Machine Learning and Combinatorial Optimization for the Dynamic Vehicle Routing Problem

Léo Baty[1], Kai Jungel[2], Patrick Klein[2], Maximilian Schiffer[2], Axel Parmentier[1]

[1]CERMICS, École des Ponts, [2]Technical University of Munich

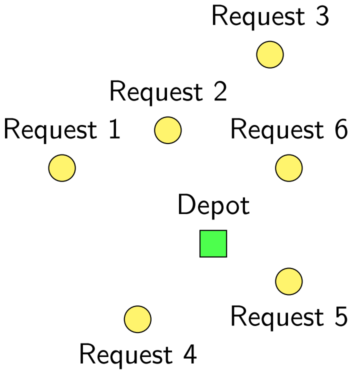
January 25, 2022

1. Problem statement
   - Static problem
   - Dynamic problem
2. Machine Learning pipeline
3. Learning approach
4. Results

Vehicle Routing Problem with Time Windows (VRPTW)

Depot: vehicles capacity Q

Requests $v \in V$

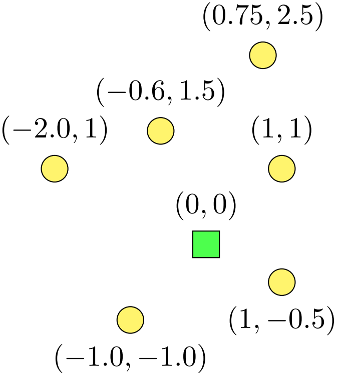


Vehicle Routing Problem with Time Windows (VRPTW)

Depot: vehicles capacity Q

Requests $v \in V$

1. Coordinates p
   $\Rightarrow$ costs $c_{v,v'}$

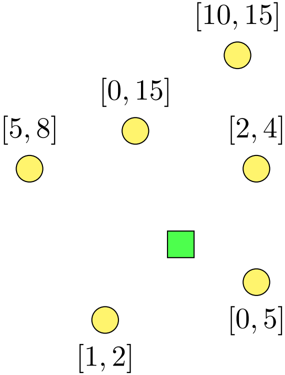


Vehicle Routing Problem with Time Windows (VRPTW)

Depot: vehicles capacity Q

Requests $v \in V$

1. Coordinates p
   $\Rightarrow$ costs $c_{v,v'}$
2. Time Windows $[\ell, u]$

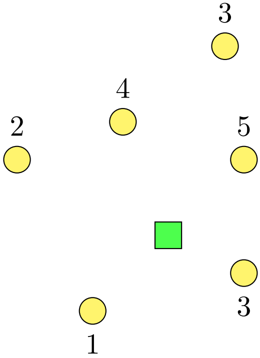


Vehicle Routing Problem with Time Windows (VRPTW)

Depot: vehicles capacity Q

Requests $v \in V$

1. Coordinates p
   $\Rightarrow$ costs $c_{v,v'}$
2. Time Windows $[\ell, u]$
3. Demand q

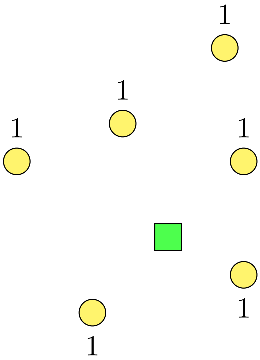


Vehicle Routing Problem with Time Windows (VRPTW)

Depot: vehicles capacity Q

Requests $v \in V$

1. Coordinates p
   $\Rightarrow$ costs $c_{v,v'}$
2. Time Windows $[\ell, u]$
3. Demand q
4. Service time s

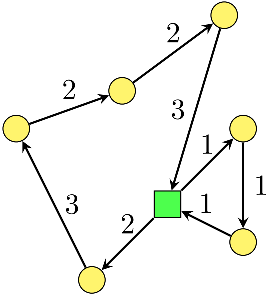


Vehicle Routing Problem with Time Windows (VRPTW)

Depot: vehicles capacity Q

Requests $v \in V$

1. Coordinates p
   $\Rightarrow$ costs $c_{v,v'}$
2. Time Windows $[\ell, u]$
3. Demand q
4. Service time s



Objective: build feasible routes serving all requests at minimum cost

State-of-the-art algorithm: Hybrid Genetic Search (HGS)

- Genetic algorithm
- Maintains a population of solutions
- Improves it over the iterations using crossover combined with neighborhood searches

See [Vidal, 2021] for details.

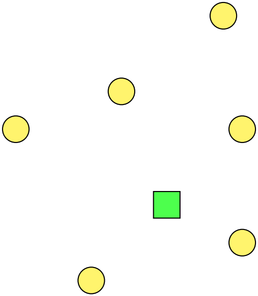
1. Problem statement
   - Static problem
   - Dynamic problem
2. Machine Learning pipeline
3. Learning approach
4. Results

Dynamic VRPTW

- Same setting as the static problem
- Discrete time horizon $[T]$, 1-hour **epochs**
- New requests arrive at the start of each epoch
  $\implies$ future requests are not known in advance

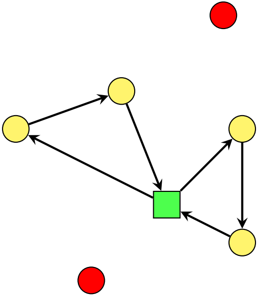
Dynamic VRPTW: example

Start of epoch 1: requests arrive



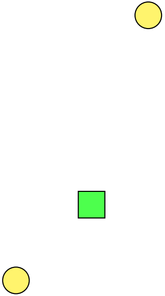
Dynamic VRPTW: example

- Decide which request to **dispatch**
- Build routes serving them, other requests are **postponed**
- Each request must be served before end of its time window
  $\Rightarrow$ some requests must be dispatched



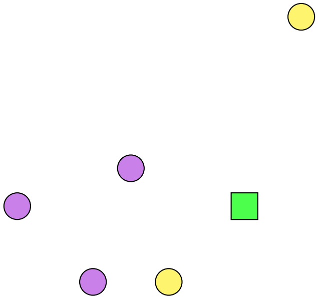
Dynamic VRPTW: example

- Decide which request to **dispatch**
- Build routes serving them, other requests are **postponed**
- Each request must be served before end of its time window
  $\Rightarrow$ some requests must be dispatched



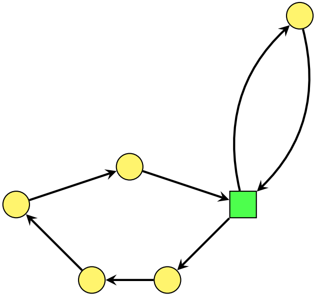
Dynamic VRPTW: example

- Start of epoch 2: new requests arrive



Dynamic VRPTW: example

- Start of epoch 2: new requests arrive
- Epoch 2 routes



Summary

- At every epoch t:
  - Decide which request to **dispatch**
  - Build routes serving them, other requests are **postponed**
  - Each request must be served before end of its time window
    $\Rightarrow$ some requests must be dispatched
- **State** x_t of the system at epoch t: set of requests arrived at t or arrived before but not yet served
- **Objective**: serve all requests, minimize total travel distance

$\Rightarrow$ no state-of-the-art

Policy based on a Deep Learning pipeline

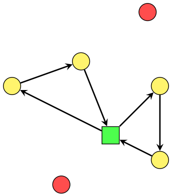
Epoch decisions can be seen as the solution of a Prize Collecting VRPTW:

- Serving requests is optional
- Serving request v gives **prize** θ_v
- **Objective**: maximize total profit minus costs

$$\max_{y \in \mathcal{Y}(x_t)} \sum_{(u,v) \in x_t^2} (\theta_v - c_{u,v}) y_{u,v}.$$

- **Algorithm**: Prize Collecting Hybrid Genetic Search

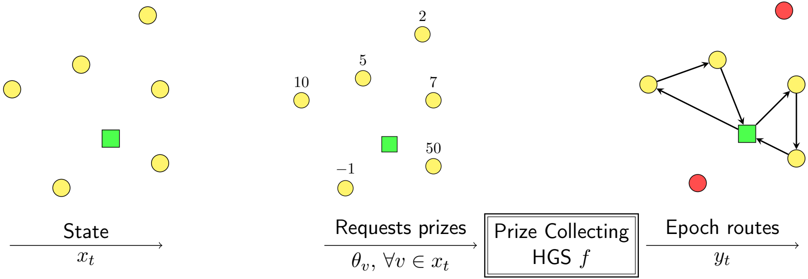
  $\Longrightarrow$ Combinatorial Optimization layer



$\xrightarrow[y_t]{\text{Epoch routes}}$

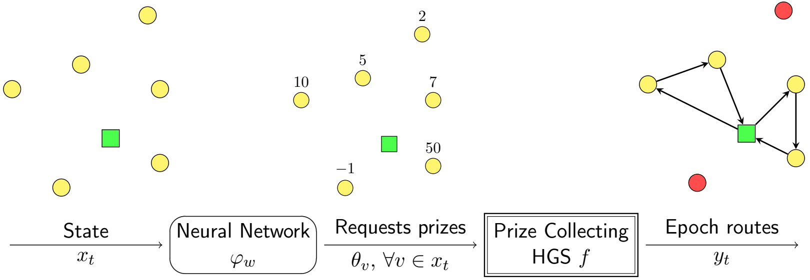
Policy based on a Deep Learning pipeline

Problem: we have no way of computing meaningful prizes



Policy based on a Deep Learning pipeline

Solution: use a neural network to predict request prizes θ_v



Goal: find parameters w such that our pipeline is a "good" policy.

1. Problem statement
   - Static problem
   - Dynamic problem
2. Machine Learning pipeline
3. Learning approach
4. Results

Learn to imitate an anticipative policy

Anticipative policy

- At $t = 0$, we assume that we know all future requests.
- Optimal solution by solving a VRPTW with release times

$\implies$ Hybrid Genetic Search

Learn to imitate an anticipative policy

Anticipative policy

- At $t = 0$, we assume that we know all future requests.
- Optimal solution by solving a VRPTW with release times

$\Longrightarrow$ Hybrid Genetic Search

Dataset labeled with anticipative decisions:

$$\mathcal{D} = \{(x^1, \overline{y}^1), \ldots, (x^n, \overline{y}^n)\}$$

Learn to imitate an anticipative policy

Anticipative policy

- At $t = 0$, we assume that we know all future requests.
- Optimal solution by solving a VRPTW with release times

$\Longrightarrow$ Hybrid Genetic Search

Dataset labeled with anticipative decisions:

$$\mathcal{D} = \{(x^1, \overline{y}^1), \ldots, (x^n, \overline{y}^n)\}$$

Can we apply classical supervised learning techniques?

A natural loss function

Combinatorial Optimization (CO) layer:

$$f : \theta \longmapsto \underset{y \in \mathcal{Y}(x_t)}{\operatorname{argmax}} \sum_{(u,v) \in x_t^2} (\theta_v - c_{u,v}) y_{u,v}.$$

$$f : \theta \longmapsto \underset{y \in \mathcal{Y}(x_t)}{\operatorname{argmax}} \, \theta^\top g(y) + h(y)$$

with $g(y) = \left(\sum_{u \in V} y_{u,v}\right)_{v \in V}$, and $h(y) = -\sum_{(u,v) \in x_t^2} c_{u,v} y_{u,v}$

A natural loss function

$$f \colon \theta \longmapsto \underset{y \in \mathcal{Y}(x_t)}{\operatorname{argmax}} \theta^\top g(y) + h(y)$$

When we apply Automatic Differentiation (AD) to a CO oracle:

- It usually doesn't work (lack of compatibility with solver)
- Even when it does, the Jacobian is either zero or undefined

A natural loss function

$$f : \theta \longmapsto \underset{y \in \mathcal{Y}(x_t)}{\operatorname{argmax}} \, \theta^\top g(y) + h(y)$$

When we apply Automatic Differentiation (AD) to a CO oracle:

- It usually doesn't work (lack of compatibility with solver)
- Even when it does, the Jacobian is either zero or undefined

Natural loss function

Non-optimality of target routes $\overline{y}$ as a solution of f

$$\mathcal{L}(\theta, \overline{y}) = \max_{y \in \mathcal{Y}(x)} \{\theta^\top g(y) + h(y)\} - (\theta^\top g(\overline{y}) + h(\overline{y}))$$

Problem: $\theta = 0$ is a trivial solution that minimizes $\mathcal{L}(\cdot, \overline{y})$

Regularization through perturbation

Perturb the objective with an additive noise [Berthet et al., 2020]:

$$\widehat{f}_\varepsilon \colon \theta \mapsto \mathbb{E}\left[\operatorname*{argmax}_{y\in\mathcal{Y}(x_t)} (\theta + \varepsilon Z)^\top g(y) + h(y)\right] = \mathbb{E}[f(\theta + \varepsilon Z)]$$

with $Z \sim \mathcal{N}(0, 1)$, and $\varepsilon \in \mathbb{R}_+$.

Intractable expectation $\Rightarrow$ Monte-Carlo sampling approximation

Fenchel-Young loss

Perturbed Fenchel-Young loss

$$\mathcal{L}_\varepsilon(\theta, \overline{y}) = \mathbb{E}\left[\max_{y \in \mathcal{Y}(x_t)} (\theta + \varepsilon Z)^\top g(y) + h(y)\right] - (\theta^\top g(\overline{y}) + h(\overline{y})),$$

$$g(\widehat{f}_\varepsilon(\theta)) - g(\overline{y}) \in \partial_\theta \mathcal{L}_\varepsilon(\theta, \overline{y}).$$

Learning problem:

$$\operatorname*{argmin}_w \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_\varepsilon(\varphi_w(x^i), \overline{y}^i)$$

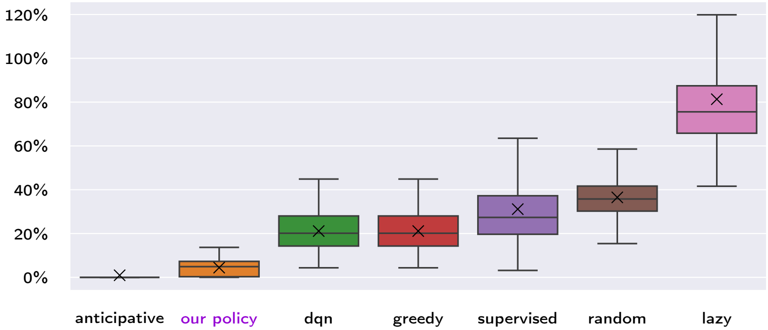
How to implement this pipeline ?

Our package `InferOpt.jl` [Dalle et al., 2022], written in Julia:

- Open source: `https://github.com/axelparmentier/InferOpt.jl`
- Easy to use
- Works with any CO oracle, independent of the implementation
- Compatible with Julia ML and AD ecosystem (through `ChainRules.jl`)

Results: 4.4% average gap

Benchmark on 2252 instances-seed combinations:



Winner team of Euro-NeurIPS competition

Rank	Team name	Dynamic cost	Static rank	Dynamic rank
1	Kléopatra	348831.56	2	1
2	OptiML	359270.09	1	3
4	Team_SB	358161.36	3	2
3	HustSmart	361803.57	5	4
5	Miles To Go Before We Sleep	369098.13	4	7

References

Berthet, Q., Blondel, M., Teboul, O., Cuturi, M., Vert, J.-P., and Bach, F. (2020).
Learning with Differentiable Perturbed Optimizers.
arXiv:2002.08676 [cs, math, stat].

Dalle, G., Baty, L., Bouvier, L., and Parmentier, A. (2022).
Learning with Combinatorial Optimization Layers: A Probabilistic Approach.

Vidal, T. (2021).
Hybrid Genetic Search for the CVRP: Open-Source Implementation and SWAP* Neighborhood.