Algorithms and Complexity Group | Institute of Logic and Computation | TU Wien, Vienna, Austria

Technical Report AC-TR-19-002

August 2019

A* Search for Prize-Collecting Job Sequencing with One Common and Multiple Secondary Resources

Matthias Horn, Günther R. Raidl and Elina Rönnberg

A* Search for Prize-Collecting Job Sequencing with One Common and Multiple Secondary Resources[*]

Matthias Horn[1], Günther Raidl[1],
Elina Rönnberg[2]

[1]Institute of Computer Graphics and Algorithms, TU Wien, Austria

[2]Department of Mathematics, Linköping University, Sweden

{horn|raidl}@ac.tuwien.ac.at, elina.ronnberg@liu.se

We consider a sequencing problem with time windows, in which a subset of a given set of jobs shall be scheduled. A scheduled job has to execute without preemption and during this time, the job needs both a common resource for a part of the execution as well as a secondary resource for the whole execution time. The common resource is shared by all jobs while a secondary resource is shared only by a subset of the jobs. Each job has one or more time windows and due to these, it is not possible to schedule all jobs. Instead, each job is associated with a prize and the task is to select a subset of jobs which yields a feasible schedule with a maximum sum of prizes. First, we argue that the problem is NP-hard. Then, we present an exact A* algorithm and derive different upper bounds for the total prize; these bounds are based on constraint and Lagrangian relaxations of a linear programming relaxation of a multidimensional knapsack problem. For comparison, a compact mixed integer programming (MIP) model and a constraint programming model are also presented. An extensive experimental evaluation on three types of problem instances shows that the A* algorithm outperforms the other approaches and is able to solve small to medium size instances with up to about 40 jobs to proven optimality. In cases where A* does not prove that an optimal solution is found, the obtained upper bounds are stronger than those of the MIP model.

Keywords. Sequencing A* algorithm particle therapy patient scheduling

[*]This project is partially funded by the Doctoral Program "Vienna Graduate School on Computational Optimization", Austrian Science Foundation (FWF) Project No. W1260-N35. The work of Elina Rönnberg is supported by the Center for Industrial Information Technology (CENIIT), Project-ID 16.05. We further thank Lukas Felician Krasel for his help in the implementation and testing.

1 Introduction

Horn et al. (2017) introduced the *Job Sequencing with One Common and Multiple Secondary Resources* (JSOCMSR) problem, which considers the scenario of scheduling a set of jobs where each job, for part of its execution, requires a common resource and, for the whole processing time, requires a secondary resource that is only shared with a subset of the other jobs. The goal of the JSOCMSR is to find a feasible schedule minimizing the makespan.

Applications of this problem can be found, for example, in manufacturing when the common resource corresponds to a machine on which the jobs are to be processed and the processing of each job also requires a certain accompanying resource like a casting mold, template etc. The important aspect is that there is a setup time (e.g. a preparation of the casting mold) at which the secondary resource is needed before the job can be executed at the machine and also a postprocessing time during which the secondary resource is needed afterwards (e.g. because the produced goods must be cooled in the casting mold). With job-dependent setup, processing and postprocessing times and a limited number of shared secondary resources, the JSOCMSR has been shown to be NP-hard (Horn et al. 2017).

A more specific application of this problem can be found in the daily scheduling of cancer patients that are to receive particle therapy (Maschler et al. 2016). There, the common resource corresponds to a sophisticated particle accelerator, which accelerates proton or carbon particles to almost the speed of light. This particle beam is directed into one of a small set of treatment rooms where one patient can be radiated at a time. The treatment rooms are here the secondary resources. During the setup time, a patient is positioned and possibly sedated and after the actual radiation with the particle beam, typically some medical examinations are to be done before the patient can leave the room and it becomes available for a next patient. In such particle therapy treatment centers, there is usually only a single particle accelerator because of its huge cost and about two to three treatment rooms. Since these treatment rooms are typically individually equipped for handling different kinds of treatments, the assignment of patients to rooms is pre-specified. Ideally, patients are scheduled in such a way that the particle beam is directly switched from one room to another so that patients are radiated without any significant breaks in between.

However, the JSOCMSR is in most cases only a strongly simplified model of real-world scenarios like the above patient scheduling. Most notably, the jobs start times are in practice frequently constrained to certain time windows arising from the availability of the underlying resources. Furthermore, in practice, it happens frequently that not all jobs can be scheduled due to these time windows and instead, a best subset of jobs that can be feasibly scheduled must be selected.

To also include such aspects is the focus of the current work: We extend the JSOCMSR by considering job-specific time windows, and instead of minimizing the makespan we aim at finding a feasible solution that maximizes the total prize of all scheduled jobs. To this end, each job has an assigned prize, which can simply take the value one if we want to maximize the number of scheduled jobs or it can take a value representing the priority of the job. Another possibility is that these prizes are correlated to the processing times of the jobs to avoid scheduling primarily short jobs.

These new aspects have a substantial impact on the structure of the problem and consequently on the algorithmic side, and existing methods for the JSOCMSR cannot be extended in efficient ways. Most importantly, the rather effective lower bound calculation for the makespan in the JSOCMSR we proposed in Horn et al. (2017) is useless for the new problem variant due to the entirely different objective function and new decision variables. Therefore we propose here a new A* search to solve this prize-collecting variant of the JSOCMSR to proven optimality. In particular, we investigate four different upper bound calculations for partial solutions which are used as heuristic functions to estimate the costs-to-go within the A* search. A further aspect of our A* search is that there is no specific target state known in advance.

An early version of this approach was already presented in our conference paper (Horn et al. 2018b). The current article extends this work by primarily considering an additional application scenario and corresponding new benchmark instances: pre-runtime scheduling of electronics within an aircraft, called avionics. The instances we include here are inspired by a sub-structure of the problem introduced in Blikstad et al. (2018), and the focus here is the part of the structure that coincides with that of the patient scheduling problem. We further remark that we fixed a bug in our implementation after the publication of (Horn et al. 2018b) and consequently reran all experiments; while the general trends remain the very same, detailed numbers have changed.

In Blikstad et al. (2018), the scheduling of an industrially relevant avionic system is addressed. The considered system consists of a set of nodes and each of these contains a set of modules (processors) with jobs to be scheduled. Each node consists of one communication module, corresponding to the common resource, and a set of application modules, corresponding to the set of secondary resources. There are three types of jobs: partition jobs, communication jobs and regular jobs. Partition jobs, which are executed on the application modules run the system's software applications. Each of these jobs has to communicate with the communication module and we consider only the case where a partition job has to communicate either at the beginning or at the end of its execution. Typically, the processing time of partition jobs is long compared to other jobs and its usage of the common resource is short compared to the total processing time of the job.

The communication is handled by communication jobs and regular jobs, both executed on the communication module. These have short processing times and in order to model that they use the common resource only, an additional artificial secondary resource is included for these jobs. The communication and regular jobs use both the common resource and the artificial secondary resource for their whole processing time. Communication jobs and regular jobs together handle different kinds of communication (system external, inter- and intranode) and the communication jobs have the specific purpose of representing jobs used for sending communication messages (Blikstad et al. 2018). For this reason, the communication jobs can only be scheduled at specific time slots where communication messages can be sent and the length of a communication job's time window is equal to the job's total processing time. This distinguish them from the regular jobs which have time windows with similar characteristics as those of the partition jobs.

To mimic the situation of creating a partial schedule for a node in the system, only a part of the total length of a schedule is considered and the tasks available exceed what is possible to include in the partial schedule. The prize of a job reflects the individual importance of including this job in the partial schedule. Note that compared to Blikstad et al. (2018), some simplifications are made with respect to the types of jobs included and by omitting precedence relations between jobs. Furthermore, we do not explicitly and fully consider the scheduling of the communication network used for communication between the nodes.

After a more formal problem definition in the next section and a survey of related work in Section 3, we describe the A* search in Section 4. This method relies on a specific state strengthening procedure and the use of a good heuristic guidance function corresponding to an upper bound calculation for the total prize that may still be a achieved from a partial solution onward. Concerning the latter, we investigate different possibilities based on linear programming and Lagrangian relaxations of a multidimensional knapsack problem formulation. For comparison purposes, we further consider, in Section 5, an order-based Mixed Integer Programming (MIP) model solved by Gurobi Optimizer Version 7.5.1 and, in Section 6, a constraint programming model solved by MiniZinc. Section 7 presents and compares computational results for all these approaches on instances of the avionics scenario as well as balanced and skewed instances of the particle therapy scenario. The results show that the proposed A* search can solve substantially larger instances to optimality in shorter times than the other methods. Section 8 concludes this work with an outline of promising future research directions.

2 Problem Formulation

The *Prize-Collecting Job Sequencing with One Common and Multiple Secondary Resources with Time Windows* (PC-JSOCMSR) considers the sequencing of a set of jobs where each job needs to respect resource constraints and time windows. The resource constraints refer both to a common resource that is used by all jobs and a set of secondary resources of which each job uses exactly one. It is assumed that it is usually not possible to find a feasible schedule that includes all jobs; instead each job is associated with a prize and the objective is to choose a subset of the jobs such that the sum of prizes of the sequenced jobs is maximized.

Let the set of all jobs be denoted by J, with $|J| = n$, and let the prize of job j be $z_j > 0$, $j \in J$. The problem is to find a subset of jobs $S \subseteq J$ that can be feasibly scheduled so that the total prize of these jobs is maximized:

$$Z^* = \max_{S \subseteq J} Z(S) = \max_{S \subseteq J} \sum_{j \in S} z_j. \tag{1}$$

The set of (renewable) resources is denoted by $R_0 = \{0\} \cup R$, with $R = \{1, \ldots, m\}$. During its execution, job j uses resource 0, referred to as the *common resource*, and one of the secondary resources $q_j \in R$, $j \in J$. Let $p_j > 0$ be the processing time of job j, during which it fully requires the secondary resource q_j, $j \in J$. Further, let $J_r = \{j \in J \mid q_j = r\}$ be the set of jobs that require resource r, $r \in R$. For job j, $j \in J$, the use of the common resource begins $p_j^{\text{pre}} \geq 0$ time units after the start of the job, has a duration of p_j^0, and ends $p_j^{\text{post}} = p_j - p_j^{\text{pre}} - p_j^0 \geq 0$ time units before the end of the job.

If a job j is scheduled, it must be performed without preemption and within one of its ω_j disjunctive time windows $W_j = \{W_{jw} \mid w = 0, \ldots, \omega_j\}$ with $W_{jw} = [W_{jw}^{\text{start}}, W_{jw}^{\text{end}}]$, where $W_{jw}^{\text{end}} - W_{j,w}^{\text{start}} \geq p_j$, $j \in J$. We assume that each job has at least one time window. For job j, let the release time be $T_j^{\text{rel}} = \min_{w=0,\ldots,\omega_j} W_{jw}^{\text{start}}$ and the deadline be $T_j^{\text{dead}} = \max_{w=0,\ldots,\omega_j} W_{jw}^{\text{end}}$. The overall time interval to consider is then $\left[T^{\min}, T^{\max}\right]$ with $T^{\min} = \min_{j \in J} T_j^{\text{rel}}$ and $T^{\max} = \max_{j \in J} T_j^{\text{dead}}$. Note that the existence of unavailability periods of resources is also covered by the above formulation since these can be translated into time windows of the jobs.

To simplify the consideration of the time windows of a job, we define the function *earliest feasible time*

$$\text{eft}(j, t) = \min(\{t' \geq t \mid \exists w : \left[t', t' + p_j\right] \subseteq W_{jw}\} \cup \{T^{\max}\}), \tag{2}$$

that yields the earliest time not smaller than the provided time $t \leq T^{\max}$ at which job j can be scheduled according to the time windows W_j, $j \in J$. Hereby, $\text{eft}(j, t) = T^{\max}$ indicates that job j cannot be feasibly included in the schedule anymore.

Since each job requires resource 0 and only one job can use this resource at a time, a solution to PC-JSOCMSR implies a total ordering of the scheduled jobs S. Vice versa, a permutation $\pi = (\pi_i)_{i=1,\ldots,|S|}$ of a subset of jobs $S \subseteq J$ that can be feasibly scheduled can be decoded into a feasible schedule in a straight-forward greedy way by, in the order given by π, placing each job from S at its earliest feasible time with respect to when the resources are available after being used by all its preceding jobs. A schedule derived from a job permutation π in this way is referred to as a *normalized schedule*. Note that if this greedy approach is applied to a permutation of jobs and some job cannot be feasibly scheduled in this way, this permutation does not correspond to a feasible solution. Also, an optimal solution is either a normalized schedule or the order of the jobs in this optimal solution can be used to derive a normalized schedule with the same objective value. For this reason the notation $Z(\pi)$ is used for the total prize of the normalized solution given by the job permutation π. Figure 1 in Section 4.1 shows an example of an instance with four jobs and two secondary resources where each job has exactly one time window and a corresponding optimal solution.

It is not difficult to see that the PC-JSOCMSR is NP-hard: The decision variant of the JSOCMSR, which looks for a feasible schedule with a makespan not exceeding a given M, has already been shown to be NP-hard in Horn et al. (2017). One can reduce this decision problem to the PC-JSOCMSR in polynomial time by setting all time windows to $W_j = \{[0, M]\}$ and all prizes to $z_j = 1$. A solution to the JSOCMSR decision problem exists if and only if a solution to the PC-JSOCMSR can be found that has all jobs scheduled.

Instance:

j	p_j	p_j^{pre}	p_j^0	q_j	z_j	W_j
1	4	1	2	1	2	$\{[0,8]\}$
2	4	1	2	1	4	$\{[0,8]\}$
3	4	0	3	2	3	$\{[3,8]\}$
4	5	1	3	2	2	$\{[8,14]\}$

Optimal Solution π: $Z(\pi) = 9$

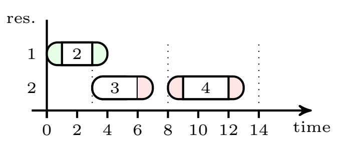

Figure 1: A PC-JSOCMSR instance with $n = 4$ jobs and $m = 2$ secondary resources. Each job has exactly one time window in W_j. The optimal solution is given by the job sequence $\pi = (2, 3, 4)$ and its normalized schedule visualized on the right side. The white region in each job denotes the part where, in addition to the job's specific secondary resource, also the common resource is needed. Job 1 cannot be additionally scheduled due to the time windows and resource requirements. The solution's total prize is $Z(\pi) = 9$.

3 Related Work

The JSOCMSR was originally proposed by Horn et al. (2017). There, a greedy heuristic, an A* algorithm, and a position-based MIP model are described. A particular contribution of that work is a method for calculating a relatively tight lower bound for the makespan, given a partial solution and the still open jobs. Experimental results show that thanks to this strong bound, already the greedy heuristic yields relatively good results by quickly deriving solutions with optimality gaps of only a few percent. Further, the A* search is capable of solving instances with up to 1000 jobs to proven optimality. In comparison, the presented MIP approach was not competitive and can only solve instances with up to 20 jobs reasonably well and it then requires substantially longer computation times. The lower bound calculation from this previous work is of no use for our PC-JSOCMSR due to the difference in objective function, the fact that usually not all jobs can be scheduled and the time windows. Note further that it is also not efficiently possible or straightforward to adapt the position based MIP model from Horn et al. (2017) to the PC-JSOCMSR due to the time windows.

Concerning the PC-JSOCMSR, Maschler and Raidl (2018) investigated heuristic methods based on multivalued decision diagrams (MDDs) and general variable neighborhood search for addressing larger problem instances with up to 300 jobs, which are so far clearly out of reach to be solved to proven optimality. Furthermore, Horn et al. (2018a) extends the works of Horn et al. (2018b) and Maschler and Raidl (2018) by proposing a novel construction method for relaxed MDDs. These relaxed MDDs are then further used to substantially speed up a heuristic search. In this way, new state-of-the-art results could be obtained for instances with up to 500 jobs. While all these works concentrated on heuristic solution approaches, the current work focuses on solving small to medium-sized instances exactly.

To the best of our knowledge, there are only a few further publications dealing with other scenarios similar to the (PC-)JSOCMSR. Van der Veen et al. (1998) considers a setup comparable to JSOCMSR with jobs requiring one common resource and individual secondary resources. However, in their case the postprocessing times are negligible compared to the total processing times of the jobs. This implies that the start time of each job only depends on its immediate predecessor. This simplifies the situation substantially, since a job j requiring a different resource than its predecessor j' can always be started after a setup time only depending on job j, while a job requiring the same resource can always be started after a postprocessing time only depending on job j'. Due to these properties, this problem can be interpreted as a *Traveling Salesman Problem* (TSP) with a special cost structure. Van der Veen et al. even show that their problem can be solved efficiently in time $O(n \log n)$.

Somehow related to the JSOCMSR are no-wait flowshop problem variants, see Allahverdi (2016) for a survey. Each job needs to be processed on each of m machines in the same order and the processing of the job on a successive machine always has to take place immediately after its processing has finished on the preceding machine. This problem can be solved in time $O(n \log n)$ for two machines via a transformation to a specially structured TSP (Gilmore and Gomory 1964). In contrast, for three and more machines the problem is NP-hard, although it can still be transformed into a specially structured TSP. Röck (1984) proved that the problem is strongly NP-hard for three machines by a reduction from the 3D-matching problem.

Furthermore, the JSOCMSR problem can be modeled as a more general Resource-Constrained Project Scheduling Problem with maximal time lags by splitting each job according to the resource usage into three sub-jobs that must be executed sequentially without any time lags; see Hartmann and Briskorn (2010) for a survey. Such an indirect solution approach, however, is unlikely to yield promising results in practice since problem-specific aspects are not exploited.

Moreover the PC-JSOCMSR is to some extent related to the well studied *Orienteering Problem* (OP), which essentially also combines the tasks of selecting a subset yielding a maximum prize with finding a sequence of the selected elements that make the solution feasible. Different variants of the OP have been studied, including OPs with time windows; for a survey see (Gunawan et al. 2016). In the OP, each node is associated with a prize and travel times are known between all pairs of nodes. The task is to find a path from a given start node to an end node within a given time budget such that the total prize of the visited nodes is maximized. Due to the time budget not all nodes can be visited. In such an OP, the arrival time at a visited node only depends on the immediate predecessor, possible time windows, and the (constant) travel time between these two nodes. For PC-JSOCMSR, the situation is more complicated due to the secondary resources: A much earlier scheduled job requiring the same secondary resource may impact the earliest starting time of the job to be scheduled next. The PC-JSOCMSR, however, also does not generalize the OP with time windows as pairwise travel times are not covered by the PC-JSOCMSR.

Concerning particle therapy patient scheduling, the PC-JSOCMSR is a practically relevant improvement over the simpler JSOCMSR model, but it still is a strongly simplified formulation addressing only certain properties of the real-life problem. In the full practical scenario, many more aspects must be considered such as large time horizons of several weeks, sequences of therapies for patients to be treated, additionally needed resources including medical staff and their availability time windows, and a combination of more advanced objectives and diverse soft constraints. Maschler et al. (2016) proposed a greedy construction heuristic which is extended towards an *Iterated Greedy* metaheuristic and a *Greedy Randomized Adaptive Search Procedure* (GRASP), which consider more of these advanced aspects. These approaches treat the whole problem as a bi-level optimization problem in which the upper level is concerned with the assignment of treatments to days and the lower level corresponds to the detailed scheduling of the treatments assigned at each day. The Iterated Greedy metaheuristic was further refined by including an improved makespan estimation for the daily scheduling problems and by considering additional soft constraints for unwanted deviations of start times of the individual treatments for each therapy (Maschler et al. 2018, 2017).

Concerning A*, we point out that it is a well-known and prominent method for finding shortest paths in possibly huge graphs and more generally an informed search method for problem-solving, see Hart et al. (1968), Rios and Chaimowicz (2010). Whereas the classical A* algorithm follows a best-first-search strategy with the goal to find a proven optimal solution as quickly as possible, anytime A* algorithms also producing promising heuristic solutions on a more regular basis from the beginning on; see, for example, the Anytime Weighted A* algorithm (Hansen and Zhou 2007), the Anytime Repairing A* (Likhachev et al. 2004), Anytime Pack Search (Vadlamudi et al. 2016), and the A*/Beam Search hybrid described for the JSOCMSR (Horn et al. 2017).

4 An A* Algorithm for the PC-JSOCMSR

The A* algorithm is a classic search algorithm from the field of path planning on possibly huge graphs (Hart et al. 1968, Rios and Chaimowicz 2010). In this section, we describe our A* search approach to solve the PC-JSOCMSR problem. The method either yields a proven optimal solution or, in case of an early termination, a heuristic solution together with an upper bound to the optimal solution value. We start by describing the state graph on which the search is performed, continue with the framework of our A* algorithm, and focus in Sections 4.3 and 4.4 on the strengthening of obtained states and propose different possibilities to determine upper bounds for the achievable total prizes of partial solutions, respectively.

4.1 State Graph

We consider a weighted directed acyclic state graph $G = (V, A)$ where each node in V represents a unique state (P, t) consisting of

- the set $P \subseteq J$ of jobs that are still available to be scheduled in further steps, and
- the vector $t = (t_r)_{r \in R_0}$ of the earliest times from which each of the resources are available for performing a next job.

The initial (root) state is $\mathbf{r} = (J, (T^{\min}, \ldots, T^{\min}))$ and represents the original PC-JSOCMSR problem instance with no jobs scheduled or excluded yet.

An arc $(u, v) \in A$ represents a transition from a state $u = (P, t)$ to a state $v = (P', t')$ that is achieved by scheduling a job j, $j \in P$, at its earliest possible time w.r.t. vector t. More precisely, the *start time* of job j, $j \in P$, w.r.t. state (P, t) is

$$\mathrm{s}((P,t),j) = \mathrm{eft}(j, \max(t_0 - p_j^{\mathrm{pre}}, t_{q_j})). \tag{3}$$

The transition function to obtain the successor state (P', t') of state (P, t) when scheduling job j, $j \in P$, next is

$$\tau((P,t),j) = \begin{cases} (P \setminus \{j\}, t'), & \text{if } \mathrm{s}((P,t),j) < T^{\max}, \\ \hat{0}, & \text{else,} \end{cases} \tag{4}$$

with

$$t'_0 = \mathrm{s}((P,t),j) + p_j^{\mathrm{pre}} + p_j^0, \tag{5}$$

$$t'_r = \mathrm{s}((P,t),j) + p_j, \qquad \text{for } r = q_j, \tag{6}$$

$$t'_r = t_r, \qquad \text{for } r \in R \setminus \{q_j\}, \tag{7}$$

where $\hat{0}$ represents the infeasible state. The prize associated with a state transition is the prize z_j of the scheduled job j. Thus, each path in this state graph originating in the initial state $\mathbf{r}$ and ending in some other state than $\hat{0}$ corresponds to a feasible solution in which the jobs associated with the arcs are greedily scheduled in the order in which they appear in the path. Note that a feasible state $(P, t) \in V$ may, in general, be reached via multiple different paths, i.e., by including different sets of jobs S and/or by different orderings of these jobs. Therefore, a feasible state does, in general, not represent a unique solution. As we want to find a solution with maximum total prize, we are primarily interested in a path from $\mathbf{r}$ to (P, t) with maximum total prize. Let $Z^{\mathrm{lp}}(P, t)$ be this maximum total prize to reach a feasible state (P, t). In order to solve the PC-JSOCMSR we are looking for a feasible state with the maximum $Z^{\mathrm{lp}}(P, t)$. To find such a state we perform the A* search detailed in the following subsection. Figure 2 shows as example the state graph for the problem instance in Figure 1.

4.2 A* Algorithm Framework

In order to solve the PC-JSOCMSR we have to find a feasible state (P, t) with maximum $Z^{\mathrm{lp}}(P, t)$. Such a state cannot have any feasible successor, hence, either $P = \emptyset$ or $\tau((P, t), j) = \hat{0}$, $j \in P$, and otherwise $Z^{\mathrm{lp}}(P, t)$ is not the maximum achievable prize.

A* search belongs to the class of *informed* search strategies that make use of a heuristic estimate for guidance in order to return a proven optimal solution possibly faster than a more naive uninformed

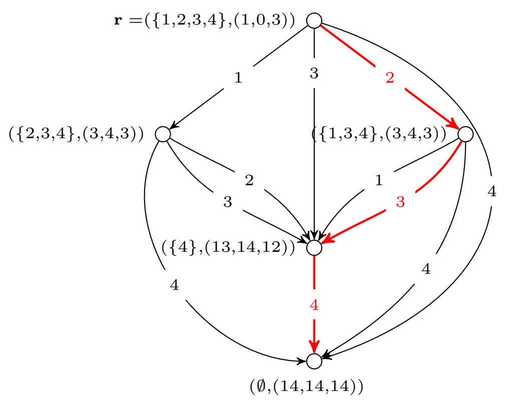

Figure 2: State graph for the problem instance from Figure 1 with $n = 4$ jobs and $m = 2$ secondary resources. Node labels denote the corresponding states (P, t), arc labels the scheduled jobs j. The path that represents the optimal solution is highlighted. Note that for the shown state graph the strengthening of states as described in Section 4.3 has been applied.

search like breadth- or depth-first-search. Within our A* search, each encountered feasible state (P, t) of the state graph is evaluated by a priority function $f(P, t) = Z^{\mathrm{lp}}(P, t) + Z^{\mathrm{ub}}(P, t)$ in which $Z^{\mathrm{lp}}(P, t)$ corresponds to the prize of the so far best (i.e., longest) known path from $\mathbf{r}$ to state (P, t) and $Z^{\mathrm{ub}}(P, t)$ is the A* search's heuristic function estimating the "costs to go". The latter is in our case an upper bound on the still achievable prize by extending the path from state (P, t) onward. The value of the priority function $f(P, t)$ is thus an upper bound on the total prize that a solution may achieve by considering state (P, t). Different options for how to compute this upper bound will be investigated in Section 4.4.

Our A* search is shown in pseudo-code in Algorithm 1. It maintains the set W of all so far encountered states, implemented by a hash table; for each contained state (P, t), this data structure also stores the values $Z^{\mathrm{lp}}(P, t)$ and $Z^{\mathrm{ub}}(P, t)$ as well as a reference $\mathrm{pred}(P, t)$ to the predecessor state of a currently longest path from $\mathbf{r}$ to (P, t), and the last scheduled job $\mathrm{j}(P, t)$. Furthermore, the algorithm maintains the *open list* Q, which contains all states queued for further expansion. It is realized by a priority queue data structure that considers the states' priority values $f(P, t)$. Last but not least, our A* search maintains a reference x_{maxlp} to the encountered state (P, t) with the so far largest $Z^{\mathrm{lp}}(P, t)$ value. Both W and Q, as well as x_{maxlp}, are initialized with the initial state $\mathbf{r}$.

At each major iteration, a state (P, t) with maximum priority value $f(P, t)$ is taken from Q. This state (P, t) is then *expanded*, which means that each job in P is considered as next job to be scheduled by calculating the respective successor state obtained by the transition $(P', t') = \tau((P, t), j)$. If a job yields the infeasible state $\hat{0}$, it is skipped. Similarly, if the obtained state has been encountered before and the new path via state (P, t) is not longer than the previously identified path to (P', t'), we skip job j. Otherwise, if a new feasible state is reached, it is added to set W. Since a new longest path to state (P', t') via (P, t) has been found, line 20 sets $Z^{\mathrm{lp}}(P', t') = Z^{\mathrm{lp}}(P, t) + z_j$ and stores (P, t) as predecessor of (P', t') and job j as the last scheduled job. The upper bound $Z^{\mathrm{ub}}(P', t')$ is then also calculated, and if there is potential for further improvement, i.e., $Z^{\mathrm{ub}}(P', t') > 0$, state (P', t') is added to the open list Q for a possible future expansion. Finally, reference x_{maxlp} is updated if a new overall longest path is obtained.

A special aspect of our A* search therefore is that we do not have a specific target state that is known in advance, and we only add states that may yield further successor states to the open list. Lines 6 to 10 makes sure that we nevertheless recognize when a proven optimal solution has been reached: This is the case when either the open list Q becomes empty or the priority value $f(P, t)$ of Q's top element (P, t) (i.e., the maximum priority value) is not larger than the length $Z^{\mathrm{lp}}(x_{\mathrm{maxlp}})$ of the so far longest identified path. Note that the priority value of Q's top element always is a valid overall upper bound for the total achievable prize. This follows from the fact that $Z^{\mathrm{ub}}(P, t)$ is an

Algorithm 1: A* Algorithm for PC-JSOCMSR

```
1  Input: Initial state $\mathbf{r}$;
2  set of encountered states $W \leftarrow \{\mathbf{r}\},\ Z^{\mathrm{lp}}(\mathbf{r}) \leftarrow 0$;
3  open list $Q \leftarrow \{(\mathbf{r}, f(\mathbf{r}) = Z^{\mathrm{ub}}(\mathbf{r}))\}$;
4  state with maximum $Z^{\mathrm{lp}}$ so far $x_{\mathrm{maxlp}} \leftarrow \mathbf{r}$;
5  do
6      if $Q = \emptyset$ then
7          return opt. solution given by $x_{\mathrm{maxlp}}$ and its predecessor chain
8      $(P,t) \leftarrow$ pop state with maximum $f(P,t)$ from $Q$;
9      if $f(P,t) \leq Z^{\mathrm{lp}}(x_{\mathrm{maxlp}})$ then
10         return opt. solution given by $x_{\mathrm{maxlp}}$ and its predecessor chain
11     // expand state $(P,t)$:
12     foreach $j \in P$ do
13         $(P',t') \leftarrow \tau((P,t),j)$; strengthen state $(P',t')$;
14         if $(P',t') = \hat{0} \lor (P',t') \in W \land Z^{\mathrm{lp}}(P,t) + z_j \leq Z^{\mathrm{lp}}(P',t')$ then
15             // infeasible or existing state reached in no better way, skip
16             continue
17         if $(P',t') \notin W$ then
18             // new state reached
19             $W \leftarrow W \cup \{(P',t')\}$;
20         $Z^{\mathrm{lp}}(P',t') \leftarrow Z^{\mathrm{lp}}(P,t) + z_j,\ \mathrm{pred}(P',t') \leftarrow (P,t),\ \mathrm{j}(P',t') = j$ ;
21         if $Z^{\mathrm{ub}}(P',t') \neq 0$ then
22             $Q \leftarrow Q \cup ((P',t'), f(P',t') = Z^{\mathrm{lp}}(P',t') + Z^{\mathrm{ub}}(P',t'))$
23         if $Z^{\mathrm{lp}}(x_{\mathrm{maxlp}}) < Z^{\mathrm{lp}}(P',t')$ then
24             $x_{\mathrm{maxlp}} \leftarrow (P',t')$;
25 while time or memory limit not reached;
26 // terminate early:
27 $(P,t) \leftarrow$ state with maximum $f(P,t)$ from $Q$;
28 derive solution $\pi$ from $x_{\mathrm{maxlp}}$ following predecessors;
29 $\pi \leftarrow$ greedily augment $\pi$ with jobs from $P$;
30 return heuristic solution $\pi$ and upper bound $f(P,t)$;
```

admissible heuristic according to (Hart et al. 1968), i.e., it never underestimates the real prize that can still be achieved from (P,t) onward. Further, an optimal solution is derived from state x_{maxlp} by following its chain of predecessor states and respectively scheduled jobs, and the corresponding solution is returned.

A particular feature of our A* search is that it can also be terminated early by providing a time or memory limit for the execution and it still yields a heuristic solution together with an upper bound on the optimal solution value in this case. This heuristic solution is derived from the so far best state x_{maxlp} by following its chain of predecessors and additionally considering all remaining jobs in P in their natural order (i.e., as given in the instance specification) for further addition in a greedy way. The returned upper bound is the priority value of Q's top element.

4.3 Strengthening of States

Frequently, we can safely replace a state (P,t) by a *strengthened* state (P',t'), with $P' \subseteq P$ and $t'_r \geq t_r$, $r \in R_0$, where either $P' \subset P$ or $t'_r > t_r$ for one or more r, $r \in R_0$, without losing possible solutions. This state strengthening is applied in Algorithm 1 at line 13 to any state that is obtained from the transition function τ.

By considering the earliest start time $\mathrm{s}((P,t),j)$ for job j, $j \in P$, we can first remove all jobs from P that actually cannot be scheduled anymore, i.e., $P' = \{j \in P \mid \mathrm{s}((P,t),j) \neq T^{\max}\}$. Then, time t'_r, $r \in R_0$, is set to the earliest possible time when resource r can be used considering all remaining jobs P', i.e.,

$$t'_0 = \min_{j \in P'} \left(\mathrm{s}((P,t),j) + p^{\mathrm{pre}}_j\right), \tag{8}$$

$$t'_r = \begin{cases} \min_{j \in J_r \cap P'} \mathrm{s}((P,t),j), & \text{if } J_r \cap P' \neq \emptyset, \\ T^{\max}, & \text{else,} \end{cases} \qquad r \in R. \tag{9}$$

Here, t'_r is set to $T^{\max}$ if no job that requires resource r remains in P'.

4.4 Upper Bounds for the Total Prize of Remaining Jobs

For a given state (P,t), an upper bound for the still achievable total prize for the remaining jobs in P can be calculated by solving a linear programming (LP) relaxation of a multi-constrained 0–1 knapsack problem

$$Z^{\mathrm{ub}}_{\text{MKP-LP}}(P,t) = \max \quad \sum_{j \in P} z_j x_j \tag{10}$$

$$\text{s.t} \quad \sum_{j \in P} p^0_j x_j \leq W_0(P,t), \tag{11}$$

$$\sum_{j \in P \cap J_r} p_j x_j \leq W_r(P,t), \qquad r \in R, \tag{12}$$

$$x_j \in [0,1], \qquad j \in P, \tag{13}$$

where x_j is a continuous relaxation of a binary variable that indicates if job j is included (=1) or not (=0), $j \in P$. The right-hand-sides of the knapsack constraints are

$$W_0(P,t) = \left| \bigcup_{\substack{j \in P, \\ w=1,\ldots,\omega_j \mid \\ W^{\mathrm{end}}_{jw} - p^{\mathrm{post}}_j \geq t_0 + p^0_j}} \left[\max\left(t_0, W^{\mathrm{start}}_{jw} + p^{\mathrm{pre}}_j\right), W^{\mathrm{end}}_{jw} - p^{\mathrm{post}}_j\right] \right| \tag{14}$$

and

$$W_r(P,t) = \left| \bigcup_{\substack{j \in P \cap J_r, \\ w=1,\ldots,\omega_j \mid \\ W^{\mathrm{end}}_{jw} \geq t_r + p_j}} \left[\max\left(t_r, W^{\mathrm{start}}_{jw}\right), W^{\mathrm{end}}_{jw}\right] \right|, \tag{15}$$

where the union of intervals is defined as $\bigcup_{i=1,\ldots,k}[\alpha_i, \beta_i] = \{\gamma \in \mathbb{R} \mid \exists i : \gamma \in [\alpha_i, \beta_i]\}$, and function $|\cdot|$ denotes the sum of the lengths of the resulting disjoint continuous intervals of this union. Thus, $W_0(P,t)$ and $W_r(P,t)$ represent the total amount of still available time for resource 0 and resource r, $r \in R$, respectively, considering the current state and the time windows.

To solve this upper bound calculation problem for each state with an LP solver is computationally rather expensive, as our experiments in the next section will document. As alternatives, we therefore consider the computation of upper bounds by solving two types of further relaxations. The first one

$$\begin{aligned} Z_0^{\text{ub}}(P,t) = \max \quad & \sum_{j\in P} z_j x_j && && (16)\\ \text{s.t} \quad & \sum_{j\in P} p_j^0 x_j \le W_0(P,t), && && (17)\\ & x_j \in [0,1], && j \in P, && (18) \end{aligned}$$

is obtained by removing inequalities (12); the second one

$$\begin{aligned} h^{\text{ub}}(P,t,u) = \max \quad & \sum_{j\in P} z_j x_j + u\left(W_0(P,t) - \sum_{j\in P} p_j^0 x_j\right) && && (19)\\ \text{s.t} \quad & \sum_{j\in P\cap J_r} p_j x_j \le W_r(P,t), && r \in R, && (20)\\ & x_j \in [0,1], && j \in P, && (21) \end{aligned}$$

is obtained by performing a Lagrangian relaxation of inequality (11), where $u \geq 0$ is the Lagrangian dual multiplier associated with inequality (11).

Both $Z_0^{\text{ub}}(P,t)$ and $h^{\text{ub}}(P,t,u)$ are computed by solving LP relaxations of knapsack problems. In the latter case, this is possible since the problem separates over the resources and for each resource, the resulting problem is an LP relaxation of a knapsack problem. An LP relaxation of a knapsack problem can be efficiently solved by a greedy algorithm that packs items in decreasing prize/time-ratio order; the first item that does not completely fit is packed partially so that the capacity is exploited as far as possible, see Kellerer et al. (2004).

It follows from weak duality (see e.g. Nemhauser and Wolsey (1988), Prop. 6.1) that $h^{\text{ub}}(P,t,u)$ yields an upper bound on $Z_{\text{MKP-LP}}^{\text{ub}}(P,t)$ for all $u \geq 0$, but the quality of this upper bound depends on the choice of u. We have chosen to consider $h^{\text{ub}}(P,t,u)$ for the values $u = 0$ and $u = z_{\bar{j}}/p_{\bar{j}}^0$, where $\bar{j}$ is the last, and typically partially, packed item in an optimal solution to the problem solved to obtain $Z_0^{\text{ub}}(P,t)$. The value $u = z_{\bar{j}}/p_{\bar{j}}^0$ is chosen since it is an optimal LP dual solution associated with inequality (17) and therefore has a chance to be a good estimate of a value for u that gives a strong upper bound.

By solving the relaxations introduced above, the strongest bound on $Z_{\text{MKP-LP}}^{\text{ub}}(P,t)$ we can obtain is

$$Z_*^{\text{ub}}(P,t) = \min\left(Z_0^{\text{ub}}(P,t), h^{\text{ub}}(P,t,0), h^{\text{ub}}(P,t,z_{\bar{j}}/p_{\bar{j}}^0)\right). \tag{22}$$

In our experimental comparisons in Section 7, we will illustrate the practical strengths of the bounds

$$\begin{aligned} & Z_{\text{MKP-LP}}^{\text{ub}}(P,t), && (23)\\ & Z_0^{\text{ub}}(P,t), && (24)\\ & Z_{00}^{\text{ub}}(P,t) = \min\left(Z_0^{\text{ub}}(P,t), h^{\text{ub}}(P,t,0)\right), \text{ and} && (25)\\ & Z_{0\bar{j}}^{\text{ub}}(P,t) = \min\left(Z_0^{\text{ub}}(P,t), h^{\text{ub}}(P,t,z_{\bar{j}}/p_{\bar{j}}^0)\right), && (26) \end{aligned}$$

and study which compromise of strength and computational effort pays off the most in the context of our A* search.

5 A Mixed Integer Programming Model

We use the binary variable t_j to indicate if job j, $j \in J$, is included in the schedule (=1) or not (=0) and the binary variable t_{jw} to indicate if job j is assigned to time window w (=1) or not (=0),

$w = 1, \dots, \omega_j$, $j \in J$. Let the continuous variable s_j be the start time of job j. Binary variable $y_{jj'}$ is further used to indicate if job j is scheduled before j' w.r.t. the common resource (=1) or not (=0), if both jobs are scheduled, $j, j' \in J,\ j \neq j'$.

Let

$$\delta_{jj'} = \begin{cases} p_j, & \text{if } q_j = q_{j'}, \\ p_j^{\text{pre}} + p_j^0 - p_{j'}^{\text{pre}}, & \text{if } q_j \neq q_{j'}, \end{cases} \tag{27}$$

be the minimum time between the start of job j and the start of job j' if job j is scheduled before job j', which depends on whether both jobs use the same resource or not.

A solution to PC-JSOCMSR can be obtained by solving the MIP model

$$\max \quad \sum_{j \in J} z_j t_j \tag{28}$$

$$\text{s.t} \quad t_j = \sum_{w=1,\dots,\omega_j} t_{jw}, \qquad j \in J, \tag{29}$$

$$y_{jj'} + y_{j'j} \geq t_j + t_{j'} - 1, \qquad j, j' \in J,\ j \neq j', \tag{30}$$

$$s_{j'} \geq s_j + \delta_{jj'} - (T_j^{\text{dead}} - p_j - T_{j'}^{\text{rel}} + \delta_{jj'})(1 - y_{jj'}), \qquad j, j' \in J,\ j \neq j' \tag{31}$$

$$s_j \geq T_j^{\text{rel}} + \sum_{w=1,\dots,\omega_j} \left(W_{jw}^{\text{start}} - T_j^{\text{rel}} \right) t_{jw}, \qquad j \in J, \tag{32}$$

$$s_j \leq T_j^{\text{dead}} - p_j + \sum_{w=1,\dots,\omega_j} (W_{jw}^{\text{end}} - T_j^{\text{dead}}) t_{jw}, \qquad j \in J, \tag{33}$$

$$t_j \in \{0, 1\}, \qquad j \in J, \tag{34}$$

$$t_{jw} \in \{0, 1\}, \qquad w = 1, \dots, \omega_j,\ j \in J, \tag{35}$$

$$s_j \in [T_j^{\text{rel}}, T_j^{\text{dead}} - p_j], \qquad j \in J, \tag{36}$$

$$y_{jj'} \in \{0, 1\}, \qquad j, j' \in J,\ j \neq j'. \tag{37}$$

Equations (29) state that each scheduled job must be assigned to a time window and inequalities (30) ensure that if two jobs j and j' are scheduled, either $y_{jj'}$ or $y_{j'j}$ must be set to one, i.e., one of them needs to precede the other. If a job is to precede another one, inequalities (31) ensure this w.r.t. the jobs' start times. If a job is assigned to a time window, inequalities (32) and (33) make its start time comply with this time window, and otherwise the job only complies with its release time and deadline.

In the previous work (Horn et al. 2017), a MIP model with position based variables was introduced for JSOCMSR since this model showed better computational performance than a MIP model with order based variables. Such position based model does, however, not extend well to the current setting with multiple time windows since the time windows require explicit knowledge of the start time of each job, and the position based model only has explicit times for the start time of a certain position.

6 A Constraint Programming Model

We further propose the following Constraint Programming (CP) model for the PC-JSOCMSR, which we implemented in the constraint modeling language MiniZinc[1]. The model makes use of so-called *option type* variables. Such a variable may either have a value of a certain domain assigned or set to the special value $\top$ that indicates the absence of a value. For job $j \in J$ we use the option type variable s_j for the job's start time. An absent start time, i.e., $s_j = \top$, indicates that the job is not scheduled.

[1]https://www.minizinc.org

The CP model is given by

$$\max \sum_{j \in J \mid \mathrm{occurs}(s_j)} z_j \tag{38}$$

$$\mathrm{disjunctive\_strict}(\{(s_j + p_j^{\mathrm{pre}}, p_j^0) \mid j \in J\}), \tag{39}$$

$$\mathrm{disjunctive\_strict}(\{(s_j, p_j) \mid j \in J \wedge q_j = r\}), \qquad r \in R, \tag{40}$$

$$\mathrm{occurs}(s_j) \rightarrow \min_{\omega=1,\dots,\omega_j} W_{j\omega}^{\mathrm{start}} \leq s_j \leq \max_{\omega=1,\dots,\omega_j} (W_{j\omega}^{\mathrm{end}} - p_j), \qquad j \in J, \tag{41}$$

$$s_j \in [T_j^{\mathrm{rel}}, \dots, T_j^{\mathrm{dead}} - p_j] \cup \{\top\}, \qquad j \in J, \tag{42}$$

where for job $j \in J$ the predicate occurs(s_j) yields true if the option type variable s_j is not absent, i.e., job j is scheduled. The strict disjunctive constraints (39) and (40) ensure that all scheduled jobs do not overlap w.r.t. their usage of the common resource and the secondary resource $r \in R$, respectively. Absent jobs are hereby ignored. Constraints (41) state that if job $j \in J$ is scheduled, it must be performed within one of the job's time windows.

7 Experimental Results

The proposed A* algorithm from Section 4 was implemented in C++ using GNU G++ 5.4.1 for compilation. The MIP model from Section 5 was solved with Gurobi[2] Optimizer Version 7.5.1. All tests were performed on a cluster of machines with Intel Xeon E5-2640 v4 processors with 2.40 GHz in single-threaded mode with a CPU time limit of 900 seconds and a memory limit of 15GB per run.

We created three non-trivial benchmark instance sets in order to test our solution approaches. The first two instance sets, called B and S, are, with respect to their basic characteristics, inspired from the particle therapy patient scheduling application, while the third set, called A, exhibits characteristics from the avionic system application. All these instances are available online[3]. Each set consists of 30 instances for each combination of $n \in \{10, 20, 30, 40, 50, 60, 70, 80, 90\}$ jobs and $m \in \{2, 3\}$ secondary resources for the particle therapy based scenario and $m \in \{3, 4\}$ secondary resources for the avionic system based scenario.

The difference between sets B and S lies in the distributions of the secondary resources and each job's pre-processing, post-processing and p^0 times. Set B consists of instances in which all secondary resources are used by the jobs equally likely and the distribution of processing times is *balanced*. This was achieved by sampling for job j, $j \in J$: (1) the secondary resource q_j from the discrete uniform distribution $\mathcal{U}\{1, m\}$, (2) the pre-processing times p_j^{pre} and the post-processing times p_j^{post} from $\mathcal{U}\{0, 8\}$ and (3) the times p_j^0 from the random variable $\mathrm{p}_{\mathrm{B}}^0 \sim \mathcal{U}\{1, 8\}$. In contrast, instances of set S exhibit a *skewed* workload such that the secondary resource m is chosen with a probability of 0.5 and all remaining secondary resources with a probability of $1/(2m-2)$. Furthermore the time to claim the common resource 0 is more dominant by sampling p_j^{pre} and p_j^{post} from $\mathcal{U}\{0, 5\}$ and p_j^0 from the random variable $\mathrm{p}_{\mathrm{S}}^0 \sim \mathcal{U}\{1, 13\}$. For both instance sets, the prize z_j is determined in a way to correlate to the usage of the common resource of job j by sampling it from $\mathcal{U}\{p_j^0, 2p_j^0\}$, $j \in J$. The time windows are chosen such that, on average about, 30% of the jobs can be scheduled. For this purpose let $T_i = \lfloor 0.3\, n\, \mathrm{E}(\mathrm{p}_i^0) \rfloor$ be the expected maximum resource usage regarding instance type i, $i \in \{\mathrm{B}, \mathrm{S}\}$. First, the number of time windows ω_j of job j is sampled from $\mathcal{U}\{1, 3\}$, i.e., a job can have up to three time windows. Second, for time window w, $w = 1, \dots, \omega_j$, we sample its start time W_{jw}^{start} from $\mathcal{U}\{0, T_i - p_j\}$ and its end time W_{jw}^{end} from $W_{jw}^{\mathrm{start}} + \max(p_j, \mathcal{U}\{\lfloor 0.1\, T_i/\omega_j \rfloor, \lfloor 0.4\, T_i/\omega_j \rfloor\})$ for instance type i, $i \in \{\mathrm{B}, \mathrm{S}\}$. Overlapping time windows are merged, and all time windows of a job are sorted according to increasing start times.

For the third instance set A, based on the avionic system scenario, a fixed time horizon $T = 1000$ is considered and the number of jobs of each type is distributed such that 20% are communication jobs,

[2]http://www.gurobi.com

[3]https://www.ac.tuwien.ac.at/research/problem-instances

40% are partition jobs and 40% are regular jobs. For communication jobs the time p_j^0 is set to $p_j^0 = 40$ and for partition jobs and regular jobs the time is sampled from $\mathcal{U}\{36, 44\}$. For partition jobs, the total processing time p_j is sampled from $\mathcal{U}\{5p_j^0, 8p_j^0\}$ and then, with equal probability, p_j^{pre} or p_j^{post} is set to 0 and the respective other value is set to $p_j - p_j^0$. Each partition job is assigned to a secondary resource and each secondary resource has the same probability to be selected.

Since the communication jobs and regular jobs do not use a secondary resource in the real scenario, an artificial secondary resource is introduced and assigned to all of these jobs. This means that the number of secondary resources for an instance is always one more than the number of application modules in the system and that for both the communication jobs and regular jobs $p_j = p_j^0$.

For partition and regular jobs, the number of time windows and the length of the time windows are computed as in the particle therapy case, but for the communication jobs the structure is rather different. The communication jobs can only be scheduled at certain points in time when the communication can be performed, and here these time points occur at $0, 80, 160, \ldots, 880$. Each time window of a communication job corresponds to one such time point and a job's total set of time windows corresponds to a number of consecutive such time points. The number of time windows for a communication job is obtained by sampling a value from the uniform distribution $\mathcal{U}\{1, 3\}$ and multiply it by three.

The prize z_j is for five of the partition jobs and ten of the communication jobs set to the high value 70 to give these jobs a higher priority, while for remaining partition jobs and communication jobs, the prize is sampled from $\mathcal{U}\{10, 50\}$. For regular jobs, the prize is sampled from $\mathcal{U}\{10, 25\}$.

Note that since we only use integral time windows and processing times, all start times can also safely be assumed to be integral. In our implementation, we therefore round down any fractional upper bound to the closest integer value. Note that due to this rounding it is even possible that $Z_*^{\mathrm{ub}}(P, t)$ yields occasionally tighter bounds than $Z_{\mathrm{MKP-LP}}^{\mathrm{ub}}(P, t)$.

7.1 Comparison of Upper Bound Functions

We start by experimentally evaluating the impact of the individual components of our combined upper bound function $Z_*^{\mathrm{ub}}(P, t) = \min(Z_0^{\mathrm{ub}}(P, t), h^{\mathrm{ub}}(P, t, 0), h^{\mathrm{ub}}(P, t, z_{\bar{j}}/p_{\bar{j}}^0))$ from Section 4.1. To this end, we performed the A* search on all benchmark instances using $Z_*^{\mathrm{ub}}(P, t)$ to evaluate all states and count for the sub-functions $Z_0^{\mathrm{ub}}(P, t)$, $h^{\mathrm{ub}}(P, t, 0)$, and $h^{\mathrm{ub}}(P, t, z_{\bar{j}}/p_{\bar{j}}^0)$ how often each one of them yields the minimum, i.e., determines $Z_*^{\mathrm{ub}}(P, t)$. Figure 3 shows the obtained average success rates grouped according to the instance type, the number of jobs n, and the number of secondary resources m for all three upper bounds.

Most importantly we can see that in most cases not a single sub-function is dominating, i.e., it makes sense to calculate all three functions and to combine their results by taking the minimum in order to get a generally tighter bound. More specifically, the success of each sub-function obviously also depends on the specific characteristics of the problem instances. For instances of type B with two secondary resources, $h^{\mathrm{ub}}(P, t, 0)$ is for each instance class on average more than 50% of the times the strongest upper bound. In all other cases $h^{\mathrm{ub}}(P, t, z_{\bar{j}}/p_{\bar{j}}^0)$ is on average most successful, and for the instances of type S it is clear that both the bounds $h^{\mathrm{ub}}(P, t, z_{\bar{j}}/p_{\bar{j}}^0)$ and $Z_0^{\mathrm{ub}}(P, t)$ are of importance.

The strongest upper bound function, however, does not necessarily yield the best performing A* search, since the time for calculating the bound also plays a major role. As already stated in Section 4.1, we consider the upper bound functions $Z_{\mathrm{MKP-LP}}^{\mathrm{ub}}(P, t)$, $Z_0^{\mathrm{ub}}(P, t)$, $Z_{00}^{\mathrm{ub}}(P, t)$, $Z_{0\bar{j}}^{\mathrm{ub}}(P, t)$, and $Z_*^{\mathrm{ub}}(P, t)$. The former is solved by the CPLEX 12.7 LP solver in single threaded mode whereas the other upper bound functions make use of the sub-functions $h^{\mathrm{ub}}(P, t, u)$, $u \geq 0$ and $Z_0^{\mathrm{ub}}(P, t)$ in different ways as stated in Eqs. (22)–(26).

Table 1 presents the aggregated results for each combination of instance type, numbers of jobs, and secondary resources for our A* search using these different upper bound calculations. Columns %-opt show the percentage of instances which could be solved to proven optimality. Columns $\overline{Z^{\mathrm{ub}}}$ state the average final upper bounds and columns $\overline{\text{%-gap}}$ list the average optimality gaps which are calculated

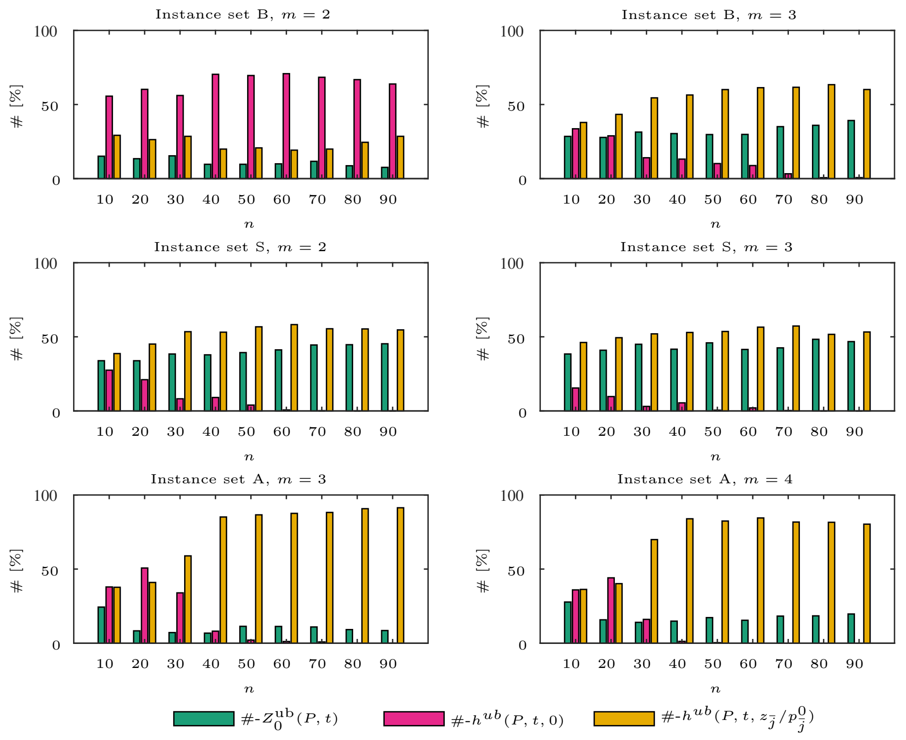

Figure 3: Success rates of upper bound subfunctions $Z_0^{\text{ub}}(P,t)$, $h^{\text{ub}}(P,t,0)$ and $h^{\text{ub}}(P,t,z_{\bar{j}}/p^0_{\bar{j}})$ to yield the smallest value, i.e., to determine $Z_*^{\text{ub}}(P,t)$.

by $100\% \cdot (Z^{\text{ub}} - Z(\pi))/Z^{\text{ub}}$, where π is the final solution and Z^{ub} the final upper bound. Columns t[s] list the median computation times in seconds, whereas columns $\overline{|W|}$ state the average number of encountered states during the A* search. Best values are printed bold.

In almost all cases, A* search with the combined bound $Z_*^{\text{ub}}(P,t)$ provides the tightest final bounds. There are only five exceptions where $Z_{00}^{\text{ub}}(P,t)$ or $Z_{0\bar{j}}^{\text{ub}}(P,t)$ yield tighter bounds on average, but $Z_*^{\text{ub}}(P,t)$ is not far behind. Using the original LP relaxation $Z_{\text{MKP-LP}}^{\text{ub}}(P,t)$ yields in almost all cases where not all instances could be solved to optimality worse final upper bounds than using $Z_0^{\text{ub}}(P,t)$, $Z_{00}^{\text{ub}}(P,t)$, $Z_{0\bar{j}}^{\text{ub}}(P,t)$ or $Z_*^{\text{ub}}(P,t)$. The reason for this is that, although the full LP relaxation $Z_{\text{MKP-LP}}^{\text{ub}}(P,t)$ may provide the tightest upper bound for a single state, substantially less nodes could be processed due to the higher computational effort to solve each LP, cf. columns $|W|$.

When considering instance sets B and S, in cases where not all instances could be solved to optimality, the A* search with $Z_0^{\text{ub}}(P,t)$ was able to provide the smallest average optimality gaps in most cases. For instances of set A the smallest average optimality gaps could be obtained from the A* search with $Z_{0\bar{j}}^{\text{ub}}(P,t)$ or $Z_*^{\text{ub}}(P,t)$ and for instances of set S with two secondary resources the smallest average gaps could be obtained from the A* algorithm with $Z_{00}^{\text{ub}}(P,t)$ or $Z_*^{\text{ub}}(P,t)$. This observation is in accordance with our previous observation concerning Fig. 3, where $h^{\text{ub}}(P,t,0)$ provides more often the strongest upper bound for instances of type S with two secondary resources and where $Z_{0\bar{j}}^{\text{ub}}(P,t)$ provides frequently more often the strongest upper bounds for instances of type A. We conclude that $Z_0^{\text{ub}}(P,t)$ might be a slightly better guidance for instances in sets B and S for our simple greedy heuristic used to find solutions when terminating early.

Considering only instance classes where all instances could be solved to optimality, the A* algorithm with upper bound function $Z_{\text{MKP-LP}}^{\text{ub}}(P,t)$ encounters less states than A* with $Z_*^{\text{ub}}(P,t)$ which in turn encounters less states than A* with one of the other functions $Z_0^{\text{ub}}(P,t)$, $Z_{00}^{\text{ub}}(P,t)$, or $Z_{0\bar{j}}^{\text{ub}}(P,t)$, respectively. This is not surprising since function $Z_{\text{MKP-LP}}^{\text{ub}}(P,t)$ solves the full LP relaxation which provides frequently the strongest upper bounds and $Z_*^{\text{ub}}(P,t)$ dominates the other functions $Z_0^{\text{ub}}(P,t)$, $Z_{00}^{\text{ub}}(P,t)$, and $Z_{0\bar{j}}^{\text{ub}}(P,t)$. However, again we see that providing the strongest upper bounds cannot

Table 1: Average results of A* search for different upper bound functions.

			$Z^{\mathrm{ub}}_{\mathrm{MKP-LP}}(P,t)$					$Z^{\mathrm{ub}}_{0}(P,t)$					$Z^{\mathrm{ub}}_{00}(P,t)$					$Z^{\mathrm{ub}}_{0\bar{j}}(P,t)$					$Z^{\mathrm{ub}}_{*}(P,t)$				
type	n	m	%-opt	$\overline{Z^{\mathrm{ub}}}$	$\overline{\%\text{-gap}}$	t[s]	$\overline{\lvert W\rvert}$	%-opt	$\overline{Z^{\mathrm{ub}}}$	$\overline{\%\text{-gap}}$	t[s]	$\overline{\lvert W\rvert}$	%-opt	$\overline{Z^{\mathrm{ub}}}$	$\overline{\%\text{-gap}}$	t[s]	$\overline{\lvert W\rvert}$	%-opt	$\overline{Z^{\mathrm{ub}}}$	$\overline{\%\text{-gap}}$	t[s]	$\overline{\lvert W\rvert}$	%-opt	$\overline{Z^{\mathrm{ub}}}$	$\overline{\%\text{-gap}}$	t[s]	$\overline{\lvert W\rvert}$
B	10	2	**100**	**30.93**	**0.00**	<0.1	$2.18\cdot10^{1}$	**100**	**30.93**	**0.00**	<0.1	$2.52\cdot10^{1}$	**100**	**30.93**	**0.00**	<0.1	$2.19\cdot10^{1}$	**100**	**30.93**	**0.00**	<0.1	$2.36\cdot10^{1}$	**100**	**30.93**	**0.00**	<0.1	$2.19\cdot10^{1}$
B	20	2	**100**	**50.37**	**0.00**	0.1	$2.06\cdot10^{2}$	**100**	**50.37**	**0.00**	<0.1	$2.66\cdot10^{2}$	**100**	**50.37**	**0.00**	<0.1	$2.07\cdot10^{2}$	**100**	**50.37**	**0.00**	<0.1	$2.55\cdot10^{2}$	**100**	**50.37**	**0.00**	<0.1	$2.06\cdot10^{2}$
B	30	2	**100**	**75.33**	**0.00**	0.6	$1.33\cdot10^{3}$	**100**	**75.33**	**0.00**	<0.1	$1.65\cdot10^{3}$	**100**	**75.33**	**0.00**	<0.1	$1.36\cdot10^{3}$	**100**	**75.33**	**0.00**	<0.1	$1.61\cdot10^{3}$	**100**	**75.33**	**0.00**	<0.1	$1.36\cdot10^{3}$
B	40	2	**100**	**98.93**	**0.00**	5.4	$1.17\cdot10^{4}$	**100**	**98.93**	**0.00**	0.3	$1.69\cdot10^{4}$	**100**	**98.93**	**0.00**	0.2	$1.22\cdot10^{4}$	**100**	**98.93**	**0.00**	0.3	$1.62\cdot10^{4}$	**100**	**98.93**	**0.00**	0.2	$1.20\cdot10^{4}$
B	50	2	**100**	**123.27**	**0.00**	55.2	$1.61\cdot10^{5}$	**100**	**123.27**	**0.00**	5.3	$2.21\cdot10^{5}$	**100**	**123.27**	**0.00**	3.1	$1.69\cdot10^{5}$	**100**	**123.27**	**0.00**	5.2	$2.13\cdot10^{5}$	**100**	**123.27**	**0.00**	3.9	$1.67\cdot10^{5}$
B	60	2	60	149.57	4.51	638.5	$8.69\cdot10^{5}$	97	146.97	0.11	89.5	$2.69\cdot10^{6}$	**100**	**146.80**	**0.00**	51.4	$1.76\cdot10^{6}$	97	146.93	0.22	84.5	$2.58\cdot10^{6}$	**100**	**146.80**	**0.00**	53.7	$1.75\cdot10^{6}$
B	70	2	3	184.90	14.40	900.0	$1.43\cdot10^{6}$	40	182.10	8.39	900.0	$1.18\cdot10^{7}$	**60**	**175.77**	**4.19**	623.1	$9.69\cdot10^{6}$	37	182.10	9.16	900.0	$1.11\cdot10^{7}$	56	175.93	4.41	712.0	$9.57\cdot10^{6}$
B	80	2	0	218.40	22.92	900.0	$1.21\cdot10^{6}$	0	219.00	16.07	900.0	$1.53\cdot10^{7}$	**3**	211.43	12.81	900.0	$1.40\cdot10^{7}$	0	218.10	16.02	900.0	$1.42\cdot10^{7}$	**3**	**211.23**	**12.71**	900.0	$1.38\cdot10^{7}$
B	90	2	**0**	256.07	29.48	900.0	$1.15\cdot10^{6}$	**0**	259.73	22.81	900.0	$1.43\cdot10^{7}$	**0**	251.63	**20.61**	900.0	$1.39\cdot10^{7}$	**0**	258.53	23.38	900.0	$1.36\cdot10^{7}$	**0**	**251.47**	20.69	900.0	$1.33\cdot10^{7}$
B	10	3	**100**	**36.17**	**0.00**	<0.1	$2.94\cdot10^{1}$	**100**	**36.17**	**0.00**	<0.1	$3.11\cdot10^{1}$	**100**	**36.17**	**0.00**	<0.1	$2.94\cdot10^{1}$	**100**	**36.17**	**0.00**	<0.1	$3.10\cdot10^{1}$	**100**	**36.17**	**0.00**	<0.1	$2.94\cdot10^{1}$
B	20	3	**100**	**59.27**	**0.00**	0.1	$2.76\cdot10^{2}$	**100**	**59.27**	**0.00**	<0.1	$3.03\cdot10^{2}$	**100**	**59.27**	**0.00**	<0.1	$2.86\cdot10^{2}$	**100**	**59.27**	**0.00**	<0.1	$2.92\cdot10^{2}$	**100**	**59.27**	**0.00**	<0.1	$2.80\cdot10^{2}$
B	30	3	**100**	**86.30**	**0.00**	1.2	$3.10\cdot10^{3}$	**100**	**86.30**	**0.00**	<0.1	$3.65\cdot10^{3}$	**100**	**86.30**	**0.00**	<0.1	$3.43\cdot10^{3}$	**100**	**86.30**	**0.00**	<0.1	$3.36\cdot10^{3}$	**100**	**86.30**	**0.00**	<0.1	$3.24\cdot10^{3}$
B	40	3	**100**	**112.00**	**0.00**	9.0	$3.37\cdot10^{4}$	**100**	**112.00**	**0.00**	0.4	$3.76\cdot10^{4}$	**100**	**112.00**	**0.00**	0.4	$3.61\cdot10^{4}$	**100**	**112.00**	**0.00**	0.4	$3.62\cdot10^{4}$	**100**	**112.00**	**0.00**	0.5	$3.50\cdot10^{4}$
B	50	3	93	140.90	0.45	143.7	$3.56\cdot10^{5}$	**100**	**140.33**	**0.00**	10.3	$5.09\cdot10^{5}$	**100**	**140.33**	**0.00**	11.6	$5.06\cdot10^{5}$	**100**	**140.33**	**0.00**	9.4	$4.96\cdot10^{5}$	**100**	**140.33**	**0.00**	10.0	$4.85\cdot10^{5}$
B	60	3	40	170.80	6.84	900.0	$1.13\cdot10^{6}$	**90**	166.07	**0.68**	132.3	$4.30\cdot10^{6}$	**90**	166.07	1.06	114.1	$4.19\cdot10^{6}$	**90**	166.03	1.01	126.7	$3.98\cdot10^{6}$	87	**166.00**	1.18	138.4	$3.95\cdot10^{6}$
B	70	3	7	208.73	15.67	900.0	$1.30\cdot10^{6}$	**40**	203.20	**7.10**	900.0	$1.24\cdot10^{7}$	37	202.90	8.84	900.0	$1.24\cdot10^{7}$	37	**202.60**	7.64	900.0	$1.25\cdot10^{7}$	**40**	**202.60**	8.41	871.8	$1.22\cdot10^{7}$
B	80	3	0	252.43	25.45	900.0	$1.22\cdot10^{6}$	**10**	247.87	**14.78**	900.0	$1.57\cdot10^{7}$	**10**	247.87	16.42	900.0	$1.54\cdot10^{7}$	**10**	247.43	16.19	900.0	$1.49\cdot10^{7}$	**10**	**247.40**	16.67	900.0	$1.44\cdot10^{7}$
B	90	3	**0**	285.43	28.03	900.0	$1.36\cdot10^{6}$	**0**	282.80	**19.00**	900.0	$1.65\cdot10^{7}$	**0**	282.73	19.60	869.4	$1.68\cdot10^{7}$	**0**	**282.50**	19.31	900.0	$1.62\cdot10^{7}$	**0**	282.60	20.12	900.0	$1.56\cdot10^{7}$
S	10	2	**100**	**50.93**	**0.00**	<0.1	$2.39\cdot10^{1}$	**100**	**50.93**	**0.00**	<0.1	$2.45\cdot10^{1}$	**100**	**50.93**	**0.00**	<0.1	$2.40\cdot10^{1}$	**100**	**50.93**	**0.00**	<0.1	$2.44\cdot10^{1}$	**100**	**50.93**	**0.00**	<0.1	$2.40\cdot10^{1}$
S	20	2	**100**	**89.93**	**0.00**	0.1	$3.25\cdot10^{2}$	**100**	**89.93**	**0.00**	<0.1	$3.43\cdot10^{2}$	**100**	**89.93**	**0.00**	<0.1	$3.30\cdot10^{2}$	**100**	**89.93**	**0.00**	<0.1	$3.37\cdot10^{2}$	**100**	**89.93**	**0.00**	<0.1	$3.28\cdot10^{2}$
S	30	2	**100**	**131.37**	**0.00**	1.9	$3.51\cdot10^{3}$	**100**	**131.37**	**0.00**	0.1	$4.01\cdot10^{3}$	**100**	**131.37**	**0.00**	0.1	$3.94\cdot10^{3}$	**100**	**131.37**	**0.00**	0.1	$3.80\cdot10^{3}$	**100**	**131.37**	**0.00**	0.1	$3.76\cdot10^{3}$
S	40	2	**100**	**180.07**	**0.00**	21.2	$7.85\cdot10^{4}$	**100**	**180.07**	**0.00**	1.0	$8.19\cdot10^{4}$	**100**	**180.07**	**0.00**	1.1	$8.01\cdot10^{4}$	**100**	**180.07**	**0.00**	1.2	$8.07\cdot10^{4}$	**100**	**180.07**	**0.00**	1.1	$7.95\cdot10^{4}$
S	50	2	87	226.57	0.97	220.3	$6.11\cdot10^{5}$	**100**	**225.67**	**0.00**	17.6	$1.03\cdot10^{6}$	**100**	**225.67**	**0.00**	16.4	$1.01\cdot10^{6}$	**100**	**225.67**	**0.00**	16.6	$1.01\cdot10^{6}$	**100**	**225.67**	**0.00**	17.4	$9.93\cdot10^{5}$
S	60	2	13	296.63	12.56	900.0	$1.32\cdot10^{6}$	**47**	288.00	**5.81**	777.2	$1.12\cdot10^{7}$	43	288.57	6.55	900.0	$1.04\cdot10^{7}$	43	288.30	5.98	900.0	$1.01\cdot10^{7}$	43	**287.90**	6.07	900.0	$1.03\cdot10^{7}$
S	70	2	**3**	353.50	17.58	900.0	$1.33\cdot10^{6}$	**3**	347.83	**11.62**	860.5	$1.73\cdot10^{7}$	**3**	347.77	11.99	883.7	$1.72\cdot10^{7}$	**3**	347.77	11.92	900.0	$1.67\cdot10^{7}$	**3**	**347.67**	11.93	900.0	$1.70\cdot10^{7}$
S	80	2	0	399.07	18.84	900.0	$1.25\cdot10^{6}$	**7**	395.33	**12.14**	827.3	$1.74\cdot10^{7}$	**7**	395.33	12.37	794.0	$1.72\cdot10^{7}$	**7**	**395.20**	12.27	865.5	$1.69\cdot10^{7}$	**7**	395.30	12.60	881.7	$1.70\cdot10^{7}$
S	90	2	**0**	463.83	25.31	900.0	$1.18\cdot10^{6}$	**0**	460.67	**17.52**	741.0	$1.76\cdot10^{7}$	**0**	460.67	17.76	735.4	$1.75\cdot10^{7}$	**0**	**460.53**	18.26	826.6	$1.72\cdot10^{7}$	**0**	460.60	18.42	856.2	$1.71\cdot10^{7}$
S	10	3	**100**	**51.97**	**0.00**	<0.1	$2.55\cdot10^{1}$	**100**	**51.97**	**0.00**	<0.1	$2.63\cdot10^{1}$	**100**	**51.97**	**0.00**	<0.1	$2.55\cdot10^{1}$	**100**	**51.97**	**0.00**	<0.1	$2.56\cdot10^{1}$	**100**	**51.97**	**0.00**	<0.1	$2.55\cdot10^{1}$
S	20	3	**100**	**96.47**	**0.00**	0.1	$2.97\cdot10^{2}$	**100**	**96.47**	**0.00**	<0.1	$3.05\cdot10^{2}$	**100**	**96.47**	**0.00**	<0.1	$3.03\cdot10^{2}$	**100**	**96.47**	**0.00**	<0.1	$3.02\cdot10^{2}$	**100**	**96.47**	**0.00**	<0.1	$3.01\cdot10^{2}$
S	30	3	**100**	**135.90**	**0.00**	1.3	$3.56\cdot10^{3}$	**100**	**135.90**	**0.00**	<0.1	$3.65\cdot10^{3}$	**100**	**135.90**	**0.00**	<0.1	$3.66\cdot10^{3}$	**100**	**135.90**	**0.00**	<0.1	$3.63\cdot10^{3}$	**100**	**135.90**	**0.00**	<0.1	$3.63\cdot10^{3}$
S	40	3	97	185.47	0.13	31.3	$1.31\cdot10^{5}$	**100**	**185.43**	**0.00**	1.6	$1.69\cdot10^{5}$	**100**	**185.43**	**0.00**	1.7	$1.62\cdot10^{5}$	**100**	**185.43**	**0.00**	2.0	$1.62\cdot10^{5}$	**100**	**185.43**	**0.00**	2.0	$1.57\cdot10^{5}$
S	50	3	70	236.83	2.29	434.9	$7.69\cdot10^{5}$	**97**	**234.53**	**0.15**	38.0	$2.42\cdot10^{6}$	**97**	**234.53**	**0.15**	37.5	$2.41\cdot10^{6}$	**97**	**234.53**	**0.15**	37.4	$2.38\cdot10^{6}$	**97**	**234.53**	**0.15**	35.2	$2.35\cdot10^{6}$
S	60	3	7	304.07	11.82	900.0	$1.44\cdot10^{6}$	40	296.37	5.84	899.4	$1.28\cdot10^{7}$	**47**	296.40	5.82	888.1	$1.26\cdot10^{7}$	43	296.27	**5.72**	900.0	$1.24\cdot10^{7}$	**47**	**296.17**	5.97	876.3	$1.24\cdot10^{7}$
S	70	3	0	356.13	18.99	900.0	$1.50\cdot10^{6}$	**10**	353.07	**12.74**	865.5	$1.72\cdot10^{7}$	**10**	353.07	13.11	831.3	$1.70\cdot10^{7}$	**10**	**352.67**	12.84	894.7	$1.66\cdot10^{7}$	**10**	352.70	12.89	900.0	$1.64\cdot10^{7}$
S	80	3	**0**	416.03	21.29	900.0	$1.40\cdot10^{6}$	**0**	**413.33**	**15.80**	800.5	$1.80\cdot10^{7}$	**0**	413.43	16.07	848.1	$1.76\cdot10^{7}$	**0**	413.37	15.99	900.0	$1.70\cdot10^{7}$	**0**	**413.33**	15.85	900.0	$1.73\cdot10^{7}$
S	90	3	**0**	465.87	27.71	900.0	$1.23\cdot10^{6}$	**0**	464.40	**21.14**	777.2	$1.73\cdot10^{7}$	**0**	464.40	**21.14**	829.4	$1.72\cdot10^{7}$	**0**	**464.37**	21.77	855.5	$1.70\cdot10^{7}$	**0**	**464.37**	21.77	887.4	$1.69\cdot10^{7}$
A	10	3	**100**	**422.13**	**0.00**	0.1	$2.08\cdot10^{2}$	**100**	**422.13**	**0.00**	<0.1	$2.42\cdot10^{2}$	**100**	**422.13**	**0.00**	<0.1	$2.08\cdot10^{2}$	**100**	**422.13**	**0.00**	<0.1	$2.08\cdot10^{2}$	**100**	**422.13**	**0.00**	<0.1	$2.08\cdot10^{2}$
A	20	3	**100**	**707.27**	**0.00**	5.5	$1.19\cdot10^{4}$	**100**	**707.27**	**0.00**	0.1	$1.96\cdot10^{4}$	**100**	**707.27**	**0.00**	0.1	$1.25\cdot10^{4}$	**100**	**707.27**	**0.00**	0.1	$1.28\cdot10^{4}$	**100**	**707.27**	**0.00**	0.1	$1.20\cdot10^{4}$
A	30	3	97	904.27	0.03	82.6	$3.05\cdot10^{5}$	**100**	**903.97**	**0.00**	4.1	$6.87\cdot10^{5}$	**100**	**903.97**	**0.00**	2.3	$4.40\cdot10^{5}$	**100**	**903.97**	**0.00**	2.7	$3.82\cdot10^{5}$	**100**	**903.97**	**0.00**	2.0	$3.53\cdot10^{5}$
A	40	3	43	1051.43	7.18	900.0	$1.10\cdot10^{6}$	83	1033.93	1.55	71.9	$8.10\cdot10^{6}$	90	1031.47	1.10	55.5	$7.49\cdot10^{6}$	**97**	**1030.00**	**0.36**	39.1	$5.48\cdot10^{6}$	**97**	**1030.00**	**0.36**	40.0	$5.42\cdot10^{6}$
A	50	3	0	1216.90	20.69	900.0	$1.45\cdot10^{6}$	27	1197.27	10.92	366.5	$2.04\cdot10^{7}$	27	1192.77	10.64	382.8	$2.03\cdot10^{7}$	**33**	1177.87	**9.89**	377.2	$1.92\cdot10^{7}$	**33**	**1177.67**	9.96	377.2	$1.92\cdot10^{7}$
A	60	3	0	1255.03	24.28	900.0	$1.44\cdot10^{6}$	**3**	1240.90	18.49	368.1	$1.96\cdot10^{7}$	**3**	1239.60	18.41	385.4	$1.96\cdot10^{7}$	**3**	1225.67	17.34	400.4	$1.96\cdot10^{7}$	**3**	**1225.57**	**17.28**	416.6	$1.96\cdot10^{7}$
A	70	3	**0**	1290.57	27.91	900.0	$1.33\cdot10^{6}$	**0**	1286.33	21.74	391.2	$1.89\cdot10^{7}$	**0**	1285.23	21.70	422.2	$1.89\cdot10^{7}$	**0**	1270.63	**20.41**	448.3	$1.89\cdot10^{7}$	**0**	**1270.53**	20.44	442.8	$1.89\cdot10^{7}$
A	80	3	**0**	1319.33	31.93	900.0	$1.31\cdot10^{6}$	**0**	1329.10	26.30	364.7	$1.80\cdot10^{7}$	**0**	1328.67	25.94	377.4	$1.80\cdot10^{7}$	**0**	**1311.53**	**24.78**	407.3	$1.80\cdot10^{7}$	**0**	**1311.53**	**24.78**	412.5	$1.80\cdot10^{7}$
A	90	3	**0**	1346.00	33.26	900.0	$1.13\cdot10^{6}$	**0**	1337.13	26.56	409.8	$1.71\cdot10^{7}$	**0**	1337.03	26.54	413.0	$1.71\cdot10^{7}$	**0**	**1323.67**	**26.13**	461.7	$1.71\cdot10^{7}$	**0**	**1323.67**	**26.13**	445.5	$1.71\cdot10^{7}$
A	10	4	**100**	**451.07**	**0.00**	0.1	$3.11\cdot10^{2}$	**100**	**451.07**	**0.00**	<0.1	$3.29\cdot10^{2}$	**100**	**451.07**	**0.00**	<0.1	$3.11\cdot10^{2}$	**100**	**451.07**	**0.00**	<0.1	$3.12\cdot10^{2}$	**100**	**451.07**	**0.00**	<0.1	$3.12\cdot10^{2}$
A	20	4	**100**	**751.73**	**0.00**	8.3	$1.66\cdot10^{4}$	**100**	**751.73**	**0.00**	0.2	$2.32\cdot10^{4}$	**100**	**751.73**	**0.00**	0.1	$1.71\cdot10^{4}$	**100**	**751.73**	**0.00**	0.1	$1.71\cdot10^{4}$	**100**	**751.73**	**0.00**	0.1	$1.67\cdot10^{4}$
A	30	4	93	952.90	0.36	113.6	$4.20\cdot10^{5}$	**100**	**951.97**	**0.00**	4.0	$7.45\cdot10^{5}$	**100**	**951.97**	**0.00**	3.6	$6.26\cdot10^{5}$	**100**	**951.97**	**0.00**	3.0	$5.00\cdot10^{5}$	**100**	**951.97**	**0.00**	2.9	$4.95\cdot10^{5}$
A	40	4	23	1094.60	10.00	900.0	$1.30\cdot10^{6}$	**93**	1062.80	0.50	81.3	$7.94\cdot10^{6}$	**93**	1062.80	0.50	75.8	$7.78\cdot10^{6}$	**93**	**1062.07**	**0.36**	65.7	$6.08\cdot10^{6}$	**93**	**1062.07**	**0.36**	68.1	$6.08\cdot10^{6}$
A	50	4	0	1240.90	19.66	900.0	$1.47\cdot10^{6}$	17	1208.53	11.40	374.6	$1.95\cdot10^{7}$	17	1208.03	11.37	372.5	$1.95\cdot10^{7}$	**20**	**1196.73**	**9.30**	374.0	$1.86\cdot10^{7}$	**20**	**1196.73**	**9.30**	397.3	$1.86\cdot10^{7}$
A	60	4	0	1294.73	24.38	900.0	$1.26\cdot10^{6}$	3	1259.47	15.72	350.6	$1.99\cdot10^{7}$	3	1259.43	15.75	378.2	$1.99\cdot10^{7}$	**10**	**1250.40**	**14.60**	393.1	$1.96\cdot10^{7}$	**10**	**1250.40**	**14.60**	402.6	$1.96\cdot10^{7}$
A	70	4	0	1328.40	29.09	900.0	$1.23\cdot10^{6}$	**3**	1300.93	19.70	354.1	$1.86\cdot10^{7}$	**3**	1300.93	19.64	362.0	$1.86\cdot10^{7}$	**3**	**1294.20**	**19.28**	393.5	$1.85\cdot10^{7}$	**3**	**1294.20**	**19.28**	393.5	$1.85\cdot10^{7}$
A	80	4	**0**	1356.93	33.87	900.0	$1.33\cdot10^{6}$	**0**	1338.07	26.25	363.2	$1.79\cdot10^{7}$	**0**	1338.07	26.25	388.1	$1.79\cdot10^{7}$	**0**	**1328.47**	**25.43**	388.6	$1.79\cdot10^{7}$	**0**	**1328.47**	**25.43**	405.8	$1.79\cdot10^{7}$
A	90	4	**0**	1374.47	31.83	900.0	$1.25\cdot10^{6}$	**0**	1356.17	24.69	361.8	$1.73\cdot10^{7}$	**0**	1356.17	24.69	379.0	$1.73\cdot10^{7}$	**0**	**1347.33**	**24.32**	415.1	$1.73\cdot10^{7}$	**0**	**1347.33**	**24.32**	425.0	$1.73\cdot10^{7}$

outweigh the disadvantage of the longer computation times such that A* with $Z^{\text{ub}}_{*}(P,t)$ terminates in almost all cases substantially earlier than A* with $Z^{\text{ub}}_{\text{MKP-LP}}(P,t)$.

Last but not least, we point out that the memory limit of 16GB was the termination reason in several runs for the largest instances. Thus, memory consumption plays a significant role in our A* algorithm. One way to save memory would be to adopt the technique applied in (Horn et al. 2017) where states with the same P are stored in an aggregated fashion. This can be done by storing P only once and include the individual vectors t and further information in an attached list of so-called non-dominated time records.

7.2 Comparison of A* Search, MIP, and CP

We finally compare our A* search using the generally dominating upper bound function $Z^{\text{ub}}_{*}(P,t)$ to solving the MIP model from Section 5 using Gurobi and the CP model from Section 6 using MiniZinc 2.1.7 with the backend solver Chuffed. Note that we considered besides Chuffed also the backend solvers Gecode and G12 LazyFD, but Chuffed clearly dominated these alternatives concerning the number of instances solved to proven optimality, as it is documented in more detail in our conference paper Horn et al. (2018b). Table 2 shows the aggregated results. Regarding the number of instances that could be solved to proven optimality, the A* search consistently outperforms Gurobi and Chuffed. For particle therapy based instances of type B and S, A* search could solve all instances with up to 50 jobs to proven optimality, except one skewed instance with three secondary resources. The avionic system based instances of type A are harder to solve. Here, A* search was only able to solve all instances with up to 30 jobs to proven optimality.

The largest instance which A* could solve to proven optimality consists of 80 jobs, whereas the largest instances that Gurobi and Chuffed could solve to proven optimality have 50 and 60 jobs, respectively. Gurobi could solve all avionic based instances with up to 20 jobs, all balanced instances with up to 40 jobs, and all skewed instances with up to 30 jobs to proven optimality. Computation times for those are, however, significantly larger than for A*. In particular for small instances with up to $n = 30$ jobs, A* only required median computation times of no more than 0.1 seconds for instances of type B and S. The CP solver Chuffed could solve all instances of type B and S up to $n = 40$ jobs to optimality and all instances of type A up to $n = 30$ jobs to optimality. The A* algorithm was able to provide equally good or better final solutions than Gurobi and Chuffed in almost all cases for instances of type B and S. Exceptions occurred only for some of the largest instances with 90 jobs, where Gurobi's heuristic performance proved to be superior.

For instances of type A, Gurobi's heuristic performance is also superior for instances of smaller sizes. Note, however, that the derivation of just heuristic solutions is not in the foreground of our research here. Concerning obtained upper bounds, the A* search again clearly outperforms the MIP approach by a large margin, especially on the largest instances. Chuffed is not able to return any upper bounds.

8 Conclusions and Future Work

We introduced the PC-JSOCMSR as a practically relevant extended variant of the formerly considered JSOCMSR (Horn et al. 2017). The essential differences are that now the considered jobs have prizes, not all jobs can, in general, be scheduled due the time windows, and that we therefore have to select a subset of the jobs to schedule. Instead of the makespan, we maximize the total prize of the scheduled jobs. This changes in the problem statement substantially affect the structure of the problem and make it in practice much more challenging to solve.

For all our experiments we considered benchmark instances inspired from two practically relevant application scenarios: daily particle therapy scheduling of cancer treatments and the pre-runtime scheduling of avionic systems. We showed that small to medium sized instances of up to about 40 jobs can almost consistently be solved to proven optimality by our A* search. The state graph plays a fundamental role in achieving the reported computational efficiency. Our state graph follows a natural

Table 2: Average results of A*, MIP, and CP.

			A*, $Z^{\text{ub}}_{*}(P,t)$					MIP					CP, Chuffed		
type	n	m	%-opt	$\overline{Z^{\text{lp}}}$	$\overline{Z^{\text{ub}}}$	$\overline{\%\text{-gap}}$	t[s]	%-opt	$\overline{Z^{\text{lp}}}$	$\overline{Z^{\text{ub}}}$	$\overline{\%\text{-gap}}$	t[s]	%-opt	$\overline{Z^{\text{lp}}}$	t[s]
B	10	2	**100**	**30.93**	**30.93**	**0.00**	<0.1	**100**	**30.93**	**30.93**	**0.00**	<0.1	**100**	**30.93**	0.8
B	20	2	**100**	**50.37**	**50.37**	**0.00**	<0.1	**100**	**50.37**	**50.37**	**0.00**	0.1	**100**	**50.37**	0.4
B	30	2	**100**	**75.33**	**75.33**	**0.00**	<0.1	**100**	**75.33**	**75.33**	**0.00**	4.4	**100**	**75.33**	0.6
B	40	2	**100**	**98.93**	**98.93**	**0.00**	0.2	**100**	**98.93**	**98.93**	**0.00**	69.0	**100**	**98.93**	3.0
B	50	2	**100**	**123.27**	**123.27**	**0.00**	3.9	30	122.73	144.60	14.13	900.3	**100**	**123.27**	56.3
B	60	2	**100**	**146.80**	**146.80**	**0.00**	53.7	0	143.73	218.37	33.91	900.2	13	142.23	900.0
B	70	2	**56**	**168.00**	**175.93**	**4.41**	712.0	0	165.33	308.00	46.12	900.1	0	154.93	900.0
B	80	2	**3**	184.13	**211.23**	**12.71**	900.0	0	**189.27**	370.97	48.95	900.4	0	171.70	900.0
B	90	2	**0**	198.87	**251.47**	**20.69**	900.0	**0**	**215.27**	457.10	52.80	900.2	**0**	188.53	900.0
B	10	3	**100**	**36.17**	**36.17**	**0.00**	<0.1	**100**	**36.17**	**36.17**	**0.00**	<0.1	**100**	**36.17**	0.6
B	20	3	**100**	**59.27**	**59.27**	**0.00**	<0.1	**100**	**59.27**	**59.27**	**0.00**	0.1	**100**	**59.27**	0.5
B	30	3	**100**	**86.30**	**86.30**	**0.00**	<0.1	**100**	**86.30**	**86.30**	**0.00**	4.6	**100**	**86.30**	0.5
B	40	3	**100**	**112.00**	**112.00**	**0.00**	0.5	**100**	**112.00**	**112.00**	**0.00**	92.2	**100**	**112.00**	4.0
B	50	3	**100**	**140.33**	**140.33**	**0.00**	10.0	10	138.70	175.73	20.02	900.5	93	140.20	116.9
B	60	3	**87**	**163.97**	**166.00**	**1.18**	138.4	0	161.97	235.63	30.87	900.6	13	160.43	900.0
B	70	3	**40**	184.57	**202.60**	**8.41**	871.8	0	**187.20**	319.23	41.22	900.1	0	179.60	900.0
B	80	3	**10**	205.40	**247.40**	**16.67**	900.0	0	**215.97**	388.50	44.21	900.0	0	202.17	900.0
B	90	3	**0**	224.90	**282.60**	**20.12**	900.0	**0**	**240.67**	469.97	48.70	900.0	**0**	220.80	900.0
S	10	2	**100**	**50.93**	**50.93**	**0.00**	<0.1	**100**	**50.93**	**50.93**	**0.00**	<0.1	**100**	**50.93**	0.5
S	20	2	**100**	**89.93**	**89.93**	**0.00**	<0.1	**100**	**89.93**	**89.93**	**0.00**	0.3	**100**	**89.93**	0.3
S	30	2	**100**	**131.37**	**131.37**	**0.00**	0.1	**100**	**131.37**	**131.37**	**0.00**	17.8	**100**	**131.37**	0.8
S	40	2	**100**	**180.07**	**180.07**	**0.00**	1.1	60	179.87	192.80	5.92	671.4	**100**	**180.07**	10.9
S	50	2	**100**	**225.67**	**225.67**	**0.00**	17.4	0	219.97	343.97	35.61	900.7	46	222.23	900.0
S	60	2	**43**	**269.70**	**287.90**	**6.07**	900.0	0	265.73	476.83	44.06	900.4	0	254.83	900.0
S	70	2	**3**	**305.23**	**347.67**	**11.93**	900.0	0	304.40	600.70	49.19	900.3	0	289.53	900.0
S	80	2	**7**	344.93	**395.30**	**12.60**	881.7	0	346.23	715.73	51.51	900.3	0	318.50	900.0
S	90	2	**0**	374.60	**460.60**	**18.42**	856.2	**0**	**382.87**	844.50	54.55	900.1	**0**	349.60	900.0
S	10	3	**100**	**51.97**	**51.97**	**0.00**	<0.1	**100**	**51.97**	**51.97**	**0.00**	<0.1	**100**	**51.97**	0.4
S	20	3	**100**	**96.47**	**96.47**	**0.00**	<0.1	**100**	**96.47**	**96.47**	**0.00**	0.4	**100**	**96.47**	0.3
S	30	3	**100**	**135.90**	**135.90**	**0.00**	<0.1	**100**	**135.90**	**135.90**	**0.00**	14.7	**100**	**135.90**	0.9
S	40	3	**100**	**185.43**	**185.43**	**0.00**	2.0	33	185.03	209.70	10.85	900.3	**100**	**185.43**	20.4
S	50	3	**97**	**234.20**	**234.53**	**0.15**	35.2	0	230.07	364.60	36.47	900.1	13	228.33	900.0
S	60	3	**47**	**277.90**	**296.17**	**5.97**	876.3	0	276.50	491.83	43.60	900.1	0	262.93	900.0
S	70	3	**10**	305.83	**352.70**	**12.89**	900.0	0	**313.13**	604.00	48.04	900.1	0	292.00	900.0
S	80	3	**0**	347.10	**413.33**	**15.85**	900.0	**0**	**357.93**	734.10	51.14	900.3	**0**	324.80	900.0
S	90	3	**0**	362.43	**464.37**	**21.77**	887.4	**0**	**394.27**	836.17	52.66	900.0	**0**	355.30	900.0
A	10	3	**100**	**422.13**	**422.13**	**0.00**	<0.1	**100**	**422.13**	**422.13**	**0.00**	14.2	**100**	**422.13**	1.0
A	20	3	**100**	**707.27**	**707.27**	**0.00**	0.1	**100**	**707.27**	**707.27**	**0.00**	15.1	**100**	**707.27**	1.1
A	30	3	**100**	**903.97**	**903.97**	**0.00**	2.0	83	**903.97**	908.40	0.47	42.6	**100**	**903.97**	3.3
A	40	3	**97**	1026.10	**1030.00**	**0.36**	40.0	0	**1027.90**	1126.87	8.76	900.7	37	979.53	900.0
A	50	3	**33**	1056.67	**1177.67**	**9.96**	377.2	0	**1114.87**	1360.57	18.03	900.2	0	887.03	900.0
A	60	3	**3**	1011.97	**1225.57**	**17.28**	416.6	0	**1105.33**	1506.27	26.56	900.1	0	807.13	900.0
A	70	3	**0**	1009.63	**1270.53**	**20.44**	442.8	**0**	**1116.90**	1696.97	34.11	900.1	**0**	803.57	900.0
A	80	3	**0**	985.60	**1311.53**	**24.78**	412.5	**0**	**1106.03**	1876.03	40.98	900.0	**0**	746.60	900.0
A	90	3	**0**	977.37	**1323.67**	**26.13**	445.5	**0**	**1095.27**	2055.27	46.65	906.0	**0**	726.13	900.0
A	10	4	**100**	**451.07**	**451.07**	**0.00**	<0.1	**100**	**451.07**	**451.07**	**0.00**	<0.1	**100**	**451.07**	0.7
A	20	4	**100**	**751.73**	**751.73**	**0.00**	0.1	**100**	**751.73**	**751.73**	**0.00**	2.6	**100**	**751.73**	1.2
A	30	4	**100**	**951.97**	**951.97**	**0.00**	2.9	83	**951.97**	959.73	0.77	208.3	**100**	**951.97**	13.9
A	40	4	**93**	1058.10	**1062.07**	**0.36**	68.1	0	**1058.80**	1191.53	11.11	900.2	17	970.57	900.0
A	50	4	**20**	1082.63	**1196.73**	**9.30**	397.3	0	**1127.73**	1443.30	21.79	900.1	0	874.40	900.0
A	60	4	**10**	1065.37	**1250.40**	**14.60**	402.6	0	**1143.90**	1592.60	28.10	900.1	0	821.60	900.0
A	70	4	**3**	1043.47	**1294.20**	**19.28**	393.5	0	**1145.40**	1769.70	35.25	900.0	0	809.90	900.0
A	80	4	**0**	990.23	**1328.47**	**25.43**	405.8	**0**	**1130.87**	1936.70	41.57	900.0	**0**	764.00	900.0
A	90	4	**0**	1019.07	**1347.33**	**24.32**	425.0	**0**	**1128.27**	2090.50	45.98	900.0	**0**	734.37	900.0

node representation and features a state strengthening, which reduces the number of nodes that must in general be considered. A particularity of our approach is that we only add states that are known to have successor states to the open list and we do not have a single dedicated target node. This also reduces the size of the open list.

Most crucial for the performance of the A* search clearly is the choice of the heuristic function, which in our case is an upper bound for the total prize that may still be achieved from a state. To this end, we considered a relaxation of the PC-JSOCMSR that corresponds to an LP relaxation of a multidimensional knapsack problem. Solving this relaxation for each node under consideration turned out not to be the most effective choice. Instead, further simplifications based on constraint and Lagrangian relaxation, respectively, yield fast-to-calculate upper bound functions $Z_0^{\mathrm{ub}}(P,t)$, $Z_{00}^{\mathrm{ub}}(P,t)$, $Z_{0\bar{j}}^{\mathrm{ub}}(P,t)$, and $Z_*^{\mathrm{ub}}(P,t)$. While $Z_*^{\mathrm{ub}}(P,t)$ is the strongest of these four, $Z_0^{\mathrm{ub}}(P,t)$ is fastest to compute. In our experiments in the context of the A* search, using $Z_*^{\mathrm{ub}}(P,t)$ showed to pay off by usually being the best choice. It yields proven optimal solutions more frequently within the allowed time and memory limits and requires less states to be considered. Not requiring a state-of-the-art LP solver might also be a further advantage in some commercial applications.

We further compared the proposed A* search to an order-based MIP model as well as a MiniZinc CP formulation that was solved with Chuffed. These other approaches, however, are clearly inferior concerning the computation times for proven optimal solutions or obtained upper bounds (actually, the CP approaches are not able to provide any upper bounds in case of early termination). An explanation for the better performance of the A* search besides the well working heuristic function seems to be that it can effectively exploit dynamic programming aspects: Frequently, a state can be reached via multiple different job sequences, and the corresponding subproblem is then only solved once.

In cases where the limits have been reached, slightly better final upper bounds could frequently be achieved by the A* search than from the CP and MIP approaches. However, we remark that our A* search should not be considered for just heuristically solving significantly larger PC-JSOCMSR instances, where it is obvious from the beginning that the search must be terminated much earlier than optimality can be proven: Complete solutions (in the sense that no further jobs can be scheduled) are usually only found very late during the whole search. The approach that we use, that is, to augment the most promising partial solution in a greedy way, was just implemented to ensure that we always return at least one complete solution.

Classical A* search is targeted towards exact solving, which was our focus here. To address substantially larger instances heuristically, other methods like metaheuristics are more appropriate, see Section 3. An interesting research direction are also anytime A* search variants, which trade longer runtimes to achieve proven optimality for the benefit of producing promising complete solutions already early.

If our A* terminates before a proven optimal solution could be found then this frequently happens due to exceeding the memory limit. In further research, it may thus be worth to think about even stronger upper bounds for the still achievable prizes of states to reduce the number of state expansions and consequently also the memory usage. A possible way to achieve this could be the usage of relaxed decision diagrams, which represent discrete relaxations of combinatorial optimization problems and can be seen as alternative to LP-based relaxations.

References

Allahverdi A (2016) A survey of scheduling problems with no-wait in process. European Journal of Operational Research 255(3):665–686

Blikstad M, Karlsson E, Lööw T, Rönnberg E (2018) An optimisation approach for pre-runtime scheduling of tasks and communication in an integrated modular avionic system. Optimization and Engineering 19(4):977–1004

Gilmore PC, Gomory RE (1964) Sequencing a one-state variable machine: A solvable case of the traveling salesman problem. Operations Research 12(5):655–679

Gunawan A, Lau HC, Vansteenwegen P (2016) Orienteering problem: A survey of recent variants, solution approaches and applications. European Journal of Operational Research 255(2):315 – 332

Hansen EA, Zhou R (2007) Anytime heuristic search. Journal of Artificial Intelligence Research 28:267–297

Hart P, Nilsson N, Raphael B (1968) A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics 4(2):100–107

Hartmann S, Briskorn D (2010) A survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Operational Research (207):1–14

Horn M, Raidl G, Blum C (2017) Job sequencing with one common and multiple secondary resources: A problem motivated from particle therapy for cancer treatment. In: The Third International Conference on Machine Learning, Optimization and Big Data, MOD 2017, Springer, LNCS, vol 10710, pp 506–518

Horn M, Maschler J, , Raidl G, Rönnberg E (2018a) A*-based construction of decision diagrams for a prize-collecting scheduling problem. Tech. Rep. AC-TR-18-011, Algorithms and Complexity Group, TU Wien

Horn M, Raidl GR, Rönnberg E (2018b) An A* algorithm for solving a prize-collecting sequencing problem with one common and multiple secondary resources and time windows. In: PATAT 2018: Proceedings of the 12th International Conference of the Practice and Theory of Automated Timetabling, Vienna, Austria, pp 235–256

Kellerer H, Pferschy U, Pisinger D (2004) Knapsack Problems. Springer

Likhachev M, Gordon GJ, Thrun S (2004) ARA*: Anytime A* with provable bounds on sub-optimality. In: Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference (NIPS-03), MIT Press, pp 767–774

Maschler J, Raidl GR (2018) Multivalued decision diagrams for a prize-collecting sequencing problem. In: PATAT 2018: Proceedings of the 12th International Conference of the Practice and Theory of Automated Timetabling, Vienna, Austria, pp 375–397

Maschler J, Riedler M, Stock M, Raidl GR (2016) Particle therapy patient scheduling: First heuristic approaches. In: PATAT 2016: Proceedings of the 11th International Conference of the Practice and Theory of Automated Timetabling, Udine, Italy, pp 223–244

Maschler J, Hackl T, Riedler M, Raidl GR (2017) An enhanced iterated greedy metaheuristic for the particle therapy patient scheduling problem. In: Proceedings of the 12th Metaheuristics International Conference, pp 465–474

Maschler J, Riedler M, Raidl GR (2018) Particle therapy patient scheduling: Time estimation for scheduling sets of treatments. In: Moreno-Díaz R, Pichler F, Quesada-Arencibia A (eds) Computer Aided Systems Theory – EUROCAST 2017, Part I, Springer, LNCS, vol 10671, pp 364–372

Nemhauser GL, Wolsey LA (1988) Integer and Combinatorial Optimization. Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons

Rios LHO, Chaimowicz L (2010) A Survey and Classification of A* Based Best-First Heuristic Search Algorithms, Springer, pp 253–262

Röck H (1984) The three-machine no-wait flow shop is NP-complete. Journal of the ACM 31(2):336–345

Vadlamudi SG, Aine S, Chakrabarti PP (2016) Anytime pack search. Natural Computing 15(3):395–414

Van der Veen JAA, Wöginger GJ, Zhang S (1998) Sequencing jobs that require common resources on a single machine: A solvable case of the TSP. Mathematical Programming 82(1-2):235–254