Integer Flow with Multipliers
— The Special Case of Multipliers 1 and 2 —

Katharina Beygang, Birgit Engels, Sven Krumke,
Rainer Schrader, Christiane Zeck

May 6, 2009

Abstract

The problem to find a valid integer flow with flow multipliers on nodes or arcs is long known to be NP-complete [10]. We show that the problem is still hard when restricted to instances with a limited number of integral multipliers. To find an integer minimum cost maximal pseudoflow, respecting only the node balance constraints of the inner nodes, is also still a difficult task. Further, we demonstrate that for the multipliers 1 and 2 optimal solutions with fractions $\frac{1}{2^n}, n \in \mathbb{N}$ can occur. For special instances which are motivated by some applications we prove that the optimal solution is halfintegral. Finally, we extend the *Successive Shortest Path Algorithm* [2, 7, 9], to the minimum cost flow problem with multipliers. For the application based instances with halfintegral optimal solutions, we try to find acceptable integral solutions.

1 Introduction

A flow $f : A \to \mathbb{R}$ is a function, which assigns a flow value $f(a_{ij})$ to each arc of a digraph $N = (V, A)$ (called network), such that the following properties hold:

- Capacity constraint:

$$\forall a_{ij} \in A : l_{ij} \leq f(a_{ij}) \leq u_{ij}$$

- Node balance constraint:

$$\forall v_i \in V : \sum_{a_{li}=(v_l,v_i)\in A} f(a_{li}) - \sum_{a_{ik}=(v_i,v_k)\in A} f(a_{ik}) = b(v_i)$$

Generalized flows or flows with gains and losses differ from such flows in networks only in one respect: flow (balance) contraints hold for all the vertices v_i with node balances $b(v_i)$ but no longer for all arcs a_{ij} or equivalently for all arcs, but no longer for the vertices: A unit of flow entering an arc a_{ij} (vertex v_i)

can result in more or less than one unit leaving the arc (the vertex), depending on the arc (node) multiplier μ_{ij} (μ_i). With $f_{in}(a_{ij})$ ($f_{in}(v_i)$) and $f_{out}(a_{ij})$ ($f_{in}(v_i)$) denoting the incoming and outgoing units of flow on a_{ij} (in and out of v_i) instead it holds:

$$f_{out}(a_{ij}) = \mu_{ij} \cdot f_{in}(a_{ij}) \text{ resp. } f_{out}(v_i) = \mu_i \cdot f_{in}(v_i)$$

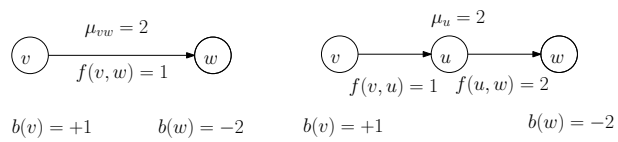

Figure 1: Multipliers on arcs and on nodes.

Both variants of flow multipliers are equivalent in the following sense: A node multiplier μ_i on v_i can be interpreted as an arc multplier μ_{ij} on all outgoing arcs $(v_i, v_j))$ of v_i. On the other hand, if all outgoing arcs of one node have the same multiplier μ_{ij}, the node multiplier also equals this value. For a number $\#\mu_{ij}$ of different arc multipliers on outgoing arcs of v_i, we introduce $\#\mu_{ij}$ new nodes $v_{ik}, 1 \leq k \leq \#\mu_{ij}$ add the arcs (v_i, v_{ik}) and set $\mu_i = 1$. The originally outgoing arcs of (v_i, x) are now changed to arcs (v_k, x) and the values μ_k are set according to the desired arc multipliers (see Figure 2). In the worst case, $m = |A|$ nodes and edges are added, but the necessary modifications are thus polynomial.

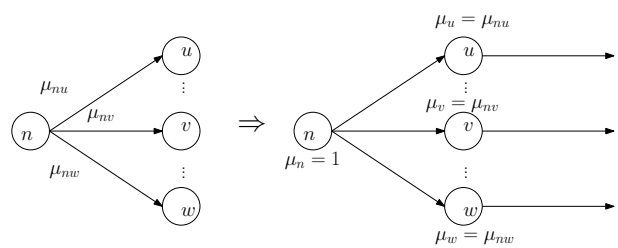

Figure 2: Multipliers on arcs and on nodes are equivalent.

Introducing the flow multipliers generally destroys total unimodularity of the network matrix and thus integral optimal solutions are no longer granted or even impossible (see example in Figure 3). It is known that obtaining an optimal integer solution is NPO-complete for generalized flows [6, 10]. There seem to be no results for the special case where the multipliers are restricted to a few fixed numbers or even to the single additional multiplier 2. However, this spacial case occured when we modelled a railway disposition problem.

We encountered some 'merely generalized' flow instances with the property that the underlying network is essentially a bipartite digraph, all excesses, de-

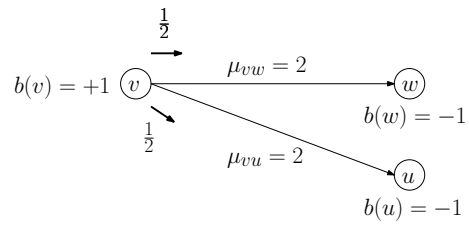

Figure 3: Example: The only valid flow is fractional.

mands, arc capacities and costs can be assumed to be integral and we need only multipliers 1 and 2. In fact, we observed that our network model obeys the following definition:

Definition 1 *A* disposition network $N = (V = X \cup Y, A)$ *is bipartite digraph with* $\mu_a \in \{1, 2\}$ *for all arcs* a *and all arcs with* $\mu(a) = 2$ *are directed from* X *to* Y*. Moreover, for all paths from a source* s *to a sink* t *the product of multipliers of all arcs along the path (path multiplier) is* either 1 or 2*.*

The restriction to additional multiplier 2 reopens the question of the problem's complexity, as we will see in section 2. In Section 3 we construct an example which shows that the additional multiplier 2 can cause optimal solutions with arbitrary small fractions $\frac{1}{2^n}, n \in \mathbb{N}$ (depending on the number of vertices of the graph). Maximum flow or minimum cost (pseudo-)flow problems in disposition networks on the other hand always yield halfintegral optimal solutions. To solve the latter instances, we modify the successive shortest path algorithm in Section 5. Finally, Section 6 is dedicated to heuristic approaches to obtain integral solutions on the basis of the halfintegral solutions in the case of disposition networks.

2 Complexity

The original NP-completeness proof for generalized flows (for the node multiplier variant) by Sahni [10] employs the subset sum problem, which is defined as follows:

> [SubsetSum] Given a multiset $S = \{s_1, \ldots, s_r\}$ of positive integers and a positive integer M, ($M \leq \sum_{i=1}^{r} s_i$) does there exist a submultiset of S that sums up to M?

For general positive integer values, the subset sum problem is NP-complete. For the convenience of the reader, we recapitulate the reduction: For a given subset sum instance S, construct a network N_S with a source node s, nodes $n_i, 1 \leq i \leq r$ with node multiplier s_i for every integer $s_i \in S$ and a sink node t with a demand of $b(t) = -M$. The source is connected to the n_i by arcs (s, n_i)

with capacity and multiplier 1. The nodes n_i are connected to t by arcs (n_i, t) with capacity s_i (see Figure 4). There is an integral $s-t$-flow if and only if there is a submultiset of S that sums up to M. (Concerning the appropriate node balance $b(s) = r$ and the node balance contraints, we have to add another sink t' with $b(t') = -(\sum_{s_i \in S} s_i - M)$ and the arcs (n_i, t') to Sahni's construction to guarantee a feasible flow solution.)

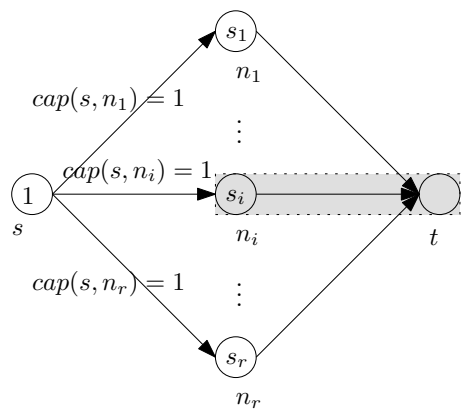

Figure 4: Original reduction to subset sum: Node multiplier values s_i in the nodes n_i, all missing capacities are unlimited and the node balance at t is $b(t) = -M$.

With all multipliers restricted to 1 or 2 and the above network construction, we can only represent subset sum instances where $s_i \in \{1, 2\}$, which does not suffice for the reduction. Therefore, in the network N_S we replace each arc (n_i, t) (grey shaded box in Figure 4) by a subgraph $N_S|n_i$ like the one shown in Figure 5. The purpose of the subgraph is to amplify the one unit of flow which is the upper capacity bound on arc (s, n_i) to the appriopriate s_i flow units which have to arrive at t for the original unit. Further we want to use only node multipliers 1 and 2 in the subgraph. We will see that producing the appropriate number of flow units resembles the flow equivalent of a bit representation of each s_i.

The subgraph $N_S|n_i$ again consists of two parts: Given s_i, we first see how many bits we need to represent the value. The number of bits z_i we need here is not the length of the binary representation $|(s_i)_2|$, but the number of set bits in it, e.g.:

$$s_i = 31 = (11111)_2 : z_i = 5 \text{ or } s_j = 27 = (11011)_2 : z_j = 4$$

For each set bit in s_i's binary representation, we need a node $b_j, 1 \leq j \leq z_i$ in $N_S|n_i$ such that one unit of flow enters b_j if and only if one unit of flow enters n_i. Let $p(b_j)$ be the valency, i.e. the position of bit b_j in the bit representation. Then we add a path $\pi_{b_j t}$ of length $p(b_j) - 1$ to $N_S|n_i$ where all node multipliers

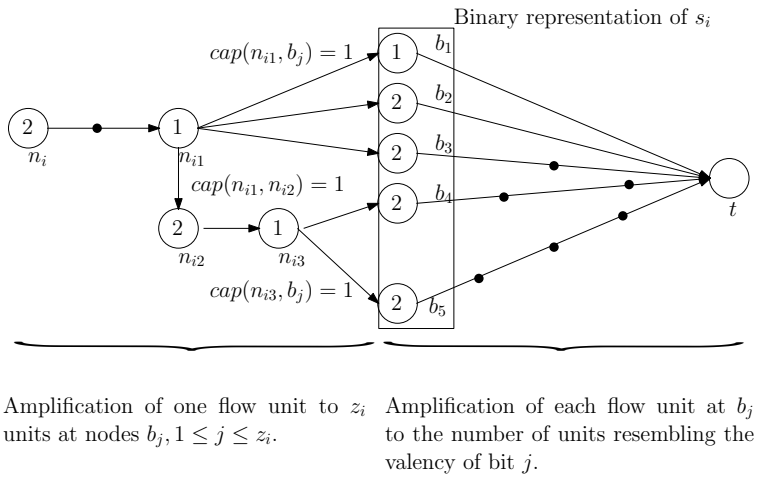

Figure 5: Subgraph $N|n_i$ for $s_i = 31$: Node multipliers are written in the nodes and all black dots are nodes with multiplier 2. Missing capacities are unlimited.

are set to 2. If $p(b_j) - 1 = 0$, we add the arc (b_j, t) and set the node multiplier of b_j to 1. Now the sum of flow units entering t from all b_j equals s_i.

Now we have to connect the b_j to s: Recall that there is the arc (s, n_i) with capacity 1 and that s has the node multiplier 1. Let l with $2^l < z_i$ be maximal. Then we add an amplification path $\pi_{sn_{i1}}$ of length l and with node multipliers 2 on every node, except for n_{i1}, which has node multiplier 1. For one unit of flow, starting on s there are now 2^l units arriving at n_{i1}. We connect n_{i1} to the nodes $b_j, 1 \leq j \leq 2^l - 1$ by arcs (n_{i1}, b_j) with capacity 1, such that the first $2^l - 1$ nodes b_j receive exactly one unit of flow if and only if one unit of flow starts from s to n_i.

Then we add two extra nodes n_{i2}, n_{i3} with node multipliers 2 and 1 and the arcs (n_{i1}, n_{i2}) and (n_{i2}, n_{i3}) with capacity 1. For the one unit left at n_{i1}, we have now two units at n_{i3}. We add the arc (n_{i3}, b_{2^l}) to cover b_{2^l} with one of the two remaining units. Further we add n_{i4} and arc (n_{i3}, n_{i4}) with capacity 1. Now at node n_{i4} we have the same situation as before at node n_{i1}, only that we have to generate $z_i - 2^l$ units of flow and distribute them appropriately to the remaining nodes $b_j, 2^l + 1 \leq j \leq z_i$. This can be done exactly as before, determining the maximum l' with $2^{l'} < z_i - 2^l$ and adding the amplification path, some new nodes and the arcs with capacity 1 to the bit nodes b_j.

The procedure terminates because each number z_i has a binary representation. Further, this representation is bounded in length by $\lceil log_2(z_i) \rceil$, thus we have to repeat the above procedure at most $\lceil log_2(z_i) \rceil$ times and each amplification path itself is at most $\lceil log_2(z_i) \rceil$ nodes long. The same is true for the number z_i of bit nodes b_j and the paths from b_j to t. Thus the construction of each $N_S|n_i$ is polynomial.

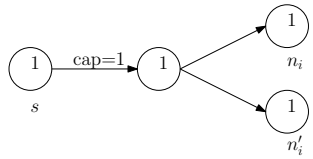

Figure 6: Forking Subgraph : Node multipliers are written in the nodes and missing capacities are unlimited.

To determine $b(s)$ and still obtain a feasible flow if and only if there is a subset $S' \subseteq S$ with $\sum_{s_i \in S'} s_i = M$, we duplicate each subgraph $N_S|n_i$ as $N_S|n_i'$ and add another sink t' which substitutes t for the duplicate subgraphs. Then we delete each arc (s, n_i) and add the subgraph depicted in Figure 6, where s, n_i, n_i' are corresponding nodes which already belong to N.

Now we can set $b(s) = r$. Thus for a valid integral flow solution r units leave s and result in s_i units entering t or t'. And it holds:

Theorem 1 *The decision problem for a valid integral flow with node multipliers 1 and 2 and where all values are integral is NP-complete.*

Proof
Given an instance $S = \{s_i, 1 \leq i \leq r\}, M$ of subset sum, we construct a network as described above. Then for each unit of flow leaving s to a node n_i or n_i', exactly s_i units of flow enter t or t'. Thus, the sinks t, t' with $b(t) = -M, b(t') = -(\sum_{i=1}^{r} s_i - M)$ can be balanced by an integral flow from s if and only if there is a subset $S' \subseteq S$ with $\sum_{s_i \in S'} s_i = M$ (and simultaniously $\sum_{s_i \in S \setminus S'} s_i = \sum_{s_i \in S} s_i - M$). Moreover the problem is in NP: An NTM guesses the flows and verifies the balance and capacity constraints.

□

As we observed above, our application deals with instances of disposition networks, which do not include the network N_S we used in the construction for the proof of Theorem 1. Now we will proceed with a reduction from the following satisfiability problem (which is also simpler) using the arc multiplier variant for the construction and see that we obtain a disposition network:

[3V2L3SAT] Given a boolean formula

$$\alpha = C_1 \wedge \ldots \wedge C_n, C_i = l_{i1} \vee l_{i2} \vee l_{i3},$$

$$l_{ij} \in \{v_k, \neg v_k | 1 \leq k \leq m\} \cup \{0\}$$

and each variable v_k can only occur 3 times in total and maximal 2 times as one of the corresponding literals. Decide whether there is a truth assignment satisfying α.

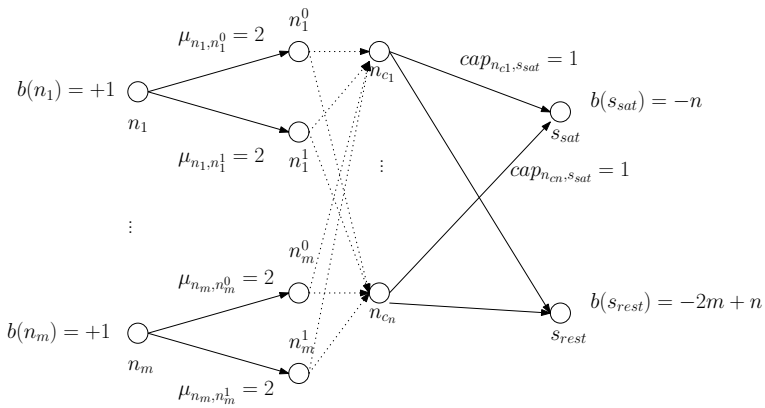

Figure 7: Construction of network N_α.

The problem 3V2L3SAT is NP-complete (as shown by C.A.Tovey [11]) where only the limited number of variable occurences is guaranteed by simply introducing k new variables $x_1, \ldots, x_k$ for k occurences of a variable x and exchange the ith occurence of x by x_i. Then the clauses $(x_i \vee \neg x_{i+1}), 1 \leq i < k$, and $(x_k \vee \neg x_1)$ are added to the original formula, which ensure an equal truth assignment on all x_i. This reduction from 3SAT shows that the satisfiability problem is still hard for instances where each clause contains two or three literals and each variable occurs only three times. Moreover the reduction results in a boolean formula where each literal is present at most two times: surely once in the additional clauses and possibly once in exchange for the original variable in the original formula. Thus even 3V2L3SAT is still NP-complete.

For a given instance α of 3V2L3SAT, we construct the following network N_α: We have vertices n_k for every variable v_k and additional two vertices n_k^0, n_k^1 for the corresponding literals $\neg l_k, l_k$, which occur in α. For every n_k the node balance b_k is $+1$ and the node balance of the literal nodes is 0. Nodes n_k are connected to 'their' literal nodes by arcs $(n_k, n_k^0), (n_k, n_k^1)$ with an arc multiplier of 2.

For each clause C_i in α we need another vertex n_i, which has incoming arcs (n_k^*, n_i) from those literal nodes which correspond to literals occuring in C_i. Finally we add two sink nodes s_{sat}, s_{rest} with node balances $b_{sat} = -n$ and $b_{rest} = -2m + n$ to the graph. We connect the clause nodes to both sinks by arcs (n_i, s_{sat}) with capacity 1 and arcs (n_i, s_{rest}) with unlimited capacity (see Figure 7).

Unless otherwise noted, any node n_j has balance $b_j = 0$ and any arc e_{ij} has multiplier $\mu_{ij} = 1$, unlimited capacity and zero costs. Note that the constructed network meets all restrictions, we found in our model.

Theorem 2 *The decision problem for a valid integral flow in a disposition network is NP-complete.*

Proof

Consider N_α as constructed above. N_α is a disposition network, as the set of vertices can be partitioned into the sets X and Y, such that for all arcs $a = (u, v)$ u and v are in different sets, all arc multipliers are 1 or 2 and all arcs with multiplier $\mu(a) = 2$ are directed from X to Y. All paths from any source to all sinks have path multiplier 2.

Further, for a valid solution, one unit of flow has to leave each variable node n_k **either** to n_k^0 **or** to n_k^1 as the flow has to be integral. Now we have 2 units of flow at one half of all literal nodes which sum up to $2m$ units of excess exactly as the sum of deficits at the sinks. Thus there is a valid flow, if there are paths to the sinks with appropriate capacity.

As the literal nodes only occur in the network, if the corresponding literal occurs in α, each such node is connected to at least one clause node n_i, which is again connected to both sinks. As the arcs (n_i, s_{rest}) have no capacity limits, the required flow units can always pass from arbitrary literal nodes to s_{rest}. On the other hand, s_{sat} can be reached by only one unit of flow via each clause node. To balance the full deficit $b_{sat} = -n$ and at the same time satisfy all node and capacity constraints, at least one unit of flow must pass each clause node.

We can thus reinterpret the flow through a literal node n_k^0 resp. n_k^1 as a truth assignment ν with $\nu(v_k) = 0$ resp. $\nu(v_k) = 1$. Each unit of flow through a clause node n_i must then correspond to a true literal in the corresponding clause. A valid flow, especially carrying n flow units to s_{sat}, therefore corresponds to a truth assignment satisfying at least one literal in every clause and thus satisfying the whole formula α.

If there is no valid integral flow, we can also not find a satisfying truth assignment. Moreover the problem is in NP (Theorem 1).

□

In our real world instances the node balances of source(s) and sink(s) are not guaranteed to be equal (in sum) and there need not be paths of enough total capacity. Thus, we are also interested in maximal integral $(s - t)$-pseudoflows with minimum costs, where $(s - t)$-pseudoflow means a pseudoflow respecting all non-negativity and capacity constraints, as usual pseudoflows, but also all node balance constraints except from the balance(s) of the sink node(s).

As a $(s - t)$-pseudoflow solution is not a valid flow solution itself, we will slightly modify the above construction to clarify that our problem remains hard. For practical instances we can assume the deficit at sinks to be greater or equal to the excess at sources.

Theorem 3 *The problem of finding a maximal integral $(s - t)$-pseudoflow with minimum costs in a disposition network is NP-complete.*

Proof

We apply the same construction as above, but add a supersink node s with

$b(s) = 3m$ and connect the former sink nodes s_{SAT} and s_{rest} to s. The balances $b(s_{SAT})$ and $b(s_{rest})$ are set to zero and arc (s_{SAT}, s) has capacity n and cost zero while (s_{rest}, s) has infinite capacity (or at least $3m$) and high costs M. The maximum $2m$ units of excess on the literal nodes emanating from the source nodes can now arrive at s at cost $2m \cdot M - n$ if and only if α is satisfiable by the construction and argumentation of 2. If α is not satisfiable the maximal integral $(s-t)$-pseudoflow solution with minimum costs has cost strictly higher than $2m \cdot M - n$.

□

3 Fractional optimal solutions

An optimal solution to the min cost flow problem with arbitrary multipliers is generally not integral. Even if we limit the possible multipliers to the values 1 and 2, we can construct an extendable example, where the fractions of flow can get arbitrarily small ($\frac{1}{2^n}$ in a Graph $G(V, A), |V| = 3n$) in the optimal solution.

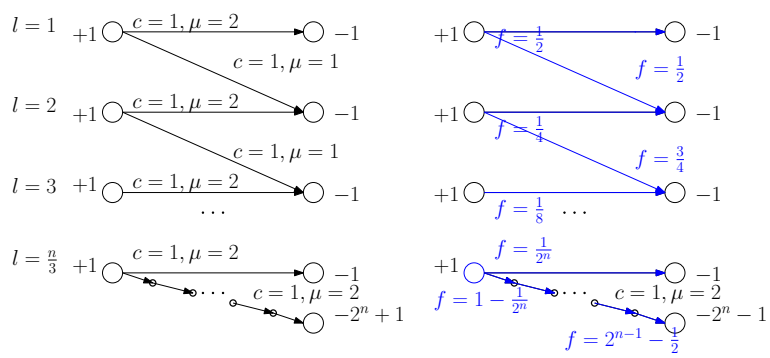

Figure 8: An example of arbitrary small flow fractions.

Consider Figure 8: At each level l the maximum possible flow is carried from source s_l to sink t_l, starting from level 1, where there is no other way of balancing t_1. The rest excess of s_l must then be transfered to t_{l+1} where it causes a fractional rest deficit of $\frac{1}{2^l}$ which is balanced by $\frac{1}{2^{l+1}}$ flow out of s_{l+1} due to the flow multiplier of 2. The remaining excess of source s_{l+1} is then $1 - \frac{1}{2^{l+1}}$ and so on. For this part of the graph we need $2n + 1$ vertices to create a flow of $\frac{1}{2^n}$ units. To ensure a valid flow solution and still remain integral in all demands, excesses, and only use the multipliers 1 and 2, we have to create a path of $n-1$ vertices connecting s_n and t_n on the last level with an arc multiplier of 2 on each arc, such that the last rest excess is completely comsumed by the last integral deficit.

The same construction can be applied to the min cost maximum pseudoflow case with appropriate cost on the arcs.

4 Halfintegral optimal solutions

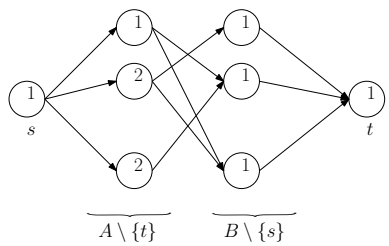

Figure 9: Simple disposition network: All arcs (s, a_i) and (b_j, t) have zero cost and capacities $cap(a_i)$ resp. $cap(b_j)$. Arcs (a_i, b_j) have infinite capacity and costs $cost(a_i, b_j)$.

Figure 9 shows a simple disposition network. We will transform a disposition network instance into a generalized minimum cost flow circulation instance G' as follows: We split node s into the nodes s_1 and s_2 and connect all nodes a_i with multplier 1 to s_1, the remaining nodes of A to s_2 while the arcs $(s_l, a_i), l \in \{1, 2\}$ have the same cost and capacity values as before. Further, we add two nodes t_1 and t_2 and arcs (t, t_l) with cost zero and unlimited capacity and $(t_l, s_l), l \in \{1, 2\}$ with cost $-\infty$ and unlimited capacity. Nodes t, t_1 have multiplier 1, node t_2 multiplier $\frac{1}{2}$. Now the IFNM1,2 solution on G is equivalent to the minimum cost flow circulation solution on G' with the following property:

Lemma 1 *Given a feasible circulation f every cycle in the residual network G'_f is a unit gain cycle, i.e. the product of the multipliers of the nodes in the cycle is* 1.

Proof
Denote by S_1 resp. S_2 the set of nodes containing the source s_1 resp. s_2 and all (supply) nodes a_i connected to s_1 resp. s_2. Let the set T contain all (demand) nodes b_j and the nodes t, t_1, t_2. By construction flow is generated (multiplied by 2) only on arcs from S_2 to T and destroyed (devided by 2) only on arcs from T to S_2. As there are no arcs between nodes of S_1 and S_2 or vice versa, each cycle has to contain as many arcs from S_2 to T as from T to S_2 and thus the product of all node multipliers along any such cycle is 1. (Moreover, it follows, that every path in the residual network has a path multiplier, i.e. product over all node multipliers of the nodes along the path, of either $\frac{1}{2}$, 1 or 2.)

□

It is well known that a feasible generalized circulation f is optimal if and only if G'_f contains no negative cost circuits, i.e. circulations that send positive flow only along the arcs of a single residual unit gain cycle (or bicycle). Based on this observation, we can solve the generalized minimum cost flow or circulation problem with the *Cycle Cancelling* (CC) Algorithm, which successively detect circuits with negative costs in the residual network and cancels them by augmenting the maximum possible amount of flow along them. (Thus at least one of the cycle's arcs is saturated. Due to the multipliers this is not neccessarily the one with least residual capacity.)

By Lemma 1, we can limit our further argumentation to the cancelling of unit cycles.

Theorem 4 *Cycle Cancelling always yields a half-integral solution to the minimum cost circulation problem on G'.*

Proof
Let all arcs with tail in S_2 comprise the set A_2, all other arcs are contained in A_1. We show that during the algorithm the residual capacity on all arcs from A_1 resp. A_2 remains integral resp. half-integral. Clearly, this property holds for the zero circulation. If a cycle cancelling step increases the flow along an arc $a \in A_1$ by θ, it is increased along any arc $a' \in A_2$ along the same cycle by $\frac{1}{2}\theta$. Obviously, if the residual capacity u_a of a determines the maximal flow which can be augmented along the current negative cost cycle and $u_a = 2k+1$, θ, i.e. the flow on arcs in A_1 is still integral and the flow on arcs in A_2 can become strictly halfintegral. Halfintegral flows on arcs in A_2 lead to halfintegral residual capacities, but those again lead to halfintegral flows θ on arcs in the same set and to integral flows and residual capacities 2θ on arcs of A_1.

□

The solution on G' can be transfered to the flow instance G and the same argumentation can be applied whenever the nodes of the network can be seperated into appropriate sets A_1 and A_2.

5 Modified SSP Algorithm

The CC algorithm is the first (and to our knowledge only) classical combinatorial minimum cost flow or circulation algorithm so far, which was extended to the generalized case by Wayne [12]. (Other approaches for solving generalized flow problems are of course provided by LP techniques and the modified network simplex method of Dantzig [3, 1].) We give a generalization of the *Successive Shortest Path* (SSP) algorithm [9, 7, 2], which is based on the principle of pseudoflows, i.e. flows respecting the non-negativity and capacity constraints, but not the node balance constraints.

The node balance constraints mean that the difference of incoming flow and outgoing flow of a vertex v is equal to a given balance value $b(v)$. If $b(v) > 0$,

v is a source, i.e. generates flow, $b(v) < 0$, v is a sink, i.e. consumes flow. Otherwise $b(v)$ is zero and the sum of incoming and outgoing flow at v has to be equal. The algorithm starts from a flow of zero units, which obviously is greater or equal to zero and less than the capacity of each arc, but violates the mass balance at sources and sinks and is thus no valid flow (primary solution), but a pseudoflow.

For convenience, we recapitulate functionality of the SSP algorithm: In general we assume a number of sources and sinks in a network, but require the sum of excesses (positive balance values at sources) and deficits (negative balances at sinks) to equal each other and to be connected by paths with sufficient total capacity. In due course of the algorithm the maximum number of flow units is augmented along a shortest path π_{st} from a (still) source node to a (still) sink node, i.e. $\min\{b(s), -b(t), \min_{e \in \pi_{st}}\{cap(e)\}\}$. After each augmentation a residual network is built in the following way: For each arc $e(u, v)$ with a positive flow $f(e)$, an arc $\bar{e} = (v, u)$ with capacity $cap(\bar{e}) = f(e)$, cost $c(\bar{e}) = -c(e)$ and flow 0 is added. If $f(e)$ equals $cap(e)$, then it is removed from the network.

While the shortest path π_{st} is determined, each vertex v is labeled with a distance $d_s(v)$ from s and we define a potential $p'(v) = p(v) - d_s(v)$ (initially $p(v) = 0$). (There are well-known variants which are generally applied to save running time, see [1], Chapter 9.) Then we define the reduced cost of an arc $e = (u, v)$ as $c_r(e) = c(e) - p(u) + p(v)$. The reduced cost of an arc measures its cost relatively to the shortest path distance of u and v. Thus repeating the shortest path calculation for s and t with reduced costs and the augmentation of flow leads to a minimum cost flow solution, as soon as all excesses and deficits are balanced.

Generally speaking, the backwards arcs in the residual network enable the algorithm to redirect flow on earlier shortest paths, if they do not contribute to an overall minimum cost solution. In particular, if we assume positive arc weights, a flow f of zero units and node potentials $p(v) = 0$, we start from an optimal dual solution, which is primarily infeasible, as f is only a pseudoflow. Augmenting flow along shortest reduced cost paths from sources to sinks stays dual optimal, because of the reduced cost optimality (see [1], Chapter 9):

The reduced costs stay nonnegative as the original costs satisfy $c(e) \geq 0$ and residual arcs only appear along shortest paths were the reduced costs are always zero. Thus we arrive at a solution which is both primal and dual feasible while staying optimal. If we further assume integral capacities and balances we also gain an integral optimal solution. So far, the running time of the algorithm is determined by the number of augmentations and the time consumed by distance calculations. The number of augmentations is limited by the sum of the supplies (or demands) as in each iteration there is at least one unit of flow balanced between a source and a sink (inductive argumentation) and no new imbalances are created.

So far, the algorithm is pseudopolynomial and needs additional scaling techniques to attain a polynomial time bound, although the use and benefit of scaling schemes depends on the character of the instances. The reduced cost optimality further ensures nonnegative arc weights throughout the application algorithm

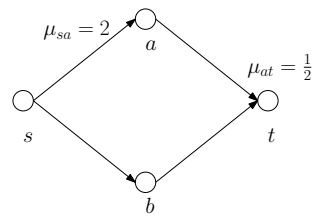

Figure 10: Two s-t-ways with identical arc cost, but different de facto per unit cost.

if we assume the original arc costs to be non-negative, $c(a) \geq 0 \forall a \in A$, such that we can use Dijkstra's algorithm [4] to determine shortest paths π_{st} and distances $d_s(v)$.

Now we describe how to adopt the SSP algorithm for the case of networks with arc costs, capacities and multipliers. First of all, we have to define a shortest path on a network with multipliers? (In the following, we will use the arc multiplier variant.) Consider Figure 10, where all arc costs and multipliers are 1 if not depicted otherwise. The two possible paths from s to t result in costs of 2 accounting only on arc costs. Yet actually sending one unit of flow along $s - a - t$ creates two units of flow at a. There are two possibilities for the second unit of flow: Firstly it could stay at a, secondly it can be passed on to t.

Both options cause difficulties. The first option could create an imbalance at a and the invariance of the SSP algorithm, that augmenting flow only decreases node imbalances, which results in a primarily valid flow solution and thus is essential for both correctness and termination of the algorithms can no longer be guaranteed. The second option of sending the spare flow unit at a further to t could also result in an imbalance (consider $\mu_{at} = 1$), if the deficit at t is not large enough. But this can be taken into account when the maximum flow along an $s - t-$path is determined.

Still, sending the multiplied flow units along to t increases the cost of the $s - t-$path $s - a - t$. The de facto cost of moving 1 unit of flow out of s along $s - a - t$ are 3 whereas due to arc multipliers of 1 along $s - b - t$ the cost of the second way are only 2. Thus we have to consider the multipliers as well as the arc costs in the calculation of a shortest path and we agree to adopt the second option and conserve the node balance constraints at inner nodes of a shortest path. We define:

Definition 2 *The cost of a flow multiplier path $\pi_{uv} = u_1 - \ldots - u_n$ with $u = u_1$ and $u_n = v$ from u to v is defined as:*

$$c'(\pi_{uv}) = \sum_{i=2}^{n} \prod_{j=1}^{i-1} \mu_{j(j+1)} \cdot c((i-1)i).$$

A shortest flow multiplier path π^*_{uv} in a network is such a path of minimal cost $c'(\pi_{uv})$.

We show that for this definition of path cost, similar to the canonical sum definition, the following Lemma holds:

Lemma 2 *Every subpath $\pi_{u_i u_j}, 1 \leq i < j \leq n$ of a shortest flow multiplier path $\pi^*_{uv} = u_1 - \ldots - u_n$ with $u = u_1$ and $u_n = v$ is a shortest flow multiplier path $\pi^*_{u_i u_j}$.*

Proof
Let $\pi_{u_i u_j} \subseteq \pi^*_{uv} = u_1 - \ldots - u_n, 1 \leq i < j \leq n$ be (shortest) flow multiplier paths, but $\pi^*_{u_i u_j}$ with $c'(\pi^*_{u_i u_j}) < c'(\pi_{u_i u_j})$. Then:

$$\begin{aligned} c'(\pi^*_{uv}) &= c'(u - \ldots - u_{i-1}) + \prod_{l=1}^{i-1} \mu_{l(l+1)} \cdot c'(\pi_{u_i u_j}) \\ &\quad + \prod_{l=1}^{j} \mu_{l(l+1)} \cdot c'(u_j - \ldots - v) \\ &> c'(u - \ldots - u_{i-1}) + \prod_{l=1}^{i-1} \mu_{l(l+1)} \cdot c'(\pi^*_{u_i u_j}) \\ &\quad + \prod_{l=1}^{j} \mu_{l(l+1)} \cdot c'(u_j - \ldots - v) \end{aligned}$$

This contradicts the assumption that π^*_{uv} was a shortest $u - v-$path.

□

Next we adjust Dijktstra's algorithm to the above defined path cost to find shortest flow multiplier paths from a source s to a sink t. Further we label all nodes v of the given network with distances $d_s(v)$, assuming again that $c(u, v) \geq 0, \forall (u, v) \in A$ and $c(u, v) = \infty, \forall (u, v) \notin A$ (see Algorithm 1). This can be done by introducing a second (third) tentative per node parameter m_v besides the tentative distance $d_s(v)$ (and predecessor), which accounts for the product of arc multipliers on the (tentative) shortest path from s to every node v.

We show the correctness of the modified algorithm by validating the following invariants of the while loop:

Lemma 3 *1. $\forall v \in F^* : d_s(v) = c'(\pi_{sv}), m_v = \prod_{(uw) \in \pi_{sv}} \mu_{uw}$*

2. $\forall v \in V \setminus F^* : d_s(v) = min_{x \in F^*}\{c'(\pi_{sx}) + m_x \cdot c(x, v)\}$

Proof
We prove by induction. After the initialization both invariants hold, as $d_s(s) = 0$ is surely $c'(\pi_{ss})$ and the empty product equals $m_s = 1$. The first pass through

Algorithmus 1 Flow Multiplier Dijkstra

```
F = {s}
F* = ∅
d_s(s) = 0,
m_s = 1
d_s(v) = ∞, m_v = 1, v ∈ V \ {s}
while F ≠ ∅ do
   d_s(u) = min_{v∈F}{d_s(v)}
   for all v ∈ V \ F* : (u, v) ∈ A do
      F = F ∪ {v}
      if d_s(v) > d_s(u) + m_u · c(u, v) then
         d_s(v) = d_s(u) + m_u · c(u, v)
         m_v = m_u · μ_uv
      end if
   end for
   F = F \ {u}
   F* = F* ∪ {u}
end while
```

the while loop updates all of s's neighbours' distances to $d_s(v) = m_s \cdot c(s, v)$ which validates the second invariance. For each additional pass through the while loop, one vertex u enters F^*. Let $\pi^*_{su} = s - \ldots - v_x - x - y - v_y - \ldots - u$ be a shortest flow multiplier path from s to u and x the last vertex on the path in F^* before the current pass through the while loop, such that $x \in F^*, y \in V \setminus F^*$. It holds:

$$\begin{aligned} c'(\pi^*_{su}) &= c'(\pi^*_{sx}) + m_x c(x, y) + m_y c'(u_j - \ldots - v) \\ {\scriptstyle (Inv.2)} &\geq d_s(y) + m_y c'(u_j - \ldots - v) \\ {\scriptstyle (d_s(u) min.)} &\geq d_s(u). \end{aligned}$$

Assuming that $c'(\pi^*_{su})$ is a shortest path gives equality and thus the first invariant is guaranteed after each while loop. The second invariant is also valid, because for every vertex $v, (u, v) \notin A$ the arc cost $c(u, v)$ is set to infinity, thus $min_{x \in F^*} c'(\pi_{sx}) + m_x \cdot c(x, v)$ cannot be decreased by $c'(\pi_{su}) + m_u \cdot c(u, v)$ and for all $v, (u, v) \in A$ a possible decrease is checked explicitly in the algorithm.

□

With Lemma 3 the correctness of the modified Dijkstra algorithm is shown as it terminates after at most n while loops (one vertex leaves F in every pass and none is added for more than one time) and all nodes v with $\pi_{sv} \subseteq A$ are in F^* (after termination), such that the invariant indicates that we know the shortest flow multiplier path distance and thus also the path π^*_{sv}.

Further, we want to adjust the SSP algorithm with the help of the new notion and computation of shortest paths, i.e. apply the Flow Multiplier Dijkstra

instead of the original Dijkstra and take the flow multipliers into account when we compute the maximum flow δ to be augmented. After each augmentation another residual network is built in the following way: For each arc $e(u,v)$ with a positive flow $f(e)$, an arc $\bar{e} = (v,u)$ with capacity $cap(\bar{e}) = f(e)$, cost $c(\bar{e}) = -c(e)$, arc multiplier $\mu_{\bar{e}} = \frac{1}{\mu_e}$ and flow 0 is added. If $f(e)$ equals $cap(e)$, then arc e is removed from the network.

Algorithmus 2 Flow Multiplier SSP

$\forall v \in V : p(v) = 0$
$\forall a = (u,v) \in A : f(a) = 0, c_r(a) = c(a), cap_r(a) = cap(a)$
$E = \{v, b(v) > 0\}, D = \{v, b(v) < 0\}$
while $E \neq \emptyset$ **do**
  $e \in E, d \in D$
  Determine shortest flow multiplier paths π^*_{ev} and $d_e(v)$ with respect to reduced costs c_r in the residual network $G(f)$
  Update $p(v) = p(v) - d_e(v)$ for each $v \in V$.
  $delta := min\{b(e), -b(d), min_{a=(uv)\in\pi_{ed}} \frac{cap_r(a)}{m_u}\}$
  Augment δ units of flow along π_{ed}
  $\forall a = (u,v) \in A : f(a), c_r(a) = c(a) - p(u) + p(v), cap_r(a) = cap(a) - f(a)$.
  Update $G(f)$.
  $E = \{v, b(v) > 0\}, D = \{v, b(v) < 0\}$
end while

To show the correctness of Algorithm 2, i.e. the optimality of the flow solution, we adopt two lemmata from [1] (Lemma 9.11, 9.12) for the flow multiplier version of the SSP Algorithm:

Lemma 4 *Suppose that a pseudoflow (or flow) $f(a) \forall a \in A$ satisfies the reduced cost optimality conditions with respect to some node potentials $p(v) \forall v \in V$, $c_r(a) = c_r(uv) = m_u \cdot c(uv) - p(u) + p(v)$ and let $d_s(v)$ be the cost of $\pi^* sv \forall v \in V$ in the residual network $G(f)$. Then it holds:*

1. *The pseudoflow f also satisfies the reduced cost optimality conditions with respect to the node potentials $p(v)' = p(v) - d_s(v)$.*
2. *The reduced costs $c'_r(a) = c'_r(uv) = m_u \cdot c(uv) - p'(u) + p'(v)$ are zero for all arcs $a \in \pi^* sv$.*

Proof

1. According to the assumption: $c'_r(a) \geq 0 \forall a \in A, G(f) = (V, A)$. Further shortest path optimality holds for shortest flow multiplier paths (see Lemma 2,3):

$$d_s(v) \leq d_s(u) + c_r(uv) \tag{1}$$

Substituting $c_r(uv)$ in 1 with the definition, we obtain:

$$\begin{aligned}
& d_s(v) \leq d_s(u) + m_u \cdot c(uv) - p(u) + p(v) && (2)\\
\Leftrightarrow\ & 0 \leq m_u \cdot c(uv) - (p(u) - d_s(u)) + (p(v) - d_s(v)) && (3)\\
\Leftrightarrow\ & 0 \leq m_u \cdot c(uv) - p(u)' + p(v)' && (4)\\
\Leftrightarrow\ & 0 \leq c'_r(uv) = c'_r(a) && (5)
\end{aligned}$$

2. For any arc $a = (uv)$ on a shortest flow multiplier path it holds:

$$\begin{aligned}
& d_s(v) = d_s(u) + c_r(uv) && (6)\\
\Leftrightarrow\ & d_s(v) = d_s(u) + m_u \cdot c(uv) - p(u) + p(v) && (7)\\
\Leftrightarrow\ & 0 = m_u \cdot c(uv) - (p(u) - d_s(u)) + (p(v) - d_s(v)) && (8)\\
\Leftrightarrow\ & 0 = m_u \cdot c(uv) - p(u)' + p(v)' && (9)\\
\Leftrightarrow\ & 0 = c'_r(uv) = c'_r(a) && (10)
\end{aligned}$$

□

Lemma 5 *Suppose that a pseudoflow (or flow) $f(a) \forall a \in A$ satisfies the reduced cost optimality conditions and we obtain $f'(a)$ from $f(a)$ by sending flow along a shortest flow multiplier path from node s to some node t; then $f'(a) \forall a \in A$ also satisfies the reduced cost optimality conditions.*

Proof
With potentials p and p' as well as reduced costs as defined in Lemma 4, the lemma implies, that f also satisfies the reduced cost optimality conditions for $c'_r(a)$. Further, for any arc $a \in \pi^* sv$, a reverse arc $\bar{a}$ is added to $G(f')$, but as $c'_r(a) = 0$ for such a, $c'_r(\bar{a}) = -c'_r(a) = 0 \geq 0$.

□

Lemmata 4 and 5 show that Algorithm 2 terminates with a minimum cost maximum flow solution through the network N: If we determine a maximum flow through N at first, we terminate with a valid flow (primal solution) while preserving the reduced cost optimality conditions, i.e. dual optimality and feasibility. By the strong duality theorem [5], the flow solution must be optimal.

Alternatively (as we do here), we just increase the flow through N as long as there is an excess e node and a deficit node d and a path π^*_{ed} between them left in the residual network. Otherwise, the algorithm terminates and we obtain the maximum possible flow with minimal cost nevertheless. Concerning the SSP for integral fows without multipliers, the running time of the (unscaled) version of the algorithm is pseudopolynomial in the sum of excesses and demands, as in each augmentation at least one unit of flow is sent. Unfortunately, δ does not

need to be integral in general any longer and we are not able to give a lower bound $\epsilon(n) < \delta(n)$, i.e. an implicit upper bound to the number of augmentations yet.

Still we can use the MSSP on the instances with guaranteed halfintegral optimal solutions (section 4). Here $\delta \geq \frac{1}{2}$, which results in a pseudopolynomial running time as for the case without flow multipliers.

6 Rounding to acceptable integer solutions

Recall the disposition network $N = (A \uplus B, E)$ in Figure 9, where the capacities on all arcs $(s, a_i), a_i \in A$ $((b_j, t), b_j \in B)$ encode excesses (demands) and the capacity on arcs $t_i = (a_i, b_j)$ is unlimited, but only the latter have costs greater than zero. All arcs t_i leaving an (implicit) excess node a_i have flow multipliers either 1 or 2.

But arriving arcs at an (implicit) demand node b_i can have different multipliers. In our application we can assume that for all b_i with incoming arc multipliers only 2, the demand is even. Otherwise a valid integral flow could not exist. Further, for b_i with mixed incoming arc multipliers, we also assume even demand, as for an odd demand any valid integer solution must contain an incoming flow on an arc with multiplier 1. Thus we can add a node b_i' with equivalent properties and demand 1 which has only incoming arcs with multiplier 1. With the above assumptions obviously only the set B_{mix} of nodes b_i with mixed incoming arc multipliers can cause halfintegral flows in the optimal solution. Let $d_{mix} = \sum_{b_i \in B_{mix}} b(b_i)$ be the total demand of all demand nodes in B_{mix} and let $e_2 = \sum_{a_i \in A_2} b(a_i)$ with $A_2 \subseteq A$ the set of all excess nodes a_i with outgoing multipliers 2. Still we can not guarantee the existence of a valid integer solution (see Figure 11).

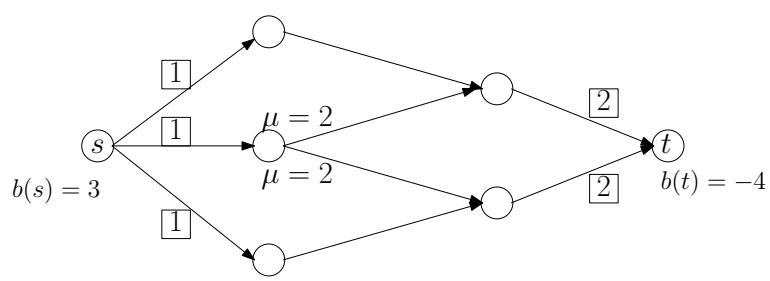

Figure 11: Disposition Network with no feasible integer solution (numbers in squares = arc capacities).

For simplification of some of the following statements, we assume:

Property 1 *There is always an uncapacitated way from s to t.*

To obtain an acceptable integral solution from a halfintegral optimal solution in a disposition network, we first apply the following naive heuristic, which temporarily allows $cap(b_j, t)$ to be violated by 1:

```
            Rounding Heuristic (RH)
1: while    (∃ halfintegral flow f)
3:          Find cheapest f from s to t
4:          if (f = f + 1/2 violates bal(t) by at most 1)
5:          Round f up, most expensive s-t-flow f' down.
6:          else Round f down, cheapest s-t-flow f' up.
8: end
```

The heuristic can always be applied to a halfintegral solution until there are only integral flows, because in each iteration, at least two halfintegral flows are rounded (up or down) to integral flows. The node balances can be violated: Demands can stay over-saturated by at most 1 in the end. But without further halfintegral flows, there must be an incoming flow on an arc with multiplier 1. We can augment 1 unit of flow back to the according excess node a_i in each such case.

Thus there can be additional unsaturated deficits (from rounding down at the 'excess side') and rest excesses (from rounding down at the 'demand side' or augmenting flow back in the last step). If we accept the solution nevertheless, we gain a 2-approximate 'solution'. A better option is to reallocate deficits and excesses by another MSSP run on the network reduced to the remaining imbalances and apply the rounding heuristic again. Iterating those steps terminates with an integral solution as we reduce the absolute demand in each step. Further:

Lemma 6 *After $O(\log(min\{e_2, d_{mix}\}))$ iterations of MSSP and RH we obtain a valid integral solution.*

Proof

The number of necessary iterations is determined by the number f_{half} of halfintegral flows created in each MSSP application. The value of f_{half} is limited by $min\{e_2, d_{mix}, |A_2 \times B_{mix}|\}$ in each iteration, because there can be one halfintegral flow from each $a_i \in A_2$ to each $b_i \in B_{mix}$ and the flow values out of A_2 can sum up at most to e_2 and into B_{mix} at most to d_{mix}. As for every flow which is rounded down and can thus cause an unsaturated demand of -1, another flow has to be rounded up, which saturates a demand with 1, we have saturated at least $\frac{d_{mix}}{2}$ demand after each iteration.

□

This also limits the possible rest demand after just one RH application:

Corollary 1 *The unsaturated rest demand d'_{mix} after one iteration of MSSP and RH is limited by:*

$$d'_{mix} \leq \frac{min\{e_2, d_{mix}, |A_2 \times B_{mix}|\}}{2}.$$

Unfortunately, although each application of HR produces an integral solution with cost no more than twice the cost of the proceeding halfintegral MSSP solution, the following simple example shows that the whole iterative procedure may increase the cost of the solution arbitrarily with regard to an initial halfintegral solution like in Figure 12:

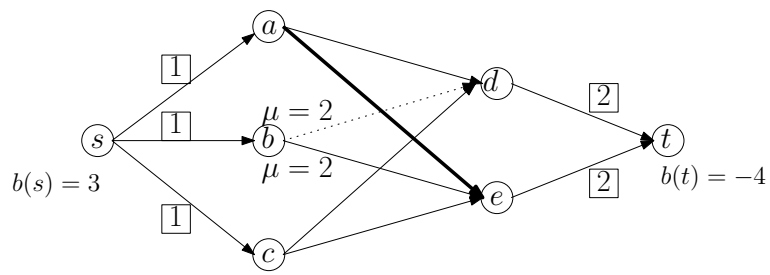

Figure 12: RH can find bad solutions.

Imagine the dotted arc has less costs $c - \delta$ and the bold arc has much bigger costs $c + \Delta$ than all normally drawn arcs. The optimal (halfintegral solution) sends one unit along $s-a-d-t$, one along $s-c-e-t$ and $\frac{1}{2}$ along $s-b-d-t$ and $s-b-e-t$ each. The costs are $OPT = 10c - \frac{1}{2}\delta$. Then RH rounds the cheapest halfintegral flow (on $s - b - d - t$) up and the more expensieve one through b via e down which gives us an integral 'solution' with cost $7c - \delta \leq 2OPT$.

Now the capacity of $d - t$ is violated by 1 and we augment the one incoming unit at d on an arc with multiplier 1 back via a to s where it constitutes a rest excess. Further, we have an unsaturated demand at t (implicitly at e). If we now apply another iteration of MSSP on the reduced network (see Figure 13) it can only augment one unit of flow along $s - a - e - t$ with arbitrary large cost δ and resultng in a solution with cost $10c - \delta + \Delta$ instead of the optimal integer solution with cost $10c$.

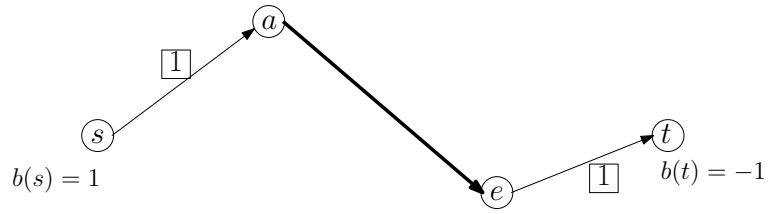

Figure 13: Small reduced network after one application of RH.

References

[1] R.K. Ahuja, T.L. Magnati, J.B. Orlin, "Network Flows - Theory, Algorithms and Applications." Prentice Hall, 1993.

[2] R.G. Busaker, P.J. Gowen "A procedure for determining minimal-cost network flow patterns." ORO Technical Report 15, Operational Research Office, John Hopkins University, Baltimore, MD, 1961.

[3] G.B. Dantzig "Linear Programming and Extensions" Princeton University Press, Princeton NJ, 1963.

[4] E. Dijkstra "A note on two problems in connexion with graphs" Numeriche Mathemacis 1 pp. 269-271, 1959.

[5] D. Gale, H.W. Kuhn, A.W. Tucker "Linear programming and the theory of games" Activity Analysis in Production and Allocation, T.C. Koopmans, ed. New York: John Wiley ans Sons pp. 317-329, 1951.

[6] M. Garey and D. Johnson "Computers and Intractability: A guide to the theory of NP-Completeness" W.H. Freeman, New York, 1979.

[7] M. Iri "A new method of solving transportation-network problems" Journal of the Operations Research Society of Japan 3, 27-87, 1960.

[8] W.S. Jewell "Optimal Flow though networks" Interim Technical Report 8, Operations Research Center, MIT, Cambridge, MA, 1958.

[9] W.S. Jewell "Optimal Flow though networks with gains" Operations Research 10, 476-499, 1962.

[10] S. Sahni "Computationally Related Problems" SIAM Jr. on Computing, 3, 4, 262-279, 1974.

[11] Craig A. Tovey "A Simplified NP-Complete Satisfiability Problem" Discrete Applied Mathematics 8, 85-89, 1984.

[12] Kevin D. Wayne "A polynomial combinatorial algorithm for generalized minimum cost flow" Mathematics of Operations Research, Vol.27, No.3, 445-459, 2002.