17] it could be classified as high-level relay hybrid (HRH) approach. The output solution of one method serves as initial solution for the other. A perturbation of the best incumbent solution is performed if there was no improvement between every sequential call of algorithms. The perturbation strategy is based on a mode change of an activity and inversion of a short sequence of activities randomly selected from the schedule. It always produces feasible solutions. The perturbed solution serves as a starting point for the CP model. We experimented with allotted execution time for every algorithm in ratio two to one and three to one in favor of CP model within an execution sequence. Slightly better results were obtained with ratio three to one (algorithm 4).
In our hybrid method data reading technicalities for MMRCMPSP and MRCPSP problems are implemented. Proper data structures (interval variable arrays, multi-dimensional cumulative functions arrays, scalar expression arrays, constraints, etc.) for the proper type of problems are generated and populated.

6 Computational results

We performed experiments on thirty benchmark MMRCMPSP problem instances provided in MISTA 2013 challenge. Algorithms presented in this challenge were executed in a computer with a 64-bit Intel Core i7 processor (3.4 GHz) CPU of eight cores, 8 GB RAM. Every algorithm was executed ten times for each instance and five minutes for every run. All of our tests were executed on a machine with 64-bit Intel Core i5 processor (3.3 GHz) CPU of four cores, 8GB RAM. Algorithms are implemented in the programming language C# (Visual Studio 2019 development environment). In order to perform experiments in approximately identical conditions, we determined running time in our machine according to benchmark program (the 64-bit version) of the 2011 International Timetabling Competition. [1] The execution of this program on our computer lasted 690 seconds, while on the computer used in the competition 645 seconds. Therefore, we set the running time on our computer

[1] https://www.utwente.nl/ctit/hstt/itc2011/benchmarking/

Algorithm 4 Hybrid approach for MMRCMPSP combining a CP model and an extended meta-heuristic

```
s ← ∅
s' ← ∅
CP.Add(Projects, Activities, Modes, Resources)
CP.Add(ConstraintsTypes : (7), (8), (9), (16), (17), (18)and(19))
{See equations (7), (8), (9), (16), (17), (18) and (19)}
CP.Add(ObjectiveFunction : equation(20))
CP.SetParameters(Time * 3, SearchType, RandomSeed)
while not Termination_Condition do
   CP.Solve()
   s ← CP.ExtractSolution()
   s'' ← ILS_Min_Con(s, Time)
   if (eval(s'') < eval(s')) then
      s' ← s''
   else
      act_i ← random.select.from(J_i)
      s ← RandomOneModeChange(s', act_i)
      s ← RandomInvertSubSequence(s)
   end if
   sp ← CP.Solution()
   sp.Set(s)
   CP.SetStartingPoint(sp)
end while
```

to 321 seconds, accordingly. Additional experiments were accomplished under different settings with and without time restrictions.

6.1 Benchmark instances

Every set of benchmark instances (A, B and X) is comprised of ten instances. Sizes of instances vary from smallest one comprised of 2 projects and 20 activities up to the biggest one with 20 projects and 600 activities. Even though, we were focused on solving MMRCMPSP problem instances, we tested our algorithms on solving multi-mode resource-constrained project scheduling problem instances introduced in [15] as well. New upper bounds were obtained for many instances from each set of problems.

6.2 Evaluation of the hybrid method

Our hybrid approach outperformed the best solver implemented in [4] for MMRCMPSP problem when running algorithms under no-restriction conditions. It generated new upper bounds for the majority of instances of this group, (Table 2). Solver presented in [4] ran 2500 times for each instance and 5 minutes for every run. We executed our solver for 24 hours only once for each instance. Many of the instances converged much earlier, within a few minutes. Comparison of results of our hybrid method with those of the solver imple-

Table 2 Comparison of our hybrid approach's results (TPD/TMS) with the best solvers results so far for MMRCMPSP problem

Inst.	[4]	[5]	[6]	Our algorithm	New upper bounds
A1	1/23	1/23	1/23	1/23	Equal
A2	2/41	2/41	2/41	2/41	Equal
A3	0/50	0/50	0/50	0/50	Equal
A4	65/42	65/42	68/50	65/42	Equal
A5	150/103	153/104	154/104	151/104	
A6	133/99	144/94	151/94	**132/90**	Yes
A7	590/190	601/206	626/194	595/189	
A8	272/148	319/162	281/147	**257/147**	Yes
A9	197/122	225/128	212/127	**186/122**	Yes
A10	836/303	920/313	983/309	854/307	
B1	345/124	349/130	358/131	348/127	
B2	431/158	481/171	431/159	**404/160**	Yes
B3	526/200	604/214	585/196	**515/204**	Yes
B4	1252/275	1283/287	1435/294	1296/283	
B5	807/245	866/252	867/254	813/250	
B6	905/225	1067/246	970/224	**888/219**	Yes
B7	782/225	827/232	876/234	800/233	
B8	3048/523	3618/565	3001/520	**2871/525**	Yes
B9	4062/738	4606/783	4753/741	4093/736	
B10	3140/436	3541/473	3123/430	**3057/437**	Yes
X1	386/137	-	392/142	**385/139**	Yes
X2	345/158	-	416/167	**342/163**	Yes
X3	310/187	-	332/177	**287/183**	Yes
X4	907/201	-	980/209	**896/204**	Yes
X5	1727/362	-	1904/369	1757/370	
X6	690/226	-	821/237	700/232	
X7	831/220	-	909/232	854/224	
X8	1201/279	-	1389/281	**1188/279**	Yes
X9	3155/632	-	3945/639	3269/641	
X10	1573/373	-	1718/377	**1572/374**	Yes

mented by [4] are also given in Figure 2, where differences between results for every benchmark instance of both solvers are visually presented.

We also evaluated our solver on MMRCSPSP instances and experiments resulted with new upper bounds for fifty instances and equal results for many more compared to results of state-of-the-art solvers (Table 3). In experiments run under time restriction conditions, i.e. 10 runs per instance and 5 minutes per each run, our hybrid solver generated best results for nine and equal results for four benchmark instances (Table 4). Average results and standard deviations are reported, too. Differences between results of our solver and best solver [4] under these restrictions are presented graphically in Figure 3. It can be noticed that for very large instances, e.g. B9 and X9, hyper-heuristic approach performs better compared to our hybrid method. This is due to the nature of the CP model that is a constituent part of the hybrid method, it performs very well for small and medium instances.

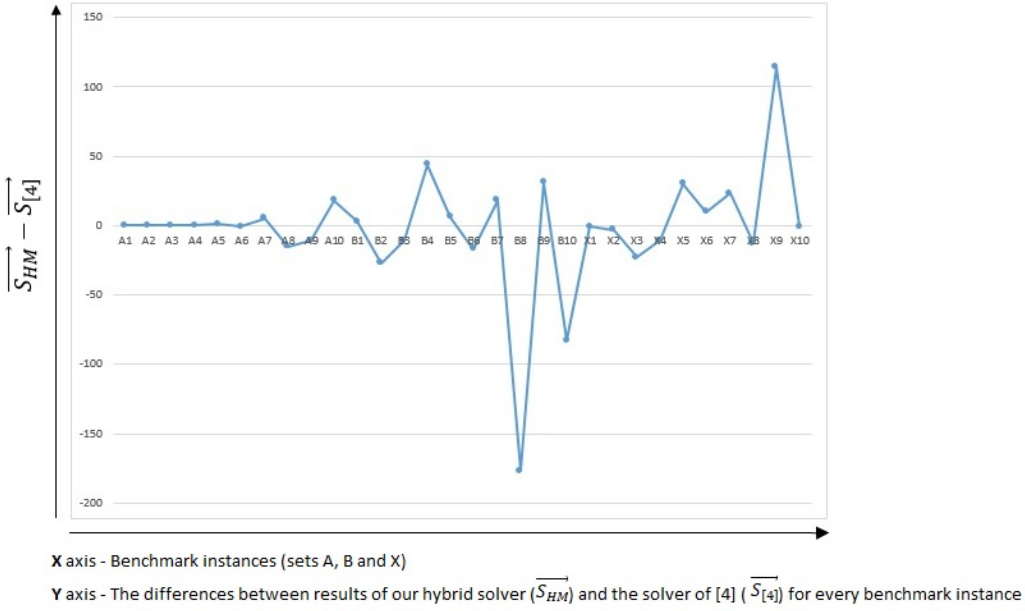

Fig. 2 Comparison of our results (TPD/TMS) with the best solver's results ([4]), under no time restriction conditions for MMRCMPSP problem

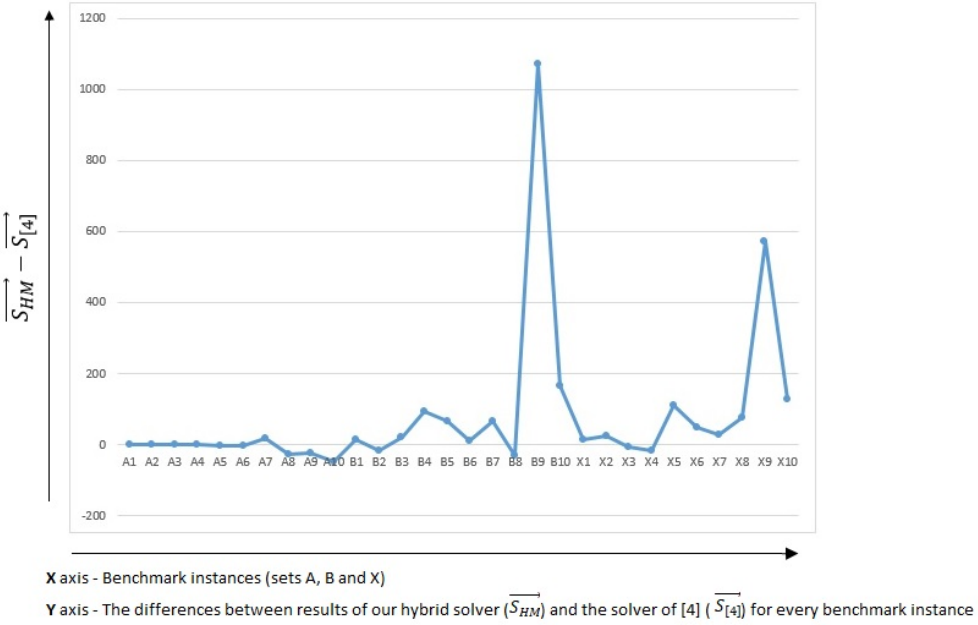

Fig. 3 Comparison of our results (TPD/TMS) with the best solver's results ([4]), under time restriction conditions for MMRCMPSP problem

Table 3 New upper bounds achieved by the hybrid method for MRCPSP problem

MMLIB Instances					
mmlib50		**mmlib100**		**mmlibPlus**	
Inst.	**Makespan**	**Inst.**	**Makespan**	**Inst.**	**Makespan**
J507_1.mm	43	J1008_5.mm	49	Jall146_1.mm	64
J507_5.mm	40	J10045_3.mm	53	Jall185_3.mm	107
J5043_5.mm	63	J10074_4.mm	75	Jall193_3.mm	75
J5045_4.mm	35	J10076_3.mm	58	Jall254_3.mm	81
J5046_5.mm	38	J10079_1.mm	128	Jall256_3.mm	113
J5047_5.mm	38	J10080_5.mm	83	Jall302_1.mm	47
J5048_2.mm	39	J10081_1.mm	85	Jall344_4.mm	83
J5080_5.mm	69	J10081_2.mm	74	Jall346_2.mm	78
-	-	J10081_3.mm	70	Jall347_3.mm	57
-	-	J10082_3.mm	78	Jall363_1.mm	57
-	-	J10082_4.mm	94	Jall371_1.mm	84
-	-	J10083_2.mm	79	Jall372_3.mm	72
-	-	J10083_3.mm	71	Jall374_4.mm	66
-	-	J10084_3.mm	78	Jall394_2.mm	102
-	-	J10084_5.mm	73	Jall399_1.mm	237
-	-	J10092_2.mm	80	Jall401_2.mm	193
-	-	J10092_5.mm	71	Jall410_5.mm	62
-	-	J10094_1.mm	54	Jall458_3.mm	79
-	-	-	-	Jall509_1.mm	138
-	-	-	-	Jall512_1.mm	132
-	-	-	-	Jall537_3.mm	133
-	-	-	-	Jall537_4.mm	104
-	-	-	-	Jall556_2.mm	99
-	-	-	-	Jall566_5.mm	124

6.3 Evaluation of the extended meta-heuristic

We have accomplished separate tests with extended meta-heuristic and our CP model under time restriction conditions. It turned out that the extended meta-heuristic slightly improved most of the MMRCMPSP results (the best results, average and standard deviation) in comparison with the implementation in [7] (Table 6). According to [7] their solver is competitive to the third ranked solver under time restrictions. Meta-heuristic approach provided better results in nine and equal in three instances compared to our CP model and none compared to hybrid method.
Additionally, we tested our extended meta-heuristic with MMRCPSP problem benchmark instances. Tests resulted in the achievement of new upper bounds for five instances (Table 5).

Table 4 Comparison of our results (TPD/TMS) with the best solver's results ([4]), under time restriction conditions for MMRCMPSP problem

	[4]			Hybrid approach		
Inst.	**Best**	**Avg**	**Std**	**Best**	**Avg**	**Std**
A1	1/23	-	-	1/23	1/23	0
A2	2/41	-	-	2/41	2/41	0
A3	0/50	-	-	0/50	0/50	0
A4	65/42	-	-	65/42	65/44	0/2
A5	153/105	-	-	**152/104**	165/106	8/2
A6	147/96	-	-	**144/92**	160/100	5/3
A7	596/196	-	-	615/205	639/201	2/5
A8	302/155	-	-	**276/150**	291/150	11/3
A9	223/119	-	-	**200/121**	215/128	11/3
A10	969/314	-	-	**921/313**	982/325	41/7
B1	349/127	352/128	-	364/126	379/128	11/2
B2	434/160	454/168	-	**419/162**	461/164	21/2
B3	545/210	554/211	-	566/212	601/212	19/1
B4	1274/289	1305/284	-	1368/291	1464/291	65/6
B5	820/254	833/254	-	887/262	923/262	18/4
B6	912/227	953/232	-	925/233	992/236	38/4
B7	792/228	801/232	-	859/239	921/244	49/5
B8	3176/533	3314/548	-	**3145/561**	3511/568	179/14
B9	4192/746	4264/755	-	5262/884	5740/922	320/30
B10	3249/456	3338/460	-	3415/473	3583/469	137/6
X1	392/142	405/142	-	408/142	432/145	21/2
X2	349/163	357/164	-	375/167	402/170	20/4
X3	324/192	330/193	-	**318/187**	346/190	19/4
X4	955/213	971/212	-	**939/210**	1033/209	53/2
X5	1768/374	1785/373	-	1878/386	1956/388	43/5
X6	719/232	738/241	-	768/240	840/249	72/8
X7	861/237	868/236	-	890/235	925/239	24/5
X8	1233/283	1257/289	-	1310/287	1452/300	92/8
X9	3268/643	3303/647	-	3840/718	4134/750	163/15
X10	1600/381	1614/382	-	1727/403	1777/403	36/5

Table 5 New upper bounds achieved by meta-heuristic for MMRCPSP problem

	Project total makespan new upper bounds	
Instance	**Our algorithm**	**Different authors**
Jall127_3.mm	142	143
Jall128_5.mm	94	95
Jall184_1.mm	177	179
Jall263_4.mm	146	147
Jall289_5.mm	196	197

6.4 Evaluation of CP Model

We also performed tests with our CP model under time restriction conditions. According to [16] the search in a CP model can be directed through configuration of several parameters, e.g. search type, random seed, time limit, etc. Results of the CP model depicted in Table 6 for MMRCMPSP instances

are performed with *Search Type = Restart*, *Random Seed = 1292619981* and *Time Limit = 300 s*. In restart search mode the algorithm restarts and executes depth first search after a parameterized number of failures. Another search type we have experimented with is automatic search that employs large neighborhood search and failure directed search [16]. The former one tries to converge quickly to a good quality solution and the latter one tries to prove that no better solution exists than the existing one when the search space is too small or LNS cannot improve further the solution. CP model uses random seed parameter for tie-breaking situations only. In Table 6 it can be seen that the CP model provided slightly better results for three instances compared to hybrid method and three compared to [4].
The hybrid approach seems to be far more successful than executing the constituent algorithms separately. In the case of very large instances, under short time restriction conditions, e.g. MISTA challenge conditions, meta-heuristic approaches appear to be slightly better. Under no time restriction conditions, the hybrid approach outperforms all solvers.

7 Conclusions and future work

In this paper we investigated three approaches for MMRCMPSP: a CP model, an extended meta-heuristic and a hybrid approach combining the first two solvers. We performed separate experiments with all solvers on existing MMRCMPSP problem benchmark instances and compared to the state-of-the-art solver for this problem. It turned out that the meta-heuristic approach provided better results in nine instances compared to our CP model and none compared to the hybrid method. CP model provided better results for three instances compared to the hybrid method and three compared to one of the best solvers ([4]) for restricted time conditions. Regarding the hybrid approach, we performed extensive tests in various experimental settings. It outperformed best existing solver for MMRCMPSP in several instances and achieved new upper bounds for fourteen out of thirty instances under no time restrictions conditions. Under time restriction, the model generated best results for nine and equal results for four benchmark instances. For some very large instances, the best existing solver ([4]) provided better results. Additionally, we tested our approach with multiple-mode resource constrained single project scheduling problems well-known instances and experiments resulted with new upper bounds for fifty instances.
Our main objective was to evaluate a hybrid approach that combines two complementary search strategies. In this study, we introduce a successful combination of an exact model with a meta-heuristic approach and a certain perturbation mechanism. We consider that it is of high interest to investigate further hybrid approaches that combine different search strategies in order to create a robust and efficient method. Especially, the role of meta-heuristics in a hybrid method should be further studied. In our case, we noticed that for very large instances (B9 and X9), under short time restrictions, hyper-

Table 6 Comparison of extended meta-heuristics results (TPD/TMS) with the best solvers results for MMRCMPSP problem under time restriction conditions

Ins.	**Extended meta heuristic**	[7]	CP Model	hybrid method	[4]	[5]	[6]
A1	1/23	1/23	1/23	1/23	1/23	-	-
A2	2/41	2/41	2/41	2/41	2/41	-	-
A3	0/50	0/50	0/50	0/50	0/50	-	-
A4	65/42	65/45	65/45	65/42	65/42	-	-
A5	162/107	163/108	173/110	**152/104**	153/105	-	-
A6	156/94	152/96	162/104	**144/92**	147/96	-	-
A7	644/203	652/208	655/197	615/205	596/196	-	-
A8	337/166	335/163	279/148	**276/150**	302/155	-	-
A9	245/139	253/137	212/124	**200/121**	223/119	-	-
A10	969/338	980/331	1017/317	**921/313**	969/314	-	-
B1	355/129	361/129	375/130	364/126	349/127	353/125	363/132
B2	498/177	502/180	438/167	**419/162**	434/160	490/176	434/160
B3	639/230	637/224	586/206	566/212	545/210	598/215	660/207
B4	1386/302	1415/290	1493/286	1368/291	1274/289	1274/289	1548/295
B5	955/275	927/268	922/267	887/262	820/254	866/254	919/254
B6	1139/261	1146/253	936/227	925/233	912/227	1044/242	1128/232
B7	890/251	864/249	1003/252	859/239	792/228	834/234	908/246
B8	3687/628	3836/626	3113/544	**3145/561**	3176/533	3585/568	3276/529
B9	5858/948	5757/926	5253/833	5262/884	4192/746	4674/796	5373/769
B10	3636/456	3654/514	3295/455	3415/473	3249/456	3518/469	3325/447
X1	435/148	427/150	443/144	408/142	398/142	394/142	392/142
X2	419/175	408/174	405/167	375/167	349/163	368/165	418/165
X3	382/202	407/206	346/194	**318/187**	324/192	372/195	326/188
X4	1035/221	1081/221	996/208	**939/210**	955/213	970/215	986/207
X5	2083/393	2089/428	1940/376	1878/386	1768/374	1938386	2043/375
X6	967/281	953/284	799/243	768/240	719/232	844/253	880/240
X7	951/245	968/248	902/233	890/235	861/237	879231	944/234
X8	1584/329	1515/324	1366/288	1310/287	1233/283	1380/296	1478/289
X9	4374/790	4167/776	4320/760	3840/718	3268/643	3645/688	4169/662
X10	1938/437	1934/427	1739/396	1727/403	1600/381	1669/402	1851/385

heuristic approach still performed better than our hybrid approach. Since the exact model relies on large neighborhood search and performs very well on small and medium instances than it is up to meta-heuristics to perform efficient search within very short time conditions in order to provide excellent results even in the case of very large instances. Furthermore, improving the complementary search nature of algorithms that comprise a potential hybrid method would be of valuable interest.

References

1. Jacek Blazewicz, Jan Karel Lenstra, and Alexander H.G. Rinnooy Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11–24, 1983.

2. Rainer Kolisch and Arno Sprecher. Psplib-a project scheduling problem library: Or software-orsep operations research software exchange program. *European journal of operational research*, 96(1):205–216, 1997.
3. Tony Wauters, Joris Kinable, Pieter Smet, Wim Vancroonenburg, Greet Vanden Berghe, and Jannes Verstichel. The multi-mode resource-constrained multi-project scheduling problem. *Journal of Scheduling*, 19(3):271–283, 2016.
4. Shahriar Asta, Daniel Karapetyan, Ahmed Kheiri, Ender Özcan, and Andrew J. Parkes. Combining monte-carlo and hyper-heuristic methods for the multi-mode resource-constrained multi-project scheduling problem. *Information Sciences*, 373:476–498, 2016.
5. Martin Josef Geiger. A multi-threaded local search algorithm and computer implementation for the multi-mode, resource-constrained multi-project scheduling problem. *European Journal of Operational Research*, 256(3):729–741, 2017.
6. Túlio A.M. Toffolo, Haroldo G. Santos, Marco A.M. Carvalho, and Janniele A. Soares. An integer programming approach to the multimode resource-constrained multiproject scheduling problem. *Journal of Scheduling*, 19(3):295–307, 2016.
7. Arben Ahmeti and Nysret Musliu. Min-conflicts heuristic for multi-mode resource-constrained projects scheduling. In Hernán E. Aguirre and Keiki Takadama, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2018, Kyoto, Japan, July 15-19, 2018*, pages 237–244. ACM, 2018.
8. Umut Beşikci, Ümit Bilge, and Gündüz Ulusoy. Multi-mode resource constrained multi-project scheduling and resource portfolio problem. *European Journal of Operational Research*, 240(1):22–31, 2015.
9. Leonardo M. Borba, Alexander J. Benavides, Tadeu Zubaran, Germano C. Carniel, and Marcus Ritt. A simple stochastic local search for multi-mode resource-constrained multi-project scheduling. In *Proceedings of the 6th Multidisciplinary International Scheduling Conference*, pages 826–830, 2013.
10. Mathias Kühn, Taiba Zahid, Michael Völker, Zhugen Zhou, and Oliver Rose. Investigation of genetic operators and priority heuristics for simulation based optimization of multi-mode resource constrained multi-project scheduling problems (mmrcmpsp). In *ECMS*, 2016.
11. Emmanuel Hebrard Christian Artigues. Mip relaxation and large neighborhood search for a multi-mode resource-constrained multi-project scheduling problem. In *Proceedings of the 6th Multidisciplinary International Scheduling Conference*, pages 815–819, 2013.
12. Federico Alonso-Pecina, Johnatan E. Pecero, and David Romero. A three-phases based algorithm for the multi-mode resource-constrained multi-project scheduling problem. In *Proceedings of the 6th Multidisciplinary International Scheduling Conference*, pages 812–814, 2013.
13. Hermann Bouly, Duc-Cuong Dang, Aziz Moukrim, and Huang Xu. Evolutionary algorithm and post-processing to minimize the total delay in multi-project scheduling. In *Proceedings of the 6th Multidisciplinary International Scheduling Conference*, pages 831–835, 2013.
14. Manuel López-Ibánez, Franco Mascia, Marie-Eléonore Marmion, and Thomas Stützle. Automatic design of a hybrid iterated local search for the multi-mode resource-constrained multi-project scheduling problem. In *Proceedings of the 6th Multidisciplinary International Scheduling Conference*, pages 820–825, 2013.
15. Vincent Van Peteghem and Mario Vanhoucke. 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, 2014.
16. Philippe Laborie, Jérôme Rogerie, Paul Shaw, and Petr Vilím. Ibm ilog cp optimizer for scheduling. *Constraints*, 23(2):210–250, April 2018.
17. El-Ghazali Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009.
18. Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization - 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers*, pages 507–523, 2011.
19. Marius Lindauer, Katharina Eggensperger, Matthias Feurer, Stefan Falkner, Andre Biedenkapp, and Frank Hutter. Smac v3: Algorithm configuration in python. `https://github.com/automl/SMAC3`, 2017.