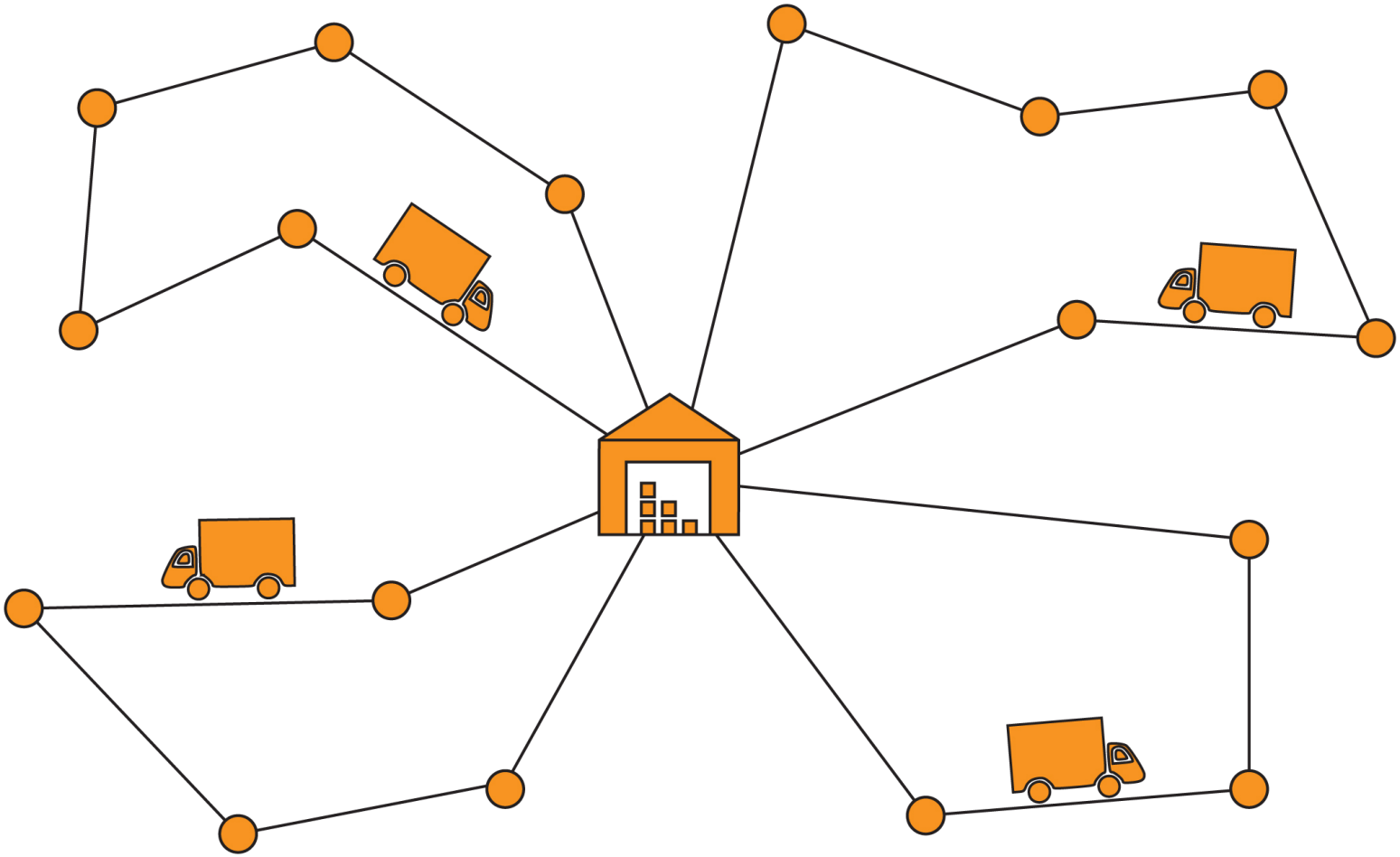
An iterative sample scenario approach for the dynamic dispatch waves problem

Leon Lan[1*] Jasper van Doorn[1]
Niels Wouda[2] Arpan Rijal[2] Sandjai Bhulai[1]

[1]Vrije Universiteit Amsterdam, The Netherlands
[2]University of Groningen, The Netherlands
[*]Presenting author

November 30, 2023

Vehicle routing problems



Same-day delivery

Same-day delivery

- Same-day delivery problems are dynamic vehicle routing problems with stochastic *delivery* requests.
- 90% of the studies on same-day delivery problems have been published in the last five years.[1]
- Applications in e-commerce, meal delivery, and urban consolidation centers.

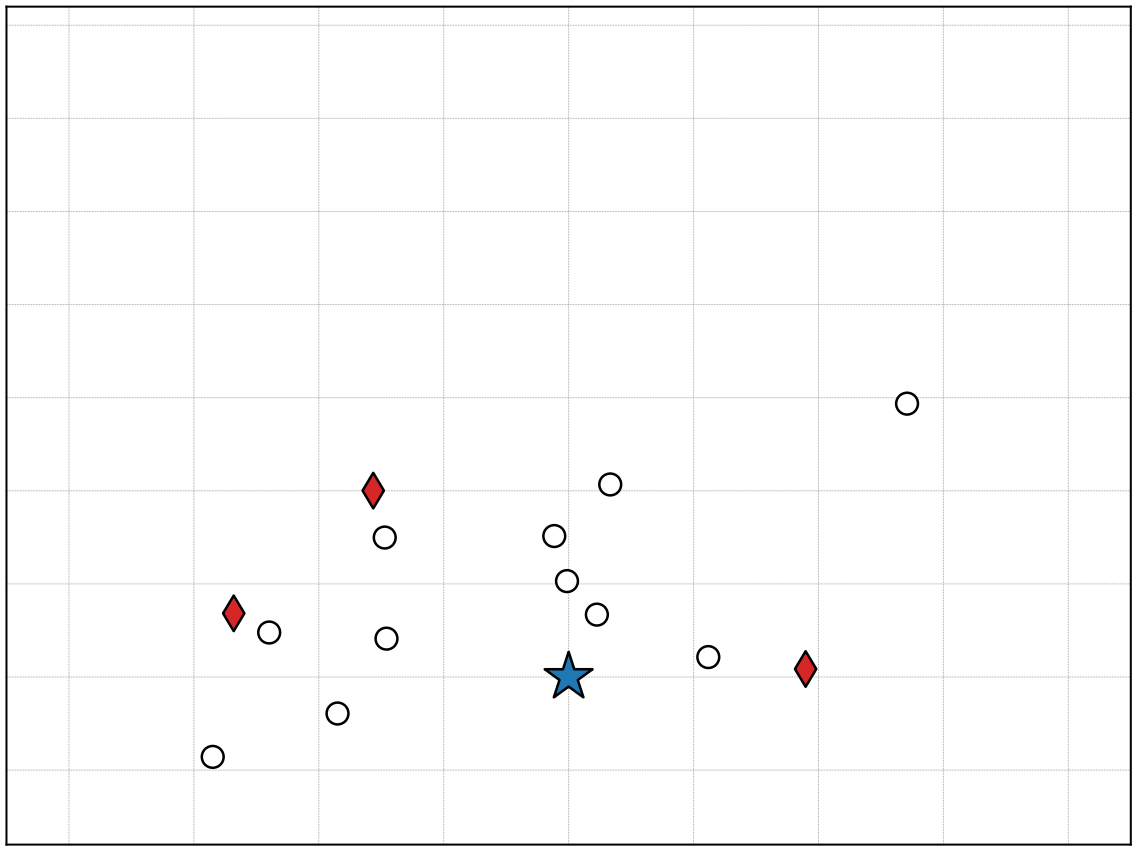
[1] J. Zhang and T. V. Woensel (2023). "Dynamic Vehicle Routing with Random Requests: A Literature Review". en. In: *International Journal of Production Economics* 256, p. 108751

Key points of this talk

- We formulate the *dynamic dispatch waves problem* (DDWP) that was introduced during the *EURO meets NeurIPS 2022* vehicle routing competition.[2]
- We propose an iterative method based on solving sample scenarios to tackle this problem.

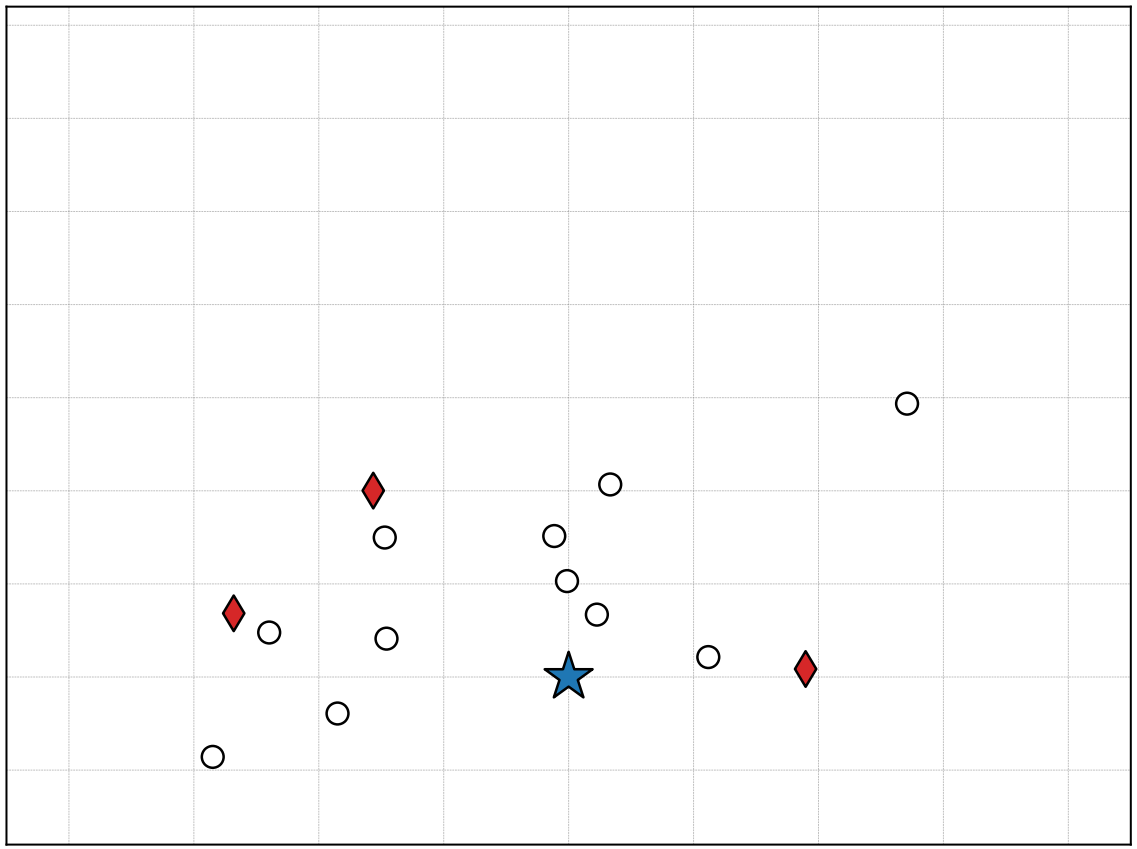
2 W. Kool, D. Numeroso, R. Reijnen, R. R. Afshar, T. Catshoek, K. Tierney, E. Uchoa, and J. Gromicho (2022). *EURO Meets NeurIPS 2022 Vehicle Routing Competition*.

Dynamic dispatch waves problem



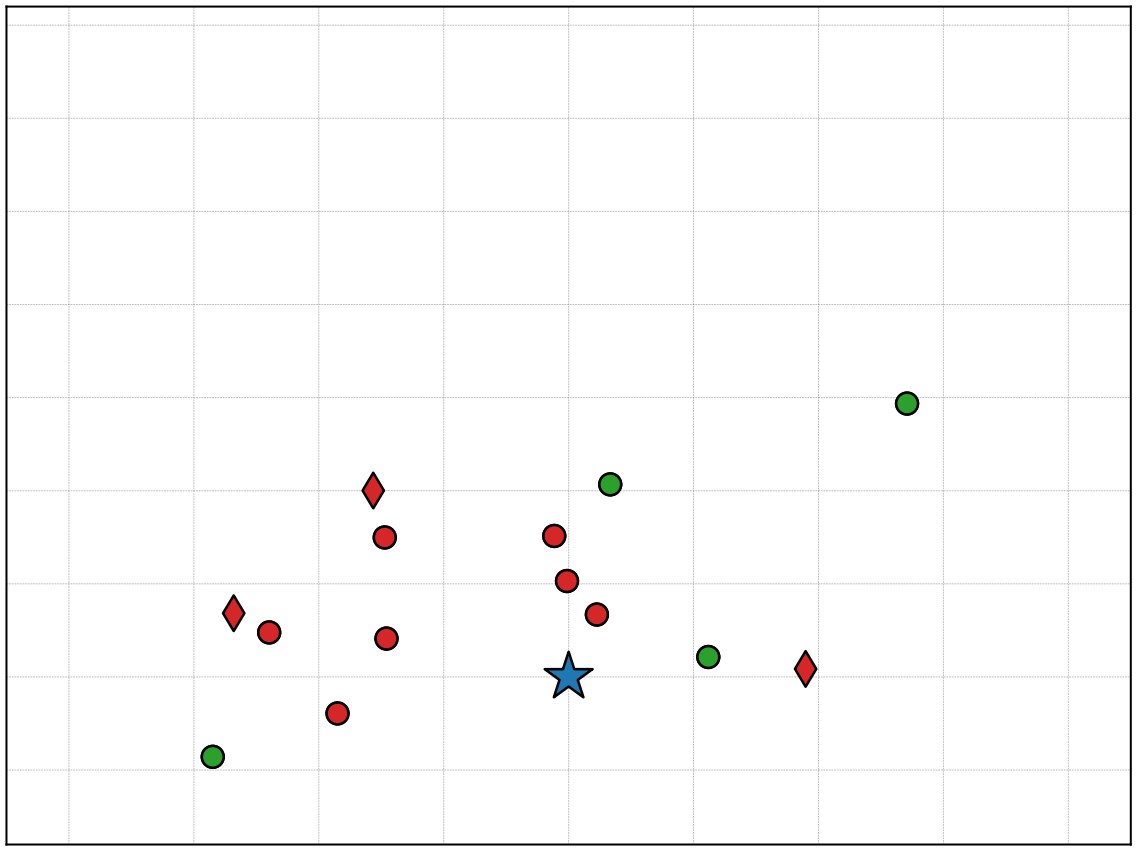
A set of delivery requests. Red diamonds correspond to must-dispatch requests. Blue star represents the depot.

Dynamic dispatch waves problem



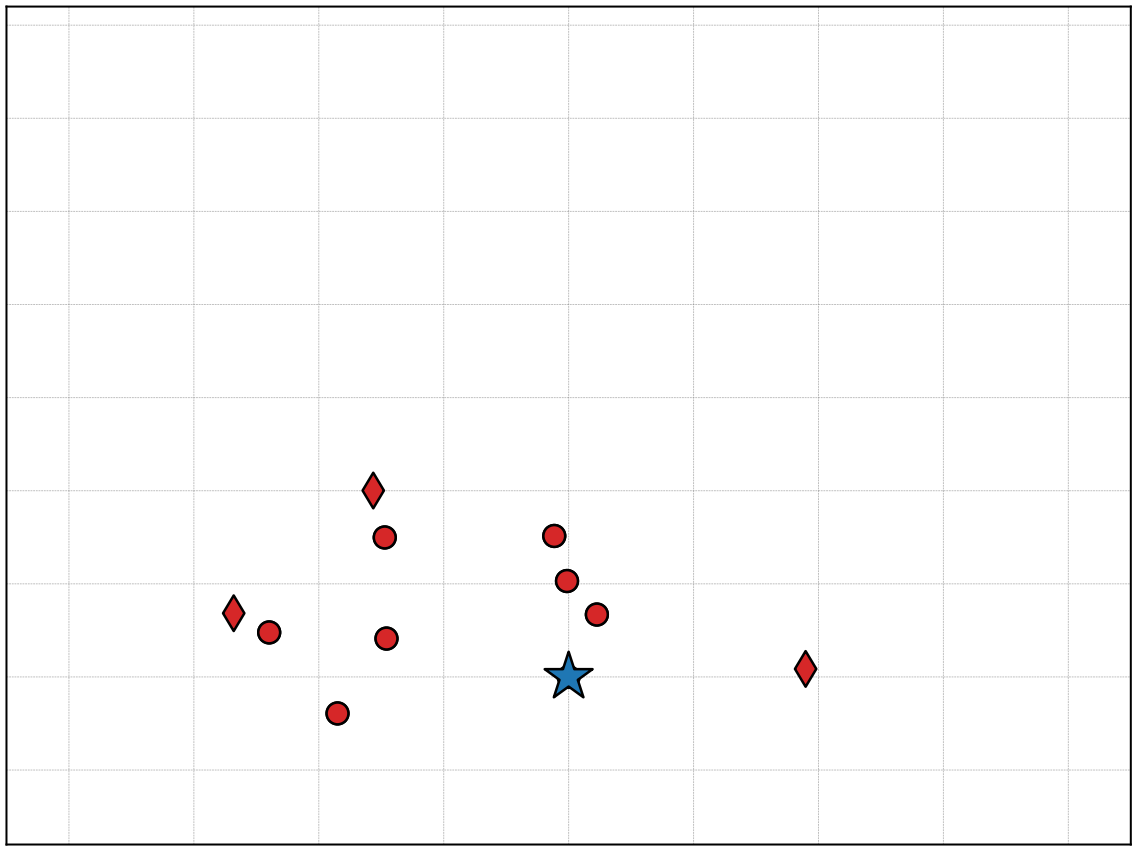
Decide which requests to dispatch (and how to route them) or postpone.

Dynamic dispatch waves problem



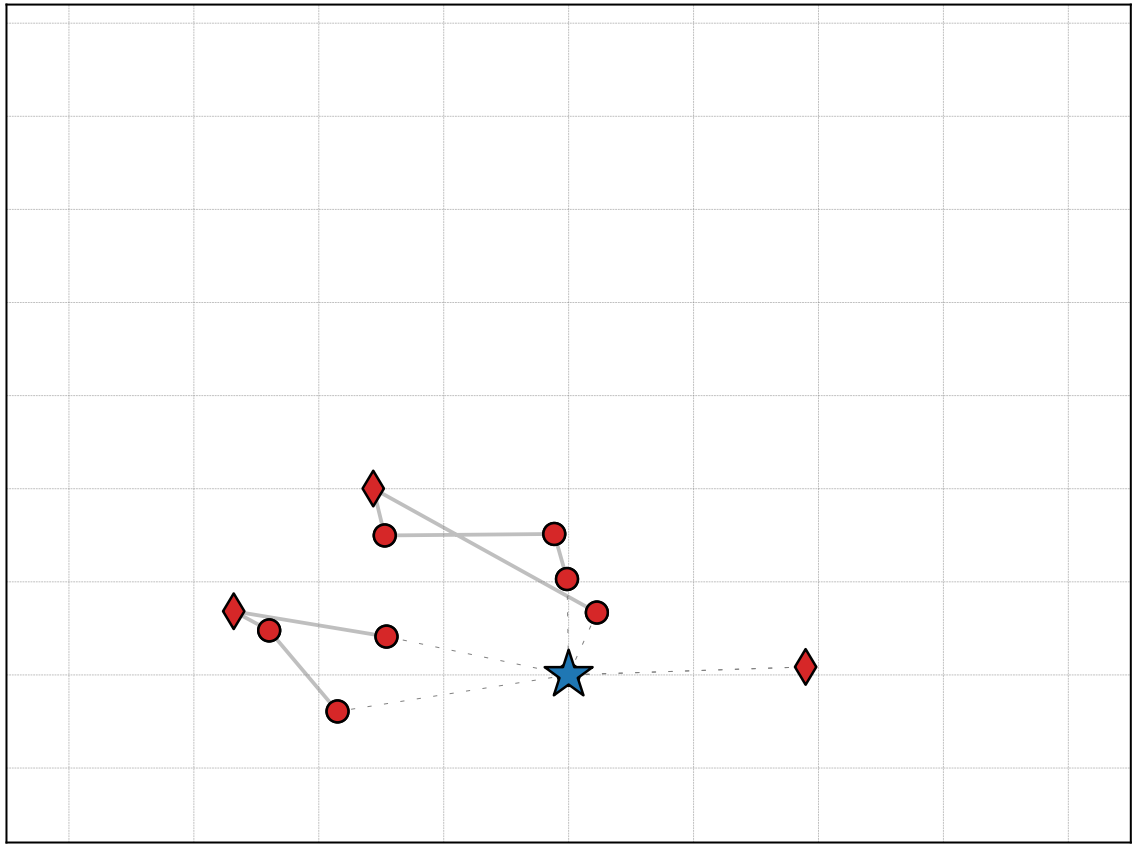
Decision made. Red means dispatched, green means postponed.

Dynamic dispatch waves problem



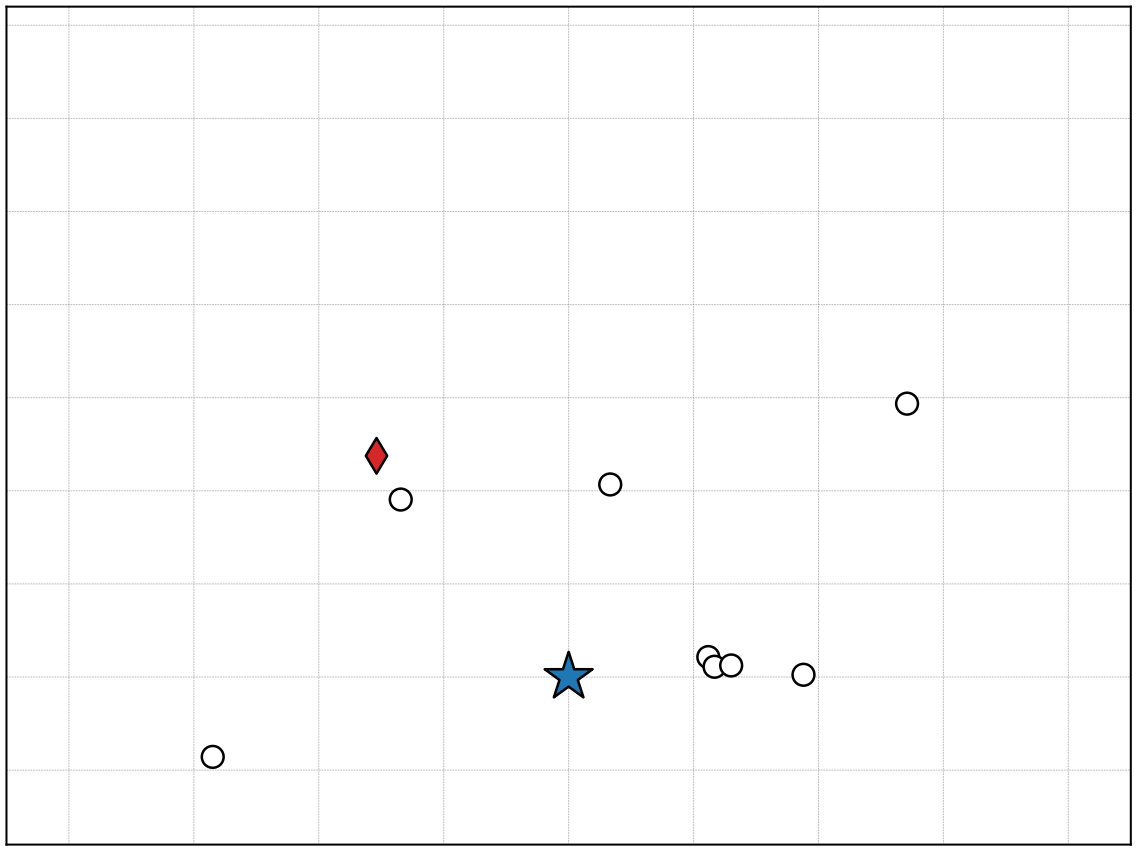
Given the dispatched requests, the next step is to determine good routes.

Dynamic dispatch waves problem



The routing solution.

Dynamic dispatch waves problem



New decision moment. The set of requests is formed by new arrivals and previously postponed requests.

Dynamic dispatch waves problem

- **Goal**: deliver all requests within their time windows at minimum total traveling distance.

Dynamic dispatch waves problem

- **Goal**: deliver all requests within their time windows at minimum total traveling distance.
- **Assumption**: arrival process distribution is assumed to be known or data is available.

Dynamic dispatch waves problem

- **Goal**: deliver all requests within their time windows at minimum total traveling distance.
- **Assumption**: arrival process distribution is assumed to be known or data is available.
- **Assumption**: unlimited fleet of homogeneous vehicles, and no multi-trips.

Dynamic dispatch waves problem

- **Goal**: deliver all requests within their time windows at minimum total traveling distance.
- **Assumption**: arrival process distribution is assumed to be known or data is available.
- **Assumption**: unlimited fleet of homogeneous vehicles, and no multi-trips.
- **Note**: the routing problem to be solved is a vehicle routing problem with time windows (VPRTW).

Markov decision process

- Let ω_t denote the set of new delivery requests revealed in epoch t.

Markov decision process

- Let ω_t denote the set of new delivery requests revealed in epoch t.

- **State**: Define the *state* s_t as

$$s_t = \{i \in \mathcal{N}_t \mid \text{request } i \text{ not yet dispatched}\},$$

where $\mathcal{N}_t = \cup_{t'=1}^{t} \omega_{t'}$ denotes the set of known requests at epoch t.

Markov decision process

- Let ω_t denote the set of new delivery requests revealed in epoch t.

- **State**: Define the *state* s_t as

$$s_t = \{i \in \mathcal{N}_t \mid \text{request } i \text{ not yet dispatched}\},$$

where $\mathcal{N}_t = \cup_{t'=1}^{t} \omega_{t'}$ denotes the set of known requests at epoch t.

- **Action**: Define the *action* space $\mathcal{A}(s_t)$ as

$$\mathcal{A}(s_t) = \{a_t \subseteq s_t \mid m_t \subseteq a_t\},$$

where $m_t \subseteq s_t$ denotes the set of must-dispatch requests.

Markov decision process

- **Transition:** The transition to the next state s_{t+1} is given by

$$s_{t+1} = \left(s_t \setminus a_t\right) \cup \omega_{t+1}.$$

Markov decision process

- **Transition:** The transition to the next state s_{t+1} is given by

$$s_{t+1} = (s_t \setminus a_t) \cup \omega_{t+1}.$$

- **Optimality equation:** The objective of the DDWP is to select for each epoch $t \in \mathcal{T}$ a minimum cost action, such that the following Bellman optimality conditions are satisfied:

$$V(s_t) = \begin{cases} C(s_t, s_t) & \text{if } t = |\mathcal{T}|, \\ \min\limits_{a \in \mathcal{A}(s_t)} \left[C(s_t, a) + \mathbb{E}_{\omega_{t+1}} \left[V(s_{t+1}) \right] \right] & \text{otherwise.} \end{cases} \tag{1}$$

Markov decision process

- **Transition:** The transition to the next state s_{t+1} is given by

$$s_{t+1} = (s_t \setminus a_t) \cup \omega_{t+1}.$$

- **Optimality equation:** The objective of the DDWP is to select for each epoch $t \in \mathcal{T}$ a minimum cost action, such that the following Bellman optimality conditions are satisfied:

$$V(s_t) = \begin{cases} C(s_t, s_t) & \text{if } t = |\mathcal{T}|, \\ \min\limits_{a \in \mathcal{A}(s_t)} \left[C(s_t, a) + \mathbb{E}_{\omega_{t+1}} [V(s_{t+1})] \right] & \text{otherwise.} \end{cases} \tag{1}$$

- **Cost**: The direct cost $C(s_t, a_t)$ is given by the cost of routing all dispatched requests a_t, i.e., the optimal cost of the VRPTW.

Sample scenario methods

- **Main idea**: sample future request realizations, solve each realization, and derive a good action.

Sample scenario methods

- **Main idea**: sample future request realizations, solve each realization, and derive a good action.
- **Big drawback:** Computationally expensive to solve many, large, NP-hard VRPs!

Sample scenario methods

- **Main idea**: sample future request realizations, solve each realization, and derive a good action.
- **Big drawback:** Computationally expensive to solve many, large, NP-hard VRPs!
- What if we instead *incrementally* build an action to minimize computational challenges?

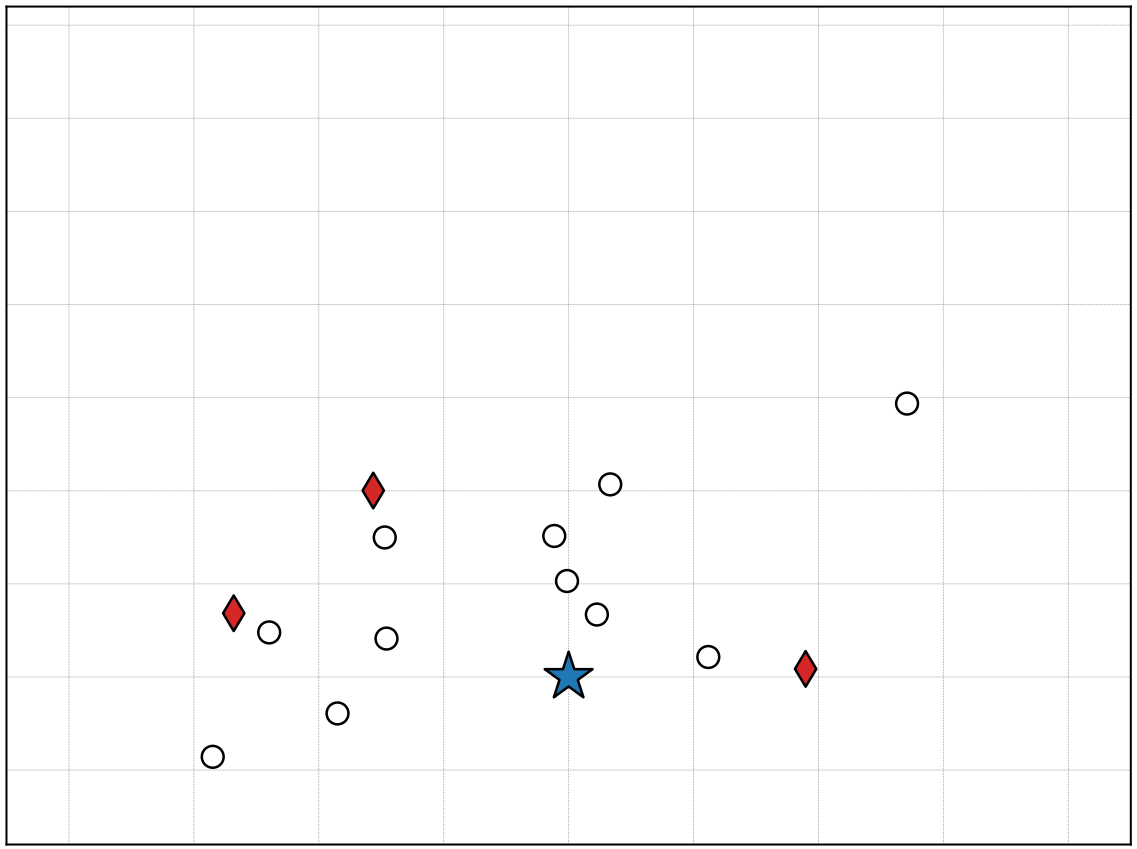
Sample scenario methods

- **Main idea**: sample future request realizations, solve each realization, and derive a good action.
- **Big drawback:** Computationally expensive to solve many, large, NP-hard VRPs!
- What if we instead *incrementally* build an action to minimize computational challenges?

Iterative conditional dispatch

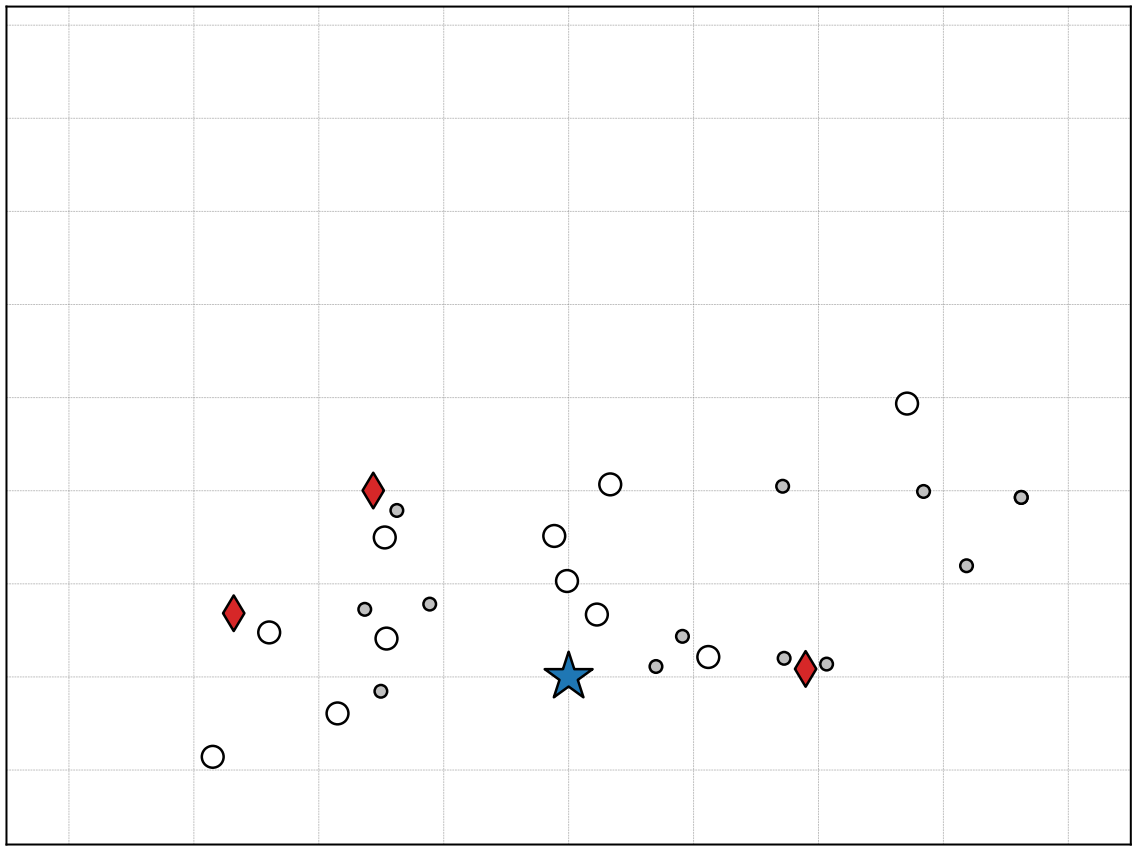
- Let d_t denote the set of dispatched requests and let p_t denote the set of postponed requests.
- **Initialize** $d_t = m_t$ and $p_t = \varnothing$.
- **Repeat** for a fixed number of iterations:
  - **Step one**: Solve a number of sample scenarios conditioned on d_t and p_t.
  - **Step two**: Classify undecided requests into either d_t, p_t, or leave undecided.
- **Return** action $a_t = d_t$.

Iterative conditional dispatch



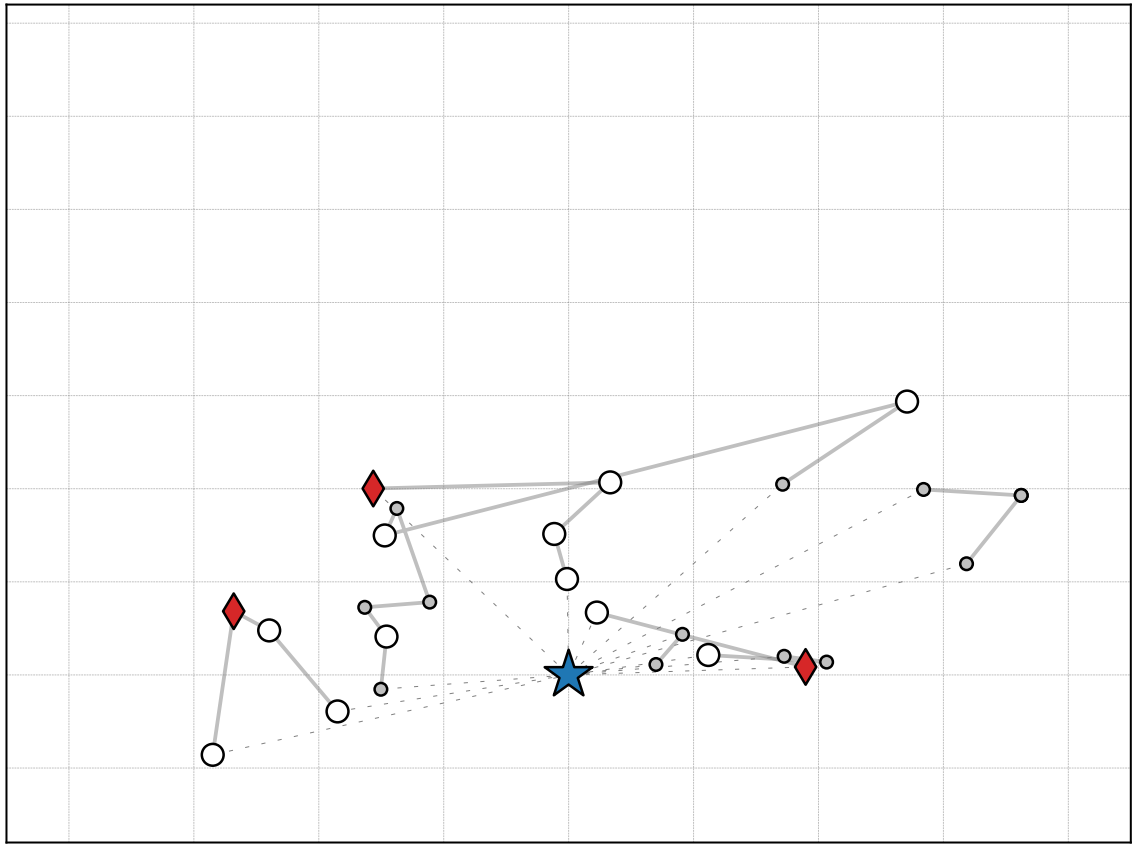
State s_t at some epoch t.

Iterative conditional dispatch



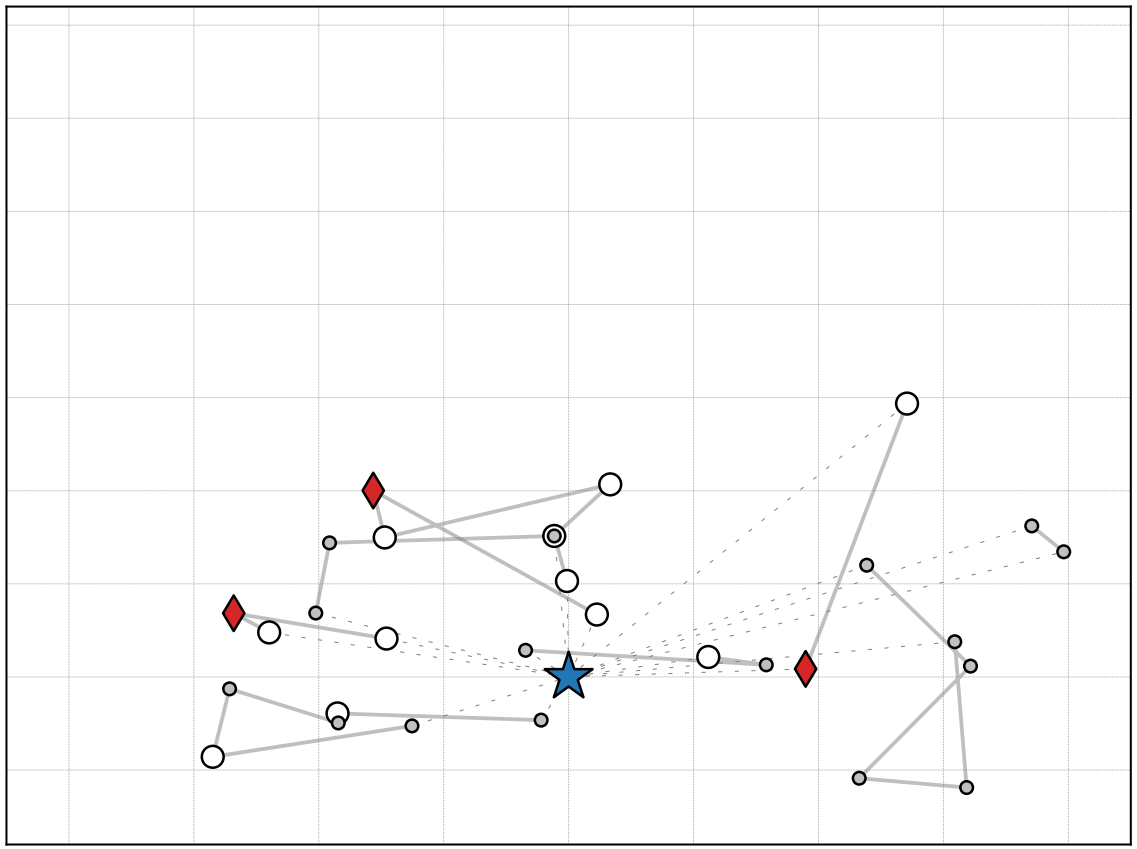
A sample scenario instance (VRPTW with release times).

Iterative conditional dispatch



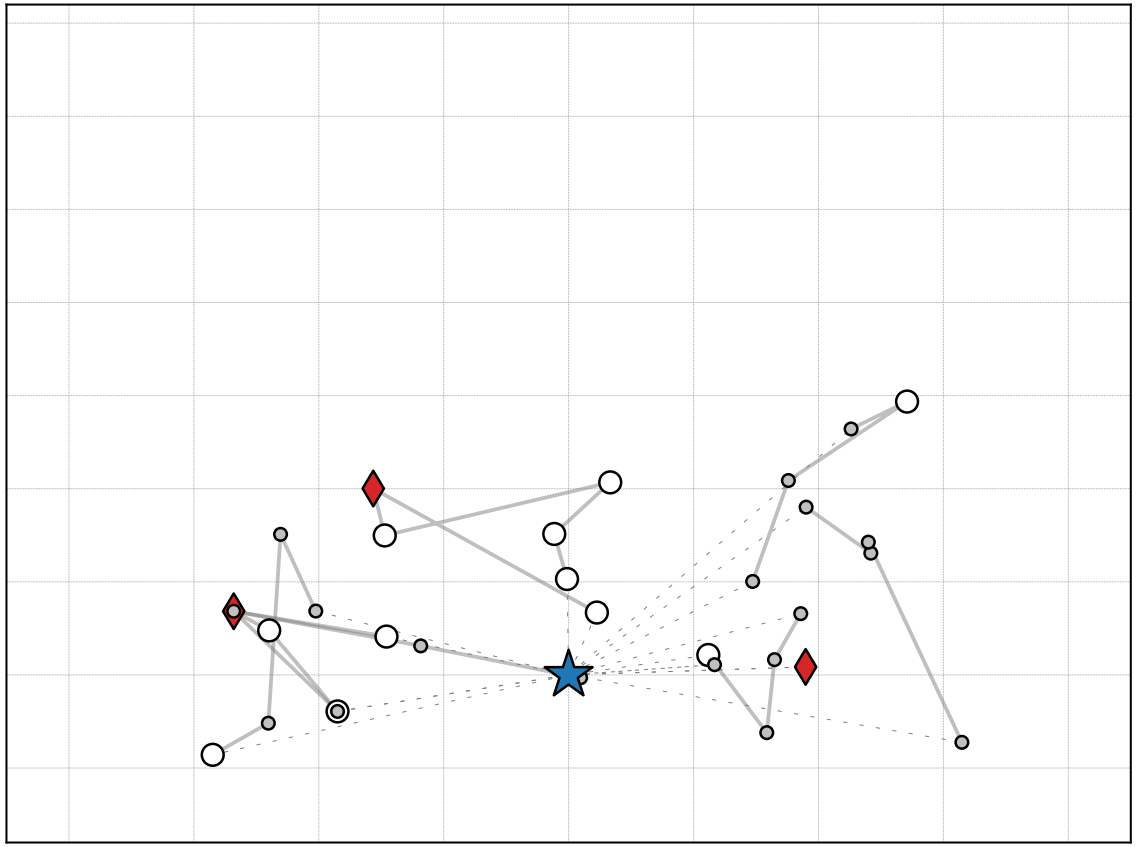
Solve the scenario instance.

Iterative conditional dispatch



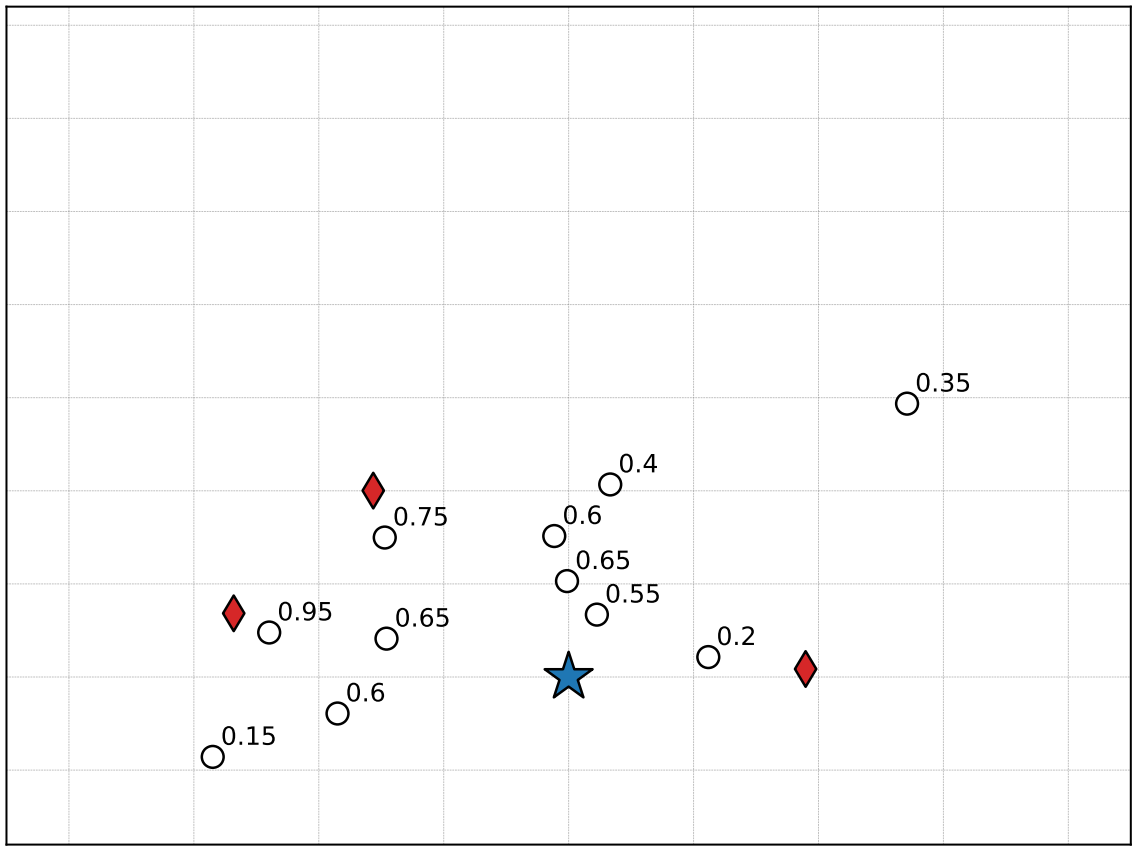
Solve another scenario instance.

Iterative conditional dispatch



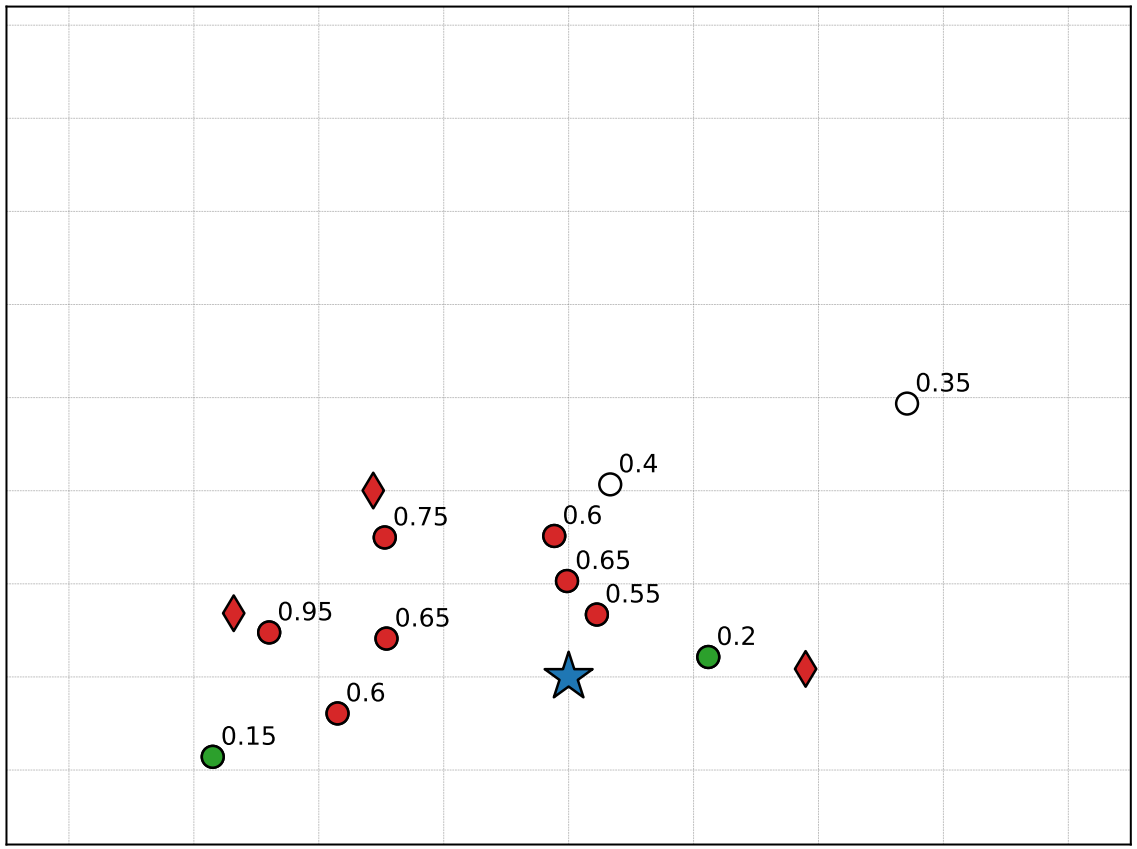
And another one... (30x in total)

Iterative conditional dispatch



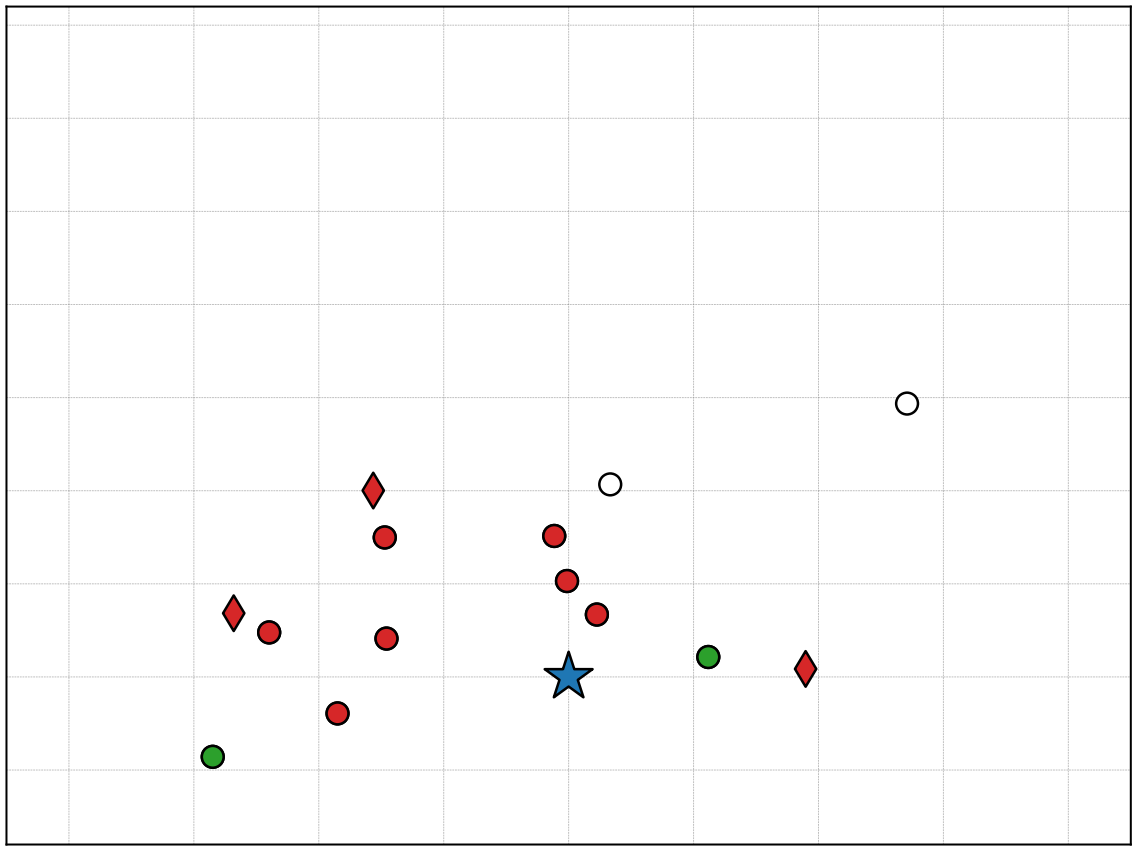
Compute the dispatch scores: how frequently was each request dispatched?

Iterative conditional dispatch



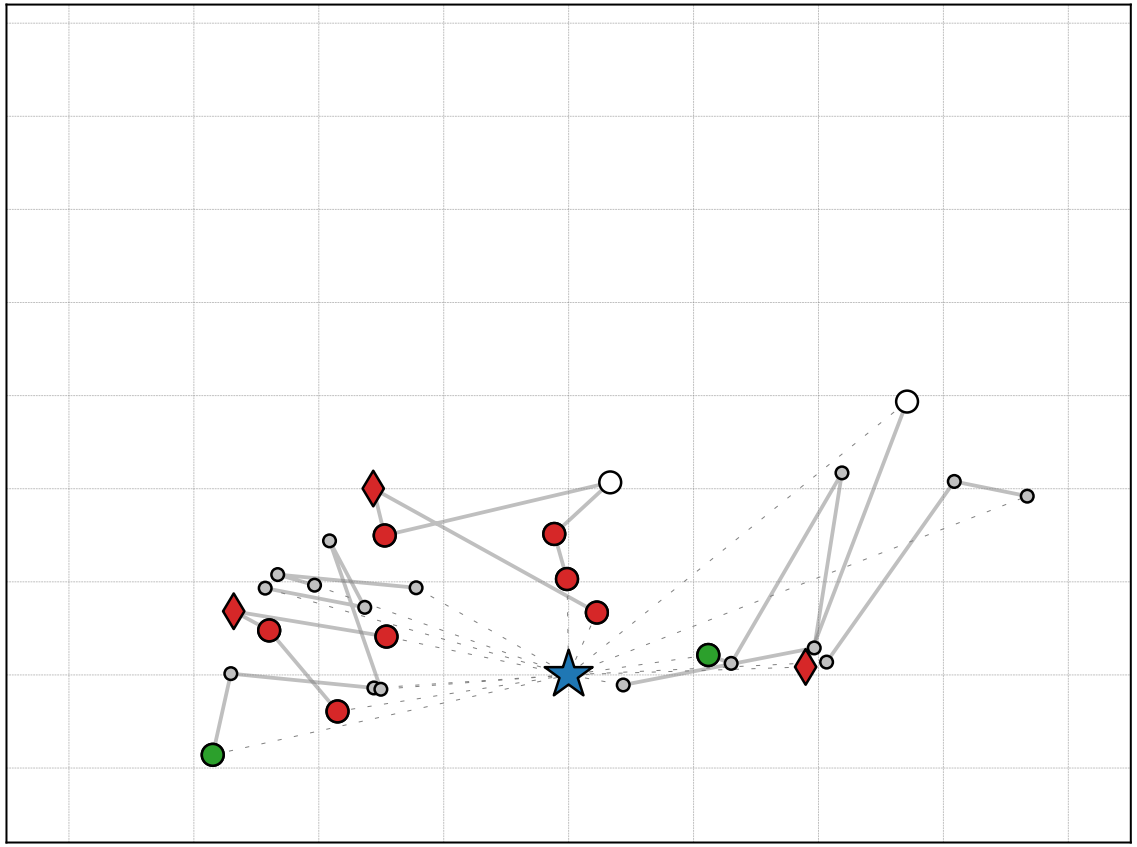
Dispatch if score ≥ 0.5, postpone if score ≤ 0.2.
Leave some undecided.

Iterative conditional dispatch



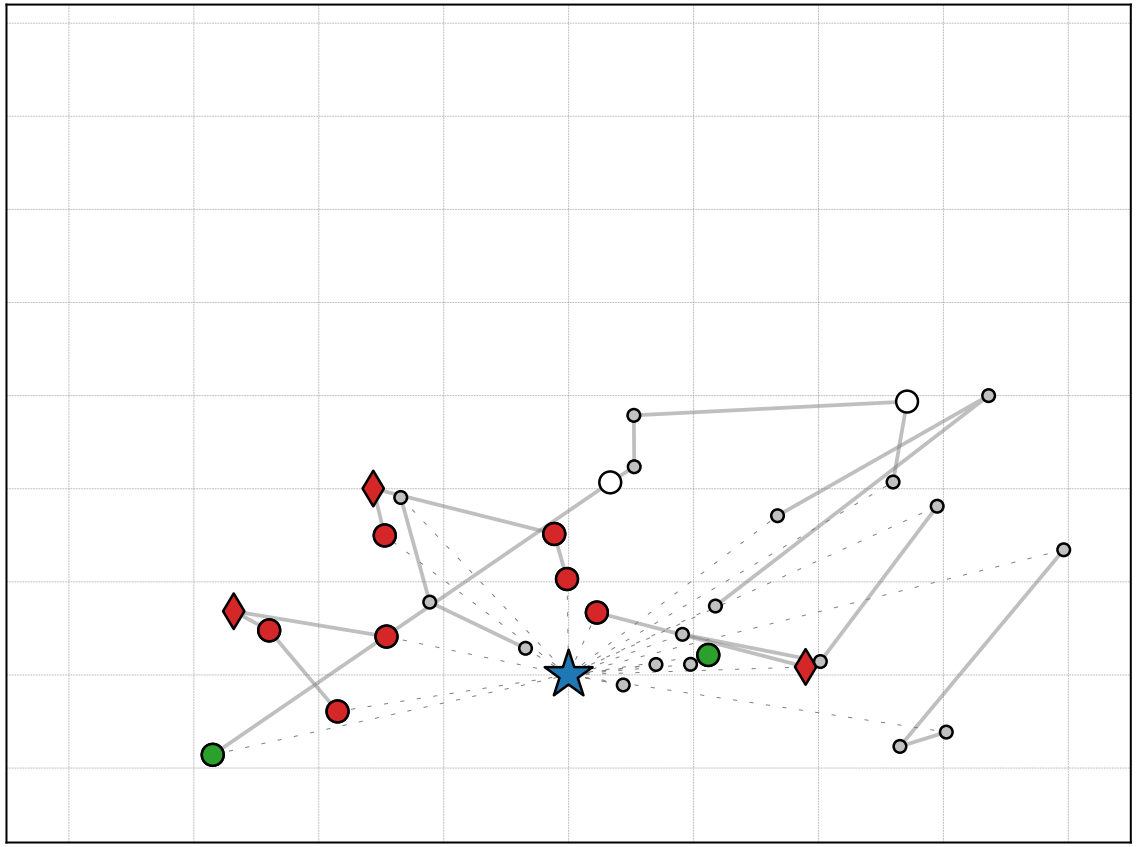
Rinse and repeat!

Iterative conditional dispatch



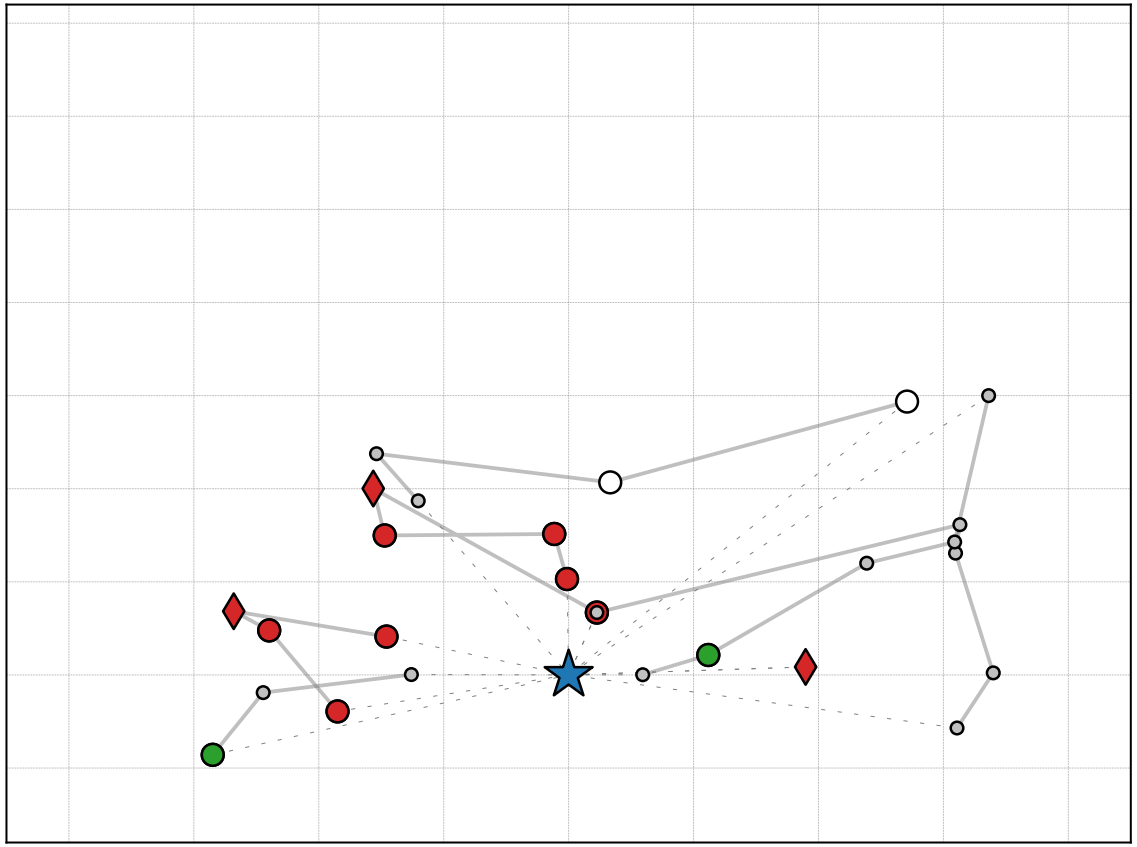
Rinse and repeat!

Iterative conditional dispatch



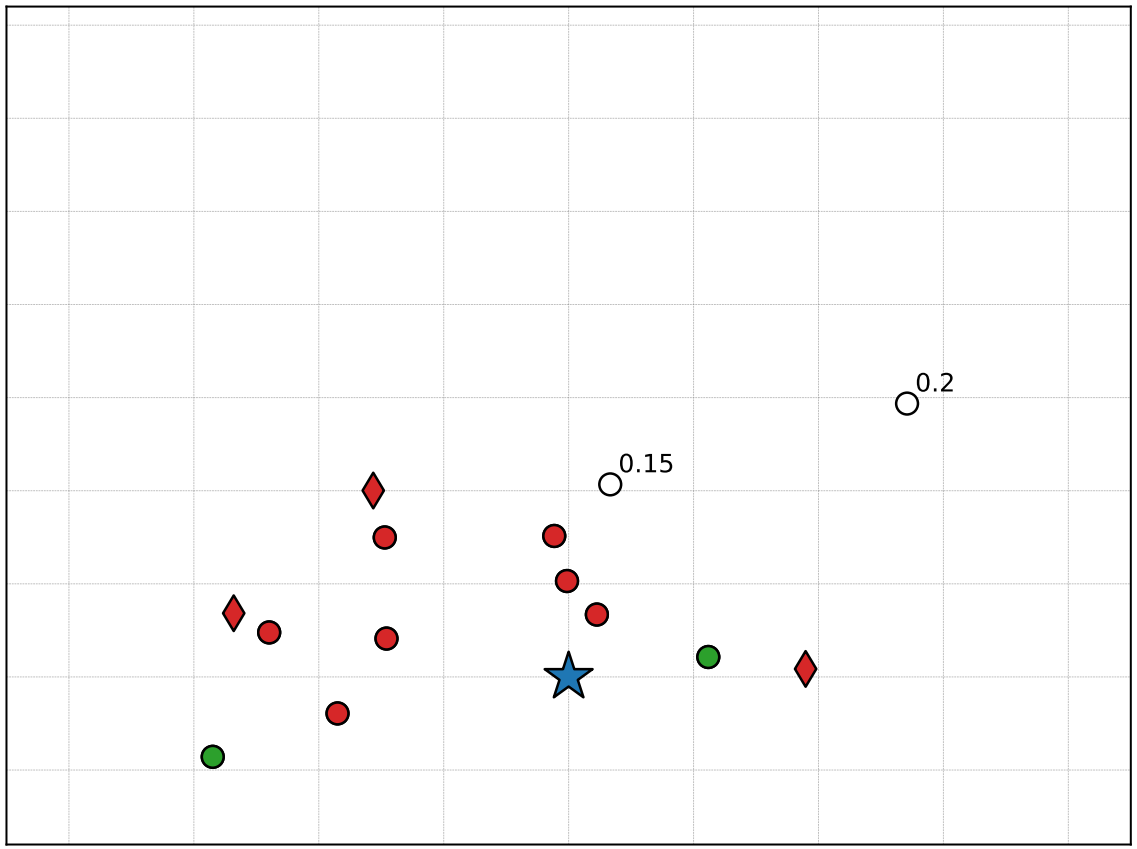
Rinse and repeat!

Iterative conditional dispatch



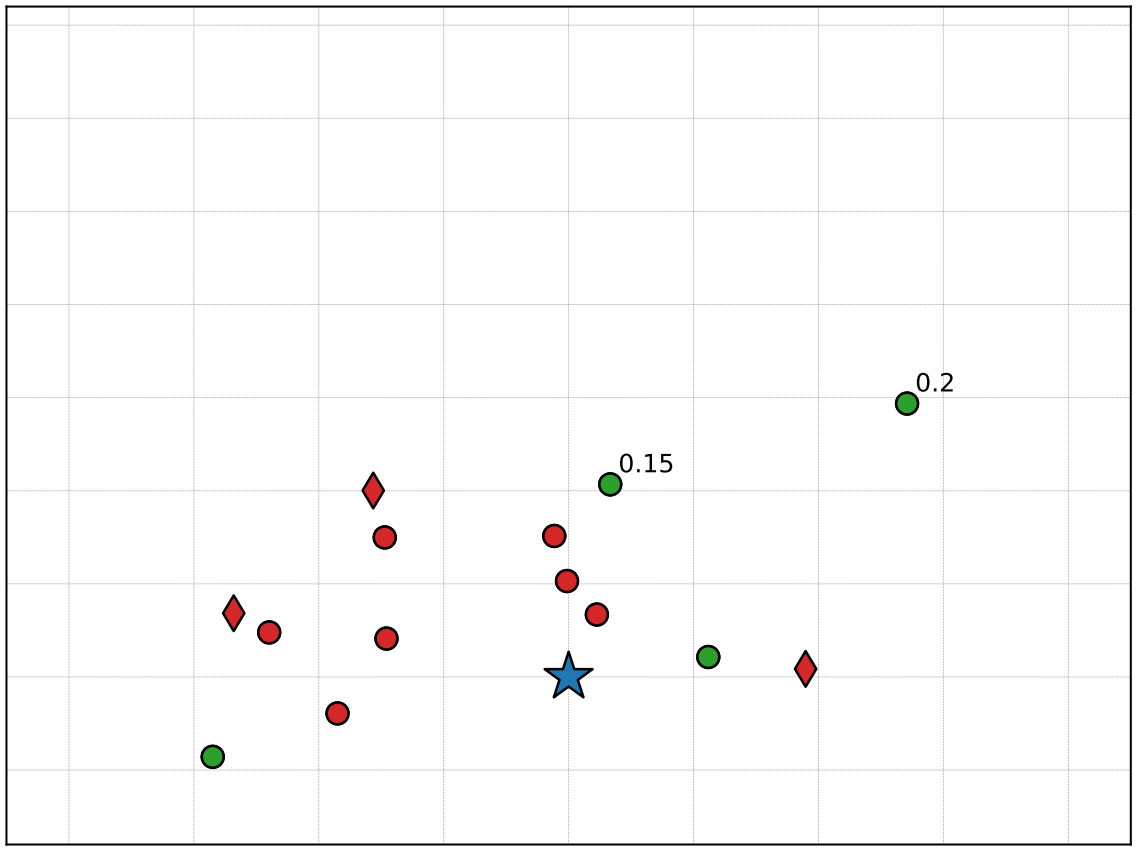
Rinse and repeat!

Iterative conditional dispatch



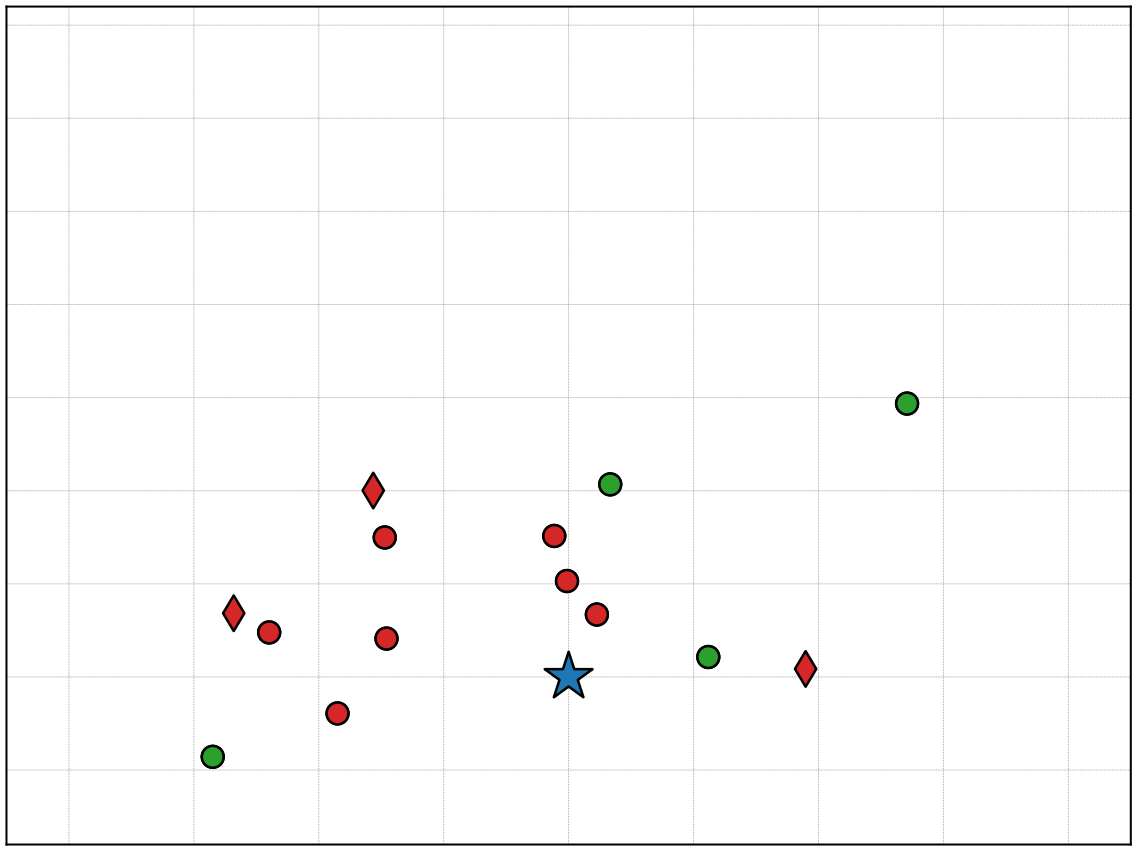
Rinse and repeat!

Iterative conditional dispatch



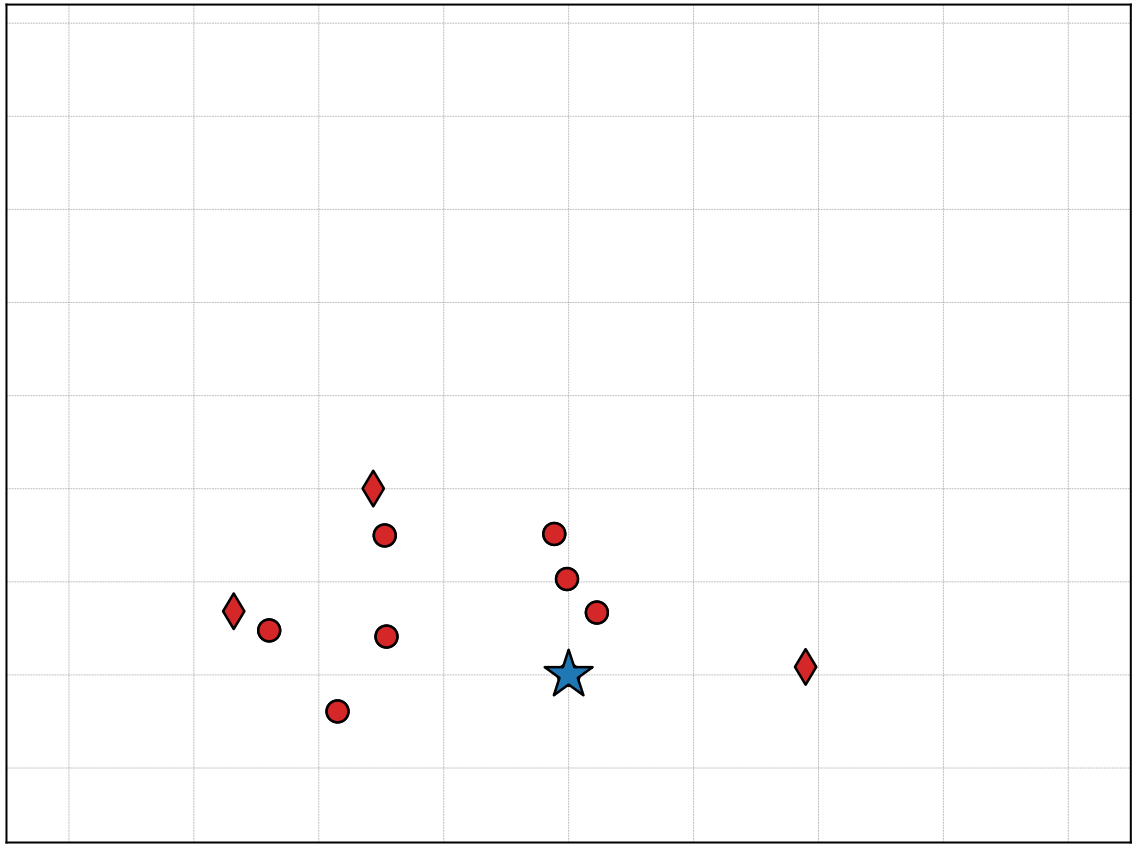
Rinse and repeat!

Iterative conditional dispatch



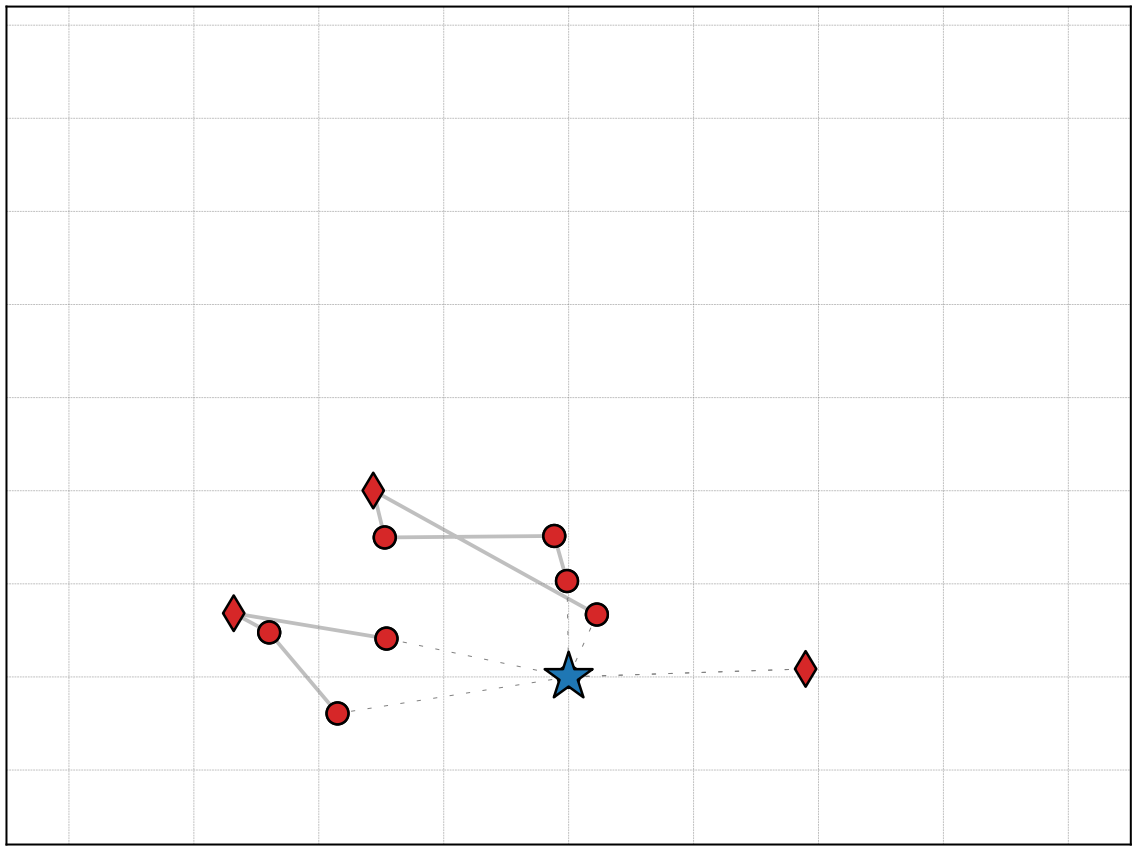
Rinse and repeat!

Iterative conditional dispatch



The requests assigned for dispatch.

Iterative conditional dispatch



Final routing solution.

Fixing decisions with dispatch windows

- **Key challenge**: how to *translate* fixed decisions (dispatched or postponed requests) to the static VRPTW?

Fixing decisions with dispatch windows

- **Key challenge**: how to *translate* fixed decisions (dispatched or postponed requests) to the static VRPTW?
- Define a request *dispatch window* $[r_i^-, r_i^+]$, where r_i^- denotes the *earliest* time that request i can be dispatched, and r_i^+ denotes the *latest* time that it can be dispatched.

Fixing decisions with dispatch windows

- For dispatched requests $i \in d_t$,

$$[r_i^-, r_i^+] = [T_t, T_t].$$

- For postponed requests $i \in p_t$,

$$[r_i^-, r_i^+] = [T_{t+1}, H].$$

- For undecided requests $i \in S \setminus (d_t \cup p_t)$,

$$[r_i^-, r_i^+] = [r_i, H],$$

where H denotes the planning horizon (end of service).

Vehicle routing problem with dispatch windows

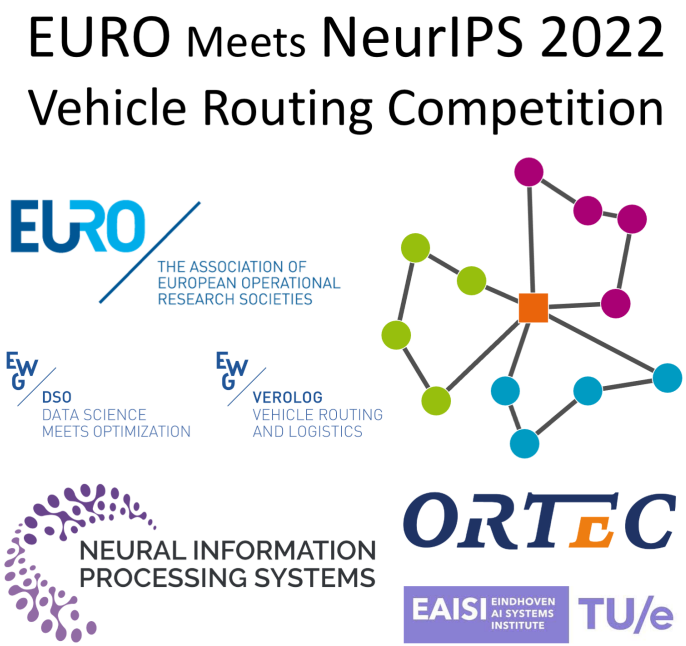
Instead of solving a VRPTW with release dates, we solve scenarios as VRPTW with dispatch windows (to condition on the dispatched requests d_t).

Route dispatch window feasibility

Consider a route R consisting of a sequence of requests and let θ_R denote its departure time. Then a route is dispatch window feasible if

$$\max_{i \in R} \{r_i^-\} \leq \theta_R \leq \min_{i \in R} \{r_i^+\}.$$

EURO meets NeurIPS 2022 vehicle routing competition



- Static and dynamic VRPTW track.
- Qualification: 1 July - 31 October.
- Daily updated leaderboard.
- Top 10 teams go to finals.

Instances

Top 10 teams are evaluated on 100 final instances.

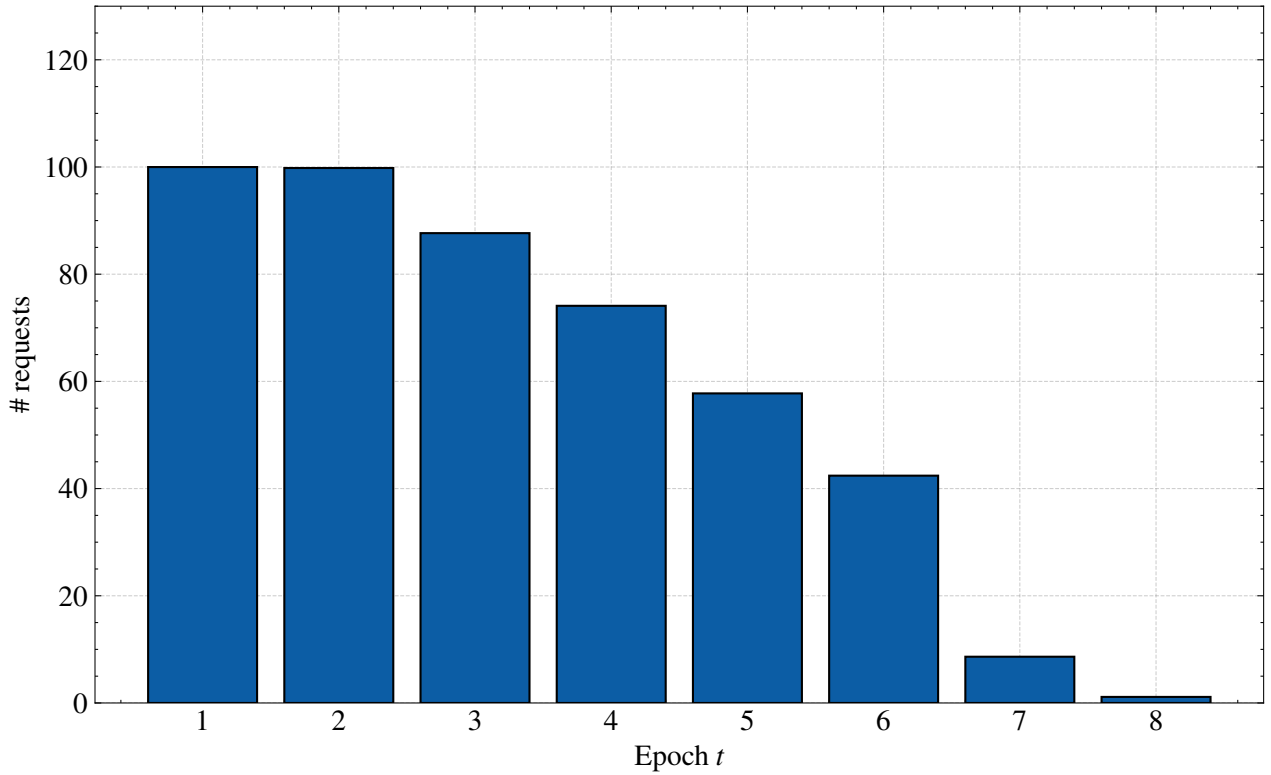
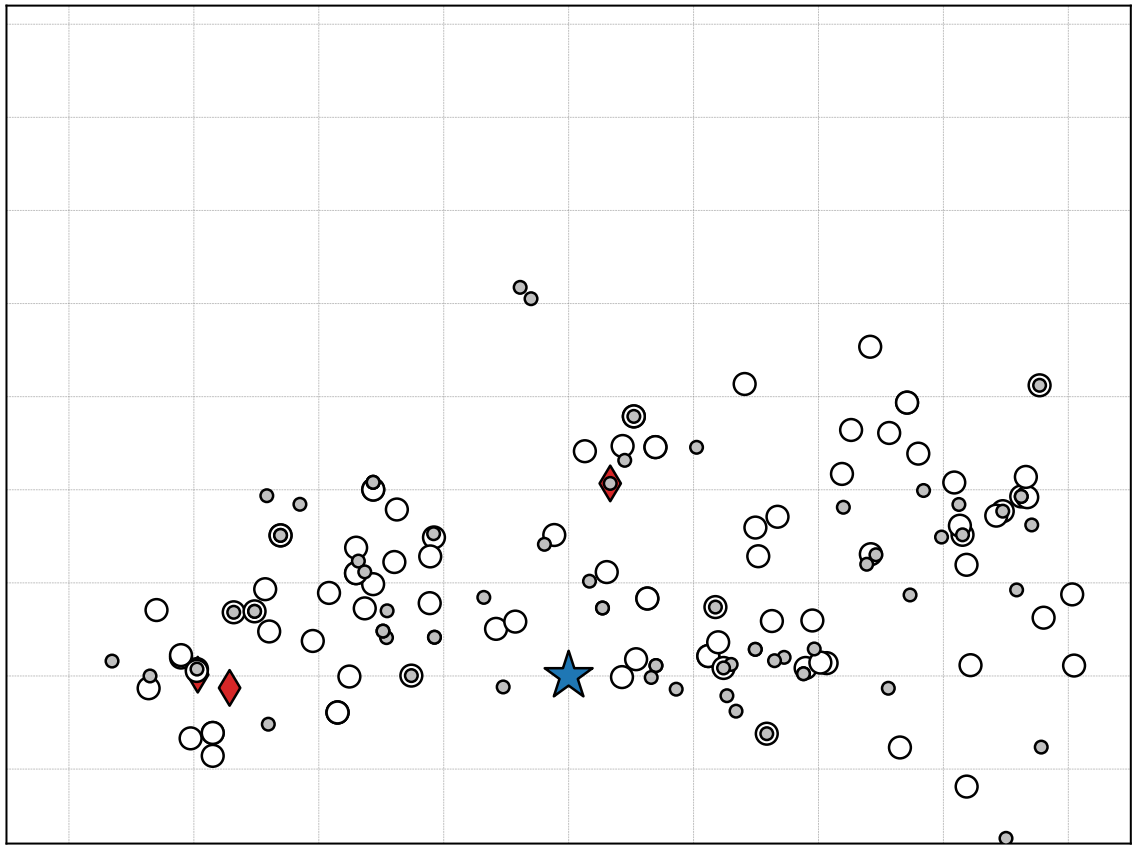


Figure: Average arrival process of a final's instance.

Just to give an impression...



Parameter settings

ICD parameters:

- 3 iterations, 30 scenarios, 50% dispatch, 20% postpone threshold.
- 90 seconds for scenarios $\rightarrow$ 1 second per scenario
- 30 seconds to compute the cost of dispatching action.

Computational details:

- Implementation runs on a single core, but can be ran in parallel trivially.
- Besides 120s time limit (competition time limit), we also evaluate 180s and 600s.

PyVRP

PyVRP 

- We use PyVRP[3] to solve all VRP instances.
- Built on top of the state-of-the-art solver.[4]

---

3 N. A. Wouda, L. Lan, and W. Kool (2023). "PyVRP: a high-performance VRP solver package". In: *INFORMS Journal on Computing*. accepted for publication. URL: https://github.com/PyVRP/PyVRP/

4 T. Vidal (2021). "Hybrid Genetic Search for the CVRP: Open-Source Implementation and SWAP* Neighborhood". In: *arXiv:2012.10384 [cs]*. arXiv: 2012.10384 [cs]. (Visited on 11/04/2021)

EURO-NeurIPS 2022 results

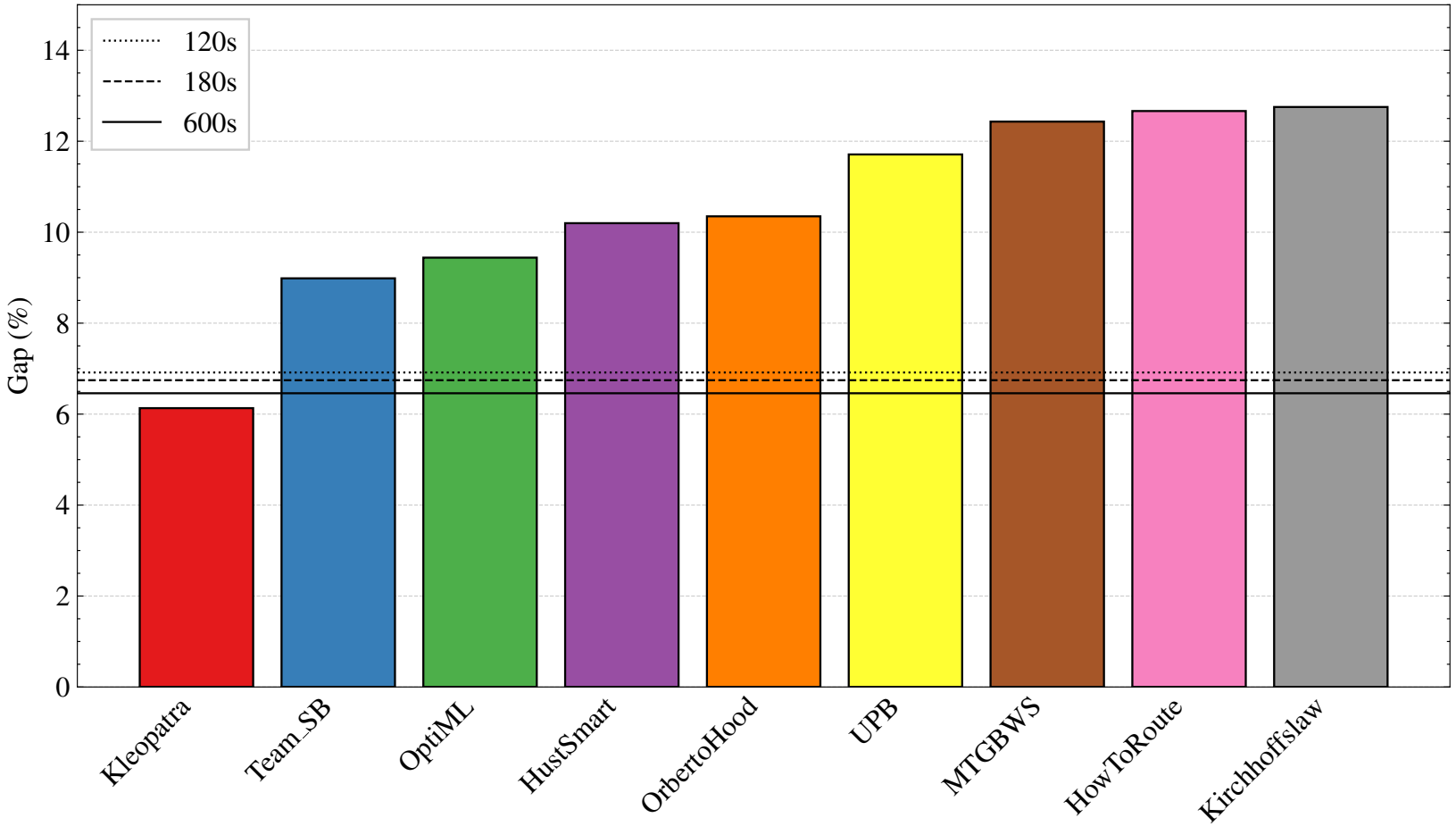


Figure: Results on EURO-NeurIPS final 100 instances. Dashed lines represent ICD with different time limits.

Conclusion

- We showed that ICD can achieve great performance on the DDWP, overcoming the large computational efforts often associated with sampling-based methods.
- ICD is conceptually simple and very easy to implement, and shows great promise for extension to other dynamic VRPs.

Thank you!

Code: https://github.com/leonlan/dynamic-dispatch-waves

Preprint: https://arxiv.org/abs/2308.14476

l.lan@vu.nl — leonlan — www.leonlan.com

Bibliography I

Kool, W., D. Numeroso, R. Reijnen, R. R. Afshar, T. Catshoek, K. Tierney, E. Uchoa, and J. Gromicho (2022). *EURO Meets NeurIPS 2022 Vehicle Routing Competition*.

Vidal, T. (2021). "Hybrid Genetic Search for the CVRP: Open-Source Implementation and SWAP* Neighborhood". In: *arXiv:2012.10384 [cs]*. arXiv: 2012.10384 [cs]. (Visited on 11/04/2021).

Wouda, N. A., L. Lan, and W. Kool (2023). "PyVRP: a high-performance VRP solver package". In: *INFORMS Journal on Computing*. accepted for publication. URL: https://github.com/PyVRP/PyVRP/.

Zhang, J. and T. V. Woensel (2023). "Dynamic Vehicle Routing with Random Requests: A Literature Review". en. In: *International Journal of Production Economics* 256, p. 108751.