Scheduling-Location-Routing Problems with Applications in Refugee Camp Management

Corinna Heßler

TU Kaiserslautern

30. June 2017

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

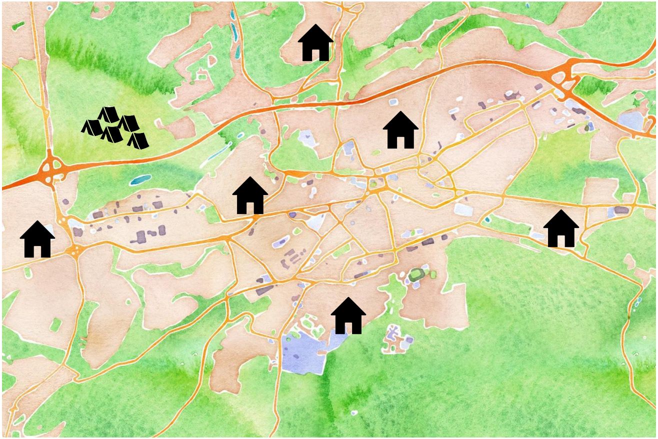
Motivation

Source:
www.maps.stamen.com
www.openstreetmap.org

Motivation

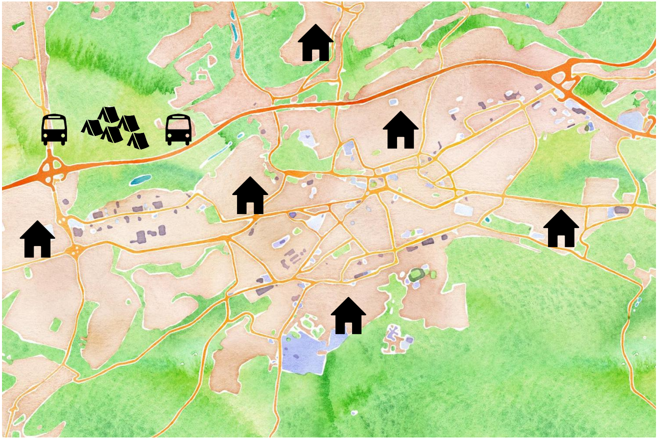
Source:
www.maps.stamen.com
www.openstreetmap.org

Motivation



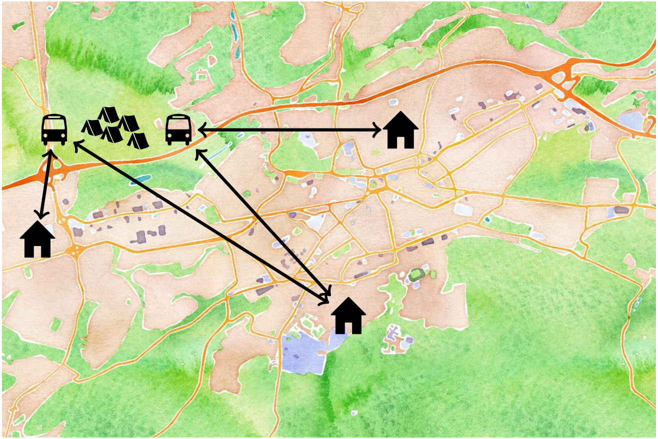
Source:
www.maps.stamen.com
www.openstreetmap.org

Motivation



Source:
www.maps.stamen.com
www.openstreetmap.org

Motivation



Source:
www.maps.stamen.com
www.openstreetmap.org

ScheLoc-Routing Problems

Formal Definition

- Given a set of jobs $\mathcal{J} = \{1, \ldots, n\}$ in job-locations a_i with processing times $p_i > 0$ and sizes $w_i \geq 0$.

Formal Definition

- Given a set of jobs $\mathcal{J} = \{1, \ldots, n\}$ in job-locations a_i with processing times $p_i > 0$ and sizes $w_i \geq 0$.
- Find locations for machines $\mathcal{M} = \{1, \ldots, m\}$ and a schedule of the jobs on the machines such that all restrictions are satisfied.

Formal Definition

- Given a set of jobs $\mathcal{J} = \{1, \ldots, n\}$ in job-locations a_i with processing times $p_i > 0$ and sizes $w_i \geq 0$.
- Find locations for machines $\mathcal{M} = \{1, \ldots, m\}$ and a schedule of the jobs on the machines such that all restrictions are satisfied.
- Transportation of jobs by a set of vehicles $\Omega = \{1, \ldots, \omega\}$.
- $cap_o = cap$ homogeneous capacity, $cap_o = \infty$ uncapacitated.

Formal Definition

- Given a set of jobs $\mathcal{J} = \{1, \ldots, n\}$ in job-locations a_i with processing times $p_i > 0$ and sizes $w_i \geq 0$.
- Find locations for machines $\mathcal{M} = \{1, \ldots, m\}$ and a schedule of the jobs on the machines such that all restrictions are satisfied.
- Transportation of jobs by a set of vehicles $\Omega = \{1, \ldots, \omega\}$.
- $cap_o = cap$ homogeneous capacity, $cap_o = \infty$ uncapacitated.
- Find routes such that each job is picked up and transported to a machine.

Formal Definition

- Given a set of jobs $\mathcal{J} = \{1, \ldots, n\}$ in job-locations a_i with processing times $p_i > 0$ and sizes $w_i \geq 0$.
- Find locations for machines $\mathcal{M} = \{1, \ldots, m\}$ and a schedule of the jobs on the machines such that all restrictions are satisfied.
- Transportation of jobs by a set of vehicles $\Omega = \{1, \ldots, \omega\}$.
- $cap_o = cap$ homogeneous capacity, $cap_o = \infty$ uncapacitated.
- Find routes such that each job is picked up and transported to a machine.
- Minimize pure scheduling objective.

Release Dates

- Jobs can only start processing after arrival at machine location.

Release Dates

- Jobs can only start processing after arrival at machine location.
- Let $\mathcal{J}_\rho$ be the set of jobs picked up on route ρ, s_ρ the starting time and l_ρ the length of route ρ.
- The release date of job $i \in \mathcal{J}_\rho$ is

$$r_i = s_\rho + l_\rho.$$

Release Dates

- Jobs can only start processing after arrival at machine location.
- Let $\mathcal{J}_\rho$ be the set of jobs picked up on route ρ, s_ρ the starting time and l_ρ the length of route ρ.
- The release date of job $i \in \mathcal{J}_\rho$ is

$$r_i = s_\rho + l_\rho.$$

- Consecutive routes must be consistent in time and location.

Further Restrictions

- Even for uncapacitated vehicles more than one route per vehicle may be optimal.

Further Restrictions

- Even for uncapacitated vehicles more than one route per vehicle may be optimal.
- Either vehicles are assigned to a machine or vehicles are free.

Further Restrictions

- Even for uncapacitated vehicles more than one route per vehicle may be optimal.
- Either vehicles are assigned to a machine or vehicles are free.

Problem is $\mathcal{NP}$-complete in general $\Rightarrow$ Heuristic approach, here: skewed general variable neighbourhood search (SGVNS)

Problem Definition

- Parallel machines, discrete location
- Minimize makespan

Problem Definition

- Parallel machines, discrete location
- Minimize makespan
- Homogeneous vehicles, each assigned to a machine
- Special case: One vehicle per machine

Problem Definition

- Parallel machines, discrete location
- Minimize makespan
- Homogeneous vehicles, each assigned to a machine
- Special case: One vehicle per machine
- Size $w_i = 1$ for $i \in \mathcal{J}$

SGVNS for ScheLoc-Routing Problem

Algorithm 1 SGVNS for DPM ScheLoc routing problems

Input: An initial solution $(X^*, S^*, \mathcal{R}^*) = (X, S, \mathcal{R})$, and a parameter $q_{\max}$.

1: Set $k = 1$, $q = 0$.
2: Perform a move from neighborhood $\mathcal{NR}$ or $\mathcal{NV}$.
3: Remove any infeasibility of routes using ejection chains.
4: Remove any infeasibility of the schedule by moving the affected jobs to the best position on the new machine.
5: Compute an optimal schedule for the given locations and routes. Let $(X, S', \mathcal{R}')$ be the obtained solution. Set $(X, S, \mathcal{R}) = (X, S', \mathcal{R}')$. ▷ Local Search
6: If $f(X, S, \mathcal{R}) < f(X^*, S^*, \mathcal{R}^*)$, set $(X^*, S^*, \mathcal{R}^*) = (X, S, \mathcal{R})$ and $q = 0$. Else, set $q = q + 1$.
7: If $q < q_{\max}$, go to Step 2.
8: If $k = m$, return $(X^*, S^*, \mathcal{R}^*)$.
9: Perform a move in $\mathcal{NL}_k$, compute new routes using Greedy heuristic, and a new schedule by the ERD rule, yielding solution $(X, S, \mathcal{R})$. ▷ Shaking
10: If $f(X, S, \mathcal{R}) < f(X^*, S^*, \mathcal{R}^*)$, set $(X^*, S^*, \mathcal{R}^*) = (X, S, \mathcal{R})$.
11: Set $q = 0$ and $k = k + 1$. Go to Step 2.

Shaking

Algorithm 1 SGVNS for DPM ScheLoc routing problems

9: Perform a move in $\mathcal{NL}_k$, compute new routes using Greedy heuristic, and a new schedule by the ERD rule, yielding solution $(X, S, \mathcal{R})$. ▷ Shaking

Local Search

Algorithm 1 SGVNS for DPM ScheLoc routing problems

2: Perform a move from neighborhood $\mathcal{NR}$ or $\mathcal{NV}$.
3: Remove any infeasibility of routes using ejection chains.
4: Remove any infeasibility of the schedule by moving the affected jobs to the best position on the new machine.
5: Compute an optimal schedule for the given locations and routes. Let $(X, S', \mathcal{R}')$ be the obtained solution. Set $(X, S, \mathcal{R}) = (X, S', \mathcal{R}')$. ▷ Local Search

Application to Evacuation Planning

Problem Definition

- Set of buses $\Omega = \{1, \ldots, \omega\}$ with homogeneous capacity, initially located in bus depot,
- Set of evacuees $\mathcal{J} = \{1, \ldots, n\}$ with locations $a_i \in V^g$ and accommodation time p_i,
- possible shelter locations $\mathcal{L}$ with capacities cap_k^S.

Problem Definition

- Set of buses $\Omega = \{1, \ldots, \omega\}$ with homogeneous capacity, initially located in bus depot,
- Set of evacuees $\mathcal{J} = \{1, \ldots, n\}$ with locations $a_i \in V^g$ and accommodation time p_i,
- possible shelter locations $\mathcal{L}$ with capacities cap_k^S.
- Select shelter locations, routes for buses and schedule for accommodation,
- such that all restrictions are kept
- and the evacuation time is minimized.

Modeling as ScheLoc-Routing Problem

- Identify shelters with machines,
- evacuees with jobs,
- accommodation time with processing time.

$\Rightarrow$ Evacuation time coincides with makespan

Modeling as ScheLoc-Routing Problem

- Identify shelters with machines,
- evacuees with jobs,
- accommodation time with processing time.

$\Rightarrow$ Evacuation time coincides with makespan

- Additional restrictions: capacities on machines, buses travel to different shelters, start at the depot
- possible routes: several gathering points, only one shelter

Scenarios

	Evacuees	Shelters	Buses [cap=50]
Scenario 1	725	3	7
Scenario 2	147	1	2
Scenario 3	976	3	7
Scenario 4	197	1	2
Scenario 5	378	2	3

Accommodation time: senior residents 2 minutes, adults 0 minutes, teenagers and children 1 minute

Computational Results

	Gap [%]	Time [s]
Scenario 1	7.7	459.8
Scenario 2	7.0	1.8
Scenario 3	14.7	619.0
Scenario 4	6.8	3.3
Scenario 5	8.0	52.2

Future Research

- Improve SGVNS

Future Research

- Improve SGVNS
- Construct different metaheuristics and compare results

Future Research

- Improve SGVNS
- Construct different metaheuristics and compare results
- Explore efficient exact methods

Thank you for your attention!