MTA SZTAKI
Hungarian Academy of Sciences
Institute for Computer Science and Control

A Branch-and-Price method for the Multiple-depot Vehicle and Crew Scheduling Problem

SCIP Workshop 2018, Aachen

Markó Horváth · Tamás Kis

Institute for Computer Science and Control
Hungarian Academy of Sciences

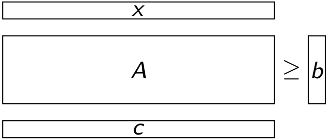
Outline

1. Introduction – Mathematical background (if needed)
    - Column Generation, Branch-and-Price
2. Introduction – Integrated Vehicle and Crew Scheduling Problem (MDVCSP)
    - Vehicle Scheduling, Crew Scheduling
3. A Branch-and-Price method for the MDVCSP
    - Modelling approach
    - Solution approach (branching strategies, pricing variables, etc.)
4. Computational experiments

Introduction

Introduction – Mathematical background

Column Generation approach for Linear Programs

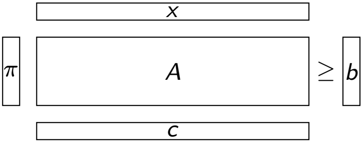


Master Problem

$$\min \{cx : Ax \geq b, x \geq 0\}$$

Introduction – Mathematical background

Column Generation approach for Linear Programs

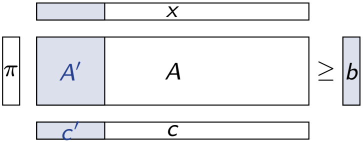


Master Problem

$$\min\{cx : Ax \geq b, x \geq 0\} = \max\{\pi b : \pi A \leq c, \pi \geq 0\}$$

Introduction – Mathematical background

Column Generation approach for Linear Programs



Master Problem

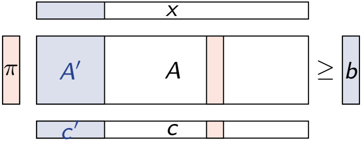
$$\min\{cx : Ax \geq b, x \geq 0\} = \max\{\pi b : \pi A \leq c, \pi \geq 0\}$$

Restricted Master Problem (RMP)

$$\min\{c'x : A'x \geq b, x \geq 0\}$$

Introduction – Mathematical background

Column Generation approach for Linear Programs



Master Problem

$$\min\{cx : Ax \geq b, x \geq 0\} = \max\{\pi b : \pi A \leq c, \pi \geq 0\}$$

Restricted Master Problem (RMP)

$$\min\{c'x : A'x \geq b, x \geq 0\}$$

Variable Pricing / Column Generation

iteratively add new variables (i.e., columns) with negative reduced cost (that is, $\bar{c} = c - \pi A$) to the problem

Introduction – Mathematical background

Branch-and-Price method for Integer Linear Programs

Master Problem

$$\min \left\{ cx : Ax \geq b, x \geq 0, x \in \mathbb{Z}^d \right\}$$

Branch-and-Price $\approx$ Branch-and-Bound + Column Generation

at each node of the search tree, columns may be added to the LP relaxation.

Introduction – Mathematical background

Branch-and-Price method for Integer Linear Programs

Master Problem

$$\min \left\{ cx : Ax \geq b, x \geq 0, x \in \mathbb{Z}^d \right\}$$

Branch-and-Price $\approx$ Branch-and-Bound $+$ Column Generation

at each node of the search tree, columns may be added to the LP relaxation.

Introduction – Mathematical background

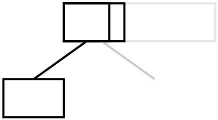
Branch-and-Price method for Integer Linear Programs

Master Problem

$$\min \left\{ cx : Ax \geq b, x \geq 0, x \in \mathbb{Z}^d \right\}$$

Branch-and-Price ≈ Branch-and-Bound + Column Generation

at each node of the search tree, columns may be added to the LP relaxation.



Introduction – Mathematical background

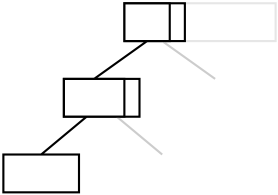
Branch-and-Price method for Integer Linear Programs

Master Problem

$$\min \left\{ cx : Ax \geq b, x \geq 0, x \in \mathbb{Z}^d \right\}$$

Branch-and-Price ≈ Branch-and-Bound + Column Generation

at each node of the search tree, columns may be added to the LP relaxation.



Introduction – Mathematical background

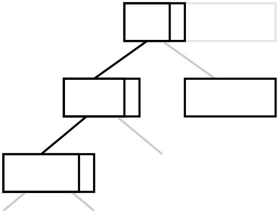
Branch-and-Price method for Integer Linear Programs

Master Problem

$$\min \left\{ cx : Ax \geq b, x \geq 0, x \in \mathbb{Z}^d \right\}$$

Branch-and-Price $\approx$ Branch-and-Bound + Column Generation

at each node of the search tree, columns may be added to the LP relaxation.



Introduction – Vehicle and Crew Scheduling Problem

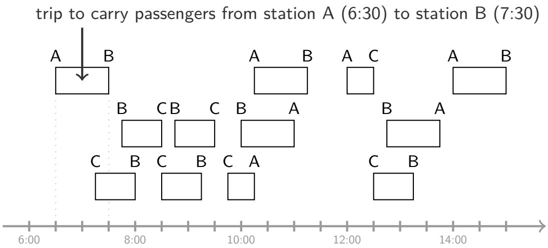
Vehicle Scheduling

Vehicle Scheduling Problem (VSP):

- **Given:**
    - a set of timetabled trips
    - a fleet of vehicles divided into depots
- **Goal:** find an assignment of trips to vehicles such that
    - each trip is assigned exactly once
    - each vehicle performs a feasible sequence of trips
    - each sequence starts and ends at the same depot
    - asset and operational costs are minimized
- Typically modelled as a multicommodity-flow problem

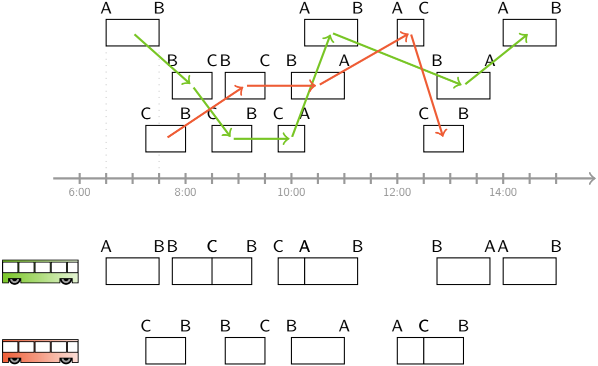
Introduction – Vehicle and Crew Scheduling Problem

Vehicle Scheduling



Introduction – Vehicle and Crew Scheduling Problem

Vehicle Scheduling



Crew Scheduling

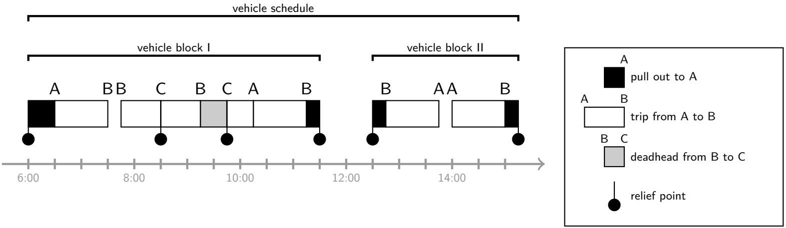


Figure: route of a vehicle and some driver activites

Crew Scheduling

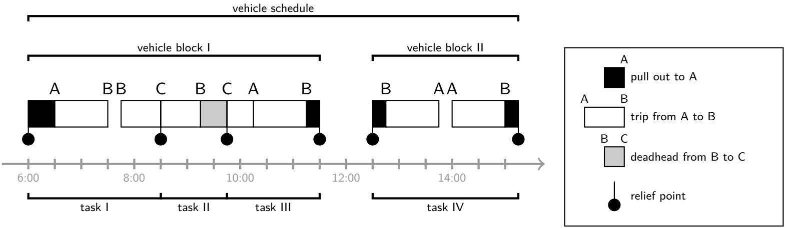


Figure: route of a vehicle and some driver activites

Introduction – Vehicle and Crew Scheduling Problem

Crew Scheduling

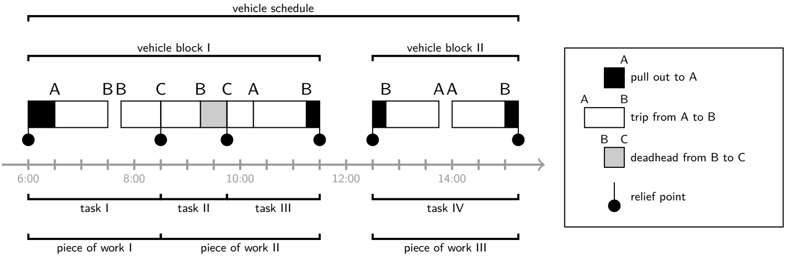


Figure: route of a vehicle and some driver activites

Introduction – Vehicle and Crew Scheduling Problem

Crew Scheduling

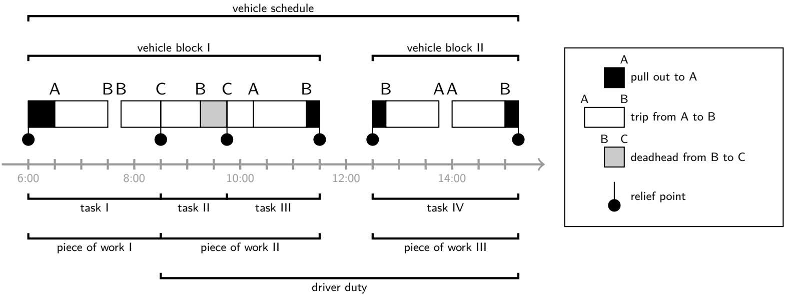


Figure: route of a vehicle and some driver activites

Crew Scheduling Problem (CSP):

- **Given:**
  - a set of tasks
- **Goal:** find a set of driver duties such that
  - each task is covered by a duty that can be performed by a single driver
  - each duty satisfies a wide variety of federal laws, safety regulations, and (collective) in-house agreements
  - labor costs are minimized
- Typically modelled as a set-partitioning (-covering) problem

Introduction – Vehicle and Crew Scheduling Problem

Sequential approach:

1. VSP: trips $\rightarrow$ vehicle schedules (and tasks)
2. CSP: vehicle schedules (and tasks) $\rightarrow$ driver duties

- seriously criticized because in the mass transit case crew costs mostly dominate vehicle operating costs

Integrated approach:

- VCSP: trips $\rightarrow$ vehicle schedules, driver duties (i.e., simultaneously)

Introduction – Vehicle and Crew Scheduling Problem

Integrated approach

The set of tasks is not fixed, hence the number of potential duties can be *vast* even for small-sized problems.

Table: example for an instance with 80 trips

depot	#pieces of work	#duties
1	340763	$\sim$ 7 billion
2	344121	$\sim$ 7.5 billion
3	151244	$\sim$ 1.5 billion
4	437611	$\sim$ 10 billion

A Branch-and-Price method for the Multiple-Depot Integrated Vehicle and Crew Scheduling Problem

Problem definition – Assumptions

1. *Each vehicle is assigned to a depot* where its daily schedule starts end ends. Each depot is unlimited in capacity, that is, it can store an unlimited number of vehicles.
2. A vehicle returns to its depot if the idle time between two consecutive trips is long enough to perform a round trip to the depot.
3. *Each driver is assigned to a depot* and may only conduct tasks on vehicles from this particular depot. However, a duty does not necessarily start and end in this depot. It may have a minimum and maximum duration.
4. *A driver is required to be present if a vehicle is outside of a depot*, while no driver is needed when the vehicle is parked in the depot.
5. *Drivers may only change their vehicle during a break*, i.e., between two pieces of work.

Problem definition – Assumptions

6. A piece of work is only restricted by its duration.
7. A duty consists of one or two pieces of work. (...)

Table: properties of duty types

	Tripper		Early		Day		Late		Split	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
start time					8:00		13:15			
end time				16:30		18:14				19:30
piece length	0:30	5:00	0:30	5:00	0:30	5:00	0:30	5:00	0:30	5:00
break length	-	-	0:45		0:45		0:45		1:30	
spread time				9:45		9:45		9:45		12:00
working time				9:00		9:00		9:00		9:00

Modelling approach

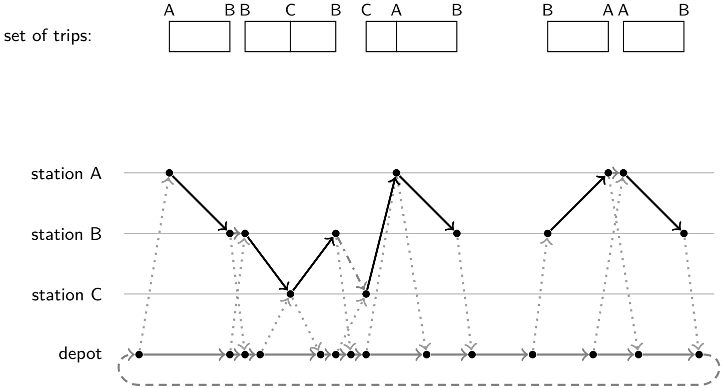


Figure: time-space network for a single depot d

Modelling approach

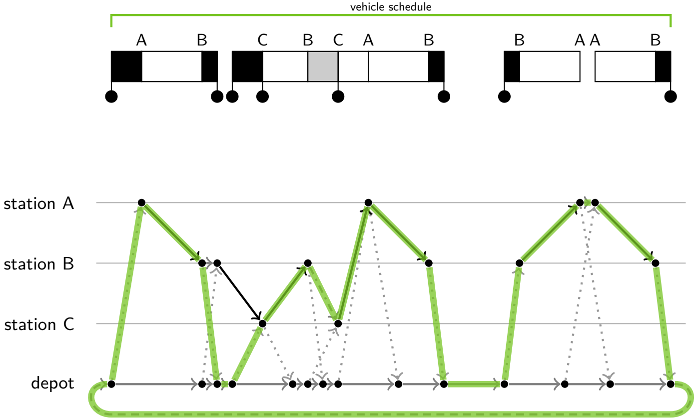


Figure: time-space network for a single depot d

Modelling approach

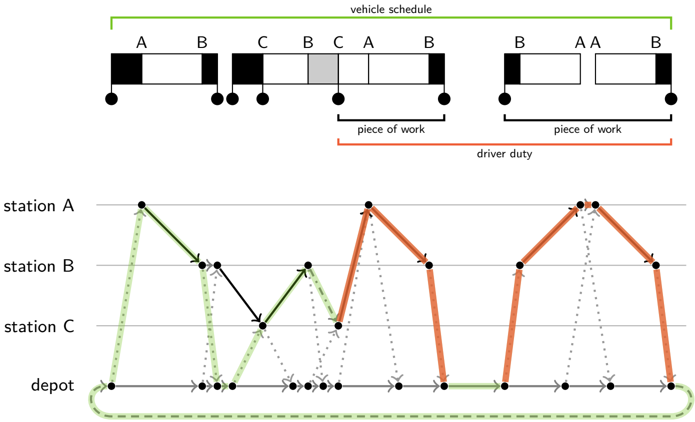


Figure: time-space network for a single depot d

Modelling approach

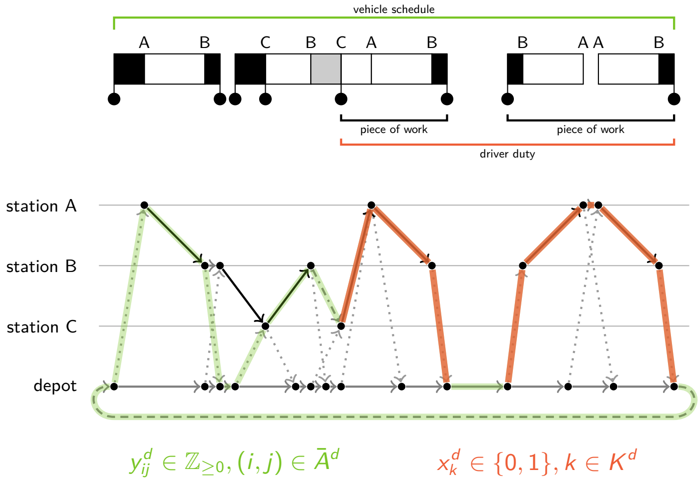


Figure: time-space network for a single depot d

Modelling approach

$$\min \sum_{d\in\mathcal{D}} \sum_{ij\in\bar{A}^d} c_{ij}^d y_{ij}^d + \sum_{d\in\mathcal{D}} \sum_{k\in K^d} \tilde{f}_k^d x_k^d \tag{1}$$

$$\sum_{d\in\mathcal{D}} \sum_{k\in K^d(t)} x_k^d = 1 \qquad \forall\ t \in \mathcal{T} \tag{2}$$

$$\sum_{k\in K_+^d(i)} x_k^d - \sum_{k\in K_-^d(i)} x_k^d = 0 \qquad \forall d \in \mathcal{D}, \forall i \in V^d \setminus \bar{V}^d \tag{3}$$

$$\sum_{ij\in\bar{A}^d} y_{ij}^d + \sum_{k\in K_+^d(i)} x_k^d - \sum_{ji\in\bar{A}^d} y_{ji}^d - \sum_{k\in K_-^d(i)} x_k^d = 0 \qquad \forall\ d \in \mathcal{D}, \forall\ i \in \bar{V}^d \tag{4}$$

$$0 \leq y_{ij}^d, y_{ij}^d \in \mathbb{Z} \qquad \forall\ d \in \mathcal{D}, \forall\ ij \in \bar{A}^d \tag{5}$$

$$x_d^k \in \{0,1\} \qquad \forall\ d \in \mathcal{D}, \forall\ k \in K^d \tag{6}$$

Modelling approach

$$\min \sum_{d\in\mathcal{D}} \sum_{ij\in\bar{A}^d} c_{ij}^d y_{ij}^d + \sum_{d\in\mathcal{D}} \sum_{k\in K^d} \tilde{f}_k^d x_k^d \tag{1}$$

$$\sum_{d\in\mathcal{D}} \sum_{k\in K^d(t)} x_k^d = 1 \qquad \forall\ t \in \mathcal{T} \tag{2}$$

$$\sum_{k\in K_+^d(i)} x_k^d - \sum_{k\in K_-^d(i)} x_k^d = 0 \qquad \forall d \in \mathcal{D}, \forall i \in V^d \setminus \bar{V}^d \tag{3}$$

$$\sum_{ij\in\bar{A}^d} y_{ij}^d + \sum_{k\in K_+^d(i)} x_k^d - \sum_{ji\in\bar{A}^d} y_{ji}^d - \sum_{k\in K_-^d(i)} x_k^d = 0 \qquad \forall\ d \in \mathcal{D}, \forall\ i \in \bar{V}^d \tag{4}$$

$$0 \leq y_{ij}^d, y_{ij}^d \in \mathbb{Z} \qquad \forall\ d \in \mathcal{D}, \forall\ ij \in \bar{A}^d \tag{5}$$

$$x_d^k \in \{0,1\} \qquad \forall\ d \in \mathcal{D}, \forall\ k \in K^d \tag{6}$$

SCIP – Worth to mention!

Variables y_{ij} are implicit integer! (`SCIP_VARTYPE_IMPLINT`)

Solution approach

Branch-and-Price

- Straightforward idea
- How to ...
  - ... create initial Restricted Master Problem?
  - ... price out new variables (i.e., driver duties)?
  - ... perform branch on variables? (Note that branching decisions must be taken into consideration during variable pricing)

Solution approach

Branching strategies

- Default 0-1 branching is weak
- Branching strategy 1: *Assign trips to depots*
    - easy to handle in the pricing problem
- Branching strategy 2: *SPP-based branching*
    - based on the Ryan-Foster branching scheme (see set-partitioning constraints (2))
    - very inconvenient to handle in the pricing problem

Solution approach

Branching strategies – Assign trips to depots

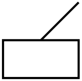
Let $(\bar{x}, \bar{y})$ be a fractional solution to the relaxation of the corresponding RMP

Assign trips to depots:

- Some trips may be committed to multiple depots in the LP-solution
- Trip t, depot d (such that $0 < \sum_{k \in K_t^d} \bar{x}_k^d < 1$)
- Partitioning (two branches)
    1. binding branch: require to cover trip t by a duty from depot d
    2. banning branch: forbid to cover trip t by a duty from depot d
- Splitting (several branches)
    1. jth (binding) branch: require to cover trip t by a duty from depot d_j

Solution approach

Branching strategies – Assign trips to depots

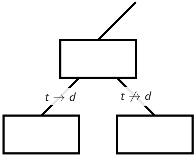


Branching rule (scip::ObjBranchrule, scip_execlp)

- performed if the LP solution of the current problem is fractional
- determine candidate (t, d)

Solution approach

Branching strategies – Assign trips to depots

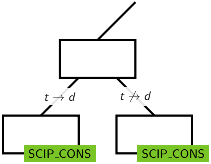


Branching rule (scip::ObjBranchrule, scip_execlp)

- performed if the LP solution of the current problem is fractional
- determine candidate (t, d)
- create child nodes

Solution approach

Branching strategies – Assign trips to depots

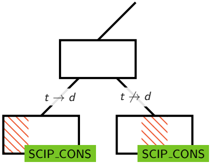


Branching rule (scip::ObjBranchrule, scip_execlp)

- performed if the LP solution of the current problem is fractional
- determine candidate (t, d)
- create child nodes
- create constraints for child nodes

Solution approach

Branching strategies – Assign trips to depots



Branching rule (scip::ObjBranchrule, scip_execlp)

- performed if the LP solution of the current problem is fractional
- determine candidate (t, d)
- create child nodes
- create constraints for child nodes

Constraint handler (scip::ObjConshdlr, scip_prop)

- propagation, i.e, node preprocessing

Solution approach

Pricing variables

$$\min \sum_{d\in\mathcal{D}} \sum_{ij\in\bar{A}^d} c_{ij}^d y_{ij}^d + \sum_{d\in\mathcal{D}} \sum_{k\in K^d} \tilde{f}_k^d x_k^d$$

$$\sum_{d\in\mathcal{D}} \sum_{k\in K^d(t)} x_k^d = 1 \quad \lambda_t \quad \forall\, t \in \mathcal{T}$$

$$\sum_{k\in K_+^d(i)} x_k^d - \sum_{k\in K_-^d(i)} x_k^d = 0 \quad \mu_i^d \quad \forall\, d \in \mathcal{D}, \forall\, i \in V^d \setminus \bar{V}^d$$

$$\sum_{ij\in\bar{A}^d} y_{ij}^d + \sum_{k\in K_+^d(i)} x_k^d - \sum_{ji\in\bar{A}^d} y_{ji}^d - \sum_{k\in K_-^d(i)} x_k^d = 0 \quad \mu_i^d \quad \forall\, d \in \mathcal{D}, \forall\, i \in \bar{V}^d$$

$$0 \le y_{ij}^d, y_{ij}^d \in \mathbb{Z} \qquad \forall\, d \in \mathcal{D}, \forall\, ij \in \bar{A}^d$$

$$x_d^k \in \{0,1\} \qquad \forall\, d \in \mathcal{D}, \forall\, k \in K^d$$

Solution approach

Pricing variables

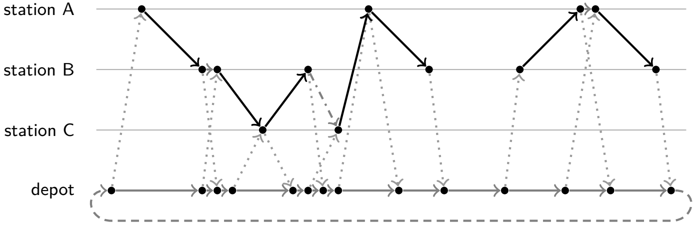


Figure: piece-of-work generation network for a single depot

Solution approach

Pricing variables

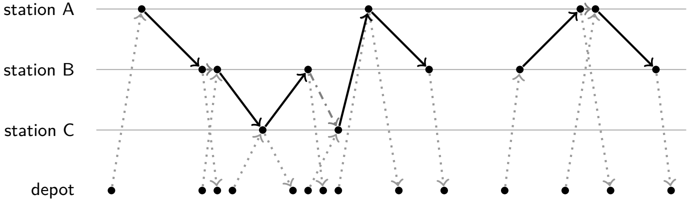


Figure: piece-of-work generation network for a single depot

Solution approach

Pricing variables

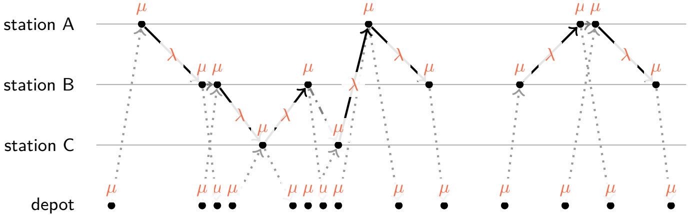


Figure: piece-of-work generation network for a single depot

Solution approach

Pricing variables

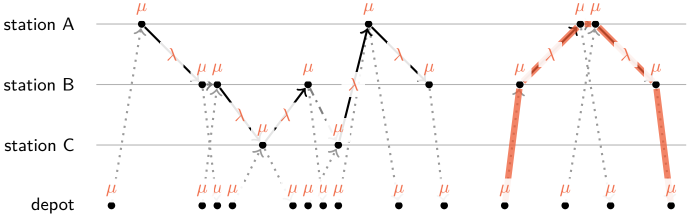


Figure: piece-of-work generation network for a single depot

Solution approach

Pricing variables

Generation of driver duties:

1. *Generation of pieces of work*
   - find shortest path (according to reduced costs) for all source-destination pair
2. *Generation of duties*
   - duties consisting of 1 piece of work: simple enumeration procedure
   - duties consisting of 2 pieces of work: smart pairing procedure

Solution approach

Pricing variables

Generation of driver duties:

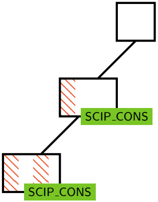
1. *Generation of pieces of work*
   - find shortest path (according to reduced costs) for all source-destination pair
2. *Generation of duties*
   - duties consisting of 1 piece of work: simple enumeration procedure
   - duties consisting of 2 pieces of work: smart pairing procedure

Branching rules must be taken into consideration during duty generation!

- *Assign trips to depots*
  - binding branch ($t \to d$): remove trip t from the piece generation network of depot $d' \neq d$
  - banning branch ($t \not\to d$): remove trip t from the piece generation network of depot d

Solution approach

Pricing variables

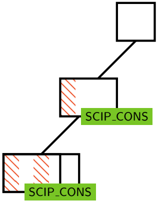


Pricer (scip::ObjPricer, scip_redcost/scip_farkas)

- called inside the price-and-cut loop of the subproblem solving process if the current LP relaxation is feasible/infeasible
- scip_farkas: similar pricing procedure (Farkas-multipliers, zero objective function)

Solution approach

Pricing variables



Pricer (scip::ObjPricer, scip_redcost/scip_farkas)

- called inside the price-and-cut loop of the subproblem solving process if the current LP relaxation is feasible/infeasible
- scip_farkas: similar pricing procedure (Farkas-multipliers, zero objective function)
- create and add priced variables

Solution approach

Initial Restricted Master Problem

- Contains all of the flow variables (y_{ij}^d)
- Contains a set of duty variables (x_k^d)
  1. use fictive columns penalized by a high cost
  2. start with an empty set of duty variables (Farkas pricing)
  3. obtain a feasible solution for the MDVCSP by using a sequential approach

Computational experiments

Computational experiments

Implementation details

Test environment and implementation

- C++ programming language
- Branch-and-Price framework: SCIP Optimization Suite (version 3.1.1)
- Graph algorithms: LEMON C++ library (version 1.3.1)

Instances and problem parameters

- Randomly generated problem instances of Dennis Huisman
- 80A: 10 instances (4 depots, 4 stations)
- 100A: 10 instances (4 depots, 5 stations)

Running details

- Gap limit: 0.5%; time limit: $20 \times$ number of trips

Computational experiments

Evaluation of the branching rules

- Goal: select the most appropriate branching rule for the problem
- Tested on $10 \times 10 = 100$ instances

Table: Summary of experiments on branching rules

Problem	Rule	Status[a]			Bound			Best solution			Time
		O	G	T	Lower	Upper	Gap	v	d	v+d	
80A	Partitioning	9	50	41	34 772.2	35 410.8	1.8%	9.5	18.5	28.0	755.6
	Splitting	8	49	43	34 769.4	35 481.4	2.0%	9.5	18.6	28.0	777.5
100A	Partitioning	6	49	45	41 624.0	42 464.2	2.0%	11.4	22.1	33.5	1136.6
	Splitting	5	43	52	41 621.7	42 533.2	2.2%	11.4	22.1	33.6	1205.4

[a] : number of instances that solved to (O)ptimality or where (G)ap limit or (T)ime limit was reached

Computational experiments

Evaluation of the integrated method

We compared four methods:

- **Seq.** : sequential approach (used to obtain the initial RMP)
- **Int. (first)**: integrated approach; interrupted right after a feasible solution was found
- **Int. (timelimit)**: integrated approach; interrupted only when the gap/time limit was reached
- **Int. [Steinzen et al., 2010]**: integrated approach of [Steinzen et al., 2010]

Note that our experiments were performed on a workstation with 4GB RAM, and XEON X5650 CPU of 2.67 GHz, and under Linux operating system, while the experiments of Steinzen et al. [2010] were performed on a Dell OptiPlex GX620 personal computer with an Intel Pentium IV 3.4 GHz processor and 2 GB of main memory under Windows XP.

Computational experiments

Evaluation of the integrated method

Table: Comparing sequential and integrated methods

Problem	Method	v	d	v+d	Cost	Time
80A	Seq.	9.2	24.3	33.5	40 588.0	1.2
	Int. (first)	9.6	18.6	28.2	35 668.5	4.1
	Int. (timelimit)	9.5	18.5	28.0	35 456.9	914.6
	Int. [Steinzen et al., 2010]	9.2	19.1	28.2		235.0
100A	Seq.	11.0	28.2	39.2	47 792.7	1.6
	Int. (first)	11.4	22.0	33.4	42 428.5	31.8
	Int. (timelimit)	11.4	21.9	33.3	42 287.4	1007.0
	Int. [Steinzen et al., 2010]	11.0	22.7	33.7		369.0

Thank you for your attention!

Horváth, M., & Kis, T. (2017). *Computing strong lower and upper bounds for the integrated multiple-depot vehicle and crew scheduling problem with branch-and-price*. Central European Journal of Operations Research, 1-29.

✉ marko.horvath@sztaki.mta.hu

References

Ingmar Steinzen, Vitali Gintner, Leena Suhl, and Natalia Kliewer. A time-space network approach for the integrated vehicle-and crew-scheduling problem with multiple depots. *Transportation Science*, 44(3):367–382, 2010.