Sink Location to Find Optimal Shelters in Evacuation Planning ☆

P. Heßler[a], H.W. Hamacher[a,*]

[a] *Department of Mathematics, University of Kaiserslautern*

Abstract

The sink location problem is a combination of network flow and location problems: From a given set of nodes in a flow network a minimum cost subset W has to be selected such that given supplies can be transported to the nodes in W. In contrast to its counterpart, the source location problem which has already been studied in the literature, sinks have, in general, a limited capacity. Sink location has a decisive application in evacuation planning, where the supplies correspond to the number of evacuees and the sinks to emergency shelters.

We classify sink location problems according to capacities on shelter nodes, simultaneous or non-simultaneous flows, and single or multiple assignments of evacuee groups to shelters. Resulting combinations are interpreted in the evacuation context and analyzed with respect to their worst-case complexity status.

There are several approaches to tackle these problems: Generic solution methods for uncapacitated problems are based on source location and modifications of the network. In the capacitated case, for which source location cannot be applied, we suggest alternative approaches which work in the original network. It turns out that latter class algorithms are superior to the former ones. This is established in numerical tests including random data as well as real world data from the city of Kaiserslautern, Germany.

Keywords: network location, sink location, network flows

1. Introduction

In recent years there has been an increasing number of natural as well as man made disasters. A crucial aspect of the preparedness to such events is the choice of shelter locations such that everyone who has to be evacuated can be accommodated. However, one must also take into consideration the traffic that

☆Partially supported by the Federal Ministry of Education and Research Germany, grant DSS_Evac_Logistic, FKZ 13N12229.

*Corresponding author

Email addresses: hessler@mathematik.uni-kl.de (P. Heßler), hamacher@mathematik.uni-kl.de (H.W. Hamacher)

will result from evacuees going from the affected area to their designated shelter. The amount of traffic has a big influence on the duration of the evacuation process. Due to the importance of the best possible choice for shelter locations, this topic has been the subject of some researcher papers during the last years.

Clearly, shelter location is not the only decision which influences the outcome of an evacuation plan. The collection points for people relying on public transportation have to be chosen, the public and private traffic from the endangered zone has to be directed to the chosen shelters, the evacuees have to be supplied with medicine, food, water, a place to rest, and more once they reach their shelter, emergency forces must be able to enter the endangered zone in order to eliminate the reason for the evacuation, etc. These aspects will be part of a decision support system we are developping in the DSS Evac Logistic project[1]. The ultimate goal of our research is an integrated model, which solves all subproblems simultaneously. To reach this goal we follow in this paper an approach in which some problems are treated in a integrative fashion while others are dealt with sequentially to make sure that our model can be solved in reasonable computing times.

An overview on the modeling of evacuations from given evacuation gathering points (sources) to given evacuation shelters (sinks) can be found in the paper by Hamacher and Tjandra[6] which focuses on the evacuation process itself. A paper which puts emphasis on the choice of shelter locations instead of the evacuation process is the one by Sherali et al.[15]. They build a complex model which incorporates capacities, staff ressource and other opening restriction on shelters. The evacuation process itself is modeled as a network flow with congestion-related travel times and the objective is to minimize the total travel time. This is one of the first quite realistic models. However, it is also too complex to solve so that one has to rely on heuristics. Even more realistic models have been given in the last years, for example by Kongsomsaksakul et al.[10] and Ng et al.[2] who both introduced a bilevel model in which in a first step a central authority chooses a number of shelters in order to minimize the total travel time, and in a second step the evacuees decide to which shelter and which way they travel. This model increases the realism but also the computational complexity. In order to overcome these computational problems we suggest a simpler model that can be solved fast and has sufficient modeling capabilities for real-world evacuation problems.

We model the evacuation problem as a sink location problem in a given network with edge capacities. These capacities model the maximal amount of traffic that can pass a road. Each of the nodes has a given number of evacuees that will individually find their way to shelter locations. The shelter locations have to be chosen among a set of admissible sinks. After these sinks have been chosen the stream of evacuees can be modeled as a network flow starting at the given nodes going to the chosen sinks. This flow must not only respect the edge capacities but also capacities given by the number of evacuees each shelter can

[1] see http://projets.li.univ-tours.fr/dssvalog/

hold. The objective is to minimize the opening costs for the shelters/sinks.

While the sink location problem is interesting in its own right, its usefulness for any decision support system in evacuation planning is obvious. To the best of our knowledge the diploma thesis of Kissel[9] was the first work to analyze this problem. There are, however, two papers which deal with similar problems. The first one is by Mamada et al.[11]. Their problem is to find a single sink in a tree such that the supply of all nodes can be discharged to the sink as fast as possible. For the discharging process they consider dynamic network flows. A second paper that is not restricted to the placement of a single sink is by Fujishige et al.[4]. They wish to pick a number of sinks, or in their context roots, such that each vertex set is covered by a single sink or each vertex is covered by a vertex dependent number of sinks. However, they do neither consider capacities on the edges nor on the chosen sinks.

Our model is closely related to another network location problem, the source location problem. In the source location problem we are given a network with edge capacities and wish to place sources on the nodes such that the demand of every node can be satisfied. Here the demand is satisfied if there exists a flow from one or all of the chosen sources to each node of the given demand which respects the edge capacities. The amount of flow that a source can emit is not bounded. If we drop the capacity constraint on the sinks in our model, then the sink location model is equivalent to a source location problem since we can simply reverse the flow direction and possibly the orientation of the edges if some or all edges of the graph are directed. We will give original references to the different types of source location in the corresponding subsection of Section 3. For a detailed overview on source location we refer to the thesis of Poetranto Groß[12] and an unpublished paper by Turner et al.[21] which also deals with dynamic source location.

The idea of source location problems has also been applied to disaster preparedness. For example, Rawls and Turnquist[13] propose a model for a priori positioning of goods to supply evacuees in the case of disasters with medicine, food, etc. Their model is quite detailed but the time to compute a solution is too high even for small instance sizes.

Our paper is structured as follows: In Section 2 different types of sink location problems are formally introduced and their interrelations are discussed. We highlight their applicability to evacuation planning. The following Section 3 contains solution methods based on existing algorithms for source location whereas Section 4 contains new solution methods and complexity results. Section 5 describes the computational setup of our experiments in which we compare our two solution approaches from Section 3 as well as the new approaches from Section 4. These experiments will then be analyzed in Section 6 showing that our algorithms perform significantly better than those that already exist for source location and far better than general integer programming solvers. The paper is concluded in Section 7 by summarizing the results of this paper and indicating directions for future research.

2. Sink Location Models

In sink location problems we are interested in finding a set of sinks such that a given supply inside a network can be discharged while capacities of the edges as well as the sinks are respected. Such a set is called a *(sink) cover*. Let $G = (V, E)$ be an undirected graph. Further, let $u(e)$ denote the capacity of edge $e \in E$, $a(v)$ the supply of node $v \in V$, $k(v)$ the capacity of $v \in V$ if it is chosen as a sink, and $c(v)$ the cost incurred for chosing $v \in V$ as a sink. There are different types of sink location problems depending on the type of cover: Single or plural, non-simultaneous or simultaneous, and independent or additive sink cover. These types will be defined in the following subsections.

In general, a solution to a sink location problem consists of a set $W \subseteq V_p$, where V_p is a given set of candidate nodes for the location of sinks, in short, perrmitted sinks, along with a set of flows $f_v : E \to \mathbb{R}_{\geq}$ from v to W for all $v \in V$. Each of these flows is characterized by

$$\text{excess}_v(w) = \sum_{u \in V} f_v(u, w) - \sum_{u \in V} f_v(w, u) \begin{cases} = -a(v), & \text{if } w = v \\ = 0, & \text{if } w \notin W \cup \{v\} \\ \geq 0, & \text{if } w \in W \end{cases} \tag{1}$$

for $v \notin W$,

$$\text{excess}_v(w) = \sum_{u \in V} f_v(u, w) - \sum_{u \in V} f_v(w, u) \begin{cases} \geq -a(v), & \text{if } w = v \\ = 0, & \text{if } w \notin W \\ \geq 0, & \text{if } w \in W \setminus \{v\} \end{cases} \tag{2}$$

for $v \in W$, and

$$\sum_{w \in W} \text{excess}_v(w) = 0 \tag{3}$$

2.1. Cover Types

In the single sink location problem the supply of each node must be discharged to a single sink whereas in the plural sink location problem the supply may be discharged to all sinks.

Definition 1. *Every solution $(W, (f_v)_{v \in V})$ of (1, 2, 3) is called a plural sink cover. If for each $v \in V$ there exists a sink $w_v \in W$ such that $\text{excess}_v(w_v) = a(v)$ then the solution is also called a single sink cover.*

Depending on whether capacities on the arcs are considered separately for each of the flows f_v, $v \in V$, in Definition 1 or together with all the other flows f'_v, $v' \in V$, we distinguish between non-simultaneous and simultaneous covers:

Definition 2. *A solution $(W, (f_v)_{v \in V})$ is called a non-simultaneous sink cover if $f_v(e) \leq u(e)$ for all edges $e \in E$. It is called a simultaneous sink cover if in addition to that it holds that $\sum_{v \in V} f_v(e) \leq u(e)$.*

Similarly the sink capacities $k(w)$, $w \in W$ can either be considered for each of the flows f_v, $v \in V$ independent of the others or in an additive fashion.

Definition 3. *A solution $(W, (f_v)_{v \in V})$ is called an independent sink cover if for all $w \in W$ and all $v \in V$ it holds that $\text{excess}_v(w) \leq k(w)$. If additionally $\sum_{v \in V} \text{excess}_v(w) \leq k(w)$ holds for all $w \in W$ then the solution is also called additive sink cover. If $k(w) = \infty$ for all $w \in V_p$, W is an uncapacitated solution.*

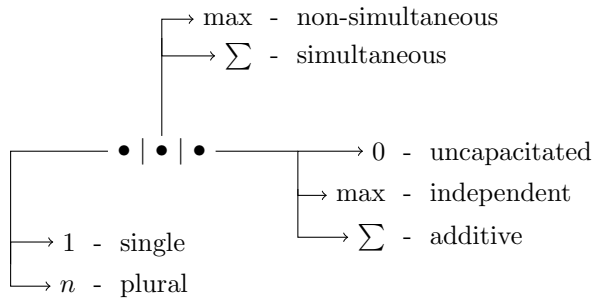


Figure 1: Classification Scheme

Figure 1 shows the 3-position scheme •|•|• which we use in this paper to classify the different types of sink location problems obtained by all combinations of Definitions 1, 2, and 3. In this paper we will focus on such combinations which can be interpreted as versions of shelter location problems (see Table 1).

Obviously, the combinations not presented in this paper are nevertheless interesting in their own right: When the flows are considered simultaneously and the sink capacities separately, then the flow f_v of some node $v \in V$ has to be distinguished from the flow $f_{v'}$ of any other node $v' \in V, v' \neq v$. Similarly, if the supply cannot be split among several sinks and the flow is considered simultaneously, then these flows have to be distinguished as well. Thus, these combinations require the use of multi-commodity network flows. In this paper we will only analyze problems which can be modeled using single-commodity network flows as only these are applicable to evacuation planning. However, we

Sink Assignment	Edge Capacity	Sink Capacity	Classification Scheme
single	non-simultaneous	independent	1|max|max
single	non-simultaneous	additive	1|max|$\sum$
plural	non-simultaneous	independent	n|max|max
plural	non-simultaneous	additive	n|max|$\sum$
plural	simultaneous	additive	n|$\sum$|$\sum$

Table 1: Types of Sink Location Problems

also have first results for the other problem types. These results will be part of the forthcoming PhD thesis [7].

2.2. Application to Evacuation Planning

The different cover types introduced in the previous subsection can be applied to various evacuation scenarios.

Example 4. *A no-notice incident in the city requires the immediate evacuation of a city district. Several safe spots have to be chosen where the people from the affected district can go to. Depending on the type of incident they will either stay at these safe spots until they can return to their homes or at a later point in time they are taken to other, more comfortable shelters where they can stay for a longer time. The choice of the safe spots can be modelled as a plural, simultaneous, additive sink location problem:*

- *people from one block can go to any safe spot → plural cover*
- *all people will leave the district immediately → simultaneous cover*
- *people accumulate at the safe spots and do not disappear → additive cover*

In a less chaotic situation with upfront notice (e.g., defusing of a bomb) the authorities dealing with the evacuation might be able to evacuate the city district block by block. In this case the network flows need not be considered simultaneously.

Furthermore, the authorities might decide to evacuate every individual block to a single shelter which would require the use of single cover models. This simplifies the evacuation process, however, it might require the use of more shelters compared to plural cover models.

3. Adaptations from Source Location Algorithms

In this section we focus on classes of capacitated sink location problems which can be formulated as source location problems, thus being solvable by algorithms from the literature for the latter class of problems. Obviously, this holds for uncapacitated sink location problems, since they are equivalent to source location problems (which are by definition also uncapacitated) and switching the roles of sources and sinks will work. This is even true for directed networks, if we reverse the direction of all arcs in the network. In the first part we will extend this observation to classes of capacitated sink location problems by a straighforward, but very useful modification of the network. This part also serves as a brief review of the source location literature. In the second part we show how algorithms for source location problems can be adapted to solve some of the sink location problems in the original network.

3.1. Known Algorithms Applied to Modified Network

The idea of node doubling is to eliminate capacities on the nodes by creating additional edges with capacities. Let $G = (V, E)$ be the given graph with edge capacities u, node capacities k, node costs c and permitted nodes $V_p \subset V$. We create a new network consisting of a graph $G' = (V', E')$ with edge capacities u' and permitted nodes V_p' as follows: Let the nodes in V be numbered such that $V_p = \{v_1, \ldots, v_k\}$ and let v_i' denote a copy of node v_i. We set $V' = \{v_1, \ldots, v_n, v_1', \ldots, v_k'\}$ and $E' = E \cup \{(v_1, v_1'), \ldots, (v_k, v_k')\}$. The capacities of the edges are defined as

$$u(e) = \begin{cases} u(e), & \text{if } e \in E \\ k(i), & \text{if } e = (v_i, v_i') \in E' \setminus E \end{cases}$$

and the set of permitted nodes is $V_p' = \{v_1', \ldots, v_k'\}$. The node costs are defined as $c(v_i') = c(v_i)$ for $v_i' \in V_p'$ and $c(v_i') = 0$ for $v_i \in V' \setminus V_p'$.

Clearly, the $1|\max|0$ *sink* location problem is equivalent to the single *source* location problem, the $n|\max|0$ *sink* location problem is equivalent to the plural *source* location problem, and the $n|\sum|0$ *sink* location problem is equivalent to the simultaneous *source* location problem. Thus, after doubling the nodes we can apply the algorithms that were originally developed for source location problems. Furthermore, this shows that the computational complexities of these problems are the same as the complexities of the corresponding source location problems, since the number of nodes and edges in G' is at most $2|V|$ and $|E|+|V|$, respectively.

In the following paragraphs we give a short review of the results on source location and reformulate the known algorithms in the sink location setting as it is applicable in the shelter location problem. It should be noted that the input of all algorithms in this subsection is the modified network $G' = (V', E')$

3.1.1. Single Source Location Problem

Tamura et al.[16–19] introduced the single source location problem and developped an $\mathcal{O}(nM(n, m))$ algorithm where $M(n, m)$ denotes the time complexity of computing a maximal flow. Initially, the algorithm was only applicable to uniform cost and uniform capacity instances but in the following papers they were able to transfer the algorithm to arbitrary cost and arbitrary capacity instances. Their algorithm is based on so-called minimal deficient sets.

Definition 5. *Let $X \subseteq V$. Then X is a deficient set if* $\mathrm{Val}(u, v) < a(u)$ *for all $u \in X$ and $v \in V \setminus X$. It is called minimal if no proper subset of X is deficient.*

The idea of their algorithm is to pick one node of minimal cost from each minimal deficient set. This algorithm can be applied to the $1|\max|0$ sink location problem (see Algorithm 1).

Algorithm 1 Exact Algorithm for the 1|max|0 Sink Location Problem

Compute the set of minimal deficient sets $\mathcal{X}$.
if V_p is no 1|max|max sink cover **then**
    STOP There exists no 1|max|max sink cover.
else
    $W \leftarrow \emptyset$
    for every minimal deficient set X **do**
        $v_i \leftarrow \arg\min\{c(v_j) : v_j \in X \cap V_p\}$
        $W \leftarrow W \cup \{v_i\}$
    end for
end if

3.1.2. Plural Source Location Problem

Tamura et al.[18, 19] introduced the plural source location problem. Their polynomial solution algorithms where later improved by Ito et al.[22] and Arata et al.[3] to a time complexity of $\mathcal{O}(nM(n,m))$ for a greedy type algorithm and to $\mathcal{O}(n(m + n \log n))$ for an algorithm based on maximum adjacency orderings. However, these algorithms are only applicable to instances where the sources can be ordered in a specific way. $\mathcal{NP}$-hardness for the general problem was initially shown by Arata et al.[3], and Sakashita et al.[14] showed that it is even $\mathcal{NP}$-hard in the strong sense. The adaptation of the $\mathcal{O}(nM(n,m))$-algorithm to the n|max|0 sink location problem is stated as Algorithm 2.

Algorithm 2 Exact Algorithm for the n|max|0 Sink Location Problem

Require: $a(v_1) \leq \ldots \leq a(v_k)$ and $c(v_1) \geq \ldots \geq c(v_k)$
if V_p is no n|max|max sink cover **then**
    STOP There exists no n|max|max sink cover.
else
    $W \leftarrow V_p$
    for $i = 1, \ldots, k$ **do**
        if $W \setminus \{v_i\}$ is a plural sink cover **then**
            $W \leftarrow W \setminus \{v_i\}$
        end if
    end for
end if

3.1.3. Simultaneous Source Location Problem

The simultaneous source location has been considered in detail by Andreev et al.[1]. They give an $\mathcal{O}(\log A)$-approximation (A is the sum of demands) and show that this approximation ratio is best possible. Moreover, they propose algorithms which find solutions exceeding the given edge capacities by at most a constant factor. Algorithm 3 states the resulting $\mathcal{O}(A)$-approximation for the $n|\sum|0$ sink location problem. Here $a_{cov}(S)$ is for any $S \subseteq V_p$ the demand that

can be covered by using only nodes in S as sinks.

Algorithm 3 Approximation for the $n|\sum|0$ Sink Location Problem

1: $W \leftarrow \emptyset$
2: **if** $a_{cov}(V_p) < A$ **then**
3: **STOP** There exists no $n|\sum|\sum$ sink cover.
4: **else**
5: **while** $a_{cov}(W) < A$ **do**
6: $v \leftarrow \arg\min_{v \in V_p \setminus W} \frac{c(v)}{a_{cov}(W \cup \{v\}) - a_{cov}(W)}$
7: $W \leftarrow W \cup \{v\}$
8: **end while**
9: **end if**

In the computation of the minimum in Step 6 of Algorithm 3 we do not recompute all flow values in each iteration. Instead we keep a list of the remaining nodes sorted by their ratio. We recompute only the flows that are needed to get the right value for the best ratio and update the list ordering until the just computed ratio is still the best ratio. This approach is feasible since the ratios will only worsen as S grows. The algorithm can be applied to the plural, simultaneous, uncapacitated sink location problem.

3.2. Modified Algorithms Applied to Original Network

In contrast to the first subsection, we will work in the original network in the following subsection. Instead of modifying the network we propose changes that can be made to the existing source location algorithms in order to make them applicable for capacitated sink location problems. Consequently, the input of the algorithms which are described subsequently, is always the original network $G = (V, E)$ and not $G' = (V', E')$ as in the first subsection.

3.2.1. Single, Non-Simultaneous, IndependentSink Location Problem

Since cost and sink capacity do not play any role in the definition of minimal deficient sets, we propose the following adaption of Step 7 in Algorithm 1:

7: $v_i \leftarrow \arg\min\{c(v_j) : v_j \in X \cap V_p \text{ and } k(v_j) \geq \tilde{a}\}$

After this modification we can use the algorithm to solve the capacitated single Sink Location Problem due to the following theorem[9].

Theorem 6. *W is a capacitated single sink cover if for each minimal deficient set X there exists an $w \in W \cap X$ such that $k(W) \geq \max_{x \in X} a(x)$.*

Proof. Due to the equivalence of uncapacitated single source and single sink problems, we can conclude from Tamura et al.[16–19]: W is an (uncapacitated) sink cover if and only if $W \cap X \neq \emptyset$ for all minimal deficient sets X of G. W is feasible for the capacitated problem if and only if there exists some $w \in W \cap X$ such that $k(W) \geq \max_{x \in X} a(x)$. □

3.2.2. Plural, Non-Simultaneous, Independent Sink Location Problem

For the plural Sink Location Problem we can easily adapt Algorithm 2 by extending Step 6 to include a capacity check. This check is done in the uncapacitated cases by adding a supersource S and edges between the nodes in W and S with infinite capacity. If these capacities are changed to $u(w, S) = k(w)$ for all $w \in W$, then the feasibility check is again equivalent to the solution of a maximal flow problem. The validity of the modified Algorithm 2 thus follows.

3.2.3. Plural, Simultaneous Sink, Additive Location Problem

In Algorithm 3 for the simultaneous Sink Location Problem we only have to adapt Step 6, where in the computation of each $a_{cov}(W \cup \{v\})$ a flow problem needs to be solved. Similarly to the plural Sink Location Problem we add capacitated edges for each chosen sink to guarantee a valid cover in each step of the algorithm.

4. Complexity and Algorithms for New Types of Sink Location Problems

The adaptions in the previous section relied on the fact that the edge capacities and sink capacities were both considered in the same fashion: additive/simultaneous or independent/non-simultaneous. Both, the $1|\max|\sum$ and the $n|\max|\sum$ sink location problem do not have this property and are, thus, not equivalent to any of the source location problems. In this section we give complexity results and heuristic solution methods for these two problems.

4.1. Single, Non-Simultaneous, Additive Sink Location Problem

In contrast to the independent case, the following result shows that it is unlikely that we will be able to solve this problem in polynomial time.

Theorem 7. *The* $1|\max|\sum$ *and the* $1|\sum|\sum$ *sink location problem are NP-hard.*

Proof. We show that any instance of the partition problem can be reduced to a single, additive sink location problem. Let an instance of the partition problem be given by items $1, \ldots, n$ with sizes $a_1, \ldots, a_n$. A solution to the partition problem is a subset $S \subseteq N = \{1, \ldots, n\}$ fulfilling

$$\sum_{i \in S} a_i = \sum_{i \in N \setminus S} a_i. \tag{4}$$

We construct a graph $G = (V, E)$ with vertex set $V = \{v_1, \ldots, v_n, v_S, v_T\}$ and edge set $E = \{(v_i, v_S) : i \in N\} \cup \{(v_i, v_T) : i \in N\}$. Let the edge capacities be $u(e) = A := \sum_{i \in N} a_i$ and the sink capacities $k(v_S) = k(v_T) = \frac{1}{2}A$ and $k(v_i) = 0$ for all $i \in N$. The supply of the nodes v_i is $a(v_i) = a_i$, $i = 1, \ldots n$, and $a(v_S) = a(v_T) = 0$. The graph is shown in Figure 2.

Since the edge capacities are chosen very large, it does not matter whether we consider a simultaneous or a non-simultaneous flow. A feasible solution to

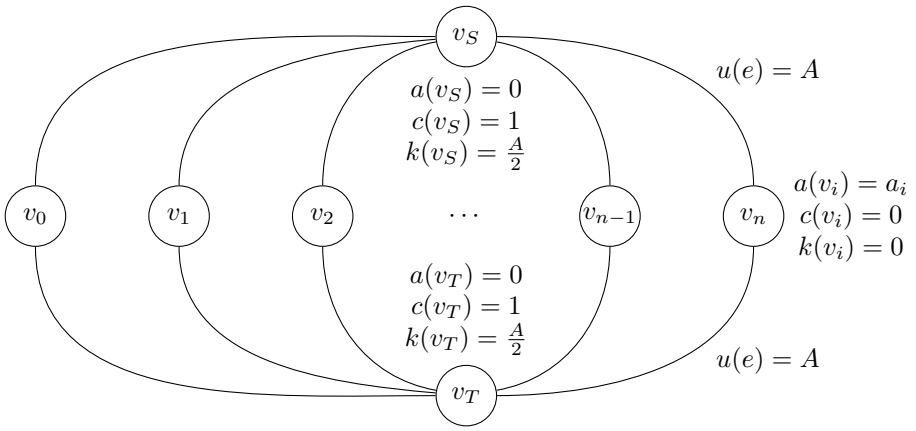


Figure 2: Reduction from Partition to Sink Location

the single, additive sink location problem must choose v_S as well as v_T as sink. Furthermore, it must assign each of the v_i to one of the two sinks. Let S denote the index set of the nodes assigned to v_S and T the index set of those nodes assigned to v_T. Then the solution is feasible if and only if $\sum_{i \in S} a(v_i) \leq \frac{1}{2}A$ and $\sum_{i \in T} a(v_i) \leq \frac{1}{2}A$. Since $\sum_{i \in N} a(v_i) = A$ this is only possible if $\sum_{i \in S} a(v_i) = \frac{1}{2}A = \sum_{i \in T} a(v_i)$. Thus, the single, additive sink location problem has a feasible solution if and only if the partition problem has a feasible solution. □

Due to the close relation to bin packing, especially to variable sized bin packing, we suggest Algorithm 4 based on the best fit heuristic (see [8]).

Algorithm 4 Heuristic for the $1|\max|\sum$ Sink Location Problem

Require: $a(v_1) \geq \ldots \geq a(v_k)$
if V_p is no $1|\max|\sum$ sink cover **then**
  STOP There exists no $1|\max|\sum$ sink cover.
else
  $S \leftarrow \emptyset$
  $rem(v) \leftarrow k(v), v \in V_p$
  for $i = 1, \ldots, k$ **do**
    if there is a $v \in S$ with $rem(v) \geq a(v_i)$ and $\mathrm{Val}(v_i, v) \geq a(v_i)$ **then**
      $v \leftarrow \arg\min_{v \in S}\{rem(v) : rem(v) \geq a(v_i)\}$
    else
      $v \leftarrow \arg\max_{v \in V_p \setminus S} \left\{ \frac{k(v)}{c(v)} : k(v) \geq a(v_i) \text{ and } \mathrm{Val}(v_i, v) \geq a(v_i) \right\}$
      $S \leftarrow S \cup \{v\}$
    end if
    $rem(v) \leftarrow rem(v) - a(v_i)$
  end for
end if

If a source can be assigned to an already chosen sink then it is assigned to the sink with the smallest remaining capacity. If it cannot be assigned to an already chosen sink than a new sink is chosen which has the best capacity to cost ratio and to which the source can be assigned.

4.1.1. Plural, Non-Simultaneous, Additive Sink Location Problem

Since the plural, non-simultaneous, independent sink location problems is already strongly NP-hard[14], this remains true for the plural, non-simultaneous, additive sink location problems.

Theorem 8. *The $n|\max|\sum$ sink location problem is strongly NP-hard.*

Proof. Obviously every plural source location problem can be written as a plural, non-simultaneous, additive sink location problem by choosing sink capacities as $\sum_{v\in V} a(v)$. Since the plural source location problem is strongly NP-hard, the plural, non-simultaneous, additive sink location problem is strongly NP-hard as well. □

On the other hand, the $n|\max|\sum$ sink location problem also has a strong relation to the $n|\sum|\sum$ sink location problems.

Theorem 9. *The modification of Algorithm 3 given in Subsection 3.2.3 can be extended to an $\mathcal{O}(A)$-approximation for the $n|\max|\sum$ sink location problem.*

Proof. Let an instance of the plural, non-simultaneous, additive sink location problem be given by an undirected graph $G = (V, E)$, a set V_p of admissable sinks, edge capacities $u(e)$, supplies $a(v)$, node capacities $k(v)$, and costs $c(v)$. Our aim is to create a new big graph which has a copy of G for each node, thus, separating the flow for each source, and connect these copies such that the sink capacities are considered simultaneously.

For this, create n copies $G^{(i)} = (V^{(i)}, E^{(i)})$ of G and build a new (directed) graph $\tilde{G} = (\tilde{V}, \tilde{A})$ with vertex set

$$\tilde{V} = \bigcup_{i=0}^{n} V^{(i)} \cup \{s, t\} \cup V_p \tag{5}$$

and arc set

$$\tilde{A} = \bigcup_{i=0}^{n} A^{(i)} \cup A_s \cup A_t \tag{6}$$

$$A^{(i)} = \{(u, v) \text{ if } (u, v) \in E \text{ or } (v, u) \in E\} \tag{7}$$

$$A_s = \{(s, v_i^{(i)}) \text{ for } i = 1, \ldots, n\} \tag{8}$$

$$A_t = \{(v_p^{(i)}, v_p) \text{ for } v_p \in V_p\} \tag{9}$$

To complete the plural, simultaneous, additive sink location instance, we set all nodes supplies to 0 except for node s which has supply $\sum_{v\in V} a(v)$. The

capacity of arcs in $A^{(i)}$ are equal to the edge capacities in the original graph, the arc capacities of $(s, v_i^{(i)}) \in A_s$ are equal to $a(v_i)$, and those of $(v_p^{(i)}, v_p) \in A_t$ are ∞. The set of admissable sinks is $\tilde{V_p} = V_p$, the costs for opening one of these sinks $v_p \in \tilde{V_p}$ is $c(v_p)$ and the capacity is $k(v_p)$.

Every solution to this instance has an equivalent solution for the plural, non-simultaneous, additive sink location problem in the original graph G.

Note that although we stated algorithm 3 only for undirected graph it can also be applied to directed graphs with the same performance guarantee. □

An example for the transformation from plural, non-simultaneous, additive to plural, simultaneous, additive sink location is given in Figure 3.

5. Setup of Computational Tests

In this section we will describe the setup for our random tests and also for some realistic tests based on data from the city of Kaiserslautern.

5.1. Random Experiments

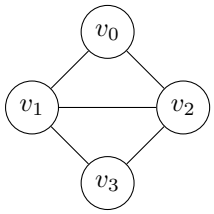
At first a random tree with n nodes is created. Then every edge that is not contained in the tree is created with probability p. We call a graph that is created in this way a connected $G(n, p)$ graph.

Initially, we created the edge capacities, node capacities, node costs, and node supplies as random values drawn uniformly from a given interval. However, we observed that using this routine, nearly every minimal deficient set consisted only of a single node except for one set which consisted of a lot of nodes. The reason for this behavior is the following: Assume the network is given with edge capacities and we wish to find node supplies such that there are some minimal deficient sets that consist of more than one node. Then the supply of the nodes in this set must be

1. large enough such that it cannot be discharged to a node outside the minimal deficient set, but on the other hand,
2. small enough such that it can be discharged to a different node inside the minimal deficient set.

As long as all edges are created with the same probability and also have the same expected capacity, these two conditions have a very low probability to be met at the same time. That is why those nodes that did not meet the first condition were part of the big minimal deficient set, whereas those nodes that did not meet the second condition formed their own single element minimal deficient sets.

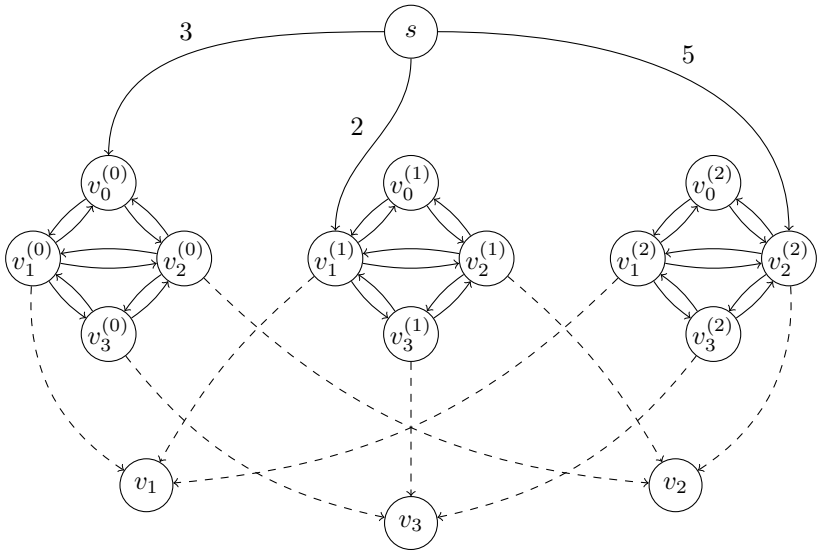
Therefore, we do not create the data independently at random. Instead, we use the following setup: At first, we randomly create a partition of the node set into k connected components $V_1, \ldots, V_k$. We would like to have a high probability that each of these components in the end forms a minimal deficient set. Let v be a node in component V_i and let δ_i denote the number of edges



(a) Original Network

Node	Supply	Capacity	Cost
v_0	3	-	-
v_1	2	5	5
v_2	5	6	5
v_3	-	7	6

(b) Original Data



(c) Transformed Network

Node	Supply	Capacity	Cost
s	10	-	-
v_1	-	5	5
v_2	-	6	5
v_3	-	7	6
$v_i^{(j)}$	-	-	-

(d) Transformed Data

Solid arcs and edges without label have capacity 3 and dashed arcs without label have infinite capacity.

Figure 3: Transformation from Plural, Non-Simultaneous, Additive to Plural, Simultaneous, Additive Sink Location

n	k	p
10	{2}	{0.1, 0.2, 0.3, 0.4, 0.5}
20	{2, 4, 5}	{0.1, 0.2, 0.3, 0.4, 0.5}
50	{2, 5, 10, 15}	{0.1, 0.2, 0.3, 0.4, 0.5}
100	{2, 5, 10, 25}	{0.1, 0.2, 0.3, 0.4, 0.5}
200	{2, 5, 10, 25, 50}	{0.1, 0.2, 0.3, 0.4, 0.5}
500	{2, 5, 10, 25, 50, 100}	{0.1, 0.2, 0.3, 0.4, 0.5}
1000	{2, 5, 10, 25, 50, 100, 250}	{0.1, 0.2, 0.3, 0.4, 0.5}

Table 2: Parameter Settings

going out of component V_i and Δ_i the set of these edges. Then the supply $a(v)$ of v is chosen randomly in $[1, \delta_i U]$, the capacity $k(v)$ is the supply $a(v)$ plus a number chosen randomly between 1 and $\delta_i U$. Finally, the cost $c(v)$ of node v is chosen randomly in $[1, U]$. For edges that have both nodes in different components, we choose a capacity randomly in $[1, U-1]$. These capacities will be small compared to the edge capacities inside the components. An edge from v to u with $u, v \in V_i$ has capacity randomly distributed in $\left[1, 2\frac{a(v)+a(u)}{p(|V_i|-1)}\right]$.

We thus obtain for the expected values $\mathbb{E}$ of the random quantities

$$\mathbb{E}[a(v)] = \frac{1+\delta_i U}{2},$$
$$\mathbb{E}[u(V_i, V \setminus V_i)] = \sum_{(u,v)\in\Delta_i} \frac{1+U-1}{2} = \frac{\delta_i U}{2}, \text{ and}$$
$$\mathbb{E}[u(v, V_i \setminus \{v\})] = \mathbb{E}\left[\sum_{u\in V_i\setminus\{v\}:(v,u)\in E} \frac{1+2\frac{a(v)+a(u)}{p(|V_i|-1)}}{2}\right] \geq \frac{p(|V_i|-1)}{2} + a(v).$$

Hence, it will most likely not be possible to discharge the supply of a node to a node outside its component and, on the other hand, the capacities of the edges inside the component are large enough to discharge the supply to a different node. Using this data generation in the single sink setting we get several minimal deficient sets with more than one element on the one hand, and on the other hand the data is still chosen in such a way that the initial components are only approximations of the minimal deficient sets that come out in the end.

We ran tests on the parameter settings shown in Table 2. U is always set to 10000. For each parameter setting we created ten instances.

5.2. Real World Tests

As a realistic example we consider the evacuation of the city of Kaiserslautern, Germany. As underlying network we use open street map data[2]. The

[2]see http://www.openstreetmap.de/

capacities of the roads - given in vehicles per hour - are set to 6000 for motorways, 4500 for primary roads, 3000 for secondary roads, 1800 for tertiary, residential, and unclassified roads, and 900 for living streets. The values have been taken from MATSim[3] and have proven to be realistic for West European countries. As supplies we use the number of inhabitants which were provided by the city of Kaiserslautern. We assume that 25% of the population will require accomodation in specifically designated shelters and public transportation, 15% only accomodation but no public transportation, and 60% will leave the city and stay with friends or family (*self-contained evacuees*). The shelters for those relying on public transportation are not chosen by our model but by a different model which integrates the location of these shelters and the evacuation by buses[5]. As possible shelters we choose the 20 biggest gymnasiums in Kaiserslautern. The capacities of these gymnasiums were again provided by the city of Kaiserslautern and are available on the internet[4]. The cost to open a shelter is chosen approximately proportional to its size with a slightly lower cost for bigger gymnasiums. In order to deal with the self-contained evacuees, we create artificial shelters at the main roads leaving the city. A person that leaves the city to stay with family or friends in case of an evacuation is then modelled as a flow unit going to one of these artificial shelters. The opening costs for these shelters are 0 but their capacities are chosen in such a way that only the given 60% of the evacuees can go to these shelters.

We consider four different scenarios. Two examples with a small evacuation radius of 500m and two examples with a larger evacuation radius of 1000m. Two different locations for the evacuation are analyzed: One in the city centre of Kaiserslautern and one in the residential zone close to the university of Kaiserslautern. Of course, shelters lying in the evacuation zone are not available in our experiments.

6. Computational Results

The computations were all done on a compute server with a 16-core Intel Xeon E5-2670 processor with 2.60 GHz and 20MB cache, 96 GM RAM and Ubuntu 12.04. All codes were written in C++ using gcc v. 4.6.3 with compile flag -O3. The integer programming formulations were solved using the MIP solver Cplex version 12.4[5], the flow computations including the flow equivalent tree computations were done using the LEMON library[6]. All computations were pinned to a single core and a memory limit of 8 GB was set.

6.1. Random Experiments

Often, in comparisons between heuristics and general IP solvers, the intermediate results of the IP solvers are ignored. As the following discussion shows,

[3] see http://www.matsim.org/
[4] see http://www.kaiserslautern.de/leben_in_kl/freizeit_und_sport/sporthallen/index.html
[5] see http://www.cplex.com
[6] see http://lemon.cs.elte.hu

Figure 4: Color Scheme for Heatmap Diagrams

we do not agree with this approach.

In Figure 5(a) we present a comparison between exact solutions delivered by Cplex and solutions obtained from our Algorithm 3 for the $n|\sum|\sum$ problem. For each instance our algorithm finds a solution with objective value C_3 after time T_3 and Cplex finds a solution with objective value C_{Cplex} after time T_{Cplex}. In order to compare the different instances with each other we compute the time ratio $\frac{T_3}{T_{\text{Cplex}}}$ and the objective ratio $\frac{C_3}{C_{\text{Cplex}}}$. If only our algorithm finds a feasible solution, then we set both ratios to 0, If both, Cplex and our algorithm find no solution we set both ratios to 1, and if only Cplex finds a feasible solution then we set both ratios to the maximal value depicted in the diagram. We collect the solutions of all instances in one diagram. Since the number of solutions is large we use a heatmap type of diagram. The color scheme for these diagrams is given in Figure 4.

In Figure 5(a) you can see that our approximation compares very good to Cplex in terms of objective value as it always finds a solution with objective value at most 1.5 times the value that Cplex finds. Furthermore a lot of instances are either solved much faster by our algorithm than by Cplex (yellow spot around (0.0, 1.0)) or in the same time (red spot around (1.0, 1.0)).

However this diagram does not tell the whole truth as Cplex finds several solutions during its branch and bound. In Figure 5(b) all incumbent solutions are plotted. Compared to 5(a) we see new green regions:

- dark green spot around $(1.0, 0.2)$ ($\sim 5\%$ of all solutions)

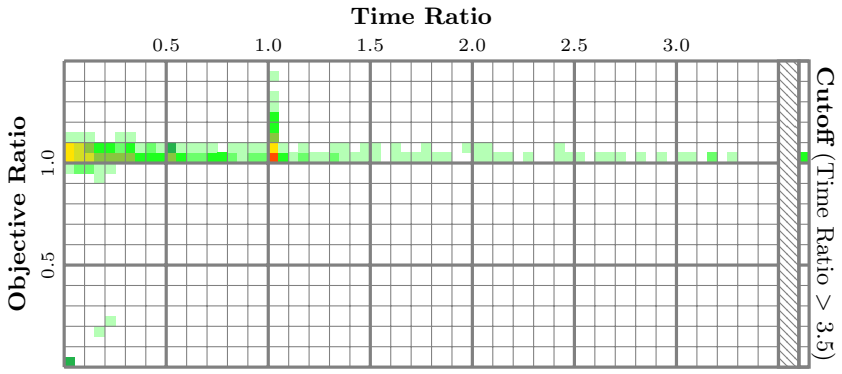
  These (bad) incumbent solutions are found pretty fast by cplex. Since our approximation is pretty fast for almost all instances, these bad incumbent solutions give a point with a time ratio close to 1.0 and a low objective ratio.

- green spots around $(t, 1.0)$ for $t > 1.0$ ($\sim 10\%$ of all solutions)

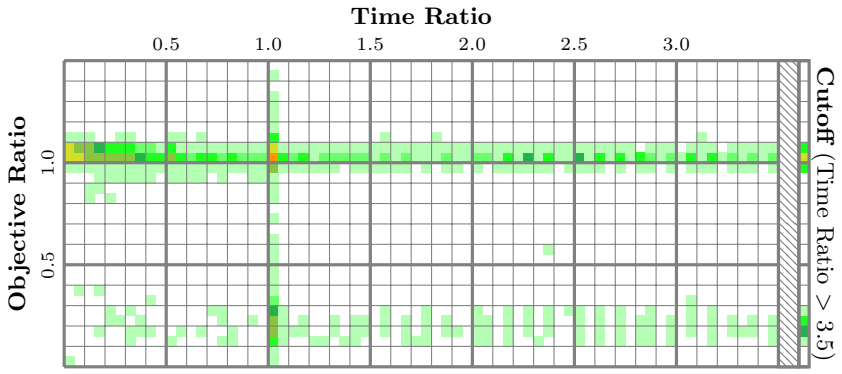
  These (good) incumbent solutions are also found reasonably fast. If our approximation algorithm is also that fast or faster then we get a point on the dark line between (0.0, 1.0) and (1.0, 1.0). However, for several instances our algorithm also takes its time which results in the new spots at $(t, 1.0)$ for $t > 1.0$.

- light green spots around $(t, 0.2)$ for $t > 1.0$ ($\sim 5\%$ of all solutions)

  The solutions corresponding to these spots are, again, solutions that cplex finds very fast. However, they have a bad objective value compared to our approximation.

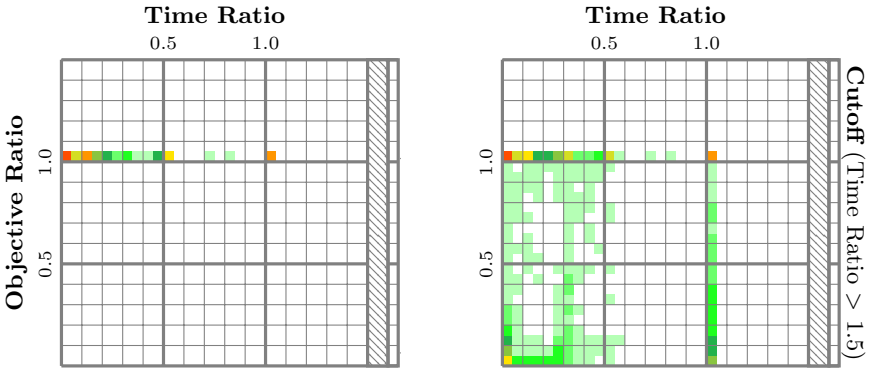


(a) 2590 Final Cplex Solutions



(b) All 21078 Cplex Solutions

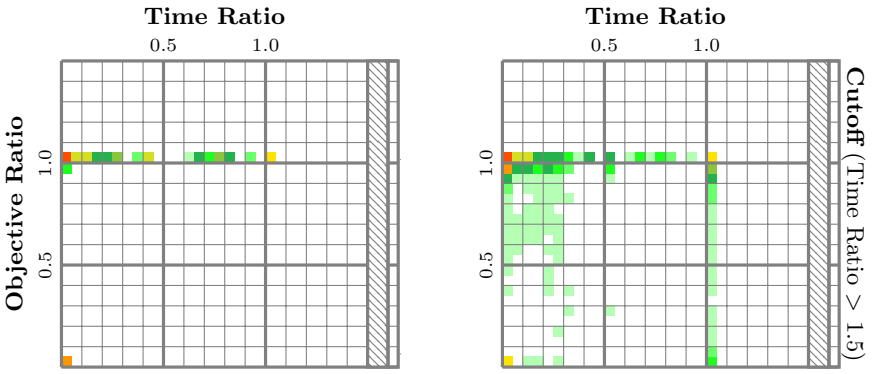
Figure 5: Comparison of Cplex Solutions with Algorithm 3 for $n|\sum|\sum$ Problems applied to the Original Graph



(a) 2590 Final Cplex Solutions

(b) All 7762 Cplex Solutions

Figure 6: Comparison of Cplex Solutions with Algorithm 1 for 1|max|max Problems applied to the Original Graph



(a) 2590 Final Cplex Solutions (b) All 6900 Cplex Solutions

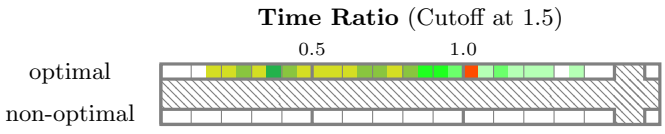
Figure 7: Comparison of Cplex Solutions with Algorithm 2 for $n|\max|\max$ Problems applied to the Original Graph

Whereas Figure 5(a) suggested that our approximation is fast and yields very good solutions, Figure 5(b) shows that in fact there are also a few instances where our approximation is (significantly) slower than Cplex.

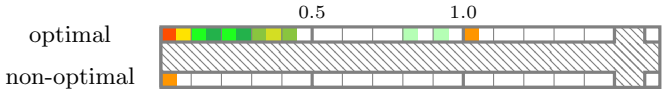
In contrast to this approximation the exact algorithms for the $1|\max|\max$ and the $n|\max|\max$ problem perform very good: Figures 6 and 7 show that even if we consider all incumbent solutions of Cplex, our algorithms still perform much better in terms of objective value and computational time.

In order to compare our adapted algorithms applied to the original network to the original algorithms applied to the node-doubled network we use a different heatmap diagram in Figure 8. In the upper rows all instances that are solved to optimality by both approaches are shown, whereas for the instances in the lower row only our adapted algorithm finds a solution. In the individual rows we show a heatmap of the time ratio between the adapted algorithm and the original algorithm. One can see that for all three problem types $1|\max|\max$, $n|\max|\max$, and $n|\sum|\sum$ our adapted algorithms perform better in general. However, for the $1|\max|\max$ and the $n|\sum|\sum$ problems there are also a lot of instances for which our adapted algorithms are not faster and even a few instances where our adapted algorithms are in fact a little slower than the existing algorithms. In contrast to this our adapted algorithm for the $n|\max|\max$ problem is much faster for most instances and there are even some instances where our adapted algorithm finds a solution in the given time bound of 15 min whereas the original algorithm applied to the node doubled network is not able to find this solution in the given time.

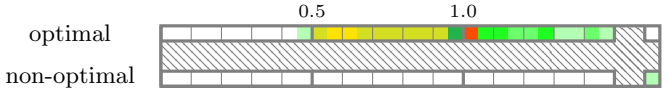
The fact that this improvement in running time for the $1|\max|\max$ and $n|\sum|\sum$ problem is not very high is not unexpected since on the one hand our adapted algorithms do not reduce the computational complexity, and on the other hand the number of flow computations in the node doubled network is seemingly not the bottleneck of the computation, at least for the size instances that we considered. For the $n|\max|\max$ problem we need to compute several



(a) Algorithm 1 for 1|max|max Problems



(b) Algorithm 2 for n|max|max Problems

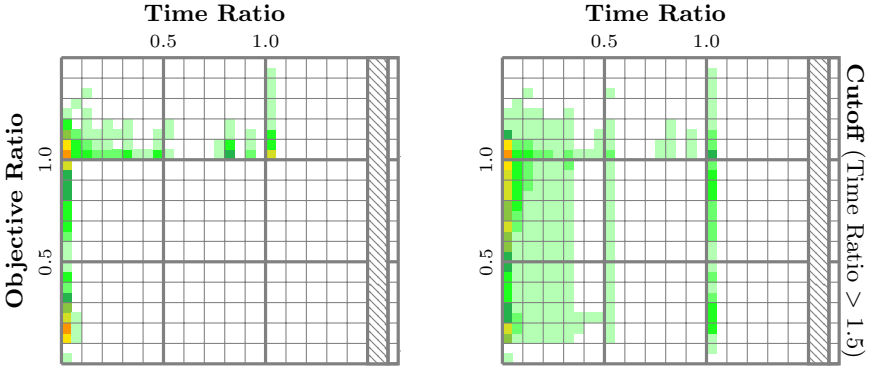


(c) Algorithm 3 for $n|\sum|\sum$ Problems

Figure 8: Comparison of Adapted Algorithms with Original Algorithms applied to the Node-Doubled Graph

flows in each iteration to check if we still have a valid cover. Therefore, for this type of problem doubling the nodes does increase the computation time significantly.

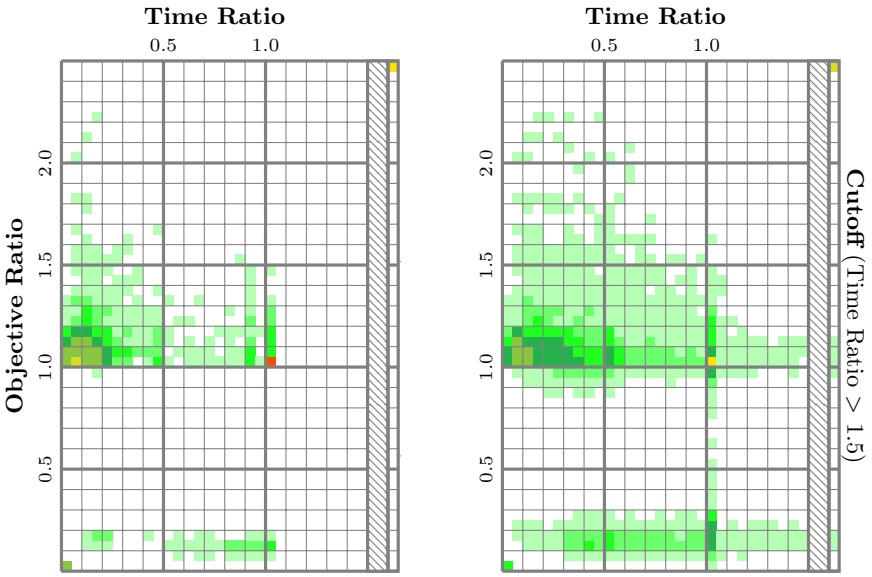
Algorithm 4 for the $1|\max|\sum$ problem performs even better: The computation times are extremely small because after the all-pairs-flow computation no more flow computations are required. In contrast Cplex takes its time even at finding only a feasible solution. Therefore, in the heat diagram in Figure 9 most points lie very close to the vertical axis.



(a) 2590 Final Cplex Solutions

(b) All 24263 Cplex Solutions

Figure 9: Comparison of Cplex Solutions with Algorithm 4 for $1|\max|\sum$ Problems



(a) 2590 Final Cplex Solutions (b) All 10175 Cplex Solutions

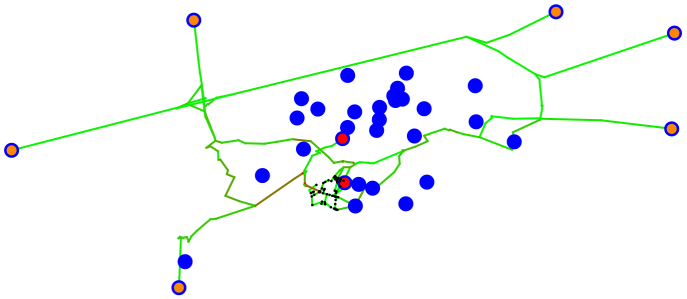
Figure 10: Comparison of Cplex Solutions with the algorithm described in Theorem 9 for $n|\max|\sum$ Problems

Unfortunately, the modifications of Algorithm 3 given in Theorem 9 result in a dramatic increase in memory requirements. As you can see in Figure 10 this algorithm in general does not find better solutions than Cplex. There are a few instances where our algorithm is faster, but there is about the same number of instances where our algorithm aborts due to the memory limit of 8GB (yellow dot around (1.5,2.5)).

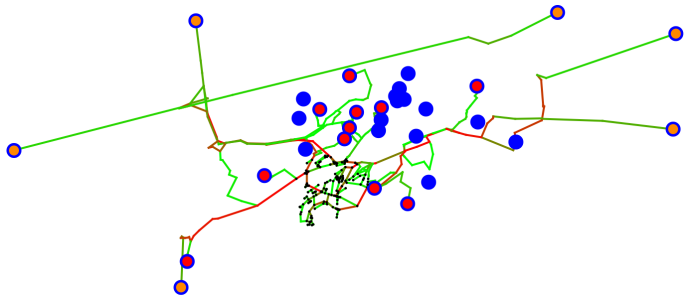
6.2. Real World Tests

In Figure 11 we show the solutions for our real world instances. We considered the situation in which all people in the endangered region have to be evacuated immediately. Thus, we chose the $n|\sum|\sum$ problem type. The exits to the motorway and to streets leaving the closer vicinity of Kaiserslautern are marked by orange nodes. The people staying at relatives or friends during the evacuation will go to these exits. The red nodes mark the shelter locations that are chosen by our algorithm, and the blue nodes are possible shelters that are not chosen. The intensity of the traffic on the roads is highlighted via a heatmap where green means small amount of traffic compared to the street capacity and read means traffic amount (almost) equal to the street capacity.

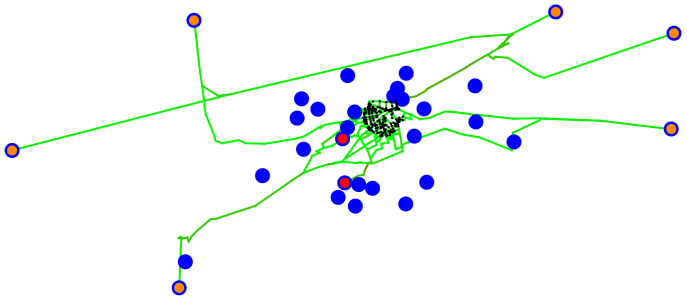
For both small radius instances you can see that only two shelters are needed to accomodate all evauees. Although our algorithm does not consider distances by design, it seems that close shelters are preferred: Those shelters that are closer to the endangered area are significantly smaller and would require to



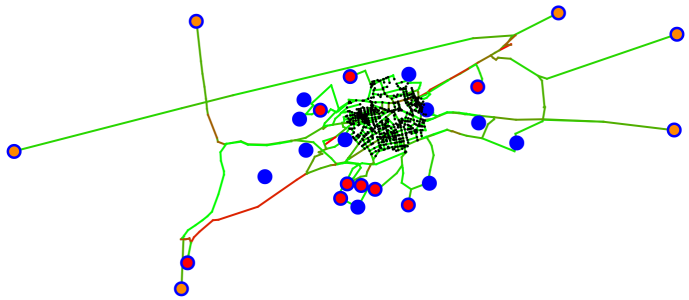
(a) Small Radius at University



(b) Large Radius at University



(c) Small Radius around City Center



(d) Large Radius around City Center

Figure 11: Real World Examples

open at least two more shelters. In the large radius instances, more shelters are needed. They are not all chosen as close as possible, this is due to the need for larger shelters.

For both small radius instances the set of chosen shelters is the same. Similarly, in the large radius instances both solutions share 6 shelter locations and the non-shared 4 other locations lie either close to the university or the city center. Thus, it suffices to maintain a relatively small number of shelters and, yet, still be prepared for an evacuation no matter what part of the city has to be evacuated.

When you look at the heatmap you can see that the road leaving the city to the south west is at its upper capacity for both large radius instances and carries heavy traffic even in the small radius university instance. Also the western as well as the eastern access road to the motorway sees some heavy traffic. Especially in the large radius university instance you can see that this will probably result in traffic jams inside and close to the endangered area.

7. Conclusion and Future Research

We have introduced the sink location problems in its different variants: single and plural, independent and additive, non-simultaneous and simultaneous covers. Although some of the sink location problems are equivalent to source location problems as far as computational complexity is concerned, we have shown that algorithms adapted to the sink location setting are much faster than applying node doubling and using algorithms from source location. For those problems that do not have an equivalent source location problem we have given new algorithms. In Table 3 an overview on the computational complexity of the different types of sink location is given. New results from this paper are highlighted in bold font.

Sink Assignment	Sink Capacity	Edge Capacity	Computational Complexity
single	indep.	non-sim.	$\mathcal{O}(nM(n,m))$ [16–19]
single	additive	non-sim.	**NP-hard (Theorem 7)**
plural	indep.	non-sim.	strongly NP-hard [3, 14]
plural	additive	non-sim.	**strongly NP-hard (Theorem 8)**
plural	additive	sim.	strongly NP-hard [1]

Table 3: Computational Complexity of Sink Location Problems

We compared all algorithms to the commercial integer programming solver Cplex. It turns out that the combinatorial algorithms perform, in general, better than Cplex. Only for the $n|\sum|\sum$and the $n|\max|\sum$ problem there is a significant but still low number of instances where Cplex performs better than our algorithms.

As mentioned at the end of Section 2 there are other sink location models that require multicommodity network flows. Although they do not have an obvious practical application they might be of theoretical interest. Results for these problems will be presented in the forthcoming PhD thesis of Heßler[7].

For source location there are also approaches to incorporate travel times or distances into the problem, see for example the diploma thesis of Turner [20]. It is worthwhile to consider dynamic network flows instead of static network flows for sink location as well. In fact, this would increase the realism of our models for evacuation planning considerably, since evacuation time is an important aspect of the evacuation. Whether these dynamic problems are still solvable in reasonable time will be the subject of future work.

References

[1] Konstantin Andreev, Charles Garrod, Daniel Golovin, Bruce Maggs, and Adam Meyerson, *Simultaneous source location*, ACM Transactions on Algorithms **6** (December 2009), no. 1, 16:1–16:17. Article 16.

[2] ManWo Ng ans Junsik Park and S. Travis Waller, *A hybrid bilevel model for the optimal shelter assignment in emergency evacuations*, Computer-Aided Civil and Instrastructure Engineering **25** (November 2010), no. 8, 547–556.

[3] Kouji Arata, Satoru Iwata, Kahuhisa Makino, and Satoru Fujishige, *Locating sources to meet flow demands in undirected networks*, Algorithm Theory - SWAT 2000, July 2000, pp. 300–313.

[4] Satoru Fujishige and Naoyuki Kamiyama, *The root location problem for arc-disjoint arborescences*, Discrete Applied Mathematics **160** (September 2012), no. 13-14, 1964–1970.

[5] Marc Goerigk, Kaouthar Deghdak, and Philipp Heßler, *A comprehensive evacuation planning model and genetic solution algorithm*, Transportation Research Part E: Logistics and Transportation Review (2014). to appear.

[6] Horst W. Hamacher and Stevanus A. Tjandra, *Mathematical modelling of evacuation problems: A state of art*, Frauenhofer Institut Techno- und Wirtschaftsmathematik, Kaiserslautern, 2001.

[7] Philipp Heßler, *Mixed planar and network location with applications in evacuation planning*, Ph.D. Thesis, 2015. forthcoming.

[8] D. Johnson, A. Demers, J. Ullman, M. Garey, and R. Graham, *Worst-case performance bounds for simple one-dimensional packing algorithms*, SIAM Journal on Computing **3** (1974), no. 4, 299–325.

[9] Lätitia Kissel, *Sink location in networks and application to shelter planning*, Diploma Thesis, 2012.

[10] Sirisak Kongsomsaksakul, Chao Yang, and Anthony Chen, *Shelter location-allocation model for flood evacuation planning*, Journal of the Eastern Asia Society for Transportation Studies **6** (2005), 4237–4252.

[11] Satoko Mamada, Takeaki Uno, Kazuhisa Makino, and Satoru Fujishige, *An $o(nlog^2n$ algorithm for the optimal sink location problem in dynamic tree networks*, Discrete Applied Mathematics **154** (November 2006), no. 16, 2387–2401.

[12] Dwi Poetranto Groß, *Network flow and location (FlowLoc)*, Ph.D. Thesis, 2008.

[13] Carmen G. Rawls and Mark A. Turnquist, *Pre-positioning planning for emergency response with service quality constraints*, OR Spectrum **33** (May 2011), no. 3, 481–498.

[14] Mariko Sakashita, Kazuhisa Makino, and Satoru Fujishige, *Minimum cost source location problems with flow requirements*, LATIN 2006: Theoretical Informatics, January 2006, pp. 769–780.

[15] Hanif D. Sherali, Todd B. Carter, and Antoine G. Hobeika, *A location-allocation model and algorithm for evacuation planning under hurricane/flood conditions*, Transporation Research Part B **25B** (1991), no. 6, 439–452.

[16] Hiroshi Tamura, Masakazu Sengoku, Shoji Shinoda, and Takeo Abe, *Location problems on undirected flow networks*, IEICE Transactions **E73-E** (December 1990), no. 12, 1989–1993.

[17] ———, *Some covering problems in location theory on flow networks*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E75-A** (June 1992), no. 6, 675–684.

[18] Hiroshi Tamura, Hidehito Sugawara, Masakazu Sengoku, and Shoji Shinoda, *On a generalization of a covering problem called single cover on undirected flow networks*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E80-A** (March 1997), no. 3, 544–550.

[19] ———, *Plural cover problems on undirected flow networks*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **J81-A** (1998), 863–869. in Japanese.

[20] Lara Turner, *Dynamic source location*, Master's Thesis, 2003.

[21] Lara Turner, Horst W. Hamacher, Sven O. Krumke, and Dwi Poetranto Groß, *Source location in static and dynamic setting*, 2009. Working paper.

[22] Hideyuki Uehara and Mitsuo Yokoyama, *A faster and flexible algorithm for a location problem on undirected flow networks*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **83** (2000), no. 4, 704–712.