A Delay-Robust Touristic Plan Recommendation Using Real-World Public Transportation Information

2nd ACM RecSys Workshop on Recommenders in Tourism

August 27th, 2017
Como, Italy

Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

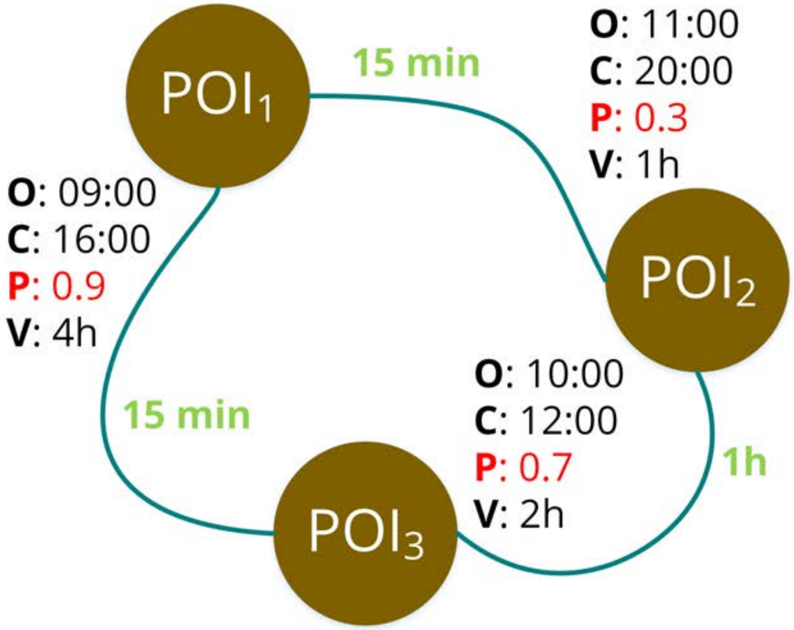
Victor Anthony Arrascue Ayala Kemal Cagin Gülsen Marco Muñiz

Anas Alzogbi Michael Färber Georg Lausen

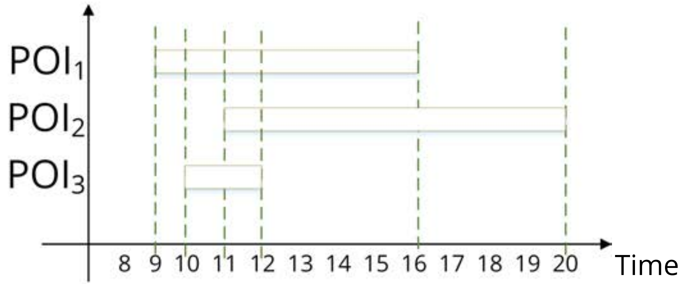
Tourist trip design problem (TTDP)

- Input
  - A user + interests
  - Set of POIs (attractions) + attributes
  - Set of constraints: user’s time budget, opening/ closing hours, travel times, etc.
- Goal: generate a visit plan (ordered visits of POIs)
  - Real-time requirement

TTDP – example

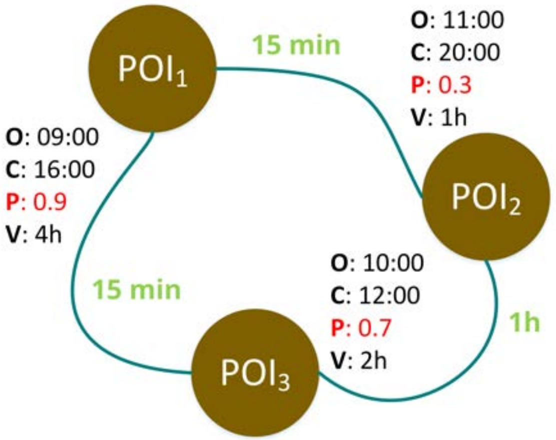


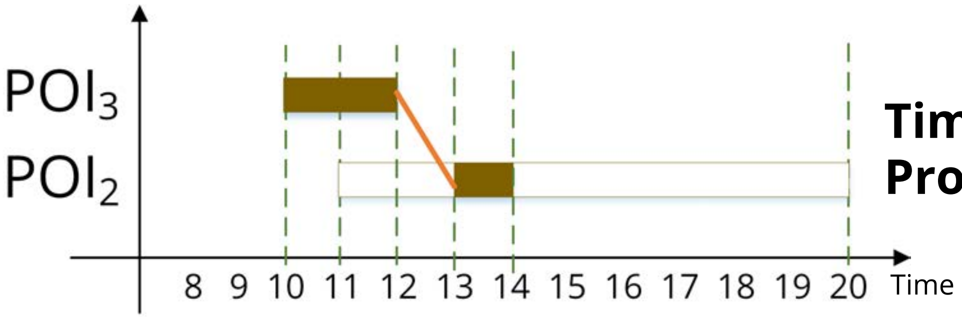
Time budget: 5h
Start: 09:00
End: 14:00



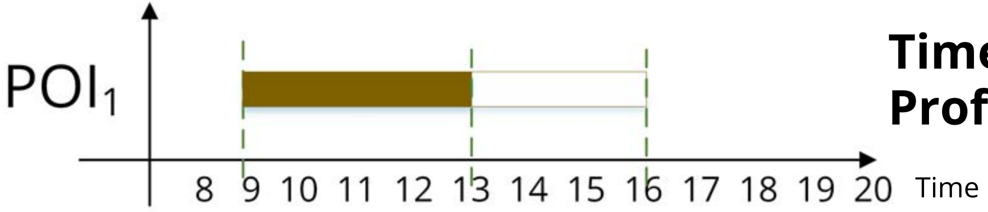
TTDP – top 2 solutions

Time budget: 5h
Start: 09:00
End: 14:00





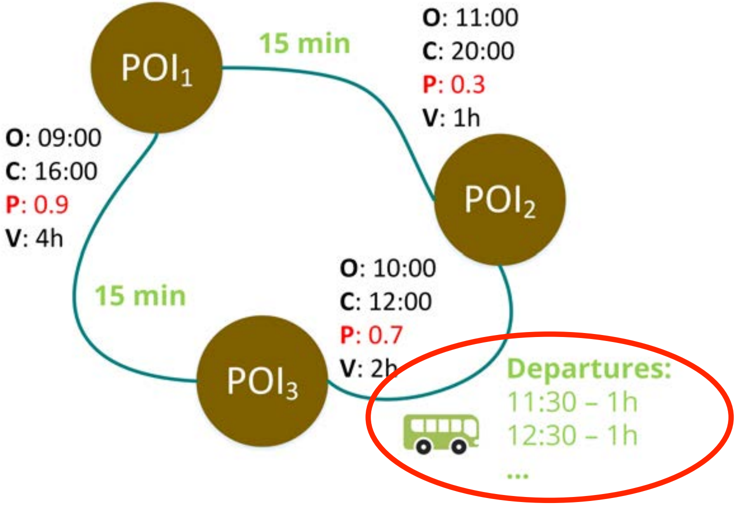
Time: 4h
Profit: 1.0

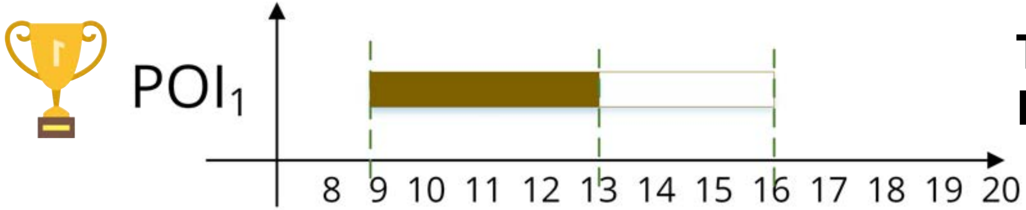


Time: 4h
Profit: 0.9

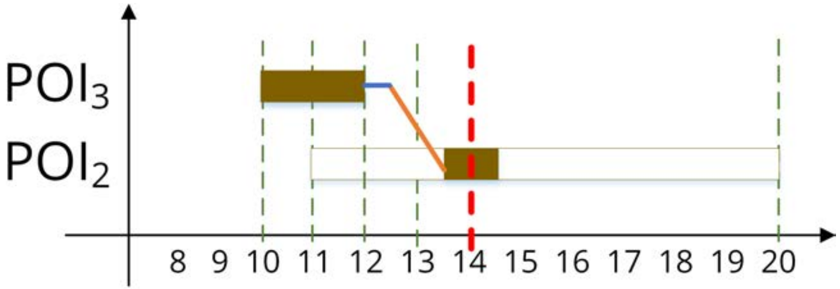
TTDP – time dependency constraint

Time budget: 5h
Start: 09:00
End: 14:00



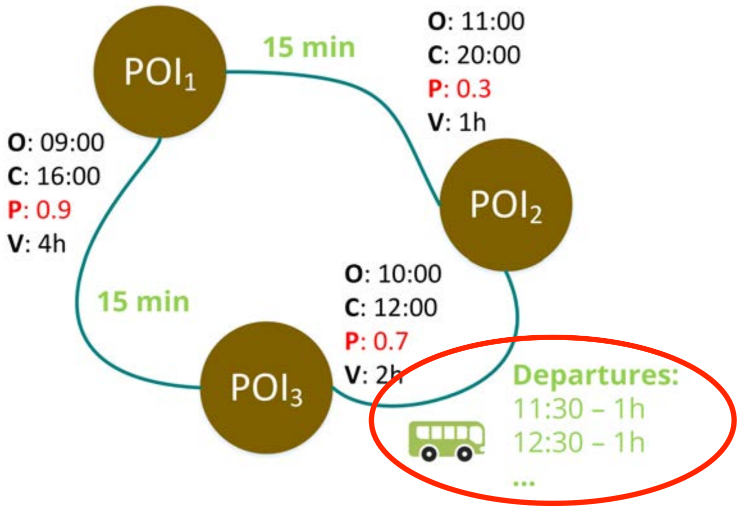


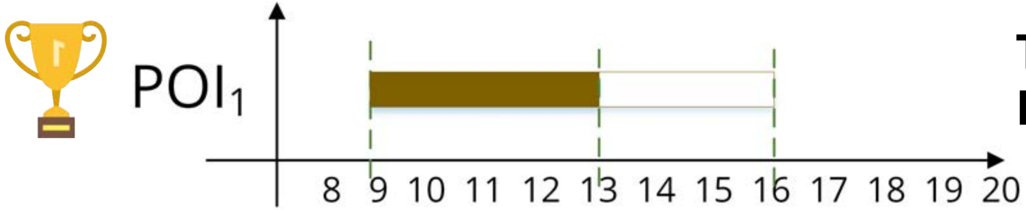
Time: 4h
Profit: 0.9



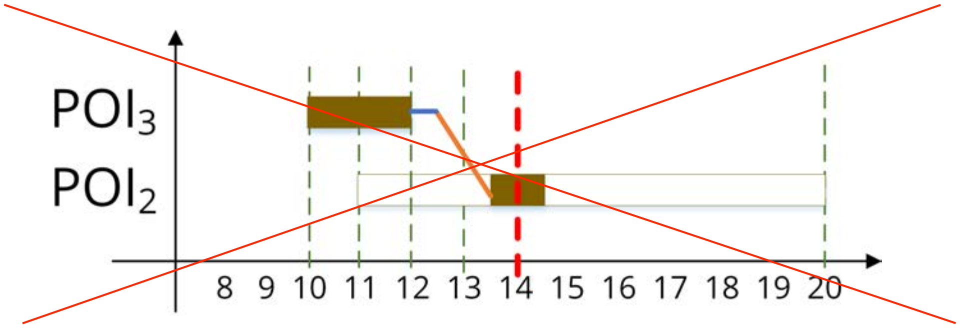
TTDP – time dependency constraint

Time budget: 5h
Start: 09:00
End: 14:00

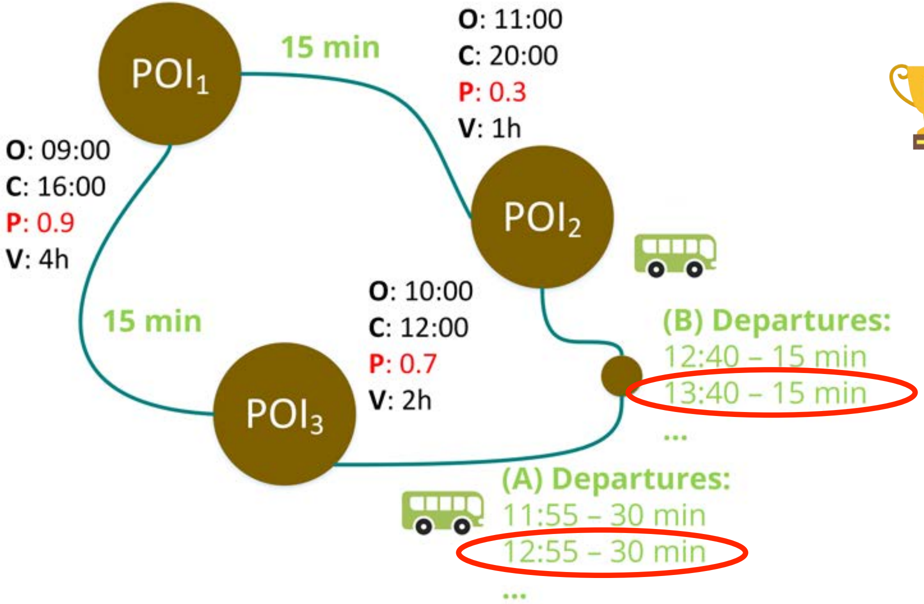


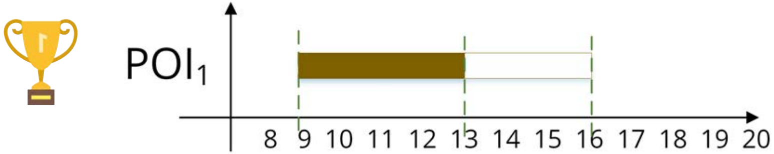


Time: 4h
Profit: 0.9

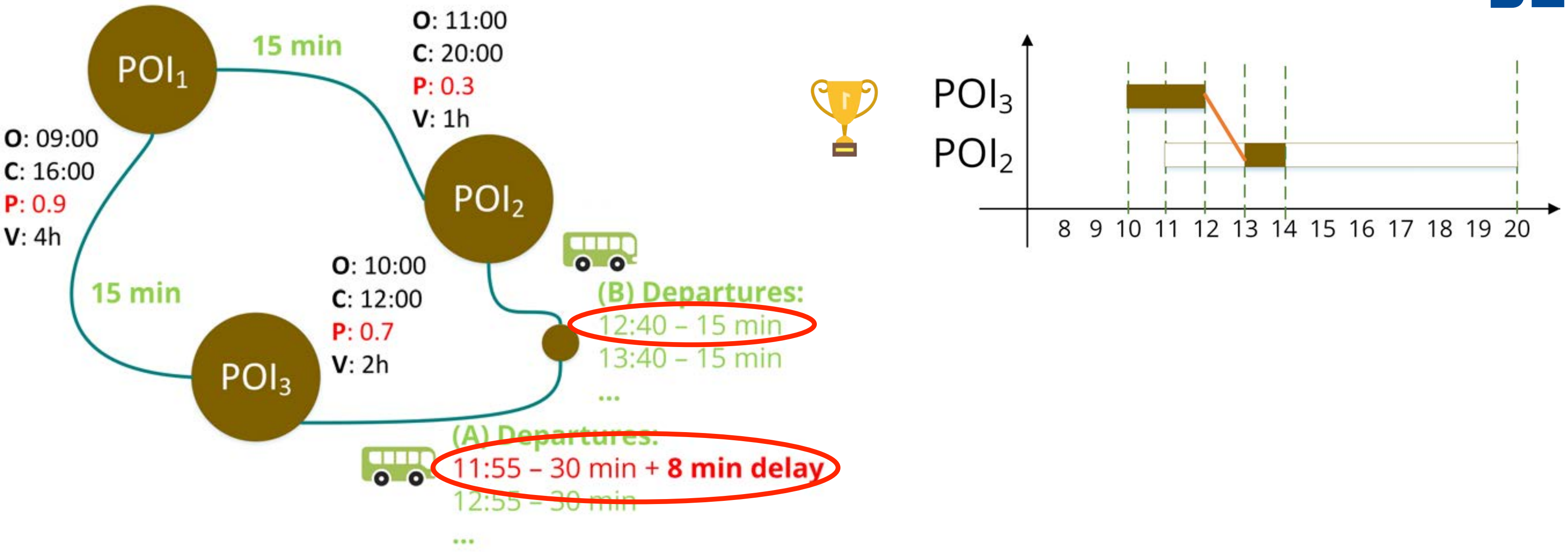


TTDP – time dependency constraint



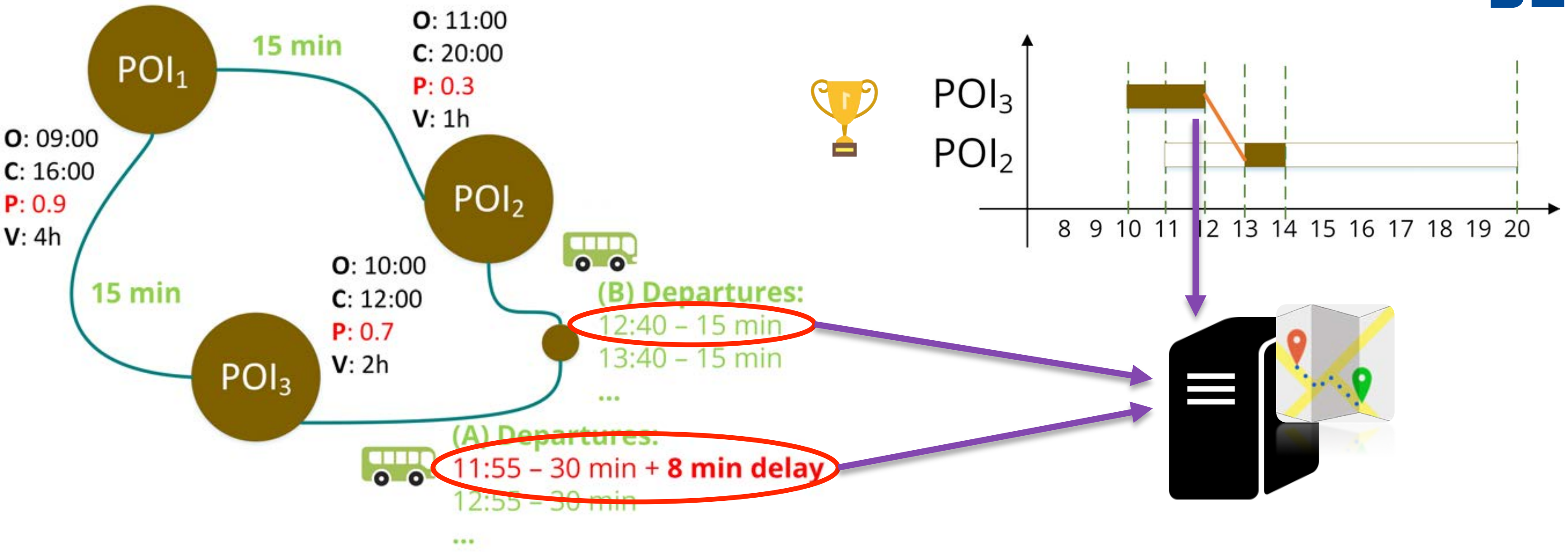


TTDP – delays



POI1
15 min
O: 11:00
C: 20:00
P: 0.3
V: 1h
O: 09:00
C: 16:00
P: 0.9
V: 4h
POI2
15 min
O: 10:00
C: 12:00
P: 0.7
V: 2h
POI3
(B) Departures:
12:40 – 15 min
13:40 – 15 min
...
(A) Departures:
11:55 – 30 min + 8 min delay
12:55 – 30 min
...
POI3
POI2
8 9 10 11 12 13 14 15 16 17 18 19 20

TTDP – delays



POI$_1$
15 min
O: 11:00
C: 20:00
P: 0.3
V: 1h
O: 09:00
C: 16:00
P: 0.9
V: 4h
POI$_2$
15 min
O: 10:00
C: 12:00
P: 0.7
V: 2h
POI$_3$
(B) Departures:
12:40 – 15 min
13:40 – 15 min
...
(A) Departures:
11:55 – 30 min + 8 min delay
12:55 – 30 min
...
POI$_3$
POI$_2$
8 9 10 11 12 13 14 15 16 17 18 19 20

Route planning for TTDP (advances)

- Transfer Patterns precomputation with Hub Stations, Fast direct-connection queries[1]
- Realistic: traffic, walking between stations, queries between geographic locations instead stations, etc.
- Real-time response (milliseconds)

[1] *Hannah Bast et al. Fast Routing in Very Large Public Transportation Networks using Transfer Patterns.*

TD(T)OPTW to model TTDP

- OP: Orienteering problem (Knapsack Problem and Traveling Salesman problem)
- TD: time-dependency
- (T): team (plans for multiple days)
- TW: predefined time windows
- Integer linear programming
- OP, OPTW, TDOP, TOP are NP-hard

Iterated Local Search (ILS[2])

- Insert Step
  - Inserts a POI into the solution
- Shake Step
  - Removes a POI to escape from local optima
- Generates good solutions (deviates 1.8% from optimal and takes ~1 sec)

[2] *Pieter Vansteenwegen et al. Iterated local search for the team orienteering problem with time windows.*

Iterated Local Search (ILS[2])

```
S ← 1;
R ← 1;
NumberOfTimesNoImprovement ← 0;
while NumberOfTimesNoImprovement < 150 do
|       while not local optimum do
|       |       Insert;
|       If Solution better than BestFound then
|       |       BestFound ← Solution;
|       |       R ← 1;
|       |       NumberOfTimesNoImprovement ← 0;
|       Else
|       |       NumberOfTimesNoImprovement
|       |               ← NumberOfTimesNoImprovement+1;
|       Shake Solution (R, S);
|       S ← S + R;
|       R ← R+1;
|       If S>=Size of smallest Tour then
|       |       S ← S - Size of smallest Tour;
|       If R==n/(3*m) then
|       |       R ← 1;
Return BestFound;
```

- Solution stable for 150 iterations

[2] *Pieter Vansteenwegen et al. Iterated local search for the team orienteering problem with time windows.*

Iterated Local Search (ILS[2])

```
S ← 1;
R ← 1;
NumberOfTimesNoImprovement ← 0;
while NumberOfTimesNoImprovement < 150 do
|    while not local optimum do
|    |    Insert;
|    If Solution better than BestFound then
|    |    BestFound ← Solution;
|    |    R ← 1;
|    |    NumberOfTimesNoImprovement ← 0;
|    Else
|    |    NumberOfTimesNoImprovement
|    |         ← NumberOfTimesNoImprovement+1;
|    Shake Solution (R, S);
|    S ← S + R;
|    R ← R+1;
|    If S>=Size of smallest Tour then
|    |    S ← S - Size of smallest Tour;
|    If R==n/(3*m) then
|    |    R ← 1;
Return BestFound;
```

- Solution stable for 150 iterations
- Insert until local optimum

[2] *Pieter Vansteenwegen et al. Iterated local search for the team orienteering problem with time windows.*

Iterated Local Search (ILS[2])

```
S ← 1;
R ← 1;
NumberOfTimesNoImprovement ← 0;
while NumberOfTimesNoImprovement < 150 do
|       while not local optimum do
|       |       Insert;
|       If Solution better than BestFound then
|       |       BestFound ← Solution;
|       |       R ← 1;
|       |       NumberOfTimesNoImprovement ← 0;
|       Else
|       |       NumberOfTimesNoImprovement
|       |            ← NumberOfTimesNoImprovement+1;
|       Shake Solution (R, S);
|       S ← S + R;
|       R ← R+1;
|       If S>=Size of smallest Tour then
|       |       S ← S - Size of smallest Tour;
|       If R==n/(3*m) then
|       |       R ← 1;
Return BestFound;
```

- Solution stable for 150 iterations
- Insert until local optimum
- Shake to escape for local optimum

[2] *Pieter Vansteenwegen et al. Iterated local search for the team orienteering problem with time windows.*

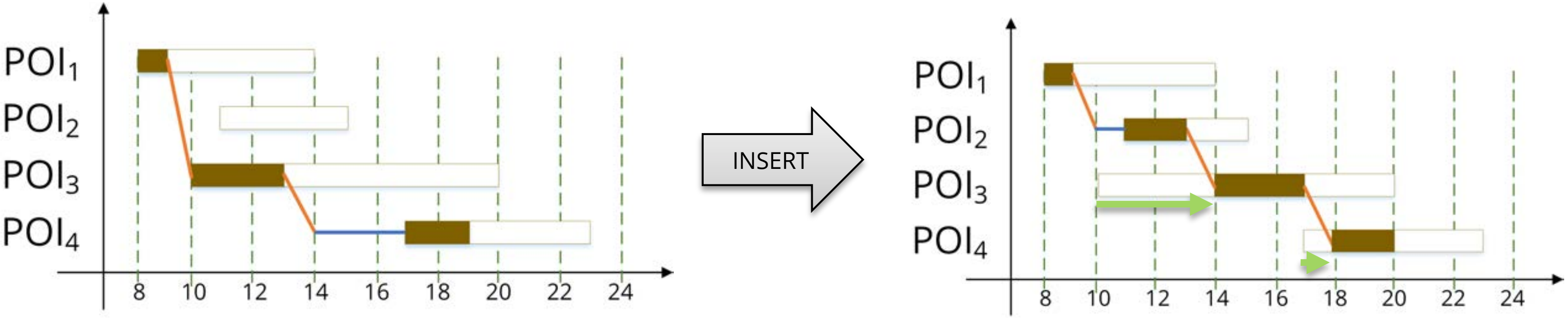
Iterated Local Search (ILS[2])

```
S ← 1;
R ← 1;
NumberOfTimesNoImprovement ← 0;
while NumberOfTimesNoImprovement < 150 do
|       while not local optimum do
|       |       Insert;
|       If Solution better than BestFound then
|       |       BestFound ← Solution;
|       |       R ← 1;
|       |       NumberOfTimesNoImprovement ← 0;
|       Else
|       |       NumberOfTimesNoImprovement
|       |               ← NumberOfTimesNoImprovement+1;
|       Shake Solution (R, S);
|       S ← S + R;
|       R ← R+1;
|       If S>=Size of smallest Tour then
|       |       S ← S - Size of smallest Tour;
|       If R==n/(3*m) then
|       |       R ← 1;
Return BestFound;
```

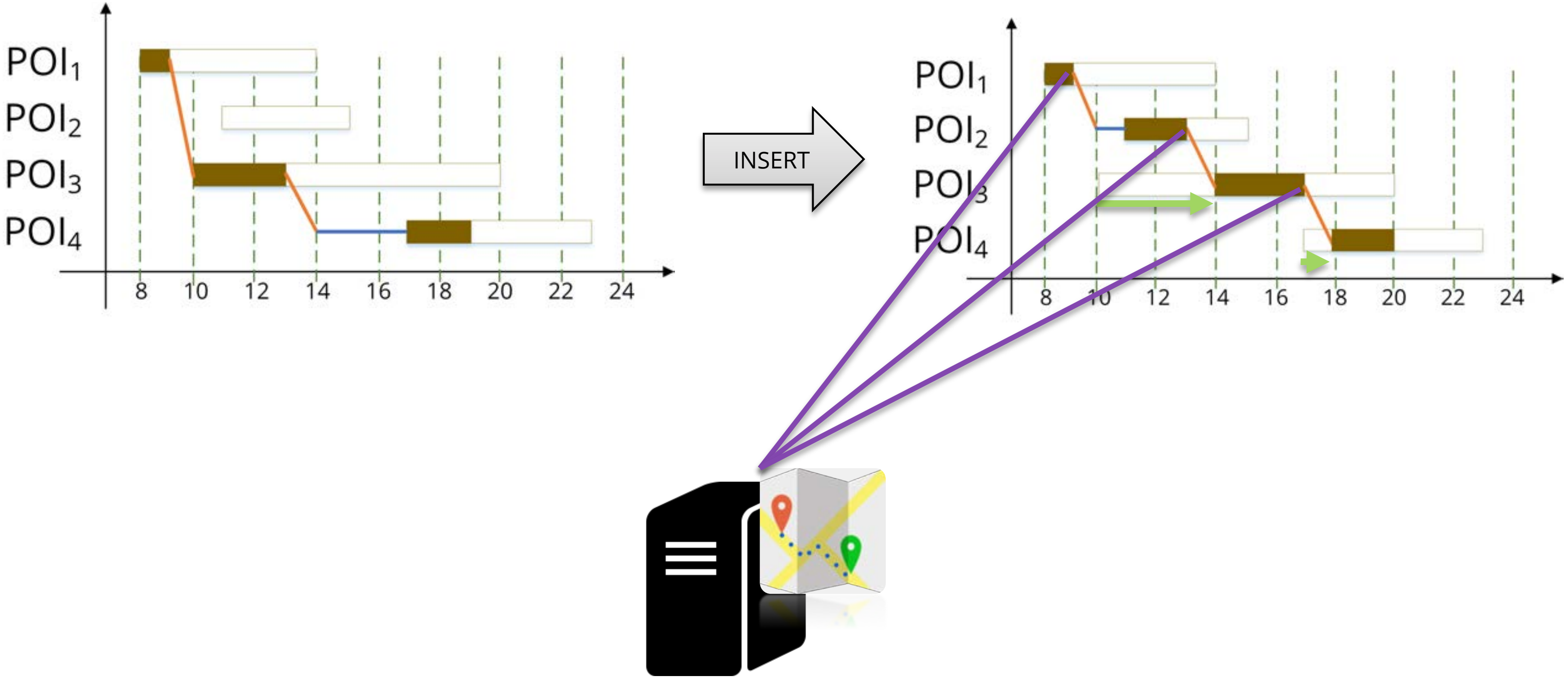
- Solution stable for 150 iterations
- Insert until local optimum
- Shake to escape for local optimum
- Variables control number of removed POIs and start position of removal

[2] *Pieter Vansteenwegen et al. Iterated local search for the team orienteering problem with time windows.*

ILS Insert step and route planning



ILS Insert step and route planning



ILS Insert step and route planning

- How to make use of information of the route planner?
- Without compromising the quality of solution
- Without violating the real-time requirement

Current solutions

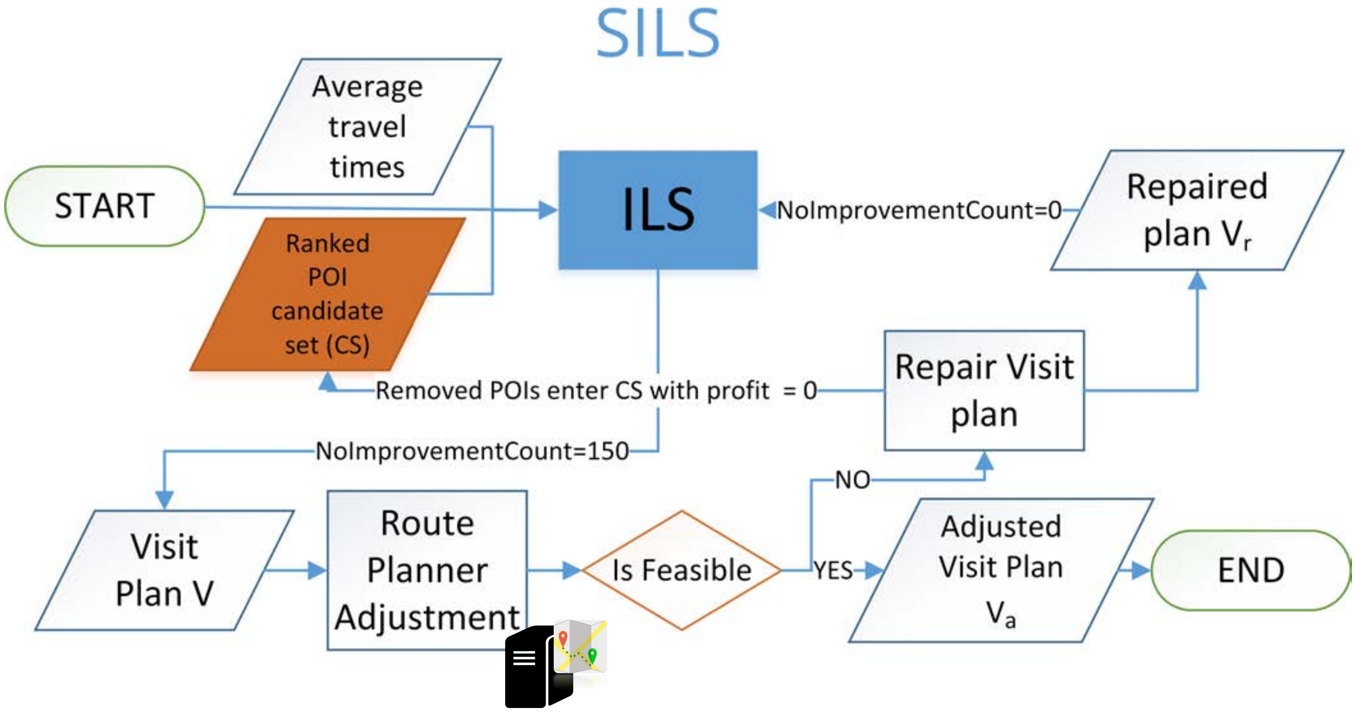
- Time-independent approximation (e.g. avg. travel times) => infeasible plans
- Pre-compute trip plans between all pairs of POIs and times => not possible in large networks
- Pre-compute travel times exploiting regularities in the schedules => not always regular
- Sacrifice route planning aspects (e.g. multi-modality, transfers, walking, etc.) => unrealistic

Our approaches

- Based on average travel times
- Aligned from time to time with Route Planner information

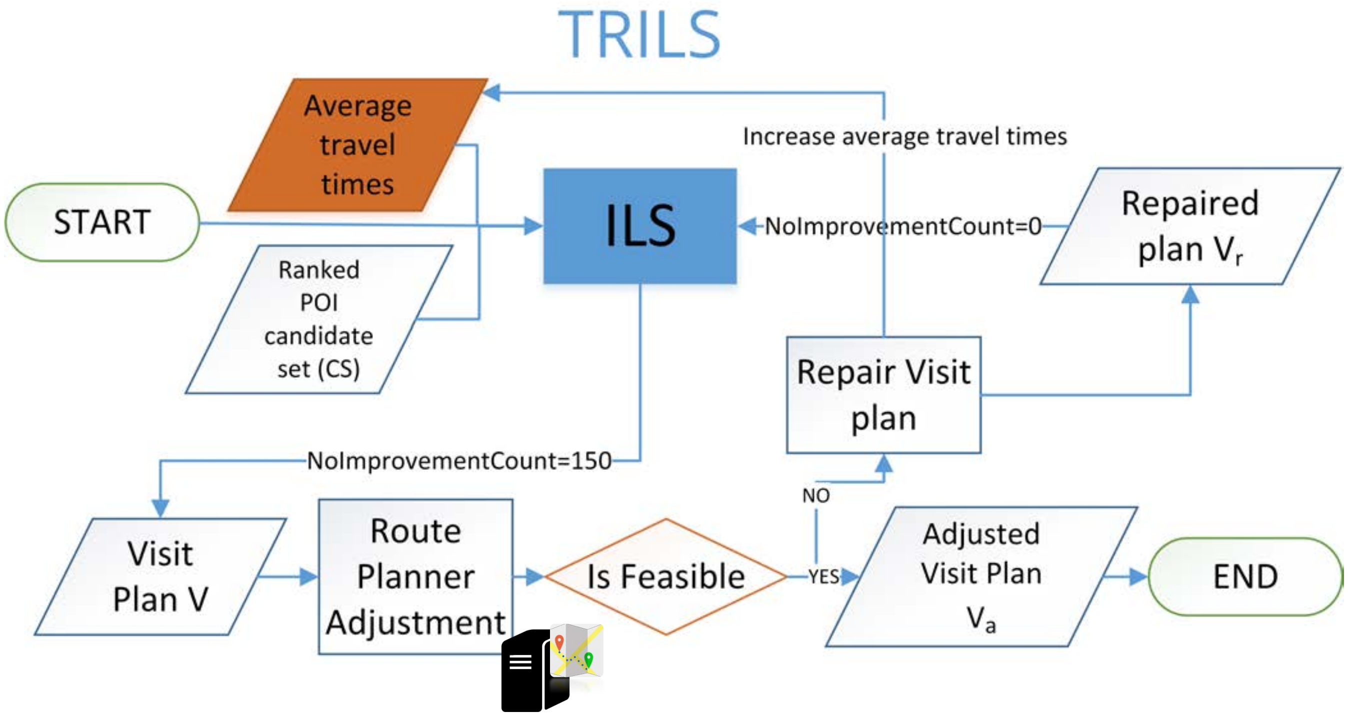
- Strict ILS (SILS)
- Time-relaxed ILS (TRILS)
- Precise Hybrid ILS (PHILS)

SILS



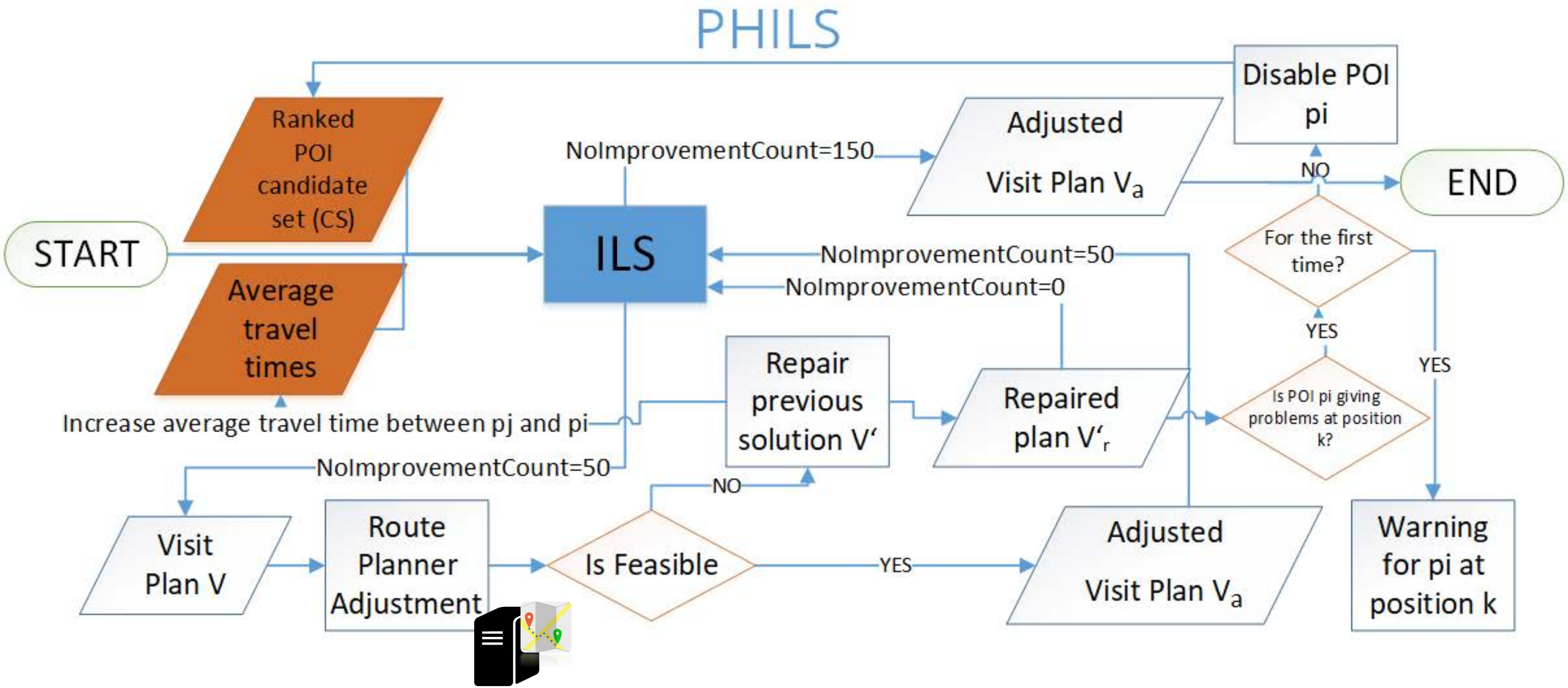
SILS
START
Average travel times
Ranked POI candidate set (CS)
ILS
NoImprovementCount=0
Repaired plan V_r
Removed POIs enter CS with profit = 0
Repair Visit plan
NoImprovementCount=150
NO
Visit Plan V
Route Planner Adjustment
Is Feasible
YES
Adjusted Visit Plan V_a
END

TRILS



TRILS
Average travel times
Increase average travel times
START
ILS
NoImprovementCount=0
Repaired plan V_r
Ranked POI candidate set (CS)
Repair Visit plan
NoImprovementCount=150
NO
Visit Plan V
Route Planner Adjustment
Is Feasible
YES
Adjusted Visit Plan V_a
END

PHILS



PHILS
Disable POI pi
Adjusted Visit Plan V_a
NoImprovementCount=150
Ranked POI candidate set (CS)
NO
END
START
ILS
NoImprovementCount=50
NoImprovementCount=0
For the first time?
Average travel times
YES
YES
Repair previous solution V'
Repaired plan V'_r
Is POI pi giving problems at position k?
Increase average travel time between pj and pi
NoImprovementCount=50
NO
Visit Plan V
Route Planner Adjustment
Is Feasible
YES
Adjusted Visit Plan V_a
Warning for pi at position k

Evaluation – set up

- 75 POIs – Izmir
- Public bus transportation (ESHOT) –
  - 7.7K stations and ~300 working bus lines
- 75 x 75 possible start-end location pairs
- Time budget: 4, 6 or 8 hours
- Starting times: 10:00 or 12:00
- 5 different user profiles
- TOTAL: ~170K requests

Evaluation – observations

- AvgILS produces infeasible plans
- RepAvgILS (baseline): simple repair strategy

- More infeasible plans are produced:
  - In the nights (close to end of time window)
  - For large time budgets

Evaluation – observations

- Average travel times are a **good** estimator!

- 170K requests – only 1.6 K (no delays), 1.0 K (delays) were infeasible

Evaluation – overall score

- Baseline: AvgILS and RepAvgILS
- No Delays:
  - SILS (+0.07%) → PHILS (+0.06%) → TRILS (+0.05%) → RepAvgILS
- Delays:
  - PHILS (+0.05%) → SILS (+0.04%) → TRILS (+0.035%) → RepAvgILS

Evaluation – profit for infeasible plans

- No Delays: SILS (+6.76%) → PHILS (+6.06%) → TRILS (+5.12%) wrt. to RepAvgILS
- Delays: PHILS (+6.9%) → SILS (+6.6%) → TRILS (+4.95%) wrt. to RepAvgILS
- SILS and TRILS: recover score of infeasible plans
- PHILS produces alternative solutions for feasible plans

Evaluation – execution times

- Execution times wrt. RepAvgILS:
  - SILS (1.2% slower) → TRILS (1.5% slower) → PHILS (26.12% slower) wrt. to RepAvgILS
  - All < 15 ms x request (on average)

Evaluation – observations

- Delays
  - Not always a bad thing if a system is aware of them
  - New possibilities for traveling might become available!

Conclusions & Future work

- Focus: modeling a realistic scenario
  - Considering delays was possible!
- SILS and PHILS performed the best
- Real-time requirement fulfilled

- Understand better user needs

Thank you!

UNI FREIBURG

Any questions?