1D Vehicle Scheduling with Conflicts

Torsten J. Gellert Felix G. König

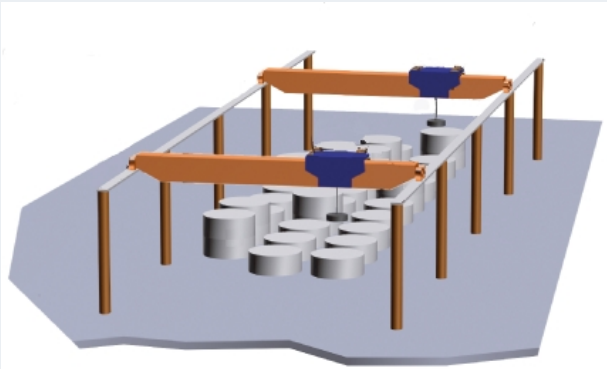
Departement of Mathematics
Berlin Institute of Technology

ALENEX11
January 22, 2011

1. Introduction
   - Motivation
   - Problem Formulation
   - Hardness
2. Algorithms
   - Insertion with Chain-Constraint
   - Clustering
   - Disjoint Domains
3. Computational Study
   - Worst Case Performance
   - Performance on Datasets
   - Open Questions

Example: rail-mounted cranes in steel production

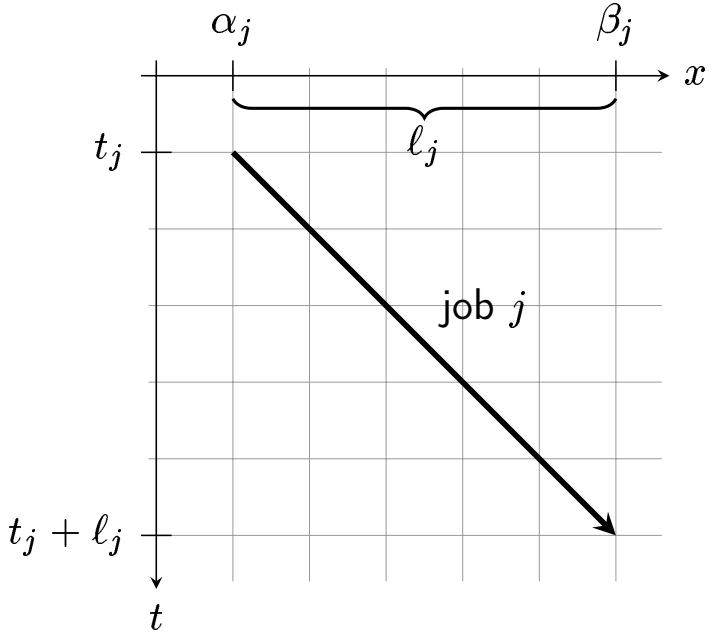




1-dimensional transport tasks with conflicts:

- gantry cranes for loading and unloading container ships
- forklifts in warehouses
- galvanization facilities

Problem Formulation



- $j = (\alpha_j, \beta_j)$
- $\alpha_j :=$ origin of job j
- $\beta_j :=$ destination of job j
- $\ell_j :=$ length of j
- $t_j :=$ start time of j

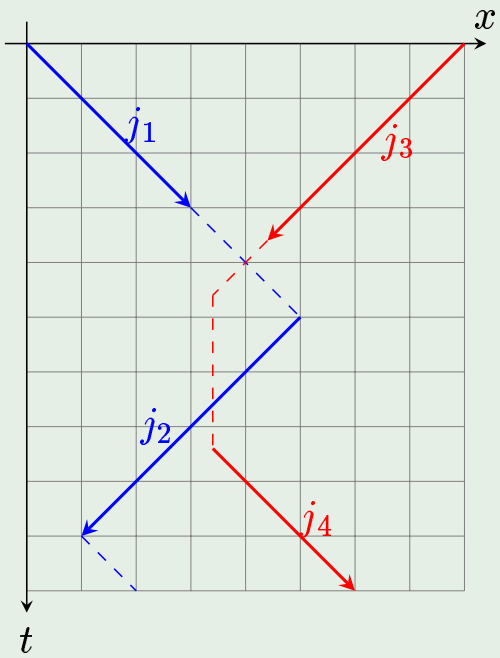
Definition (1D Vehicle Scheduling(1D-VS))

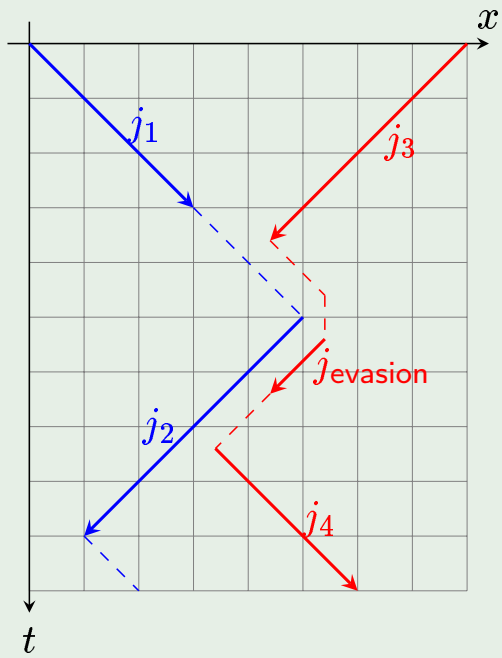
Instance: family of n transport requests J,
k vehicles with unit-speed, initial positions p_i for all vehicles
Solution: partition of J into subfamilies $J_1, \ldots, J_k$,
start times t_j for all jobs s.t. no collision occur
Goal: minimize the makespan i.e., $\min \max_{j \in J} t_j + \ell_j$

Tours: vehicles drive immediately to jobs and wait for the start time

$\rightsquigarrow$ evasion jobs are needed to make room for another vehicles

An additional evasion job removes the collision

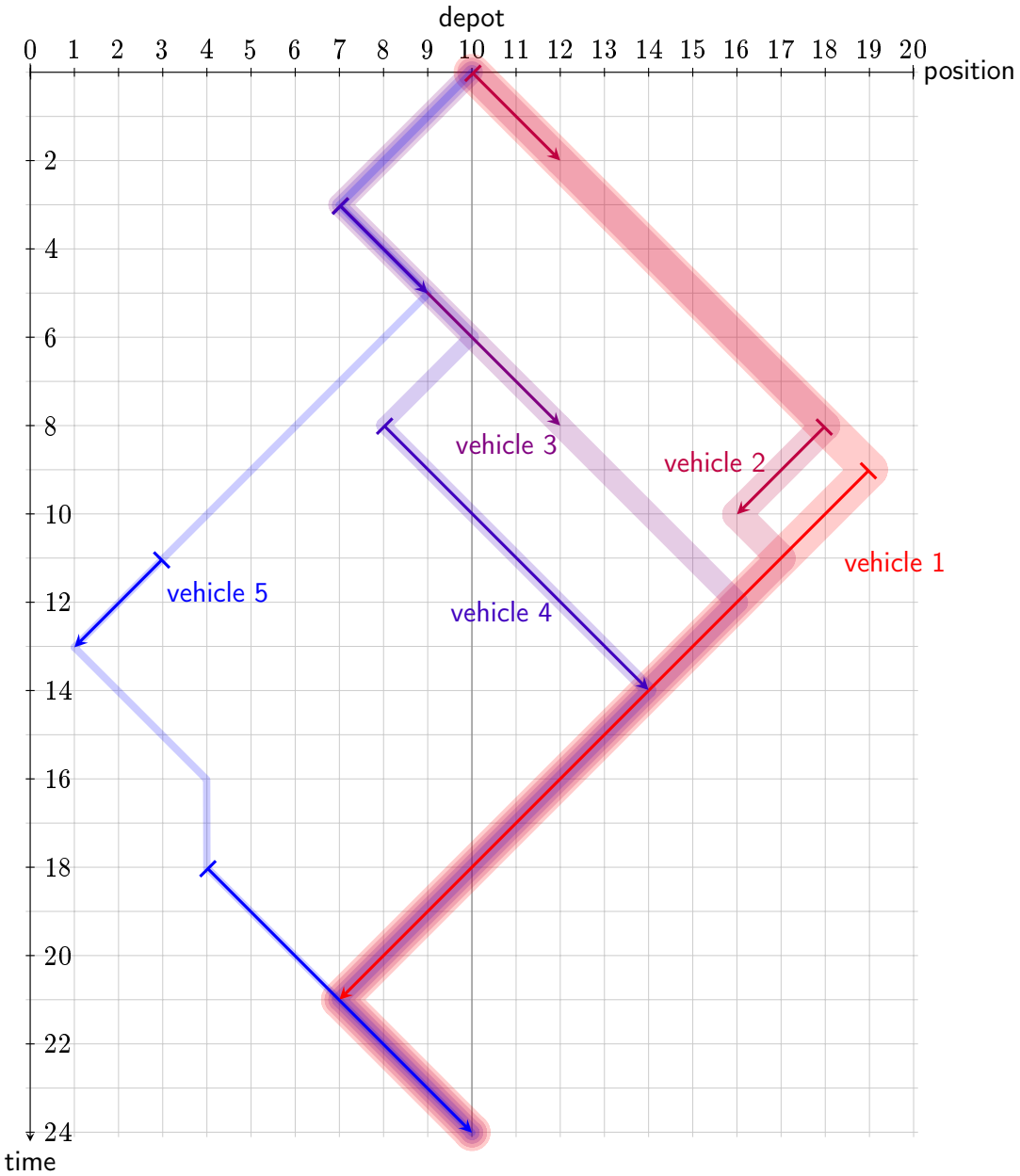




Remark

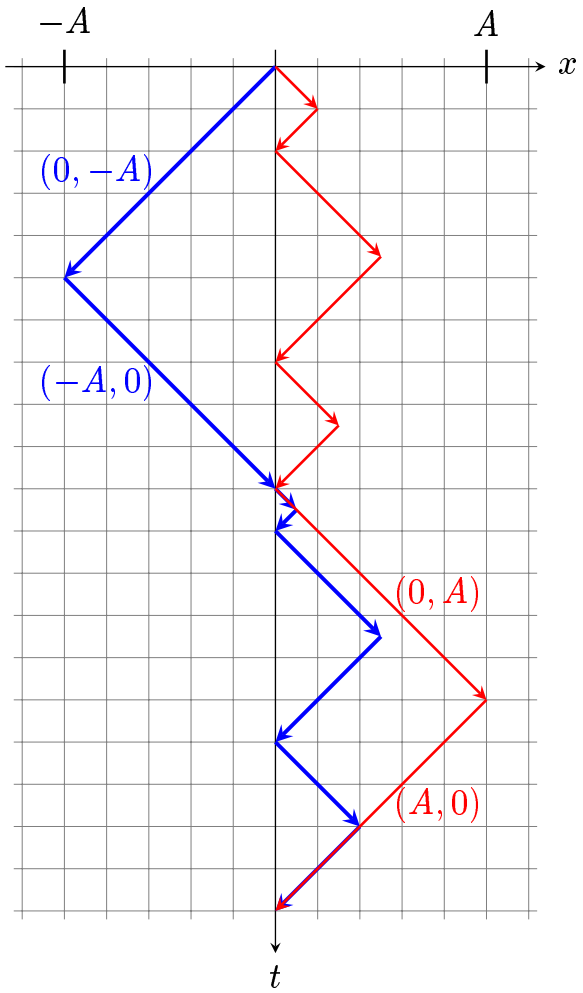
Vehicles are allowed to "touch" each other, only crossing is forbidden

An example solution for an 1D-VS-instance



NP-hardness: Reduction from Partition

Partition instance: $\mathcal{A} = \{a_1, \ldots, a_n\}$ with $\sum_{i=1,\ldots,n} a_i = A$



Construction of the 1D-VS-instance:

- $k = 2$ vehicles (w.l.o.g.)
- positions $p_1 = p_2 = 0$
- two jobs $(0, a_i)$, $(a_i, 0)$ with length a_i for every number a_i
- four long jobs $(0, A)$, $(A, 0)$,$(0, -A)$, $(-A, 0)$

Important known results:

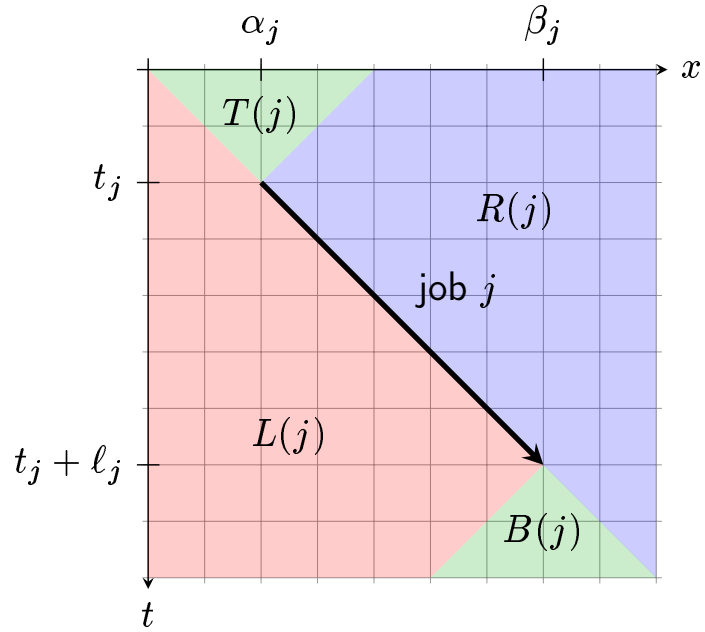
- $k = 1 \rightsquigarrow$ 1D-VS is a special case of the TSP [Gilmore,Gomory,'64] thus polynomially solvable
- 1D-VS with fixed start times polyn. solvable [Heinrichs,Moll,'97] $\rightsquigarrow$ MIP formulation for 1D-VS

Heuristic ideas:

- decomposition into sub-problems with $k = 1$
  solve it and construct a solution for multiple vehicles
- use the construction of Heinrichs & Moll
  fix the start times of the jobs in a certain order

Definition (binary relation $\prec$)

Let $(i, t_i), (j, t_j)$ be some jobs with their start times.
$(j, t_j) \prec (i, t_i) \Leftrightarrow$ vehicle that conducts j has to be on the left of the vehicle that conducts i



Lemma (Heinrichs&Moll)

For jobs with assigned start times (J, T), we have:

- *J can be scheduled with k vehicles $\Leftrightarrow$ chain-graph G is cycle-free and* $\text{diam}(G) \leq k$
- *a schedule can be constructed in $\mathcal{O}(|J|^2)$*

Algorithm:

1. calculate insertion order
   (increasing start times in the single vehicle case)
2. start with an empty chain-graph, insert jobs (p_i, p_i) with start time 0
3. insert all jobs
   - insert job with start time 0
   - update chain-graph and increment start time until chain-graph is valid
4. calculate collision free tours via fixed start times

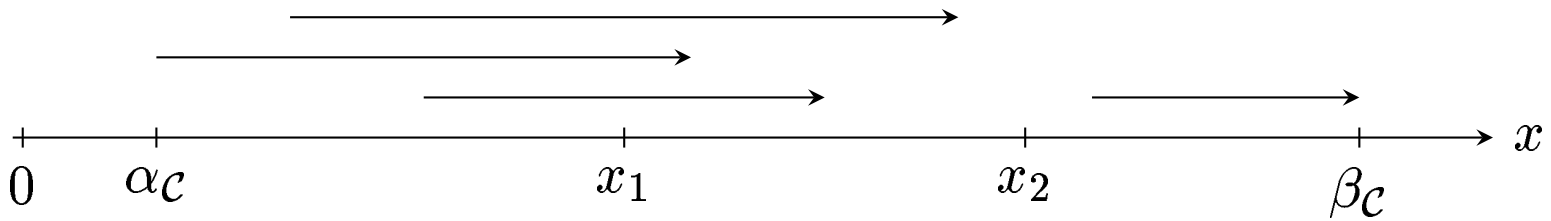
Remark

Different insertion order can be tested $\rightsquigarrow$ localsearch or metaheuristic algorithms

Clustering

Idea: k vehicles form a vehicle with capacity k

$\rightsquigarrow$ find a bunch of jobs and merge them to a single one



Algorithm:

1. $\mathcal{C}^+ := k$-clustering of all jobs with $\alpha_j < \beta_j$
2. $\mathcal{C}^- := k$-clustering of all jobs with $\beta_j < \alpha_j$
3. merge jobs in the same cluster $\rightsquigarrow$ $J' = \{j_{\mathcal{C}_i} | \mathcal{C}_i \in \mathcal{C}^+ \cup \mathcal{C}^-\}$
4. solve 1D-VS instance J' with $k = 1$
5. assign proper start times to each original job

Disjoint Domains:

Idea: split-up space in disjoint domains, assign vehicles to them
$\rightsquigarrow$ no collisions possible

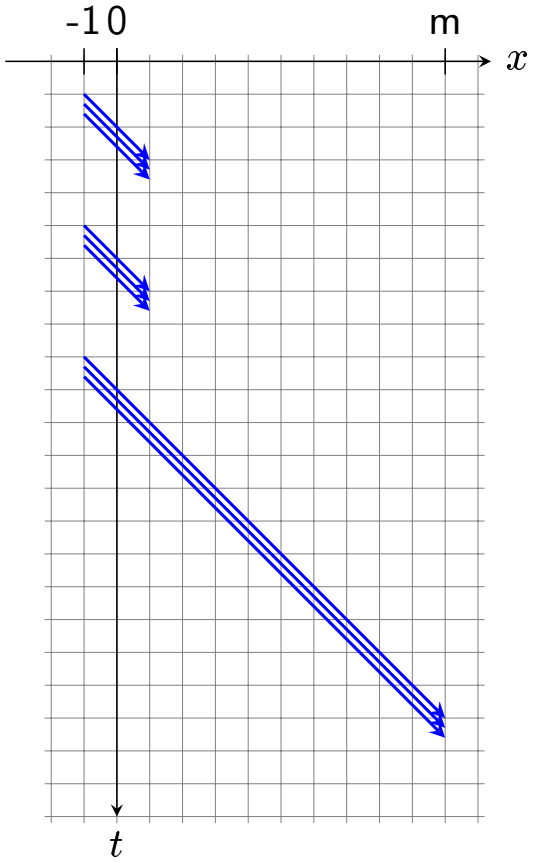
not all jobs lie completely in a domain $\rightsquigarrow$ conduct all jobs inside the domains, merge neighboring domains afterward

Algorithm:

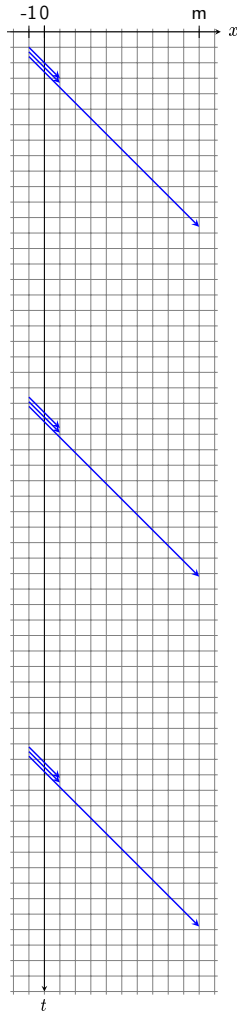
1. find domains $D := \{d_i | 1 \leq i \leq k\}$, assign vehicle i to d_i
2. while some jobs are left
   - solve the problem for each domain with a single vehicle
   - remove the scheduled jobs
   - merge domains: $D := \{d_{2i} \cup d_{2i+1} | 1 \leq i \leq \left\lfloor \frac{|D|}{2} \right\rfloor\}$

Worst-Case Instance:

- k long jobs $(-1, m)$
- $(k-1) \cdot k$ short jobs $\approx (-1, 1)$



makespan $\approx m$



makespan
$\approx (2k-1) \cdot m$

Makespan comparison for $k = 2$

job length:	$0 - 5\%$	$0 - 15\%$	$0 - 30\%$	$0 - 100\%$
`disjoint`	1.00	1.00	1.05	1.59
`cluster`	1.39	1.15	1.00	1.00
`insert`	1.37	1.20	1.14	1.21
`insert l/s`	1.21	0.98	0.89	0.97

Makespan comparison for $k = 5$

job length:	$0 - 5\%$	$0 - 15\%$	$0 - 30\%$	$0 - 100\%$
`disjoint`	1.00	1.28	2.18	3.50
`cluster`	1.36	1.00	1.00	1.00
`insert`	1.34	1.07	1.19	1.33
`insert l/s`	1.02	0.74	0.86	0.96

average makespan on the testsets

Speedup comparison with $k = 2$

job length (%)	insert	cluster	disjoint
0–5	1.32	1.30	1.81
0–15	1.36	1.41	1.62
0–30	1.33	1.51	1.44
0–50	1.33	1.55	1.17
0–100	1.35	1.63	1.03

Speedup comparison with $k = 5$

job length (%)	insert	cluster	disjoint
0–5	2.14	1.66	2.27
0–15	2.26	2.41	1.88
0–30	2.55	3.02	1.39
0–50	2.57	3.19	1.10
0–100	2.62	3.47	1.00

Open questions

- how to find good lower bounds?
- better insertion sequence for the chain-graph based procedure? the optimal one?
- is the problem harder with conflicts than without?
- strongly NP-hard for a fixed k?

Thank you very much!
Any questions?