Learning from user in constraint solving

Tias Guns <*tias.guns@kuleuven.be*>

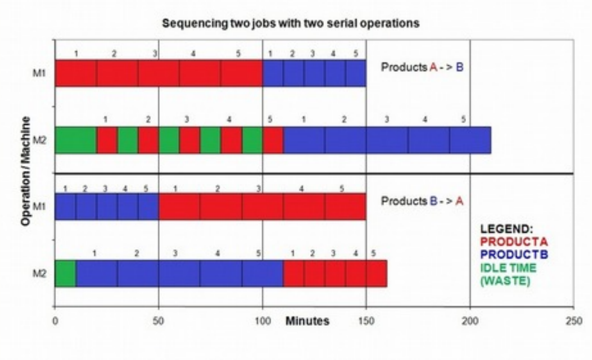
Joint work with:

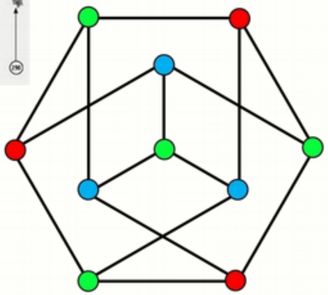
- Rocs Canoy
- Jayanta Mandi
- Victor Bucarey Lopez

Combinatorial optimisation

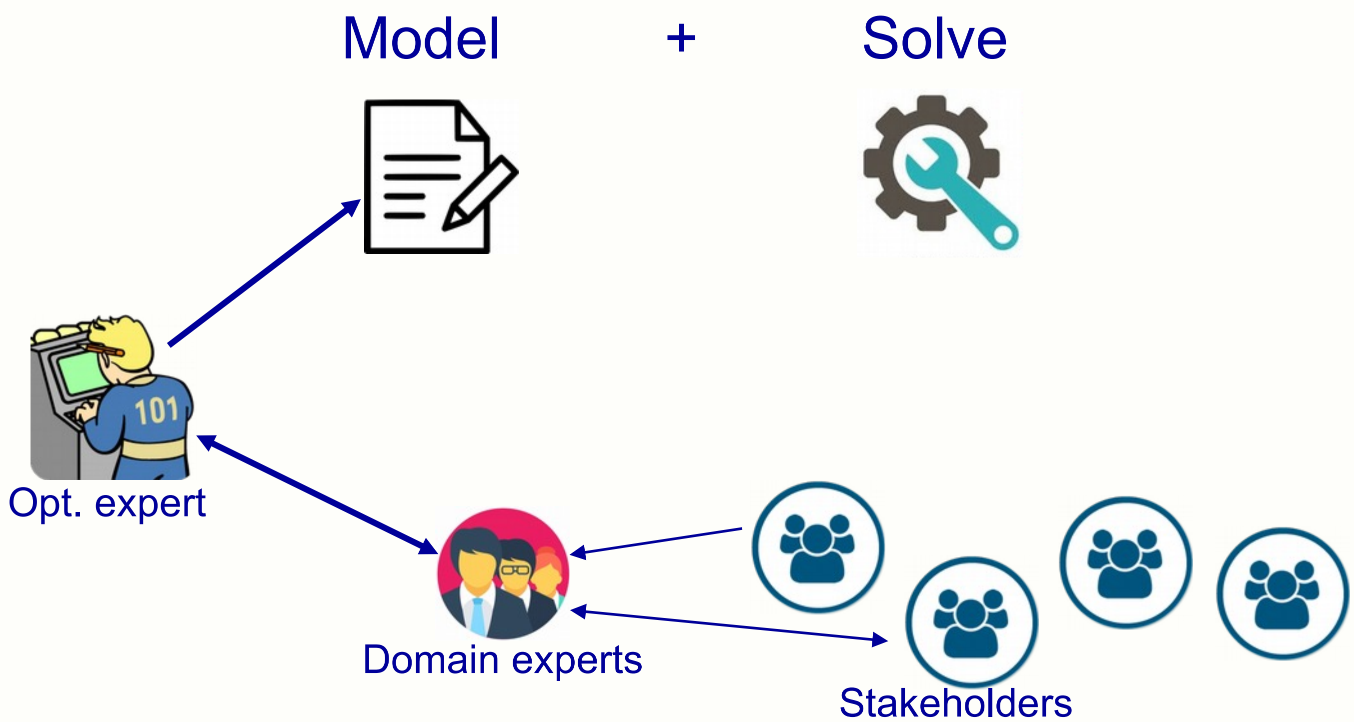
"Solving *constrained* optimisation problems"

- Vehicle Routing
- Scheduling
- Configuration
- Graph problems

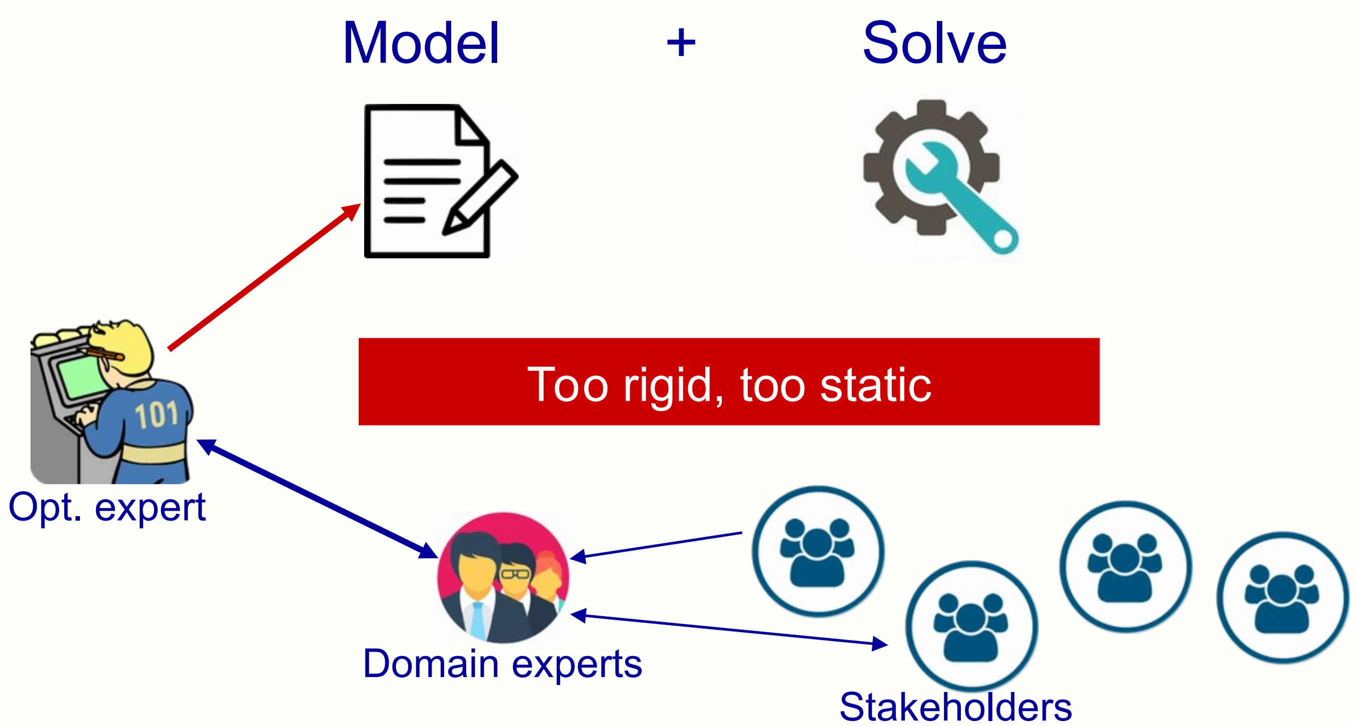




Current constraint solving practice



Current constraint solving practice, problem



Research trend

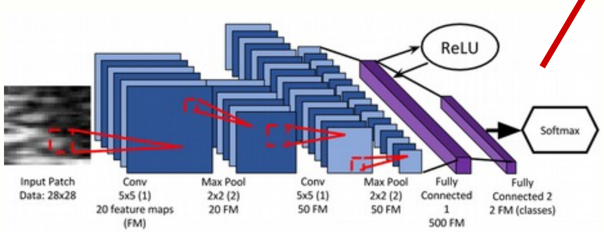
Model + Solve

1) learn the constraints

2) learn the objective function

3) learn to predict the solution

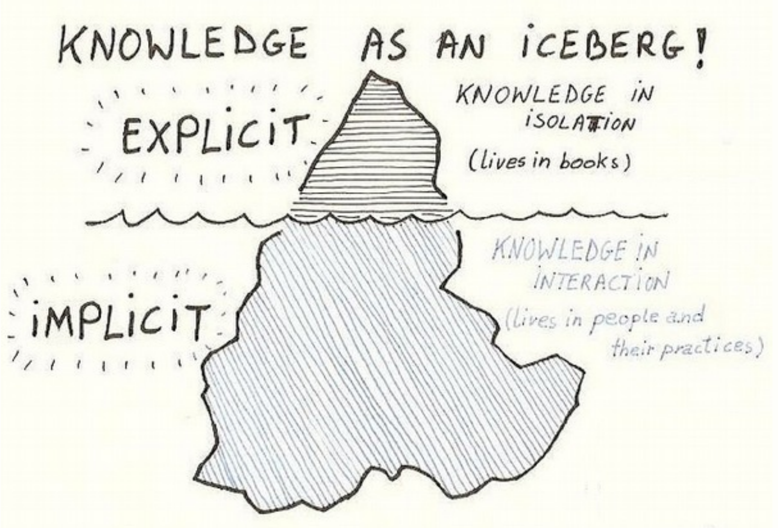
Can we *learn* it instead?



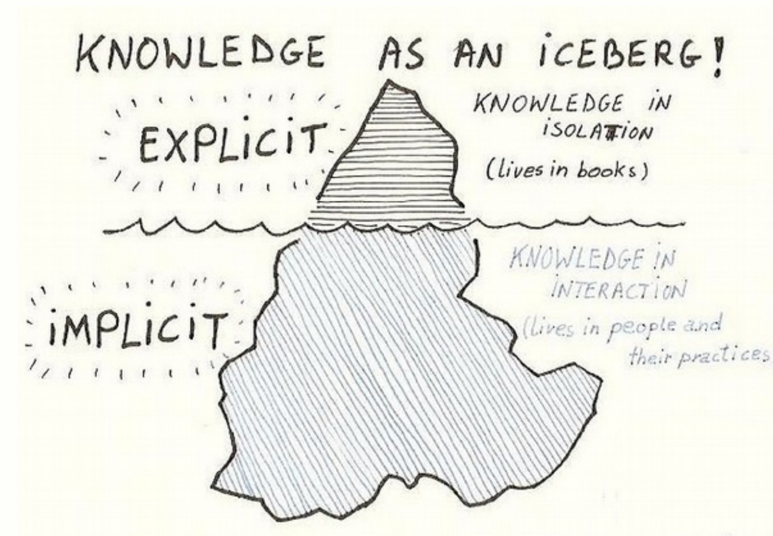
Prediction + constraint solving

- Part <u>explicit</u> knowledge:
  in a formal language

- Part <u>implicit</u> knowledge:
  learned from data



Prediction + constraint solving



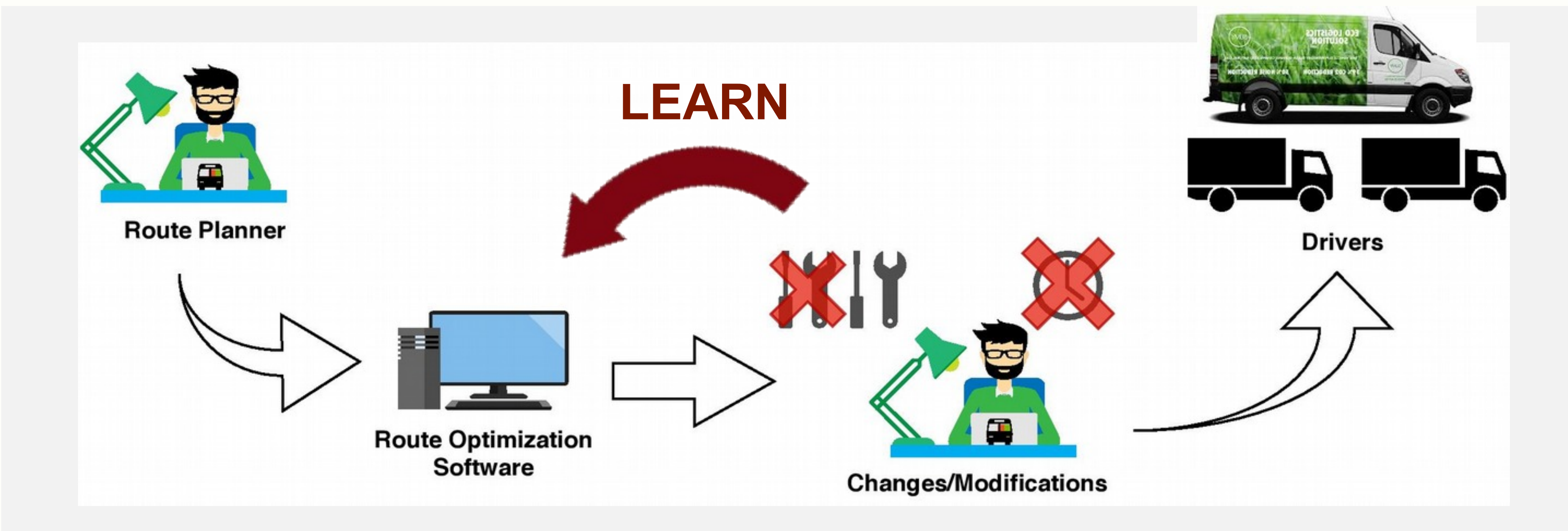
- Part explicit knowledge:
  in a formal language

- Part implicit knowledge:
  learned from data
  - **tacit knowledge *(user preferences, social conventions)***
  - complex environment *(demand, prices, defects)*
  - perception *(vision, natural language, audio)*

Tacit knowledge (user preferences)

"Vehicle routing by learning from historical solutions"

[Rocsildes Canoy and Tias Guns, CP19], **Best student paper award**



GOAL: Learn preferences, reduce manual effort, adapt to changes over time!

Tacit knowledge (user preferences)

Small data: 6 months = 26 weeks = 130 week days (instances)

Tacit knowledge (user preferences)

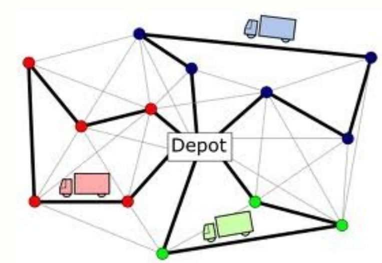
For <u>single</u> vehicles, in mobility mining literature:

- Driver turn prediction *[Krumm, 2008]*
- Prediction of remainder of route early in the trip *[Ye et al., 2015]*
- Prediction of route given origin and destination *[Wang et al., 2015]*

Can we use similar techniques (Markov Models)

to learn <u>preferences across routings</u> of multiple vehicles?

And can we <u>optimize</u> over them with constraint solving?



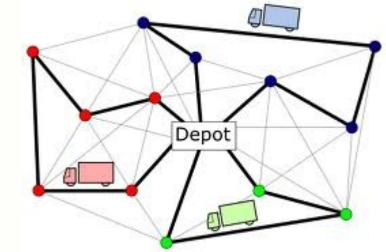
Learning and prediction part

<u>Key idea:</u>

if we can capture the 'preferences' in a probabilistic model

then we can evaluate the *likelihood* of a routing

P([v1_stop1,v1_stop2,...],[v2_stop1, v2_stop2, …], …)



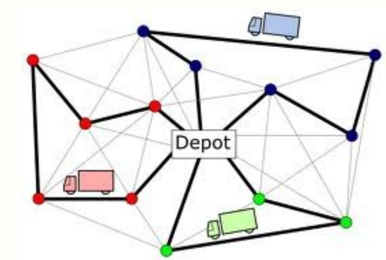
Learning and prediction part

<u>Key idea:</u>

if we can capture the 'preferences' in a probabilistic model

then we can evaluate the *likelihood* of a routing

P([v1_stop1,v1_stop2,...],[v2_stop1, v2_stop2, …], …)



One route is a chain of stops → treat as Markov Chain

1) for convenience, daisy-chain all routes into one

$P(<s1,s2,s3,s4,...>) = P(s1)*P(s2|s1)*P(s3|s2,s1)*P(s4|s3,s2,s1)*...$

1) a *1st order* approximation: P([s1,s2,s3,...) = P(s1)*P(s2|s1)*P(s3|s2)*...
(depend only on previous stop)

Learning and prediction part

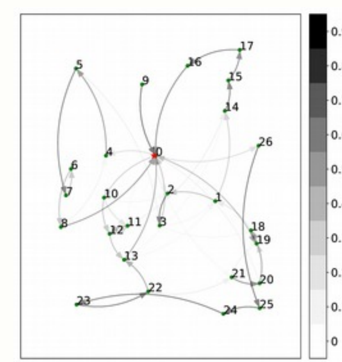
1st order Markov Model:

P([s1,s2,s3,...) = P(s1)*P(s2|s1)*P(s3|s2)*...

→ estimate the $P(s_y|s_x)$ by observing the transitions in the actually driven routes

probability of transition = relative nr of observations in the data

$$t_{ij} = \frac{f_{ij} + \alpha}{N_i + \alpha d},$$



Constrained optimisation: what now?

Goal: find maximum likelihood solution:

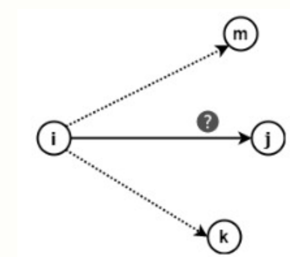
maximize P([s1,s2,s3,...) = P(s1)*P(s2|s1)*P(s3|s2)*...

s.t. VRP([s1,s2,s3,...])

Standard probability computation trick: log-likelihood

$$\max \prod_{(i,j)\in X} \mathbf{Pr}(\text{next stop}=j \mid \text{current stop}=i),$$

$$= \quad \max \sum_{(i,j)\in A} log(t_{ij})x_{ij}.$$



→ *VRP:* replace *distance matrix* by *negative log-likelihood* matrix!

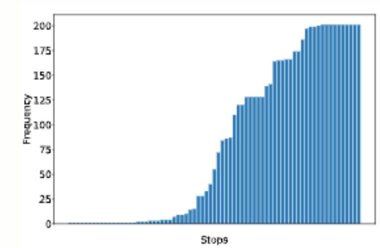
$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \quad \Longrightarrow \quad \min \sum_{(i,j)\in A} -log(t_{ij})x_{ij}.$$

Back to the learning...

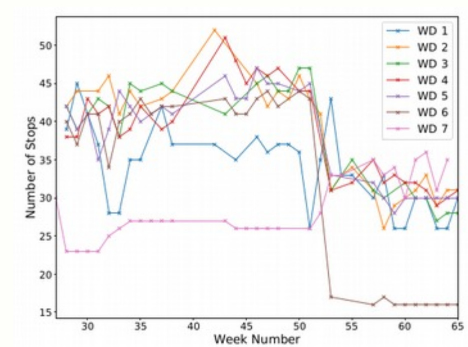
Can we do the learning better?

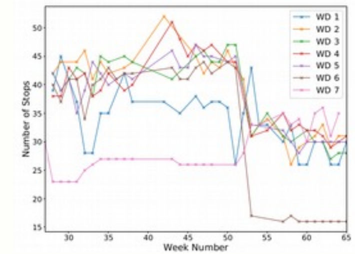
Training data = a *sequence* (one for every day) of observed routing *sequences*

→ each routing is over slightly different sets of customers



→ preferences can change over time *(concept drift)*





Concept drift

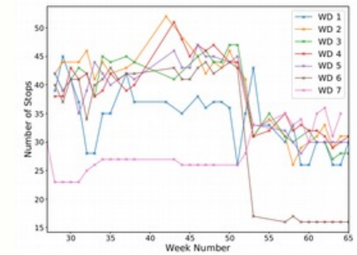
When 'counting' the probabilities:

- can include a *prior* on each historic instance wrt. current day
- e.g. weighing of the instance:

$$\mathbf{F} = \sum_t w_t \mathbf{A}^t.$$

  - uniform = unit weight
  - by time = more recent instances get higher weight
  - by similarity = how much overlap in clients with current day

Concept drift



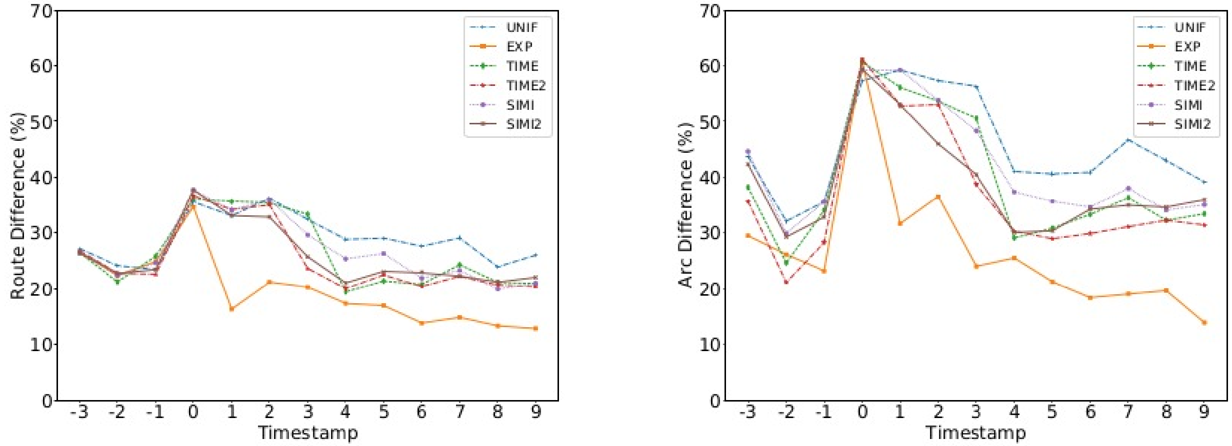


Fig. 7 Route and arc difference during concept drift (drop in number of stops)

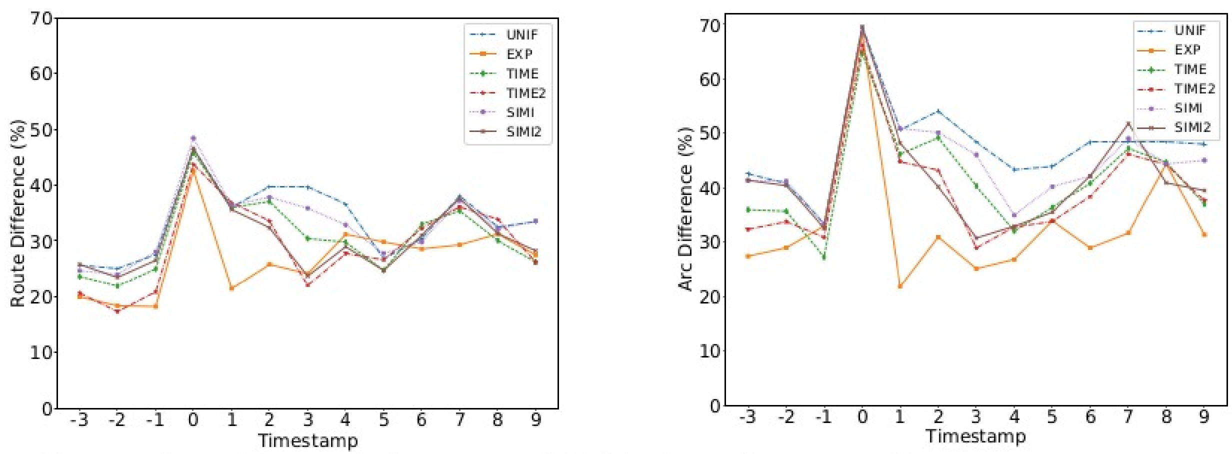


Fig. 8 Route and arc difference during concept drift (rise in number of stops)

Learning the preferences

= mimicking the user choices → copying, not *intelligence*?

Optimisation software is meant to do *better* than a user

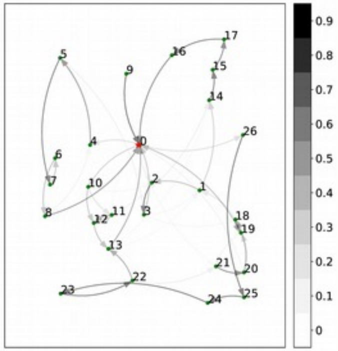
(by considering larger nr of candidates and better resolving of conflicts)

I prefer route X even if it is 2 kilometers longer

→ trade's off distance versus preference

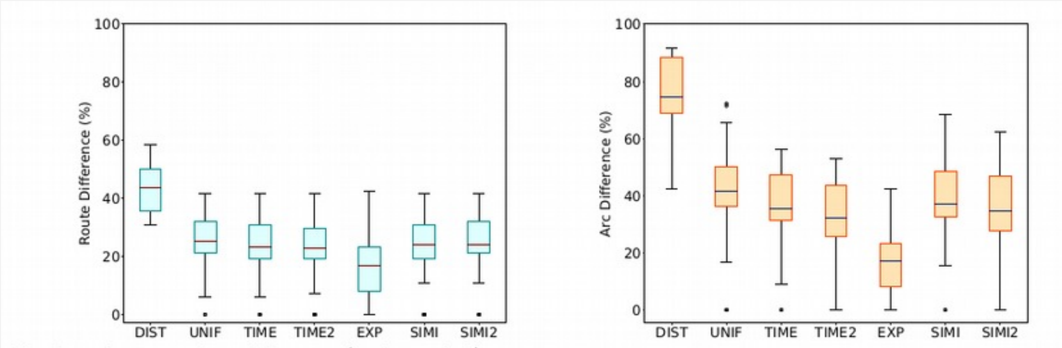
Optimize combination of both: $t'_{ij} = \beta t_{ij} + (1 - \beta) d_{ij}$.

Tacit knowledge (user preferences)



$$\max \prod_{(i,j)\in X} \mathbf{Pr}(\text{next stop} = j \mid \text{current stop} = i).$$

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \quad \Longrightarrow \quad \min \sum_{(i,j)\in A} -log(t_{ij})x_{ij}.$$



- Solvable with **<u>any VRP solver</u>**, including constraints
- Better than traditional approaches, multiple weighing schemes possible