Pickup and Delivery Problems

THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

Michael Forbes and Ali Alyasiry

School of Mathematics and Physics

Pickup And Delivery Problem with Time Windows

One-to-One vehicle routing problem

Each request has:

- A single origin and a single destination
- Size and/or weight
- Time windows for pickup and/or delivery

Minimize Cost (e.g. fixed and variable vehicle and driver costs)

Subject to:

- Each customer is visited only once
- Vehicle capacity
- Time windows
- Pairing and precedence (pickup location must precede the delivery location on the same route).

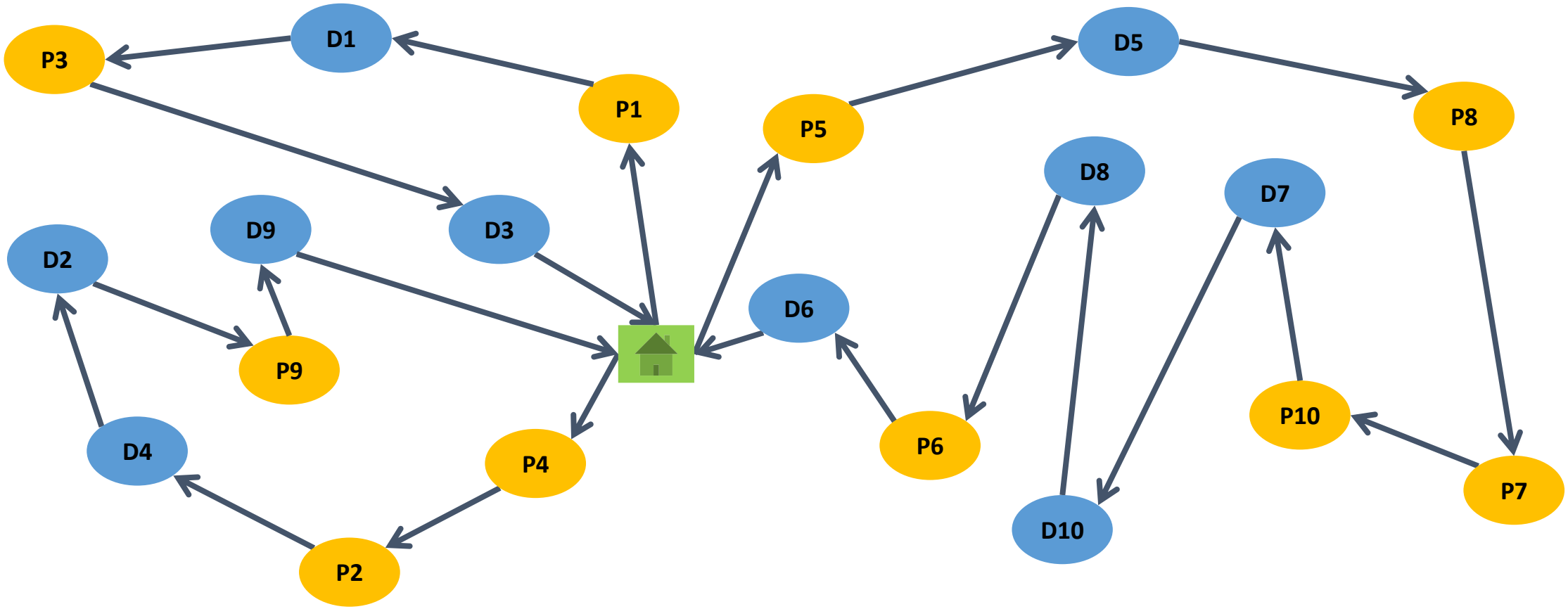
Pickup And Delivery Problem with Time Windows

Why is this problem important?

It is the hidden intelligence behind:

- Uber, for people and goods (e.g. food)
- Dial-a-ride transport (health sector)
- Drayage
- And more…

Pickup And Delivery Problem with Time Windows



Pickup And Delivery Problem with Time Windows

State of the art (2016)

Dominated by heuristics and Branch-and-Price-and-Cut (BPC)

BPC Negatives

- Very difficult: pricing, cutting and branching can all be hard
- Fragile: small changes to the problem can break one of the components
- Doesn’t get faster as solvers get faster
- Have to multi-thread it yourself

BPC Positives

- Only available exact approach
- You get a PhD for making it work on a new problem!

Pickup And Delivery Problem with Time Windows

So how can we do better?

We'd like to wrap up the "difficult" constraints the same way BPC does
We'd like to get a tight lower bound and remove symmetry the way BPC does

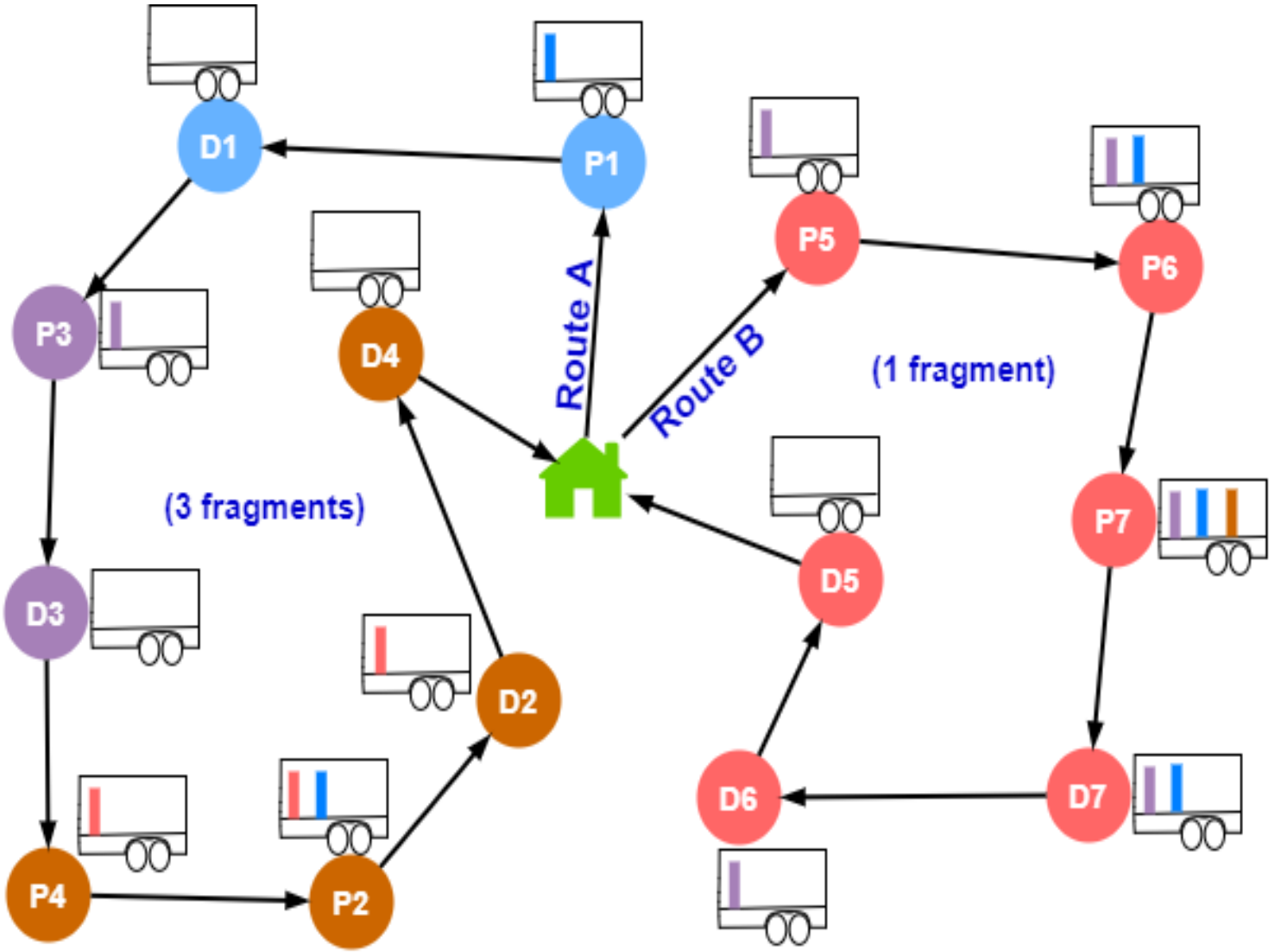
But we'd like to avoid delayed column generation and writing our own branching framework

Fragments

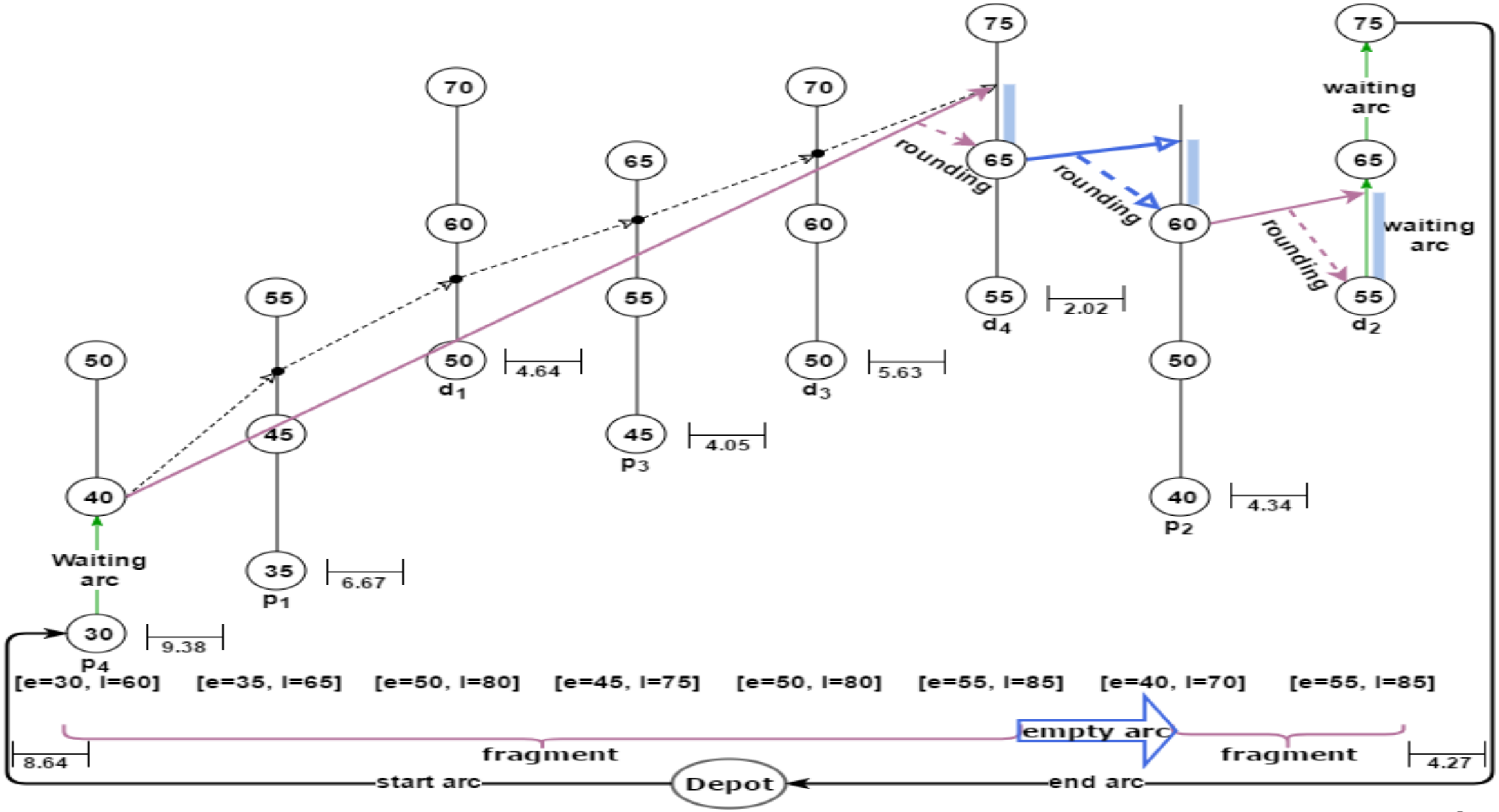
Pickup And Delivery Problem with Time Windows

Fragments:

- A sub-path where the vehicle arrives empty at the pickup node and departs empty from the delivery node.
- Vehicle cannot be empty at any intermediate node.
- Fragments respect all capacity, time window, pairing and precedence constraints, as well as any other loading constraint e.g. LIFO.



Time Discretization



Method Summary

- Generate all undominated fragments.
- Make multiple copies of each fragment for time discretization.
- Do the same for every possible arc connecting delivery to pickup nodes which may be used to connect the fragments.
- Construct an “optimistic” network flow model.
  - Network consists of start arcs, waiting arcs, empty arcs, end arcs
- Check that every chain of fragments is a feasible route whenever the solver finds a candidate improved integer feasible solution.
- Use lazy constraints to cut off the combination of variables corresponding to the smallest infeasible chain of fragments for each infeasible route, and eliminate cycles.

- Very easy to adjust to new rules around fragments.

Formulation

$$Min \sum_{a \in A} c_a y_a + \sum_{\omega \in \Omega} \sigma_\omega x_\omega$$

$Subject\ to$:

$$\sum_{\omega \in \Omega(p)} x_\omega = 1, \qquad \forall p \in P$$

$$\sum_{a \in A, a^+ = i} y_a + \sum_{\omega \in \Omega, \omega^+ = i} x_\omega = \sum_{a \in A, a^- = i} y_a + \sum_{\omega \in \Omega, \omega^- = i} x_\omega \ , \qquad \forall i \in N$$

$$y_a \ , x_\omega \in \{0,1\}, \qquad \forall a \in A, \forall \omega \in \Omega$$

c_a: the cost of (timed) empty arc a

σ_ω: the internal cost of a (timed) fragment

x_ω: equal to 1 if (timed) fragment ω used

y_a: equal to 1 if timed arc a used

The lazy constraint to cut off any time window illegal chain is:

$$\sum_{\alpha \in \{\cup_{\beta \in \beta(r)} T(\beta)\}} y_\alpha + \sum_{\omega \in \{\cup_{f \in f(r)} T(f)\}} x_\omega \ \leq \ 2|f(r)| - 2$$

The lazy constraint to cut off any subtour (cycle) is:

$$\sum_{\alpha \in \{\cup_{\beta \in \beta(r)} T(\beta)\}} y_\alpha + \sum_{\omega \in \{\cup_{f \in f(r)} T(f)\}} x_\omega \ \leq \ 2|f(r)| - 1$$

PDPTWL Results

Python 3.5 and Gurobi 7.0.2 was used to implement our method.

The pickup and delivery problem with time windows and last-in-first-out loading				
	Cherkesly et al. [1]		Our method	
Group	Solved	Average time	Solved	Average time
AA	10	144.6	10	1.3
BB	10	197.6	10	2.2
CC	6	892.1	7	189.0
DD	5	476.4	8	39.0
AA*	8	319.3	10	1.3
BB*	7	461.1	10	2.0
Total	46		55	

[1] M. Cherkesly, G. Desaulniers and G. Laporte, “Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and last-in-first-out loading”, Transportation Science, 49(4), 752–766, 2015.

PDPTWMS Results

The pickup and delivery problem with time windows and multiple stacks						
	Cherkesly et al. [2]		Our method		Our method	
Group	Solved	Average time	Solved	Average time	Previously unsolved	Average time
C1_1 stack	291	433.0	319	0.2	28	4.3
C1_2 stacks	109	1,298.0	315	0.8	206	111.0
C1_3 stacks	84	1,215.6	312	1.4	228	184.0
C2_1 stack	316	77.2	319	0.3	3	4.5
C2_2 stacks	239	644.7	319	2.9	80	74.6
C2_3 stacks	208	835.7	319	2.7	111	222.7
Total	1247		1903		656	

[2] M. Cherkesly, G. Desaulniers, S. Irnich and G. Laporte, “Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and multiple stacks”, European Journal of Operational Research, 250(3), 782–793, 2016.

But what do we do if there are too many fragments?

- Dial-a-ride problem (DARP)
- PDPTW plus maximum ride time restriction
- Many strung out fragments: $(P_1, P_2, D_1, P_3, D_2, P_4, D_3, P_5, \dots)$
- E.g. 48 requests, 394,035 fragments (one with 23 requests!)
- The solution: restricted fragments, fragments with one transition from pickup to delivery. Reduction to 453 restricted fragments, at most 4 requests.

- One route: $(P_1, P_2, D_1, P_3, D_3, D_2, P_4, P_5, P_6, D_4, D_5, P_7, D_7, D_6)$
- Two fragments: $(P_1, P_2, D_1, P_3, D_3, D_2) + (P_4, P_5, P_6, D_4, D_5, P_7, D_7, D_6)$
- Four restricted fragments:
  - $(P_1, P_2, D_1, \cancel{D_2}) + (\cancel{P_2}, P_3, D_3, D_2) + (P_4, P_5, P_6, D_4, D_5, \cancel{D_6}) + (\cancel{P_6}, P_7, D_7, D_6)$
- Restricted fragments can join in two ways:
  - Standard empty join: D_2 to P_4
  - "Overlap" non-empty join: D_1 to P_3 with request 2 on the vehicle.

Restricted Fragments

Pros

- Enormously fewer variables
- Moves many problems from impossible to possible
- Two options for joining overlapping restricted fragments
  - Directly with variables for every pair
  - By arcs (looser but fewer variables)
- Exactly the same formulation as fragment formulation, but with different definition of nodes and arcs

Cons

- Significantly more constraints (more flow conservation)
- Looser LP relaxation
- Trickier code (especially around fragment domination)
- Time discretisation really blows out model size

DARP Results

The dial a ride problem – difficult instances Times in seconds – 3600 second limit								
	Orig		4/3		5/3		6/3	
Instance	[3]	**New**	[3]	**New**	[3]	**New**	[3]	**New**
B6-72	31.4	**2.9**	1691.4	**9.9**	-	**30.5**	696.8	**39.2**
B7-56	50.3	**2.2**	26.8	**6.8**	38.5	**23.7**	347.4	**108.4**
B7-70	13.0	**3.4**	50.0	**8.4**	47.8	**16.8**	-	**117.0**
B7-84	71.7	**4.6**	518.6	**9.6**	-	**77.9**	-	**478.2**
B8-64	23.1	**1.9**	9.3	**8.9**	73.5	**18.5**	-	**366.2**
B8-80	10.3	**1.0**	15.4	**6.7**	-	**34.9**	-	**438.5**
B8-96	898.8	**7.3**	2135.6	**19.7**	-	**95.6**	-	**3446.4**

[3] T. Gschwind and S. Irnich, “Effective Handling of Dynamic Time Windows and Its Application to Solving the Dial-a-Ride Problem”, Transportation Science 49, 335-354 (2015).

Cuts

Lift lazy constraints

$$\sum_{\alpha \in Arcs\ in\ Chain} y_\alpha + \sum_{\omega \in Fragments\ in\ Chain} x_\omega \leq Length\ Chain - 1$$

Alternating chain of fragments and (empty) arcs, starting and ending with a fragment.

1. Augment sum of empty arcs with extra forward empty arcs
2. Augment first (last) fragment in chain with all fragments that make the chain infeasible
3. Change the logic to look at options for feasible fragments at the start or end of the chain (must be tighter)

$$\sum_{\alpha \in Arcs\ in\ Chain} y_\alpha + \sum_{\omega \in Fragments\ in\ Chain-end} x_\omega \leq Length\ Chain - 2 + \sum_{\omega \in Feasible\ Ends} x_\omega$$

Cuts

Add cuts to LP relaxation

Fragment f ending at location D_i

$$x_f \leq y_{D_i Depot} + \sum_{j | f + (P_j, D_j)\ feasible} y_{D_i P_j}$$

Can augment f with all "supersets" of f
Similar cut at start of fragment
Similar cuts with arcs on the LHS and sum of fragments on the RHS

Big impact if time is not discretised (very important for DARP results)

Restricted Fragments
Open Questions

- Additional problem variants
  - “Practical” PDPTW, Dial a flight, Practical DARP, Drayage, PDPTW with Service Lines, …
- Best ways to link restricted fragments (with time discretisation?)
- Use DDD instead of fixed discretisation
- Details and proofs around dominating (restricted) fragments
- Different objectives (especially duration)
- Other types of cuts
- Theory around combination of Benders and Dantzig-Wolfe
- Application of combined Benders and Dantzig-Wolfe to other problems
- …