MaxSAT and pseudo-Boolean Solutions for the Multi-Skill Project Scheduling Problem

Wilber B. Quito
University of Girona, Spain

Jordi Coll
University of Girona, Spain

Mateu Villaret
University of Girona, Spain

Abstract

The Multi-Skill Project Scheduling Problem (MSPSP) is a variant of the paradigmatic Resource-Constrained Project Scheduling Problem (RCPSP). In the RCPSP the goal is to find a start time for each one of the activities (schedule) of a project such that the makespan is minimised. The schedule must satisfy a set of constrains on resources and precedences. MSPSP extends the RCPSP since in the MSPSP the activities do not directly ask for resources but they ask for skills. These skills are supplied by renewable resources, and every resource is specialized to master a subset of the skills. Logic-based approaches have shown to be state-of-the-art for solving many scheduling problems. In this work we propose a MaxSAT and a pseudo-Boolean Constraints encoding for MSPSP.

2012 ACM Subject Classification Theory of computation Logic

Keywords and phrases MaxSAT; pseudo-Boolean; Scheduling; Encoding

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

The Resource-Constrained Project Scheduling Problem (RCPSP) [2] is a fundamental scheduling challenge that serves as a model for a wide range of industrial applications. Due to its broad applicability, it has been extensively studied and continues to attract attention today. In the RCPSP, a set of non-preemptive activities must be scheduled (i.e., setting their start times) while adhering to both precedence and resource constraints. Precedence constraints ensure that certain activities cannot begin until others have been completed, while resource constraints prevent the total resource usage at any given time from exceeding the available supply. The primary objective is typically to minimize the total duration of the project, also known as the makespan. There exist many variants of this problem, we refer the reader to [10] for a complete survey on different variations and extensions of the RCPSP.

Among these variants is the Multi-Skill Project Scheduling Problem (MSPSP), a generalization of the RCPSP. In the MSPSP, activities require specific skills rather than resources directly. These skills are provided by renewable resources, each of which is specialized in a subset of the available skills. For example, resources may represent workers, and the skills correspond to their individual abilities. The resource constraints state that one resource can only work at one skill of one activity at a time, and that a resource can only supply skills that it masters. The resources are unary, i.e., they can only supply one unit of skill at a time, but the activities may require many units of each skill. Also, the set of resources that an activity is using cannot change at any moment of execution, i.e., the resource usage of the activities is also non-preemptive.

It was argued in [3] that an instance of MSPSP can be reformulated as an instance of the Multi-Mode Resource-Constrained Project Scheduling Problem (a related variant within the RCPSP family) by representing different combinations of resource assignments to an activity as distinct execution modes. However, this transformation leads to a *combinatorial explosion* as determining whether there exists a combination of modes that satisfies the non-renewable resource constraint, which the Multi-mode variant also considers, is already NP-hard. As a result, the MSPSP is strongly NP-hard [3].

Several MILP models have been proposed for MSPSP [3, 8], with time-indexed formulations showing particularly strong performance. In these models, a set of cumulative cuts—analogous to the renewable resource constraints in the RCPSP—were introduced, significantly accelerating the solving process. Later, [13] developed a constraint programming (CP) model combined with a tailored search strategy, solved using a Lazy Clause Generation (LCG) solver. In their approach, the cumulative cuts are also exploited, interpreted in the CP context as implied or redundant constraints that enhance propagation.

There exist several works showing the convenience of using logic-based methods to solve scheduling problems and in particular RCPSP and variants. In these works the models are SAT modulo theories models (SMT): in [7], precedence and resource constraints were encoded numerically using Linear Integer Arithmetic theory, then in [4] it was shown that encoding resource constraints (that are basically pseudo-Boolean constraints) directly into SAT was a better choice and in [5, 6] further sophisticated encodings to SAT of those pseudo-Boolean Constraints provided an even better performance. Notice that these last models only delegate precedence constraints to the difference logic theory, while the rest of the problem is fully encoded into SAT. The optimization process in the last works was delegated to iterative Satisfiability checks for tighter upper bounds.

In this work we propose to solve the MSPSP directly with a MaxSAT encoding and with a pseudo-Boolean encoding. This way we try to answer the following question: *are current pseudo-Boolean solvers appropriate tools to solve scheduling problems, in particular MSPSP, where there occur plenty of pseudo-Boolean constraints or is it better to encode them into SAT?*

We hope this work further stimulates the upcoming research of the PhD thesis that will revolve in the use of logic-based tools for scheduling. These tools will be used as black-boxes for a model-and-solve approach and as cristal-boxes where we aim at tunning them to be better suited for the problem at hand via specialized propagators or heuristics.

2 The MSPSP

The Multi-Skill Project Scheduling Problem (MSPSP) is defined by a tuple (V, p, E, R, L, m, b) where:

- $V = \{A_0, A_1, ..., A_n, A_{n+1}\}$ is a set of activities. Activities A_0 and A_{n+1} are dummy activities that represent, by convention, the start and the end of the schedule, respectively. The set of non-dummy activities is defined by $A = \{A_1, ..., A_n\}$.
- $p \in \mathbb{N}^{n+2}$ is a vector of naturals, with p_i being the duration of activity A_i. For the dummy activities, $p_0 = p_{n+1} = 0$, and $p_i > 0, \forall A_i \in A$.
- E is a set of pairs of activities representing the precedence relations between them. Concretely, $(A_i, A_j) \in E$ iff the execution of activity A_i must precede that of activity A_j, i.e., activity A_j must start after activity A_i has finished. We assume that we are given a precedence activity-on-node graph $G = (V, E)$ that contains no cycles, that A_0 is a predecessor of all other activities and A_{n+1} is a successor of all other activities.
- $R = \{R_1, ..., R_v\}$ is a set of unary renewable resources.
- $L = \{L_1, ..., L_s\}$ is a set of skills.

- $m \in \mathbb{B}^{v \times s}$ is a Boolean matrix, with $m_{k,l}$ being the true iff resource R_k masters the skill L_l.
- $b \in \mathbb{N}^{(n+2) \times s}$ is a matrix of naturals, where $b_{i,l}$ represents the number of resources mastering skill L_l that activity A_i requires during its execution. The start and end dummy activities do not require skills, i.e., $b_{0,l} = b_{n+1,l} = 0$ for any skill L_l.

In this context, a solution of the MSPSP is a schedule and a resource assignment minizing the total makespan. A schedule S is defined as a vector of natural numbers $S = (S_0, S_1, \ldots, S_n, S_{n+1})$, where each S_i represents the start time of activity A_i. The start time of the dummy initial activity is set to $S_0 = 0$, and the start time of the dummy last activity S_{n+1} must be minimized. A resource assignment $RA \in \mathbb{B}^{(n+2) \times v \times s}$ is defined as a matrix of three dimensions of Booleans, where $RA_{i,k,l}$ is true iff activity A_i uses resource R_k to perform skill L_l.

Hence, MSPSP can be stated as follows.

Minimise:

$$S_{n+1}$$

subject to the following constraints:

- Precedence contraints state that, for any $(A_i, A_j) \in E$, activity A_j can not start until A_i has finished:

$$S_j - S_i \geq p_i \qquad \forall (A_i, A_j) \in E$$

- A resource cannot perform a skill that it does not master:

$$\neg m_{k,l} \rightarrow \neg RA_{i,k,l} \qquad \forall A_i \in A,\ \forall R_k \in R,\ \forall L_l \in L$$

- Each activity must have the required number of resources covering each of the skills:

$$\sum_{R_k \in R} RA_{i,k,l} = b_{i,l} \qquad \forall A_i \in A, \forall L_l \in L$$

- A resource can only work at one skill of one activity at a time:

$$\sum_{L_l \in L} \sum_{A_i \in A} ite((S_i \leq t < S_i + p_i); RA_{i,k,l}; 0) \leq 1 \qquad \forall R_k \in R, \forall t \in H$$

By *ite* we denote the *if then else* construction that returns the second parameter in case of the first parameter being true and returns the third parameter otherwise.

3 MaxSAT and pseudo-Boolean Encodings for MSPSP

Handling renewable resource limits in the RCPSP is one of the hardest parts of the problem and has been widely studied. In [12], several main ways to model this are explained. The *Task* approach checks how much of each resource is used when each activity starts. A similar method, called the *Event* approach, looks at specific time points when activities can begin. It schedules these moments, assigns activities to them, and then checks that the resource limits are not exceeded at any of them. Another method, the *Flow* approach, works as a network where once an activity ends, the resources it is using are passed on to other activities that need them.

In this work we focus on time-indexed models, commonly known as the *Time* approach [11], which have shown to provide good performance when solving the different benchmark instances available in the literature. This approach consists in discretising the scheduling horizon H (the period of time in which the schedule will take place) into unit intervals from 0 to an *upper bound* UB, i.e. $H = \{0, 1, \ldots, UB\}$. The upper bound must be large enough so that the optimal solution (if any) will have a makespan not greater than UB. It must be ensured that the capacity of a resource is not exceeded at any of the time instants in H. A usual way of achiving this purpuse is to introduce auxiliary 0/1 variable per each time interval.

We introduce the following two precomputed sets that we use to define the encodings.

$R(A_i)$	This is the subset of resources that master at least one of the skills demanded by activity A_i
$L(A_i)$	This is the subset of skills that activity A_i demand, i.e. the skills L_l such that $b_{i,l} > 0$.

In both (MaxSAT and pseudo-Boolean) encodings, we use the following sets of (pseudo) Boolean variables:

$s_{i,t}$	True iff A_i has *already started* at time t. $\forall A_i \in A$, $\forall t \in H$
$x_{i,t}$	True iff activity A_i *is running* at time t. $\forall A_i \in A$, and $\forall t \in H$
$ar_{i,k}$	True iff activity A_i *uses resource* R_k. $\forall A_i \in A$, and $\forall R_k \in R(A_i)$
$ars_{i,k,l}$	True iff activity A_i *uses resource* R_k *for skill* L_l. $\forall A_i \in A$, $\forall R_k \in R(A_i)$, and $\forall L_l \in L(A_i)$ such that $m_{k,l}$, i.e. such that resource R_k masters skill L_l
$art_{i,k,t}$	True iff activity A_i *uses resource* R_k *at time* t. $\forall A_i \in A$, $\forall R_k \in R(A_i)$, and $\forall t \in H$

Using these sets and variables, we can now define our SAT-based encodings for both MaxSAT and pseudo–Boolean (PB). For each constraint and for the objective function, we first present the MaxSAT formulation, followed by its pseudo–Boolean equivalent. For the sake of a clearer presentation, we provide some constraints for the MaxSAT and PB encodings using higher level syntax than clauses or PB constraints respectively. Later, in Subsections 3.1 and 3.2, we explain how these constraints are converted to clauses and PB constraints.

Initial dummy activity

Constraints (1) and (2) ensure that the initial dummy activity starts at time 0.

$$s_{0,0} \tag{1}$$

$$s_{0,0} \geq 1 \tag{2}$$

Activity continuity

Constraints (3) and (4) state that if an activity has started at time t_i it has also started at time t_{i+1}.

$$\neg s_{i,t} \vee s_{i,t+1} \qquad \forall A_i \in A, \forall t \in H \tag{3}$$

$$\neg s_{i,t} + s_{i,t+1} \geq 1 \qquad \forall A_i \in A, \forall t \in H \tag{4}$$

We must handle the case where the temporal indexing goes out of the valid time range, i.e., when $t \notin \{0, \ldots, UB\}$ for variables $s_{i,t}$. Variables with time indices that overflow to the right (i.e., $t > UB$) are trivially true, while those that overflow to the left as in (5) (i.e., $t < 0$) are trivially false.

Precedences

Constraints (5) and (6) define the precedences between activities.

$$\neg s_{j,t} \vee s_{i,t-p_i} \qquad \forall (A_i, A_j) \in E, \forall t \in H \qquad (5)$$

$$\sum_{\forall t \in H} s_{i,t} - \sum_{\forall t \in H} s_{j,t} \geq p_i \qquad \forall (A_i, A_j) \in E \qquad (6)$$

Execution definition

Constraints (7) and (8) state that an activity A_i is running at time t iff it has already started at time t and had not started at $t - p_i$.

$$x_{i,t} \leftrightarrow s_{i,t} \wedge \neg s_{i,t-p_i} \qquad \forall A_i \in A, \forall t \in H \qquad (7)$$

$$x_{i,t} \leftrightarrow s_{i,t} + \neg s_{i,t-p_i} \geq 2 \qquad \forall A_i \in A, \forall t \in H \qquad (8)$$

Resource assigned to activity

Constraints (9) and (10) ensure that a resource R_k is assigned to an activity A_i iff R_k supplies some skill L_l to A_i.

$$ar_{i,k} \leftrightarrow \bigvee_{\substack{L_l \in L(A_i) \\ m_{k,l}}} ars_{i,k,l} \qquad \forall A_i \in A, \forall R_k \in R(A_i) \qquad (9)$$

$$ar_{i,k} \leftrightarrow \sum_{\substack{L_l \in L(A_i) \\ m_{k,l}}} ars_{i,k,l} \geq 1 \qquad \forall A_i \in A, \forall R_k \in R(A_i) \qquad (10)$$

Resource assigned to activity at specific time

Constraints (11) and (12) capture that a resource R_k is considered to be working on activity A_i at time t precisely when A_i is running at t and R_k has been assigned to it.

$$art_{i,k,t} \leftrightarrow x_{i,t} \wedge ar_{i,k} \qquad \forall A_i \in A,\ \forall R_k \in R(A_i),\ \forall t \in H \qquad (11)$$

$$art_{i,k,t} \leftrightarrow x_{i,t} + ar_{i,k} \geq 2 \qquad \forall A_i \in A,\ \forall R_k \in R(A_i),\ \forall t \in H \qquad (12)$$

Skill coverage requirement

Constraints (13) and (14) ensure that each activity has enough resources to cover each of its required skills.

$$\underset{\substack{R_k \in R, \\ s.t.:\ m_{k,l}}}{ALK} (ars_{i,k,l}, b_{i,l}) \qquad A_i \in A,\ \forall L_l \in L(A_i) \qquad (13)$$

$$\sum_{\substack{R_k \in R, \\ s.t.:\ m_{k,l}}} ars_{i,k,l} \geq b_{i,l} \qquad A_i \in A,\ \forall L_l \in L(A_i) \qquad (14)$$

By $ALK(list, bound)$ we refer to the *at least k* constraint requiring at least *bound*-many values of *list* to be true.

At most one skill per resource per activity

Constraints (15) and (16) state that a resource can contribute with at most one skill in one activity.

$$\underset{\substack{L_l \in L(A_i),\\ s.t.:\ m_{k,l}}}{AMO} \quad ars_{i,k,l} \qquad \forall A_i \in A, \forall R_k \in R(A_i) \tag{15}$$

$$\sum_{\substack{L_l \in L(A_i),\\ s.t.:\ m_{k,l}}} ars_{i,k,l} \leq 1 \qquad \forall A_i \in A, \forall R_k \in R(A_i) \tag{16}$$

By $AMO(list)$ we refer to the *at most one* constraint requiring that at most one value of *list* is true.

Resource capacity per time step

Constraints (17) and (18) ensure that a resource does not contribute to more than one activity at a time.

$$\underset{\substack{A_i \in A,\ s.t.:\\ t \in H,\\ R_k \in R(A_i)}}{AMO} \quad art_{i,k,t} \qquad \forall R_k \in R,\ \forall t \in H \tag{17}$$

$$\sum_{\substack{A_i \in A,\ s.t.:\\ t \in H,\\ R_k \in R(A_i)}} art_{i,k,t} \leq 1 \qquad \forall R_k \in R, \forall t \in H \tag{18}$$

Objective function

For both the MaxSAT and pseudo–Boolean encodings, the objective function is conceptually the same but expressed differently.

Finding an optimal solution for MaxSAT, consists in finding an assignment that maximizes the number satisfied soft clauses (or, equivalently, minimizes the number of the unsatisfied soft clauses), while satisfying all hard clauses. In the MaxSAT encoding, all constraints given so far will be hard clauses and we introduce soft clauses $(s_{n+1,t},\ 1)$ for the dummy activity A_{n+1} at each time point $t \in H$, aiming for the solver to satisfy as many of these clauses as possible.

In contrast, the PB encoding aims to minimize the sum of negated pseudo variables $\neg s_{n+1,t}$ for the dummy activity A_{n+1} at each time point $t \in H$, which is equivalent to maximizing the number of time points t for which $s_{n+1,t}$ holds true.

$$(s_{n+1,t},\ 1) \qquad \forall t \in H \tag{19}$$

$$minimizing \quad \sum_{\forall t \in H} \neg s_{n+1,t} \tag{20}$$

3.1 Clausal form of the MaxSAT encoding

Notice also that most of the constraint stated in Section 3 are naturally encoded into clauses, but some of them need a reification process, such as constraints (8), (10), and more. Other constraints are naturally cardinality constraints, such as (13), that we will encode to SAT using the sorting network based encoding from [1]. Constraints (15) and (17) are At Most One constraints (AMO), that we will encode into SAT using pairwise mutual exclusions between variables, i.e. clauses of the form $\neg x \vee \neg y$ for any pair of variables in the AMO constraint.

3.2 PB form for the pseudo–Boolean Encoding

In the MSPSP formulation, many of the PB constraints arise naturally. However, for constraints (8), (10), and (12), it becomes necessary to introduce *fresh variables*. Specifically, we require the ability to define a fresh Boolean variable y to represent the reification of a constraint like $\sum_i a_i l_i \geq A$, meaning that y is true iff the constraint holds. This is expressed as:

$$y \leftrightarrow \sum_i a_i l_i \geq A$$

For a detailed explanation of the reification process in the context of pseudo-Boolean solving, we refer the reader to [9].

4 Further work

We aim at conduct an empirical evaluation [13] of the proposed encodings with state-of-the-art benchmarks of the MSPSP and also using state-of-the-art solvers for MaxSAT and for pseudo-Booleans. This evaluation could serve to identify wether current pseudo-Boolean solvers not based on encodings to SAT are well-suited for this kind of problems. If the results do not provide any clear evidence we aim at generating instances with extreme properties such as large durations, larger amount of skill demands, large or small amount of precedences, etc. and explore the relative effect is both approaches. We hope that this information allows us to identify weaknesses and strengths of both solving technologies.

References

1 Ignasi Abío, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell. A Parametric Approach for Smaller and Better Encodings of Cardinality Constraints. In *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP)*, pages 80–96. Springer, 2013.

2 Christian Artigues, Sophie Demassey, and Emmanuel Neron. *Resource-constrained project scheduling: models, algorithms, extensions and applications.* John Wiley & Sons, 2013.

3 Odile Bellenguez-Morineau. Methods to solve multi-skill project scheduling problem. *4OR*, 6(1):85–88, 2008.

4 Miquel Bofill, Jordi Coll, Josep Suy, and Mateu Villaret. Solving the multi-mode resource-constrained project scheduling problem with SMT. In *28th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2016, San Jose, CA, USA, November 6-8, 2016*, pages 239–246. IEEE Computer Society, 2016. `doi:10.1109/ICTAI.2016.0045`.

5 Miquel Bofill, Jordi Coll, Josep Suy, and Mateu Villaret. Compact mdds for pseudo-boolean constraints with at-most-one relations in resource-constrained scheduling problems. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial*

Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017, pages 555–562. ijcai.org, 2017. URL: `https://doi.org/10.24963/ijcai.2017/78`, `doi:10.24963/IJCAI.2017/78`.

6 Miquel Bofill, Jordi Coll, Josep Suy, and Mateu Villaret. Smt encodings for resource-constrained project scheduling problems. *Computers & Industrial Engineering*, 149:106777, 2020.

7 Miquel Bofill, Miquel Palahí, Josep Suy, and Mateu Villaret. Solving constraint satisfaction problems with SAT modulo theories. *Constraints An Int. J.*, 17(3):273–303, 2012. URL: `https://doi.org/10.1007/s10601-012-9123-1`, `doi:10.1007/S10601-012-9123-1`.

8 Carlos Eduardo Montoya Casas. *New methods for the multi-skills project scheduling problem.* PhD thesis, Ecole des mines de Nantes, 2012.

9 Stephan Gocht and Jakob Nordström. Certifying parity reasoning efficiently using pseudo-boolean proofs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3768–3777, 2021.

10 Sönke Hartmann and Dirk Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1):1–14, 2010. URL: `https://www.sciencedirect.com/science/article/pii/S0377221709008558`, `doi:10.1016/j.ejor.2009.11.005`.

11 A Alan B Pritsker, Lawrence J Waiters, and Philip M Wolfe. Multiproject scheduling with limited resources: A zero-one programming approach. *Management science*, 16(1):93–108, 1969.

12 Josep Suy. *A Satisfiability Modulo Theories Approach to Constraint Programming.* PhD thesis, PhD thesis, Universitat de Girona, 2013.

13 Kenneth D Young, Thibaut Feydy, and Andreas Schutt. Constraint programming applied to the multi-skill project scheduling problem. In *Principles and Practice of Constraint Programming: 23rd International Conference, CP 2017, Melbourne, VIC, Australia, August 28–September 1, 2017, Proceedings 23*, pages 308–317. Springer, 2017.