A Large Neighborhood Search for a Cooperative Optimization Approach for Distributing Service Points in Mobility Applications

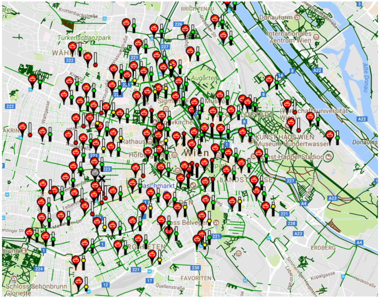
Thomas Jatschka, Tobias Rodemann, Günther Raidl

April 20, 2021

Motivation

Goal: find an optimal set of locations within a certain geographical area for placing service points

- for mobility purposes:
  - bike sharing stations
  - rental stations for car sharing
  - charging stations for electric vehicles
  - ...



Motivation

Cooperative Optimization Approach

⇒ **Cooperative Optimization Approach:**

- solves demand data acquisition and optimization in one process:
  - preference-based optimization algorithm
  - customers interacting with the algorithm
- expected benefits:
  - faster and cheaper data acquisition
  - stronger emotional link of users to the product
  - better and more accepted optimization results

The Generalized Service Point Distribution Problem (GSPDP)

Problem Formalization

We are given

- a set of locations $V = \{1, \ldots, n\}$ at which service points may be built,
- a set of potential users $U = \{1, \ldots, m\}$,
- building costs z_v^{fix} and maintenance costs z_v^{var} for each location $v \in V$,
- a maximum budget B for building service points,
- and a prize q that is earned for each unit of satisfied customer demand

The Generalized Service Point Distribution Problem (GSPDP)

Problem Formalization

User Information:

- set of use cases C_u for each user u:
  - going to work
  - recreational facilities
  - shopping
  - ...
- use case demands $D_{u,c}$
- **service point requirements** (SPRs) $R_{u,c}$:
  - EV charging: one station necessary
  - bike sharing: two stations necessary (pickup & return)

The Generalized Service Point Distribution Problem (GSPDP)

Objective Function

$$\max \; f(X) = \underbrace{q \cdot \sum_{u \in U} \sum_{c \in C_u} D_{u,c} \cdot \min_{r \in R_{u,c}} \left(\max_{v \in X} w_{r,v} \right)}_{\text{total price earned}} - \underbrace{\sum_{v \in X} z_v^{\mathrm{var}}}_{\text{maintenance costs}}$$

$$\underbrace{\sum_{v \in V} z_v^{\mathrm{fix}} x_v}_{\text{building costs}} \leq B$$

The Generalized Service Point Distribution Problem (GSPDP)

Suitability of a Service Point

- $w_{r,v} \in [0,1]$: suitability of a service point at location v w.r.t. SPR r
- suitability values not explicitly known
- infeasible to ask all suitability values from users
- $\Rightarrow$ reduce user interaction as much as possible
- $\Rightarrow$ confront users with easy questions that provide strong guidance for the target system

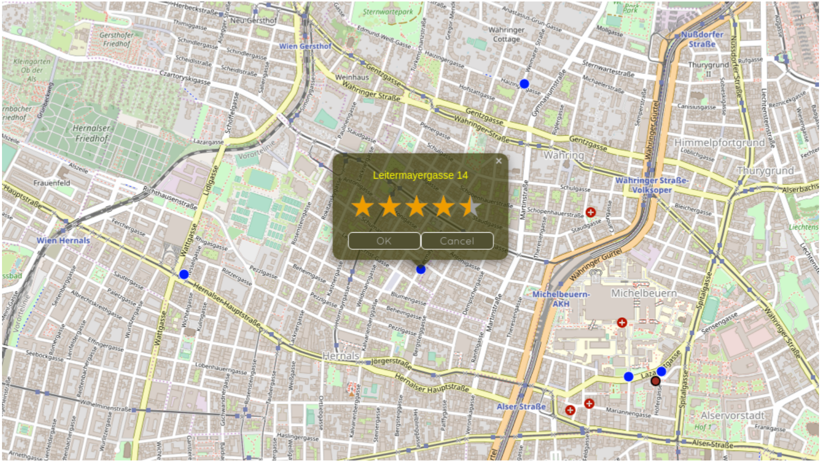
The Generalized Service Point Distribution Problem (GSPDP)

User Interaction

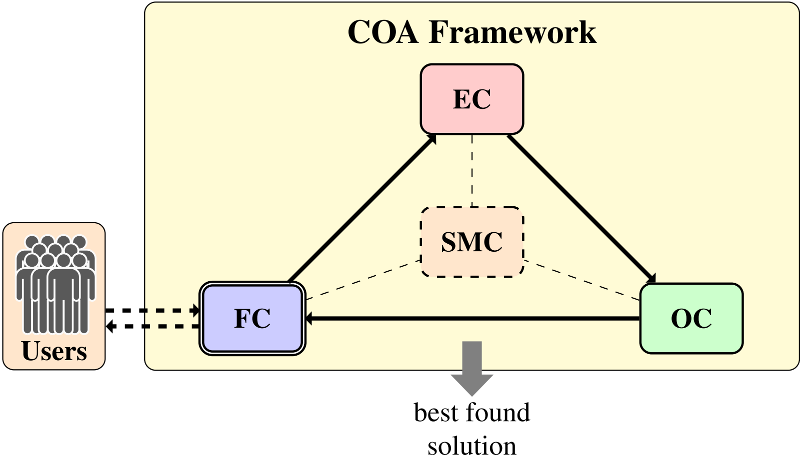
- a small number of location scenarios presented to users
- users are asked to evaluate location scenario S w.r.t. to one of their SPRs
- user selects most suitable location from S and provides suitability value on a five valued scale

The Generalized Service Point Distribution Problem (GSPDP)

User Interaction



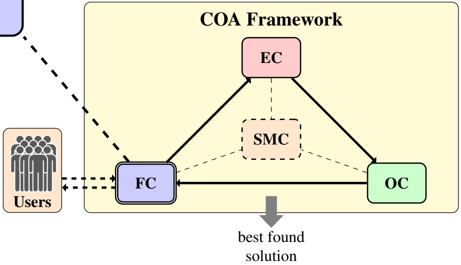
Cooperative Optimization Approach (COA)



Feedback Component (FC)

- gathers use case information of customers
- generates location scenarios for users to evaluate

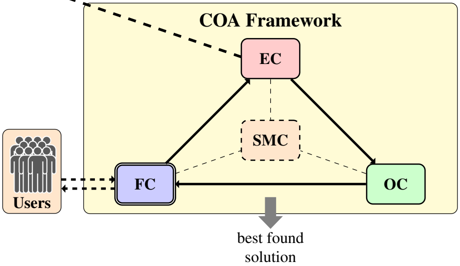
⇒ interacts with users



Evaluation Component (EC)

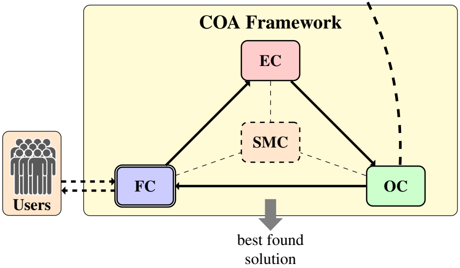
- processes user feedback obtained from the FC
- derives the means for evaluating candidate solutions without relying on users

$\Rightarrow$ surrogate function



Optimization Component (OC)

- generates optimal or close-to-optimal solution
- based on current surrogate objective function



Optimization Component (OC)

The Generalized Service Point Distribution Problem (GSPDP)

$$\max \;\; \tilde{f}_{\Theta}(X) = q \cdot \sum_{u \in U} \sum_{c \in C_u} D_{u,c} \cdot \min_{r \in R_{u,c}} \left(\max_{v \in X} \tilde{w}_{\Theta}(r, v) \right) - \sum_{v \in X} z_v^{\mathrm{var}}$$

$$\sum_{v \in V} z_v^{\mathrm{fix}} x_v \leq B$$

Optimization Component (OC)

Mixed Integer Linear Programming Formulation

$$
\begin{aligned}
\max \quad & q \cdot \sum_{u \in U} \sum_{c \in C_u} D_{u,c}\, y_{u,c} - \sum_{v \in V} z_v^{\mathrm{var}}\, x_v \\
& \sum_{v \in V} o_{r,v} \leq 1 && \forall r \in R \\
& o_{r,v} \leq x_v && \forall v \in V,\ r \in R \\
& y_{u,c} \leq \sum_{v \in V} \tilde{w}_{\Theta}(r, v) \cdot o_{r,v} && \forall u \in U,\ c \in C_u,\ r \in R_{u,c} \\
& \sum_{v \in V} z_v^{\mathrm{fix}}\, x_v \leq B \\
& x_v \in \{0, 1\} && \forall v \in V \\
& 0 \leq y_{u,c} \leq 1 && \forall u \in U,\ c \in C_u \\
& 0 \leq o_{r,v} \leq 1 && \forall r \in R,\ v \in V
\end{aligned}
$$

Optimization Component (OC)

Large Neighborhood Search

- follows a classical local search framework but much larger neighborhoods considered in each iteration
- iterative destroy and repair scheme
    1. incumbent solution is destroyed
    2. destroyed solution is repaired w.r.t. to a subset of V

Large Neighborhood Search

Potential

- solutions are destroyed and repaired by greedy procedures
- greedy criterion: (surrogate) objective value not suitable $\Rightarrow$ potential $\tilde{\Pi}_\Theta(X)$ of a solution X :

$$C(u, X) = \left\{ c \in C_u \mid \min_{r \in R_{u,c}} \left(\max_{v \in X} \tilde{w}_\Theta(r, v) \right) > 0 \right\}$$

$$R(u, c, X) = \left\{ r \in R_{u,c} \mid \max_{v \in X} \tilde{w}_\Theta(r, v) > 0 \right\}$$

$$\tilde{\Pi}_\Theta(X) = \tilde{f}_\Theta(X) + \beta \cdot q \cdot \sum_{u \in U} \sum_{c \in C_u \setminus C(u,X)} \frac{D_{u,c} \cdot \min_{r \in R(u,c,X)} \left(\max_{v \in X} \tilde{w}_\Theta(r, v) \right) \cdot |R(u, c, X)|}{|R_{u,c}|}$$

Destroy Procedure:

- greedy approach
- select k "worst" locations in X w.r.t.

$$\omega^{\text{destroy}}(v, X) = \frac{1}{\tilde{\Pi}_{\Theta}(X) - \tilde{\Pi}_{\Theta}(X \setminus \{v\})}$$

- choose random location from k selected to remove from X
- repeat k' times

Large Neighborhood Search

Repair Procedure

Repair Procedure:

- greedy approach
- select k "best" locations in V w.r.t.

$$\omega^{\text{repair}}(v, X) = \tilde{\Pi}_{\Theta}(X \cup \{v\}) - \tilde{\Pi}_{\Theta}(X)$$

- choose random location from k selected to add to X
- repeat until budget is exhausted

Large Neighborhood Search

Parameterization

- two destroy operators:
  - $k = k' = 5$
  - $k = k' = 7$
- two repair operators:
  - $k = 3$
  - $k = 5$
- LNS terminates after 20 iterations without improvement
- $\beta = 0.1$

Large Neighborhood Search

Evaluating Solutions

Surrogate objective function:

$$\tilde{f}_\Theta(X) = q \cdot \sum_{u \in U} \sum_{c \in C_u} D_{u,c} \cdot \min_{r \in R_{u,c}} \left(\max_{v \in X} \tilde{w}_\Theta(r, v) \right) - \sum_{v \in X} z_v^{\text{var}}$$

Potential:

$$\tilde{\Pi}_\Theta(X) = \tilde{f}_\Theta(X) + \beta \cdot q \cdot \sum_{u \in U} \sum_{c \in C_u \setminus C(u,X)} \frac{D_{u,c} \cdot \min_{r \in R(u,c,X)} \left(\max_{v \in X} \tilde{w}_\Theta(r, v) \right) \cdot |R(u,c,X)|}{|R_{u,c}|}$$

$\Rightarrow$ time consuming to evaluate from scratch

Large Neighborhood Search

Evaluation Graph

- representation of objective function as graph
- consists of four layers:
  - location layer
  - SPR layer
  - use case layer
  - evaluation layer
- each node has function $\alpha()$ for calculating output propagated to node in the next layer
- nodes store all outputs from previous generated solution $\Rightarrow$ incremental evaluation of solution

Large Neighborhood Search

Evaluation Graph

$$\alpha_{LL}(l_v, X) = \begin{cases} 1 & \text{if } v \in X \\ 0 & \text{otherwise} \end{cases}$$

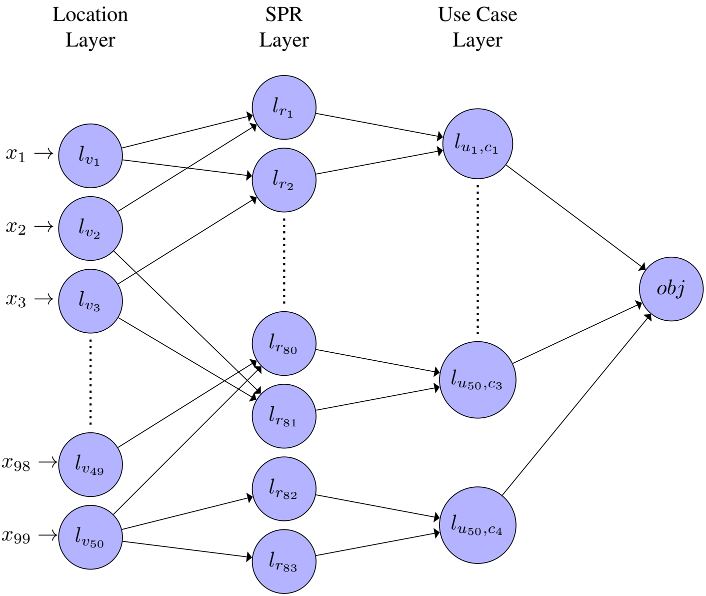
$$\alpha_{SL}(l_{u,r}, X) = \max_{(l_v, l_{u,r}) \in A_{LL}} (\alpha_{LL}(l_v, X) \cdot \tilde{w}_\Theta(v, r))$$

$$\alpha_{CL}(l_c, X) = \min_{(l_{u,r}, l_c) \in A_{SL}} \alpha_{SL}(l_{u,r}, X)$$

$$\alpha_{eval}(l_{\text{obj}}, X) = \sum_{(l_c, l_{\text{obj}}) \in A_{CL}} \alpha_{SL}(l_c, X) - \sum_{v \in X} z_v^{\text{var}}$$

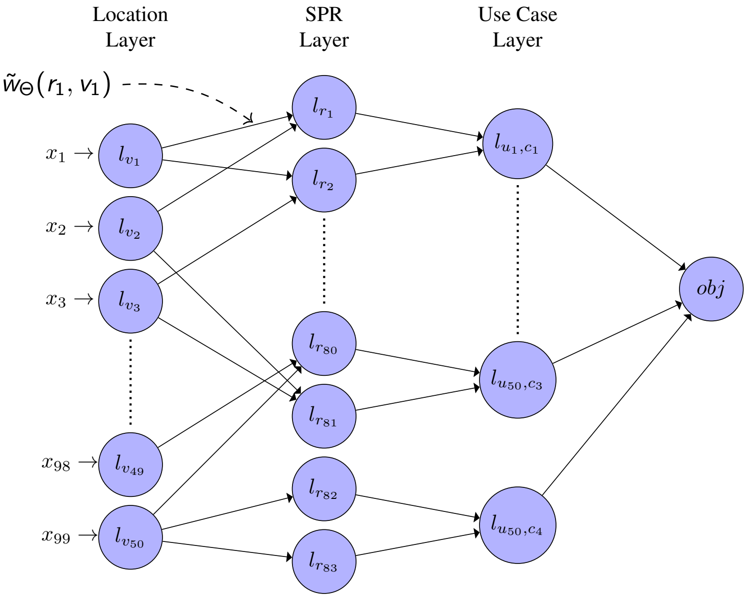
Large Neighborhood Search

Evaluation Graph



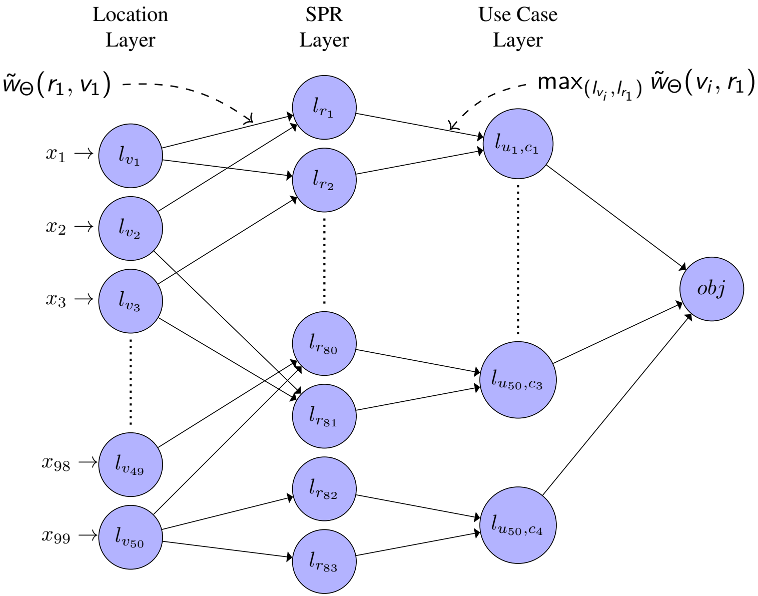
Large Neighborhood Search

Evaluation Graph



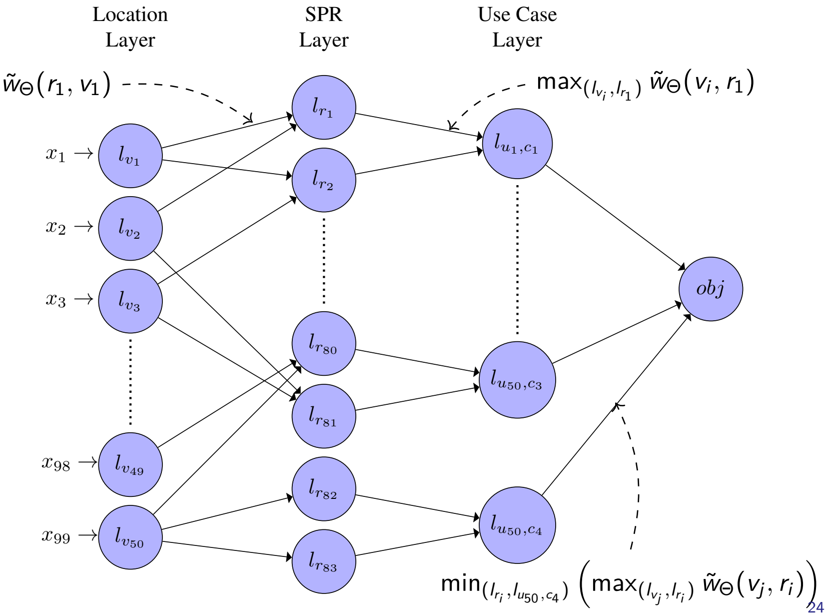
Large Neighborhood Search

Evaluation Graph



Large Neighborhood Search

Evaluation Graph



Feedback Component

Generation of Scenarios

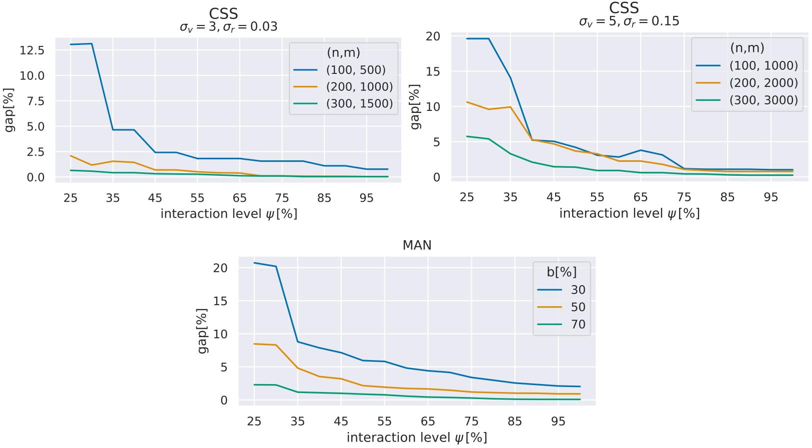
- iteratively present users scenarios containing all locations for which no suitability values are known yet w.r.t. SPR r
- $V(r) \leftarrow$ set of relvant locations for r, i.e., $V(r) = \{v \in V \mid w_{r,v} > 0\}$
- in each iteration new location in $V(r)$ identified
- if none of the locations in scenario are suitable for $r \Rightarrow V(r)$ completely known

$\Rightarrow$ $V(r)$ completely known after $|V(r)| + 1$ user interactions

$\Rightarrow$ upper bound I_u^{UB} on the total number of required interactions with user u:

$$I_u^{\mathrm{UB}} = \sum_{r \in R_u} (|V(r)| + 1)$$

Development of Solution Quality

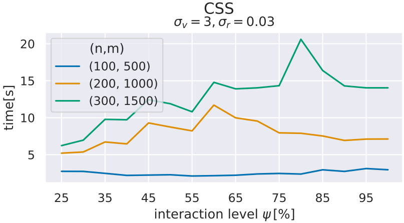


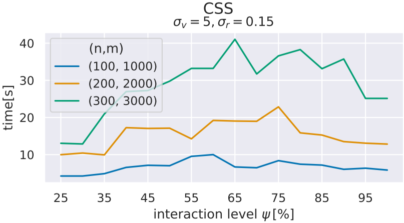
Computational Experiments

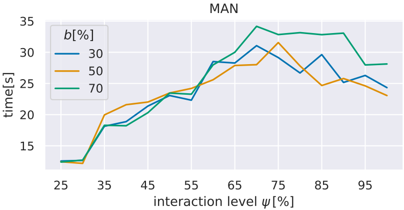
- Programming Languages: Python 3.9, Julia 1.6, C++
- Test runs have been executed on an Intel Xeon E5-2640 v4 with 2.40GHz
- 2 types of instances:
  - CSS:
    - inspired by car sharing scenario
    - use cases have two SPRs
  - MAN:
    - inspired by car sharing scenario
    - generated from real world data (New York Yellow Taxi Data)

Computational Experiments

OC - Computation Times (LNS)

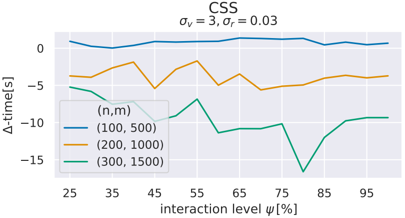


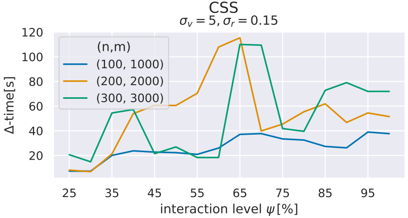


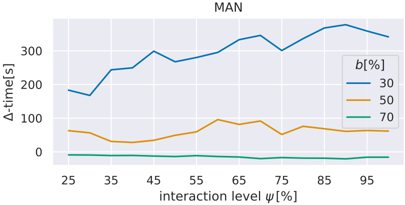


Computational Experiments

MILP/LNS - Runtime Comparison

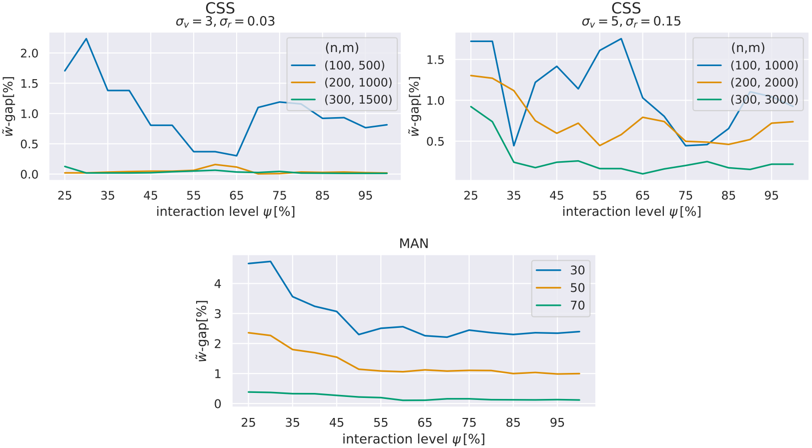






Computational Experiments

LNS - Optimality w.r.t. $\tilde{w}_\Theta$



Computational Experiments

MILP/LNS - Optimality Gaps

	CSS											
(n,m)	(100, 500)		(100, 1000)		(200, 1000)		(200, 2000)		(300, 1500)		(300, 3000)	
ψ	MILP	LNS	MILP	LNS	MILP	LNS	MILP	LNS	MILP	LNS	MILP	LNS
30	**12.00**	13.12	**19.39**	19.62	1.39	**1.17**	**6.60**	9.59	0.74	**0.57**	5.56	**5.40**
40	**3.70**	4.64	9.27	**5.22**	**1.06**	1.44	**4.05**	5.28	0.46	**0.42**	**2.00**	2.09
50	**2.10**	2.42	4.31	**4.20**	0.75	**0.69**	**2.07**	3.67	**0.23**	0.29	**1.32**	1.39
60	**0.65**	1.81	**1.90**	2.83	**0.19**	0.41	**1.59**	2.25	**0.18**	0.20	**0.55**	0.91
70	**0.20**	1.56	**0.49**	3.12	0.12	**0.11**	**0.79**	1.78	0.13	**0.09**	**0.21**	0.61
80	**0.02**	1.56	**0.92**	1.09	**0.04**	0.08	**0.36**	0.90	**0.03**	0.03	**0.12**	0.41
90	**0.02**	1.10	**0.06**	1.09	**0.01**	0.07	**0.03**	0.77	**0.01**	0.03	**0.02**	0.25

	MAN					
b[%]	30		50		70	
ψ	MILP	LNS	MILP	LNS	MILP	LNS
30	**15.71**	20.19	**7.46**	8.32	**2.09**	2.27
40	**6.16**	7.87	**3.16**	3.54	1.12	**1.09**
50	**3.15**	5.95	**1.81**	2.16	**0.73**	0.87
60	**2.19**	4.82	**0.93**	1.73	**0.46**	0.57
70	**1.32**	4.16	**0.49**	1.47	**0.29**	0.37
80	**0.49**	2.98	**0.20**	1.10	**0.09**	0.17
90	**0.27**	2.33	**0.01**	1.01	**0.01**	0.09

Conclusion

- Large Neighborhood Search (LNS) for COA
- potential as greedy criterion
- evaluation graph for incremental evaluation of solution
- LNS scales significantly better than MILP w.r.t. computation times
- LNS requires more tuning
- test LNS on more difficult instances to emphasize scalability even more

Thank you for your attention!
Questions?