Minimum Cost Flows in Graphs with Unit Capacities

Andrew V. Goldberg[*1], **Haim Kaplan**[2], **Sagi Hed**[2], and **Robert E. Tarjan**[*3,4]

1 **Amazon.com Inc.**
2 **School of Computer Science, Tel Aviv University**
3 **Department of Computer Science, Princeton University**
4 **Intertrust Technologies, Sunnyvale CA**

Abstract

We consider the minimum cost flow problem on graphs with unit capacities and its special cases. In previous studies, special purpose algorithms exploiting the fact that capacities are one have been developed. In contrast, for maximum flow with unit capacities, the best bounds are proven for slight modifications of classical blocking flow and push-relabel algorithms.

In this paper we show that the classical cost scaling algorithms of Goldberg and Tarjan (for general integer capacities) applied to a problem with unit capacities achieve or improve the best known bounds. For weighted bipartite matching we establish a bound of $O(\sqrt{r}m \log C)$ on a slight variation of this algorithm. Here r is the size of the smaller side of the bipartite graph, m is the number of edges, and C is the largest absolute value of an arc-cost. This simplifies a result of [Duan et al. 2011] and improves the bound, answering an open question of [Tarjan and Ramshaw 2012]. For graphs with unit vertex capacities we establish a novel $O(\sqrt{n}m \log(nC))$ bound. We also give the first cycle canceling algorithm for minimum cost flow with unit capacities. The algorithm naturally generalizes the single source shortest path algorithm of [Goldberg 1995].

1998 ACM Subject Classification E.1 Data Structures, F.2.2 Nonnumerical Algorithms and Problems, G.2.2 Graph Theory

Keywords and phrases minimum cost flow, bipartite matching, unit capacity, cost scaling

Digital Object Identifier 10.4230/LIPIcs.STACS.2015.406

1 Introduction

The input to the *minimum cost flow (MCF)* problem is a directed graph $G = (V, E)$ with integer arc costs and integer capacities. The goal is to find a circulation (flow) of minimum cost, where a circulation is a function on arcs that satisfies capacity constraints for all arcs and conservation constraints for all vertices. We denote $|V|$ by n, $|E|$ by m, the maximum capacity by U and the maximum cost value by C.[1] MCF has many applications (see e.g., [1]) and has been extensively studied.

An important subclass of MCF problems are the problems with unit capacities *(UMCF)*. In particular, the *weighted bipartite matching (WBM)* problem is a special case of UMCF. The goal in WBM is to find a matching of maximum weight in a weighted bipartite graph. The following two problems are closely related to WBM and are also a special case of UMCF. The *assignment (AS)* problem is to find a perfect matching of maximum weight, and the

* Part of the work was done while the author was at Microsoft Research
[1] When writing $\log U$ and $\log C$, we assume that $U > 1$ and $C > 1$, respectively.

32nd Symposium on Theoretical Aspects of Computer Science (STACS 2015).
Editors: Ernst W. Mayr and Nicolas Ollinger; pp. 406–419
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
SYMPOSIUM ON THEORETICAL ASPECTS OF COMPUTER SCIENCE

imperfect assignment (IAS) problem is to find a maximum weight matching of a given size F. The *single source shortest path (SSSP)* problem can be reduced to AS. Another special case is the UMCF problem on graphs with unit vertex capacities (or equivalently with an out-degree or an in-degree of one for every vertex).

The unit capacity maximum flow problem and the maximum bipartite matching problem can be reduced, respectively, to the UMCF and WBM problems with most arc costs zero and some arc costs plus or minus one. The best combinatorial bounds for these problems are $O(\min(m^{1/2}, n^{2/3})m)$ and $O(\sqrt{n}m)$, respectively [13, 7]. Recently, Madry [15] proved an $\tilde{O}(m^{10/7})$ bound using interior point methods. Better combinatorial bounds are known for important special cases of the maximum bipartite matching problem when the flow value (which is the size of the matching) F is small and when the smaller of the two sizes of the input bipartite graph, r, is small. The bound for the former case is $O(\sqrt{F}m)$ [12] and for the latter case, $O(\sqrt{r}m)$ [2]. The best combinatorial bound for maximum flow with unit vertex capacities is $O(\sqrt{n}m)$ [7].

Many of the results on unit-capacity maximum flow have been extended to UMCF. Gabow and Tarjan [8] showed an $O(\min(m^{1/2}, n^{2/3})m\log(nC))$ bound for UMCF and $O(\sqrt{n}m\log(nC)$ bound for WBM, AS, and IAS problems. Goldberg [9] gave an $O(\sqrt{n}m\log C)$ algorithm for SSSP, and Duan et al. [6] gave an algorithm for WBM with the same complexity. Ramshaw and Tarjan [16] developed an $O(\sqrt{r}m\log(rC))$ algorithm for WBM and an $O(\sqrt{F}m\log(FC))$ algorithm for IAS. However, these extensions are ad hoc: the algorithms are not MCF algorithms, and their connection to common MCF algorithms is not straightforward.

In this paper we develop a unified MCF framework for problems with unit and small integer capacities. The resulting algorithms are simple, intuitive, and match or improve the previous bounds. In essence we show that classical algorithms with slight modifications and a careful analysis give the same or better bounds for UMCF problems as special purpose algorithms.

We show, in fact, two unified frameworks for UMCF. The first, in Section 4, is a novel cycle canceling algorithm extending [9], which is the first cycle canceling algorithm that has good time bounds for UMCF problems. The second, in Section 3, is a pseudoflow framework which consists of the MCF cost-scaling algorithms of [11] with a fresh analysis for UMCF problems. Section 5 shows how to apply these frameworks to the special cases of UMCF. When applied to these special cases, the algorithms stay almost the same but require a more careful analysis.

Our paper unifies (or replaces) [11, 8, 16, 9] and the second part of [6] and obtains the following results:

- A novel $O(\sqrt{r}m\log(C))$ time bound for WBM, answering an open question of [16].[2]
- A novel $O(\sqrt{n}m\log(nC))$ time bound for UMCF with unit vertex capacities, which is the first extension of Even and Tarjan [7] from maximum flow to MCF.
- All the previous time bounds for UMCF, AS, IAS and SSSP. Namely $O(\min(m^{1/2}, n^{2/3})m\log(nC))$ for UMCF [3], $O(\sqrt{n}m\log(nC))$ for AS, $O(\sqrt{F}m\log(FC))$ for IAS and $O(\sqrt{n}m\log C)$ for SSSP.

[2] Note that $r < n$, so this bound improves both the $O(\sqrt{n}m\log C)$ bound of Duan et al. [6] and the $O(\sqrt{r}m\log(rC))$ bound of Ramshaw and Tarjan [16].

[3] And an $O(\min(n, \sqrt{mU}, n^{2/3}U^{1/3})m\min(U, \log n)\log(nC))$ bound for MCF.

2 Definitions and Notation

The input to the MCF problem is a directed graph $G = (V, E)$ where $|V| = n$ and $|E| = m$. Each arc $(v, w) \in E$ has an associated integer capacity $0 \leq u(v, w) \leq U$ and an associated integer cost $c(v, w)$ such that $|c(v, w)| \leq C$. For every input arc e, we add to E a *reverse arc* e^R with $u(e^R) = 0$ and $c(e^R) = -c(e)$. We also define $(e^R)^R = e$. To simplify notation, we usually assume that the graph has no parallel or anti-parallel arcs, so every arc is uniquely defined by its endpoints, and $(v, w)^R = (w, v)$.

A *circulation* f is a function $f : E \to R$ which is anti-symmetric: $\forall_{(v,w)\in E} f(v, w) = -f(w, v)$, satisfying the capacity constraints $\forall_{(v,w)\in E} f(v, w) \leq u(v, w)$ and the flow conservation constraints $\forall_{v\in V} \sum_{w\in V|(v,w)\in E} f(v, w) = 0$. For any flow function f we define $\text{cost}(f) = \frac{1}{2} \sum_{(v,w)\in E} f(v, w) \cdot c(v, w)$. A circulation is *optimal* if its cost is minimal. The *MCF problem* is to find an optimal circulation in G. The *MCF problem with vertex capacities of* F has the added constraint that $\forall_{v\in V} \sum_{(v,w)\in E, f(v,w)>0} f(v, w) \leq F$.

A *pseudoflow* f is a function $f : E \to R$ satisfying $f(v, w) = -f(w, v)$ and $f(v, w) \leq u(v, w)$ for every arc $(v, w) \in E$. We define $E_f = \{(v, w) \in E \mid f(v, w) < u(v, w)\}$ as the set of *residual arcs*. For a vertex v we define $e_f(v) = \sum_{(w,v)\in E} f(w, v)$. We call a vertex v with $e_f(v) > 0$ an *excess* and a vertex v with $e_f(v) < 0$ a *deficit*. Note that a circulation is just a pseudoflow with no excesses or deficits.

Given a *potential function* $p : V \to R$ we call $c_p(v, w) = c(v, w) - p(v) + p(w)$ the *reduced cost* of the arc (v, w). We say that a pseudoflow (or a circulation) f with potentials p is ϵ-*optimal* if for every arc $(v, w) \in E_f$ we have $c_p(v, w) \geq -\epsilon$. By linear programming duality, a 0-optimal circulation is optimal. We define $E_A = \{(v, w) \in E_f \mid c_p(v, w) < 0\}$ as the set of *admissible arcs*. We define $G_A = (V, E_A)$ as the *admissible subgraph*.

A graph G is *bipartite* if we can partition V into V_1 and V_2 such that $E \subseteq V_1 \times V_2$. A bipartite graph G is *balanced* if $|V_1| = |V_2|$. Given a bipartite graph, a *matching* is a set of edges that do not have common vertices. A *perfect matching* is a matching such that any vertex is adjacent to an edge in the matching. If the edges have weights, the weight of a matching is the sum of the edge weights. The WBM problem is to compute a matching with maximum weight among all matchings. The AS problem is to compute a perfect matching in a bipartite graph with maximum weight among all perfect matchings.[4] The IAS problem is to compute a matching of size F in a bipartite graph with maximum weight among all such matchings. In these problems we use C as the maximum weight of an edge in the bipartite graph. The WBM, AS, and IAS problems can be formulated as UMCF problems (see Section 5).

A circulation is optimal if and only if there exists a potential function such that all residual arcs have non-negative reduced costs [14]. Another optimality condition is that a circulation is optimal if and only if the residual graph has no cycles of negative cost.

The cost-scaling method was introduced in [3, 17]. We use the more efficient variant due to Goldberg and Tarjan [11]. The cost-scaling method proceeds in iterations which we call *cost scales*; each one starts with a 2ϵ-optimal circulation and ends with an ϵ-optimal circulation. It starts with the zero circulation which is C-optimal. When $\epsilon < 1/n$, an ϵ-optimal circulation is optimal, so the method takes $O(\log(nC))$ iterations. For efficiency, we restrict ourselves to values of ϵ that are integral powers of two. Unless mentioned otherwise, we use $2^{\lceil \log C \rceil}$ as the initial value of ϵ. We assume that the reader is familiar with Dijkstra's algorithm [5] and its bucket-based implementation due to Dial [4].

[4] Note that a perfect matching exists only in a balanced bipartite graph.

Algorithm 1 The cost scaling blocking flow algorithm of Goldberg and Tarjan [11].

```
function MCF(G)
    ε ← C, p ← 0, f ← 0
    while ε ≥ 1/n do (ε, f, p) ← REFINE(ε, f, p)
    end while
    return f
end function

function RAISE-POTENTIALS(ε, f, p)
    while ∄ E_A-path from excess to deficit do
        S ← {v ∈ V | ∃ E_A-path from an excess to v}
        ∀v ∈ S : p(v) ← p(v) + ε
    end while
end function

function REFINE(ε', f', p')
    ε ← ε'/2, p ← p', f ← f'
    ∀(v, w) ∈ E_A: f(v, w) ← u(v, w)
    while f is not a circulation do
        RAISE-POTENTIALS(ε, f, p)
        f ← f + a blocking flow in G_A
    end while
    return (ε, f, p)
end function
```

3 Pseudoflow Framework

Our pseudoflow framework for UMCF is based on the cost scaling algorithms of [11]. In this section we analyze the blocking flow algorithm of [11] for UMCF. One can get similar bounds for the push-relabel variant of [11]; we defer this result to the full version of the paper.

Algorithm 1 describes the blocking flow cost scaling algorithm of [11]. We say f is a *blocking flow* in G_A if f is a pseudoflow that: (i) Satisfies conservation constraints everywhere except at the excesses and deficits of G_A. (ii) Saturates at least one arc on every residual path from an excess to a deficit in G_A.

The algorithm starts a cost scale (the `refine` method in Algorithm 1) by saturating all admissible arcs, which turns the circulation into a pseudoflow. It then performs the following two steps until the pseudoflow becomes a circulation:

1. Iteratively increase by ϵ the potential of all vertices reachable from some excess in G_A, creating new admissible arcs. Stop when there is an admissible excess-to-deficit path.
2. Compute a blocking flow in G_A and add it to the current pseudoflow.

We implement step (1) (the `raise-potentials` method in Algorithm 1) with a modification that makes its running time linear, as described in Section 3.1. To analyze step (2), recall that Goldberg and Tarjan [11] show that G_A is acyclic throughout the execution of the algorithm. Since G_A also has unit capacities, we can compute a blocking flow in G_A in $O(m)$ time by a depth-first search in G_A.

Let f' and p' be the circulation and potentials at the start of the current cost scale (f' with p' is 2ϵ-optimal). Let f and p be the current pseudoflow and potentials maintained by the algorithm (f is ϵ-optimal w.r.t. p). Let $\lambda = \min(m^{1/2}, n^{2/3})$. Let $d(v) = \frac{p(v)-p'(v)}{\epsilon}$. Note that $d(v)$ must be integral. We divide every cost scale into two phases. The second phase begins when every excess v has $d(v) \geq \lambda$.

▶ **Lemma 1.** *The first phase of every cost scale runs in $O(m\lambda)$ time.*

Proof. After every blocking flow computation, for every excess v, $d(v)$ increases by at least one. It follows that for every excess v, $d(v) \geq \lambda$ after at most λ blocking flow computations. Since both step (1) and step (2) take $O(m)$ time, the lemma follows. ◀

▶ **Lemma 2.** *If k is the total excess at the start of the second phase, then the second phase takes $O(mk)$ time.*

Proof. Every blocking flow computation sends at least one unit of flow from some excess to some deficit. Since both step (1) and step (2) take $O(m)$ time, the lemma follows. ◀

Lemma 1 and 2 imply an $O(\lambda m)$ time bound on the Goldberg-Tarjan algorithm on UMCF, if we can show that when the second phase starts there is $O(\lambda)$ excess left. We introduce the following notation. We define $E^+ = \{(v,w) \in E \mid f'(v,w) > f(v,w)\}$ and let $G^+ = (V, E^+)$ be the subgraph of G induced by the arcs of E^+. Note that if $(v,w) \in E^+$ then $(v,w) \in E_f$ and $(w,v) \in E_{f'}$. We say $Y \subseteq E$ is an *excess-deficit cut* if every path from an excess to a deficit contains an arc in Y.

Lemma 3 and 4 are the core of our analysis of the pseudoflow framework. They allow us to extend the analysis of maximum flow algorithms to MCF by applying the classical arguments to E^+ rather then to E_f. Lemma 4 is an extension of Lemma 5.7 of [11]. It essentially states that an excess-deficit cut in E^+ bounds the total amount of excess, just as a cut on E_A does. Lemma 5 will then conclude the analysis.

▶ **Lemma 3.** *If $(v,w) \in E^+$ then $d(v) \leq d(w) + 3$.*

Proof. Since $(w,v) \in E_{f'}$ then $c_{p'}(w,v) \geq -\epsilon' = -2\epsilon$ so we have $c_{p'}(v,w) \leq 2\epsilon$. Since $(v,w) \in E_f$ we maintain the invariant $c_p(v,w) \geq -\epsilon$. We subtract the inequalities and get $c_p(v,w) - c_{p'}(v,w) \geq -\epsilon - 2\epsilon$ and so $d(w)\epsilon - d(v)\epsilon \geq -3\epsilon$. It follows that $d(v) \leq d(w) + 3$. ◀

▶ **Lemma 4.** *Let f' and f be a circulation and pseudoflow in $G = (V,E)$. Let Y be an excess-deficit cut in G^+. Then*
$\sum_{v \in V, e_f(v) > 0} e_f(v) \leq \sum_{(v,w) \in Y} (f'(v,w) - f(v,w)) \leq \sum_{(v,w) \in Y} u_f(v,w)$.

Proof. The flow $f' - f$ is a feasible flow in G_f since for every arc $(v,w) \in E_f$ we have $f'(v,w) - f(v,w) \leq u(v,w) - f(v,w) = u_f(v,w)$. By definition we get that $f + (f' - f) = f'$. Since f' is a circulation, we get that $f' - f$ drains all the excess in f. ◀

▶ **Lemma 5.** *When the second phase starts there is $O(\lambda)$ excess left.*

Proof. During the second phase every excess v has $d(v) \geq \lambda$ and every deficit v has $d(v) = 0$.

Assume $\lambda = \sqrt{m}$. Consider the $\frac{1}{3}\lambda$ sets of arcs $A_i = \{(v,w) \in E^+ \mid d(v) \in \{3i, 3i - 1, 3i - 2\} \wedge d(w) \in \{3i - 3, 3i - 4, 3i - 5\}\}$ where $1 \leq i \leq \frac{\lambda}{3}$.[5] By Lemma 3 any set A_i is an excess-deficit cut in G_f. By the pigeon hole principle there must be a set A_i such that $|A_i| \leq \frac{m}{\lambda/3} = 3\sqrt{m}$. So by Lemma 4 the remaining excess is at most $3\sqrt{m} = 3\lambda$.

Assume $\lambda = n^{2/3}$. We apply the pigeon hole principle to the sets of nodes $N_i = \{v \in V \mid d(v) \in \{6i, 6i - 1, 6i - 2, 6i - 3, 6i - 4, 6i - 5\}\}$ where $1 \leq i \leq \frac{\lambda}{6}$.[6] There must be a set N_i such that $|N_i| \leq \frac{n}{\lambda/6} = 6n^{1/3}$. So there are at most $36n^{2/3}$ arcs in A_{2i} and therefore by Lemma 4 the remaining excess is at most $36n^{2/3} = 36\lambda$. ◀

Our analysis extends to networks with integer capacities bounded by U and produces a time bound of $O(\min\{n, m^{1/2}U^{1/2}, n^{2/3}U^{1/3}\} m \min\{\log n, U\} \log(nC))$. We defer an extended description to the full version of the paper.

3.1 Raise Potentials

We describe a faster implementation of `raise-potentials` from Algorithm 1 using Dial's shortest path algorithm. This kind of implementation is well known, see e.g. [8, 9, 16].

[5] Note that $A_1 = \{(v,w) \in E^+ \mid d(v) \in \{3,2,1\} \wedge d(w) = 0\}$. We assume for simplicity of presentation that $\frac{\lambda}{3}$ is an integer.

[6] Here we assume for simplicity that $\frac{\lambda}{6}$ is an integer.

Consider the length function $\ell(v,w) = \lfloor \frac{c_p(v,w)}{\epsilon} \rfloor + 1$ and the graph G_f with an additional source vertex r connected to every excess v with an arc (r,v) of length $\ell(r,v) = 0$. For every vertex v reachable from r, we find the distance $d_\ell(v)$ from r to v w.r.t. ℓ. Let u be the deficit with the shortest distance $d_\ell(u)$. For a vertex v such that $d_\ell(v) < d_\ell(u)$ we increase the potential $p(v)$ by $(d_\ell(u) - d_\ell(v))\epsilon$. Lemma 6 shows that this computes the same potentials as in the naive implementation of `raise-potentials` from Algorithm 1.

Lemma 7 shows that we need to compute distances only up to $3n$. This implies that Dial's algorithm runs in linear time and space. We terminate the computation once we reach a deficit; alternatively, we can add an arc (r,v) for every vertex v with $\ell(r,v) = 3n$ and run the algorithm to the end.

▶ **Lemma 6.** *Consider a run of the naive implementation of* `raise-potentials` *from Algorithm 1. Let S_i be the set S computed at the i-th iteration, and let $S_0 = \{v \in V : \exists u$ with $e_f(u) > 0$ and v reachable from u in $G_A\}$. Then $S_i = \{v \in V \mid d_\ell(v) \leq i\}$.*

Proof. The proof is by induction on iterations of the naive implementation. For the basis, note that S_0 is the set of vertices reachable from excesses in the admissible graph. If (v,w) is admissible, then $\ell(v,w) = 0$, which implies the claim for S_0. We assume the claim is true for $S_0, .., S_{k-1}$ and prove it for S_k. Note that p and ℓ are the potentials and lengths before running `raise-potentials`.

First, we show that for every vertex v with $d_\ell(v) \leq k$ we have $v \in S_k$. By definition of the naive implementation $S_i \subseteq S_k$ for $i < k$ since all admissible arcs into vertices in S_i in iteration i remain admissible in subsequent iterations. Therefore by the induction hypothesis we need to consider only vertices v with $d_\ell(v) = k$. Let P be the shortest path from r to v (by ℓ distance). Going backwards from v towards r on P let (u,x) be the first arc we encounter with $d_\ell(u) < d_\ell(x) = k$ (possibly $x = v$). It follows that the arcs along P from x to v have zero length ℓ and are therefore admissible. By the induction hypothesis we have that $u \in S_{d_\ell(u)} \setminus S_{d_\ell(u)-1}$. So in the k'th iteration the reduced cost of (u,x) is $c_p(u,x) - \epsilon \cdot (k - d_\ell(u)) = c_p(u,x) - \epsilon \cdot (d_\ell(x) - d_\ell(u)) = c_p(u,x) - \epsilon \cdot \lfloor \frac{c_p(u,x)}{\epsilon} \rfloor - \epsilon = (c_p(u,x) \bmod \epsilon) - \epsilon < 0$. So (u,x) is admissible in the k'th iteration and since there is an admissible path from x to v the claim follows.

Finally, we show that for every vertex $v \in S_k$ we have $d_\ell(v) \leq k$. By the induction hypothesis we need to consider only vertices $v \in S_k \setminus S_{k-1}$. By definition of the naive implementation, in the k'th iteration we found an admissible path P from some vertex in S_{k-1} to v. Let (u,x) be the last arc on P such that $u \in S_{k-1}$ (possibly $x = v$). By the induction hypothesis we get that $u \in S_{d_\ell(u)} \setminus S_{d_\ell(u)-1}$. So the reduced cost of (u,x) in the k'th iteration is $c_p(u,x) - \epsilon \cdot (k - d_\ell(u)) < 0$. It follows that $\frac{c_p(u,x)}{\epsilon} - (k - d_\ell(u) - 1) < 1$ and so $\lfloor \frac{c_p(u,x)}{\epsilon} - (k - d_\ell(u) - 1) \rfloor \leq 0$. So by definition of shortest distances we get that $d_\ell(x) \leq d_\ell(u) + \ell(u,x) \leq k$. Since the subpath of P from x to v was always admissible we get that $d_\ell(v) \leq d_\ell(x) \leq k$. ◀

▶ **Lemma 7.** *For every $v \in V$ with $e_f(v) < 0$ we have $d_\ell(v) \leq 3n$.*

Proof. We assume for contradiction that $d_\ell(v) > 3n$. By Lemma 4 there must be an E^+ path from some excess w to v. We run the naive implementation of `raise-potentials` from Algorithm 1 for $d_\ell(v)$ iterations (we do not terminate if we find a deficit). By Lemma 6 after $d_\ell(v)$ iterations we have $d(w) \geq d_\ell(v) > 3n$ and $d(v) = 0$. Applying Lemma 3 telescopically along the E^+ path from w to v we get that $d(w) \leq d(v) + 3n = 3n$ which is a contradiction. ◀

Algorithm 2 A scaling iteration of the UMCF cycle canceling algorithm.

```
function REFINE(ε', f, p)
    ε ← (1/2)ε'
    while |E_bad| > 0 do
        k ← |E_bad|
        Cancel admissible cycles
        S ← FIND-SET-OR-CHAIN()
        if S is a set then ∀v ∈ S : p(v) ← p(v) + ε
        else ELIMINATE-CHAIN(S)
        end if
    end while
    return (ε, f, p)
end function

function ELIMINATE-CHAIN(S = (v_0, .., v_{|S|-1}))
    ∀_{t∈{1...|S|-2}} : PUSH(v_t, v_{t+1})
    RAISE-POTENTIALS(ε, f, p)
    ∀(v, w) ∈ E_A-path from v_{|S|-1} to v_0 : PUSH(v, w)
end function

function FIND-SET-OR-CHAIN()
    ∀(v, w) ∈ E_bad : ℓ'(v, w) = −1
    ∀(v, w) ∉ E_bad : ℓ'(v, w) = 0
    G' ← G_A with a new vertex r
    ∀v ∈ V: connect r to v with ℓ'(r, v) = 0
    Compute ℓ' distances from r in G'
    if ∃v ∈ V d_{ℓ'}(r, v) ≤ −√k then
        return E_A-path from r to v
    else
        ∀_{i<√k}: A_i ← {v ∈ V | d_{ℓ'}(r, v) = −i}
        j ← argmax_i |{(v, w) ∈ E_bad | w ∈ A_i}|
        return ∪_{j≤i≤√k} A_i
    end if
end function

function PUSH(v, w)
    f(v, w) ← f(v, w) + 1, f(w, v) ← f(w, v) − 1
end function
```

4 Cycle Canceling Framework

We present a cycle canceling algorithm that solves the UMCF problem in $O(m^{3/2}\log(nC))$ time. Unlike the methods of the previous section, the cycle-canceling algorithm always maintains a feasible circulation, which is desirable in some contexts. On the other hand, we were unable to prove an $O(mn^{2/3}\log(nC))$ bound as we did in the preflow framework. Our algorithm is based on the shortest path algorithm of Goldberg [9]. The main difference is that the shortest path algorithm halts when it finds a negative cycle and the MCF algorithm cancels such cycles and proceeds.

We introduce more notation. We define $E_{bad} = \{(v, w) \in E \mid c_p(v, w) < -\epsilon\}$ as the set of *bad arcs*. Note that $E_{bad} \subseteq E_A$. For a length function ℓ and vertices $x, y \in V$ we define $d_\ell(x, y)$ to be the distance according to ℓ from x to y. Given $S \subset V$ and $T \subset E$, we say S is *T-closed* if for every arc $(v, w) \in T$, $v \in S$ implies $w \in S$.

We describe the cycle-canceling variant of `refine` in Algorithm 2. We use k to denote the number of bad arcs. We perform iterations of canceling admissible cycles, finding either a large E_A-closed set or a long E_A path and then using the set or path to raise potentials or cancel more cycles and reduce the number of bad arcs. In each iteration we reduce k by at least $\sqrt{k}$. The `eliminate-chain` method is the main place where our algorithm differs from Goldberg's shortest path algorithm. The latter stops when it finds a negative cycle, while we need to cancel such cycles.

We define an *iteration* of the algorithm as one iteration of the loop of the `refine` method. In every iteration we first cancel all admissible cycles in G. Goldberg and Tarjan [10] describe how to perform this in $O(m)$ time for unit capacity networks. Next we run the `find-set-or-chain` method. This method finds either a path in G_A with at least $\sqrt{k}$ bad arcs or an E_A-closed set of vertices with at least $\sqrt{k}$ incoming bad arcs.

We find the path or the set by creating a new graph G' from G_A with a length function ℓ' as defined in Algorithm 2. Since we canceled admissible cycles in G, the graph G' is acyclic and we find a topological ordering of vertices in G' in $O(m)$ time. Using this ordering, we compute the distance $d_{\ell'}(r, v)$ for every vertex v in $O(m)$ time. If some vertex v has $d_{\ell'}(r, v) \le -\sqrt{k}$ then the path from r to v is the path we seek. Otherwise, by the pigeon hole principle, there is a set A_i of all the vertices with distance $-i$ that has at least $\sqrt{k}$ incoming bad arcs. The set of all vertices with distance $-i$ or less is therefore an E_A-closed set of

vertices with at least $\sqrt{k}$ incoming bad arcs.

If we find an E_A-closed set S, we increase the potentials of the vertices in S by ϵ. Since S is E_A-closed, raising the potential of S by ϵ does not create new bad arcs out of S. On the other hand each arc into S that was bad before the increase is no longer bad after the increase (f is 2ϵ-optimal so reduced costs are at least -2ϵ). Since S contained at least $\sqrt{k}$ incoming bad arcs, we reduce the number of bad arcs by at least $\sqrt{k}$.

If we find a path, we run the `eliminate-chain` method. This method eliminates the bad arcs on the path by adjusting potentials and canceling cycles with bad arcs, without creating new bad arcs. The `eliminate-chain` method runs in linear time. See Section 4.1.

Each iteration in `refine` finds either a set or a chain and in either case eliminates at least $\sqrt{k}$ bad arcs. So the number of iterations is $O(\sqrt{m})$. Each iteration takes $O(m)$ time. It follows that a cost scale takes $O(m^{3/2})$ time, yielding an $O(m^{3/2}\log(nC))$ UMCF algorithm.

4.1 Chain Elimination

The input to `eliminate-chain` is an admissible path $S = (v_0, v_1, \ldots v_{|S|-1})$. It is possible to implement `eliminate-chain` so that it cancels only negative cycles, but we describe a simpler implementation that may cancel cycles with positive cost, and a variant for which the circulation cost is monotonically decreasing.

The procedure starts by pushing a unit of flow from v_0 to $v_{|S|-1}$ along S. This saturates the bad arcs on S without creating new ones, and introduces a unit of deficit at v_0 and a unit of excess at $v_{|S|-1}$. We then run the `raise-potentials` method from Section 3.1 with a modified length function (see below). This increases the potentials and creates an admissible path, P, from $v_{|S|-1}$ to v_0 without introducing new bad arcs. To convert the pseudoflow into a circulation, we push a unit of flow on P. Since P is admissible, we do not create new bad arcs. Since bad arcs have reduced costs in the interval $[-2\epsilon, -\epsilon)$ the length ℓ of these arcs (defined in Section 3.1) is -1. We re-define ℓ to be zero on bad arcs, that is $\ell(v, w) = \max(0, \lfloor \frac{c_p(v,w)}{\epsilon} \rfloor + 1)$. After `raise-potentials` updates p, the reduced costs of admissible arcs (and in particular of bad arcs) cannot decrease, so no new bad arcs are created.

Pushing a unit of flow on S and then returning it on P changes the flow on a set of cycles. Reduced costs of arcs on P may be positive w.r.t. the potentials at the beginning of `eliminate-chain`, and the cycles may have positive cost. We can modify the algorithm to make sure the total cost of these cycles is negative; then at least one of the cycles is negative. We use a different length function ℓ_m. The reverse arcs of arcs along S have reduced costs in $(0, 2\epsilon]$. For such arcs with reduced costs in $(0, \epsilon]$ we set $\ell_m = 0$ (instead of 1) and for such arcs with reduced costs in $(\epsilon, -2\epsilon]$ we $\ell_m = 1$ (instead of 2). For every other residual arc e we define $\ell_m(e) = \ell(e)$. Also, if the reversal of S become admissible, we always select it as the admissible path we push the flow back on; in this case `eliminate-chain` does not change f. Running `raise-potentials` with ℓ_m is equivalent to running the naive implementation from Algorithm 1 if we also treat reverse arcs of arcs along S with reduced costs in $(0, \epsilon]$ as admissible. One can prove this using the same arguments as in the proof of Lemma 6. It follows that the residual path we find from $v_{|S|-1}$ to v_0 may contain reverse arcs of arcs in S with reduced costs in $(0, \epsilon]$. Pushing the flow back on these arcs creates new admissible arcs but cannot create new bad arcs. Lemma 8 proves that with ℓ_m we get decreasing circulation costs.

▶ **Lemma 8.** *Let f' be the flow before* `eliminate-chain` *and f after. If $f' \neq f$ then $cost(f) < cost(f')$.*

Proof. Let P be the return path we select after `raise-potentials` and $\bar{S}$ the reversal of S. P is a shortest path w..r.t. ℓ_m and $\bar{S}$ is not, so we have $\sum_{(v,w)\in P} \ell_m(v,w) < \sum_{(v,w)\in \bar{S}} \ell_m(v,w)$ and so $\sum_{(v,w)\in P\setminus\bar{S}} \ell_m(v,w) < \sum_{(v,w)\in \bar{S}\setminus P} \ell_m(v,w)$. It follows that $\sum_{(v,w)\in P\setminus\bar{S}} \frac{c_p(v,w)}{\epsilon} \leq \sum_{(v,w)\in P\setminus\bar{S}} \max(0, \lfloor \frac{c_p(v,w)}{\epsilon} \rfloor + 1) = \sum_{(v,w)\in P\setminus\bar{S}} \ell_m(v,w)$ $< \sum_{(v,w)\in \bar{S}\setminus P} \ell_m(v,w) = \sum_{(v,w)\in \bar{S}\setminus P} (\lceil \frac{c_p(v,w)}{\epsilon} \rceil - 1) \leq \sum_{(v,w)\in \bar{S}\setminus P} \frac{c_p(v,w)}{\epsilon}$. So we get that $\sum_{(v,w)\in P} c_p(v,w) < \sum_{(v,w)\in \bar{S}} c_p(v,w)$ and therefore $\sum_{(v,w)\in P\cup S} c_p(v,w) < 0$. ◀

5 Special Case Improvements and Generalizations

In this section we improve the bounds for our frameworks from Sections 3 and 4 for the special cases of SSSP, IAS, WBM and UMCF with unit vertex capacities.

First we review reductions from the WBM, AS and IAS problems to UMCF. The reduction from WBM is as follows. Given an instance $H = (V_1 \cup V_2, E_H)$ of WBM, we create a graph $G = (V, E)$. We add two new vertices s and t so that $V = V_1 \cup V_2 \cup \{s, t\}$. For each edge $(v, w) \in E_H$ we have an arc $(v, w) \in E$ whose cost in G is the negation of its cost in H (so that a maximum weight becomes a minimum cost). For every vertex $v \in V_1$ we add a new arc (s, v) with $c(s, v) = 0$. For every vertex $v \in V_2$ we add a new arc (v, t) with $c(v, t) = 0$. We also add the arc (t, s) with $c(t, s) = 0$. Let $r = \min\{|V_1|, |V_2|\}$. We set the capacities of all arcs to 1 except for the arc (t, s), which gets a capacity of $u(t, s) = r$ (we can replace (t, s) by r parallel unit capacity arcs to get an UMCF instance). Clearly a minimum cost flow in G corresponds to a matching of maximum weight in H.

The reduction of AS to UMCF is the same as for WBM except that we set $c(t, s) = -nC$. Clearly a minimum cost flow in G corresponds to a perfect matching of maximum weight in H. The maximum absolute value of an arc cost in is nC rather than C. However we still state our bounds as a function of C defined as the largest absolute value of a weight in H. We reduce IAS to UMCF using the same reduction as for AS, except $u(t, s) = F$ and $c(t, s) = -FC$.

5.1 The Single Source Shortest Path Problem

We show how the cycle canceling algorithm of Section 4 can be slightly modified to match the $O(\sqrt{n}m \log C)$ time bound of [9]. In the modified algorithm we maintain f as the zero circulation except temporarily inside the `eliminate-chain` procedure. When the algorithm terminates, the reduced costs of all arcs are non-negative, so we can find all shortest paths from a source using Dijkstra's algorithm in $O(m + n \log n)$ time.

We run the $O(\log C)$ cost scales of the cycle canceling algorithm of Section 4 with the length function ℓ_m until $\epsilon = 1$. However, we terminate if we find an admissible cycle in `refine` or if the return path in `eliminate-chain` is not the reversal of S. In these cases there is a negative cycle in G_f. An analysis similar to the one of Section 4 shows that the running time of a cost scale is $O(\sqrt{n}m)$ rather than $O(m^{3/2})$. This is done by defining a vertex as *bad* if it has a bad incoming arc and fixing at least $\sqrt{k}$ bad vertices in each iteration of `refine` where k is the total number of bad vertices. The potential increase of an E_A-closed set fixes all the bad vertices in the set. Lemma 9 shows that running `eliminate-chain` with ℓ_m either fixes all the bad vertices along S or there is a negative cycle and we can terminate.

▶ **Lemma 9.** *If (x, v_i) is bad and $v_i \in S$ then there is a negative cycle or $d_{\ell_m}(x) > d_{\ell_m}(v_i)$*

Proof. Let S' be the part of the reverse path of S from $v_{|S|-1}$ to v_i. If $d_{\ell_m}(x) \leq d_{\ell_m}(v_i)$ then there is a path P from $v_{|S|-1}$ to x with $\sum_{(v,w)\in P} \ell(v,w) \leq \sum_{(v,w)\in S'} \ell(v,w)$. Using the same

arguments as in the proof of Lemma 8 we get that $\sum_{(v,w)\in P} c_p(v,w) - \sum_{(v,w)\in S'} c_p(v,w) \leq 0$. So the cycle of P then (x, v_i) and then the part of S from v_i to $v_{|S|-1}$ is a negative cycle. ◀

When $\epsilon = 1$ we run one last, slightly modified, cost scale. We change the definition of an *admissible arc* to be an arc with non-positive reduced cost. Since all potentials and costs at this point are integral and thereby multiples of ϵ, the reduced cost of an admissible arc is either -1 or 0 and it is ≥ 1 for all other arcs. A *bad arc* is an admissible arc of reduced cost -1. Since zero reduced cost arcs are admissible, there are no zero reduced cost arcs outgoing from an E_A-closed set, and an increase of the potentials of the vertices in the set by 1 does not create new bad arcs. We define the length function ℓ in `eliminate-chain` accordingly: for every residual arc (v, w) we define $\ell(v, w) = \max(0, c_p(v, w))$. The arguments of Lemma 6 show that running `raise-potentials` with this length function does not create new bad arcs. The arguments of Lemma 8 show that if there isn't a negative cycle then we can push the flow back on the reversal of S.

It remains to show how to make the admissible network (by the new definition of admissibility) acyclic. We start every iteration of `refine` by contracting strongly connected components of G induced by zero reduced cost arcs. In the contracted graph we set the potentials of vertices to zero and the costs of arcs to be the reduced costs of the corresponding arcs before the contraction. This transformation takes linear time. At the end of the iteration we set the potential of every original vertex v to be its potential before the contraction plus the potential computed for the vertex into which v was contacted. See [9]. After the contraction G_A must be acyclic (otherwise there is a negative cycle and we terminate).

5.2 Imperfect Assignment Problem

We show an $O(\sqrt{F}m \log(FC))$ time bound for IAS using either the pseudoflow or the cycle canceling framework. Since the cost of (t, s) is $-FC$, the zero circulation is not C-optimal with respect to the zero potential function. So instead of starting with the zero circulation we start with the circulation f consisting of a flow of value F from s to t together with a flow of F through the arc (t, s), thereby saturating (t, s) (we assume that the value of the maximum flow from s to t is at least F as otherwise the problem is infeasible). This circulation is C-optimal with respect to the zero potential function.

The other modification we perform is a transformation of potentials before every cost scale as follows. For every vertex $v \in V_1$ we set $p(v) = p(v) + \epsilon$. For every vertex $v \in V_2$ we set $p(v) = p(v) + 2\epsilon$. We also set $p(t) = p(t) + 3\epsilon$. Lemma 10 specifies the outcome of the transformation.

▶ **Lemma 10.** *After applying the potential transformation we have* $\sum_{(v,w)\in E_A} u_f(v,w) \leq 3F$.

Proof. Consider an arc $(v, w) \in \{(s, t)\} \cup (\{s\} \times V_1) \cup (V_1 \times V_2) \cup (V_2 \times \{t\})$. Before the transformation $c_p(v, w) \geq -\epsilon$. After the transformation $c_p(v, w) \geq \epsilon - \epsilon = 0$. So $(v, w) \notin E_A$.

Let $R = E_f \cap ((V_1 \times \{s\}) \cup (V_2 \times V_1) \cup (\{t\} \times V_2))$ and consider an arc $(v, w) \in R$. Since $f(v, w) < u(v, w) = 0$ then $f(w, v) > 0$. Since f is a circulation and since all flow cycles of f contain (t, s) we get that $|R| \leq 3f(t, s)$. So $\sum_{(v,w)\in E_A} u_f(v, w) = |R| + u_f(t, s) \leq 3f(t, s) + u_f(t, s) = 3f(t, s) + u(t, s) - f(t, s) = 2f(t, s) + u(t, s) \leq 3F$. ◀

Lemma 10 implies that we start every cost scale in the cycle canceling framework with $O(F)$ bad arcs. An $O(\sqrt{F}m)$ time bound per cost scale follows using the same analysis as in Section 4. Lemma 11 implies an $\log(FC)$ bound on the number of cost scales:

▶ **Lemma 11.** *An ϵ-optimal circulation f with $\epsilon < \frac{1}{2F+4}$ is optimal.*

Proof. Consider a simple residual cycle Y. Pair up consecutive arcs that are not incident to s and t along Y. By the definition of G at least one arc in each pair is in $V_2 \times V_1$. It follows that $|Y| \leq 4 + 2|E_f \cap (V_2 \times V_1)| = 4 + 2|\{(w,v) \in V_1 \times V_2 \mid f(w,v) > 0\}| \leq 4 + 2F$, where the last inequality holds since by the conservation constraints in G we have that $|\{(w,v) \in V_1 \times V_2 \mid f(w,v) > 0\}| \leq u(t,s) \leq F$. It follows that $cost(Y) = \sum_{(v,w)\in Y} c(v,w) = \sum_{(v,w)\in Y} c_p(v,w) \geq \sum_{(v,w)\in Y} -\epsilon \geq -(4+2F)\epsilon > -1$. Since costs are integral it follows that $cost(Y) \geq 0$ so there are no negative residual cycles and therefore f is optimal. ◀

For the pseudoflow framework we use the same notation as in Section 3 except that we let $\lambda = \sqrt{F}$. We split a cost scale of the algorithm into two phases. The second phase starts when every excess v has $d(v) \geq \lambda$. Lemma 12 and the analysis from Section 3 yield the $O(\sqrt{F}m)$ time bound per cost scale for the pseudoflow framework.

▶ **Lemma 12.** *When the second phase starts there is $O(\sqrt{F})$ excess left.*

Proof. Consider the decomposition of the current pseudoflow f into simple cycles and paths from deficits to excesses. By the definition of G every simple flow cycle or path is of length at most 4. By Lemma 10 the total excess generated when we start a cost scale is at most $3F$ so there are at most $3F$ flow paths from deficits to excesses. Since $u(t,s) = F$ there are at most F flow cycles. It follows that $|\{(v,w) \in E \mid f(v,w) > 0\}| \leq 4 \cdot 3F + 4F = 16F$.

Let $R = E_f \cap ((V_1 \times \{s\}) \cup (V_2 \times V_1) \cup (\{t\} \times V_2) \cup \{(t,s)\})$. We have $\sum_{(v,w)\in R} u_f(v,w) \leq |\{(w,v) \in E \mid f(w,v) > 0\}| + F \leq 17F$. For $1 \leq i \leq \frac{\lambda}{12}$ let $A_i = \{(v,w) \in E_f \mid d(v) \in \{12i, 12i-1, ..., 12i-11\}\}$.[7] We apply the pigeon hole principle to the sets of arcs $A_i \cap R$. Since $\sum_{(v,w)\in R} u_f(v,w) \leq 17F$ there must be a set $A_i \cap R$ such that $\sum_{(v,w)\in A_i\cap R} u_f(v,w) \leq 17F / \frac{1}{12}\lambda = 204\sqrt{F}$. We show that each $A_i \cap R$ is an excess-deficit cut in G^+. A bound of $204\sqrt{F}$ on the remaining excess follows using Lemma 4.

Consider an excess-deficit E^+ residual path P. We show that it has an arc in each set $A_i \cap R$. Clearly every simple residual path in G_f, and P in particular, must have at least one arc in R out of every 4 consecutive arcs. Furthermore, since an excess has $d(v) = \lambda$, a deficit has $d(v) = 0$, and by Lemma 3, P must contain 4 consecutive arcs in each A_i. It follows that one of these 4 arcs is in R and therefore in $A_i \cap R$. ◀

5.3 Weighted Bipartite Matching

In this section we show the $O(\sqrt{r}m \log(C))$ time bound for the WBM problem. Recall that r is the number of vertices in the smaller "side" of the bipartite graph. We use the same ideas as Duan at al. [6], but within our cost-scaling frameworks, obtaining a simpler algorithm with the improved time bound. First we show how to obtain an $O(\sqrt{r}m \log(rC))$ time bound using either framework. Then we show how to improve the bound to $O(\sqrt{r}m \log C)$ by introducing a preprocessing stage based on the pseudoflow framework and a postprocessing stage based on the cycle canceling framework.

Before every cost scale we perform the same potential transformation as in Section 5.2. A lemma analogous to Lemma 10 shows that following this transformation $\sum_{(v,w)\in E_A} u_f(v,w) \leq 3r$. This gives a bound of $O(\sqrt{r}m)$ time for a cost scale in the cycle canceling framework. For the pseudoflow framework we define $\lambda = \sqrt{r}$ and split a cost scale into two phases as in Section 5.2. Lemma 13 implies the $O(\sqrt{r}m)$ time bound for a cost sale in the pseudoflow framework. For both frameworks, a lemma analogous to Lemma 11 implies $O(\log rC)$ cost scales yielding the $O(\sqrt{r}m \log(rC))$ bound.

[7] As in the previous section, to simplify the presentation we assume that $\frac{\lambda}{12}$ is an integer.

▶ **Lemma 13.** *When the second phase starts there is $O(\sqrt{r})$ excess left.*

Proof. Assume without loss of generality that $r = |V_1| \leq |V_2|$. For $1 \leq i \leq \frac{\lambda}{12}$ let $K_i = \{v \in V \mid d(v) \in \{12i, 12i-1, ..., 12i-10, 12i-11\}\}$ and let $N_i = K_i \cap V_1$.[8] By the pigeon hole principle there must be a set N_j such that $|N_j| \leq \frac{r}{\frac{1}{12}\lambda} = 12\sqrt{r}$. We argue in the next paragraph that the set of E^+ arcs outgoing from every N_i is an excess-deficit cut in G^+. Since each vertex in V_1 has only one outgoing arc that may be in E^+, the size of the cut corresponding to N_j is at most $12\sqrt{r}$ and the lemma follows by Lemma 4.

Consider a simple path P of E^+ arcs from an excess to a deficit. By Lemma 3, P must have at least 4 consecutive vertices in K_i. Since P is simple, one of these vertices v has $v \in V_1$ and so we get $v \in N_i$. The single outgoing E^+ arc of v must be on P. ◀

Next we improve the running time to $O(\sqrt{r}m \log C)$. The improved algorithm has three stages. First we find an $\frac{C}{\sqrt{r}}$-optimal circulation in $O(\sqrt{r}m)$ time. Then we run cost scales from either framework to get an $\frac{1}{\sqrt{r}}$-optimal circulation in $O(\sqrt{r}m \log C)$ time. Finally we convert the circulation to an optimal one in $O(\sqrt{r}m)$ time.

For the first stage we set $\epsilon = \frac{C}{\sqrt{r}}$. We set $p(v) = -C$ for every vertex $v \in \{s\} \cup V_1$ and $p(v) = 0$ for every vertex $v \in V_2 \cup \{t\}$. We initialize f with the zero circulation. Finally we call the `refine` method of the blocking flow algorithm from Section 3. Lemma 14 shows that this call to `refine` performs at most $O(\sqrt{r})$ blocking flow computations. It was first proved for a balanced bipartite graph by Duan at al. [6]. It follows that this stage runs in $O(\sqrt{r}m)$ time.

▶ **Lemma 14.** *The first stage performs at most $\sqrt{r} + 2$ blocking flow computations.*

Proof. Consider the initial potentials. Arcs outgoing from s or incoming into t have reduced cost 0. Every arc $(v, w) \in V_1 \times V_2$ has $c_p(v, w) \geq -C + C + 0 \geq 0$. The arc (t, s) has $c_p(t, s) = -C$. It follows that $E_A = \{(t, s)\}$. So after saturating admissible arcs we have $e_f(s) > 0$, $e_f(t) < 0$ and $E^+ = \{(s, t)\}$. Since t is the only deficit we have $p(t) = 0$. After y blocking flows we have that $p(s) = -C + y\frac{C}{\sqrt{r}}$. It follows that $c_p(s, t) = -p(s)$ is nonnegative for $y \leq \sqrt{r}$ and is negative for $y = \sqrt{r} + 1$. So (s, t) is admissible after $\sqrt{r} + 1$ blocking flows and the $(\sqrt{r} + 2)$'th blocking flow drains all remaining excess from s through the arc (s, t). ◀

For the finish-up stage we run the last cost scale of the SSSP algorithm from Section 5.1, except that whenever we encounter an admissible (negative) cycle we do not terminate. Instead we cancel the cycle and start a new iteration of `refine`. Consider a negative cycle. By integrality, its cost is -1 or less, so by ϵ-optimality (recall that $\epsilon \leq 1/\sqrt{r}$ at this point) the cycle must contain at least $\sqrt{r}$ arcs with negative reduced cost. This observation implies that canceling a negative cycle saturates at least $\sqrt{r}$ bad arcs. It follows that the number of time we restart an iteration of `refine` is $O(\sqrt{r})$.

5.4 Minimum Cost Flow with Unit Vertex Capacities

We show an $O(\sqrt{n}m \log(nC))$ time bound on the MCF problem with unit vertex capacities using our pseudoflow framework. We consider the formulation of the problem on networks where every vertex has an out-degree of 1 or an in-degree of 1. The time bound is proved in the same way as in Section 3 except we let $\lambda = \sqrt{n}$ and we let Lemma 15 replace Lemma 5.

[8] To simplify the presentation we assume that $\lambda/12$ is an integer.

▶ **Lemma 15.** *When the second phase starts there is $O(\sqrt{n})$ excess left.*

Proof. During the second phase every excess v has $d(v) \geq \lambda$ and every deficit v has $d(v) = 0$. Consider the $\frac{1}{3}\lambda$ sets of vertices $V_i = \{v \in V \mid d(v) \in \{3i, 3i-1, 3i-2\}\}$ where $1 \leq i \leq \frac{\lambda}{3}$. Also consider $A_i = \{(v, w) \in E^+ \mid (v \in V_i \text{ and has out-degree 1}) \text{ or } (w \in V_i \text{ and has in-degree 1}) \}$. By Lemma 3 for any set V_i, every E^+ excess to deficit path must contain a vertex in V_i. It follows that every A_i is an excess-deficit cut in G_f. By the pigeon hole principle there must be a set V_i such that $|V_i| \leq \frac{n}{\lambda/3} = 3\sqrt{n} = 3\lambda$ and therefore $|A_i| \leq 3\sqrt{n}$. So by Lemma 4 the remaining excess is at most 3λ. ◀

Acknowledgements We thank an anonymous reviewer for a simplification of the algorithm of Section 4.

References

1. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, 1993.
2. R. K. Ahuja, J. B. Orlin, C. Stein, and R. E. Tarjan. Improved Algorithms for Bipartite Network Flow. *SIAM J. Comput.*, 23:906–933, 1994.
3. R. G. Bland and D. L. Jensen. On the Computational Behavior of a Polynomial-Time Network Flow Algorithm. *Math. Prog.*, 54:1–41, 1992.
4. R. B. Dial. Algorithm 360: Shortest Path Forest with Topological Ordering. *Comm. ACM*, 12:632–633, 1969.
5. E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numer. Math.*, 1:269–271, 1959.
6. Ran Duan, Seth Pettie, and Hsin-Hao Su. Scaling algorithms for approximate and exact maximum weight matching. *CoRR*, abs/1112.0790, 2011.
7. S. Even and R. E. Tarjan. Network Flow and Testing Graph Connectivity. *SIAM J. Comput.*, 4:507–518, 1975.
8. H. N. Gabow and R. E. Tarjan. Faster Scaling Algorithms for Network Problems. *SIAM J. Comput.*, 18:1013–1036, 1989.
9. A. V. Goldberg. Scaling Algorithms for the Shortest Paths Problem. *SIAM J. Comput.*, 24:494–504, 1995.
10. A. V. Goldberg and R. E. Tarjan. Finding Minimum-Cost Circulations by Canceling Negative Cycles. *J. Assoc. Comput. Mach.*, 36:873–886, 1989.
11. A. V. Goldberg and R. E. Tarjan. Finding Minimum-Cost Circulations by Successive Approximation. *Math. of Oper. Res.*, 15:430–466, 1990.
12. J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ Algorithm for Maximum Matching in Bipartite Graphs. *SIAM J. Comput.*, 2:225–231, 1973.
13. A. V. Karzanov. O nakhozhdenii maksimal'nogo potoka v setyakh spetsial'nogo vida i nekotorykh prilozheniyakh. In *Matematicheskie Voprosy Upravleniya Proizvodstvom*, volume 5. Moscow State University Press, Moscow, 1973. In Russian; title translation: On Finding Maximum Flows in Networks with Special Structure and Some Applications.
14. Jr. L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton Univ. Press, Princeton, NJ, 1962.
15. A. Madry. "navigating central path with electrical flows: From flows to matchings, and back". In *FOCS*, pages 253–262, 2013.
16. L. Ramshaw and R.Endre Tarjan. "a weight-scaling algorithm for min-cost imperfect matchings in bipartite graphs". In *FOCS*, pages 581–590, 2012.

17 H. Röck. Scaling Techniques for Minimal Cost Network Flows. In U. Pape, editor, *Discrete Structures and Algorithms*, pages 181–191. Carl Hansen, Münich, 1980.