A HYBRID HEURISTIC BASED ON SELF-ORGANISING MAPS AND BINARY LINEAR PROGRAMMING TECHNIQUES FOR THE CAPACITATED P-MEDIAN PROBLEM

Mike Steglich
University of Applied Sciences Wildau
Hochschulring 1, 15745 Wildau, Germany
Email: mike.steglich@th-wildau.de

KEYWORDS

capacitated p-median problem, heuristic, self-organising maps, alternating location-allocation algorithm, binary linear programming, partial neighbourhood optimisation

ABSTRACT

This paper proposes a new hybrid heuristic (SomAla) 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. To improve the performance of the algorithm, the structure of the CPMP is exploited for several size-reduction methods and also for variable-fixing techniques. The capability of this algorithm to find good solutions in reasonable times for large problem instances has been tested on several benchmark instances.

INTRODUCTION

The capacitated p-median problem (CPMP) is a well-known model intended to find optimal locations of sources which have to serve a set of demand nodes in order to minimise the total distances between the sources and the destinations. It can be described by using a network $G = (N,A)$ with a set N of demand nodes and a set $A = \{(i,j) \mid i \in N, j \in N\}$ of directed arcs joining the nodes. Each node, for which a particular demand $q_j; j \in N$ exists, is a potential candidate for the location of one of p sources. If a source is located at the demand node $i \in N$, then the binary variable $y_i; i \in N$ equals one. All p sources have an identical supply Q. The weights of the arcs represent the distances $d_{ij}; (i,j) \in A$ between the nodes $i \in N$ and $j \in N$. A matrix of binary variables $x_{ij} \in \{0,1\}; (i,j) \in A$ is needed for the decision of allocating each demand node to exactly one source. The variable x_{ij} equals one, if the demand node j is allocated to a source located at the node i or $x_{ij} = 0$ if not so. The mathematical model can be formulated as follows (Daskin and Maass, 2015):

$$\sum_{(i,j)\in A} d_{ij} \cdot x_{ij} \rightarrow min \tag{1}$$

$s.t.$

$$\sum_{i\in N} x_{ij} = 1 \quad ; j \in N \tag{2}$$

$$\sum_{j\in N} q_j \cdot x_{ij} \leq Q \cdot y_i \quad ; i \in N \tag{3}$$

$$\sum_{i\in N} y_i = p \tag{4}$$

$$x_{ij} \in \{0,1\} \quad ; (i,j) \in A \tag{5}$$

$$y_i \in \{0,1\} \quad ; i \in N \tag{6}$$

The objective is to minimise the total distances as per (1). A demand node $j \in N$ can only be served by exactly one source $i \in N$ due to constraints (2). Constraints (3) ensure, that a demand node can only assigned to an established source ($y_i = 1$) and that the outgoing flows of a source $i \in N$ cannot exceed the supply Q. Exact p sources have to be established due to constraint (4).

This paper proposes a new hybrid heuristic which combines a self-organising map (SOM), integer linear programming, an alternating location-allocation algorithm (ALA) and a partial neighbourhood algorithm.

One novelty of this approach is the combination of a SOM and a generalised assignment problem (GAP) in the first step of the algorithm. This approach exploits the structure of the CPMP, because the first clusters of demand nodes found by the SOM define a neighbourhood in which the optimal locations of the sources in the CPMP can potentially be expected. A first capacitated solution as basis for the next working steps are found by solving a GAP using these first clusters. The next working steps combine several known techniques (integer linear programming, alternating location-allocation algorithm, partial neighbourhood optimisation) in a new manner. All of these approaches are extended by several size-reduction methods and a variable relaxing-fixing heuristic to improve the capability of the entire algorithm to find good solutions for large problem instances as fast as possible.

This paper starts with a literature review, followed by the description of the proposed algorithm. The capability of this algorithm to find good solutions in reasonable times for large problem instances has been tested on several benchmark instances. The computational results are reported in the next section. The paper is finished with conclusions.

LITERATURE REVIEW

This literature review is only focused on the results of recently published CPMP algorithms for which the test results of three groups of problem instances are given in tables 1, 2 and 3. A good overview of the CPMP-related literature can be found in Lorena et al. (2003).

Lorena and Senne (2003) introduced the *sjc* instances, which are derived from a geographic database of the Brazilian City Sao Jose dos Campos. Diaz and Fernandez (2006) presented the *spain* instances, which contain 737 Spanish cities as demand nodes, that have to be served by 74 or 148 sources. The instances *p*-3038 were introduced by Lorena et al. (2003), whereby they adapted the TSPLIB instance *pcb*3038 (Reinelt, 1994) by including from 600 up to 1000 sources.

As shown in table 1, the approaches by Boccia et al. (2008) and Stefanello et al. (2015) provide the best objec-

Communications of the ECMS, Volume 33, Issue 1,
Proceedings, ©ECMS Mauro Iacono, Francesco Palmieri,
Marco Gribaudo, Massimo Ficco (Editors)
ISBN: 978-3-937436-65-4/978-3-937436-66-1(CD) ISSN 2522-2414

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

tive function values for the *sjc* instances. These objective function values are equal to the optimal solutions. Both algorithms are able to find these solutions in a reasonable time. It is in general difficult to compare runtimes of different algorithms when the results were obtained on different computer systems. Stefanello et al. (2015) tried to solve this problem by comparing the relative time gaps between the times needed to solve the original CPMP exactly and to solve the problem by using the heuristic. It seems, by comparing these gaps, that their algorithm has a better performance compared to Boccia et al. (2008) (Stefanello et al., 2015).

Table 2 shows the results of the *spain* instances. The averages of the objective function values found by Stefanello et al. (2015) are slightly better compared to Diaz and Fernandez (2006) and Janosikova et al. (2017). Stefanello et al. (2015) analysed the computational times between their algorithm and the approach by Diaz and Fernandez (2006). It seems, even when different hardware is considered, that the heuristic by Stefanello et al. (2015) provides a better performance with an average computational time of 854 seconds compared to the 32,008 seconds needed to solve these instances with the algorithm by Diaz and Fernandez (2006). There could also be an improvement compared to Janosikova et al. (2017), because they always let their genetic algorithm run with a stopping criterion of 3,600 seconds to find the best solutions (Janosikova et al., 2017).

Table 3 shows that the column generation approach by Lorena and Senne (2004) provides the best objective function values for all *p*-3038 instances followed by Stefanello et al. (2015). But it seems, even considering the different hardware, that Stefanello et al. (2015) produce the solutions faster, because the average computational time is only 2,239 seconds compared to 34,739 seconds for the approach by Lorena and Senne (2004). All other approaches proposed by Janosikova and Vasilovsky (2017) and values Janosikova et al. (2017) achieve poorer objective function values. The computational times of these approaches seem similar to Stefanello et al. (2015), since they use a stopping criterion of 3,600 seconds for these algorithms (Janosikova and Vasilovsky, 2017; Janosikova et al., 2017).

It can be summarised according to all of these results that the algorithm proposed by Stefanello et al. (2015) is the most competitive approach of the recently published algorithms. It is a hybrid heuristic, using local search and mathematical programming techniques and is called IRMA (Iterated Reduction Matheuristic Algorithm) (Stefanello et al., 2015).

PROPOSED ALGORITHM: SOMALA

Overview

SomAla is a new hybrid heuristic which combines a self-organising map, integer-programming, an alternating location-allocation algorithm and a partial neighbourhood optimisation heuristic. This section describes the proposed algorithm in general with the following three working steps.

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. The locations of the sources and the allocations of the demand nodes

are the basis to find a solution for a continuous, capacitated p-median problem by solving a GAP.

2. Capacitated alternating location-allocation heuristic: The solution found in the first step is used in a capacitated ALA heuristic to find and improve a solution for the CPMP. 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. The locations are modified by determining new medians for each allocation cluster. Both steps are repeated as long as improvements for the CPMP occur or a maximum number of steps is not reached.
3. Partial neighbourhood optimisation heuristic: In the last step, the best solution found so far is improved by using a partial neighbourhood optimisation heuristic. In each step, a subregion, surrounding a selected median, is solved. If the solution found for this subregion improves the objective function value of the entire problem, then the partial solution updates the entire solution. This procedure is repeated until all medians are optimised or a maximum number of steps with no improvements is reached.

Step 1: SOM-GAP-based heuristic to solve a continuous, capacitated p-median problem

This section describes the generation of a solution for a continuous, capacitated p-median problem using a SOM and a GAP.

Self-organising maps

A SOM is a particular type of an artificial neural network and was introduced by Kohonen (1982). The main characteristic of a SOM is the capability 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 topographical mapping means that neighbouring areas of the input space have to activate neighbouring output units.

A SOM can be described as a feed-forward network consisting of two layers of units. The input units are completely connected with the output units in the second layer. Let a SOM consist of I input units and O output units. The weights on the edges between the input and outputs unit can be formulated as a Matrix $w = \{w_{mn} \in \mathbb{R} \mid m \in \{1,2,\dots,I\}, n \in \{1,2,\dots,O\}\}$. The output units are usually organised as a one- or two-dimensional array, where the units are laterally connected (Kohonen, 1982).

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 T times.

Given an (randomly selected) input vector $\boldsymbol{\theta}_l \in \mathbf{\Theta} = \{\boldsymbol{\theta}_m | m \in \{1,2,\dots,I\}\}; l \in \{1,2,\dots,L\}$, only one unit of the output layer can be activated according to the winner-takes-all output function (7). This is the output unit $\eta \in \{1,2,\dots,O\}$ with the minimum squared Euclidean distance between the input vector $\boldsymbol{\theta}_l$ and its weight vector w_η (Kohonen, 2001, p. 106).

$$\eta = \arg \min_{n \in \{1,2,\dots,O\}} \|\boldsymbol{\theta}_l - w_n\|^2 \tag{7}$$

After finding the winning output unit, the weights of this unit have to be updated in order to improve the matching. Since the projection between the input patterns and the activity of the output units have to be topographically correct, the weights of the neighbours of the winner unit also have to be updated. Kohonen proposed the Gaussian neighbourhood function as shown in expression (8) for a one-dimensional array of output units, where σ defines the width of the kernel (Kohonen, 2001, p. 111).

$$\delta(n,\eta) = \exp\left(-\frac{(n-\eta)^2}{2\sigma^2}\right); n \in \{1,2,\dots,O\}. \tag{8}$$

To update a weight w_{mn} between an input unit m and an output unit n, the difference $(\theta_m - w_{mn})$ is multiplied by the neighbourhood value $\delta(n,\eta)$ and a learning-rate factor α as shown in the learning rule (9) (Kohonen, 2001, p. 111).

$$\begin{aligned} &\Delta w_{mn} = \alpha \cdot \delta(n,\eta) \cdot (\theta_m - w_{mn}) \\ &; m \in \{1,2,\dots,I\}, n \in \{1,2,\dots,O\} \end{aligned} \tag{9}$$

Both parameters α and σ are usually decreased in each step of the learning phase to ensure faster changes of the weights in a broader neighbourhood at the beginning of the learning process and to refine it at the end of the process.

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

$$\begin{aligned} &\omega_l = \arg \min_{n \in \{1,2,\dots,O\}} \|\boldsymbol{\theta}_l - w_n\|^2 \\ &; l \in \{1,2,\dots,L\}. \end{aligned} \tag{10}$$

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

There are some similarities between a SOM and a continuous, uncapacitated p-median problem (Cooper, 1964). Therefore, Lozano et al. (1998), Hsieh and Tien (2004) and Aras et al. (2006) proposed several SOM-based approaches to solve such a location-allocation problem.

A continuous, uncapacitated p-median problem can be described as follows. Additionally to the definitions of the parameters and the variables of the CPMP, let $S = \{1,2,\dots,p\}$ be a set of sources, for which the coordinates $\boldsymbol{w}_k = \{w_{k1}, w_{k2}\}; k \in S$ have to be determined. The coordinates of the demand nodes $\boldsymbol{\theta}_j = \{\theta_1, \theta_2\}; j \in N$ are known. As shown in the model below, the intention is to find the locations of the sources over a two-dimensional continuous space and to allocate the destinations to the sources in order to minimise the total distances as per (11), whereby, due to the expressions (12), each destination has to be served by exactly one source (Cooper, 1967).

$$\sum_{k \in S} \sum_{j \in N} d(\boldsymbol{w}_k, \boldsymbol{\theta}_j) \cdot x_{kj} \rightarrow min \tag{11}$$

$s.t.$

$$\sum_{k \in S} x_{kj} = 1 \quad ; j \in N \tag{12}$$

$$x_{kj} \in \{0,1\} \quad ; k \in S, j \in N \tag{13}$$

$$w_{k1}, w_{k2} \in \mathbb{R} \quad ; k \in S \tag{14}$$

SomAla uses an adapted SOM to solve the continuous, uncapacitated p-median problem. Therefore, a network with $I = 2$ input units and $O = p$ output units organised in a one-dimensional array has to be created.

This SOM is intended to find a projection $f : \boldsymbol{\Theta} \rightarrow \boldsymbol{\Omega}$, where $\boldsymbol{\Theta}$ describes the coordinates $\boldsymbol{\theta}_l = \{\theta_1, \theta_2\}; l \in \{1, 2, \ldots, L\} = N$ of the demand nodes and $\boldsymbol{\Omega}$ the vector of the indices of the winner units per input pattern, which have to be obtained by the winner-takes-all function (7). This projection is equivalent to the allocation of the demand nodes to the sources.

In the initialisation step, the coordinates of randomly chosen demand nodes are used as initial values of the weights. The output nodes are forced to become specialised for particular areas of the input space and have to update their weights in order to minimise the distances $d(\boldsymbol{w}_n, \boldsymbol{\theta}_l); n \in S, l \in N$. It is possible to use alternative distance functions (squared Euclidean distance, Euclidean distance, Manhattan distance, great circle distance) to determine the winner output unit in SomAla.

After the learning process, the weights $w_{mn}; m \in \{1, 2\}, n \in \{1, 2, \ldots, p\}$ can be interpreted as the coordinates of the p sources. The allocation of the demand nodes to the sources and therefore the values of the assignment variables $x_{kj}; k \in S, j \in N$ can be obtained by applying the adapted winner-takes-all function:

$$x_{kj} = \begin{cases} 1, for \;\; k = \arg \min\limits_{n \in \{1,2,\ldots,p\}} d(\boldsymbol{w}_n, \boldsymbol{\theta}_l) \\ 0, otherwise \end{cases} \quad ; k \in S, j = l \in N \tag{15}$$

This solution can be interpreted as an initial solution for the rest of the SomAla algorithm which exploits the structure of the CPMP because the first clusters of demand nodes found by the SOM define a neighbourhood in which the optimal locations of the sources can be expected. These explorations improve the objective function value and partially the runtime of the entire algorithm in comparison to other relevant initial solution approaches

This positive effects can be shown with several tests in comparison to a simple randomised initial solution, a greedy initial heuristic and the approach reported by Mulvey and Beck (1984) which Stefanello et al. (2015) applied as an initial solution. Using the instances which are introduced in the section *COMPUTATIONAL RESULTS*, the SOM initial solution improves the average of the total distances after the first SomAla step by 8 %, 31 % and 5 % and overall by 2 %, 3 % and 2 % in comparison to the greedy initial solution, the randomised initial solution and the approach by Mulvey and Beck (1984). There is a deterioration of the runtime of the entire algorithm of 7 % and 12 % in comparison to the greedy initial solution and the approach by Mulvey and Beck (1984) which is unproblematic because the tests in the section *Runtime comparison for SomAla and IRMA* show that SomAla's runtimes seem faster than the computational times of the most competitive approach. If a randomised initial solution is used instead of the SOM-based initial solution then there is an increase of the runtime of 26 %. This effect is mainly based on the exploration of the structure of the CPMP using the SOM which helps to decrease the number of the time-consuming iterations in the following alternating location-allocation heuristic.

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. It can be described by using a directed network with a set of source nodes S, a set of demand nodes N and a set of directed arcs $\tilde{A}$ joining the sources and the demand nodes. The weights of the arcs represent the distances $d_{kj}; (k, j) \in \tilde{A}$. If a demand node $j \in N$ is assigned to a source $k \in S$, then the corresponding binary allocation variable x_{kj} equals one, or zero otherwise. All other parameters are defined as before. The mathematical model can be formulated as follows (Lorena and Senne, 2003):

$$\sum_{(k,j) \in \tilde{A}} d_{kj} \cdot x_{kj} \rightarrow min \tag{16}$$

$$s.t.$$

$$\sum_{\{k|(k,j) \in \tilde{A}\}} x_{kj} = 1 \quad ; j \in N \tag{17}$$

$$\sum_{\{j|(k,j) \in \tilde{A}\}} q_j \cdot x_{kj} \leq Q \quad ; k \in S \tag{18}$$

$$x_{kj} \in \{0, 1\} \quad ; (k, j) \in \tilde{A} \tag{19}$$

The intention is to allocate the demand nodes to the sources in order to minimise the total distances between the sources and the destinations as per (16). Since it is a continuous model, the distances have to be determined on the basis of the coordinates of the sources found by the SOM and the coordinates of the demand nodes. Each demand node has to be served by exactly one source due to the constraints (17). The supply constraints (18) ensure that the deliveries of each source cannot exceed the supply.

To decrease the effort solving this binary, linear programme, several size reducing, variable relaxing and fixing techniques are used.

Assuming that a source usually serves only demand nodes in its neighbourhood (Stefanello et al., 2015), the size of the model can be reduced by ignoring all arcs between the sources and the destinations with distances which are greater than a defined radius $\bar{d}$ surrounding the sources:

$$\tilde{A} = \{(k, j) \mid k \in S, j \in N, d_{kj} \leq \bar{d}\} \tag{20}$$

This size-reduced model is solved in two steps using a simple but effective variable relaxing-fixing heuristic. In the first step the problem is solved as a continuous, linear programme by relaxing the integer constraints (19):

$$0 \leq x_{kj} \leq 1 \quad ; (k, j) \in \tilde{A} \tag{21}$$

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 and therefore deleted

in the binary problem. For underutilised sources, an unused relation is uniformly fixed at random with a particular probability.

The set $\tilde{A}$ of the directed arcs joining the sources and the destinations can be now determined as follows:

$$\tilde{A} = \{(k,j) \mid k \in S, j \in N, \\ d(k,j) \leq \bar{d}, x_{kj} \; not \; fixed\} \tag{22}$$

In addition to the size reducing, relaxing-fixing techniques described above, two solver specific options are used to decrease the solving time of the binary GAP. The solving process is stopped if a specific relative mip gap or a maximum solving time is reached.

The variable-fixing heuristic is also used for the following ALA heuristic and has a huge positive impact on the runtime of the entire SomAla algorithm. This effect was tested compared to the exactly solved GAP as part of the SomAla algorithm by solving the problem instances used to compare SomAla's and IRMA's results which will be introduced in the section *COMPUTATIONAL RESULTS*. If the GAP as part of the SomAla algorithm is solved exactly as a binary linear programme then there is an increase of the average of the runtimes of 28.0 % with a very small improvement of the average of the objective function values of 0.3 % compared to the variable-fixing heuristic used for the GAP.

Step 2: Capacitated alternating location-allocation algorithm

An alternating location-allocation algorithm is often used as a heuristic to solve location problems, for that, the location decision and the allocation decision have to be made simultaneously.

Cooper (1964, 1972) was the first to propose this approach to solve continuous p-median problems. An example for the CPMP is the approach reported by Mulvey and Beck (1984), which is utilised by Stefanello et al. (2015) as an initial solution. Lorena and Senne (2003) use an ALA as part of a local search heuristic for the CPMP, where they re-allocate the demand nodes by solving a GAP as binary linear programme.

The capacitated ALA heuristic proposed in this section is also based on a GAP and consists of the following working steps:

1. Find new medians minimising the total distances for each of the p clusters.
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.

In step 1, the medians for the p clusters have to be calculated, where each cluster $C_k; k \in S$ can be found by analysing the activities of the binary allocation variables $x^*_{kj}; k \in S, j \in D$:

$$C_k = \{j \mid j \in N, x^*_{kj} = 1, (k,j) \in \tilde{A}\}; k \in S \tag{23}$$

The median M_k is the node $s \in C_k$, that minimises the distances to all other nodes in this cluster:

$$M_k = \arg\min_{s \in C_k} \sum_{t \in C_k} d_{st} \quad ; k \in S \tag{24}$$

These medians define the new locations of the sources. The basis of these calculations is a distance matrix $D = \{d_{ij} \mid i \in N, j \in N\}$, with which the distance between a source $k \in S$ and a demand node $j \in N$ can be found via M_k, which is the index of the node $i \in N$ hosting the source:

$$d_{kj} = d_{M_k j} \quad ; k \in S, j \in N \tag{25}$$

This distance matrix can be based on road network distances or on the same distances measures used for the first SomAla step to obtain a solution for a continuous, capacitated p-median problem (squared Euclidean distance, Euclidean distance, Manhattan distance, great circle distance).

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.

The results of the ALA steps are the positions of the sources and the allocations of the destinations to the sources. The locations can be determined by the last medians $M_k; k \in S$ found in step 1. The allocations can be obtained by the activities of the allocation variables $x^*_{kj}; (k,j) \in \tilde{A}$ in the solution of the last GAP solved in step 2.

Step 3: Partial neighbourhood optimisation heuristic

In the last step of the SomAla algorithm 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).

The idea is to solve in each step of this heuristic a CPMP for a subregion surrounding a selected median. If the solution found for this subregion improves the objective function value of the entire problem, then the partial solution is used to update the entire solution. This procedure is repeated until all medians are optimised or a maximum number of steps without an improvement is reached.

The working steps can be described as follows:

0. The initial solution consists of the medians $M_k; k \in S$ and the activities of the allocation variables $x_{kj}; (k,j) \in \tilde{A}$ found with the ALA steps.
1. Mark all sources $k \in S$ as non-optimised.
2. Select randomly a non-optimised source $\kappa \in S$.
3. Create a set of sources $\tilde{S} \subset S$, which consists of κ and the z closest sources surrounding κ.
4. Create a set of demand nodes $\tilde{N} = \{j \mid j \in C_k, k \in \tilde{S}\}$, which belong to the sources $\tilde{S}$.
5. Calculate the total distances C for the sources $\tilde{S}$ and the corresponding demand nodes $\tilde{N}$.
6. Solve a CPMP for the demand nodes $\tilde{N}$ and $\tilde{p} = |\tilde{S}|$ sources and obtain the objective function value $\tilde{C}$ for this sub-problem.
7. If there is an improvement $\tilde{C} < C$, then update the solution of the entire problem by using the new locations of the sources and the allocations of the demand nodes to the sources found in step 6.
   If there is no improvement $\tilde{C} \geq C$, then mark the source κ as optimised.
8. If there exists at least one non-optimised source and a maximum number of no-improvements is not reached, then continue with step 2.

This heuristic follows the idea proposed by Stefanello et al. (2015), but there are some differences. The first difference is that Stefanello et al. (2015) mark all sources $\tilde{S}$ as optimised if no improvement occurs after solving the subproblem. However, in SomAla only the selected source κ is marked as optimised in this case. That means the neighbourhood of each source has to be examined. But this procedure is interrupted if a maximum number of succeeding no-improvements is reached.

An additional difference is that in SomAla there are two alternatives to solving the CPMP in step 6. The first alternative is to use the first two SomAla working steps. The second alternative is an size-reduced CPMP. As in the original model, each of the demand nodes in $\tilde{N}$ is a potential location of the $\tilde{p} = |\tilde{S}|$ sources. But it is assumed that sources only serve destinations in their neighbourhood, so that the size of the model can be reduced by using a maximum distance $\bar{d}$ for the arcs $\tilde{A}$.

After completing all working steps, a best solution is found with locations of the p sources at the demand nodes and the allocations of the demand nodes to these sources.

COMPUTATIONAL RESULTS

Test environment and instances

This last section is intended to prove the capability of the proposed SomAla algorithm to find feasible solutions with good objective function values in reasonable runtimes for large problem instances.

Since it could be shown in section *LITERATURE REVIEW* that IRMA by Stefanello et al. (2015) is the most competitive approach to solving CPMPs, this algorithm is the benchmark for SomAla. Therefore, the objective function values and the runtimes of both approaches are compared in this section by computing six of the largest instances which Stefanello et al. (2015) used to test the results of their algorithm. These instances are *u*-724, *spain*, *rl*-1304, *pr*-2392, *p*-3038 and *fnl*-4461.

The *spain* instances were introduced by Diaz and Fernandez (2006) and contain 737 Spanish cities as demand nodes served by 74 or 148 sources. Both combinations are additionally separated in two cases with different demands and supplies. Lorena et al. (2003) proposed the *p*-3038 instances, which are adapted from the TSPLIB instance *pcb*3038 (Reinelt, 1994) by introducing five cases with 600, 700, 800, 900 and 1000 sources. All other instances *u*-724, *rl*-1304, *pr*-2392 and *fnl*-4461 are introduced by Stefanello et al. (2015) and adapted from the TSPLIB.

All distances are Euclidean distances, except the distances for the *spain* instances, which are calculated as great-circle distances.

These instances provide a wide range of problem sizes and different ratios between the demand nodes and the sources. All instances used for the benchmark calculations have a supply-demand ratio of $\approx 125\,\%$.

The last instances *g*-11056 are not used for the comparison. These instances are only intended to show SomAla's capability to find feasible solutions with good objective function values in reasonable times for very large instances and also to provide instances for future comparisons. The demand nodes are the 11,056 German cities and communities published by the Statistisches Bundesamt (2017) with their geographical coordinates. Four instances are created with different ratios between the number of the sources and the number of the the destinations.

Hereinafter, the name of a problem contains the number of the source and of the destinations (name-nrOfSources-nrOfDestinations). All examples and the corresponding solution files are available at the SomAla project website (http://stegger.net/ SomAla).

Objective function value comparison for SomAla and IRMA

This section reports the results of the objective function value comparison between IRMA and SomAla. The results for SomAla have to be distinguished due to the alternative approaches for the partial neighbourhood optimisation heuristic as described before. Either the first two SomAla working steps or a size-reduced CPMP can be used to solve the CPMP. The abbreviation for the first approach is SomAla-MS and SomAla-MM for the a size-reduced CPMP. Table 4 shows the results for IRMA, SomAla-MM and SomAla-MS.

The best known solutions per instance are given in the column BKS. The best known solutions for the *p*-3038 instances were reported by Diaz and Fernandez (2006). Most of the other best solutions were found with IRMA by Stefanello et al. (2015), excluding the best known solutions for *spain*734-74-1, *spain*734-148-1, *rl*-1304-010, *pr*-2392-020, *pr*-2392-075 and *fnl*-4461-100, which were obtained with SomAla. The average objective function values for ten runs per instance for IRMA, SomAla-MM and SomAla-MS are shown in the next three columns. Each instance has been run with particular parameters, which are published for IRMA in Stefanello et al. (2015). For SomAla, all parameters can be obtained in the example files for all instances at the SomAla project website (http://stegger.net/SomAla).

For a better comparison, the relative gap between the objective function values Z and the best known solutions BKS ($ObjGap = (Z/BKS - 1) \cdot 100$) are shown in the last three columns for the three approaches.

The results for IRMA are slightly better, but the difference of the average $ObjGap$ for all instances between IRMA and SomAla-MM is only $0.21\,\%$. For most of the instances the objective function values of SomAla-MS are greater than the results of the other two algorithms, but with a small difference of the average $ObjGap$ for all instances of $0.93\,\%$ compared to SomAla-MM and $1.14\,\%$ to IRMA.

It seems that IRMA and SomAla-MM are similar in their objective function values for all instances and also on average over all instances. This can be seen in table 5. The first column shows four classes for $ObjGap$. The relative number of instances belonging to the particular GAP class for the three approaches are given in the next three columns. The results for IRMA and SomAla-MM are identical. With both algorithms it is possible to find a solution with an $ObjGap \leq 1\,\%$ to the best known solution for $86.2\,\%$ of the instances. For $93.1\,\%$ of the instances, a solution with an $ObjGap \leq 2\,\%$ and for $96.6\,\%$ with an $ObjGap \leq 3\,\%$ are

Table 4: Objective function values for IRMA and SomAla

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

Table 5: ObjGap distributions for IRMA and SomAla

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

found. Due to the maximum $ObjGap$ over all instances, both algorithms found solutions with an $ObjGap \leq 4\,\%$ for all instances. Both approaches are better than SomAla-MS. Nevertheless, it is possible to find solutions with an $ObjGap \leq 3\,\%$ for 89.7 % and with an $ObjGap \leq 4\,\%$ for 93.1 % of all instances with SomAla-MS.

Runtime comparison for SomAla and IRMA

In this section, a comparison of the runtimes of IRMA and SomAla will be attempted. In general it is difficult to compare the computational times, as the tests for IRMA and SomAla were computed on different computer systems. Stefanello et al. (2015) used a computer with an Intel i5-2300 2.80GHz CPU with 4 GB RAM running on Ubuntu 10.10, whereby the tests for SomAla were computed on a MacBook Pro with a Core i7-6820HQ CPU and 16 GB RAM on macOS 12.13.4. There is an advantage for the SomAla tests due to the newer and more performant hardware. Both algorithms use CPLEX to solve linear programmes as part of the algorithm, whereby CPLEX 12.8 has been used for SomAla and CPLEX 12.3 for IRMA (Stefanello et al., 2015). There could be an additional advantage for the SomAla tests due to the release-wise performance improvements of CPLEX.

Therefore, it does not make sense to compare the original computational times reported by Stefanello et al. (2015) with the SomAla runtimes. There is, however, an opportunity to estimate computational times for IRMA, assuming that the same hardware and the same CPLEX version would have been used for the tests.

To compute the time savings due to different hardware, three selected CPU benchmarks (PassMark, UserBenchmark and GeekBench) are considered. Table 6 shows the results of these benchmarks, distinguished in single-thread and multi-thread tasks for the CPUs used for IRMA (i5-2300) and for SomAla (i7-6820HQ). The results for the i5-2300 CPU ($si5$) and for the i7-6820HQ CPU ($si7$) in the second and the third columns are scores, computed in a particular time. Therefore, the rounded ratios i7/i5 ($si7/si5 \cdot 100$) in the last column are also interpretable as the relative computational time of the i7-6820HQ system compared to the i5-2300 system. The computational time of an i7-6820HQ system is on average for single-thread tasks 80 % and for multi-thread tasks 61 % of the computational time on an i5-2300 system.

Without any other information, it is furthermore assumed that 10 % of IRMA's algorithm runs in a single-thread mode and 90 % is multi-threaded. Depending on this assumption and the benchmarks, a weighted average of $62\,\% = \lfloor 0.1 \cdot 0.80 + 0.9 \cdot 0.61 \rfloor \cdot 100$ can be calculated as a first estimation of IRMA's runtime. That means that IRMA would only need 62 % of its original computational time, if Som-

Table 6: Selected CPU benchmarks (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

Ala's environment was used and all other impacts on the runtime are ignored.

To estimate the potential time savings depending on different CPLEX versions, it is assumed (in absence of any other information) that each CPLEX major release provides a 15 % time saving for size-reduced CPMP instances. (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 %.) This assumption of an release-wise 15 % time saving can be combined with the benchmark based runtime estimation for IRMA, whereby it is assumed that CPLEX causes 90 % of IRMA's entire computational runtime. Therefore, IRMA would only need $30\,\% = \lfloor 0.1 \cdot 0.62 + 0.9 \cdot 0.62 \cdot (1-0.15)^5 \rfloor \cdot 100$ of its original computational time, if SomAla's test environment and CPLEX 12.8 was used.

The runtime comparisons are reported in table 7. The original averages of the runtimes of ten runs per instance are shown in the second, third and fourth columns. Based on the runtimes for IRMA t_{IRMA} and SomAla t_{SomAla} the runtime ratios between SomAla and IRMA $t_{SomAla}/t_{IRMA} \cdot 100$ are shown in the last two columns for both SomAla variants. These ratios can be used to compare SomAla's runtimes with the runtime estimation of 30 % calculated for IRMA.

As shown in the last row, SomAla-MM only needs 15.09 % and SomAla-MS only 7.63 % on average over all instances of the computational time needed for IRMA. Both averages are less than the boundary of 30 % of the estimated relative computational time assuming that IRMA is running on SomAla's test environment. Additionally, the maximum ratios of 27.93 % for SomAla-MM and 24.11 % for SomAla-MS are also less than the boundary of 30 % of IRMA's runtime estimation, which means that IRMA's runtime estimations could be undercut by 100 % of the instances solved with SomAla.

The results of a second test corresponds with these runtime comparisons. SomAla was additionally tested on an older Mac mini with an i7-4578U CPU and 8 GB Ram running on macOS 12.13.4 using SCIP 6.0 instead of CPLEX. The same benchmarks and CPU improvement assumptions as before have been used to determine a new IRMA runtime estimation with the result, that IRMA would need 112 % of its original computational time, if this Mac mini was used. Not included in this estimation is the fact that SCIP is usually more slowly compared to CPLEX which is used for IRMA. These tests show that SomAla-MM only needs 89,12 % and SomAla-MS only 19,01 % of the computational time needed for IRMA on average over all instances which is less than IRMA's runtime estimate of 112 % for this environment.

The results of the comparisons in this section cannot be evidence that SomAla solves CPMPs faster then IRMA. But, if the assumptions for the estimations of IRMA's relative computational times on SomAla's test environment are correct, then the results are strong indicators that there are runtime improvements for all of the tested instances solved with SomAla.

Results for the g-11056 instances

These instances are only introduced to show SomAla's capability to find feasible solutions with good objective function values in reasonable times for very large CPMP instances and also to provide instances for future comparisons. These instances are the largest instances which have ever been published for the CPMP.

As described above, the demand nodes are all 11,056 German cities and communities published by the Statistisches Bundesamt (2017) with their geographical coordinates. Four instances are created with different ratios between the number of the sources and the number of the destinations, whereby for the ratio the values 1 %, 5 %, 10 %, 20 % and 30 % have been chosen. The integer demands in an interval $[1;100]$ are computed randomly and the supplies are obtained with a supply/demand ratio of $\approx 143\,\%$.

The results were obtained on a computer with an AMD Ryzen 7 1800X CPU with 64GB RAM running on Kubuntu 17.10.1 using CPLEX 12.8 for solving linear programmes.

Table 8 shows the objective function values and the computational times for these five instances. The columns two, three and four contain the best known solutions (BKS) and the averages of the objective function values of the ten runs per instance for SomAla-MM and SomAla-MS. The best known solutions are best solutions found during the test of the instances. In the next columns the relative $ObjGap$ are shown. The runtimes for SomAla-MM and SomAla-MS are given in the last two columns. The averages of the relative gaps with 0.42 % for SomAla-MM and 1.16 % for SomAla-MS are quite small and were found with an average computational time of 2,059 seconds with SomAla-MM and 1,423 seconds with SomAla-MS which is more than satisfactory for such large instances.

The relative gaps and the computational times show that SomAla is a heuristic which enables decision makers to find feasible solutions for very large instances with good total distances in a reasonable time.

CONCLUSIONS

This paper proposes a new hybrid heuristic for the CPMP which combines a self-organising map (SOM), integer linear programming, an alternating location-allocation algorithm (ALA) and a partial neighbourhood heuristic in three working steps.

In the first step, a solution for a continuous, capacitated p-median problem is found using a SOM and a generalised

Table 7: Computational times for IRMA and SomAla

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

Table 8: Results for the $g11056$ instance

Instance	Obj. value			ObjGap [%]		Computational time [sec]	
	BKS	SomAla-MM	SomAla-MS	SomAla-MM	SomAla-MS	SomAla-MM	SomAla-MS
g-11056-111	208,570	209,025	208,970	0.22	0.19	523	491
g-11056-553	88,708	89,472	90,318	0.86	1.81	1,800	559
g-11056-1106	58,753	58,847	58,832	0.16	0.13	2,649	1,014
g-11056-2212	38,400	38,554	38,686	0.40	0.75	1,743	1,605
g-11056-3317	30,611	30,755	31,502	0.47	2.91	3,580	3,447
Minimum				0.16	0.13		
Average	85,009	85,331	85,661	0.42	1.16	2,059	1,423
Maximum				0.86	2.91		

assignment problem (GAP). This solution is used in the second step in an ALA based on GAPs to find and improve a solution for the CPMP. In the last step, the best solution found so far is improved by using a partial neighbourhood optimisation heuristic.

The combination of a SOM with a GAP in the first step of the algorithm exploits the structure of the CPMP and improves the objective function value of the entire algorithm in comparison to alternative initial solutions. The next working steps combine several known techniques (integer linear programming, ALA, partial neighbourhood optimisation) in a new manner. All of these approaches are extended by several size-reduction methods and a variable relaxing-fixing heuristic to improve the SomAla's capability to find good solutions for large problem instances as fast as possible.

SomAla was tested on several benchmark instances in comparison to IRMA by Stefanello et al. (2015), which is, according to the literature review, the most competitive approach to solving CPMPs. It could be shown that IRMA and SomAla-MM are more or less similar in their objective function values. Both approaches provide better results than SomAla-MS. But this approach is also able to find solutions for 89.7 % of all instances with a relative gap of ≤ 3 % to the best known solutions. Since the tests for IRMA and SomAla were run on different test systems, the original computational times cannot be compared. Therefore, estimations of the computational times for IRMA, assuming this algorithm would run on SomAla's test system, 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. The comparison of SomAla's runtimes and the runtime estimations for IRMA indicates runtime improvements for all of the tested instances solved with SomAla.

These results and the results of the g-11056 instances, which are the largest instances ever published for the CPMP, show that SomAla is a heuristic which enables decision makers to find good feasible solutions for large instances with reasonable computational times.

REFERENCES

Aras, N., Özkisacik, K.C., Altinel, I.K., 2006. Solving the 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.

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

Herda, M., 2015. Combined genetic algorithm for capacitated 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 rectilinear distances. *Computers & Operations Research* 31, 10171031.

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

Janosikova, L., Herda, M., Haviar, M., 2017. Hybrid Genetic Algorithms with Selective Crossover for the Capacitated 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.

Lorena, L.A.N., Senne, E.L.F., 2003. Local Search Heuristics 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.

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.

AUTHOR BIOGRAPHY

Mike Steglich is a Professor of Business Administration, Quantitative Methods and Management Accounting in the Faculty of Business, Computing and Law of the Technical University of Applied Sciences Wildau in Germany. Additionally, he is intensively involved in the development of software for mathematical modeling and optimisation in several open-source projects.