

Decision Diagrams for Real-Time Routing

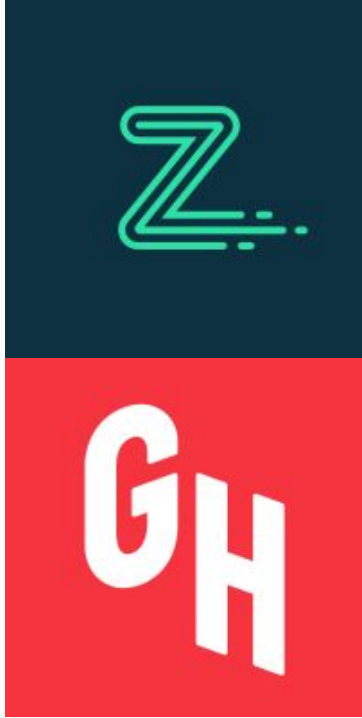
Ryan J. O'Neil, nextmv.io
Karla Hoffman, George Mason University

INFORMS Annual Meeting
October 22, 2019

This talk is about optimizing small pickup and delivery routes very quickly.

Motivation

Problem Origin: Meal Delivery in Industry

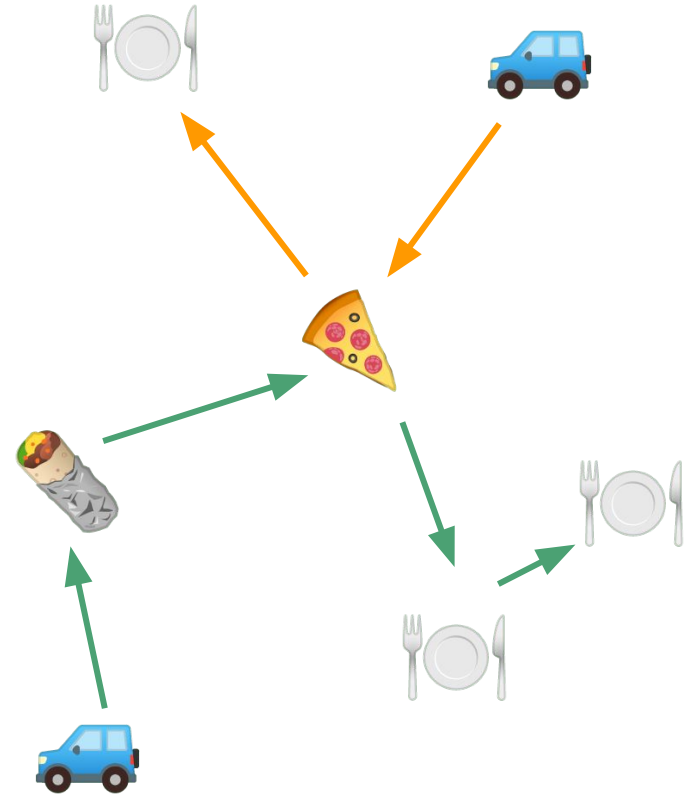


Original motivation came from routing at Zoomer. Work continued at Grubhub Delivery after they picked up our Decision Engineering team.

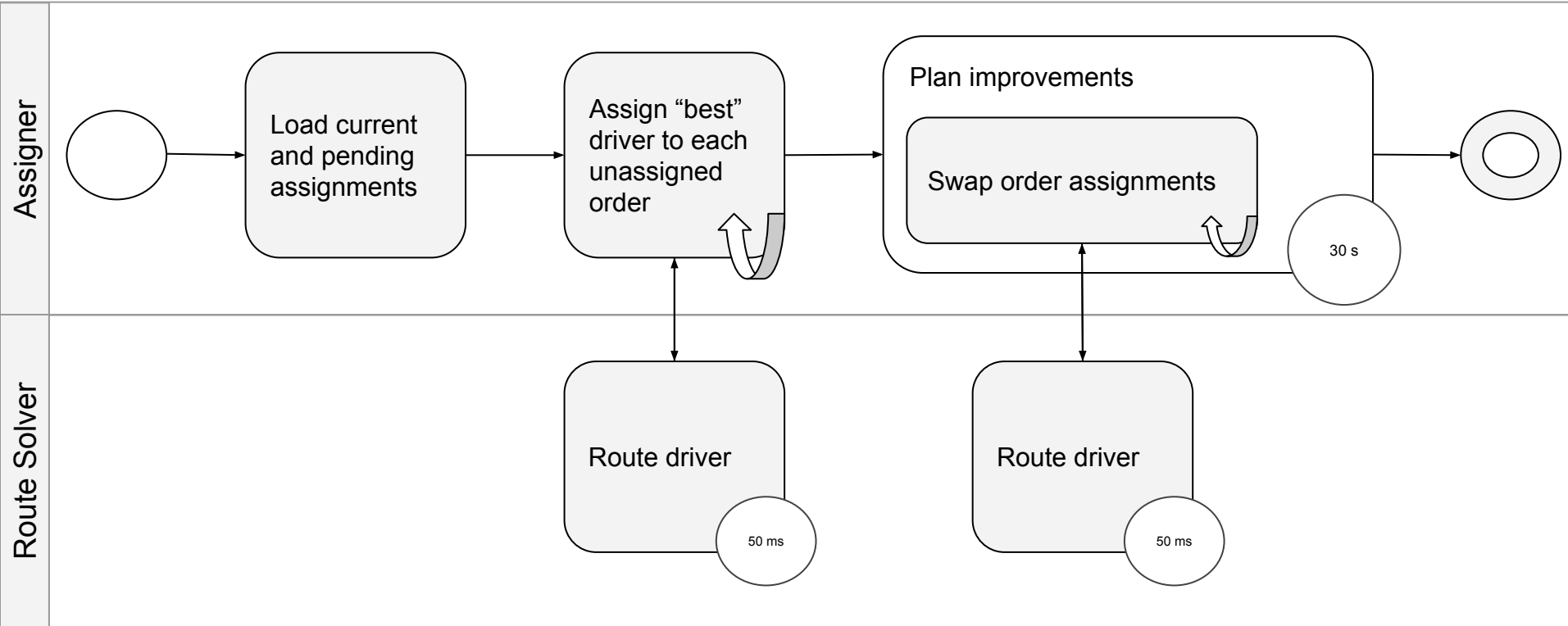
Both solve dynamic meal delivery problems. The biggest differences in operational context are scale, and that Zoomer did not have a marketplace.

Dynamic Meal Delivery

- Orders arrive dynamically throughout operational period.
- Shared pool of drivers serves many restaurants.
- Multiple orders can be consolidated on drivers for efficiency.
- Problems get large (e.g. thousands of orders, hundreds of drivers).



Assignment & Routing Process

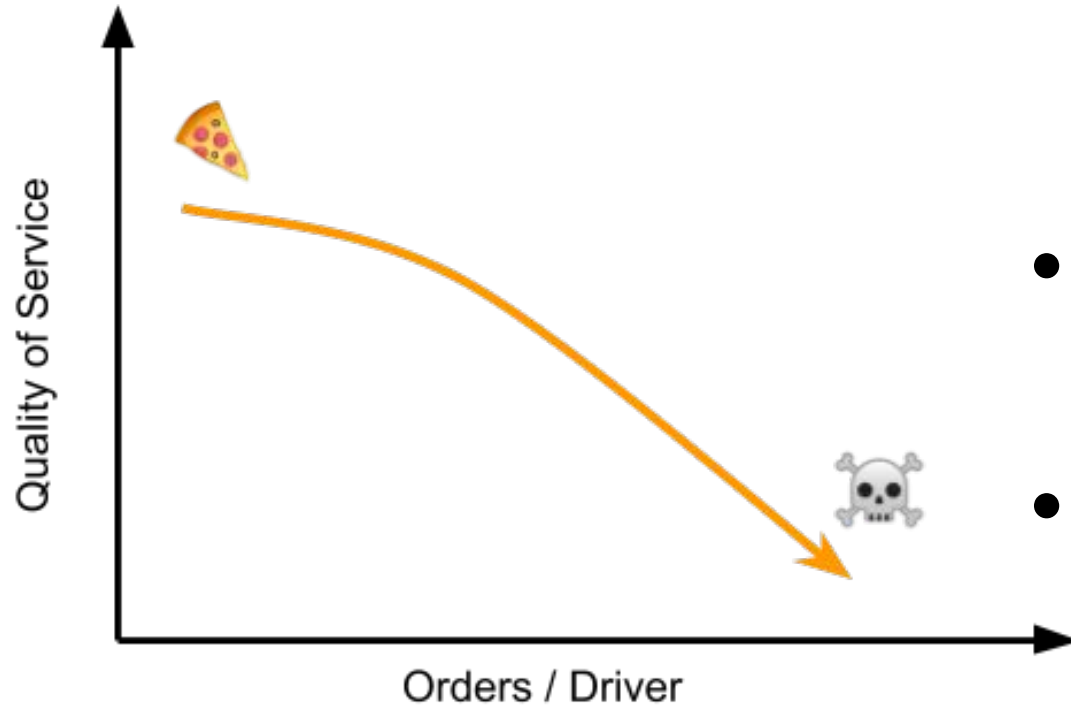


Periodic replanning. Assignment uses improvement heuristic. Route solver heuristically finds "good" routes given a driver and set of orders.

Assignment & Routing Process

- Quality of assignments and routes are interdependent.
- Normal planning periods are every 30 seconds to 2 minutes.
- Planners time out. If an algorithm exceeds its limit, it isn't done improving yet. But it still has to provide a plan.
- Every millisecond matters. Saving time solving one problem means we can spend that time solving another.

Effect of Supply-Demand Imbalance



- Some service degradation is unavoidable. Too many orders, not enough drivers.
- Some is algorithmic. Longer routes are harder to optimize!
- This can happen when solving 8-order routes in high volume.

Options for Handling Undersupply

- **Wait and See:** Demand is stochastic. Maybe it'll pass?
- **Add Drivers:** Even if possible, increases financial liability.
- **Underserve Orders:** Affects LTV. Probably violates SLAs.
- **Manage Supply with ETAs:** Requires control of order intake.
- **Cancel Orders or Blackout Restaurants:** Worst case scenario.

More robust service through better routing is less disruptive.

Our Mission: Faster Single-Vehicle Route Solving

- Orders were assigned as late as possible to maximize options.
- Orders were continually planned for to set expectations with restaurants and dispatchers.
- During periods of supply-demand imbalance, most of the planning time was spent constructing single-vehicle pickup and delivery routes.

Our Approach: Use Exact Methods



Tallys Yunes

@thyunes



Replying to [@thserra](#)

My version: When you finish a PhD in mathematical optimization, they take you to a special room and explain that, in real life, large companies with gigantic problems actually use heuristics way more often than optimization. 🗑️ #orms

6:40 AM · Sep 25, 2019 · [Twitter Web App](#)

Our Approach: Use Exact Methods

- Solvers (e.g. Gecode) allow us to separate model structure from search algorithm. That's good software design!
- Single-vehicle routes are small. If a solver can prove optimality quickly, that saves time.
- We want techniques that find good (e.g. within 10% of optimal) solutions very quickly and are capable of proving optimality.

Single-Vehicle Pickup & Delivery Problems

The TSP with Pickup & Delivery (TSPPD)

Notation:

$(+i, -i)$ are a pickup and delivery pair

$+i$ = pickup i

$-i$ = delivery i

$(+0, -0)$ are the start and end locations

The TSPPD

Objective:

Find a minimum-cost Hamiltonian tour in which $+i$ precedes $-i$ for all pickup & delivery pairs $(+i, -i)$.

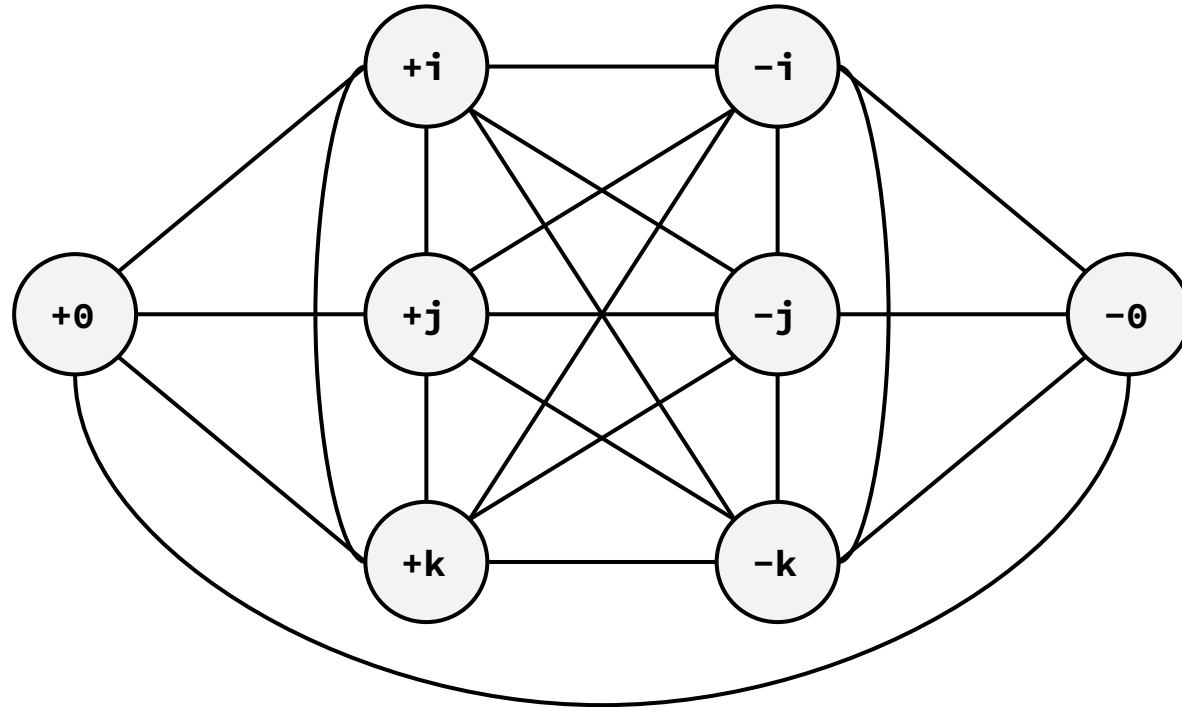
$(+0 \ +i \ +j \ -i \ +k \ -j \ -k \ -0)$

Feasible TSPPD tour

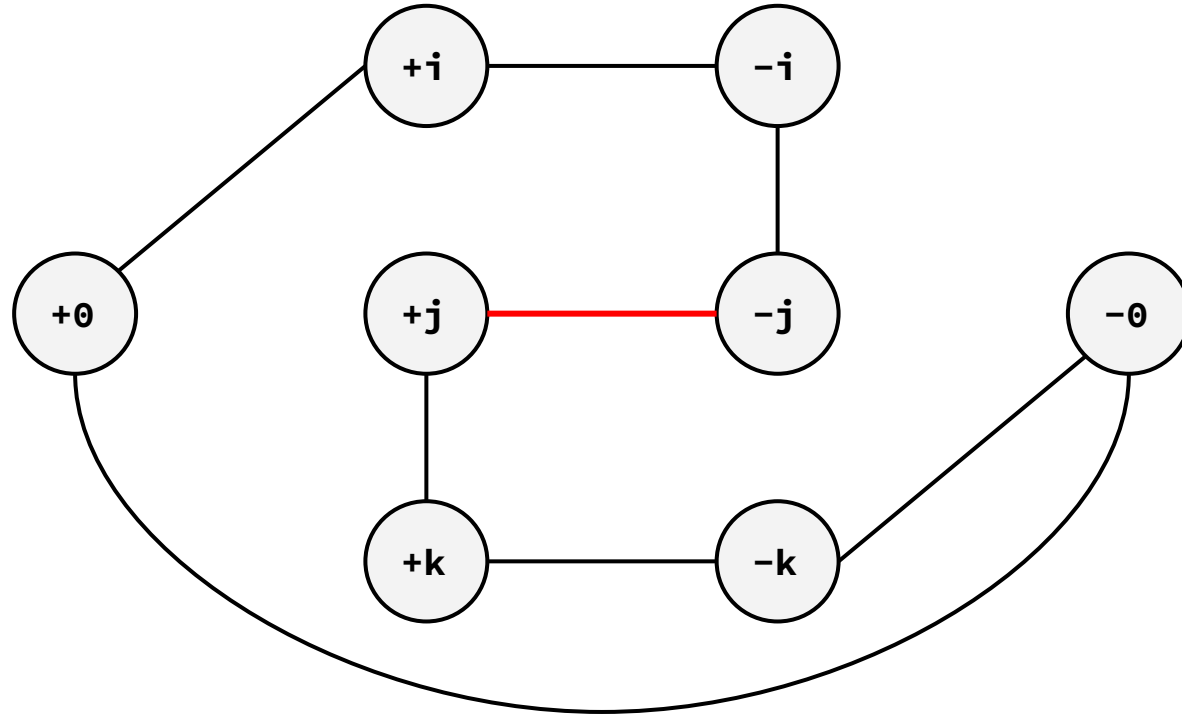
$(+0 \ +i \ +j \ -i \ -k \ -j \ +k \ -0)$

Infeasible TSPPD tour

TSPPD graph structure



TSPPD Graph Structure: Precedence Violation

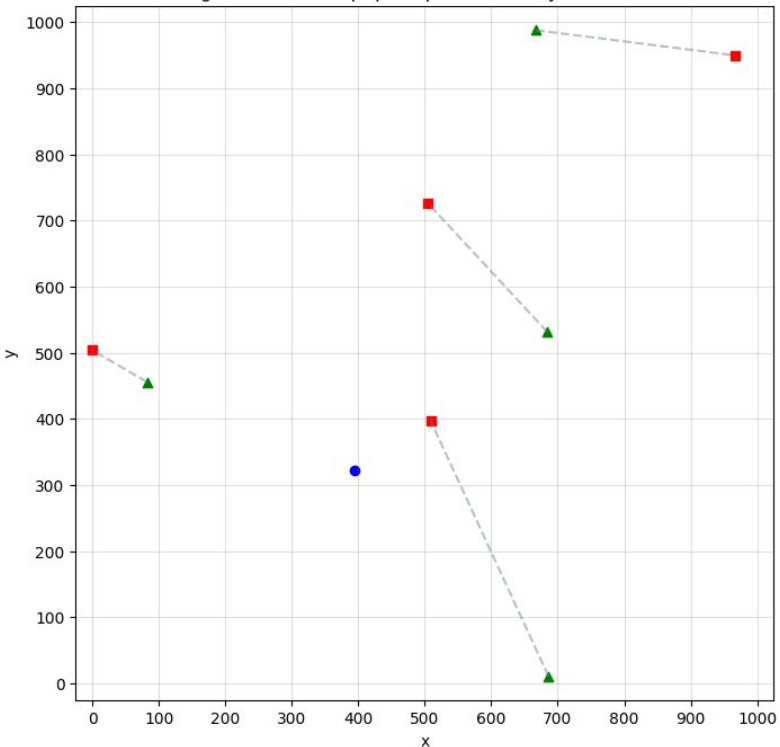


Grubhub TSPPD Test Set

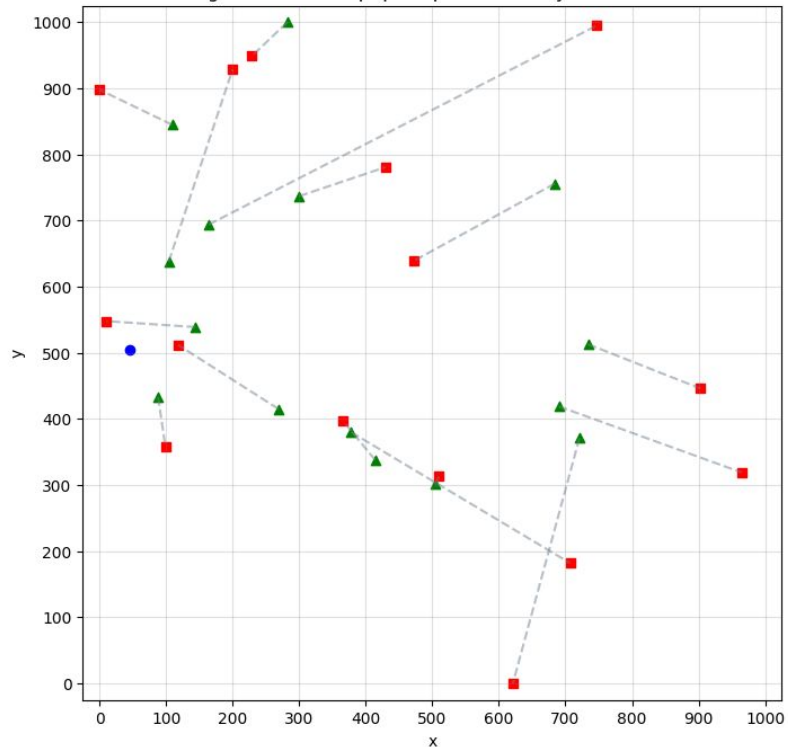
- Meal delivery problems have unique characteristics, so we built a test set from real delivery data and expected travel times.
- Restaurants often appear in clusters. Most orders originate from a subset of them.
- There are many more diners than restaurants. Diners are often dispersed in residential areas.

Grubhub Test Set: TSPPD Problems

grubhub-04-7.tsp: pickup and delivery locations



grubhub-15-4.tsp: pickup and delivery locations



- Courier
- ▲ Pickup
- Delivery

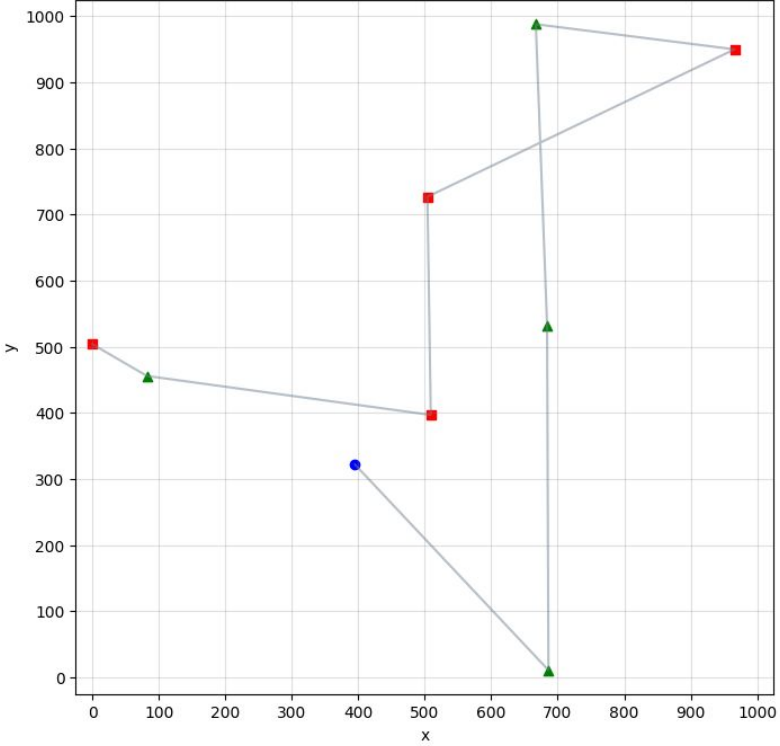
People, Meals, Perishable Goods

Groceries, Packages, Non-Perishable Goods

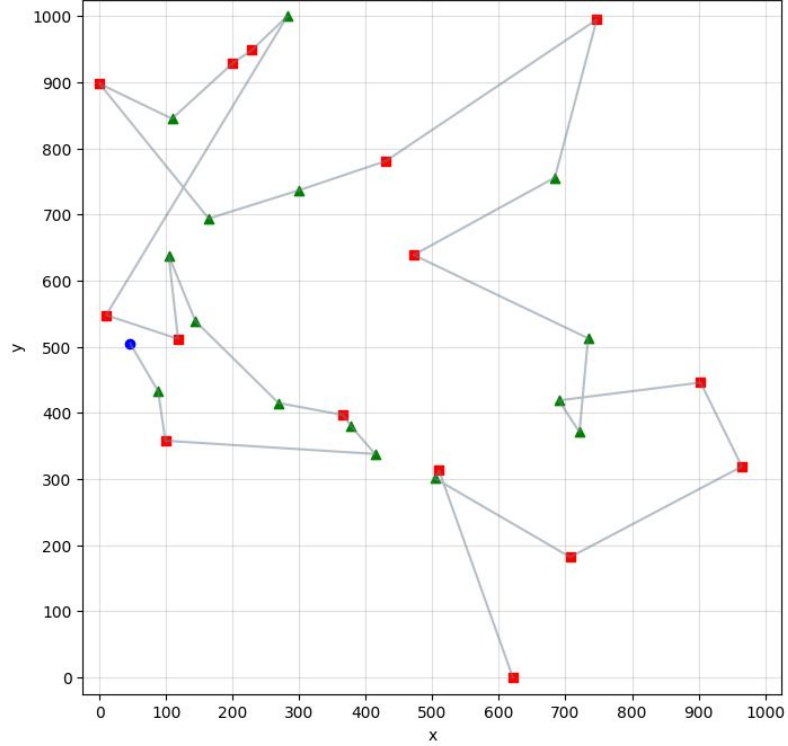





Grubhub Test Set: TSPPD Solutions

grubhub-04-7.tsp: optimal tour cost = 3987



grubhub-15-4.tsp: optimal tour cost = 10035



-  Courier
-  Pickup
-  Delivery

People, Meals, Perishable Goods

Groceries, Packages, Non-Perishable Goods



Models & Results

Enumeration, CP, and MIP Models

- In a previous study, we tested a number of models (enumerative, MIP, CP, hybrid) for time to prove optimality, find a solution within 10% of optimal, and other measures.
- The best of these for finding good solutions quickly (and warm starting MIP) was a circuit-based CP model with an Assignment Problem inference dual.

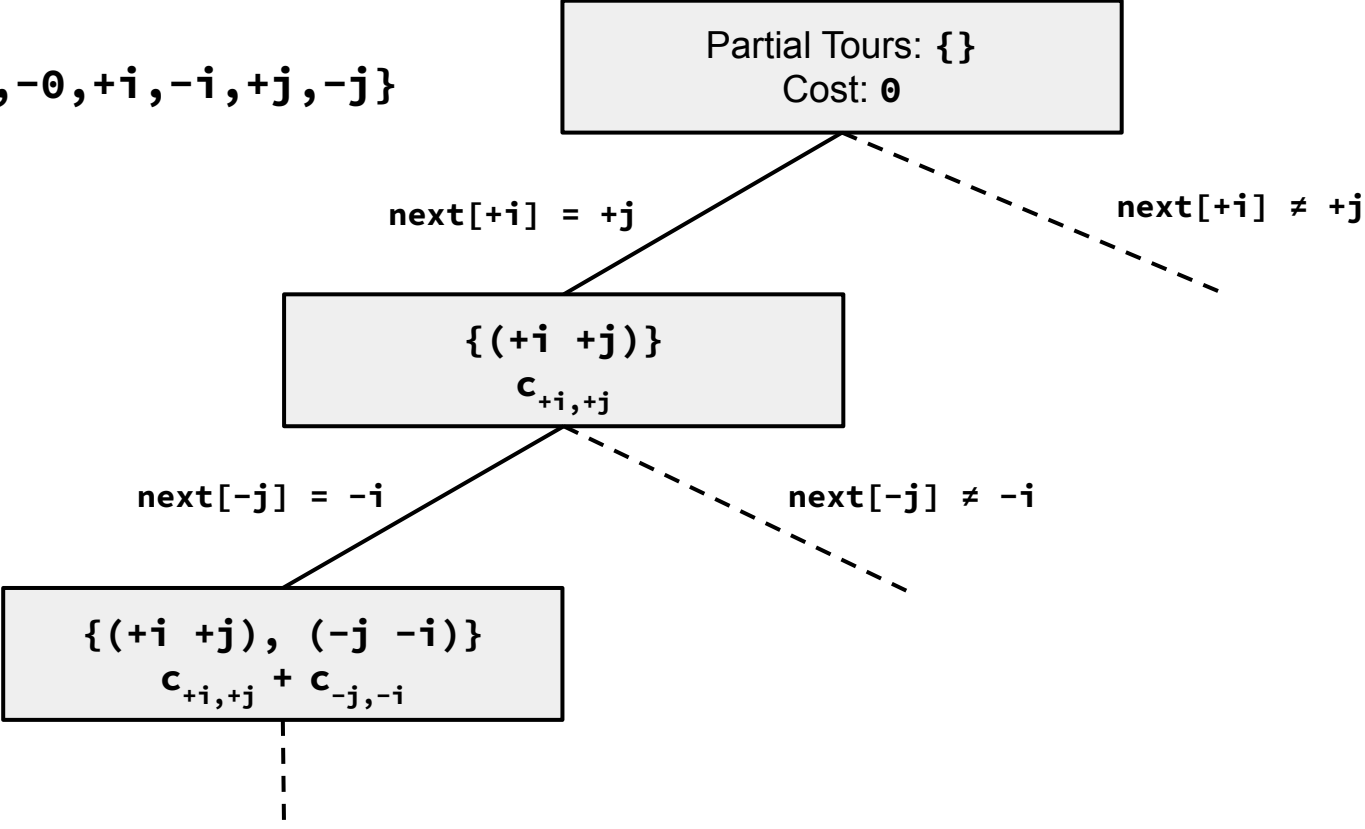
CP Model: next Vector

- If **next**[**i**] = **j** then proceed directly from **i** to **j**
- Each element of **next** has a feasible domain of nodes
- Assignment of **next**[**i**] is implied if **dom(next**[**i**]) is a singleton

Variable	Initial Domain
next [+0]	+i +j +k
next [+i]	+j +k -i -j -k
next [-i]	+j +k -j -k -0
next [+j]	+i +k -i -j -k
next [-j]	+i -i +k -k -0
next [+k]	+i +j -i -j -k
next [-k]	+i +j -i -j -0
next [-0]	+0

CP Model: Branching Rules

Nodes: $\{+0, -0, +i, -i, +j, -j\}$



CP Model: Branching Rules

Branching on variable assignment:

$$\mathbf{next[i] = j} \quad \forall \quad \mathbf{next[i] \neq j}$$

- Sequential Closest Neighbor (SCN): $(\mathbf{i} \ \mathbf{j})$ is the least-cost feasible arc starting at $\mathbf{+0}$ and building a single, connected tour.
- Regret: \mathbf{i} is the max-regret node, $(\mathbf{i} \ \mathbf{j})$ is the least cost feasible arc from \mathbf{i} .

$$\mathbf{regret(i)} = \text{second smallest arc cost from } \mathbf{i} \\ - \text{smallest arc cost from } \mathbf{i}$$

CP Model: `circuit` Constraint

The **`circuit`** constraint in Gecode ensures that:

- No two **`next`** variables have the same value
- There are no subtours (i.e. cycles)
- Tour cost propagates into a variable (**`z`**)

Variable	Domain		Variable	Domain
<code>next[+0]</code>	<code>+i +j</code>	<code>next[-j] = +i</code> →	<code>next[+0]</code>	<code>+j</code>
<code>next[+i]</code>	<code>+j -i -j</code>		<code>next[+i]</code>	<code>+j -i</code>
<code>next[-i]</code>	<code>+j -j -0</code>		<code>next[-i]</code>	<code>-0</code>
<code>next[+j]</code>	<code>+i -i -j</code>		<code>next[+j]</code>	<code>-i -j</code>
<code>next[-j]</code>	<code>+i -i -0</code>		<code>next[-j]</code>	<code>+i</code>
<code>next[-0]</code>	<code>+0</code>		<code>next[-0]</code>	<code>+0</code>

CP Model: circuit Precedence

- We maintain predecessor and successor sets:

$$\mathbf{pred}[-i] = \{+i\}, \mathbf{succ}[+i] = \{-i\}$$

- These sets must be disjoint:

$$\mathbf{pred}[i] \cap \mathbf{succ}[i] = \emptyset$$

- Branching propagates to **pred** and **succ**:

$$\begin{aligned} \mathbf{next}[i] = j \rightarrow \mathbf{pred}[j] \supset \mathbf{pred}[i] \\ \wedge \mathbf{succ}[i] \supset \mathbf{succ}[j] \end{aligned}$$

Plus additional inferences rules.

Decision Diagram Approach

- Use Restricted DDs to generate primal solutions and bounds.
- Vary the diagram width: {0, 5, 10, 15, 20}.
- Use the Assignment Problem solver for dual bounds and reduced cost-based variable domain filtering.
- Integrate the DD primal and AP dual into a branch-and-bound.

Decision Diagram TSPPD Model

$$\min \sum_{i=0, \dots, 2n} c(\pi_i, \pi_{i+1})$$

Route cost

$$\text{s.t. } \pi_i = +0$$

Start at +0

$$\pi_{i+1} \in D_i, \quad i = 0, \dots, 2n-2$$

Domain for next nodes

$$D_0 = V_+$$

Go from +0 to a pickup

$$D_{i+1} = D_i \cup \{-\pi_i\} \setminus \{\pi_i\} \quad \text{if } \pi_i \in V_+$$

Pickups precede deliveries

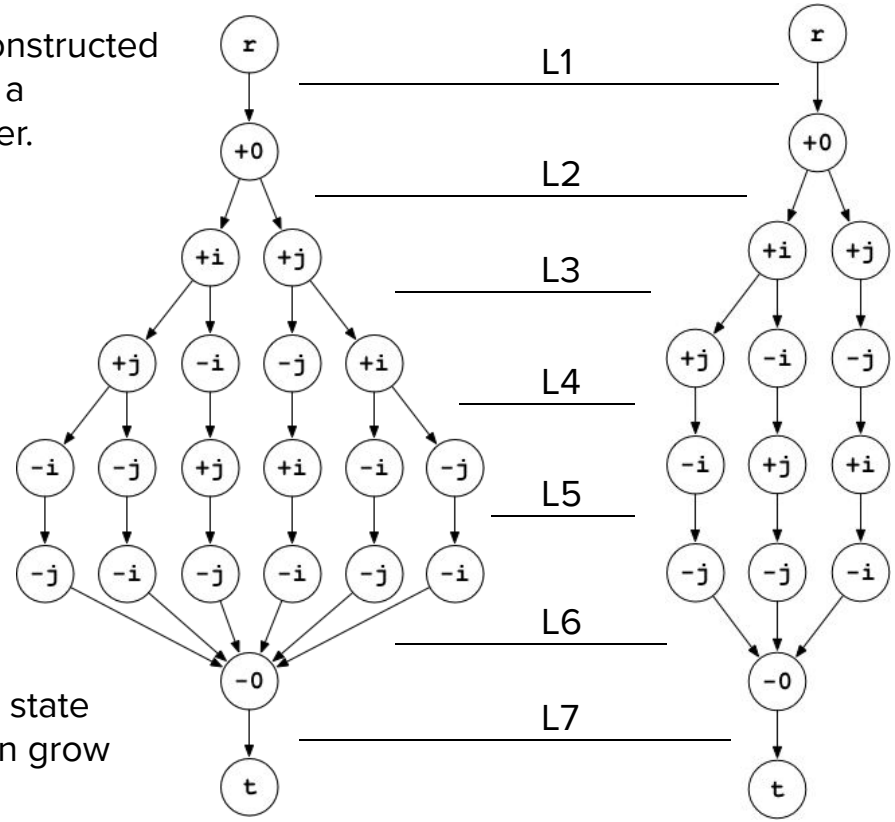
$$D_{i+1} = D_i \setminus \{\pi_i\} \quad \text{if } \pi_i \in V_-$$

$$D_{2n} = \{-0\}$$

End at -0

Exact and Restricted Diagrams for TSPPD

Diagrams are constructed layer by layer in a top-down manner.



Restricted diagrams act as a primal heuristic by dropping the worst states at each layer.

Exact diagrams fully explore the state space, which can grow very large.

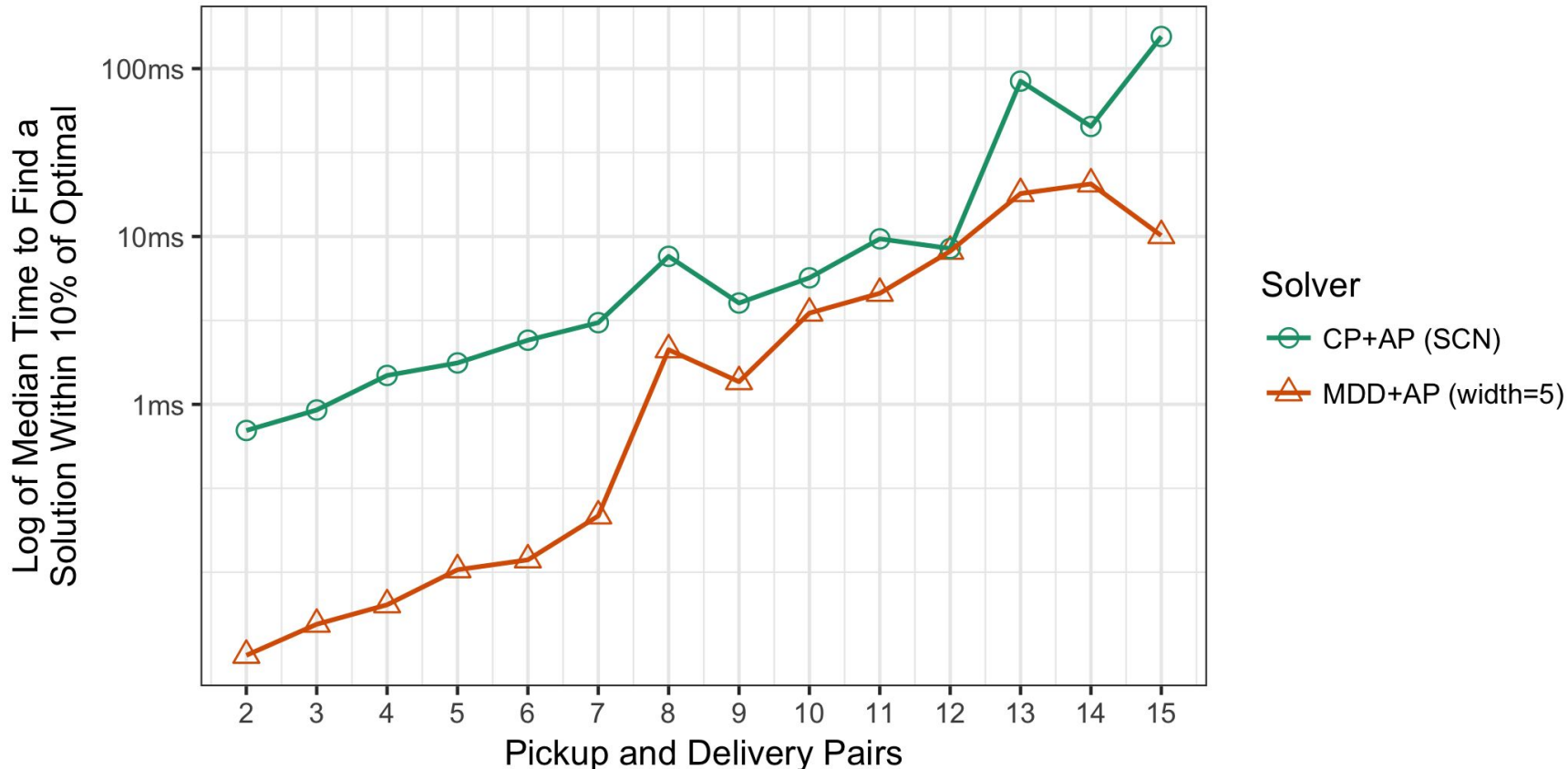
Assignment Problem Inference Dual

- Maintain an Assignment Problem (AP) inference dual throughout the search tree to provide dual bounds.
- As nodes are appended to the tour, they are fixed in the AP inference dual.
- Primal-dual algorithm also yields reduced costs for feasible arcs. If a reduced cost for an arc plus the cost of the partial tour exceeds the incumbent cost, that arc is eliminated.
- First solution of AP executes in $O(n^3)$ time. Subsequent iterations are $O(n^2)$ time.

Assignment Problem

$$\begin{aligned} \mathbf{min} \quad & \sum_{i,j} c_{ij} x_{ij} \\ \mathbf{s.t.} \quad & \sum_j x_{ij} = 1 \quad \forall i \\ & \sum_i x_{ij} = 1 \quad \forall j \\ & x_{ij} \in \{0,1\} \quad \forall i,j \end{aligned}$$

Hybrid CP vs Hybrid DD for Finding Good Solutions



MDD model is more effective than CP model at finding high quality solutions quickly.

Conclusions

Key Differences: CP & DD Models

- DD state is extremely compact. It consists of the current node, a pointer to the parent state, and a single domain vector (instead of n variables domains).
- DD model lacks finite domain propagation.
- CP branch-and-bound in Gecode uses depth-first search. Top-down DD construction approximates a breadth-first search.

Using DDs: Considerations

- There are no commercial-grade DD solvers (*yet*).
- Incorporating relaxation and inference duals is challenging.
- DDs can find good solutions to very complex problems quickly.
- With work, they can prove optimality faster than MIP on some problems (e.g. certain set covers, MISPs).
- State formulations allow *composable* and *testable* models.

Contributions

- Show applicability of Decision Diagrams to real-time routing problems in meal delivery.
- Hybridize DD model with Assignment Problem inference dual. Comparison with baseline CP model.
- Realistic benchmark set of TSPPD instances for meal delivery.

Future Work

- Extend TSPPD model to multi-vehicle routing and assignment.
- Incorporate and test additional relaxation and inference duals.
- Add side constraints common to meal delivery, such as ready-by times and driver capacity.

References

- This work:
Decision Diagrams for Solving Traveling Salesman Problems with Pickup and Delivery in Real Time (Operations Research Letters, 2019)
- Test instances and source code:
<https://github.com/grubhub/tsppdlib>
<https://github.com/ryanjoneil/tsppd-dd>
- CP model:
Exact Methods for Solving Traveling Salesman Problems with Pickup and Delivery in Real Time (Optimization Online)

Other References

- A. A. Cire and W.-J. van Hoeve. Multivalued Decision Diagrams for Sequencing Problems. *Operations Research*, 61(6):1411–1428, 2013.
- F. Focacci, A. Lodi, and M. Milano. A Hybrid Exact Algorithm for the TSPTW. *INFORMS Journal on Computing*, 14(4):403–417, 2002.
- K. Ruland and E. Rodin. The Pickup and Delivery Problem: Faces and Branch-and- Cut Algorithm. *Computers & Mathematics with Applications*, 33(12):1–13, 1997.

