The Complexity Landscape of Resource-Constrained Scheduling

Robert Ganian[1], **Thekla Hamm**[1] and **Guillaume Mescoff**[2]
[1]Vienna University of Technology
[2]Rennes University
rganian@gmail.com, thamm@ac.tuwien.ac.at, guillaume.mescoff@ens-rennes.fr

Abstract

The Resource-Constrained Project Scheduling Problem (RCPSP) and its extension via activity modes (MRCPSP) are well-established scheduling frameworks that have found numerous applications in a broad range of settings related to artificial intelligence. Unsurprisingly, the problem of finding a suitable schedule in these frameworks is known to be NP-complete—however, aside from a few results for special cases, we have lacked an in-depth and comprehensive understanding of the complexity of the problems from the viewpoint of natural restrictions of the considered instances.

In the first part of our paper, we develop new algorithms and give hardness-proofs in order to obtain a detailed complexity map of (M)RCPSP that settles the complexity of all 1024 considered variants of the problem defined in terms of explicit restrictions of natural parameters of instances. In the second part, we turn to implicit structural restrictions defined in terms of the complexity of interactions between individual activities. In particular, we show that if the treewidth of a graph which captures such interactions is bounded by a constant, then we can solve MRCPSP in polynomial time.

1 Introduction

The Resource-Constrained Project Scheduling Problem (RCPSP) provides a generic and well-established framework for the formal description of scheduling problems. RCPSP has been the subject of extensive theoretical as well as empirical research in the context of Artificial Intelligence [Smith and Pyle, 2004; Kuster *et al.*, 2007; Varakantham *et al.*, 2016; Song, 2017], Operations Research and Scheduling [van Bevern *et al.*, 2016; Fu *et al.*, 2010; Fu *et al.*, 2016]; see also the survey [Kolisch and Padman, 2001] and book [Artigues *et al.*, 2008] dedicated to the topic. RCPSP falls within the wider framework of so-called *scheduling problems* which are classical and have been at the focus of a vast and diverse amount of works [Schwindt and Zimmermann, 2015].

On a high level, in scheduling problems one is given a set of *activities* that have to be processed in a given time frame while adhering to certain conditions. Solutions to scheduling problems are also called *schedules*. RCPSP represents the subclass of scheduling problems where the processing of activities requires the use of resources; these have certain *capacities* that limit how many activities can be processed concurrently, and activities have certain *resource requirements* and *durations* which describe what resources each activity needs to be assigned to and for how long. It is assumed that an activity cannot be interrupted (one also calls this *non-preemptiveness*). Now for RCPSP, a schedule consists of an assignment of the activities to certain *points in time* (simply modeled by natural numbers) such that the time it takes to process all activities satisfies a given *makespan* bound. Often one requires that activities also adhere to a precedence order.

A prominent generalisation of RCPSP that has received considerable attention [Bofill *et al.*, 2017; Barrios *et al.*, 2011; Poppenborg and Knust, 2016] is based on the addition of activity modes, capturing scenarios where it is possible to complete activities in multiple ways—each possibly requiring different amounts of time and resources. This gives rise to the Multi-mode Resource-Constrained Project Scheduling Problem (MRCPSP).

Contribution. It is known that RCPSP is NP-complete, and in fact remains NP-complete even when we consider only a single resource and when there are no precedence constraints [van Bevern *et al.*, 2016; Garey and Johnson, 1975]. However, so far we have lacked a comprehensive understanding of the complexity of these fundamental scheduling problems under explicit and natural restrictions of considered instances; interestingly, already Blazewicz, Lenstra and Kan (1983) called for such a theoretical investigation in their seminal paper which formalized RCPSP: "The obvious research program would be to determine the borderline between easy and hard resource constrained scheduling problems." For example, is (M)RCPSP restricted to instances of constant makespan and number of resources NP-hard, or does the problem become polynomial-time solvable?

Our first contribution is a complete complexity map for (M)RCPSP which takes into account all combinations of variants arising from the following explicit restrictions/attributes which are immediately tied to numerical properties of the input or have been established in previous literature:

- Fixed upper-bound on number of activities (n), number

of resources (m), maximum duration of an activity (t), maximum capacity of a resource (c), makespan (C_{max}), and/or on the number of activities that can use each resource (r_{deg});
- No precedence constraints ($\neg P$);
- "Simple" instances, where each activity only uses a single resource (S) (see, e.g., the work of Damay et al. [2007]);
- Whether we consider modes (MRCPSP) or not (RCPSP);
- All numbers are encoded in unary[1] (U).

With the exception of the modes attribute, we will adopt the convention of listing the attributes considered in a given fragment in angular brackets—for instance, MRCPSP$\langle c, r_{\text{deg}}\rangle$ refers to instances of MRCPSP where each resource has capacity bounded (by a constant), and each resource is only used by a (constant-)bounded number of activities.

Each of the above attributes can be viewed as an independent binary "switch"; altogether this amounts to 2^{10} considered fragments of (M)RCPSP. Our first contribution is a complete classification of all of these problems in terms of classical complexity theory; we show that 736 fragments are polynomial-time solvable and 288 are NP-hard. This is achieved by a collection of 3 new hardness proofs (in addition to 4 known NP-hard cases) and 6 polynomial-time algorithms, utilizing a range of diverse algorithmic techniques. An illustration of our complexity map is provided in Figure 1.

In the second part of our paper, we shift our focus from explicit restrictions on instances to implicit ones. More specifically, we ask whether one can exploit the structure of interactions between activities and/or resources to lift any of the obtained polynomial-time algorithms towards more general classes of instances. A natural way of capturing such structure is the concept of *treewidth*. As our second contribution, we show that treewidth allows us to push the frontiers of tractability for MRCPSP when applied to the *activity graph*—a graph which represents activities as vertices and adds edges between activities which interact either by sharing a resource or a precedence constraint.

Related work. While the treewidth of instances has not been considered for RCPSP yet, the parameter has found numerous applications in prominent subfields and problems that are relevant for AI research, such as SAT [Gottlob *et al.*, 2002], ILP [Ganian and Ordyniak, 2018] and CSP [Cohen *et al.*, 2015]. It is worth noting that instances of low treewidth may arise naturally in a variety of problems and settings—for example, the treewidth of control flow graphs arising from goto-free programs is known to be at most 6 [Thorup, 1998]. RCPSP is known to be polynomial-time solvable when the *poset width* of the precedence constraints is bounded [van Bevern *et al.*, 2016].

2 Preliminaries

For an integer i, we use $[i]$ as shorthand for $\{1,\ldots,i\}$. The function argmin refers to an (arbitrary) argument of the minimum. We assume that $\mathbb{N}$ is the set of non-negative integers. For a vector R, we use $R[\ell]$ to denote its ℓ-th coordinate.

Problem definition. An instance $\mathcal{I}$ of MRCPSP is a tuple $\langle A, R, C, \mathcal{M}, T, \mathcal{Q}, <_P, C_{\text{max}}\rangle$, of
- $A = \{a_1,\ldots,a_n\}$ a set of *activities*;
- $R = \{r_1,\ldots,r_{m'},r_{m'+1},\ldots,r_{m'+m''}\}$ a set of *resources*, where we distinguish between m' *renewable* $(r_1,\ldots,r_{m'})$ and m'' *non-renewable* $(r_{m'+1},\ldots,r_{m'+m''})$ resources, and let $m = m' + m''$;
- $C : R \to \mathbb{N}$ a mapping from resources to *capacities*;
- $\mathcal{M} = \{M_1,\ldots,M_n\}$ a set of (pairwise disjoint) *activity mode* sets, and let $B = \bigcup_{i\in[n]} M_i$ be the set of all modes;
- $T : B \to \mathbb{N}\setminus\{0\}$ a mapping from modes to *durations*;
- $\mathcal{Q} : B \to \mathbb{N}^m$ a mapping of modes to *resource requirements*;
- $<_P$ a strict partial order on A which represents *precedence constraints*;
- $C_{\text{max}} \in \mathbb{N}$ is the allowed *makespan*; we also refer to numbers in $[C_{\text{max}}]\cup\{0\}$ as *time points* or *time steps*.

A solution or *schedule* for $\mathcal{I}$ is a pair (ω,α), where ω is a mapping from each activity a_i to a mode $w_i \in M_i$ and α is a mapping from each a_i to a *starting time* in $[C_{\text{max}}]\cup\{0\}$, satisfying the following four types of constraints.

Makespan constraints: For each activity a_i: $\alpha(a_i) + T(w_i) \le C_{\text{max}}$.

Resource constraints: For each resource r_ℓ:
- if r_ℓ is renewable, i.e., $\ell \in [m']$, for each time point $j \in [C_{\text{max}}]$: $R_j[\ell] \le R[\ell]$, where R_j denotes the vector of resource capacities being used at time point j—formally, $R_j = \sum_{i:\alpha(a_i)\le j<\alpha(a_i)+T(w_i)} \mathcal{Q}(w_i)$; and
- if r_ℓ is non-renewable, i.e., $\ell \in [m]\setminus[m']$, $\sum_{a_i\in A} \mathcal{Q}(w_i)[\ell] \le R[\ell]$.

Precedence constraints: For each $a_i, a_{i'} \in A$ such that $a_i <_P a_{i'}$: $\alpha(a_i) + T(w_i) \le \alpha(a_{i'})$.

The task in MRCPSP is to decide whether the instance admits a solution (in which case we also wish to compute such a solution), or not.

RCPSP is the restriction of MRCPSP to the case where each activity has a single mode and all resources are renewable. In this case, we can simplify the notation by omitting $\mathcal{M}$ and having T and $\mathcal{Q}$ directly refer to activities in A.

The problem definition suggests a number of interesting and natural parameters which we want to consider as flags used to define the basic fragments of (M)RCPSP considered in this paper. A class $\mathcal{D}$ of MRCPSP instances has the flag $\langle n\rangle$ if there exists some integer z such that each instance in $\mathcal{D}$ has at most z activities. The flags $\langle m\rangle$ (total number of resources—renewable as well as non-renewable), $\langle t\rangle$ (maximum value of T), $\langle c\rangle$ (maximum value of C), $\langle C_{\text{max}}\rangle$ are defined analogously. $\mathcal{D}$ has the flag $\langle r_{\text{deg}}\rangle$ if there exists some integer z such that, for each instance $\mathcal{I}\in\mathcal{D}$ and for each resource r_ℓ in that instance, there are at most z activities which can use r_ℓ—formally, $|\{\, a_i \mid \forall b \in M_i, C(b)[\ell] = 0\}| \ge n - z$. Intuitively, $\langle r_{\text{deg}}\rangle$ represents a natural generalization of

[1]This captures the distinction between weak and strong NP-hardness. Unary instances arise when encoding certain problems.

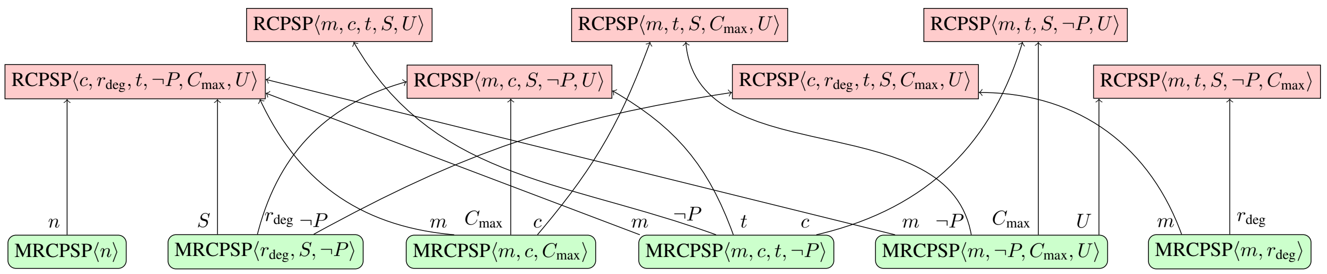
$\mathrm{RCPSP}\langle m, c, t, S, U\rangle$
$\mathrm{RCPSP}\langle m, t, S, C_{\max}, U\rangle$
$\mathrm{RCPSP}\langle m, t, S, \neg P, U\rangle$
$\mathrm{RCPSP}\langle c, r_{\deg}, t, \neg P, C_{\max}, U\rangle$
$\mathrm{RCPSP}\langle m, c, S, \neg P, U\rangle$
$\mathrm{RCPSP}\langle c, r_{\deg}, t, S, C_{\max}, U\rangle$
$\mathrm{RCPSP}\langle m, t, S, \neg P, C_{\max}\rangle$
n
S
$r_{\deg}$ $\neg P$
m $C_{\max}$ c
m $\neg P$ t c
m $\neg P$ $C_{\max}$ U
m $r_{\deg}$
$\mathrm{MRCPSP}\langle n\rangle$
$\mathrm{MRCPSP}\langle r_{\deg}, S, \neg P\rangle$
$\mathrm{MRCPSP}\langle m, c, C_{\max}\rangle$
$\mathrm{MRCPSP}\langle m, c, t, \neg P\rangle$
$\mathrm{MRCPSP}\langle m, \neg P, C_{\max}, U\rangle$
$\mathrm{MRCPSP}\langle m, r_{\deg}\rangle$

Figure 1: The obtained complexity map for MRCPSP showing polynomial-time solvable and NP-hard fragments of MRCPSP. Arrows indicate tightness of the polynomially tractable fragments w.r.t. the flags; removing any flag from a tractable fragment results in an NP-hard fragment.

the flag $\langle n\rangle$, since it does not restrict the number of activities globally but only relatively to each resource.

Three of the four remaining flags—notably the ones signifying the lack of precedence constraints ($\langle\neg P\rangle$), an unary encoding of the numbers ($\langle U\rangle$), and whether we have modes or not—are self-explanatory. The last remaining flag is $\langle S\rangle$ (short for "simple"), which signifies that for every activity a_i in an instance in the class $\mathcal{D}$, there is at most one resource used by a_i in any mode—formally, $\forall i \in [n]\ |\{\ell \in [m] \mid \exists b \in M_i\ \mathcal{Q}(b)[\ell] > 0\}| \leq 1$. Simple instances represent a middle ground between instances with a single resource and general instances [Damay *et al.*, 2007] and have a natural correspondence to classical scheduling over m types of machines [Gehrke *et al.*, 2018].

We will use $|\mathcal{I}|$ to denote the size of a (unary or binary, depending on the flag "U") encoding of the instance $\mathcal{I}$.

Treewidth and Graph Representations. A *nice tree-decomposition* $\mathcal{T}$ of a graph $G = (V, E)$ is a pair $(T, \mathcal{X})$, where T is a tree rooted at a node r and $\mathcal{X}$ is a function that assigns each tree node t a set $\mathcal{X}(t) = X_t \subseteq V$ of vertices such that the following conditions hold:

- For every vertex $u \in V$, there is a tree node t such that $u \in X_t$.
- For every edge $uv \in E(G)$ there is a tree node t such that $u, v \in X_t$.
- For every vertex $v \in V(G)$, the set of tree nodes t with $v \in X_t$ forms a subtree of T.
- $|X_r| = |X_\ell| = 1$ for every leaf ℓ of T.
- There are only three kinds of non-leaf nodes in T:

  Introduce node: a node t with exactly one child t' such that $X_t = X_{t'} \cup \{v\}$ for some vertex $v \notin X_{t'}$.
  Forget node: a node t with exactly one child t' such that $X_t = X_{t'} \setminus \{w\}$ for some vertex $w \in X_{t'}$.
  Join node: a node t with two children t_1, t_2 such that $X_t = X_{t_1} = X_{t_2}$.

The sets X_t are called *bags* of the decomposition $\mathcal{T}$ and X_t is the bag associated with the tree node t. The *width* of a nice tree-decomposition $(T, \mathcal{X})$ is the size of a largest bag minus 1. The *treewidth* of a graph G, denoted by $\mathrm{tw}(G)$, is the minimum possible width of a nice tree-decomposition of G.

For every fixed k, a nice tree-decomposition of a graph G of treewidth k can be computed efficiently if one exists [Bodlaender *et al.*, 2016; Kloks, 1994; Arnborg *et al.*, 1987]. We use $\mathcal{X}^{\downarrow}(t)$ to denote the set of all vertices in bags of the subtree of T rooted at t.

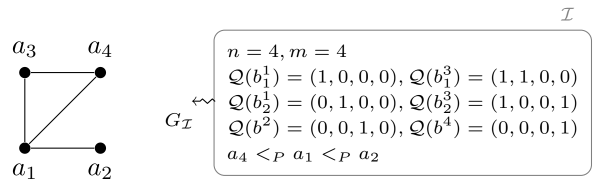
$\mathcal{I}$
a_3
a_4
a_1
a_2
$G_{\mathcal{I}}$
$n = 4, m = 4$
$\mathcal{Q}(b_1^1) = (1,0,0,0), \mathcal{Q}(b_1^3) = (1,1,0,0)$
$\mathcal{Q}(b_2^1) = (0,1,0,0), \mathcal{Q}(b_2^3) = (1,0,0,1)$
$\mathcal{Q}(b^2) = (0,0,1,0), \mathcal{Q}(b^4) = (0,0,0,1)$
$a_4 <_P a_1 <_P a_2$

Figure 2: Graph representations of a MRCPSP-instance. (A mode is in M_i if i is its upper index, e.g., $b_2^1 \in M_1$.)

In our initial analysis of the potential applications of treewidth, we will restrict our attention to a natural graph representation of a MRCPSP instance $\mathcal{I}$ which captures how activities may interact with each other. Notably, the *activity graph* $G_{\mathcal{I}}$ has vertex set A and edges represent precedences as well as the possibility of using the same resource—in particular, its edge set is $\{\, a_i a_j \mid (a_j <_P a_i) \vee (a_i <_P a_j) \vee (\exists b \in M_i, b' \in M_j, \ell \in [m] \quad \mathcal{Q}(b)[\ell] \neq 0 \wedge \mathcal{Q}(b')[\ell] \neq 0) \,\}$. An illustration is provided in Figure 2.

3 A Complexity Map for (M)RCPSP

In this section we give the polynomial-time algorithms and lower bounds (NP-hardness proofs) from which the complexity of all fragments obtained by considering any combination of considered flags follows (see Figure 1).

3.1 Polynomially Tractable Fragments

We present our six tractability results in an order roughly corresponding to the technical difficulty of the algorithms. Our first result is a simple observation identifying a basic polynomial-time fragment of MRCPSP.

Observation 1. $\mathrm{MRCPSP}\langle n\rangle$ is in P.

Proof Sketch. Branch over all permutations of the activities and all assignments of activities to their modes. In each branch, greedily build a solution which assigns activities to the selected modes and starts them in an order which does not violate the permutation, or decide that this is not possible, in quadratic time. The time complexity of this procedure lies in $\mathcal{O}(n! \prod_{i=1}^{n} |M_i| \cdot |\mathcal{I}|^2) \subseteq \mathcal{O}(|\mathcal{I}|^{n+2})$. ☐

The following fragment—consisting of simple instances without precedence constraints and with a bound on the number of activities that use any particular resource—can be solved via a reduction to the $\mathrm{MRCPSP}\langle n\rangle$ fragment.

Corollary 2. MRCPSP$\langle r_{\text{deg}}, S, \neg P\rangle$ is in P.

Proof Sketch. Since every activity uses at most a single resource (regardless of the mode it is set to), and since there are no precedence constraints between activities, an instance $\mathcal{I}$ of MRCPSP$\langle r_{\text{deg}}, S, \neg P, \rangle$ can be split into a set of independent instances, each containing a single resource and at most r_{deg} activities. Using the previous observation this yields an algorithm in $\mathcal{O}(m \cdot r_{\text{deg}}! |\mathcal{I}|^{r_{\text{deg}}+2}) \subseteq \mathcal{O}(|\mathcal{I}|^{r_{\text{deg}}+3})$. □

We now proceed to fragments with non-trivial algorithms.

Theorem 3. MRCPSP$\langle m, c, C_{\max}\rangle$ is in P.

Proof. Any solution (ω, α) to an instance $\mathcal{I}$ contains at most $q = C_{\max} \cdot c \cdot m$ modes which use at least one resource—i.e., the set $B_{>0} = \{\, b \in \omega(A) \mid \exists j \in \mathbb{N} : \mathcal{Q}(b)[j] > 0 \,\}$ has cardinality at most q. Let $A_{>0} = \{a \in A \mid \omega(a) \in B_{>0}\}$ be the set of activities using these modes in the solution.

We can solve $\mathcal{I}$ using the algorithm $\mathcal{A}$ which begins by branching over all subsets of modes of cardinality at most q containing at most one mode from each of the pairwise disjoint M_i as options for $B_{>0}$. From such a choice for a possible $B_{>0}$ we infer a corresponding ω by setting $\omega(a_i) = b$ for $a_i \in A_{>0}$ whenever $B_{>0} \cap M_i = \{b\}$, and for all other activities $a_i \in A \setminus A_{>0}$ choosing $\omega(a_i)$ as $b \in M_i$ such that $\mathcal{Q}(b) = 0^m$ (i.e., b requires no resources) and minimizes $T(b)$ among these modes. It is easy to see that, whenever a solution with the chosen $B_{>0}$ exists, a solution with the chosen $B_{>0}$ and deduced ω exists.

Now, we proceed similarly as in the proof of Observation 1 in which we branched on the order in which the activities are scheduled in a solution and then greedily constructed α which conforms to this ordering whenever such an α exists. The caveat here is that this exact approach would introduce a linear dependency on $n!$ which is in general not in $\text{poly}(|\mathcal{I}|)$. Instead, $\mathcal{A}$ branches only on the order in which the activities in $A_{>0}$ are scheduled by a solution, inserts the activities in $A \setminus A_{>0}$ into this ordering, at the respective smallest positions respecting the precedence relation, and then a greedy starting time assignment is performed just as before. The overall running time of $\mathcal{A}$ can be shown to lie in $\mathcal{O}(|B|^{C_{\max}\cdot c\cdot m} \cdot (C_{\max} \cdot c \cdot m)! \cdot |\mathcal{I}|^2) \subseteq \mathcal{O}(|\mathcal{I}|^{C_{\max}\cdot c\cdot m+2})$. □

The proof strategy for Theorem 3 can be combined with that of Observation 1 to obtain a polynomial-time algorithm when r_{deg} and m are bounded. The resulting algorithm runs in time $\mathcal{O}((r_{\text{deg}} \cdot m)! \cdot |\mathcal{I}|^{r_{\text{deg}}\cdot m+2})$.

Corollary 4. MRCPSP$\langle m, r_{\text{deg}}\rangle$ is in P.

The final two (and arguably most difficult) fragments for which we show polynomial-time tractability both have no precedence constraints and have boundedly many resources.

Theorem 5. MRCPSP$\langle m, \neg P, C_{\max}, U\rangle$ is in P.

Proof Sketch. Let a *resource snapshot* be a $C_{\max} \times m$ matrix over $[c] \cup \{0\}$ (i.e., the maximum capacity of a resource). Observe that the number of resource snapshots is upper-bounded by $(c+1)^{C_{\max}\cdot m}$. The resource snapshot J of a partial schedule (i.e., a solution restricted to a subset of activities) (ω, α) is the matrix where, for each $x \in [C_{\max}]$ and $y \in [m]$, the entry $J[x, y]$ equals the amount of resource r_y left at time step x.

Given an instance $\mathcal{I}$, let $\mathcal{J}_i$ be the set of resource snapshots of all partial schedules for the activities $\{\, a_j \mid j \leq i \,\}$. Clearly, $\mathcal{J}_0$ contains a single resource snapshot, namely the one where $J[x, y] = C(r_y)$ for all x, y. On the other hand, if $\mathcal{J}_n \neq \emptyset$ then $\mathcal{I}$ is clearly a YES-instance.

To prove the theorem, we describe a dynamic programming algorithm $\mathcal{A}$ which computes $\mathcal{J}_{i+1}$ from $\mathcal{J}_i$. $\mathcal{A}$ begins by looping over all resource snapshots in $\mathcal{J}_i$, branching over each mode $b \in M_{i+1}$ of activity a_{i+1} and branching over each starting time $s \in [C_{\max} - T(b)]$. For each such choice of resource snapshot J, b and s, it creates a new possible resource snapshot J'. If any entry of the constructed J' is negative, it is not a resource snapshot and hence not added to $\mathcal{J}_{i+1}$; otherwise J' is added to $\mathcal{J}_{i+1}$.

If the algorithm $\mathcal{A}$ results in a set $\mathcal{J}_{n+1}$ that is non-empty, we can reconstruct a solution from the run of the algorithm by standard means; otherwise we conclude that $\mathcal{I}$ is a NO-instance. Note that each combination of mode, starting time and resource snapshot is considered at most once when updating the resource snapshots. Hence time complexity lies in $\mathcal{O}(|B| \cdot C_{\max} \cdot (c+1)^{C_{\max}\cdot m}) \subseteq \mathcal{O}(|\mathcal{I}|^{C_{\max}\cdot m+1})$. □

Our last algorithm can be viewed as an extension of Theorem 5 to instances of larger makespan, by replacing the bound on the makespan by a weaker restriction, bounding t. This comes at a cost of requiring a bound on c.

Theorem 6. MRCPSP$\langle m, c, t, \neg P\rangle$ is in P.

Proof Sketch. We may assume w.l.o.g. that the image of $\mathcal{Q}$ is a subset of $[c]^m$ (modes mapped by $\mathcal{Q}$ outside of this range are irrelevant because of resource constraints).

Define the *type* of an activity $a_i \in A$, denoted $\tau(a_i)$, as $\{\, (\mathcal{Q}(b), T(b)) \mid b \in M_i \,\}$. Observe that the property of having the same type describes an equivalence relation between activities, which has at most $2^{c^m \cdot t}$ many equivalence classes, each of which we refer to as an *activity type*. Let $\mathcal{T}$ be the set of non-empty activity types.

If there is a solution, there is a solution (ω, α) such that $\max_{i\in[n]} \alpha(a_i) + T(\omega(a_i)) \leq t \cdot n$ and any activity with a mode b with $T(b) \leq C_{\max}$ which requires no resources is scheduled to start at time 0 using mode b. In such a solution at any time point between 0 and $t \cdot n$ at most $c \cdot m$ of the remaining activities are being processed concurrently as they have to be assigned to modes using at least some resource.

For the remaining activities we build up partial solutions ((ω', α') where ω' and α' are defined on a subset of A instead of A satisfying all constraints on that subset) along the time steps. We do so by backtracking on the choice of a multiset of at most $c \cdot m$ activity types and modes conforming to these activity types such that activities of these type may be scheduled using these modes in each time step. More formally, we iterate through $i = 0 \ldots \min\{t \cdot n, C_{\max}\} - 1$. Within this iteration we iterate through the activity types (with multiplicities) that can be scheduled at time step i. To determine these activity types and their multiplicities we maintain, for each partial solution constructed in each iteration, the resource snapshot $J \in ([c] \cup \{0\})^{C_{\max}\cdot m}$ (defined as in the proof of Theorem 5) induced by this partial solution and a vector $s \in \bigtimes_{\tau \text{ activity type}}([|\tau|] \cup \{0\})$, describing how many activities of each activity type are not yet in the domain of the partial solution. A multiset $\{\tau_1, \dots, \tau_z\}$ of $z \le c \cdot m$ activity types, can be scheduled at time step i if the multiplicity with which each activity type occurs in the multiset is bounded by the corresponding entry in s and there are $(Q_j, T_j) \in \tau_j$ such that subtracting all Q_j from the $(i+1)$-th through $(i+1+T_j)$-th rows of J does not result in negative entries in J. For each such choice of $\{(Q_j, T_j) \in \tau_j \mid j \in [z]\}$, we find an unscheduled activity a with $\tau(a) = \tau_j$ and can set $\omega(a) = b$ such that $(\mathcal{Q}(b), T(b)) = (Q_j, T_j)$ and $\alpha(a) = i$. Appropriate modifications for J and s are straightforward. If we complete iteration $\min\{t \cdot n, C_{\max}\} - 1$ without having scheduled all activities in any encountered solution, we can conclude that no schedule for the instance exists.

The described approach is an iterative branching procedure which is exhaustive modulo activity type equivalence. Hence correctness follows from the fact that activities of the same type can easily be interchanged in a schedule by an easy transformation. The complexity lies in $\mathcal{O}((\min\{t \cdot n, C_{\max}\} - 1) \cdot (2^{c^m \cdot t} \cdot |B|)^{c \cdot m} \cdot |\mathcal{I}|) \subseteq \mathcal{O}(|\mathcal{I}|^{c^m \cdot t + 2})$. □

3.2 Lower Bounds

We now turn towards hardness results for fragments of MRCPSP. First, we state a few previously known lower bounds:

Fact 7 (Uetz [2011], Lemma 5.1.1)**.** RCPSP$\langle c, r_{\deg}, t, \neg P, C_{\max}, U\rangle$ is NP-hard.

Fact 8 (Blazewicz, Lenstra and Kan [1983], Theorem 7)**.** RCPSP$\langle m, c, t, S, U\rangle$ is NP-hard.

The third and last known NP-hardness result that we need concerns the fragment RCPSP$\langle m, c, S, \neg P, U\rangle$. Du and Leung (1989, Theorem 2) proved that a scheduling problem equivalent to this fragment is NP-hard (one merely needs to represent the identical machines used in their reduction by capacity units of a single resource).

Fact 9. RCPSP$\langle m, c, S, \neg P, U\rangle$ is NP-hard.

Moreover, it is easy to observe that a trivial reduction from BIN PACKING [Garey and Johnson, 1979] yields:

Observation 10. RCPSP$\langle m, t, S, \neg P, U\rangle$ is NP-hard.

Our following three new reductions complete the complexity map for MRCPSP in terms of explicit restrictions.

Theorem 11. RCPSP$\langle c, r_{\deg}, t, S, C_{\max}, U\rangle$ is NP-hard.

Proof Sketch. We give a reduction from 3-SAT by constructing an instance $\mathcal{I}$ from a 3-CNF formula F as follows. $\mathcal{I}$ has $C_{\max} = 3$ and all processing times of activities 1. For each variable x in F we create a resource r_x with capacity one and two activities x_T, x_F, each requiring one of r_x. Moreover, for each clause C we create a resource r_C with capacity 3, and for each literal ℓ in C we create one activity C_ℓ which requires one r_C. If $\ell = x$ for some variable x (i.e., ℓ is a positive literal), we create the precedence constraint requiring C_ℓ to start after x_T is completed; otherwise we create the precedence constraint requiring C_ℓ to start after x_F is completed.

For each clause C, we now create three new activities C_0, C_1 and C_2, where $C_0 <_P C_1 <_P C_2$ and which require 0, 2 and 1 resources of type r_C, respectively. This completes our construction. To complete the proof, it suffices to verify that $\mathcal{I}$ is a YES-instance iff F is satisfiable. □

Theorem 12. RCPSP$\langle m, t, S, C_{\max}, U\rangle$ is NP-hard.

Proof Sketch. We give a polynomial-time reduction from the NP-hard CLIQUE problem, which asks whether a given graph contains a clique of a certain size. Given a graph $G = (V, G)$ and a natural number k (i.e., the desired clique size), we construct an RCPSP-instance $\mathcal{I}$ as follows. We create an activity a_v for every vertex v of G and an activity a_{vw} for every edge vw of G, and we then set $a_v <_P a_{vw}$ and $a_w <_P a_{vw}$. All activities require one time step to process, and we fix $C_{\max} = 3$.

The idea of the set-up we describe in the following is to restrict the activities that can be scheduled to start at the first time step to exactly k activities that correspond to vertices. These k vertices should correspond to the vertices of a clique in G. In the next time step, the $\frac{k \cdot (k-1)}{2}$ activities corresponding to the edges of that clique and the remaining vertex activities can be scheduled, allowing for the remaining edge activities to be scheduled in the last time step that $C_{\max} = 3$ allows for. For this to work, all we need to do is to restrict the number of vertex activities that can be scheduled to start in the first time point to be exactly k, the number of vertex activities which can start in the second time point to be exactly $|V| - k$ and the number of edge activities that can be scheduled to start in the third time point to be exactly $|E| - \frac{k \cdot (k-1)}{2}$.

We do this by setting up the resources and resource requirements as follows and introducing 'filler' activities. We consider two resources r_1 and r_2 with capacities $|V|$ and $|E|$, respectively. Each a_v will require one of r_1 and each a_{vw} will require one of r_2. We introduce three further activities a_1, a_2, a_3, each requiring one time step, and set $a_1 <_P a_2 <_P a_3$; a_1 requires $|V| - k$ of r_1, a_2 requires k of r_1, a_3 requires $\frac{k \cdot (k-1)}{2}$ of r_2. It is easy to verify that $\mathcal{I}$ has a solution iff G has a clique of size k. □

Theorem 13. RCPSP$\langle m, t, S, \neg P, C_{\max}\rangle$ is NP-hard.

Proof. This time, we start from the weakly NP-hard PARTITION problem [Garey and Johnson, 1979]: decide whether a given multiset $S = \{m_1, m_2, \dots, m_n\}$ of positive integers such that $\sum_{m_i \in S} = 2b$ can be partitioned into two subsets S_1, S_2 such that $\sum_{m_i \in S_1} = \sum_{m_i \in S_2} = b$.

Given an instance of PARTITION as described above, we create an instance $\mathcal{I}$ of RCPSP with a single resource of capacity b. For each number $m_i \in S$, we now create an activitiy a_i with duration 1 which requires m_i-many units of our resource. It is now easy to see that the PARTITION instance has a solution iff $\mathcal{I}$ has a makespan of 2: indeed, there is a one-to-one correspondence between the activities scheduled at time 0 (in a schedule with makespan 2) and the numbers assigned to S_1 (in a solution to PARTITION). □

3.3 Summary and Discussion

Note that the following modifications leave instances invariant with respect to polynomial-time solvability: (1) in simple instances without precedence constraints one can assume m to be bounded (see the argument used for Corollary 2), and (2) in instances with bounded $C_{\max}$ one can assume t to be bounded. These two simple observations together with a complete enumeration of all combinations of flags yield that the 6 polynomial-time algorithms, give rise to a total of 736 fragments of MRCPSP which are in P. Similarly, the 7 hardness results imply a total of 288 NP-hard fragments of MRCPSP. This completely settles the complexity of all fragments of the problem defined in terms of the 10 considered flags.

4 Solving (M)RCPSP via Structural Restriction

Here, we use the structure of interactions between activities to push beyond the frontiers of tractability delimited by the complexity map based on explicit restrictions of instance parameters. As mentioned in the introduction, this approach has been very successful for many other prominent problems, and we believe it is highly promising also for (M)RCPSP. However, due to the sheer volume of possible cases and fragments to consider, the result presented in this section should be viewed primarily as a "proof of concept" and, perhaps, the tip of a (potentially very large) iceberg. Indeed, a thorough investigation of how the structure of instances can be algorithmically exploited is beyond the scope of this work.

Theorem 14. MRCPSP$\langle U\rangle$ can be solved in time $\mathcal{O}(|\mathcal{I}|^{5\text{tw}(G_{\mathcal{I}})})$, where $\mathcal{I}$ is the input instance.

We note that Theorem 14 is a generalization of Observation 1 when dealing with unary instances, since the activity graphs of instances in the MRCPSP$\langle n, U\rangle$ fragment have boundedly-many vertices. Similarly, each connected component in the activity graph of an instance in MRCPSP$\langle r_{\text{deg}}, S, \neg P, U\rangle$ has boundedly-many vertices, and so the result also generalizes Corollary 2 in the unary setting.

Proof Sketch of Theorem 14. We begin by computing a nice tree-decomposition $\mathcal{T} = (T, \mathcal{X})$ of width $k = \text{tw}(G_{\mathcal{I}})$ [Arnborg *et al.*, 1987]. Let a configuration $\beta(t)$ of a node t in T be a tuple $(\texttt{Mode}, \texttt{Time}, \texttt{Mkspan})$ where

- $\texttt{Mode}$ is a mapping from each activity $a_i \in X_t$ to a mode in M_i,
- $\texttt{Time}$ is a mapping from X_t to $[C_{\max} - 1]$, and
- $\texttt{Mkspan}$ is an integer from $[C_{\max}]$.

Intuitively, we use configurations to store possible ways of assigning modes and starting times of activities in X_t that allow to schedule activities in $\mathcal{X}^{\downarrow}(t)$ with makespan $\texttt{Mkspan}$. For $t \in V(T)$ we let the record $\mathcal{R}(t)$ consist of all admissible configurations of $\mathcal{X}^{\downarrow}(t)$, i.e., $(\texttt{Mode}, \texttt{Time}, \texttt{Mkspan}) \in \mathcal{R}(t)$ iff there is an assignment (satisfying all precedence and other constraints) (ω', α') of the activities in $\mathcal{X}^{\downarrow}(t)$ with makespan $\texttt{Mkspan}$ such that $\texttt{Mode}$ and $\texttt{Time}$ are the restrictions of ω' to X_t and α' to X_t respectively.

As the total number of configurations is upper-bounded by ${C_{\max}}^{k+1} \cdot |B|^k$ and the instance is unary, $|\mathcal{R}(t)|$ is bounded by a polynomial in $|\mathcal{I}|$. Moreover it is easy to compute $\mathcal{R}(t)$ for any leaf t of T by brute-forcing over all assignments and modes of the single activity in that leaf. $\mathcal{I}$ is a YES-instance iff the record $\mathcal{R}(r)$ for the root r is non-empty—moreover, in this case it is easy to reconstruct a solution to $\mathcal{I}$ by backtracking from the root r to determine which entries in the records lead to a non-empty $\mathcal{R}(r)$. Hence, in order to complete the proof it suffices to show how to compute the records for forget, join and introduce nodes.

If t is a forget node with child t' such that $X_{t'} \setminus X_t = \{a_i\}$, then for each configuration $\beta(t') \in \mathcal{R}(t')$ we compute a configuration $\beta(t)$ by removing a_i from the two mappings in $\beta(t')$. We add each such computed configuration to $\mathcal{R}(t)$.

If t is an introduce node with child t' such that $X_t \setminus X_{t'} = \{a_i\}$, then we branch over all mappings $\omega^*(a_i) \in M_i$ and $\alpha^*(a_i) \in [C_{\max} - 1]$. In each branch and for each record $(\texttt{Mode}, \texttt{Time}, \texttt{Mkspan}) \in \mathcal{R}(t')$, we check whether the instance contains sufficient resources for the activities in X_t to be scheduled at times $\alpha^* \cup \texttt{Time}$ and in modes $\omega^* \cup \texttt{Mode}$. Moreover, we check that a_i satisfies the precedence constraints w.r.t. the other activities in X_t. If both checks are successful, we check when a_i ends based on the current choice of α^* and ω^*, and we update $\texttt{Mkspan}$ accordingly (naturally, we discard records where a_i ends after $C_{\max}$). We then add the configuration with the new $\texttt{Mkspan}$ and with the mappings $\omega^* \cup \texttt{Mode}$, $\alpha^* \cup \texttt{Time}$ to $\mathcal{R}(t)$.

If t is a join node with children t', t'', then we check compatibility (i.e., equality) of every pair of elements of $\mathcal{R}(t')$ and $\mathcal{R}(t'')$ and add compatible elements to $\mathcal{R}(t)$.

The running time of these steps is dominated by the running time of the join node, which requires time at most $({C_{\max}}^{k+1} \cdot |B|^k)^2 \cdot |\mathcal{I}|$. Hence, the total running time of the algorithm is upper-bounded by $\mathcal{O}(|\mathcal{I}|^{5k})$. $\square$

5 Concluding Remarks

We introduced a series of new algorithmic upper and lower bounds that together paint a complete picture of the classical complexity of (M)RCPSP in terms of explicit restrictions on its instances. An extension of RCPSP which we did not directly address in this work is RCPSP/max, where instead of simple precedence constraints one can specify a desired maximum and minimum time gap between finishing one and starting another activity. Naturally, all our lower bounds also carry over to this more general problem. Moreover, the three positive results for fragments with precedence constraints (Observation 1, Theorem 3 and Corollary 4) extend to the RCPSP/max setting with minimal changes.

It would be interesting to refine the obtained complexity map from the *parameterized complexity* viewpoint [Downey and Fellows, 2013; Cygan *et al.*, 2015]. In particular, most of the tractability results presented in this paper do not readily translate to *fixed-parameter tractability*, and it would certainly be worthwhile to determine which parameterizations of (M)RCPSP give rise to fixed-parameter algorithms.

Finally, we also indicated how one can algorithmically exploit the structural properties of activity interactions through the use of graph representations and structural parameters. We believe this is a promising direction for future research; for instance, a similar approach could also be used to consider graph representations of interactions between resources.

Acknowledgments

R. Ganian and T. Hamm acknowledge support by the Austrian Science Fund (FWF, project P31336). T. Hamm also acknowledges support by the FWF project W1255-N23.

References

[Arnborg *et al.*, 1987] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, 8(2):277–284, 1987.

[Artigues *et al.*, 2008] Christian Artigues, Sophie Demassey, and Emmanuel Neron. *Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications*. ISTE/Wiley, 2008.

[Barrios *et al.*, 2011] Agustín Barrios, Francisco Ballestín, and Vicente Valls. A double genetic algorithm for the mrcpsp/max. *Computers & OR*, 38(1):33–43, 2011.

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

[Bodlaender *et al.*, 2016] Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A c^k n 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016.

[Bofill *et al.*, 2017] Miquel Bofill, Jordi Coll, Josep Suy, and Mateu Villaret. An efficient SMT approach to solve mrcpsp/max instances with tight constraints on resources. In *CP 2017*, pages 71–79, 2017.

[Cohen *et al.*, 2015] David A. Cohen, Martin C. Cooper, Peter G. Jeavons, and Stanislav Zivny. Tractable classes of binary csps defined by excluded topological minors. In *IJCAI 2015*, pages 1945–1951, 2015.

[Cygan *et al.*, 2015] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[Damay *et al.*, 2007] Jean Damay, Alain Quilliot, and Eric Sanlaville. Linear programming based algorithms for preemptive and non-preemptive RCPSP. *European J. of Operational Research*, 182(3):1012–1022, 2007.

[Downey and Fellows, 2013] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[Du and Leung, 1989] Jianzhong Du and Joseph Y.-T. Leung. Complexity of scheduling parallel task systems. *SIAM J. Discrete Math.*, 2(4):473–487, 1989.

[Fu *et al.*, 2010] Na Fu, Pradeep Varakantham, and Hoong Chuin Lau. Towards finding robust execution strategies for rcpsp/max with durational uncertainty. In *ICAPS 2010*, pages 73–80, 2010.

[Fu *et al.*, 2016] Na Fu, Pradeep Varakantham, and Hoong Chuin Lau. Robust partial order schedules for rcpsp/max with durational uncertainty. In *ICAPS 2016*, pages 124–130, 2016.

[Ganian and Ordyniak, 2018] Robert Ganian and Sebastian Ordyniak. The complexity landscape of decompositional parameters for ILP. *Artif. Intell.*, 257:61–71, 2018.

[Garey and Johnson, 1975] M. R. Garey and David S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM J. Comput.*, 4(4):397–411, 1975.

[Garey and Johnson, 1979] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[Gehrke *et al.*, 2018] Jan Clemens Gehrke, Klaus Jansen, Stefan E. J. Kraft, and Jakob Schikowski. A PTAS for scheduling unrelated machines of few different types. *Int. J. Found. Comput. Sci.*, 29(4):591–621, 2018.

[Gottlob *et al.*, 2002] Georg Gottlob, Francesco Scarcello, and Martha Sideri. Fixed-parameter complexity in AI and nonmonotonic reasoning. *Artif. Intell.*, 138(1-2):55–86, 2002.

[Kloks, 1994] Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *LNCS*. Springer, 1994.

[Kolisch and Padman, 2001] R. Kolisch and R. Padman. An integrated survey of deterministic project scheduling. *Omega*, 29(3):249–272, 2001.

[Kuster *et al.*, 2007] Jürgen Kuster, Dietmar Jannach, and Gerhard Friedrich. Handling alternative activities in resource-constrained project scheduling problems. In *IJCAI 2007*, pages 1960–1965, 2007.

[Poppenborg and Knust, 2016] Jens Poppenborg and Sigrid Knust. Modeling and optimizing the evacuation of hospitals based on the MRCPSP with resource transfers. *EURO J. Computational Optimization*, 4(3-4):349–380, 2016.

[Schwindt and Zimmermann, 2015] Christoph Schwindt and Jürgen Zimmermann. *Handbook on Project Management and Scheduling Vol.1*. Springer, 2015.

[Smith and Pyle, 2004] Tristan B. Smith and John M. Pyle. An effective algorithm for project scheduling with arbitrary temporal constraints. In *AAAI 2004*, pages 544–549, 2004.

[Song, 2017] Wen Song. Project scheduling in complex business environments. In *AAAI 2017*, pages 5052–5053, 2017.

[Thorup, 1998] Mikkel Thorup. All structured programs have small tree-width and good register allocation. *Inf. Comput.*, 142(2):159–181, 1998.

[Uetz, 2011] Marc Uetz. *Algorithms for Deterministic and Stochastic Scheduling*. PhD thesis, 2011.

[van Bevern *et al.*, 2016] René van Bevern, Robert Bredereck, Laurent Bulteau, Christian Komusiewicz, Nimrod Talmon, and Gerhard J. Woeginger. Precedence-constrained scheduling problems parameterized by partial order width. In *DOOR 2016*, pages 105–120, 2016.

[Varakantham *et al.*, 2016] Pradeep Varakantham, Na Fu, and Hoong Chuin Lau. A proactive sampling approach to project scheduling under uncertainty. In *AAAI 2016*, pages 3195–3201, 2016.