A hybrid heuristic based on self-organising maps and binary linear programming techniques for the capacitated p-median problem

Mike Steglich
Technical University of Applied Sciences Wildau

ECMS 2019

12 June 2019

The capacitated p-median problem (CPMP)

- The capacitated p-median problem (CPMP) is intended to find optimal locations of sources with a uniform supply which have to serve a set of destinations with particular demands in order to minimise the total transportation costs. (Daskin and Maass, 2015):

$$\sum_{(i,j)\in A} d_{ij} \cdot x_{ij} \rightarrow min$$

$$s.t.$$

$$\sum_{i\in N} x_{ij} = 1 \quad ; j \in N$$

$$\sum_{j\in N} q_j \cdot x_{ij} \leq Q \cdot y_i \quad ; i \in N$$

$$\sum_{i\in N} y_i = p$$

$$x_{ij} \in \{0,1\} \quad ; (i,j) \in A$$

$$y_i \in \{0,1\} \quad ; i \in N$$

Parameters:

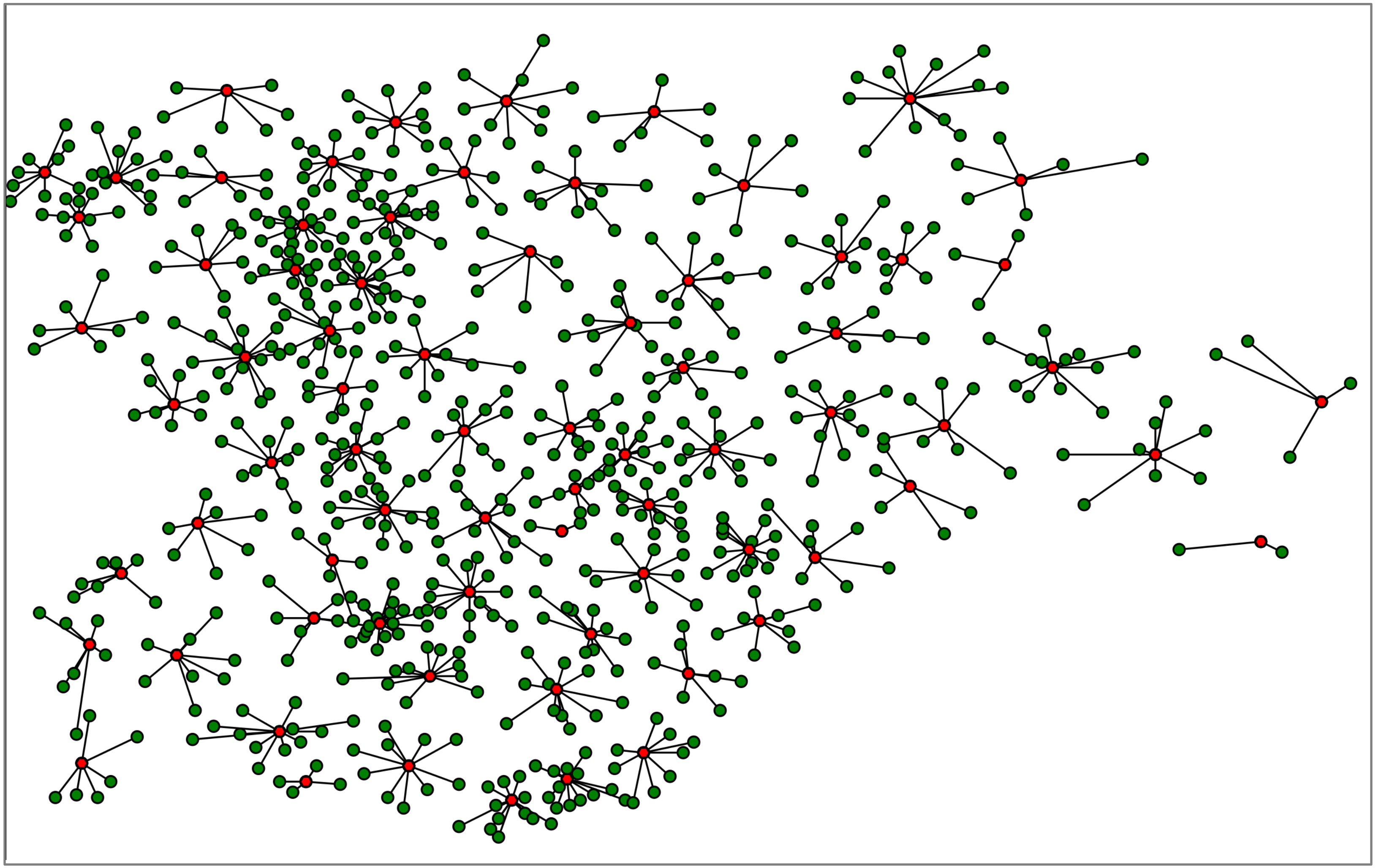
N	set of destination nodes (also potential candidates for the locations of the sources)
A	set of arcs joining the nodes, $A = \{(i,j) \mid i \in N, j \in N\}$
p	number of sources
q_j	demand of destination $j \in N$
Q	uniform supply of all sources
d_{ij}	distance between the nodes $i \in N$ and $j \in N$

Variables:

y_i	location variable for node $i \in N$
x_{ij}	allocation variable of destination $j \in N$ and source $i \in N$

The capacitated p-median problem (CPMP)

- 737 destinations and 74 sources



Literature review

Table 1: Objective function values for the *sjc* instances

Instance	Scheuerer and Wendolsky (2006)	Fleszar and Hindi (2008)	Chaves et al. (2007)	Boccia et al. (2008)	Stefanello et al. (2015)	Herda (2015)	Herda (2017)
sjc1	17,289	17,289	17,289	17,289	17,289	17,657	
sjc2	33,293	33,271	33,271	33,271	33,271	33,485	
sjc3a	45,338	45,335	45,335	45,335	45,335	45,880	
sjc3b	40,636	40,636	40,636	40,636	40,636	41,199	40,705
sjc4a	61,926	61,926	61,929	61,926	61,926	62,893	62,138
sjc4b	52,531	52,470	52,531	52,458	52,458	53,051	52,605
Average	41,836	41,821	41,832	41,819	41,819	42,361	

Table 2: Objective function values for the *spain* instances

Instance	Diaz and Fernandez (2006)	Stefanello et al. (2015)	Janosikova et al. (2017) GA (post-processing)
spain-737-74-1	8,967	8,876	8,959
spain-737-74-2	8,970	8,894	8,945
spain-737-148-1	6,012	5,917	5,913
spain-737-148-2	6,009	5,917	5,920
Average	7,490	7,401	7,434

Table 3: Objective function values for the *p*-3038 instances

Instance	Lorena and Senne (2004)	Stefanello et al. (2015)	Janosikova and Vasilovsky (2017) SGA	Janosikova and Vasilovsky (2017) GGA	Janosikova et al. (2017) GA (hypermutation)	Janosikova et al. (2017) GA (post-processing)
p-3038-600	122,021	122,725	125,929	125,392	125,638	126,010
p-3038-700	108,686	109,696	113,769	112,633	114,978	113,374
p-3038-800	98,531	100,084	105,633	104,208	105,484	105,004
p-3038-900	90,240	92,318	99,168	99,546	100,373	99,015
p-3038-1000	83,232	85,857	92,805	92,765	96,290	93,175
Average	100,542	102,136	107,461	106,909	108,553	107,315

Literature review

Table 1: Objective function values for the *sjc* instances

Instance	Scheuerer and Wendolsky (2006)	Fleszar and Hindi (2008)	Chaves et al. (2007)	Boccia et al. (2008)	Stefanello et al. (2015)	Herda (2015)	Herda (2017)
sjc1	17,289	17,289	17,289	17,289	17,289	17,657	
sjc2	33,293	33,271	33,271	33,271	33,271	33,485	
sjc3a	45,338	45,335	45,335	45,335	45,335	45,880	
sjc3b	40,636	40,636	40,636	40,636	40,636	41,199	40,705
sjc4a	61,926	61,926	61,929	61,926	61,926	62,893	62,138
sjc4b	52,531	52,470	52,531	52,458	52,458	53,051	52,605
Average	41,836	41,821	41,832	41,819	41,819	42,361	

Table 2: Objective function values for the *spain* instances

Instance	Diaz and Fernandez (2006)	Stefanello et al. (2015)	Janosikova et al. (2017) GA (post-processing)
spain-737-74-1	8,967	8,876	8,959
spain-737-74-2	8,970	8,894	8,945
spain-737-148-1	6,012	5,917	5,913
spain-737-148-2	6,009	5,917	5,920
Average	7,490	7,401	7,434

Table 3: Objective function values for the *p*-3038 instances

Instance	Lorena and Senne (2004)	Stefanello et al. (2015)	Janosikova and Vasilovsky (2017) SGA	Janosikova and Vasilovsky (2017) GGA	Janosikova et al. (2017) GA (hypermutation)	Janosikova et al. (2017) GA (post-processing)
p-3038-600	122,021	122,725	125,929	125,392	125,638	126,010
p-3038-700	108,686	109,696	113,769	112,633	114,978	113,374
p-3038-800	98,531	100,084	105,633	104,208	105,484	105,004
p-3038-900	90,240	92,318	99,168	99,546	100,373	99,015
p-3038-1000	83,232	85,857	92,805	92,765	96,290	93,175
Average	100,542	102,136	107,461	106,909	108,553	107,315

Literature review

Table 1: Objective function values for the *sjc* instances

Instance	Scheuerer and Wendolsky (2006)	Fleszar and Hindi (2008)	Chaves et al. (2007)	Boccia et al. (2008)	Stefanello et al. (2015)	Herda (2015)	Herda (2017)
sjc1	17,289	17,289	17,289	17,289	17,289	17,657	
sjc2	33,293	33,271	33,271	33,271	33,271	33,485	
sjc3a	45,338	45,335	45,335	45,335	45,335	45,880	
sjc3b	40,636	40,636	40,636	40,636	40,636	41,199	40,705
sjc4a	61,926	61,926	61,929	61,926	61,926	62,893	62,138
sjc4b	52,531	52,470	52,531	52,458	52,458	53,051	52,605
Average	41,836	41,821	41,832	41,819	41,819	42,361	

Table 2: Objective function values for the *spain* instances

Instance	Diaz and Fernandez (2006)	Stefanello et al. (2015)	Janosikova et al. (2017) GA (post-processing)
spain-737-74-1	8,967	8,876	8,959
spain-737-74-2	8,970	8,894	8,945
spain-737-148-1	6,012	5,917	5,913
spain-737-148-2	6,009	5,917	5,920
Average	7,490	7,401	7,434

Instance
p-3038-600
p-3038-700
p-3038-800
p-3038-900
p-3038-1000
Average

- As a result of an analysis of the objective function values and the computational times, it can be concluded that **IRMA** (Iterated Reduction Matheuristic Algorithm) proposed **by Stefanello et al.** satisfies the goal, to find a feasible solution with a good objective function value in a reasonable time, better than other recently published approaches.
- It is a hybrid heuristic, using local search and mathematical programming techniques (Stefanello et al., 2015).

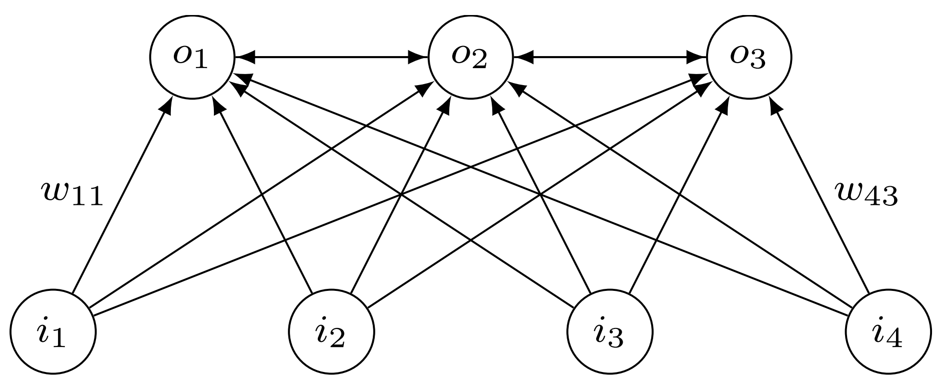
SomAla

- SomAla is a new hybrid heuristic for the capacitated p-median problem (CPMP) which combines a self-organising map (SOM), integer linear programming, an alternating location-allocation algorithm (ALA) and a partial neighbourhood optimisation.

  1. **SOM-GAP-based heuristic to solve a continuous, capacitated p-median problem**
     - A SOM is used to solve a continuous, uncapacitated p-median problem.
     - This solution is the basis to solve a continuous, capacitated p-median problem using a generalised assignment problem (GAP).
  2. **Capacitated alternating location-allocation heuristic**
     - The locations are modified by determining new medians for each allocation cluster.
     - The allocation of the destinations to the sources is based on a GAP which is solved by using a size reducing technique and variable relaxing-fixing heuristic.
  3. **Partial neighbourhood optimisation heuristic**
     - In each step, a CPMP for a subregion, surrounding a selected median, is solved.
     - If the solution improves the objective function value of the entire problem, then the partial solution updates the entire solution.

SOM-GAP-based heuristic to solve a continuous, capacitated p-median problem

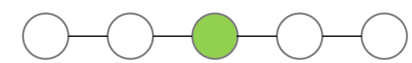
- **Self-organising maps**
  - A self-organising map (SOM) is a particular type of an artificial neural network introduced by Kohonen (1982).
  - A SOM is able to find a topographical projection $f : \mathbf{\Theta} \rightarrow \mathbf{\Omega}$, where $\mathbf{\Theta}$ describes L input patterns of attributes and $\mathbf{\Omega}$ a usually one- or two-dimensional representation of the input patterns (Kohonen, 1982).
  - A SOM can be described as a feed-forward network consisting of two layers of units. The I input units are completely connected with the O output units in the second layer. The weights on the edges between the input and output units can be formulated as a Matrix $\boldsymbol{w} = \{w_{mn} \in \mathbb{R} \mid m \in \{1,2, \dots, I\}, n \in \{1,2, \dots, O\}\}$.



SOM-GAP-based heuristic to solve a continuous, capacitated p-median problem

- **Self-organising maps**
  - The projection $f: \mathbf{\Theta} \rightarrow \mathbf{\Omega}$ has to be found during the so-called learning phase, which means that the following steps have to be carried out several times (Kohonen, 2001, p. 106).
    1. Initialisation: Initialise all weights by assigning random values.
    2. Propagation: Choose randomly an input pattern $\theta_l \in \mathbf{\Theta} = \{\theta_m | m \in \{1,2, \dots ,I\}\}; l \in \{1,2, \dots ,L\}$ and determine the winner unit η

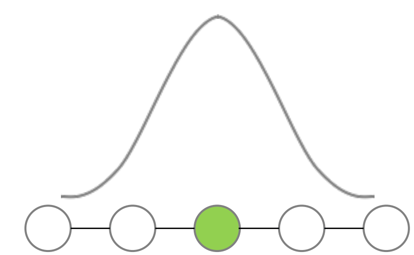
$$\eta = \arg \min_{n \in \{1,2,\dots,O\}} \|\boldsymbol{\theta}_l - \mathbf{w}_n\|^2$$



    3. Updating weights for the winner and its neighbourhood :

$$\Delta w_{mn} = \alpha \cdot \delta(n,\eta) \cdot (\theta_m - w_{mn}) \;\; ; m \in \{1,2,\dots,I\}, n \in \{1,2,\dots,O\}$$

$$\delta(n,\eta) = \exp\left(-\frac{(n-\eta)^2}{2\sigma^2}\right) ; n \in \{1,2,\dots,O\}$$



    4. Stop, if a stopping rule is satisfied or else continue with step 2.

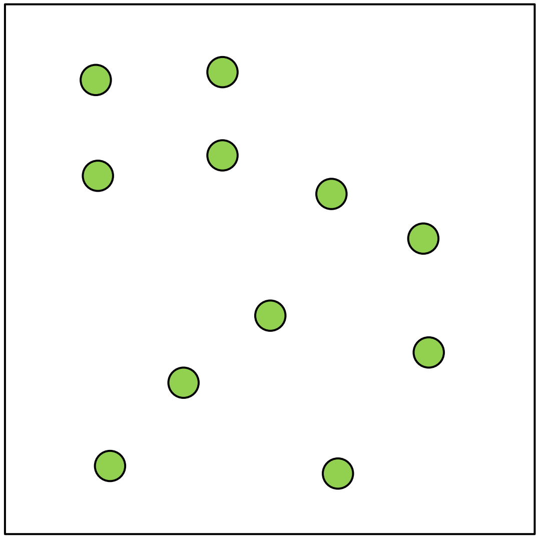
  - Afterwards, for all patterns $\theta_l \in \mathbf{\Theta}$ the corresponding outputs have to be determined by applying the winner-takes-all function:

$$\omega_l = \arg \min_{n \in \{1,2,\dots,O\}} \|\boldsymbol{\theta}_l - \mathbf{w}_n\|^2 ; l \in \{1,2,\dots,L\}$$

SOM-based approach to solve a continuous, uncapacitated p-median problem

- **SOM-based approach to solve a continuous, uncapacitated p-median problem**

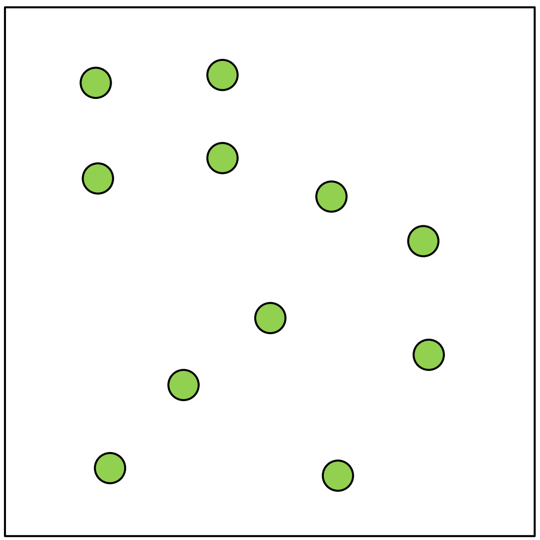
Map with the locations of the destina-tions



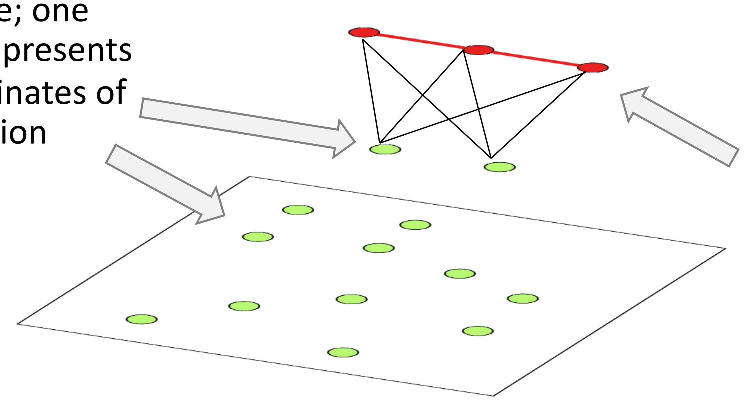
SOM-based approach to solve a continuous, uncapacitated p-median problem

- **SOM-based approach to solve a continuous, uncapacitated p-median problem**

Map with the locations of the destina- tions



One input unit per coordinate; one pattern represents the coordinates of a destination

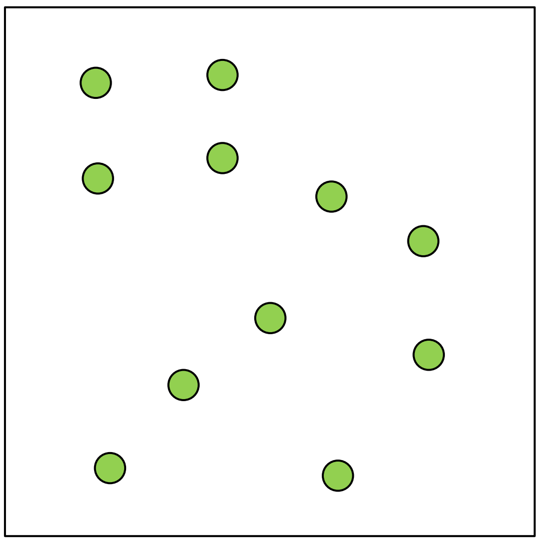


There are *p* output units, which represent areas of the map after the learning phase. The weights of the output units are the coordinates of the sources.

SOM-based approach to solve a continuous, uncapacitated p-median problem

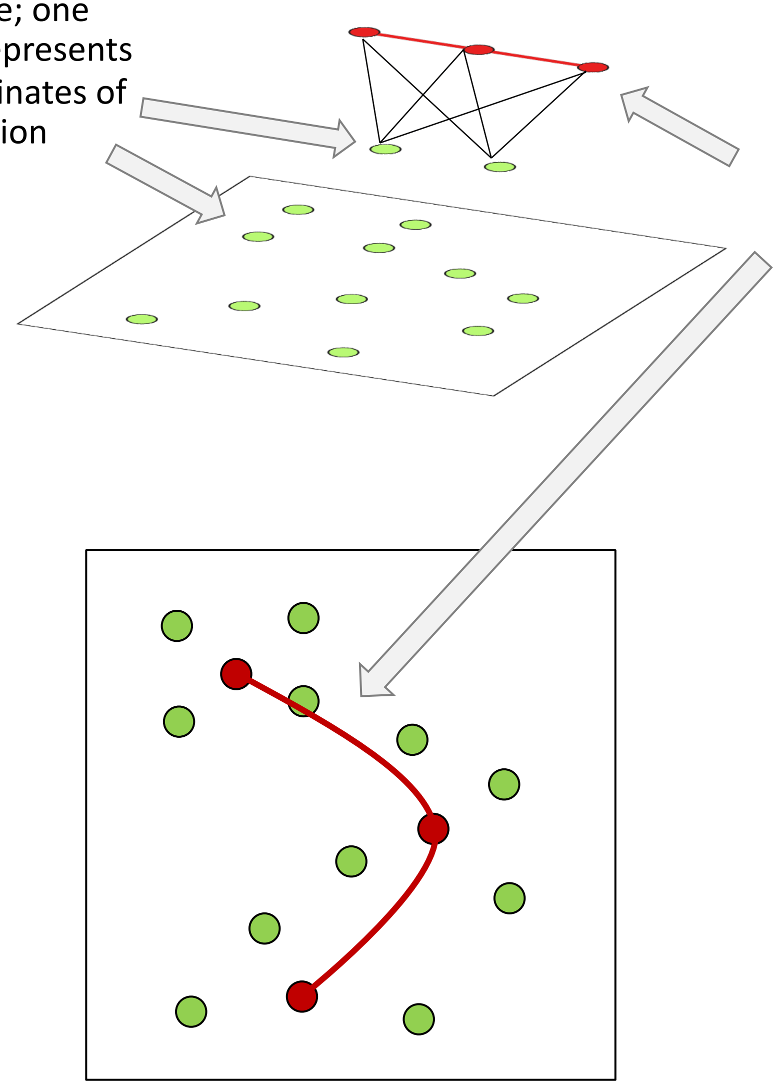
- **SOM-based approach to solve a continuous, uncapacitated p-median problem**

Map with the locations of the destina-tions



One input unit per coordinate; one pattern represents the coordinates of a destination

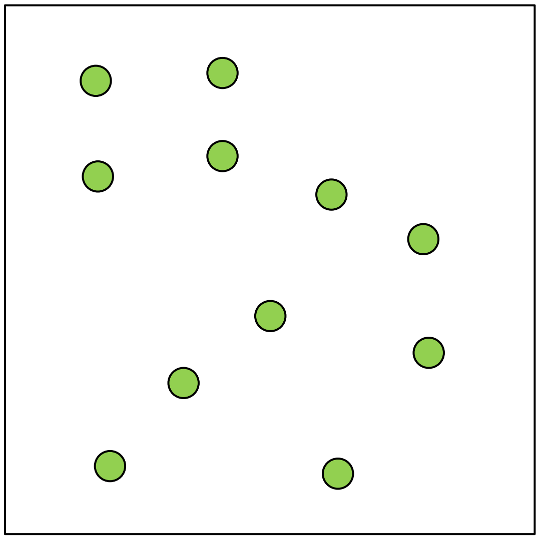
There are *p* output units, which represent areas of the map after the learning phase. The weights of the output units are the coordinates of the sources.



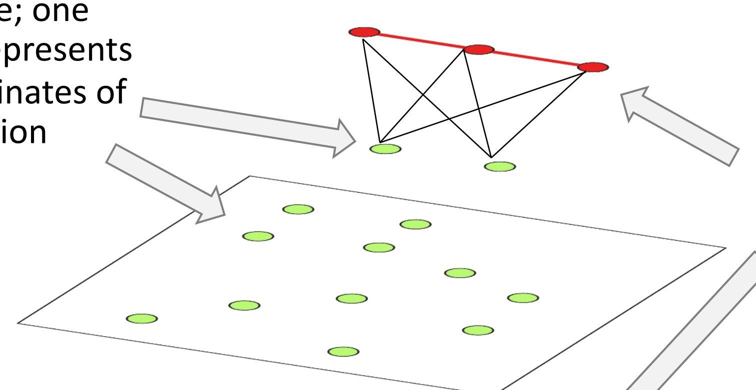
SOM-based approach to solve a continuous, uncapacitated p-median problem

- **SOM-based approach to solve a continuous, uncapacitated p-median problem**

Map with the locations of the destina-tions

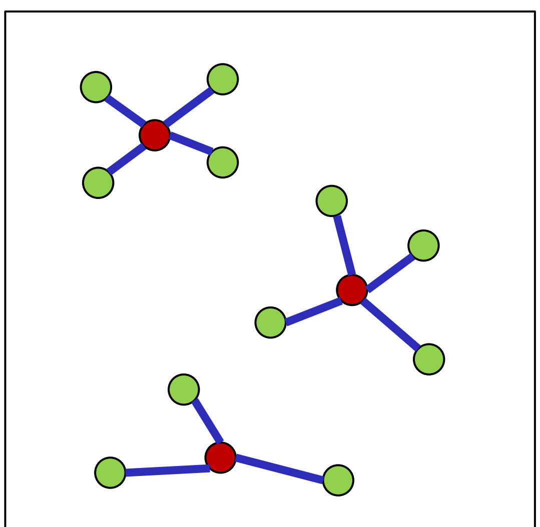


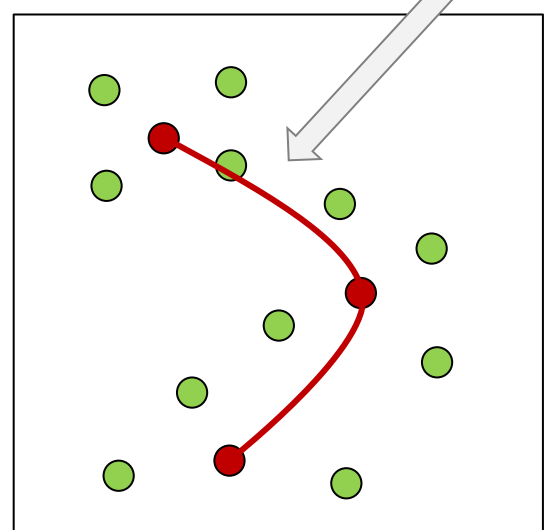
One input unit per coordinate; one pattern represents the coordinates of a destination



There are *p* output units, which represent areas of the map after the learning phase. The weights of the output units are the coordinates of the sources.

The winner unit per input (location of a demand node) represents the allocation of the destination to the source





SOM-based approach to solve a continuous, uncapacitated p-median problem

- **GAP-based approach to solve a continuous, capacitated p-median problem**
  - The activities of the allocation variables and the coordinates of the *p* sources found with a SOM are used to solve a GAP to assign the demand nodes to the sources considering the supply of the sources.
  - The mathematical model can be formulated as follows (Lorena and Senne, 2003)

$$\sum_{(k,j)\in\tilde{A}} d_{kj} \cdot x_{kj} \rightarrow min$$

$$s.t.$$

$$\sum_{\{k|(k,j)\in\tilde{A}\}} x_{kj} = 1 \quad ; j \in N$$

$$\sum_{\{j|(k,j)\in\tilde{A}\}} q_j \cdot x_{kj} \leq Q \quad ; k \in S$$

$$x_{kj} \in \{0,1\} \quad ; (k,j) \in \tilde{A}$$

Parameters:

S	set of source nodes
N	set of destinations
$\tilde{A}$	set of arcs joining the sources and the destinations
d_{kj}	distance between source $k \in S$ and destination $j \in D$
Q	supply

Variables:

x_{kj}	allocation variables; $k \in S, j \in D$

SOM-based approach to solve a continuous, uncapacitated p-median problem

- **GAP-based approach to solve a continuous, capacitated p-median problem**
  - To decrease the effort solving the GAP, a size reducing and a variable relaxing and fixing techniques are used.
  - **Size reduction:**
    - A source usually serves only demand nodes in its neighbourhood (Stefanello et al., 2015).
    - Therefore, all arcs between the sources and the destinations with distances which are greater than a defined radius $\bar{d}$ can be deleted: $\tilde{A} = \{(k,j) | k \in S, j \in N, d_{kj} \leq \bar{d}\}$.
  - **Variable relaxing and fixing techniques**:
    - In the first step, the problem is solved as a continuous, linear programme by relaxing the binary constraints.
    - In the second step, the GAP is solved as a binary, linear programme, whereby several variables x_{kj}, which have an activity $x^*_{kj} = 0$ in the solution of the continuous problem, are fixed to zero.
      - If the supply of a source $k \in S$ is completely shipped $\sum_{\{j|(k,j)\in\tilde{A}\}} q_j \cdot x^*_{kj} = Q$ then all unused relations $x^*_{kj} = 0; j \in N$ are fixed to zero.
      - For underutilised sources, an unused relation is uniformly fixed at random with a particular probability.
    - The set $\widetilde{A}$ of the directed arcs joining the sources and the destinations can be now determined as follows: $\tilde{A} = \{(k,j) | k \in S, j \in N, d_{kj} \leq \bar{d}, x_{kj}\ not\ fixed\}$.

Capacitated alternating location-allocation algorithm

- **Alternating location-allocation algorithm**: For each of the p clusters, built by the allocations of the demand nodes to the sources, a new median and therefore a new location of the particular source is recomputed. Afterwards, all demand nodes are re-allocated to the new medians. (Cooper, 1964, 1972; Lorena and Senne, 2003)

- **SomAla‘s capacitated Alternating Location-Allocation**:
  1. Find new medians for each of the p clusters. The median of a cluster (new discrete location of the source) is the node, that minimises the distances to all other nodes in this cluster.
  2. Solve a GAP to allocate the demand nodes to the sources found in step 1 in order to minimise the total distances.
  3. If the objective function value is improved then continue with step 1.

- For the allocations of the demand nodes to the sources, a GAP is solved as in the first SomAla step using the same size-reducing technique and the variable relaxing-fixing heuristic.

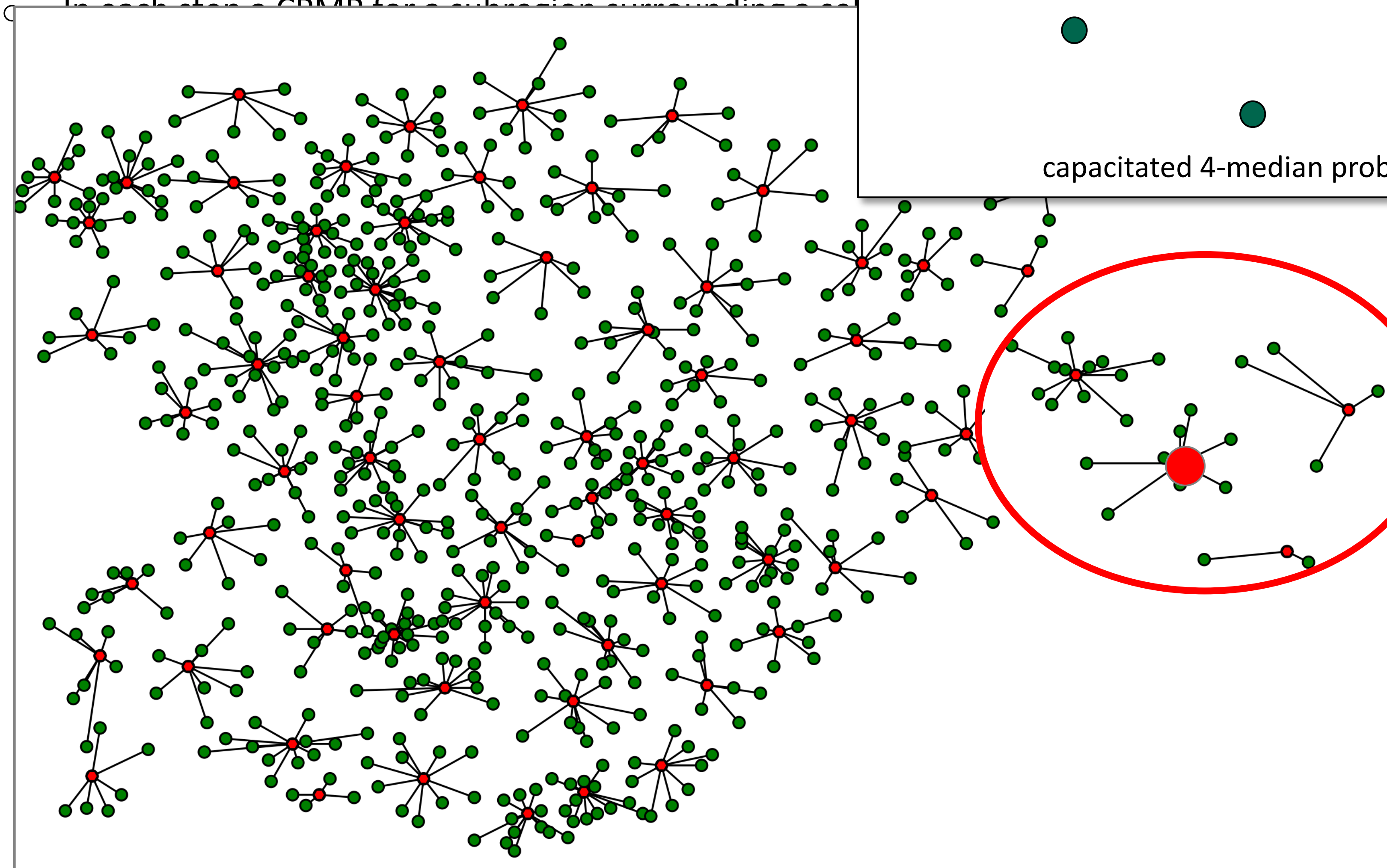
Partial neighbourhood optimisation heuristic

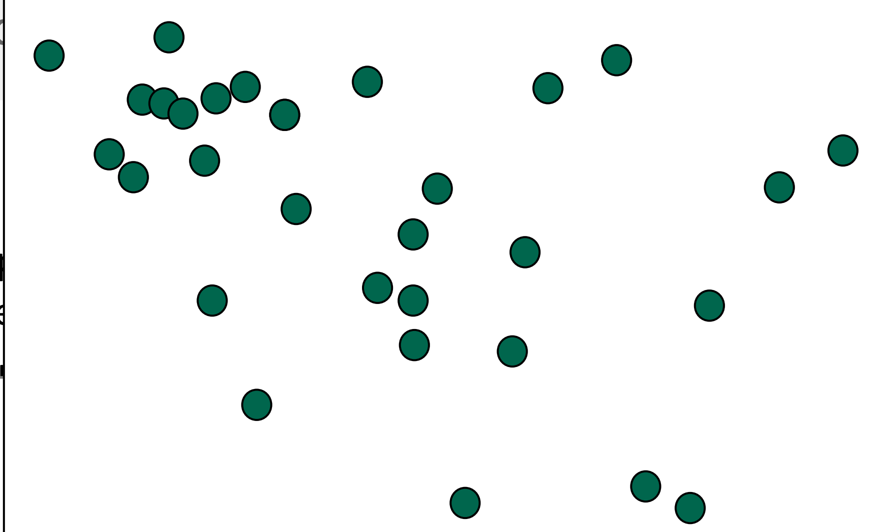
- The best solution found so far is to be improved by a partial neighbourhood optimisation heuristic, which is adapted from an approach proposed by Stefanello et al. (2015).
- In each step a CPMP for a subregion surrounding a selected median is solved. If the solution improves the objective function value of the entire problem, then the partial solution is used to update the entire solution.

1. Mark all sources $k \in S$ as non-optimised.
2. Select randomly a non-optimised source $\kappa \in S$.
3. Create a subregion
   - Set of sources $\tilde{S} \subset S$, which consists of κ and the z closest sources surrounding κ.
   - Set of demand nodes $\widetilde{N}$, which belong to the sources $\tilde{S}$.
   - Total distances C for the sources $\tilde{S}$ and the corresponding demand nodes $\widetilde{N}$.
4. Solve a CPMP for the demand nodes $\widetilde{N}$ and $\tilde{p} = |\tilde{S}|$ sources and obtain the objective function value $\tilde{C}$ for this sub-problem.
   - Either the first two SomAla working steps (*SomAla-MS*) or
   - a size-reduced CPMP (*SomAla-MM*) can be used to solve the CPMP.
5. If there is an improvement $\tilde{C} < \mathrm{C}$, then update the solution of the entire problem
   If there is no improvement $\tilde{C} \geq \mathrm{C}$, then mark the source κ as optimised.
6. If there exists at least one non-optimised source and a maximum number of no-improvements is not reached, then continue with step 2.

Partial neighbourhood optimisation heuristic

- The best solution found so far is to be improved by a
  is adapted from an approach proposed by Stefanello e
- In each step a CPMP for a subregion surrounding a sel





capacitated 4-median problem

nction value

ements is not

Partial neighbourhood optimisation heuristic

- The best solution found so far is to be improved by a partial neighbourhood optimisation heuristic, which is adapted from an approach proposed by Stefanello et al. (2015).
- In each step a CPMP for a subregion surrounding a selected median is solved. If the solution improves the objective function value of the entire problem, then the partial solution is used to update the entire solution.

1. Mark all sources $k \in S$ as non-optimised.
2. Select randomly a non-optimised source $\kappa \in S$.
3. Create a subregion
   - Set of sources $\tilde{S} \subset S$, which consists of κ and the z closest sources surrounding κ.
   - Set of demand nodes $\widetilde{N}$, which belong to the sources $\tilde{S}$.
   - Total distances C for the sources $\tilde{S}$ and the corresponding demand nodes $\widetilde{N}$.
4. Solve a CPMP for the demand nodes $\widetilde{N}$ and $\tilde{p} = |\tilde{S}|$ sources and obtain the objective function value $\tilde{C}$ for this sub-problem.
   - Either the first two SomAla working steps (*SomAla-MS*) or
   - a size-reduced CPMP (*SomAla-MM*) can be used to solve the CPMP.
5. If there is an improvement $\tilde{C} < \mathrm{C}$, then update the solution of the entire problem
   If there is no improvement $\tilde{C} \geq \mathrm{C}$, then mark the source κ as optimised.
6. If there exists at least one non-optimised source and a maximum number of no-improvements is not reached, then continue with step 2.

Computational results

- IRMA by Stefanello et al. is the most competitive approach to solve CPMPs and therefore the benchmark for SomAla.
- Six of the largest instances with which Stefanello et al. test the results of their algorithm Stefanello et al. (2015) are used.
- The results were obtained on a MacBook Pro with a Core i7-6820HQ CPU and 16GB RAM using Cplex 12.8 to solve linear programmes.
- The examples and the corresponding solution files are available at http://stegger.net/somala.

Instance	n	p	demands	supply per source	p/n ratio [%]	supply/ demand ratio [%]
u724-010	724	10	35,480	4175	1.4	117.7
u724-030	724	30	35,480	1479	4.1	125.1
u724-075	724	75	35,480	592	10.4	125.1
u724-125	724	125	35,480	355	17.3	125.1
u724-200	724	200	35,480	222	27.6	125.1
spain737-74-1	737	74	1,357,408	22930	10.0	125.0
spain737-74-2	737	74	236,888	4002	10.0	125.0
spain737-148-1	737	148	1,181,229	9977	20.1	125.0
spain737-148-2	737	148	208,330	1760	20.1	125.0
rl1304-010	1,304	10	65,126	7237	0.8	111.1
rl1304-050	1,304	50	65,126	1629	3.8	125.1
rl1304-100	1,304	100	65,126	815	7.7	125.1
rl1304-200	1,304	200	65,126	408	15.3	125.3
rl1304-300	1,304	300	65,126	272	23.0	125.3
pr2392-020	2,392	20	121,855	7616	0.8	125.0
pr2392-075	2,392	75	121,855	2031	3.1	125.0
pr2392-150	2,392	150	121,855	1016	6.3	125.1
pr2392-300	2,392	300	121,855	508	12.5	125.1
pr2392-500	2,392	500	121,855	305	20.9	125.1
p3038-600	3,038	600	154,482	321	19.7	124.7
p3038-700	3,038	700	152,913	273	23.0	125.0
P3038-800	3,038	800	152,693	238	26.3	124.7
P3038-900	3,038	900	155,824	216	29.6	124.8
p3038-1000	3,038	1,000	153,240	191	32.9	124.6
fnl4461-0020	4,461	20	224,395	14025	0.4	125.0
fnl4461-0100	4,461	100	224,395	2805	2.2	125.0
fnl4461-0250	4,461	250	224,395	1122	5.6	125.0
fnl4461-0500	4,461	500	224,395	561	11.2	125.0
fnl4461-1000	4,461	1,000	224,395	281	22.4	125.2

Objective function value comparison for SomAla and IRMA

Instance	BKS	Avg. objective function value			ObjGap [%]		
		IRMA	SomAla-MM	SomAla-MS	IRMA	SomAla-MM	SomAla-MS
u-724-010	181,783	182,611	182,588	182,382	0.46	0.44	0.33
u-724-030	95,034	95,160	95,824	95,344	0.13	0.83	0.33
u-724-075	54,735	54,735	54,867	55,626	0.00	0.24	1.63
u-724-125	38,977	38,977	39,088	39,808	0.00	0.28	2.13
u-724-200	28,080	28,083	28,292	28,881	0.01	0.76	2.85
spain-737-74-1	8,785	8,876	8,799	8,901	1.04	0.16	1.33
spain-737-74-2	8,870	8,894	8,926	8,996	0.27	0.63	1.42
spain-737-148-1	5,879	5,902	5,886	6,019	0.39	0.12	2.37
spain-737-148-2	5,914	5,917	5,960	6,077	0.05	0.78	2.75
rl-1304-010	2,146,252	2,166,552	2,148,798	2,149,366	0.95	0.12	0.15
rl-1304-050	802,283	806,425	804,258	804,254	0.52	0.25	0.25
rl-1304-100	498,091	498,412	500,294	502,275	0.06	0.44	0.84
rl-1304-200	276,978	276,984	279,339	280,879	0.00	0.85	1.41
rl-1304-300	191,225	191,259	192,440	195,690	0.02	0.64	2.33
pr-2392-020	2,231,213	2,250,292	2,236,889	2,235,790	0.86	0.25	0.21
pr-2392-075	1,091,983	1,098,560	1,097,700	1,092,917	0.60	0.52	0.09
pr-2392-150	711,111	711,315	714,315	715,719	0.03	0.45	0.65
pr-2392-300	458,145	458,222	460,720	462,726	0.02	0.56	1.00
pr-2392-500	316,043	316,092	317,466	323,348	0.02	0.45	2.31
p-3038-600	122,021	122,725	123,152	124,558	0.58	0.93	2.08
p-3038-700	108,686	109,696	110,034	111,653	0.93	1.24	2.73
p-3038-800	98,531	100,084	100,351	102,324	1.58	1.85	3.85
p-3038-900	90,240	92,318	92,942	95,417	2.30	2.99	5.74
p-3038-1000	83,232	85,857	86,315	88,501	3.15	3.70	6.33
fnl-4461-0020	1,283,537	1,292,622	1,284,555	1,284,037	0.71	0.08	0.04
fnl-4461-0100	548,845	550,758	551,031	549,126	0.35	0.40	0.05
fnl-4461-0250	335,889	336,007	336,901	337,205	0.04	0.30	0.39
fnl-4461-0500	224,662	224,684	225,423	226,283	0.01	0.34	0.72
fnl-4461-1000	145,862	145,871	146,577	148,454	0.01	0.49	1.78
Minimum					0.00	0.08	0.04
Average	420,412	422,893	421,966	422,951	0.52	0.73	1.66
Maximum					3.15	3.70	6.33

Objective function value comparison for SomAla and IRMA

Instance	BKS	Avg. objective function value			ObjGap [%]		
		IRMA	SomAla-MM	SomAla-MS	IRMA	SomAla-MM	SomAla-MS
u-724-010	181,783	182,611	182,588	182,382	0.46	0.44	0.33
u-724-030	95,034	95,160	95,824	95,344	0.13	0.83	0.33
u-724-075	54,735	54,735	54,867	55,626	0.00	0.24	1.63
u-724-125	38,977	38,977	39,088	39,808	0.00	0.28	2.13
u-724-200	28,080	28,083	28,292	28,881	0.01	0.76	2.85
spain-737-74-1	8,785	8,876	8,799	8,901	1.04	0.16	1.33
spain-737-74-2	8,870	8,894	8,926	8,996	0.27	0.63	1.42
spain-737-148-1	5,879	5,902	5,886	6,019	0.39	0.12	2.37
					0.05	0.78	2.75
					0.95	0.12	0.15
					0.52	0.25	0.25
					0.06	0.44	0.84
					0.00	0.85	1.41
					0.02	0.64	2.33
					0.86	0.25	0.21
					0.60	0.52	0.09
					0.03	0.45	0.65
					0.02	0.56	1.00
					0.02	0.45	2.31
					0.58	0.93	2.08
					0.93	1.24	2.73
					1.58	1.85	3.85
					2.30	2.99	5.74
					3.15	3.70	6.33
					0.71	0.08	0.04
					0.35	0.40	0.05
					0.04	0.30	0.39
					0.01	0.34	0.72
					0.01	0.49	1.78
					0.00	0.08	0.04
Average	420,412	422,893	421,966	422,951	0.52	0.73	1.66
Maximum					3.15	3.70	6.33

ObjGap	IRMA[%]	SomAla-MM[%]	SomAla-MS[%]
$\leq 1\%$	86.2	86.2	44.8
$\leq 2\%$	93.1	93.1	62.1
$\leq 3\%$	96.6	96.6	89.7
$\leq 4\%$	100.0	100.0	93.1

$$GAP = \left(\frac{Z}{BKS} - 1\right) \cdot 100$$

Z objective function values
BKS best known solution BKS

Runtime comparison for SomAla and IRMA

- Since the tests for IRMA and SomAla were run on different test environments, the original computational times cannot be compared.
- Therefore, estimations of the computational times for IRMA, when this algorithm would run on SomAla's test environment, were generated. These estimations depend
  - on benchmark-based assumptions about time savings of different hardware as well as
  - assumptions about release-wise runtime improvements of CPLEX.

- **Selected Benchmarks for CPUs** used for IRMA (i5-2300) and for SomAla (i7-6820HQ) (PassMark,2018b,a; UserBenchmark, 2018; GeekBenchmark, 2018)

Benchmark	Intel Core i5-2300	Core i7-6820HQ	Ratio i7/i5 [%]
PassMark			
single-thread	1,577	1,884	84
multi-thread	5,345	8,798	61
UserBenchmark.com			
single-thread	77	93	82
multi-thread	288	483	60
GeekBench			
single-thread	2,890	3,991	72
multi-thread	7,975	12,787	62
Avg. single-thread			80
Avg. muti-thread			61

Runtime comparison for SomAla and IRMA

- **CPLEX assumptions:**
  - Both algorithms use CPLEX to solve linear programmes as part of the algorithm.
    - SomAla - CPLEX 12.8
    - IRMA - CPLEX 12.3
  - It is assumed that each major release provides a 15 % time saving for the CPMP. (Several own tests of the changes of the runtimes between CPLEX 12.7 and CPLEX 12.8 of selected size-reduced CPMP instances have observed an improvement of only 3.43 %.)

- **IRMA runtime estimation assumptions:**
  - It is assumed that 10 % of IRMA's algorithm runs in a single-thread mode and 90 % is multi-threaded.

$$62\% = \lfloor 0.1 \cdot 0.80 + 0.9 \cdot 0.61 \rfloor \cdot 100$$

  - Furthermore, it is assumed that CPLEX causes 90 % of IRMA's entire computational runtime.

$$30\% = \lfloor 0.1 \cdot 0.62 + 0.9 \cdot 0.62 \cdot (1 - 0.15)^5 \rfloor \cdot 100$$

  - IRMA would only need 30 % of its original computational time, if SomAla's test environment and CPLEX 12.8 was used.

Runtime comparison for SomAla and IRMA

Instance	Computational time [sec]			Ratio SomAla/IRMA [%]	
	IRMA	SomAla-MM	SomAla-MS	SomAla-MM	SomAla-MS
u-724-010	59.65	0.79	0.98	1.33	1.64
u-724-030	300.72	2.42	6.63	0.80	2.20
u-724-075	546.39	109.65	14.68	20.07	2.69
u-724-125	643.31	106.59	25.99	16.57	4.04
u-724-200	706.29	79.96	34.53	11.32	4.89
spain-737-74-1	1,131.32	175.84	23.89	15.54	2.11
spain-737-74-2	979.12	128.99	51.73	13.17	5.28
spain-737-148-1	652.34	134.56	67.78	20.63	10.39
spain-737-148-2	653.79	134.62	98.24	20.59	15.03
rl-1304-010	181.66	15.29	1.63	8.42	0.90
rl-1304-050	1,199.98	15.43	14.70	1.29	1.23
rl-1304-100	1,634.18	208.39	35.04	12.75	2.14
rl-1304-200	1,227.78	140.46	71.41	11.44	5.82
rl-1304-300	951.75	165.33	117.68	17.37	12.36
pr-2392-020	551.82	6.05	6.07	1.10	1.10
pr-2392-075	825.85	24.83	51.11	3.01	6.19
pr-2392-150	2,019.23	0.00	118.98	26.01	5.89
pr-2392-300	2,382.39	266.80	171.14	11.20	7.18
pr-2392-500	2,402.6	547.65	232.99	22.79	9.70
p-3038-600	2,685.38	549.00	252.85	20.44	9.42
p-3038-700	2,239.84	625.67	327.76	27.93	14.63
p-3038-800	2,819.26	765.91	462.45	27.17	16.40
p-3038-900	1,578.17	439.31	380.47	27.84	24.11
p-3038-1000	1,874.08	478.40	405.19	25.53	21.62
fnl-4461-0020	538.97	38.31	21.24	7.11	3.94
fnl-4461-0100	3,880.37	110.59	92.58	2.85	2.39
fnl-4461-0250	4,592.66	1,239.09	178.86	26.98	3.89
fnl-4461-0500	3,912.36	665.33	286.88	17.01	7.33
fnl-4461-1000	3,433.26	662.38	575.98	19.29	16.78
Minimum				0.80	0.90
Average	1.607.05	270.26	142.39	15.09	7.63
Maximum				27.93	24.11

Runtime comparison for SomAla and IRMA

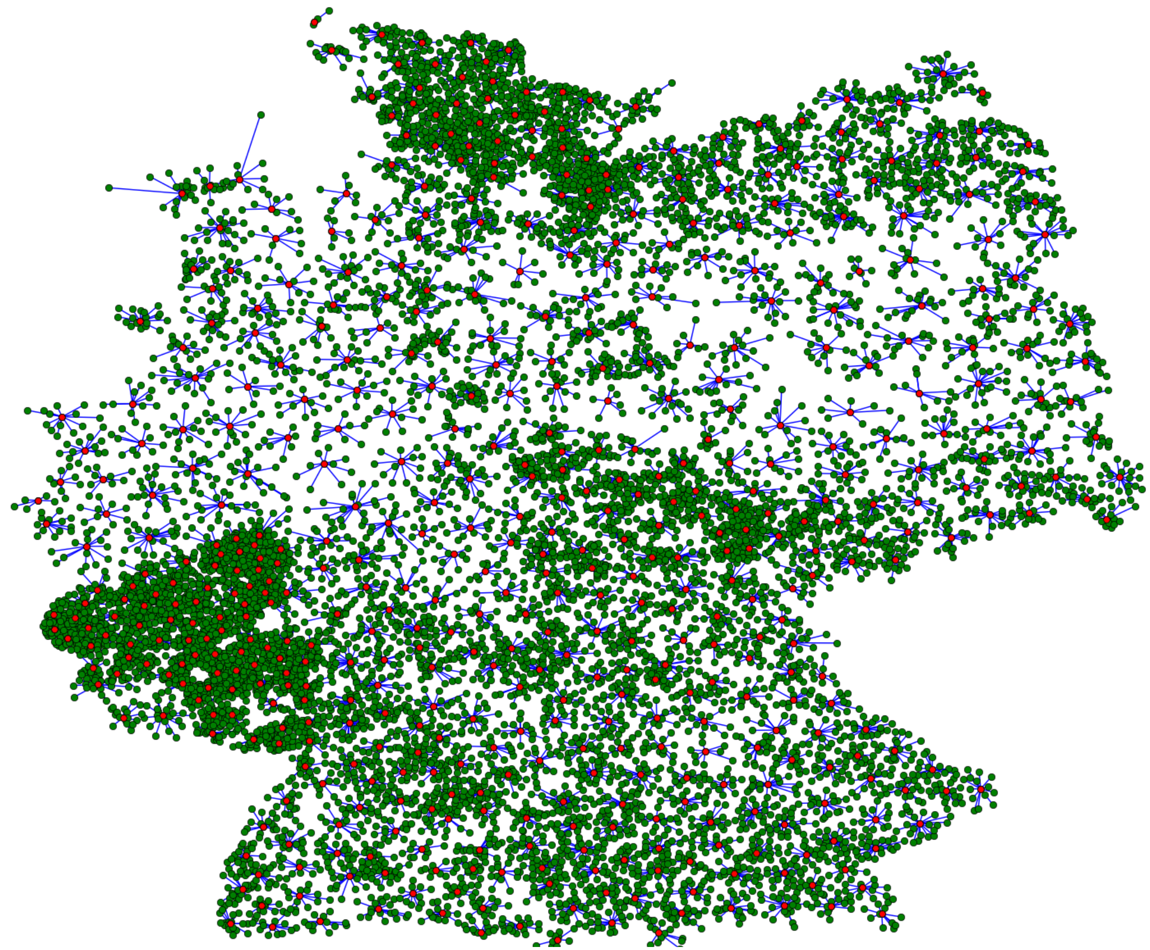
Instance	Computational time [sec]			Ratio SomAla/IRMA [%]	
	IRMA	SomAla-MM	SomAla-MS	SomAla-MM	SomAla-MS
u-724-010	59.65	0.79	0.98	1.33	1.64
u-724-030	300.72	2.42	6.63	0.80	2.20
u-724-075	546.39	109.65	14.68	20.07	2.69
u-724-125	643.31	106.59	25.99	16.57	4.04
u-724-200	706.29	79.96	34.53	11.32	4.89
spain-737-74-1	1,131.32	175.84	23.89	15.54	2.11
spain-737-74-2	979.12	128.99	51.73	13.17	5.28
spain-737-148-1	652.34	134.56	67.78	20.63	10.39
spain-737-148-2	653.79	134.62	98.24	20.59	15.03
[illegible]	[illegible]	[illegible]	[illegible]	8.42	0.90
[illegible]	[illegible]	[illegible]	[illegible]	1.29	1.23
[illegible]	[illegible]	[illegible]	[illegible]	12.75	2.14
[illegible]	[illegible]	[illegible]	[illegible]	11.44	5.82
[illegible]	[illegible]	[illegible]	[illegible]	17.37	12.36
[illegible]	[illegible]	[illegible]	[illegible]	1.10	1.10
[illegible]	[illegible]	[illegible]	[illegible]	3.01	6.19
[illegible]	[illegible]	[illegible]	[illegible]	26.01	5.89
[illegible]	[illegible]	[illegible]	[illegible]	11.20	7.18
[illegible]	[illegible]	[illegible]	[illegible]	22.79	9.70
[illegible]	[illegible]	[illegible]	[illegible]	20.44	9.42
[illegible]	[illegible]	[illegible]	[illegible]	27.93	14.63
p-3038-800	2,819.26	765.91	462.45	27.17	16.40
p-3038-900	1,578.17	439.31	380.47	27.84	24.11
p-3038-1000	1,874.08	478.40	405.[illegible]	25.53	21.62
fnl-4461-0020	538.97	38.31	21.24	7.11	3.94
fnl-4461-0100	3,880.37	110.59	92.58	2.85	2.39
fnl-4461-0250	4,592.66	1,239.09	178.86	26.98	3.89
fnl-4461-0500	3,912.36	665.33	286.88	17.01	7.33
fnl-4461-1000	3,433.26	662.38	575.98	19.29	16.78
Minimum				0.80	0.90
Average	1.607.05	270.26	142.39	15.09	7.63
Maximum				27.93	24.11

Minimum	0.80	0.90
Average	15.09	7.63
Maximum	27.93	24.11

IRMA's runtime estimations could be undercut by 100 % of the instances solved with SomAla.

g11056 instance

- The g11056 instance is the largest CPMP instance which has been ever solved and published.
- The demand nodes are the 11,056 German cities and communities published by the Statistisches Bundesamt (2017) with their geographical coordinates.
- Four instances with 111 up to 3317 sources
- These instances show that SomAla is able to find good feasible solutions in reasonable computational times (e.g. ~3,500 seconds for for g11056-3317) for very large instances.



References

Aras,N., Özkisacik,K.C., Altinel,I.K., 2006.Solvingthe Uncapacitated Multi-Facility Weber Problem by Vector Quantization and Self-Organizing Maps. *The Journal of the Operational Research Society* 57, 1, 82–93.

Boccia, M., Sforza, A., Sterle, C., Vasilyev, I., 2008. A Cut and Branch Approach for the Capacitated p-Median Problem Based on Fenchel Cutting Planes. *Journal of Mathematical Modelling and Algorithms* 7, 1, 43–58.

Chaves, A.A., de Assis Correa, F., Lorena, L.A.N., 2007. Clustering Search Heuristic for the Capacitated p-Median Problem. In Corchado, E., Corchado, J.M. and Abraham, A. (eds), *Innovations in Hybrid Intelligent Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 136–143.

Cooper, L., 1964. Heuristic Methods for Location- Allocation Problems. *SIAM Review* 6, 1, 37–53.

Cooper, L., 1967. Solutions of Generalized Locational Equilibrium Models. *Journal of Regional Science* 7 (1), 1–18.

Cooper, L., 1972. The Transportation-Location Problem. *Operations Research* 20, 1, 94–108.

Daskin, M.S., Maass, K.L., 2015. The p-Median Problem. In Laporte, G., Nickel, S. and Saldanha da Gama, F. (eds), *Location Science*. Springer, Cham, Heidelberg, New York, Dordrecht, London, book section 2, pp. 21– 45.

Diaz, J., Fernandez, E., 2006. Hybrid Scatter Search and Path Relinking for the capacitated p-median problem. *European Journal of Operational Research* 169, 570– 585.

Fleszar, K., Hindi, K.S., 2008. An Effective VNS for the Capacitated p-median Problem. *European Journal of Operational Research* 191, 3, 612–622.

References

GeekBenchmark, 2018. GeekBenchmark. https://browser. geekbench.com/processor-benchmarks. Accessed: 11 January 2018.

Herda, M., 2015. Combined genetic algorithm for capaci- tated p-median problem. In *Computational Intelligence and Informatics (CINTI), 2015 16th IEEE International Symposium on*, IEEE, pp. 151–154.

Herda, M., 2017. Parallel genetic algorithm for capacitated p-median problem. In *TRANSCOM 2017: International scientific conference on sustainable, modern and safe transport*, Procedia Engineering, p. 313 317.

Hsieh, K.H., Tien, F.C., 2004. Self-organizing feature maps for solving locationallocation problems with rectilin- ear distances. *Computers & Operations Research* 31, 10171031.

Janosikova, L., Vasilovsky, P., 2017. Grouping Genetic Algorithm for the Capacitated p-median Problem. In *Inter- national Conference on Information and Digital Technologies,*, pp. 152–159.

Janosikova, L., Herda, M., Haviar, M., 2017. Hybrid Genetic Algorithms with Selective Crossover for the Ca- pacitated p-median Problem. *Central European Journal of Operations Research* 25, 1, 651664.

Kohonen, T., 1982. Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics* 43, 59–69.

Kohonen, T., 2001. *Self-Organizing Maps* (3 edn.). Springer, Berlin et al.

Lorena, L.A.N., Pereira, M.A., Salomao, S.N.A., 2003. A relaxacao lagrangeana/surrogate e o metodo de geracao de colunas: novos limitantes e novas colunas. *Pesquisa Operacional* 23, 29 – 47.

References

Lorena, L.A.N., Senne, E.L.F., 2003. Local Search Heuris- tics For Capacitated P-Median Problems. *Networks and Spatial Economics* 3, 407–419.

Lorena, L.A.N., Senne, E.L.F., 2004. A column generation approach to capacitated p-median problems. *Computers & Operations Research* 31, 6, 863–876.

Lozano, S., Guerrero, F., Onieva, L., Larraneta, J., 1998. Kohonen maps for solving a class of location-allocation problems. *European Journal of Operational Research* 108, 106–117.

Mulvey, J., Beck, M., 1984. Solving Capacitated Clustering Problems. *European Journal of Operational Research* 18, 339–348.

PassMark, 2018a. PassMark CPU Benchmark - Multiple CPU Systems. https://www.cpubenchmark.net/multi cpu.html. Accessed: 11 January 2018.

PassMark, 2018b. PassMark CPU Benchmark - Single Thread performance. https://www.cpubenchmark.net/ singleThread.html. Accessed: 11 January 2018.

Reinelt, G., 1994. *The Traveling Salesman: Computational Solutions for TSP Applications*. Springer-Verlag, Berlin, Heidelberg.

Scheuerer, S., Wendolsky, R., 2006. A scatter search heuristic for the capacitated clustering problem. *European Journal of Operational Research* 169, 2, 533 – 547. Feature Cluster on Scatter Search Methods for Optimization.

References

Statistisches Bundesamt, 2017. Statistisches Bundesamt, Gemeinden in Deutschland nach Fläche, Bevölkerung und Postleitzahl am 31.03.2017 (1. Quartal). https: //www.destatis.de/DE/ZahlenFakten/LaenderRegionen/ Regionales/Gemeindeverzeichnis/Administrativ/ Archiv/GVAuszugQ/AuszugGV1QAktuell.xlsx?blob=publicationFile. Accessed: 20 September 2017.

Stefanello, F., Arajo, O.C.B.d., Mller, F.M., 2015. Matheuristics for the capacitated p-median problem. *International Transactions in Operational Research* 22, 1, 149–167.
UserBenchmark, 2018. UserBenchmark.

http://cpu.userbenchmark.com/Compare/ Intel-Core-i5-2300-vs-Intel-Core-i7-6820HQ/ m291vsm43500. Accessed: 11 January 2018.

Thank you for your attention.

If you have any questions, please feel free to ask.