Markó Horváth

Integer programming approaches for solving routing and scheduling problems

Summary of the PhD thesis

Supervisor: Tamás Kis, DSc
scientific advisor, Institute for Computer Science and Control
assistant professor, Department of Operations Research

Doctoral School of Mathematics
director: Professor Tibor Jordán, DSc

Doctoral Program of Applied Mathematics
director: Professor János Karátson, DSc

Department of Operations Research
Faculty of Science, Eötvös Loránd University

2019
1 Introduction

Today’s companies and industries are teeming with combinatorial optimization problems. For example, at a public transport company on the operational planning phase one has to create (i) daily schedules for the vehicles of their fleet to perform the timetabled trips, minimizing some assets and operational costs; (ii) daily shifts and (iii) rosters over a longer planning period (e.g., weeks or months) for the drivers, satisfying a wide variety of federal regulations, minimizing the labor costs. As another example, machine scheduling problems permanently arise in a company producing or assembling some products. Such a problem involves sequencing a set of jobs (e.g., manufacturing operations of the products) to be processed on a set of resources. Without any claim to completeness, we mention that these problems have different flow patterns (e.g., single machine, parallel machines, shop models), constraints (e.g., precedence constraints, setup times, release dates, no-wait operations) and objectives (e.g., minimizing costs, penalties, makespan, or increasing production throughput) based on the particularity of the companies and the products. These problems require complex systems containing suitable mathematical models and efficient algorithms to take more and more details of the planning/production process into consideration while keeping the problem computationally tractable.

In the thesis entitled ‘Integer programming approaches for solving routing and scheduling problems’ we investigate three combinatorial optimization problems with great practical relevance. Namely, we deal with (i) the integrated vehicle and crew scheduling problem which is a combination of the vehicle scheduling and the crew scheduling problems mentioned before, (ii) the resource constrained shortest path problem which has several direct real-world applications and also arises as a subproblem to be solved repeatedly in another problems (e.g., the crew scheduling problem), and (iii) the position-based scheduling problem which is a kind of machine scheduling problem. For all these problems we provide exact algorithms, and for one of them we also present an approximation algorithm. In all these exact algorithms, integer programming approaches play a key role. In the booklet at hand we summarize the results of the thesis.

2 Multi-criteria approximation scheme for the resource constrained shortest path problem

2.1 Problem definition

Consider the budgeted version of combinatorial optimization problems in the following form. Given a set $U$ of elements along with a cost function $c : U \to Q$, a finite set $S \subseteq 2^U$ of feasible solutions, and a set of $k$ weight functions $w : U \to Q^k_{\geq 0}$ on the elements along with budget limits $L_i \in Q^k_{\geq 0}$. The $k$-budgeted combinatorial optimization problem is formalized as

$$\text{minimize (or maximize) } c(S) \text{ subject to } S \in S, \text{ and } w_i(S) \leq L_i \text{ for all } i = 1, \ldots, k$$

(1)
with \( c(S) := \sum_{e \in S} c(e) \) and \( w_i(S) := \sum_{e \in S} w_i(e) \) for all \( i = 1, \ldots, k \).

The Resource Constrained Shortest Path Problem (RCSPP) refers to the minimization version of the \( k \)-budgeted \( s \)-\( t \) Path Problem \((1)\), where \( S \) is the set of all \( s \)-\( t \) paths of a directed graph \( D = (V, U) \) with designated nodes \( s,t \in V \).

### 2.2 Preliminaries

A multi-criteria \((a_0; a_1, \ldots, a_k)\)-approximation algorithm, \( a_i \geq 1 \), for a \( k \)-budgeted optimization problem \( \Pi \) is an algorithm which finds an \( a_0 \)-approximate solution \( S \subseteq S \) to the problem (that is, \( c(S) \geq c(S_{OPT})/a_0 \) if \( \Pi \) is a maximization problem, and \( c(S) \leq a_0 c(S_{OPT}) \) if \( \Pi \) is a minimization one, where \( S_{OPT} \) is an optimal solution) such that \( w_i(S) \leq a_i L_i \) for all \( i = 1, \ldots, k \). A multi-criteria polynomial time approximation scheme (multi-criteria PTAS) contains a multi-criteria \((a_0; a_1, \ldots, a_k)\)-approximation algorithm \( A_\epsilon \) with \( a_i \leq 1 + \epsilon \) for any \( \epsilon > 0 \). A multi-criteria fully polynomial time approximation scheme (multi-criteria FPTAS) is a PTAS such that \( A_\epsilon \) runs in polynomial time in \( 1/\epsilon \) as well.

### 2.3 Related work

In the case of \( 2 \leq k = O(1) \), one can obtain a multi-criteria \((1 + \epsilon; 1 + \epsilon, \ldots, 1 + \epsilon)\)-FPTAS for several \( k \)-budgeted combinatorial optimization problems (including RCSPP) based on the general technique of Papadimitriou and Yannakakis (2000), however, unless \( P=NP \), there exists no \((a_0; a_1, \ldots, a_k)\)-approximation algorithm with two or more \( a_i \)'s equal to 1 for some of these problems (see (Grandoni et al., 2014)). Further on, Grandoni et al. (2014) describe multi-criteria \((1; 1 + \epsilon, \ldots, 1 + \epsilon)\)-PTASs for a number of \( k \)-budgeted combinatorial optimization problems, however, they do not provide such an algorithm for RCSPP, which was one of the motivations for our work. Another motivation was that the method of Papadimitriou and Yannakakis (2000) and the results of Grandoni et al. (2014) work only if the number of weight functions, \( k \), is a constant.

### 2.4 Our contribution

We investigated multi-criteria approximation algorithms for some budgeted combinatorial optimization problems. Our positive and negative results, along with the results of Grandoni et al. (2014) give a complete picture on the approximability of RCSPP in terms of approximation schemes. We emphasize that our results are valid for general graphs with non-negative weights and arbitrary costs, however, cycles with negative total cost are not allowed. These results were published in (Horváth and Kis, 2018).

**Theorem 1** (Horváth and Kis (2018)). If \( P \neq NP \), and the number of weight functions, \( k \), is not a constant (i.e., part of the input), then there is no polynomial time multi-criteria approximation scheme either for the minimization or for the maximization versions of the \( k \)-budgeted \( s \)-\( t \) Path Problem,
the k-budgeted Spanning Tree Problem, the k-budgeted Matroid Basis Problem, and the k-budgeted Bipartite Perfect Matching Problem.

Theorem 2 (Horváth and Kis (2018)). If the number of weight functions, k, is a constant (i.e., not part of the input), then there exists a fully polynomial time \((1; 1 + \epsilon, \ldots, 1 + \epsilon)\)-approximation scheme for the Resource Constrained Shortest Path Problem.

3 LP-based methods for the resource constrained shortest path problem

3.1 Problem definition and formulation

Recall that an instance of the Resource Constrained Shortest Path Problem (RCSPP) is given by a directed graph \(D = (V, A)\) with designated nodes \(s, t \in V\), a cost function \(c : A \to \mathbb{Q}\) on the arcs, and a set of \(k\) weight functions \(w : A \to \mathbb{Q}^k\) on the arcs (also called resource functions) along with limits \(L \in \mathbb{Q}^k\) (also called resource limits); and a minimal cost \(s\)-\(t\) path \(\pi\) is sought such that the resource limits on the path are not violated, that is, \(\sum_{e \in \pi} w_i(e) \leq L_i\) holds for all \(i = 1, \ldots, k\). Note that we allow negative weights and thus non-positive limits as well, however, we assume that the underlying directed graph \(D\) contains directed cycle neither of negative total cost nor of negative total resource consumptions for any of the resources.

We omit the standard integer linear programming formulation of the problem, however, we mention that in this formulation a binary variable \(x_e\) is assigned to each arc \(e \in A\) indicating whether the path sought goes through the arc \(e\) or not. By this, we denote with \(S^{RCSPP}\) the set of incidence vectors \(x \in \{0, 1\}^A\) corresponding to resource feasible \(s\)-\(t\) paths, and with \(P^{RCSPP} := \text{conv}(S^{RCSPP})\) the polytope of these feasible \(s\)-\(t\) paths.

3.2 Our contribution

We proposed linear programming (LP) based branch-and-bound methods to solve RCSPP. These methods include several components, these are, (i) a primal heuristics, (ii) a variable fixing procedure, and (iii) the separation of valid inequalities for \(P^{RCSPP}\). These results were published in (Horváth and Kis, 2016).

3.2.1 Primal heuristics

The basic idea of our primal heuristics is to find a feasible \(s\)-\(t\) path in the support graph \(D_\bar{x}\) of the solution \(\bar{x} \in [0, 1]^A\) of the node-LP of the corresponding enumeration tree node (that is, \(D_\bar{x} = (V, A_\bar{x})\) with \(A_\bar{x} = \{ e \in A : \bar{x}_e > 0 \}\)), if any, and update the best upper bound on an optimal solution to improve the branch-and-bound procedure. The basis of this idea is the observation that \(\bar{x}\) is a convex combination of \(s\)-\(t\) paths (note we can omit cycles with 0 cost, if any). Briefly stated, we perform a depth-first search from \(s\) on \(D_\bar{x}\), and once we reach a processed node \(v\), we can examine one or more \(s\)-\(t\) paths through \(v\).
3.2.2 Variable fixing procedure

In an enumeration tree node some variables may be already fixed to 0 or 1 (e.g., due to branching decisions), that is, in the subtree rooted at that tree node we seek a minimum cost, feasible s-t path which passes through every arc \( e \) such that \( x_e \) is fixed to 1, and avoids each arc \( e \) such that \( x_e \) is fixed to 0. We call such an s-t path proper. Briefly stated, we obtained a directed graph from the original (possibly preprocessed) graph by removing nodes and arcs that cannot appear in any proper path, then we applied the preprocessing procedure of Dumitrescu and Boland (2003) (DB-preprocessing) in order to eliminate further nodes and arcs from this graph that cannot be in any feasible s-t path with respect to a single resource, or in any s-t path with cost at most \( U \), where \( U \) is an upper bound on an optimal s-t path. We fixed the variables corresponding to the deleted arcs to 0.

3.2.3 Valid inequalities for \( P_{RCSPP} \)

We generalized two classes of valid inequalities of Garcia (2009). The first class is based on the observation that a feasible s-t path through an arc \( e \) can use only those arcs \( f \) from an s-t cut such that \( e \) and \( f \) are compatible (i.e., the shortest path through these arcs with respect to a single resource does not violate the limit). We generalized these inequalities by considering two arcs instead of a single one, and similarly to Garcia (2009) we also provided a polynomial-time separation procedure. The second class is based on the observation that a feasible s-t path cannot contain all the arcs of an infeasible subpath (that is, a path which cannot be extended to a feasible s-t path with respect to a single resource). Garcia (2009) introduced inequalities where either the first or the last arc of an infeasible subpath is fixed, and we generalized these inequalities by fixing both the starting and ending arcs. We proved that the separation of these inequalities (both the original and the generalized versions) is NP-hard, and similarly to Garcia (2009) we provided a heuristic separation procedure.

3.2.4 Computational results: Evaluation of cutting planes, primal heuristics, and variable fixing

We made thorough computational experiments, where the main goals were (i) to show that some of the new cutting planes can improve the performance of a branch-and-bound procedure for solving RCSPP, (ii) to assess the effectiveness of the new primal heuristics and the variable fixing procedure, and (iii) to find the best combination of the various techniques for solving hard instances. Our solution method was implemented in C++ programming language using Xpress\(^1\) as a branch-and-cut framework.

Our experiments suggest that our primal heuristics and mainly our variable fixing procedure can significantly reduce the execution time of the plain branch-and-bound procedure of Xpress. These tests also show that both of our cutting planes and the cutting planes from the literature can reduce the computation time and the number of investigated enumeration

---

\(^1\)FICO Xpress Solver, version 28.01.09
tree nodes of the plain branch-and-bound procedure, however, we can conclude that adding cutting planes on top of heuristics and variable fixing methods does not improve, and in most cases degrades the overall performance.

3.2.5 Computational results: Comparison with state-of-the-art methods

We also compared our branch-and-cut approach with other approaches from the literature on widely-used problem instances, namely the Reference Point Method of Pugliese and Guerriero (2013) and the Pulse Algorithm of Lozano and Medaglia (2013).

Our experiments suggest that LP-based methods are competitive with other solution approaches to solve RCSPP. We found that the vast majority of the instances can be solved optimally using only the well-known DB-preprocessing procedure. On harder instances, preprocessing techniques (both prior to forming the integer linear programming formulation, and in the course of branch-and-bound) play a key role in reducing the computation times.

4 Position-based scheduling of chains on a single machine

4.1 Problem definition and formulation

We considered a scheduling problem, where a set \( J = \{J_1, \ldots, J_n\} \) of unit-time jobs has to be sequenced on a single machine without any idle times between the jobs. Preemption of processing is not allowed. The processing cost of a job is determined by the position in the sequence, that is, for each job \( J_i \) and each position \( j \) there is an associated weight \( w_{ij} \), and one has to determine a sequence \( S \) of jobs which minimizes the total weight incurred by the positions of the jobs, i.e., \( \sum_{j=1}^{n} w_{ij} \sigma_j \). Using the notation of Graham et al. (1979), we denote the problem as \( 1 \mid p_{j} = 1 \mid \sum w_{ij} \sigma_j \). In addition, the ordering of the jobs must satisfy the given chain-precedence constraints, that is, the jobs are partitioned into nonempty chains, where each chain is an ordered tuple \( (I_{i1}, \ldots, I_{i\ell}) \) specifying that job \( I_{ip} \) must be processed before job \( I_{iq} \) for each \( 1 \leq p < q \leq \ell \). This problem is denoted with \( 1 \mid \text{chains}, p_{j} = 1 \mid \sum w_{ij} \sigma_j \). In case of two-chains, there are an even number of jobs, and each chain consists of exactly two jobs. This problem is denoted with \( 1 \mid 2\text{-chains}, p_{j} = 1 \mid \sum w_{ij} \sigma_j \).

We omit the straightforward integer linear programming formulation of the problem, however, we mention that it contains binary variables \( x_{ij} \) indicating whether job \( J_i \) is assigned to position \( j \). We denote with \( S_{n}^{\text{chain}} \) the set of incidence vectors \( x \in \{0,1\}^{n \times n} \) corresponding to feasible job-position assignments, and with \( P_{n}^{\text{chain}} := \text{conv}(S_{n}^{\text{chain}}) \) the polytope of these assignments.

4.2 Our contribution

First, we investigated the complexity of the problems described before. Then, we obtained a class of valid inequalities for \( P_{n}^{\text{chain}} \), and we showed that a subclass of these inequalities is
Summary of the PhD thesis

 facet-defining in case of two-chains. We also provided a polynomial time separation procedure for this class. Finally, we made thorough computational experiments in order to test the effectiveness of these inequalities. These results were published in (Horváth and Kis, 2019b).

4.2.1 Complexity results

**Theorem 3 (Horváth and Kis (2019b)).** Problem 1 | 2-chains, \( p_j = 1 \) | \( \sum w_{j,\sigma_j} \) is strongly NP-hard.

**Corollary 1.** Problem 1 | chains, \( p_j = 1 \) | \( \sum w_{j,\sigma_j} \) is strongly NP-hard.

**Corollary 2.** Problem 1 | prec, \( p_j = 1 \) | \( \sum w_{j,\sigma_j} \) is strongly NP-hard.

4.2.2 Valid inequalities for \( P_{n}^{\text{chain}} \)

We obtained valid inequalities for \( P_{n}^{\text{chain}} \) by establishing a connection to the parity polytope (see e.g., (Lancia and Serafini, 2018)) where the so-called parity inequalities constitute the non-trivial facets of that polytope. For this, let \( \{C_1, \ldots, C_m\} \) be the set of chains, and let \( |C_i| \) denote the length (i.e., the number of the contained jobs) of chain \( C_i \). We say that a subset \( S \subseteq \{1, \ldots, m\} \) is an odd-subset (even-subset) if its cardinality \( |S| \) is odd (even).

**Theorem 4 (Horváth and Kis (2019b)).** The following inequalities are valid for \( P_{n}^{\text{chain}} \):

\[
\sum_{i \in S} \left( \sum_{k=1}^{|C_i|} (-1)^{k-1} \sum_{p=1}^{j} x_{i,p} \right) - \sum_{i \notin S} \left( \sum_{k=1}^{|C_i|} (-1)^{k-1} \sum_{p=1}^{j} x_{i,p} \right) \leq |S| - 1
\]

for each even position \( j \) and odd-subset \( S \subseteq \{1, \ldots, m\} \), (2)

and

\[
\sum_{i \in S} \left( \sum_{k=1}^{|C_i|} (-1)^{k-1} \sum_{p=1}^{j} x_{i,p} \right) - \sum_{i \notin S} \left( \sum_{k=1}^{|C_i|} (-1)^{k-1} \sum_{p=1}^{j} x_{i,p} \right) \leq |S| - 1
\]

for each odd position \( j \) and even-subset \( S \subseteq \{1, \ldots, m\} \). (3)

Of course, inequalities (2) and (3) are also valid in case of two-chains, moreover, we showed that some of them are facet-defining. To ease our notation, in this case we assume that there are \( 2n \) jobs and thus \( m = n \) chains with \( C_i = (J_{2i-1}, J_{2i}) \), and we denote with \( P_{2n}^{2\text{-chains}} \) the polytope of incidence vectors \( x \in \{0, 1\}^{2n \times 2n} \) corresponding to feasible job-position assignments.

**Theorem 5 (Horváth and Kis (2019b)).** Let \( S \subseteq \{1, \ldots, n\} \) be an odd-subset with \( 3 \leq |S| < n \), and \( 1 \leq k < n \) such that \( |S| < 2k \) and \( |S| < 2(n - k) \). Then, the following inequalities are facet-defining for \( P_{2n}^{2\text{-chains}} \):

\[
\sum_{i \in S} \sum_{j=1}^{2k} (x_{2i-1,j} - x_{2i,j}) - \sum_{i \notin S} \sum_{j=1}^{2k} (x_{2i-1,j} - x_{2i,j}) \leq |S| - 1.
\] (4)
4.2.3 Computational results

We made computational experiments where the main goal was to examine the effectiveness of our inequalities (2) and (3). Since we proved that some of these inequalities are facet-defining if each chain has length two, our experiments focused on problems $1 \mid 2$-chains, $p_j = 1 \mid \sum w_j x_j$, and $1 \mid \text{chain-length} \in \{1, 2\}, p_j = 1 \mid \sum w_j x_j$, where in the latter case each chain has length at most two. Our solution method was implemented in C++ programming language using CPLEX as a branch-and-cut framework.

Our experiments showed that separating inequalities (2) and (3) can significantly improve an LP-based branch-and-bound procedure if the length of each chain is at most two, that is, methods separating these inequalities considerably outperformed the other ones in all aspects. First, only these methods were able to solve optimally all of our randomly generated instances with the given time limit. Second, for each instance, these methods needed shorter execution time than the default branch-and-bound and branch-and-cut methods of CPLEX. Finally, these methods significantly reduced the number of the explored enumeration tree nodes as well.

5 Multiple-depot vehicle and crew scheduling problem

5.1 Problem definition

A (timetabled) trip is a project for vehicles to carry passengers between two given stations with fixed departure and arrival times. A depot is a facility with homogeneous fleet of vehicles. The aim of the Integrated Vehicle and Crew Scheduling Problem (VCSP) is to solve the following problems simultaneously such that some operational and labor costs are minimized: (i) find a feasible assignment of trips to vehicles, and (ii) find a set of duties for the drivers based on the trip-vehicle assignment, satisfying a wide variety of regulations. Based on the number of depots we have the Single-Depot Vehicle and Crew Scheduling Problem (SDVCSP), or the Multiple-Depot Vehicle and Crew Scheduling Problem (MDVCSP).

5.2 Our contribution

We presented a novel problem formulation for MDVCSP, where we combined the advantages of the existing modeling approaches, and we developed an exact branch-and-price procedure to solve this formulation. To our best knowledge, the only paper proposing an exact method for MDVCSP is that of Mesquita et al. (2009), where a variant of the problem is studied, but in this variant some of the common assumptions we and other authors make on feasible crew schedules are neglected. We also presented our computational results where our method was compared with other well-known solution approaches. These results were published in (Horváth and Kis, 2019a).

---

2IBM ILOG CPLEX Optimization Studio, version 12.6.3.0
5.2.1 Problem formulation

Without going into details we provide our formulation for MDVCSP which was obtained from that of Steinzen et al. (2010):

\[
\begin{align*}
\text{minimize} & \quad \sum_{d \in D} \sum_{(i,j) \in \bar{A}} c_{ij}^d y_{ij}^d + \sum_{d \in D} \sum_{k \in K^d} f_k^d x_k^d \\
\text{subject to} & \quad \sum_{d \in D} \sum_{k \in K^d} x_k^d = 1 \quad \text{for all } t \in T \\
& \quad \sum_{k \in K^d \setminus (i)} x_k^d - \sum_{k \in K^d \setminus (i)} x_k^d = 0 \quad \text{for all } d \in D, \; i \in V^d \setminus \bar{V}^d \\
& \quad \sum_{(i,j) \in \bar{A}} y_{ij}^d + \sum_{k \in K^d \setminus (i)} x_k^d - \sum_{(j,i) \in \bar{A}} y_{ij}^d - \sum_{k \in K^d \setminus (i)} x_k^d = 0 \quad \text{for all } d \in D, \; i \in \bar{V}^d \\
& \quad 0 \leq y_{ij}^d, \; y_{ij}^d \in \mathbb{Z} \quad \text{for all } d \in D, \; (i,j) \in \bar{A}^d \\
& \quad x_k^d \in \{0, 1\} \quad \text{for all } d \in D, \; k \in K^d.
\end{align*}
\]

Briefly stated, for each depot \(d \in D\) and the set \(T\) of trips we created a time-space network (Kliewer et al., 2006), then we used two types of variables. First, we associated a flow variable \(y^d\) with some arcs indicating the number of vehicles crossing on them. Second, we assigned a binary duty variable \(x_k^d\) to each duty \(k \in K^d\) (i.e., feasible duties that can be operated from depot \(d\)) indicating whether the duty is selected or not. Note that almost all of the feasibility constraints on the duties are crammed into the definition of the set \(K^d\), therefore its cardinality and thus the number of duty variables can be vast.

5.2.2 Solution procedure

We developed an exact branch-and-price solution approach for formulation (5)–(10), that is, we created an initial system consisting of all the flow variables and a subset of duty variables only, which is gradually augmented by the missing variables, if needed. Again, without going into details our branch-and-price procedure includes (i) an effective pricing procedure to add the missing duty variables to the initial system, (ii) some problem-tailored branching strategies, and (iii) a simple primal heuristics for searching feasible solutions to the problem.

5.2.3 Computational results

We made computational experiments on widely-used instances, where the main goals were (i) to evaluate our integrated method, and (ii) to compare our method with other solution approaches from the literature.

Our experiments showed that with limited computational resources (computation time and single CPU thread), near optimal schedules can be found for problems with 80–100 trips and 4 depots. On these instances our integrated method found solutions with fewer vehicles plus drivers than the heuristic solution approach of Steinzen et al. (2010). We also made
experiments on bigger instances, however, we were not able to solve any of these instances neither optimally nor with gap limit, and our best solutions were worse than that of Steinzen et al. (2010). In order to increase the problem size, one possible direction is to exploit multiple CPU cores/threads, but for that, one needs a parallel branch-and-price solver.

Acknowledgment

I have been working at SZTAKI (formerly MTA SZTAKI) since the beginning of my PhD studies. During this time, my research was supported by the GINOP-2.3.2-15-2016-00002 grant on an ”Industry 4.0 research and innovation center of excellence”.

The thesis is based on the following papers:


References


