A simple stochastic local search for multi-mode resource-constrained multi-project scheduling

Leonardo M. Borba · Alexander J. Benavides · Tadeu Zubaran · Germano M. Carniel · Marcus Ritt

Abstract In the multi-mode resource-constrained multi-project scheduling (MMRCMPSP) problem we have to find execution modes and starting times for the jobs of multiple projects, such that the total project delay is minimized. The delay of a project is the difference between its makespan and a lower bound on its duration. A secondary objective is to minimize the total makespan of all projects. The jobs of a project compete for renewable and non-renewable local resources, and for global renewable resources shared among the projects. The duration and resource requirements of a job depend on its execution mode.

We present a simple stochastic local search method for this problem. It maintains a feasible permutation and mode selection of the jobs, and improves it by swapping the order of two jobs, or changing the mode of a job. We propose several techniques to reduce size of the analyzed neighborhoods. The method has ranked fourth during the qualification phase of the MISTA challenge, and was qualified for the second phase. We present computational experiments on the twenty instances of the challenge and compare them to the results of the qualification phase.

1 Introduction

In the MMRCMPSP (multi-mode resource-constrained multi-project scheduling problem) we have to find a common schedule for multiple projects. Each project consists of several jobs which have to be scheduled in a given partial order after the projects’ release time. Each job can choose among multiple execution modes of different processing time and resource requirements. Resources may be local to a project, or global, i.e. shared among the projects. Local resources may be renewable, with a fixed capacity per time, or non-renewable, with a given total capacity. Global resources are always renewable.

Formally, let $P = \{n\}$ be the set projects, $J_i = \{m_i\}$ the set of jobs, and $r_i$ the release time of project $i \in P$. The jobs in $J_i$ have to satisfy a partial order $\preceq$ on $J_i$. Each project $i \in P$ has $\rho_i$ local renewable resources of capacity $c_{ij}, j \in [\rho_i]$ and $\rho_i$ local non-renewable resources of capacity $c_{ij}, j \in [\rho_i]$. Further, there are $\rho$ global renewable resources of capacity $c_j, j \in [\rho]$. A job $j \in J_i$ has $\mu_{ij}$ execution modes. In execution mode $m \in [\mu_{ij}]$ it has processing time
A solution of the MMRCMPSP is a selection of starting times $s_{ij}$ and modes $m_{ij} \in [\mu_{ij}]$, $i \in \mathcal{P}$, $j \in \mathcal{J}_i$. Let $f_{ij} = s_{ij} + p_{ijm_{ij}}$ be the finishing time of job $j \in \mathcal{J}_i$ in project $i \in \mathcal{P}$, and

$$R_{irt} = \sum_{i \in \mathcal{P}} \sum_{t \in [s_{ij}, f_{ij}]} r_{ijm_{ij}} r, \quad \forall i \in \mathcal{P}, r \in [\rho], t \in [0, T],$$

be the local renewable resource of resource $r$ in project $i$ at time $t$, and the global renewable requirements of resource $r$ at time $t$, respectively. Let further

$$R_{ir} = \sum_{i \in \mathcal{P}} \sum_{t \in [s_{ij}, f_{ij}]} r_{ijm_{ij}} r, \quad \forall i \in \mathcal{P}, r \in [\rho]$$

be the local non-renewable requirement of resource $r$ in project $i$. Then, a valid solution satisfies the precedence constraints

$$f_{ij} \leq s_{ij'}, \quad \forall i \in \mathcal{P}, j, j' \in \mathcal{J}_i, j \preceq j',$$

the release time constraint

$$r_i \leq s_{ij}, \quad \forall i \in \mathcal{P}, j \in \mathcal{J}_i,$$

and the resource constraints

$$R_{irt} \leq c_{ir}, \quad \forall i \in \mathcal{P}, r \in [\rho], t \in [0, T],$$

$$R_{ir} \leq c_{ir}, \quad \forall i \in \mathcal{P}, r \in [\rho],$$

for some upper bound on the total time horizon of any valid schedule $T$ (e.g. $T = \max_{i \in \mathcal{P}} r_i + \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{J}_i} \max_{m \in [\mu_{ij}]} p_{ijm}$).

The objective of the MMRCMPSP is to minimize the total project delay $TPD = \sum_{i \in \mathcal{P}} PD_i$. The delay $PD_i$ of project $i$ is the difference between its makespan $MS_i = f_i - r_i$ and its critical path duration $CP_i = \max_{j \in \mathcal{J}_i} e_{ij}$ for earliest finishing times

$$e_{ij} = \min_{m \in [\mu_{ij}]} p_{ijm} + \max_{j' \in \mathcal{J}_i, j' \preceq j} e_{ij'},$$

(here we take the maximum of the empty set to be 0). If there are several solutions of the same total project delay, the total makespan $TMS = \max_{i \in \mathcal{P}} f_i - \min_{i \in \mathcal{P}} r_i$ serves as a tie breaker.

2 A stochastic local search for the MMRCMPSP

We propose a simple stochastic local search for the MMRCMPSP. It pre-processes the problem, creates an initial solution by solving each project individually and merging these solutions into a single solution for the whole problem, applies a local search and two variants of an (iterated) stochastic local search in parallel to improve the solution. In the following, these steps are described in more detail.
2.1 Pre-processing

In a pre-processing step dominated and unsatisfiable modes are removed. A mode is dominated if there is another mode whose execution time is not longer and whose resource requirements are not larger. A mode is unsatisfiable if some renewable resource requirement exceeds the capacity. For the instance set of the challenge the pre-processing has a limited effect: there are no dominated modes, and only four instances, namely A4, B1, B4, and B7, have between one to nine unsatisfiable modes.

2.2 Representation and evaluation

A solution to the MMRCMPSP is represented by a permutation of the jobs and a mode selection for each job. From this, a complete, feasible solution is constructed by processing each job in the order of the permutation, starting it at the earliest time after the termination of all its predecessors which is able to satisfy its resource requirements during its execution. For the construction we use a data structure that stores only time points at which the capacity of some renewable resource changes, and maintains them ordered by time. The construction needs time $O(m^2k)$ and space $O(mk)$ for a project with $m$ jobs and $k$ renewable resources.

2.3 Initial solution

We guarantee during the search that every solution representation leads to a valid solution of the problem. This is achieved by permitting only permutations that satisfy the precedence constraints (1) and mode selections that satisfy the non-renewable resource constraints (4).

An initial feasible solution satisfying constraints (1) and (4) is obtained as follows. The permutation is computed by a topological sorting, which chooses in every step a random minimal element of the remaining (not yet scheduled) jobs. Of all mode selections which satisfy (4) we choose one of shortest total processing time. This mode selection is found by dynamic programming: For a project with $m$ jobs and $\rho$ non-renewable resources, let $S(i, c)$ be the smallest total processing time of jobs $i, i+1, \ldots, m$ given non-renewable resources $c \in \mathbb{Z}^\rho$. Let $\mu_i$ be the number of modes of job $i$, $p_{ik}$ its processing time in mode $k$ and $r_{ik} \in \mathbb{Z}^\rho$ its non-renewable resource requirements in mode $k$. The shortest total processing times satisfy the recurrence

$$S(i, c) = \begin{cases} 0 & \text{if } i > m \\ \infty & \text{if } r_{ik} \not\leq c \text{ for all } k \in [\mu_i] \\ p_{ik} + S(i + 1, c - r_i) & \text{otherwise, where } k = \arg\min_{k' \in [\mu_i]} r_{ik'} \leq c p_{ik'} \end{cases},$$

and the smallest total processing time is $S(1, c)$ for capacities $c$. The recurrence can be computed by dynamic programming in time $O(m\rho \max_{i \in [m]} \mu_i \prod_{i \in [\rho]} c_i)$ and space $O(m \prod_{i \in [\rho]} c_i)$.

2.4 Local search

Our local search uses two types of moves: a swap move, that interchanges two jobs of the current permutation, and a mode move, that changes the mode of some job. To maintain the feasibility of the solution, as described above, only moves that satisfy the precedence
Table 1  Parameter settings used in the computational experiments.

<table>
<thead>
<tr>
<th>Parameter settings used in the computational experiments.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample size $s$ in sample stochastic local search</td>
</tr>
<tr>
<td>Maximum rank difference $\delta$ for swap moves</td>
</tr>
<tr>
<td>Number of trials $T$ until convergence in stochastic local search</td>
</tr>
<tr>
<td>Number of trials $T_0$ that accept only improving moves</td>
</tr>
</tbody>
</table>

constraints (1) and non-renewable resource constraints (4), are permitted. For the current solution we maintain the minimal and maximal permissible rank of each job in the permutation, as well as the total consumption of non-renewable resources. This allows to check the validity of a swap move in time $O(1)$ and that of a mode move in time $O(\pi)$. To limit the size of the neighborhood, swap moves are only considered between jobs whose rank in the current permutation differs by at most $\delta$.

We use three local search algorithms in our approach. The basic first improvement local search visits systematically all swap and mode moves, and repeatedly accepts the first improving move, or stops if no such move exists. The stochastic local search repeatedly chooses a random move, and checks if it is acceptable. It stops after at most $T$ unsuccessful trials. A move is accepted if it improves the current solution during the first $T_0 \leq T$ trials, or if it does not worsen the solution in the remaining trials. Finally, the sample stochastic local search chooses a sample of $s$ of random moves without replacement, and processes them in order of non-decreasing change in objective function value. The acceptance and stopping criteria are as in the stochastic local search. The sample is maintained during several iterations, until all moves have been consumed, and then recreated with moves from the current solution.

The complete search proceeds in three steps. In the first step each project of the initial solution is optimized individually by applying the sample stochastic local search. Next, the first improvement local search is applied. Finally, the search continues with one of the two stochastic local searches. If one these terminates before the time limit, a random permutation satisfying (1) is generated, and the search continues from this solution, maintaining the current mode selection. The final version uses multiple threads, where half of the threads executes to stochastic local search in the third step, and the other half the sample stochastic local search.

3 Computational results

We implemented our heuristic in C++ and evaluated it on a PC with an Intel Core i7 930 processor running at 2.8 GHz and with 12 GB of main memory. We used the two sets of 10 instances (A and B) provided during the challenge. More information on the instances is available in Kinable, Smet, Vancroonenburg, Berghe, Verstichel, and Wauters [1]. After some preliminary experimentation, we chose the parameters shown in Table 1.

Table 2 presents the results of twelve replications with different random seeds on a single processor. For both variants of the stochastic local search we present the mean objective function value, its standard deviation, the best value found in all 12 runs, and the number of repetitions of the local search. Best values are highlighted in bold. We can see that the two variants obtain similar results. Both are reasonably robust, having small standard deviations. Both find the best known values from the qualification phase only for the smallest three instances, and have relative deviations from about 7% to 40% on the remaining instances. However, these values improved over the qualification phase, and would rank the method
fourth or better, showing a great variance among the competitors. We can also observe that for the larger instances, both stochastic local searches make very few repetitions, and in some cases do not even converge during the time limit of 300s.

4 Conclusions

We have proposed a heuristic for the MMRCMPSP based on simple stochastic local searches, combined with some techniques to reduce the effort of the neighborhood search. The method turned out to be competitive in the qualification, and has been improved further. We believe the short time limit of 300s favours simple methods, and our approach may be competitive in the final round. Its main limitation is the slow convergence, which could be improved by more directed methods of overcoming (non-strict) local minima.

References