MISTA 2013

Automatic Design of a Hybrid Iterated Local Search for the Multi-Mode Resource-Constrained Multi-Project Scheduling Problem

Manuel López-Ibáñez · Franco Mascia · Marie-Eléonore Marmion · Thomas Stützle

1 Introduction

This paper details our submission to the MISTA 2013 challenge, which deals with the multi-mode resource-constrained multi-project scheduling problem (MRCMPSP). Kolisch and Hartmann [4] recommend metaheuristics as the best-performing methods when tackling the resource-constrained project scheduling problem with a single project and without multiple-modes. Most of these metaheuristics share many similar components, such as neighborhoods used in local search, schedule generation procedures, etc. Given the number of possible combinations of algorithmic components and the various ways in which simple metaheuristics can be combined into a single algorithm, finding the right combination for the MRCMPSP would be, in principle, an arduous task of experimentation by trial-and-error.

Instead, we used a recent automatic method [10], which combines (*i*) a description (given as a grammar) of the space of potentially valid algorithms for a problem and (*ii*) a method for searching the best algorithm by instantiating algorithms from this grammar. The approach shares some similarities with genetic programming (GP) [11] and grammatical evolution (GE) [3], yet there are crucial differences. First, GP/GE typically attempt to generate programs from very basic components, whereas our approach relies on humans to provide problem-specific components for the particular problem. Second, GP/GE use a tree-based and a codon-based representation, respectively, to instantiate programs from the grammar. Instead, we use a parametric representation, that is, the grammar description is transformed into a number of categorical and numerical parameters, and some of them might be only enabled depending on other parameters (conditional parameters). This transformation requires to specify the maximum number of times that each rule in the grammar can be applied (similar restrictions exist with other representations, given the existence of recursive derivation rules). Finally, GP/GE use evolutionary algorithms to search for the best instantiation of the grammar. Our approach, by contrast, relies on irace [6], an automatic configuration tool typically used for offline parameter tuning. The characteristics of irace makes it ideal to handle complex parameter spaces, with categorical, numerical and conditional parameters. Moreover, irace is designed for handling heterogeneous problem instances.

Manuel López-Ibáñez, Franco Mascia, Marie-Eléonore Marmion, and Thomas Stützle
IRIDIA, CoDE, Université Libre de Bruxelles (ULB), Brussels, Belgium
E-mail: {mmarmion;fmascia;manuel.lopez-ibanez;stuetzle}@ulb.ac.be

2 Automatic Bottom-up Design of Hybrid ILS Algorithms

The grammar (in Backus-Naur Form) used in this work is divided into a problem-independent part and a problem-specific part. Following previous work [9], the problem-independent grammar (Fig. 1) generates hybrid iterated local search (ILS) algorithms. The main algorithm starts with a problem-specific initialization followed by an ILS. Each ILS is described by four components, a perturbation, a subsidiary local search (LS), an acceptance criterion and a time ratio for the subsidiary LS. The subsidiary LS may be either an ILS, a simulated annealing (SA) or an iterative descent LS using a problem-specific neighborhood. The iterative descent may be either a best-improvement one (*BestImpr*), a first-improvement (*FirstImpr*) one, or a first-improvement descent that continues the search at the same position in the neighborhood after moving to an improving neighbor (*FirstImprCont*). The acceptance criterion determines the solution that replaces the current solution in the ILS. All acceptance criteria accept a better solution but they may also accept a solution with equal or worse objective value in some circumstances. The ones used here are well-known acceptance criteria, their precise definition can be found in our previous work [9]. Some of them have extra parameters. For example, *Cooling* accepts solutions according to the Metropolis criterion, that is, by biasing a probabilistic decision according to a temperature parameter that is adjusted according to a cooling schedule. Finally, each call to the subsidiary LS is given a time limit $t_{\text{subls}} = t_{\text{ratio}} \cdot t_{\text{ls}}$, where t_{ls} is the time limit of the caller and t_{ratio} is a parameter. Each LS is also limited by the maximum time assigned to the whole algorithm.

In order to generate an algorithm for the MRCMPSP from this grammar, we need to implement the required problem-specific components. In our implementation, the representation of schedules is: (i) a standard permutation of the activities, where each activity always appears later than any of its predecessors and earlier than any of its successors, and (ii) a vector of modes, where each entry is the execution mode assigned to each activity. We also apply a well-known pre-processing step in order to reduce the search space [12]. Other MRCMPSP-specific components of our algorithm are configurable, and, thus, they are shown in the grammar of Fig. 2.

Evaluating a solution requires applying a schedule generation (SG) procedure. Unless specified otherwise in the grammar, we use the forward serial SG for a precedence feasible activity list [2]. This procedure is faster than the classical serial SG but produces semi-active schedules instead of active ones [13]. In some specific cases, the SG used is configurable `<ps:sg>`: *ForwardSerial* applies the forward serial SG, *Forward-BacwardsSerial* applies the same procedure forward and backwards and keeps the best solution found, and *multi-mode backward-forward* (MM-BF) generates a schedule using the forward serial SG and then applies MM-BF [8] until no improvement is found.

Initialization constructs one or more solutions heuristically and keeps the best solution constructed. Each solution is constructed by a parallel SG (c.f. Brooks's parallel method [1]) using a heuristic for selecting activities and another for selecting modes. The constructed solutions are re-scheduled using one of the three SGs in the hope that they can be further improved.

The heuristics for selecting activities include the well-known RANDOM-ACTIVITY, the static late finish time (MIN-LFT), static feasible slack (MIN-FSLK), dynamic feasible slack (MIN-DFSLK), and shortest activity from shortest project (SASP) [5]. In addition, MAXTWK-EST, MAXTWK-LST, TWK-LST, TWK-EST and TWK-GDR all use the concept of total work content (TWK) [5]. Both MAXTWK-EST and MAXTWK-LST are applied to all schedulable activities at once and they differ on how they break ties; TWK-LST, TWK-EST, and TWK-GDR first select a project according to TWK, and then select an

```
 <algorithm> ::= <ps:initialization> <ils>
       <ils> ::= ILS(<ls>, <accept>, <ps:perturb>, <time_ratio>)
        <ls> ::= <ils> | <sa> | <descent>(<ps:neighborhood>)
   <descent> ::= BestImpr | FirstImpr | FirstImprCont
       <sa>  ::= SA(<ps:perturb>, <cooling_schedule>)

    <accept> ::= Always | Improves | ImprovesEqual | Prob(<prob>) | ProbRandom
               | Threshold(<threshold>) | Metropolis(<cooling_sched>)
      <prob> ::= [0, 1]
 <threshold> ::= [0, 1]
<cooling_sched> ::= Cooling(<init_temp>, <final_temp>, <decr_temp_ratio>, <span>)
     <init_temp> ::= {1, 2,..., 10000}
    <final_temp> ::= {1, 2,..., 100}
<decr_temp_ratio> ::= [0, 1]
          <span> ::= {1, 2,..., 10000}
    <time_ratio> ::= {0.1, 0.2, ..., 1}
```

Fig. 1 Problem-independent grammar for generating hybrid ILS algorithms.

```
           <ps:sg> ::= ForwardSerial | ForwardBackwardSerial | multimodeFBSerial
<ps:initialization> ::= SGforInit(<ps:sg>) <ps:initsols>
      <ps:initsols> ::= <ps:init_heur>
                      | <ps:init_heur> <ps:initsols>
     <ps:init_heur> ::= init(<ps:activity_heu>, <ps:mode_heu>)
  <ps:activity_heu> ::= RANDOM_ACTIVITY | MIN_LFT | MIN_FSLK | TWK_LST | TWK_EST
                      | TWK_GDR | MAXTWK_EST | MAXTWK_LST | SASP
      <ps:mode_heu> ::= RANDOM_MODE | MIN_MNR | MIN_DUR | BEST_MODE
  <ps:neighborhood> ::= ModeNeighborhood | ActivityNeighborhood
       <ps:perturb> ::= PerturbFeasibleMode(<ps:perturb_pct>)
                      | PerturbActivity(<ps:perturb_pct>)
                      | PerturbReleaseManyModes(<ps:perturb_pct>)
                      | PerturbHeuRandMode(<ps:activity_heu>, <ps:sg>)
                      | PerturbHeuRandActivity(<ps:mode_heu>, <ps:sg>)
   <ps:perturb_pct> ::= {0, 1, ..., 100}
```

Fig. 2 Problem-specific part of the grammar.

activity according to a secondary rule [7]. For computing these heuristic values in the multi-mode case, we consider all feasible modes. In addition, we save the mode used by the activity heuristic so it can be selected later (by BEST-MODE).

The heuristics for selecting modes are random (RANDOM-MODE), minimum normalized resources (MIN-MNR) [8], a dynamic version of MIN-MNR that takes into account the resources available in the current solution being constructed (MIN-DMNR), minimum duration (MIN-DUR, also called shortest feasible mode, which selects the feasible mode with the shortest duration), and BEST-MODE, which is a heuristic that either selects whatever mode was chosen by the activity heuristic, if any, otherwise, it selects the shortest feasible mode (MIN-DUR).

We consider two neighborhoods in our iterative improvement algorithms. The mode neighborhood simply replaces the mode of each activity with a different mode that respects the non-renewable constraints, considering the resources released by the discarded mode and required by the other activities [2]. The shift (insertion) neighborhood of activities removes one activity from the activity list and reinserts it in a position between its latest predecessor and its earliest successor [2].

We implemented four perturbation methods: *PerturbFeasibleMode* uses the mode neighborhood to randomly change the mode of a number of activities, *PerturbActivity* performs a shift move on a number of random activities, *perturbReleaseManyModes* releases the non-renewable resources assigned to a number of random activities all at once, then tries to assign new random modes to those activities without violating the non-renewable resource constraints. These methods only modify a percentage of the activities. By contrast, *PerturbHeuRandMode* and *PerturbHeuRandActivity* generate a

```
SGforInit(MM-BF)
Init(MIN-FSLK, MIN-DUR)
Init(TWK-EST, MIN-DMNR)
// Additional heuristics manually added (Fig. 4)
ILS(FirstImprCont(ModeNeighborhood),
    ImprovingEqual, perturbActivity(0%),
    t_subls = 1)
```

Fig. 3 Algorithm instantiated from the grammar and submitted to the 2013 MISTA Challenge.

```
Init(RANDOM-ACTIVITY, MIN-MNR)
Init(MIN-LFT, MIN-MNR)
Init(TWK-EST, RANDOM-MODE)
Init(TWK-EST, BEST-MODE)
Init(TWK-LST, MIN-DMNR)
Init(TWK-LST, BEST-MODE)
```

Fig. 4 Additional initialization heuristics manually added to the algorithm.

completely new solution by using the initialization method discussed above with either the RANDOM-MODE or the RANDOM-ACTIVITY heuristic, respectively. These two latter perturbations are similar to a restart mechanism.

3 Our algorithm for the MISTA 2013 Challenge

In order to find the best instantiation of the grammar described above, we run irace using as training instances the two benchmark sets provided by the MISTA 2013 Challenge, and with a maximum budget of 50 000 runs (each run evaluates an instantiation of the grammar on a single instance for 300 seconds). During this training phase, each instantiation of the grammar is a sequential algorithm. The best instantiation found by irace in shown in Fig. 3.

Although we set a limit of two derivations per rule, which would have allowed for two levels of ILS, the algorithm generated by irace is simpler than what we expected (Fig. 3). It is an ILS that uses a *FirstImprCont* descent applied to the mode neighborhood as the subsidiary local search. This descent does not have a time limit: it continues until no improvement is found (or the overall time limit is reached). If the solution generated by the descent is better or equal than the initial one, it replaces the initial one as the current solution. The perturbation is then applied to the current solution, and it modifies one random activity (0%, but the minimum is one activity) by reinserting it in a different position in the schedule. Our intuition is that for very large instances, the best solution found is continually being improved, and that additional levels in the ILS do not speed up this convergence.

In our experiments, we noticed that generating more than two initial solutions, using different heuristic rules, leads to better results and requires little extra computation time. Hence, we manually added six additional initializations (Fig. 4). Thus, the final algorithm generates eight initial solutions and passes the best one to the ILS procedure. Finally, we embed this ILS algorithm within a multi-threaded solver, which runs the algorithm four times in parallel with different random seeds and returns the best solution found.

4 Conclusions

Given the limited space, we do not provide here experimental results. Nonetheless, it is our intention to carry out a complete experimental analysis of the proposed algorithm and compare it with other proposals.

The automatic design method helped us to develop an effective algorithm in a very short time with no previous experience in project scheduling. Moreover, if new algorithmic components that are expected to improve the results become available,

re-generating a new hybrid ILS algorithm only requires computing effort. The same method could be used to design other hybrid metaheuristics for the MRCMPSP.

Acknowledgements The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 246016. Marie-Éléonore Marmion acknowledges support from the ERCIM "Alain Bensoussan" Fellowship Programme. This work was supported by the META-X project, an *Action de Recherche Concertée* funded by the Scientific Research Directorate of the French Community of Belgium and the FRFC project *Méthodes de recherche hybrides pour la résolution de problèmes complexes*. This research has also received funding from the COMEX project within the Inter-university Attraction Poles Programme of the Belgian Science Policy Office. Manuel López-Ibáñez, Franco Mascia and Thomas Stützle acknowledge support from the Belgian F.R.S.-FNRS, of which they are postdoctoral researchers and a research associate, respectively.

References

1. Bedworth, D.D., Bailey, J.E.: Integrated Production Control Systems: Management, Analysis, Design, vol. 2. John Wiley & Sons, New York, NY (1982)
2. Bouleimen, K., Lecocq, H.: 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 (2003).
3. Burke, E.K., Hyde, M.R., Kendall, G.: Grammatical evolution of local search heuristics. IEEE Trans. Evol. Comput. **16**(7), 406–417 (2012).
4. Kolisch, R., Hartmann, S.: Experimental investigation of heuristics for resource-constrained project scheduling: An update. European Journal of Operational Research **174**(1), 23–37 (2006).
5. Kurtulus, I., Davis, E.W.: Multi-project scheduling: Categorization of heuristic rules performance. Management Science **28**(2), 161–172 (1982).
6. López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., Birattari, M.: The irace package, iterated race for automatic algorithm configuration. Tech. Rep. TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium (2011).
7. Lova, A., Tormos, P.: Analysis of scheduling schemes and heuristic rules performance in resource-constrained multiproject scheduling. Annals of Operations Research **102**(1-4), 263–286 (2001).
8. Lova, A., Tormos, P., Cervantes, M., Barber, F.: An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. International Journal of Production Economics **117**(2), 302–316 (2009).
9. Marmion, M.E., Mascia, F., López-Ibáñez, M., Stützle, T.: Automatic design of hybrid stochastic local search algorithms. In: Blesa, M.J., et al. (eds.) Hybrid Metaheuristics, *LNCS*, vol. 7919, pp. 144–158. Springer, Heidelberg (2013).
10. Mascia, F., López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T.: From grammars to parameters: Automatic iterated greedy design for the permutation flow-shop problem with weighted tardiness. In: Learning and Intelligent Optimization, LION 7, *LNCS*, vol. to appear. Springer, Heidelberg, (2013).
11. Mckay, R.I., Hoai, N.X., Whigham, P.A., Shan, Y., O'Neill, M.: Grammar-based genetic programming: A survey. Genetic Programming and Evolvable Machines **11**(3-4), 365–396 (2010).
12. Sprecher, A., Hartmann, S., Drexl, A.: An exact algorithm for project scheduling with multiple modes. OR Spektrum **19**(3), 195–203 (1997).
13. Sprecher, A., Kolisch, R., Drexl, A.: Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. European Journal of Operational Research **80**(1), 94–102 (1995).