Generalized Max Flow in Series-Parallel Graphs

Katharina Beygang* Sven O. Krumke* Christiane Zeck*

Abstract

In the generalized max flow problem, the aim is to find a maximum flow in a generalized network, i.e., a network with multipliers on the arcs that specify which portion of the flow entering an arc at its tail node reaches its head node. We consider this problem for the class of series-parallel graphs. First, we study the continuous case of the problem and prove that it can be solved using a greedy approach. Based on this result, we present a combinatorial algorithm that runs in $\mathcal{O}(m^2)$ time and a dynamic programming algorithm with running time $\mathcal{O}(m \log m)$ that only computes the maximum flow value but not the flow itself. For the integral version of the problem, which is known to be $\mathcal{NP}$-complete, we present a pseudo-polynomial algorithm.

Keywords: generalized flow, max flow problem, series-parallel graphs, integral flow

1 Introduction

Given a directed graph $G = (V, A)$ with n nodes and m arcs, source $s = v_1$ and sink $t = v_n$, capacity $c_{ij} \geq 0$ and multiplier $\gamma_{ij} > 0$ for each arc a_{ij} going from v_i to v_j, the generalized max flow problem consists in finding a feasible $s-t$ flow f^* such that the amount of flow reaching the sink t is maximized. More formally, the aim is to find a mapping $f^* : A \to \mathbb{R}$ which fulfills

- $0 \leq f^*(a_{ij}) \leq c_{ij}$ for all arcs a_{ij},
- $\sum_{j:a_{ji} \in A} \gamma_{ji} f^*(a_{ji}) - \sum_{j:a_{ij} \in A} f^*(a_{ij}) = 0$ for all $i \notin \{1, n\}$,
- $\sum_{i:a_{in} \in A} \gamma_{in} f^*(a_{in})$ maximized.

In a generalized network, the multiplier of a path is defined as the product of all multipliers of arcs along the path. By definition, if an arc has multiplier γ, the multiplier of the corresponding backward arc in the residual network is $1/\gamma$.

*Department of Mathematics, University of Kaiserslautern, Paul-Ehrlich-Str. 14, 67663 Kaiserslautern, Germany. {beygang,krumke,zeck}@mathematik.uni-kl.de

Our research was motivated by a real world disposition problem of empty freight cars on the German railroad network. In our model, we use arc multipliers to take into account situations in which substitutions of cars are allowed, for instance, if there are requests that can either be fulfilled by one big car or by two small cars. Additionally, we have to require integrality of the solution, i.e., integrality of the flow entering and leaving each arc. A more detailed description of the model can be found in [3]. Engels et al. [5] present heuristics for the above-mentioned cargo problem.

Since, in the continuous case, the generalized max flow problem can be formulated as a linear program, it can be solved in polynomial time using interior point methods [6, 7]. Tardos and Wayne [11] present combinatorial algorithms that solve the problem in $\tilde{\mathcal{O}}(m^2n^3 \log^2 B)$ and $\tilde{\mathcal{O}}(m^2(m + n \log\log B) \log B)$ time. Here, the capacities are assumed to be integers between 1 and B, the multipliers are given as ratios of integers between 1 and B and $\tilde{\mathcal{O}}(g(n))$ denotes $\mathcal{O}(g \log^k m)$ for some k. If, in addition, the flow f^* is required to be integral, the generalized max flow problem becomes weakly $\mathcal{NP}$-complete, as shown by Sahni [10].

We study the problem on the class of series-parallel graphs. For the continous case, we give an algorithm that runs in $\mathcal{O}(m^2)$ time, and for the integral case, we present an algorithm with running time $\mathcal{O}((m^5C^4\Gamma^2)$ where C is the maximum capacity of an arc rounded up and Γ is the maximum multiplier of an arc rounded up.

A two-terminal series-parallel graph (sp graph) G is defined as a graph produced by a sequence of the following operations:

- create a new graph G consisting of the terminal nodes s and t and one arc from s to t,
- (series composition) given two two-terminal sp graphs G_1 and G_2 with terminals s_1, t_1 and s_2, t_2, respectively, identify t_1 with s_2 to obtain G which has terminals s_1 and t_2,
- (parallel composition) given two two-terminal sp graphs G_1 and G_2 with terminals s_1, t_1 and s_2, t_2, respectively, identify s_1 with s_2 and identify t_1 with t_2 to obtain G.

It is well known [12], that for a given sp graph, a decomposition tree has linear size and can be computed in linear time. Note that in an sp graph $n = \mathcal{O}(m)$.

Greedy approaches similar to ours have been successfully applied to minimum cost flow problems in sp graphs by Bein et al. [2] and by Booth et al. [4]. Bein et al. [1] show that the greedy method can be applied to network flow problems whose linear programming representation arises from a series parallel composition of linear programs. It might be conceivable that our problem falls within this framework, but the following example shows that this is not true. Consider two linear programs G_1 and G_2 which describe the generalized maximum flow problem for one arc with multiplier and capacity γ_1 and c_1, and γ_2 and c_2, respectively.

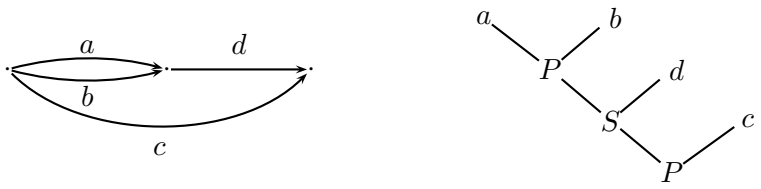


Figure 1: An sp graph (left) with corresponding decomposition tree on its right. S denotes a series composition and P a parallel composition.

$$\begin{array}{ll} \max & \gamma_1 x_1 \\ \text{s.t.} & x_1 \leq c_1 \\ & x_1 \geq 0, \end{array} \qquad \begin{array}{ll} \max & \gamma_2 x_2 \\ \text{s.t.} & x_2 \leq c_2 \\ & x_2 \geq 0. \end{array}$$

Applying the series composition as described in [1] to G_1 and G_2, we get the following linear program:

$$\begin{array}{ll} \max & \gamma_1 \gamma_2 x_{12} \\ \text{s.t.} & x_{12} \leq c_1 \\ & x_{12} \leq c_2 \\ & x_{12} \geq 0, \end{array}$$

Obviously the constraints do not fit into the correct programming description of a maximum flow problem in a graph with two serial edges which looks as follows

$$\begin{array}{ll} \max & \gamma_1 \gamma_2 x_{12} \\ \text{s.t.} & x_{12} \leq c_1 \\ & \gamma_1 x_{12} \leq c_2 \\ & x_{12} \geq 0 \end{array}$$

where x_{12} stands for the flow leaving the source.

2 Continuous Generalized Max Flow for Series-Parallel Graphs

In this section, we consider the continuous case of the generalized max flow problem in sp graphs, i.e., the flow is not required to be integral. First, we will show that this problem can be solved by a greedy strategy which leads to a combinatorial algorithm with running time $\mathcal{O}(m^2)$. Then, we present a dynamic programming algorithm which is also based on the greedy scheme. It only computes the maximum flow value but not the flow itself and has a running time of $\mathcal{O}(m \log m)$.

2.1 Combinatorial Algorithm

First, we will show that, for an sp graph G with terminals s and t, the amount of flow arriving at t subject to the constraint that x units

of flow leave s can be maximized by applying the following greedy strategy:

We choose a directed path in G from s to t with highest multiplier and, respecting the capacity constraints, send as much flow as possible along this path. Once an arc is satisfied, we iterate and choose the next best among the remaining paths in G and send flow along it from s to t. We continue until x units of flow have been sent or there do not exist any more residual paths from s to t.

Note that, even though we work in the residual network, we do not involve any backward arcs but only take into account residual paths consisting merely of forward arcs.

Radzik [9] used this greedy strategy in order to get approximate solutions for the generalized max flow problem. We will show that, for the case of sp graphs, it actually yields an optimal solution.

Definition 2.1. A flow generating cycle is a cycle in the residual network with multiplier strictly greater than 1. A generalized augmenting path (GAP) consists of a flow generating cycle connected to a path that leads to the sink t.

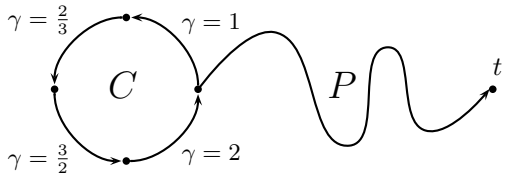


Figure 2: Example of a GAP consisting of a flow generating cycle C with multiplier 2 and a path P to t.

The following optimality condition is derived from a result by Onaga [8] who considers the generalized max flow problem for arbitrary networks with the difference that the flow is fixed at the sink and not at the source.

Theorem 2.1. *Let f_x be a generalized $s-t$ flow in G such that x units of flow leave the source s. Then the amount of flow reaching the sink t is maximized if and only if there is no GAP in the corresponding residual network $G(f_x)$.*

It also follows from [8] (see also Tardos et al. [11]) that for a generalized $s-t$ flow in G, the following holds.

Theorem 2.2. *Let f be a generalized $s-t$ flow in G. Then it is optimal if there is no augmenting path and no GAP in the corresponding residual network $G(f)$.*

Our aim is to show that, in the case of an sp graph, the above optimality condition remains fulfilled throughout the application of the greedy strategy. The following result is similar to Onaga [8, Theorem 3] except that our greedy strategy only consider paths consisting of forward arcs.

Theorem 2.3. *Let G be an sp graph with terminals s and t. For a given amount of flow x leaving the source s, let f_x be obtained by the above greedy strategy. Then the residual graph $G(f_x)$ does not contain any GAPs.*

Proof. By induction on the decomposition of G, we will show that, for any x and f_x obtained by the greedy strategy, the residual network $G(f_x)$ does not contain a cycle with multiplier greater than 1.

First, consider the smallest possible graphs: graphs G containing only one arc a with multiplier γ. The only possible cycle in the residual graph uses arc a and its backward arc and hence has multiplier $\gamma \cdot (1/\gamma) = 1$.

Now let G be a parallel composition of the sp graphs G_1 and G_2 and assume that the claim holds for G_1 and G_2. Given x, let f_x be a flow in G obtained by the greedy strategy. Restricting it to G_1 and G_2, respectively, yields the corresponding flows f_{1,x_1} and f_{2,x_2} with $x = x_1 + x_2$. Note that, since each path chosen by the algorithm is either completely contained in G_1 or G_2, the flows f_{1,x_1} and f_{2,x_2} can be seen as results of the greedy strategy themselves. Therefore, by our assumption, $G_1(f_{1,x_1})$ and $G_2(f_{2,x_2})$ do not contain any flow generating cycles.

Suppose that $G(f_x)$ which is composed of $G_1(f_{1,x_1})$ and $G_2(f_{2,x_2})$, contains a flow generating cycle C. Then, since it cannot be entirely contained in one of the subgraphs $G_1(f_{1,x_1})$ or $G_2(f_{2,x_2})$, C must contain s or t.

If a cycle C contains only one of the terminals, it must be entirely contained in one of the subgraphs since otherwise, it is not simple. Hence, its multiplier is at most 1.

If both terminals are contained in C, the cycle consists of a path P_1 from s to t and a path P_2 from t to s, w.l.o.g. entirely contained in $G_1(f_{1,x_1})$ and $G_2(f_{2,x_2})$, respectively. We can assume that P_1 consists only of forward arcs and P_2 consists only of backward arcs. To see this, assume that there occurs a backward arc in P_1, i.e., the path only uses forward arcs until it reaches the node $t' \neq t$ and then uses a backward arc for the first time. Note that, in the course of the sp composition of G_1, t' occurs as sink node but loses this property at some point due to a series composition since $t' \neq t$. Let G' be the sp subgraph with terminals s' and t' right before the series composition that causes t' to lose the property of being end terminal. Then, any path in G_1 from s to t' has to pass s'. Also, in order to go from an inner node of G' to t, it is necessary to pass s' or t'. So, after using the backward arc starting at t', we must pass s' or t' on our way to t, which means P_1 must contain a cycle C_1. This cycle has a multiplier of at most 1 since it is completely contained in the subgraph $G_1(f_{1,x_1})$. Hence, if we remove C_1 from C, the multiplier of C does not become smaller. By a similar argument, we can assume that P_2 consists only of backward arcs.

So $G(f_x)$ contains paths P_1 and P_2 with multipliers Γ_1 and Γ_2, respectively, such that $\Gamma_1 \cdot \Gamma_2 > 1$. Note that the path P_2' consisting only of forward arcs that is obtained by reversing P_2 has multiplier

$1/\Gamma_2$. The residual path P_2 was generated when the last missing arc a of P_2' was used to carry flow, i.e., when a was part of a residual path P in G_2 going from s to t, using only forward arcs and with the highest multiplier Γ at that time, as depicted in Figure 3.

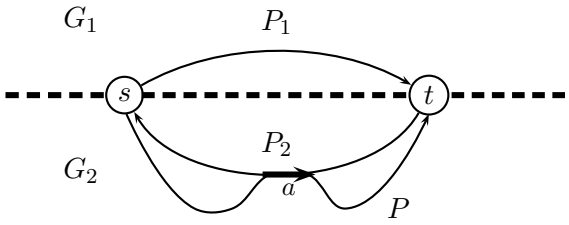


Figure 3: When the greedy algorithm sends flow over the residual path P, it uses the last missing (bold) arc from P_2' and the path P_2 develops in the residual network.

The path multiplier Γ satisfies $\Gamma_2 \cdot \Gamma \leq 1$. (After sending an infinitesimal amount of flow along P, both P and P_2 have a positive residual capacity. If we had $\Gamma_2 \cdot \Gamma > 1$, then we would have a cycle in the residual network entirely contained in G_2 with multiplier greater than 1, which is not possible.) So this means that flow was sent along path P with multiplier $\Gamma \leq 1/\Gamma_2 < \Gamma_1$ even though P_1 also had a positive residual capacity and a higher multiplier. This is a contradiction to the way our greedy strategy works.

Now let G be a series composition of G_1 and G_2 and assume that the claim holds for G_1 and G_2. Given an amount of flow x leaving the source s, the greedy strategy is applied to find f_x. Restricting this flow to G_1 and G_2, respectively, yields the generalized flows f_{1,x_1} and f_{2,x_2}. In each step of the algorithm, a path P in the residual network with highest multiplier and consisting only of forward arcs is chosen. Note that P decomposes into two paths P_1 and P_2 contained in G_1 and G_2. Now P_1 must be a path from s_1 to t_1 with highest multiplier among those paths consisting only of forward arcs, and the analog holds for P_2. So, as in the previous case, f_{1,x_1} and f_{2,x_2} can be seen as the results of a greedy strategy themselves and therefore $G_1(f_{1,x_1})$ and $G_2(f_{2,x_2})$ do not contain flow generating cycles.

Suppose that $G(f_x)$ contains a flow generating cycle. Then, as before, it cannot be entirely contained in $G_1(f_{1,x_1})$ or $G_2(f_{2,x_2})$, and, hence, it must pass $t_1 = s_2$. This means that it decomposes into at most two cycles such that each of them is entirely contained in $G_1(f_{1,x_1})$ or $G_2(f_{2,x_2})$. By assumption, those cycles have a multiplier of at most 1 and, therefore, the entire cycle must have a multiplier of at most 1, which yields a contradiction. □

The above theorem proves the optimality of our greedy strategy. So we can compute a generalized max flow in an sp graph by successively sending flow along paths in the residual network consisting only of forward arcs with maximum multiplier.

In each step, we need to find a residual path from s to t with highest multiplier. It is well known that by substituting all arc multipliers γ by $-\log(\gamma)$, we can convert the problem into a regular shortest path problem with sum objective function. Since an sp graph is directed and acyclic, this problem can be solved in $\mathcal{O}(m)$ time if the nodes are processed in a topological order. Sending flow along a path takes $\mathcal{O}(m)$ time. Since, in each step, at least one arc is satisfied, we have at most m steps. Hence, we can compute a generalized max flow in $\mathcal{O}(m^2)$ time.

2.2 Dynamic Programming Algorithm

Next, for an sp graph G with terminals s and t, we will use dynamic programming on its decomposition tree to compute a function $h : x \mapsto h(x)$ which maps an amount of flow x leaving the source s to the maximum amount of flow $h(x)$ that can arrive at the terminal t given that x units leave the source s.

Since the function h can also be seen as the result of our greedy strategy, which successively sends flow along residual paths with highest multiplier, it is piecewise linear, continuous and concave:

If x units of flow are sent from s to t along a specific path P_1 with multiplier Γ_1, $\Gamma_1 \cdot x$ units reach t. When the capacity of this path is used up, we choose the next residual path P_2 with maximum multiplier $\Gamma_2 \leq \Gamma_1$ and send flow along it at a rate of Γ_2 and so on.

We will denote a function h by a corresponding set of pairs

$$H = \{(\Gamma_1, C_1), (\Gamma_2, C_2), \ldots, (\Gamma_l, C_l)\}$$

with $\Gamma_1 > \ldots > \Gamma_l$. Each pair (Γ_i, C_i) represents a linear part of the function, that is, for each i, the slope of the function on the interval $[\sum_{j=1}^{i-1} C_j, \sum_{j=1}^{i} C_j]$ is Γ_i. We will show that the number of breakpoints is $\mathcal{O}(m)$. In terms of the corresponding network flow, this means that the first C_1 units can be sent over paths with path multiplier Γ_1, the next C_2 units can be sent along paths with multiplier Γ_2 and so on.

We start by considering the smallest possible graphs: graphs G that contain only one arc a with multiplier γ and capacity c. Then, clearly $H = \{(\gamma, c)\}$, and $f_x(a) = x$.

Now, let G be composed of two graphs G_1 and G_2 such that

$$H_1 = \{(\Gamma_{1,1}, C_{1,1}), \ldots, (\Gamma_{1,l_1}, C_{1,l_1})\},$$
$$H_2 = \{(\Gamma_{2,1}, C_{2,1}), \ldots, (\Gamma_{2,l_2}, C_{2,l_2})\}.$$

Consider the case that G is the result of the parallel composition of G_1 and G_2. Then we obtain H as follows: We merge the sets H_1 and H_2 such that the Γ values of the pairs remain in descending order. If there are two pairs with the same Γ value, i.e., there exist two pairs $(\Gamma_{1,i}, C_{1,i})$ and $(\Gamma_{2,j}, C_{2,j})$ with $\Gamma_{1,i} = \Gamma_{2,j}$, then we replace them by only one pair with this Γ value and the sum of the corresponding C

values as C value, i.e., by $(\Gamma_{1,i}, C_{1,i} + C_{2,j})$. Hence, the number of line segments of H is at most the sum of the numbers of line segments of H_1 and H_2.

If G arises from a series composition, we need to consider the products of the multipliers $\Gamma_{1,i}$ and $\Gamma_{2,j}$ in descending order and distribute the corresponding capacities as follows: We start by considering the line segments with the highest multipliers, $(\Gamma_{1,1}, C_{1,1})$ and $(\Gamma_{2,1}, C_{2,1})$, which, in the series composition, correspond to paths with multiplier $\Gamma_{1,1} \cdot \Gamma_{2,1}$. We want to determine the maximal amount of flow x that can be sent from s along these paths. It is bounded by $C_{1,1}$ and by $C_{2,1}/\Gamma_{1,1}$ since $\Gamma_{1,1} \cdot x$ units arrive at $s_1 = t_2$ given that x units are sent over a path with multiplier $\Gamma_{1,1}$. Hence, we can send $\min\{C_{1,1}, C_{2,1}/\Gamma_{1,1}\}$ units of flow and, therefore, we set $C_1 := \min\{C_{1,1}, C_{2,1}/\Gamma_{1,1}\}$. As we iterate and consider the next highest product of Γ values, we need to make sure to take into account how much capacity has already been used. Therefore, for each capacity $C_{1,i}$ and $C_{2,j}$ occuring in H_1 or H_2, we maintain a variable $\Delta_{1,i}$ or $\Delta_{2,j}$, respectively, which gives the amount of capacity that has already been used up. So all Δ values are initially 0. If the max flow that can be sent from s_1 over a residual path with multiplier $\Gamma_{1,i} \cdot \Gamma_{2,j}$ to t_1 is x, we increase $\Delta_{1,i}$ by x and $\Delta_{2,j}$ by $\Gamma_i \cdot x$. Thus, in order to determine the amount of flow that can be sent over a path with multiplier $\Gamma_k = \Gamma_{1,i} \cdot \Gamma_{2,j}$, we need to determine $C_k = \min\{C_{1,i} - \Delta_{1,i}, (C_{2,j} - \Delta_{2,j})/\Gamma_{1,i}\}$.

If the minimum is attained for $C_{1,i} - \Delta_{1,i}$, this implies that the paths represented by the line segment $(\Gamma_{1,i}, C_{1,i})$ are saturated. In order to get the next highest multiplier, we need to consider the segment $(\Gamma_{1,i+1}, C_{1,i+1})$ of H_1. If the minimum is attained for $(C_{2,j} - \Delta_{2,j})/\Gamma_{1,i}$, the next line segment $(\Gamma_{2,j+1}, C_{2,j+1})$ of H_2 is considered. In any case, in each iteration, at least one of the line segments of H_1 or H_2 is finished in the sense that all paths represented by it are saturated and we move on to the next line segment with the highest multiplier. So the number of line segments of the new function H belonging to G is at most the sum of the numbers of line segments of H_1 and H_2.

Note that h has at most m line segments if G has m arcs: For graphs with only one arc, this is obvious. If G is composed of two sp graphs G_1 and G_2, it follows by the above procedure for computing the corresponding functions h_1 and h_2.

Now, we will analyze the running time of our algorithm. Computing the decomposition tree takes $\mathcal{O}(m)$ time and its size is $\mathcal{O}(m)$, which implies that $\mathcal{O}(m)$ composition steps are considered during the algorithm. Using a straightforward implementation, we need $\mathcal{O}(m)$ time to compute h for each composition, so the total running time is $\mathcal{O}(m^2)$.

The running time can be reduced using the same data structure as Booth and Tarjan [4] for the minimum cost maximum flow problem in series parallel graphs. Their algorithm also uses the decomposition tree and for each subgraph computes a so called flow list which consists

of pairs of cost and capacity, and hence, is very similar to our function H.

In the case of a parallel composition of two graphs, the two flow lists are merged and sorted by cost. This corresponds exactly to our algorithm.

In the case of a series composition, they consider the sums of costs occuring in different lists in nondecreasing order and compute the maximum capacity that can be sent over the corresponding paths. This is very similar to our approach which considers the products of multipliers in nondecreasing order. By taking the logarithm, we can in fact adjust the situation such that it matches the situation in [4].

Booth and Tarjan [4, Theorem 1] prove that using finger search trees, the flow lists can be computed fast.

Theorem 2.4. *Computing the flow list of an m-edge sp graph requires $\mathcal{O}(m \log m)$ time.*

Hence, with an implementation using finger search trees, our dynamic programming algorithm runs in $\mathcal{O}(m \log m)$ time.

3 Integral Generalized Max Flow for Series-Parallel Graphs

In this section, we consider the generalized max flow problem with the additional requirement that the flow must be integral. First we need to define what integrality of a generalized flow means. Unlike the ordinary max flow problems without multipliers, there are several possibilities to define an integral flow here. Some alternatives (from less to more restrictive) are the following:

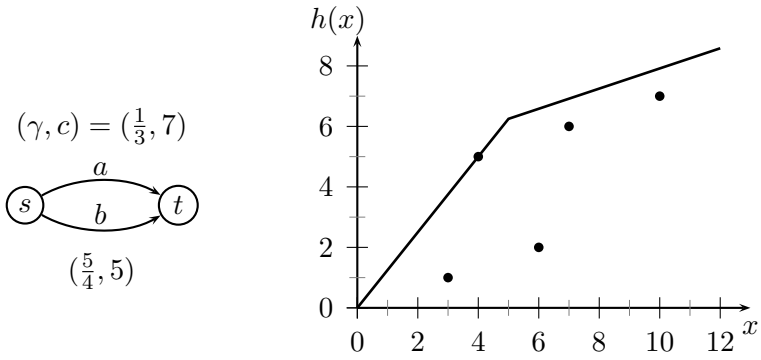
1. an integral amount of flow enters each arc,
2. an integral amount of flow enters each arc and each node,
3. an integral amount of flow enters and leaves each arc.

From now on, we will use Version 3. of this definition, i.e., we require the amount of flow entering and leaving each arc to be integral.

Note that, with this integrality condition, the function h as defined in the previous section is no more concave or even monotone. The following example contrasts the functions h for the continuous and integral case for a specific sp graph.

For the continuous case, we get a piecewise linear, concave function. If integrality of the flow is required, the outcome is different. It is not possible to send 1 or 2 units of flow while fulfilling Version 3. But we can send 3 units along arc a such that 1 flow unit reaches t. We can send 4 units along b, which yields 5 units at t. It is not possible to send 5 units without violating 3. We can send 6 units along a. If we need to send 7 units, we can send 3 along a and 4 along b, which yields a flow value of 6 at t, and so on.

Unlike in the continuous case, an integral generalized flow f is not maximal if and only if there are no GAPs or augmenting paths with



(a) graph (b) plot

Figure 4: Example of a graph and a plot of the corresponding function h for the continuous and integral case.

integral capacity in the residual graph $G(f)$ as the example in Figure 5 shows:

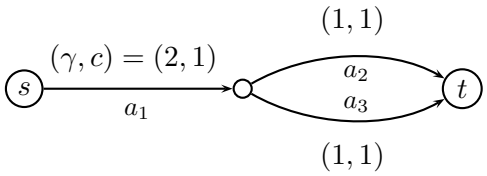


Figure 5: Example of an sp graph which shows that for the integral case an optimality condition similar to 2.2 does not apply. Considering the flow $f \equiv 0$ in, the residual network $G(f) = G$ has no flow generating cycles, and, hence, no GAPs, and it has no path from s to t along which we could send integral flow. However, f is not optimal since f' with $f'(a_1) = f'(a_2) = f'(a_3) = 1$ is a generalized flow that yields a higher flow value arriving at t.

Sahni [10] showed by a reduction from Subset Sum that the max integral generalized flow problem is $\mathcal{NP}$-complete for Version 1. Since the graph used in the reduction is series-parallel, the result also holds for the restriction to instances with sp graphs. Moreover, since all multipliers are integral, an integral amount of flow entering an arc implies an integral amount of flow leaving an arc, and, hence, the proof also applies to Version 3. of the problem.

Next we will show how we can solve the problem in pseudo-polynomial time using a similar framework as in the continuous case. As before, for a given sp graph G, we use its decomposition tree. For

each graph G' occuring in the decomposition, we will compute a table T' such that entry $T'(a,b) = 1$ if there exists an integral flow from s' to t' such that a units leave s' and b units of flow arrive at t', and $T'(a,b) = 0$ otherwise. For each graph G' we compute such a table with $a = 1, \dots, m \cdot C$ and $b = 1, \dots, m \cdot C \cdot \Gamma$, where m denotes the number of arcs of G, C is the maximum capacity of an arc rounded up, and Γ is the maximum multiplier of an arc rounded up. Note that $m \cdot C$ is an upper bound on the flow that can be sent from any source s', and $m \cdot C \cdot \Gamma$ is an upper bound on the flow that can reach a sink t'. Thus, each table has $m^2C^2\Gamma$ entries.

Consider a graph G' with only one arc a with capacity c and multiplier γ. Note that, if γ is irrational, there is no integral number l with $0 \leq l \leq c$ such that $l \cdot \gamma$ is integral. In this case, $T'(a,b) = 0$ for all (a,b). Otherwise, let $\gamma = p/q$ with integers p and q such that their greatest common divisor is $\gcd(p,q) = 1$. Then, the flow arriving at t' is integral given that an integral amount of flow l is sent from s' if and only if $l \in q \cdot \mathbb{Z}$. So we have $T'(r \cdot q, r \cdot p) = 1$ for all integers r with $0 \leq r \cdot q \leq c$, and $T'(a,b) = 0$ else.

Consider a graph G' that is composed of G_1 and G_2 and assume we know the tables T_1 and T_2.

If G' is the parallel composition of G_1 and G_2, $T'(a,b) = 1$ if and only if there exist entries $T_1(a_1,b_1) = 1$ and $T_2(a_2,b_2) = 1$ with $a = a_1 + a_2$ and $b = b_1 + b_2$. So we need $\mathcal{O}(m^2 \cdot C^2 \cdot \Gamma)$ time to compute one entry, which, in total, amounts to $\mathcal{O}(m^4 \cdot C^4 \cdot \Gamma^2)$ time for the whole table T'.

If G' is the series composition of G_1 and G_2, we have $T'(a,b) = 1$ if and only if there exist entries $T_1(a_1,b_1) = 1$ and $T_2(a_2,b_2) = 1$ with $b_1 = a_2$. So we can compute an entry in $\mathcal{O}(m \cdot C)$ time and the whole table in $\mathcal{O}(m^3 \cdot C^3 \cdot \Gamma)$ time.

Putting everything together, we obtain a method to compute the table T for the original graph G in $\mathcal{O}(m^5 \cdot C^4 \cdot \Gamma^2)$ time. The value of a maximal integral generalized flow is the maximal value b for which $T(a,b) = 1$, which can be found in $\mathcal{O}(m^2 \cdot C^2 \cdot \Gamma)$ time once the table T has been computed. If we are not only interested in the maximum flow value but also in the flow itself, we can use the same approach and store, for each 1-entry in a table belonging to a graph with more than one arc, why it was set to 1, i.e., if we set $T'(a,b) = 1$ because $T_1(a_1,b_1) = 1$ and $T_2(a_2,b_2) = 1$ with $a = a_1 + a_2$ and $b = b_1 + b_2$ holds, we also store $T_1(a_1,b_1)$ and $T_2(a_2,b_2)$. This enables us to go back from the entry representing the maximum flow value through the different tables until we reach the tables for the graphs with only one arc in order to see how the flow is put together. Thus, we have found a method to compute the max integral generalized flow of an sp graph in pseudo-polynomial time.

References

[1] W.W. Bein, P. Brucker, and A.J. Hoffman. Series parallel composition of greedy linear programming problems. *Mathematical*

Programming, 62:1–14, 1993.

[2] W.W. Bein, P. Brucker, and A. Tamir. Minimum cost flow algorithms for series-parallel networks. *Discrete Applied Mathematics*, 10(2):117 – 124, 1985.

[3] K. Beygang. Modelle und Algorithmen für die Leerwagendisposition im Schienengüterverkehr. Master's thesis, TU Kaiserslautern, Germany, 2008. In German.

[4] H. Booth and R.E. Tarjan. Finding the minimum-cost maximum flow in a series-parallel network. *Journal on Algorithms*, 15(3):416–446, 1993.

[5] B. Engels, S.O. Krumke, R. Schrader, and C. Zeck. Integer flow with multipliers: The special case of multipliers 1 and 2. In *CTW*, pages 239–243, 2009.

[6] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[7] L. Khachiyan. A polynomial time algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1096, 1979. In Russian.

[8] K. Onaga. Dynamic programming of optimum flows in lossy communication nets. *IEEE Transactions on Circuit Theory*, 13:282–287, 1966.

[9] T. Radzik. Faster Algorithms for the Generalized Network Flow Problem. *Mathematics of Operations Research*, 23(1):69–100, 1998.

[10] S. Sahni. Computationally related problems. *SIAM Journal on Computing*, 3(4):262–279, 1974.

[11] É. Tardos and K.D. Wayne. Simple generalized maximum flow algorithms. In *Proceedings of the 6th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 310–324, London, UK, 1998. Springer.

[12] J. Valdes, R.E. Tarjan, and E.L. Lawler. The recognition of series parallel digraphs. In *STOC '79: Proceedings of the eleventh annual ACM symposium on Theory of computing*, pages 1–12, New York, NY, USA, 1979. ACM.