ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute for Operations Research
ETH Zurich HG G21-22

Prof. Dr. Rico Zenklusen and Assistants
Contact: math.opt@ifor.math.ethz.ch

IFOR
Institute for Operations Research
Department of Mathematics

Fall 2019

Mathematical Optimization – Solutions to problem set 8

https://moodle-app2.let.ethz.ch/course/view.php?id=4844

Problem 1: Flow decomposition

We claim that the following algorithm achieves the desired.

Algorithm 1 (Flow decomposition)

Input: Digraph $G = (V, A)$ with arc capacities $u\colon A \to \mathbb{Z}_{\geq 0}$, distinct $s, t \in V$, s-t flow $f\colon A \to \mathbb{Z}_{\geq 0}$.

Output: s-t paths $P_1, \dots, P_k$ and cycles $C_1, \dots, C_\ell$ in G, values $\gamma_1, \dots, \gamma_k \in \mathbb{Z}_{\geq 0}$ and $\eta_1, \dots, \eta_\ell \in \mathbb{Z}_{\geq 0}$ such that $k + l \leq |A|$ and $f = \sum_{i=1}^{k} \gamma_i \chi^{P_i} + \sum_{j=1}^{\ell} \eta_j \chi^{C_j}$.

1. **Initialization:**
   $i = 1$, $j = 1$.
2. **while** $(\nu(f) > 0)$**:**
   Find an s-t path P_i in $\operatorname{supp}(f)$, and let $\gamma_i = \min\{u(a)\colon a \in P_i\}$.
   Decrease $f(a)$ by γ_i for all a in P_i, increase i by 1.
3. **while** $(\operatorname{supp}(f) \neq \emptyset)$ **do:**
   Find a cycle C_j in $\{a \in A\colon f(a) > 0\}$, and let $\eta_j = \min\{u(a)\colon a \in C_j\}$.
   Decrease $f(a)$ by η_j for all a in C_j, increase j by 1.
4. **return** $(P_1, \dots, P_k, \gamma_1, \dots, \gamma_k, C_1, \dots, C_\ell, \eta_1, \dots, \eta_\ell)$.

Note that the algorithm above does not specify how to actually find s-t paths P_i and cycles C_j. We will specify this below.

Let us consider the first while loop, which is executed if $\nu(f) > 0$. Note that in this case, $\operatorname{supp}(f)$ contains an s-t path: If not, there would be an s-t cut $C \subseteq V$ such that $f(a) = 0$ for all $a \in \delta^+(C)$, i.e., $f(\delta^+(C)) = 0$. But then, $\nu(f) = f(\delta^+(C)) - f(\delta^-(C)) \leq f(\delta^+(C)) = 0$, contradicting the condition $\nu(f) > 0$. Thus, an s-t path P_i in $\operatorname{supp}(f)$ exists, and hence can be found using BFS in time $O(|V|+|A|)$. The value of γ_i can be found in the same time. Note that $\gamma_i > 0$ because $u(a) \geq f(a) > 0$ for all $a \in P_i$. Observe that in the i^{th} iteration of the while-loop, when decreasing values of f along P_i, f remains a feasible flow, but its value is strictly decreased by γ_i. Thus, after a finite number of iterations, say k, we are left with a flow f' in G of value $\nu(f') = 0$, and the first while-loop terminates. Furthermore, we have $f = \sum_{i=1}^{k} \gamma_i \chi^{P_i} + f'$ by construction.

Consequently, the second while loop starts with a flow f' of value $\nu(f') = 0$. In particular, this implies that for every vertex $v \in V$, we have $f'(\delta^-(v)) = f'(\delta^+(v))$ (this holds for all vertices $v \in V \setminus \{s, t\}$ by definition of a flow, and for $v \in \{s, t\}$ because $\nu(f) = 0$). Hence any vertex with positive indegree in the graph $(V, \operatorname{supp}(f'))$ has positive outdegree, so a cycle in $\operatorname{supp}(f')$ can be found greedily: Start a walk at a vertex with non-zero degree in $(V, \operatorname{supp}(f'))$, and follow outgoing edges until a cycle C_j is closed. Thus, a cycle C_j and the corresponding value η_j can be found in time $O(|A|)$. Note that $\eta_j > 0$ because $u(a) \geq f(a) > 0$ for all $a \in C_j$. Observe that in the j^{th} iteration of the while-loop, when decreasing values of f' on C_j, f' remains a feasible flow of value $\nu(f) = 0$. Moreover, by choice of η_j, at least one flow value on an edge of C_j is reduced to 0 in the j^{th} iteration, reducing $\operatorname{supp}(f)$ by at least one. Consequently, the second while-loop terminates after a finite number of steps, say ℓ. By construction, we furthermore obtain $f' = \sum_{j=1}^{\ell} \eta_j \chi^{C_j}$.

Combining the two steps, we thus have

$$f = \sum_{i=1}^{k} \gamma_i \chi^{P_i} + \sum_{j=1}^{\ell} \eta_j \chi^{C_j} \ .$$

Observe that in both while loops, γ_i and η_j are chosen such that in each step, the flow on at least one arc is reduced from a non-zero value to zero, i.e., $|\operatorname{supp}(f)|$ is reduced by at least one in each step. This implies that $k + l \leq \operatorname{supp}(f) \leq |A|$. As seen above, a single iteration of each of the while-loops takes time at most $O(|V| + |A|)$, and thus we obtain an overall running time bound of $O(|A|^2 + |V| \cdot |A|)$, i.e., the proposed procedure is efficient.

Problem 2: Improving over Edmonds-Karp: Blocking flows and Dinic's algorithm

(a) We have to show that f' satisfies both capacity and balance constraints. For capacity constraints, note that $f_0(a) \leq u_f(a) = u(a) - f(a)$ because f_0 satisfies capacity constraints with respect to u_f, hence

$$f'(a) = f(a) + f_0(a) - f_0(a^R) \leq f(a) + u(a) - f(a) = u(a)$$

holds for all $a \in A$, which is precisely the capacity constraints. To derive balance constraints, observe that

$$\begin{aligned}
f'\big(\delta_G^+(v)\big) - f'\big(\delta_G^-(v)\big) &= \sum_{a \in \delta_G^+(v)} f'(a) - \sum_{a \in \delta_G^-(v)} f'(a) \\
&= \sum_{a \in \delta_G^+(v)} \big(f(a) + f_0(a) - f_0(a^R)\big) - \sum_{a \in \delta_G^-(v)} \big(f(a) + f_0(a) - f_0(a^R)\big) \\
&= f\big(\delta_G^+(v)\big) - f\big(\delta_G^-(v)\big) + \left(\sum_{a \in \delta_G^+(v)} f_0(a) + \sum_{a \in \delta_G^-(v)} f_0(a^R)\right) - \left(\sum_{a \in \delta_G^+(v)} f_0(a^R) + \sum_{a \in \delta_G^-(v)} f_0(a)\right) \\
&= \Big[f\big(\delta_G^+(v)\big) - f\big(\delta_G^-(v)\big)\Big] + \Big[f_0\big(\delta_{G_f}^+(v)\big) - f_0\big(\delta_{G_f}^-(v)\big)\Big] \ .
\end{aligned}$$

By balance constraints for f in G and f_0 in G_f, we obtain that both brackets in the last line above are 0 whenever $v \in V \setminus \{s,t\}$, non-negative if $v = s$ and non-positive if $v = t$. Consequently, balance constraints for f' in G are satisfied.

In particular, plugging in $v = s$ into the above, we get that

$$\begin{aligned}
\nu(f') &= f'\big(\delta_G^+(s)\big) - f'\big(\delta_G^-(s)\big) \\
&= \Big[f\big(\delta_G^+(s)\big) - f\big(\delta_G^-(s)\big)\Big] + \Big[f_0\big(\delta_{G_f}^+(s)\big) - f_0\big(\delta_{G_f}^-(s)\big)\Big] = \nu(f) + \nu(f_0) \ ,
\end{aligned}$$

as desired.

(b) Let f be a maximum s-t flow in a graph $G = (V, A)$ with edge capacities $u \colon A \to \mathbb{Z}_{\geq 0}$. Assume that f is not blocking, i.e., there exists an s-t path P in G such that no edge of P is saturated by f. In other words, $f(a) < u(a) - \varepsilon$ for all $a \in P$ and some $\varepsilon > 0$. But then P is an augmenting path in the residual graph G_f that allows an augmentation with volume at least ε, because the residual capacities u_f satisfy $u_f(a) = u(a) - f(a) \geq \varepsilon$ for all $a \in P$. Thus, we can augment f along P to obtain a flow of strictly larger value—contradicting the assumption that f is a maximum flow already. Thus, f must be blocking.

(c) An example of a flow that is blocking but not maximum is given in Figure 1.

(d) Let P be a shortest s-t path in (V, U_f). Then by definition of the s-t layered subgraph of (V, U_f), all vertices and edges of P are included in the s-t layered subgraph, hence P is an s-t path in that graph.

For the other direction, note that for all edges $e = (u, v)$ in the s-t layered subgraph of (V, U_f), we have $d(s, v) = d(s, u) + 1$. This is true because the edge (u, v) is included in the s-t layered

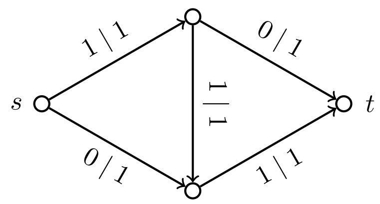

Figure 1: A flow f and arc capacities u are indicated in the form $f(a) \mid u(a)$ on every arc a. A maximum flow would have value 2, and the indicated flow of value 1 is blocking.

subgraph iff it lies on a shortest s-t path in (V, U_f)—but then the subpaths from s to u and s to v are shortest s-u and s-v paths, respectively, giving the claimed relation for the distances. Consequently, for any s-t path $P = (v_0, e_1, v_1, \dots, e_k, v_k)$ in the s-t layered subgraph of G, where $v_0 = s$ and $v_k = t$, we have $d(s, v_i) = i$, and hence $d(s,t) = k = \text{length}(P)$. Thus, P is a shortest s-t path in (V, U_f).

(e) Apply the flow decomposition theorem from Problem 1 of this problem set to obtain s-t paths $P_1, \dots, P_k$ and cycles $C_1, \dots, C_\ell$ in the s-t layered subgraph of (V, U_f), and values $\gamma_1, \dots, \gamma_k \in \mathbb{Z}_{>0}$ and $\eta_1, \dots, \eta_\ell \in \mathbb{Z}_{>0}$ such that $f_0 = \sum_{i=1}^k \gamma_i \chi^{P_i} + \sum_{i=1}^\ell \eta_i \chi^{C_i}$. But note that the s-t layered subgraph of (V, U_f) does not have (oriented) cycles, as the edges are always oriented along strictly increasing distance from s, as we proved in part (d). Thus, we must have $\ell = 0$, and hence

$$f_0 = \sum_{i=1}^k \gamma_i \chi^{P_i} \quad . \tag{1}$$

Also note that as P_i is an s-t path in the s-t layered subgraph of (V, U_f), we get from the previous subproblem that P_i is a shortest s-t path in (V, U_f) for every $i \in [k]$. To see that the second property holds true, we proceed by induction on k, the number of paths that f_0 is decomposed into.

For $k = 1$, the induction basis, we have $f_0 = \gamma_1 \chi^{P_1}$. In this case, by definition, augmenting f along f_0 is precisely the same as augmenting f along the augmenting path P_1 with augmentation volume γ_1. Note that γ_1 is a feasible augmentation volume because f_0 is a feasible flow in (V, U_f) with respect to residual capacities.

For the inductive step, assume that we know the result if f_0 is decomposed into $k-1$ paths for some $k \geq 2$, and let's prove it for k paths. Thus, assume that $f_0 = \sum_{i=1}^k \gamma_i \chi^{P_i}$. Let $f_0^{(1)} = \sum_{i=1}^{k-1} \gamma_i \chi^{P_i}$, and let $f_0^{(2)} = \gamma_k \chi^{P_k}$. Let $f^{(1)}$ denote the flow obtained from augmenting f along $f_0^{(1)}$. By assumption, we know that $f^{(1)}$ equals the flow obtained from consecutively augmenting f along the paths P_i with augmentation volume γ_i for $i \in [k-1]$. We have to prove that the flow $f^{(2)}$ obtained by augmenting $f^{(1)}$ along $f_0^{(2)}$ is the same as the flow obtained from augmenting $f^{(1)}$ along the augmenting path P_k with augmentation volume γ_k. Again, it is immediate by definition that both ways of augmenting $f^{(1)}$ result in the same flow—but it is not clear that P_k is in fact a feasible augmenting path for $f^{(1)}$ along which we can augment with volume γ_k: Note that P_k is a path in the residual graph G_f, but we need one in $G_{f^{(1)}}$ and with respect to residual capacities $u_{f^{(1)}}$. To see that P_k is indeed such a path, we prove that

$$u_{f^{(1)}}(b) \geq \gamma_k \quad \text{for all } b \in P_k \quad . \tag{2}$$

Observe that $b \in P$ can either be an arc a of the original graph or a reverse arc a^R of an arc a in the original graph (recall the construction of the residual graph). Also note that if one of the paths P_i uses an arc $a \in A$, no other path P_i can use the arc $a^R \in A^R$, because arcs in the s-t layered subgraph of (V, U_f) (and hence arcs in paths P_i) connect vertices with consecutive and increasing distances from s. We need this observation in both of the following two cases.

Case 1: $b = a$ for some $a \in A \cap P_k$. In this case, (2) is equivalent to

$$c(a) - f^{(1)}(a) \geq \gamma_k \quad .$$

By definition, $f^{(1)}(a) = f(a) + f_0^{(1)}(a) - f_0^{(1)}(a^R)$. But as $a \in P_k$, we said above that $a^R \notin P_i$ for any $i \in [k]$, hence $f_0^{(1)}(a^R) = 0$. Additionally, we have $f_0^{(1)}(a) = f_0(a) - \gamma_k$ because $a \in P_k$. Together, this gives $f^{(1)}(a) = f(a) + f_0(a) - \gamma_k$, and thus

$$u_{f^{(1)}}(a) = c(a) - f^{(1)}(a) = c(a) - f(a) - f_0(a) + \gamma_k \geq \gamma_k \ ,$$

where the last inequality is true because f_0 is assumed to be a valid flow in (V, U_f) with respect to the residual capacities u_f, hence $f_0(a) \leq u_f(a) = c(a) - f(a)$, and thus $c(a) - f(a) - f_0(a) \geq 0$.

Case 2: $b = a^R$ for some $a^R \in A^R \cap P_k$. In this case, (2) is equivalent to

$$f^{(1)}(a) \geq \gamma_k \ .$$

Again, by definition, $f^{(1)}(a) = f(a) + f_0^{(1)}(a) - f_0^{(1)}(a^R)$. But as $a^R \in P_k$, we said above that $a \notin P_i$ for any $i \in [k]$, hence $f_0^{(1)}(a) = 0$. Additionally, we have $f_0^{(1)}(a^R) = f_0(a^R) - \gamma_k$ because $a^R \in P_k$. Together, this gives $f^{(1)}(a) = f(a) - \big(f_0(a^R) - \gamma_k\big)$, and thus

$$u_{f^{(1)}}(a) = f(a) - f_0(a^R) + \gamma_k \geq \gamma_k \ ,$$

where the last inequality is true because f_0 is assumed to be a valid flow in (V, U_f) with respect to the residual capacities u_f, hence $f_0(a^R) \leq u_f(a^R) = f(a^R)$, and thus $f(a) - f_0(a^R) \geq 0$.

This proves (2), hence P_k is indeed an augmenting path in $G_{f^{(1)}}$ along which we can augment with augmentation volume γ_k, hence the result follows.

(f) Assume for contradiction that there is an s-t path P in $(V, U_{f'})$ of length equal to the distance of s and t in (V, U_f). Note that by definition of $U_{f'}$, the residual capacities $u_{f'}$ on edges of P are all strictly positive, hence P is a shortest augmenting path for f'.

We claim that the path P was not present in (V, U_f). To see this, assume the opposite, in which case it was a shortest s-t path in that graph, and hence appeared in the s-t layered subgraph of (V, U_f). But then, it was blocked by f_0, i.e., there is an arc on P such that f_0 uses the full remaining residual capacity on that arc, implying that this arc is no longer present in $(V, U_{f'})$. This contradicts the assumption that P is a path in $(V, U_{f'})$ and proves the claim.

The claim implies that there is at least on arc $b \in P$ that is present in $(V, U_{f'})$ but not in (V, U_f). This can only happen if $f_0(b^R) > 0$. Applying part (e) and interpreting the augmentation along f_0 as consecutive augmentations along paths P_i with positive volumes, we see that at least one of these paths P_i must use the arc b^R.

However, note that P and the paths P_i are all of the same length (namely, the s-t distance in (V, U_f)), hence we can interpret them as paths appearing in the same phase of the Edmonds-Karp algorithm (see the proof of Theorem 4.34 in the script). Thus by the hint, there cannot be a path P_i that uses b^R, as P uses b. This is the desired contradiction, and we can thus conclude that the distance of s and t in $(V, U_{f'})$ is strictly larger than in (V, U_f).

(g) By part (f), we know that the s-t distance in (V, U_f) strictly increases in every iteration of the while-loop. As this distance can be at most $n - 1$, the while-loop terminates after at most $n - 1$ iterations. Moreover, by part (e), we can interpret every augmentation along a blocking flow as successive augmentations along shortest s-t paths in the corresponding residual graphs, hence correctness follows from correctness of the Edmonds-Karp algorithm.

(h) Constructing (V, U_f) can be done in time $O(m)$ by checking residual capacities of all edges (and their reverse edges). To determine the vertices appearing in the s-t layered subgraph of (V, U_f), we can for example determine the distance of from s to every vertex (by a BFS in (V, U_f)), and the distance to t from every vertex (by a BFS in (V, U_f) with reversed edges): Precisely those vertices appear in the s-t layered subgraph where the two distances from s and to t sum to the distance of s and t. For edges (u, v), we can then simply check if $d(s, v) = d(s, u) + 1$ and include the edge iff it holds true. Both BFS as well as checking all vertices and edges takes time $O(m)$, as desired (recall that we assumed $n = O(m)$).

(i) The initialization step takes time $O(m)$, the while loop is executed at most $n-1$ times (as argued in part (g), constructing the s-t layered subgraph at the beginning of each iteration takes time $O(m)$ (as argued in part (h)). Putting this together with the running time bound $\beta(m,n)$ for finding a blocking flow, we get a running time bound of $O(n(\beta(m,n)+m))$ for Dinic's algorithm.

(j) In general, we can find a blocking flow using Algorithm 2.

Algorithm 2 (Finding blocking flows)

Input: Digraph $G=(V,A)$ with capacities $u\colon A \to \mathbb{Z}$, distinct $s,t \in V$.
Output: Blocking flow in G.

1. **Initialization:**
   $f(a)=0$ for all $a \in A$.
2. **while** ($\exists s$-t path $P \subseteq \{a \in A\colon u(a) > 0\}$) **do:**
   Let $\gamma := \min\{u(b)\colon b \in P\}$.
   Increase $f(a)$ by γ, and decrease $u(a)$ by γ for all $a \in P$.
3. **return** f.

Note that in every step of the while-loop, the capacity $u(a)$ of at least one arc in A is decreased to zero, thus after at most m many iterations, the while loop will terminate. Checking if an s-t path in $\{a \in A\colon u(a) > 0\}$ exists (and finding it if it does) takes one BFS from s, hence time $O(m)$. Thus, in the generality above, the algorithm has running time $O(m^2)$. Before going into how to improve this for finding blocking flows in s-t layered graphs, let us discuss that the returned flow f is indeed blocking. To this end, note that whenever the flow value $f(a)$ on an arc a is increased, the corresponding capacity $u(a)$ is decreased accordingly, so $u(a) \geq 0$ will, at any stage of the algorithm, always denote the remaining capacity on a, i.e., the current flow $f(a)$ and the remaining capacity $c(a)$ together never exceed the initial capacity on a. This has two implications:

- The final f and the input capacities u satisfy $f(a) \leq u(a)$ for all $a \in A$. As additionally, f is increased only along s-t paths with a uniform value on the edges of each path, we get that the returned f is an s-t flow in G respecting the input capacities u.
- In the condition of the while-loop, we always check if there is an s-t path in G with remaining capacity, and we only stop if this is no longer the case. Thus, the final f is a blocking flow.

Hence, the algorithm above finds a blocking flow in arbitrary digraphs in time $O(m^2)$. In the special case where the input graph is an s-t layered graph, we can do better. By definition, an s-t layered graph only contains edges that lie on shortest s-t paths. Thus, when looking for s-t paths, we know that we can simply follow *any* outgoing edges starting from s until we arrive at t after at most $n-1$ steps, and every s-t path can be found like this. Thus, instead of an $O(m)$ time BFS, we can do an $O(n)$ step greedy approach to find an s-t path (if there is one). Together with the $O(m)$ bound on the number of iterations of the while loop, we get an $O(mn)$ running time bound for finding a blocking flow in s-t layered graphs.

By part (i), we thus immediately get that Dinic's algorithm can be implemented in time $O(mn^2)$.