Decision Diagram Based Limited Discrepancy Search for a Job Sequencing Problem[⋆]

Matthias Horn and Günther R. Raidl

Institute of Logic and Computation, TU Wien, Vienna, Austria
`{horn|raidl}@ac.tuwien.ac.at`

Abstract. We consider the *Price-Collecting Job Sequencing with One Common and Multiple Secondary Resources* problem. The task is to feasibly schedule a subset of jobs from a given larger set. Each job needs two resources: a common resource for a part of the job's execution time and a secondary resource for the whole execution time. Furthermore each job has one or more time windows and an associated prize. In addition to previous work, we also consider precedence constraints on the jobs. We aim to maximize the total prize over the actually scheduled jobs. To solve large instances heuristically we propose a hybrid of limited discrepancy search and beam search approach that utilize a relaxed decision diagram. We could show that the use of a relaxed decision diagram substantially speed-up the computation times of the search approach.

Keywords: sequencing problem, decision diagrams, limited discrepancy search

1 Introduction

The *Price-Collecting Job Sequencing with One Common and Multiple Secondary Resources* (PC-JSOCMSR) problem without precedence constraints was first introduced by [6, 7] and consists of a set of jobs, one common resource, and a set of secondary resources. The common resource is shared by all jobs whereas a secondary resource is shared only by a subset of jobs. Each job has at least one time window and is associated with a prize. A feasible schedule requires that there is no resource used by more than one job at the same time and each job is scheduled within one of its time windows. Due to the time windows it may not be possible to schedule all jobs. The task is to find a subset of jobs that can be feasible scheduled and maximizes the total prize. There are at least two applications [5]. The first is in the field of the daily scheduling of particle therapies for cancer treatments. The second application can be found in the field of hard real time scheduling of electronics within an aircraft, called avionics, where the PC-JSOCMSR appears as a subproblem.

[⋆] This project is partially funded by the Doctoral Program Vienna Graduate School on Computational Optimization, Austrian Science Foundation Project No. W1260-N35.

The PC-JSOCMSR was tackled on the exact side by Horn et al. [6], with an A* based algorithm which is able to solve instances up to 40 jobs to proven optimality. On the heuristic side, Maschler and Raidl [7] applied *decision diagrams* (DDs) to obtain lower and upper bounds for large problem instances with up to 300 jobs. DDs are rooted weighted directed acyclic graphs and provide graphical representations of the solution spaces of combinatorial optimization problems. In particular relaxed DDs represent a superset of the feasible set of solutions and are therefore a *discrete relaxation* of the solution space, providing upper bounds on the objective value. The counterparts are restricted DDs which represent subsets of feasible solutions and therefore provide heuristic solutions. Both types of DDs were investigated in [7] and were compiled with adapted standard methods from the literature. For more details on DDs we refer to [1]. New state of the art results for the PC-JSOCMSR could be obtained by Horn et al. [5] by applying a *beam search* (BS) heuristic that uses a relaxed DD to speed up the search. The relaxed DD is constructed by a novel A*-based construction algorithm.

In particular in the avionic system scenario it often appears that some jobs need to be finished before other jobs may start. To address this aspect, we consider in this work also *precedence constraints*. Thus, there are given relationships between pairs of jobs as additional input such that one job can only be scheduled if the other job is already completely scheduled earlier. These new constraints require an adaption on the algorithmic side of [5] to incorporate the new precedence constraints. The goal is to solve large problem instances of the PC-JSOCMSR with precedence constraints heuristically. Our solution approach builds upon the ideas from [5] but extended them to a *limited discrepancy search* (LDS) combined with BS that exploits structural information contained in a relaxed DD. The usage of the relaxed DD is two-folded: (1) to reduce computation time of the LDS and (2) to provide besides a heuristic solution also an upper bound on the total prize objective.

2 PC-JSOCMSR with Precedence Constraints

The PC-JSOCMSR with precedence constraints is formally defined as follows. Given are a set of n jobs $J = \{1, \dots, n\}$, a common resource 0 and a set of m secondary resources $R = \{1, \dots, m\}$. Let $R_0 = \{0\} \cup R$ be the complete set of resources. Each job $j \in J$ needs during its whole execution time $p_j > 0$ one secondary resource $q_j \in R$ and, in addition, after some preprocessing time $p_j^{\mathrm{pre}} \geq 0$ also the common resource 0 for some time $p_j^0 > 0$. Furthermore each job has associated (1) ω_j time windows $W_j = \bigcup_{\omega=1,\dots,\omega_j} [W_{j\omega}^{\mathrm{start}}, W_{j\omega}^{\mathrm{end}}]$, where $W_{j\omega}^{\mathrm{end}} - W_{j\omega}^{\mathrm{start}} \geq p_j$, $\omega = 1, \dots, \omega_j$, (2) a set of preceding jobs Γ_j, which must be scheduled before job j can be scheduled w.r.t. the common resource 0, and (3) a prize $z_j > 0$. The task is to find a subset of jobs $S \subseteq J$ which can be feasible scheduled such that the total prize of these jobs is maximized: $Z^* = \max_{S \subseteq J} Z(S)$, $Z(S) = \sum_{j \in S} z_j$. A feasible schedule assigns each job in S a starting time in such a way that all constraints are satisfied. Note that a unique ordered sequence $\pi = (\pi)_{i=1,\dots,|S|}$ of jobs is implied by each feasible schedule of jobs $S \subseteq J$, since the common resource is required by each job and only one job can use this resource at a time. For each given sequence π of jobs S that can be associated with a feasible schedule, a *normalized schedule* without unnecessary waiting times can be computed greedily (see [6] for further details).

3 Exact/Relaxed Decision Diagrams and Filtering

In order to describe our approach in Section 4 we have to introduce some definition and structures beforehand. In our context a DD for the PC-JSOCMSR is a weighted directed acyclic graph $M = (V, A)$ with one root node $\mathbf{r} \in V$, corresponding to the empty schedule and one target node $\mathbf{t} \in V$ corresponding to all feasible schedules that cannot be further extended by any job. Each arc $a = (u, v) \in A$ corresponds to adding a specific job, denoted by $\text{job}(a) \in J$, as the next job after the ones already scheduled up to node u. The length of an arc $a \in A$ is associated width the prize $z_{\text{job}}(a)$. Hence, each path from $\mathbf{r}$ to any node $u \in V$ corresponds to a specific sequence of jobs π and the length of the path is equal to the sum of prizes of jobs in π.

In an exact DD each feasible normalized schedule $S \subseteq J$ has a corresponding path in the exact DD originating from $\mathbf{r}$ and vice versa. The length of such a path corresponds exactly to the total prize $Z(S)$. Therefore a longest path from $\mathbf{r}$ to $\mathbf{t}$ corresponds to an optimal solution of the PC-JSOCMSR. Furthermore, each node $u \in V$ is associated to a state $(P(u), t(u))$, where set $P(u)$ contains all jobs that can be feasibly scheduled next, and vector $t(u) = (t_r(u))_{r \in R_0}$ contains the earliest times from which on each of the resources are available for performing a next job. The transition function to obtain the successor state $(P(v), t(v))$ of state $(P(u), t(u))$ when scheduling job $j \in P(u)$ is

$$\tau\left((P(u), t(u)), j\right) = \begin{cases} (P(u) \setminus \{j\}, t(v)), & \text{if } \mathrm{s}((P(u), t(u)), j) < T^{\max} \wedge \\ & \qquad P(u) \cap \Gamma_j \neq \emptyset, \\ \hat{0}, & \text{else}, \end{cases} \tag{1}$$

with

$$t_0(v) = \mathrm{s}((P(u), t(u)), j) + p_j^{\text{pre}} + p_j^0, \tag{2}$$

$$t_r(v) = \mathrm{s}((P(u), t(u)), j) + p_j, \qquad \text{for } r = q_j, \tag{3}$$

$$t_r(v) = t_r(u), \qquad \text{for } r \in R \setminus \{q_j\}, \tag{4}$$

where $\hat{0}$ represents the infeasible state and $s((P(u), t(u)), j)$ corresponds to the earliest start time of job j w.r.t. to state $(P(u), t(u))$ and job j's time windows. If it is not possible to schedule job j feasible then function $s(.,.)$ will return $T^{\max}$. States that are related to exact DDs will be denoted as *exact* states.

Relaxed DDs merge exact states in order to get a more compact DD. Thereby new paths will emerge which correspond to infeasible schedules, denoted as *infeasible* paths. Let merge two nodes $u, v \in V$. The merged state is $(P(u), t(u)) \oplus (P(v), t(v)) = (P(u) \cup P(v), (\min(t_r(u), t_r(v)))_{r \in R_0})$. We compile relaxed DDs with the A*-based compilation (A*C) method from [5] since it could be shown that at least for the PC-JSOCMSR without precedence constraints A*C can produce smaller relaxed DDs in shorter time that represent stronger relaxations than relaxed DDs compiled with standard methods from the literature. Note that we initially ignore the precedence constraints in this compilation of a relaxed DD. Otherwise, we would need to extend the states of the nodes with additional information in order to define a feasible merging rule for two nodes. Preliminary experiments had shown that those larger states cause substantially longer compilation times, which we want to avoid. However, we consider the precedence constraint after the initial construction by applying a respective filtering on the compiled relaxed DD. We try to identify arcs which belong only to infeasible paths. Those arcs can be safely removed from the relaxed DD to reduce the number of infeasible paths without removing paths that correspond to feasible schedules. To identify arcs that violate precedence constraints we adopted the corresponding filter operation suggested by Cire and van Hoeve [2]. Moreover, if we got already a primal solution then we can in addition filter arcs which only belong to paths corresponding to solutions that are worse than this known primal solution. Hence, paths that encode sub-optimal solutions will be removed from the relaxed DD. This cost-based filter operations are adopted from [5].

4 Limited Discrepancy Search

Limited discrepancy search was originally proposed by Harvey and Ginsberg [4] for heuristic binary searches where at each decision point a heuristic $h(.)$ decides between two possibilities to extend the current partial solution. If $h(.)$ would be a perfect heuristic than the algorithm would return the optimal solution as soon as a complete solution is encountered during the search. However, in most cases $h(.)$ will fail at some point and only a non-optimal solution can be returned. To overcome this, LDS allows in a systematic way discrepancies during the search. A *discrepancy* means that at some decision point the algorithm decides against $h(.)$. Hence, if k discrepancies are allowed than LDS will encounter all paths in the search tree where the algorithm exactly decides k times against $h(.)$. To apply LDS on the PC-JSOCMSR we have to consider in general multiple possibilities at each decision point instead of just two, and we do this by counting $i-1$ discrepancies if we take the i-th best decision according to $h(.)$.

Algorithm 1 shows our LDS-based approach. The search is applied on the exact states defined in Section 3. Note that we do not build an exact DD, but we rather keep all not yet expanded nodes in memory and assign to each node v' the so far best encountered partial solution $\pi(v')$. Furthermore, we extend LDS in similar ways as Furcy and Koening [3] by incorporating a BS like approach at each level into LDS. Instead of expanding always one node at each step Algorithm 1 expands at each step β nodes and keeps the $(k+1)\beta$-best successor nodes according to $h(.)$. As heuristic decision function $h(v')$ for node v'

```
Algorithm 1: LDSprobe
  Input: node set $N'$, relaxed DD $M = (V, A)$, allowed discrepancies $k$, beam
         width $\beta$
  Output: sequence of jobs $\pi$
1 if $N' = \emptyset$ then return empty sequence;
2 node set $W' \leftarrow \emptyset$; job sequence $\pi_{\text{best}} \leftarrow \emptyset$;
3 foreach $u' \in N'$ do
4     let $u \in V$ be the node corresponding to $u'$ w.r.t. the path from the root;
5     foreach outgoing arc $a = (u, v)$ of node $u$ do
6         if $|W'| = (k+1)\,\beta \wedge$ node $v$ would be removed from $W' \cup \{v\}$ then
7             continue with next arc;
8         if $\tau((P(u'), t(u'), \text{job}(a)) = \emptyset$ then continue with next arc;
9         add new node $v'$ to $W'$ and set $(P(v'), t(v')) \leftarrow \tau((P(u'), t(u')), \text{job}(a))$;
10        if $|W'| > (k+1)\beta$ then remove worst node from $W'$ according to $h(.)$;
11 if $W' = \emptyset$ then return $\arg\max_{\pi(u') | u' \in N'} Z(\pi(u'))$;
12 sort $W'$ according to $h(.)$ and split $W'$ into $k+1$ slices $W'[i]$, $i = 0, \ldots, k$;
13 foreach $i = k, \ldots, 0$ do
14     $\pi = \text{LDSprobe}(W'[i], M, k - i, \beta)$;
15     if $Z(\pi_{\text{best}}) < Z(\pi)$ then $\pi_{\text{best}} \leftarrow \pi$;
16 return $\pi_{\text{best}}$;
```

```
Algorithm 2: LDS+BS
  Output: sequence of jobs $\pi$
1 compile relaxed DD $M = (V, A)$ by A$^*$C, ignoring precedence constraints;
2 $\pi_{\text{best}} \leftarrow \text{LDSprobe}(\mathbf{r}, M, 0, 10)$;
3 for $k \leftarrow 0$; $k \leq k_{\max} \wedge$ time limit not exceeded; $k \leftarrow k+1$ do
4     apply filtering on $M$;
5     $\pi \leftarrow \text{LDSprobe}(\mathbf{r}, M, k, \beta)$;
6     if $Z(\pi_{\text{best}}) < Z(\pi)$ then $\pi_{\text{best}} \leftarrow \pi$;
7 return $\pi_{\text{best}}$;
```

we use the ratio $Z(\pi(v'))/t_0(v')$. In order to quickly identify those $(k+1)\beta$-best successor nodes we use the structural information contained in the relaxed DD $M = (V, A)$ similar as in [5]. For node $u' \in N$ a corresponding node $u \in V$ from M can be determined by following the job sequence $\pi(u')$ from $\mathbf{r}$ in M. We do not consider transitions to successor nodes of u' where the corresponding arcs were removed from the relaxed DD during the filtering step. Furthermore we can estimate $h(.)$ without creating the successor nodes of u' by using the corresponding nodes in the relaxed DD. Based on these estimation we can decide quickly if a successor node is a candidate to be one of the β-best successor nodes or not. Note that for simplification reasons Algorithm 1 shows a recursive version of LDS, our implementation however, is implemented in an iterative way.

Algorithm 2 gives an overview of the overall approach to tackle the PC-JSOCMSR. First a relaxed DD M is compiled with A*C with the same parameter settings as in [5] and by ignoring the precedence constraints. In order to get quickly an initial primal solution for filtering, Algorithm 1 is applied with the small beam with $\beta = 10$ and no allowed discrepancies. In the main loop we apply first the filtering for sup-optimal paths according to our current best primal solution and precedence constraints violations. Then we apply Algorithm 1 with beam width β and the number of current maximum allowed discrepancies k. After updating the incumbent solution π_{best}, k is increased by one. The algorithm terminates if the maximum allowed discrepancies $k_{\max}$ is reached or a certain time limit is exceeded.

5 Computational Study

The LDS-based algorithm for the PC-JSOCMSR with precedence constraints was implemented in C++ using GNU g++ 5.4.1. 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 memory limit of 16 GB per run. We extended the two sets of benchmark instances for the particle therapy application scenario (denoted as P) and for the avionic system scheduling scenario (denoted as A) from [5] by adding randomly precedence constraints between n pairs of jobs such that circular dependencies between jobs are voided. The instance sets contain 30 instances for each combination of different values of n and m. For further details on the benchmark characteristics we refer to [5].

Figure 1 compares the obtained average total prizes and median computation times between LDS+BS and a standalone variant of LDS+BS without using a relaxed DD dependent on different values of beam width β and different maximum allowed discrepancies $k_{\max}$. The diagrams on the top visualize the obtained average total prizes. There are two main observations regarding the solution quality: First, as expected the solution quality tends to increase with increasing β and/or $k_{\max}$; second, similar results could be obtained from both LDS+BS variants. However, regarding computation times, the LDS+BS approach using the relaxed DD is in almost all cases except for $k_{\max} = 0$ and smaller β substantially faster. Note that we do not show the obtained results from standalone LDS+BS for $k_{\max} = 2$, since the approach exceeded in most cases the time limit of two hours.

Figure 2 compares the LDS+BS approach against a mixed linear integer programing (MIP) approach and a constrained programming (CP) approach. The MIP formulation as well as the CP formulation from [5] were adapted to additionally consider the precedence constraints and are solved with Gurbi Optimizer 7.5.1 and MiniZinc 2.1.7 with backbone solver Chuffed, respectively. All tested approaches use a time limit of 900 seconds. For LDS+BS the maximum allowed discrepancies $k_{\max}$ are set to infinity and β is set to 1000 and 10000 for instance set of type P and A, respectively. The first bar of each group of bars show the obtained average longest path length of the compiled relaxed DD

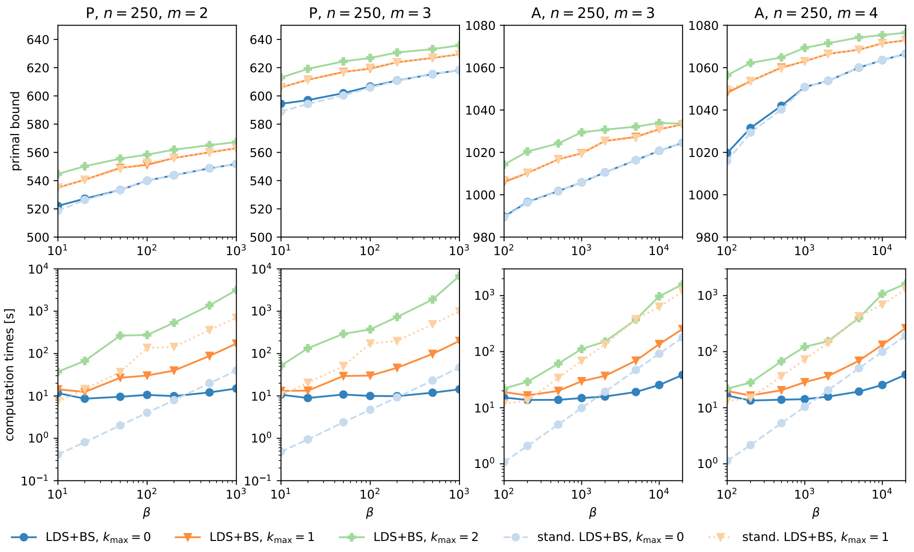

Fig. 1. Comparison between LDS+BS and standalone LDS+BS for middle sized instances with 250 jobs.

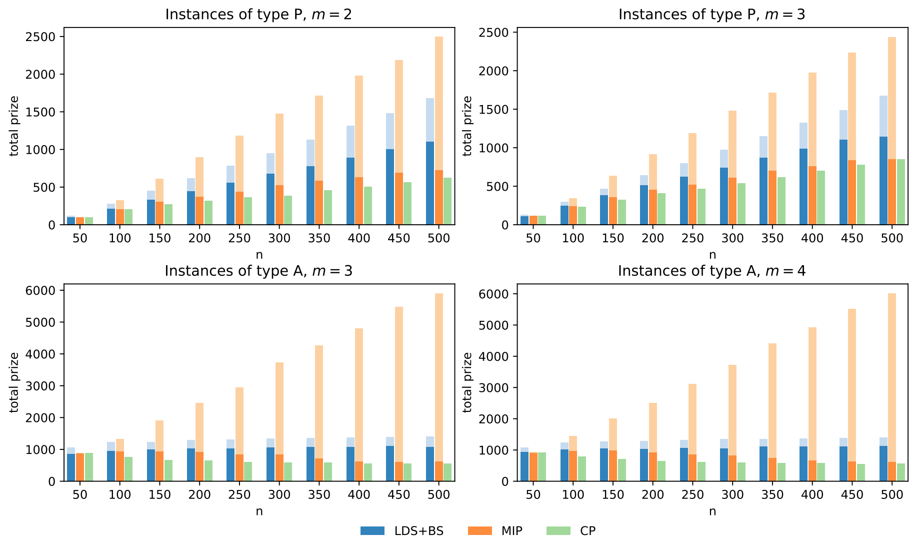

Fig. 2. Primal and Dual Bounds obtainded from LDS+BS, MIP and CP.

during the LDS+BS approach and the block at the bottom show the obtained average primal bounds. In the same manner, the second bar shows the obtained upper- and primal bounds from the MIP approach. The third bar shows the obtained average primal bounds obtained from the CP approach. On average the LDS+BS approach finds in all considered cases better or equally good solutions than the MIP or the CP solvers. Moreover, LDS+BS is able to return in most cases on average stronger upper bounds than the MIP solver.

6 Conclusion

Exploiting the structural information of relaxed DDs within LDS has following advantages: (1) a substantial speed up of the heuristic search allows to scan larger regions of the search space compared to a standalone LDS approach and (2) a dual bound can be obtained from the relaxed DD. Although we demonstrate this advantages specifically for the PC-JSOCMSR, the general approach also appears promising for other combinatorial optimization problems. Next steps would be to incorporate other filtering techniques to further strengthen the relaxed DD by removing more arcs to speed-up the computation times even more. Another promising research direction would be to apply the general idea of using the structural information of relaxed DDs on further search heuristics and meta-heuristics.

References

1. Bergman, D., Cire, A.A., van Hoeve, W.J., Hooker, J.N.: Decision Diagrams for Optimization. Artificial Intelligence: Foundations, Theory, and Algorithms, Springer (2016)
2. Cire, A.A., van Hoeve, W.: Multivalued decision diagrams for sequencing problems. Operations Research **61**(6), 1411–1428 (2013)
3. Furcy, D., Koenig, S.: Limited discrepancy beam search. In: Proceedings of the 19th International Joint Conference on Artifical Intelligence. pp. 125–131. IJCAI'05, Morgan-Kaufmann (2005)
4. Harvey, W.D., Ginsberg, M.L.: Limited discrepancy search. In: Mellish, C.S. (ed.) Proceedings of the 14th International Joint Conference on Artificial Intelligence. pp. 607–615. Morgan-Kaufmann, Montreal, Que Canada (1995)
5. Horn, M., Maschler, J., Raidl, G., Rönnberg, E.: A*-based construction of decision diagrams for a prize-collecting scheduling problem. Tech. Rep. AC-TR-18-011, Algorithms and Complexity Group, TU Wien (2018), submitted
6. Horn, M., Raidl, G.R., Rönnberg, E.: An A* algorithm for solving a prize-collecting sequencing problem with one common and multiple secondary resources and time windows. In: Proceedings of PATAT 2018 – The 12th International Conference of the Practice and Theory of Automated Timetabling. pp. 235–256. Vienna, Austria (2018)
7. Maschler, J., Raidl, G.R.: Multivalued decision diagrams for a prize-collecting sequencing problem. In: Proceedings of PATAT 2018 – Proceedings of the 12th International Conference of the Practice and Theory of Automated Timetabling. pp. 375–397. Vienna, Austria (2018)