Minimum Weight Cycles and Triangles: Equivalences and Algorithms

Liam Roditty
Department of Computer Science
Bar Ilan University
Ramat Gan 52900, Israel
Email: liamr@macs.biu.ac.il

Virginia Vassilevska Williams
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720, USA
Email: virgi@eecs.berkeley.edu

Abstract— We consider the fundamental algorithmic problem of finding a cycle of minimum weight in a weighted graph. In particular, we show that the minimum weight cycle problem in an undirected n-node graph with edge weights in $\{1,\ldots,M\}$ or in a directed n-node graph with edge weights in $\{-M,\ldots,M\}$ and no negative cycles can be efficiently reduced to finding a minimum weight *triangle* in an $\Theta(n)-$node *undirected* graph with weights in $\{1,\ldots,O(M)\}$. Roughly speaking, our reductions imply the following surprising phenomenon: a minimum cycle with an arbitrary number of weighted edges can be "encoded" using only *three* edges within roughly the same weight interval!

This resolves a longstanding open problem posed in a seminal work by Itai and Rodeh [SIAM J. Computing 1978] on minimum cycle in unweighted graphs.

A direct consequence of our efficient reductions are $\tilde{O}(Mn^{\omega}) \leq \tilde{O}(Mn^{2.376})$-time algorithms using fast matrix multiplication (FMM) for finding a minimum weight cycle in both undirected graphs with integral weights from the interval $[1,M]$ and directed graphs with integral weights from the interval $[-M,M]$. The latter seems to reveal a strong separation between the all pairs shortest paths (APSP) problem and the minimum weight cycle problem in directed graphs as the fastest known APSP algorithm has a running time of $O(M^{0.681}n^{2.575})$ by Zwick [J. ACM 2002].

In contrast, when only combinatorial algorithms are allowed (that is, without FMM) the only known solution to minimum weight cycle is by computing APSP. Interestingly, any separation between the two problems in this case would be an amazing breakthrough as by a recent paper by Vassilevska W. and Williams [FOCS'10], any $O(n^{3-\varepsilon})$-time algorithm ($\varepsilon > 0$) for minimum weight cycle immediately implies a $O(n^{3-\delta})$-time algorithm ($\delta > 0$) for APSP.

Keywords-minimum cycle, girth, triangle, equivalence, reduction, matrix multiplication

1. Introduction

We consider the algorithmic problem of finding a cycle of minimum weight in weighted directed and undirected graphs. Surprisingly, although the problem is very fundamental, the state of the art for it dates back to a seminal paper by Itai and Rodeh [10] from the 1970s, that deals only with the *unweighted* variant of the problem. Itai and Rodeh presented an $O(n^{\omega})$-time algorithm for an n-node unweighted undirected graph and an $O(n^{\omega}\log n)$-time algorithm for an n-node unweighted directed graph. (Here ω is the exponent of square matrix multiplication over a ring; $\omega < 2.376$ [4].) In the same paper, Itai and Rodeh posed the question whether similar results exist for weighted graphs. In this paper we provide a positive answer to this longstanding open problem by presenting $\tilde{O}(Mn^{\omega})$-time algorithms for directed graphs with integral edge weights in $[-M,M]$ (and no negative cycles) and for undirected graphs with integral edge weights in $[1,M]$.

Our algorithmic results are obtained using new reductions that carefully combine new algorithmic ideas and special combinatorial properties of the minimum weight cycle. More specifically, we reduce the problem to finding a minimum weight *triangle* in a $\Theta(n)-$node *undirected* graph with weights in $\{1,\ldots,O(M)\}$. This reveals also a surprising phenomenon: a minimum cycle with an arbitrary number of weighted edges can be efficiently "encoded" using a cycle of only *three* edges whose weights are roughly within the same interval! Moreover, our results imply a strong *equivalence* between the cycle and triangle problems.

Minimum cycle and APSP: Recently, Vassilevska W. and Williams [19] showed that the minimum weight cycle problem is equivalent to many other graph and matrix problems for which no truly subcubic ($O(n^{3-\varepsilon})$-time for constant $\varepsilon > 0$) algorithms are known. They showed that if there is a truly subcubic algorithm for the minimum weight cycle problem, then many other problems such as the all pairs shortest paths (APSP) problem also have truly subcubic algorithms. Hence, the minimum weight cycle problem has a pivotal role in understanding the complexity of many fundamental polynomial problems in a similar spirit to the role of 3SAT for NP-hard problems.

It is interesting to compare the minimum cycle

problem with APSP. In directed graphs, the minimum weight cycle can be computed easily by computing APSP. Given the distance $d[u,v]$ between all pairs of vertices u, v, the weight of the minimum cycle is $\min_{u,v} w(u,v) + d[v,u]$. Hence, we can compute the minimum weight cycle in cubic time using Floyd-Warshall's APSP algorithm [6], [18] (or Pettie's $O(mn + n^2 \log\log n)$ time algorithm [12] if the graph has m edges). If the edge weights are integers in $[-M, M]$, we can use Zwick's [22] $O(M^{0.681}n^{2.575})$ time algorithm to obtain an algorithm for minimum cycle with the same runtime. Improving Zwick's running time, and in particular obtaining an $\tilde{O}(Mn^\omega)$ running time for APSP in directed graphs, is one of today's frontier questions in graph algorithms. Our new $\tilde{O}(Mn^\omega)$-time algorithm for minimum cycle in directed graphs shows that it is not really necessary to compute all pairs shortest paths in order to compute the minimum weight cycle in directed graphs. This seems to reveal a strong separation between APSP and the minimum cycle problem in directed graphs.

The minimum cycle problem in undirected graphs differs from the problem in directed graphs in that the reduction to APSP no longer works: an edge (u,v) might also be the shortest path from v to u, and $\min_{u,v} w(u,v) + d[v,u]$ might be $2w(u,v)$ and not the weighted girth of the graph. This represents a nontrivial hurdle. Nevertheless, in this paper we show how to overcome this hurdle and obtain an $\tilde{O}(Mn^\omega)$ time algorithm for undirected graphs with integer weights in $[1, M]$. This matches the runtime of the best APSP algorithm in such graphs: In 1992, Seidel [16] showed that APSP in undirected, unweighted n-node graphs can be solved in $\tilde{O}(n^\omega)$ time. Later in 1999, Shoshan and Zwick [17] (following Galil and Margalit [8]) showed that APSP in undirected n-node graphs with integer edge weights in $[0, M]$ can be solved in $\tilde{O}(Mn^\omega)$ time, thus extending Seidel's running time to weighted undirected graphs.

Our results: reductions and algorithms: We develop our algorithms by first obtaining extremely efficient reductions from the minimum weight cycle problem to the minimum weight triangle problem which preserve the interval in which the weights lie, within a constant factor.

Undirected graphs. Our results are as follows.

Theorem 1. *Let $G(V,E,w)$ be an undirected graph with $w : E \to \{1,\ldots,M\}$ and let C be a minimum cycle in G. There is an $O(n^2(\log nM)\log n)$ time deterministic algorithm that computes a cycle $\hat{C}$ and constructs $O(\log n)$ graphs $G'_1,\ldots,G'_k$ on $\Theta(n)$ nodes and edge weights in $\{1,\ldots,O(M)\}$ such that either $w(\hat{C}) = w(C)$ or the minimum out of all weights of triangles in the graphs G'_i is exactly $w(C)$.*

Since a minimum weight triangle in a graph with weights bounded by $O(M)$ can be found via a single distance product computation in $\tilde{O}(Mn^\omega)$ time [1], [21], we obtain the following corollary.

Corollary 1. *A minimum weight cycle in an n-node undirected graph with integer edge weights in $[1, M]$ can be found in $\tilde{O}(Mn^\omega)$ time.*

Directed graphs. Our reduction for undirected graphs relies on the fact that distances are symmetric. It is unlikely that it is possible to modify the reduction so that it works for directed graphs as well. Hence, for directed graphs new ideas are required. The reduction to minimum triangle is not as efficient, however, the resulting algorithm for minimum cycle in directed graphs has the same running time as the one for undirected graphs with nonnegative weights. Our approach for directed graphs can be combined with our approach for undirected graphs to yield an efficient algorithm also for *mixed* graphs, that is, graphs which contain both directed and undirected edges. The approach works, provided the weights of the mixed graph are nonnegative. (The details for mixed graphs are in the full version of the paper [14].)

When negative edge weights are allowed, a negative cycle may exist. Finding a minimum weight cycle when its weight is negative is an NP-hard problem, as it solves Hamiltonian cycle. When negative weights are allowed, the minimum cycle problem in the absence of negative cycles is in P for both directed and undirected graphs, but is NP-hard for mixed graphs [3]. Our techniques for directed graphs are strong enough to support negative edge weights within the same running time as when the weights are nonnegative. This is extremely interesting, as the typical way to reduce the general problem to the nonnegative weights problem involves computing node *potentials* (see e.g. [9]). These potentials however typically increase the magnitude of the weights to $\Omega(Mn)$, which would be bad if our goal is to use algorithms that have exponential dependence on the bit representation of the weights, such as $\tilde{O}(Mn^\omega)$. We circumvent the potential approach by focusing on the general problem directly. We obtain:

Theorem 2. *Let $G(V,E,w)$ be a directed graph on n nodes, $w : E \to \{-M,\ldots,M\}$. In $\tilde{O}(Mn^\omega)$ time one can construct $O(\log n)$ graphs $G'_1,\ldots,G'_k$ on $\Theta(n)$ nodes and edge weights in $\{1,\ldots,O(M)\}$ so that the*

minimum out of all weights of triangles in the graphs G'_i is exactly the weighted girth of G.

Our results: equivalences: Vassilevska W. and Williams [19] showed that the minimum triangle and minimum cycle problems are equivalent, under subcubic reductions. Their reduction from minimum triangle to minimum cycle only increased the number of nodes and the size of the edge weights by a constant factor. However, their reduction from minimum cycle to minimum triangle was not tight; it only proved that an $O(n^{3-\varepsilon})$ algorithm for minimum triangle would imply an $O(n^{3-\varepsilon/3})$ algorithm for minimum cycle. Our reductions, on the other hand, imply a much stronger equivalence between the two problems. This equivalence is especially strong for undirected graphs with integral weights from the range $[1, M]$.

Corollary 2. *If there is a $T(n, M)$ time algorithm for the minimum cycle problem in undirected graphs with integral edge weights in $[1, M]$, then there is a $T(O(n), O(M)) + O(n^2)$ time algorithm for the minimum triangle problem in such graphs. Conversely, if there is a $T(n, M)$ time algorithm for the minimum triangle problem in undirected graphs with integral edge weights in $[1, M]$, then there is an $O(T(O(n), O(M)) \log n + n^2 \log n \log nM)$ time algorithm for the minimum cycle problem in such graphs.*

A natural question is whether the triangle problem is special. Do similar reductions exist between minimum cycle and minimum k-cycle for constant $k > 3$? We answer this in the affirmative. (The proof of Theorem 3 below is in the full version of the paper [14].)

Theorem 3. *Let k be a fixed constant. Let $G(V, E, w)$ be an n-node graph, $w : E \to \{1, \ldots, M\}$. One can construct $O(\log n)$ undirected graphs $G'_1, \ldots, G'_\ell$ on $\Theta(kn)$ nodes and edge weights in $\{1, \ldots, O(M)\}$ so that the minimum out of all weights of k-cycles in the graphs G'_i is the weighted girth of G. Moreover, given the minimum weight k-cycle of the G'_i, one can find a minimum weight cycle of G in $O(n)$ additional time. If G is directed, the reduction runs in $\tilde{O}(Mn^\omega)$ time, and if G is undirected, it runs in $\tilde{O}(n^2 \log nM)$ time.*

Our results: approximation: Another approach to gain efficiency for problems with seemingly no subcubic time exact algorithms has been to develop fast approximation algorithms (see [5], [22] in the context of shortest paths). Lingas and Lundell [11] gave two approximation algorithms for minimum cycle: an $\tilde{O}(n^{1.5})$ time $8/3$-approximation for undirected unweighted graphs, and an $O(n^2(\log n) \log nM)$ time 2-approximation for undirected graphs with weights in $\{1, \ldots, M\}$. Very recently, Roditty and Tov [13] improved the approximation factor to $4/3$-approximation for the weighted case while keeping the running time unchanged. Due to Zwick's [22] $\tilde{O}(n^\omega/\varepsilon \log(M/\varepsilon))$ time $(1+\varepsilon)$-approximation for APSP and the simple reduction from minimum weight cycle in directed graphs to APSP, the girth of a directed graph admits a $(1+\varepsilon)$-approximation in $\tilde{O}(n^\omega/\varepsilon \log(M/\varepsilon))$ time. Our reduction from Theorem 1 implies the same result for undirected graphs with nonnegative weights as well, following up on the work of Roditty and Tov [13].

Theorem 4. *There is an $\tilde{O}(n^\omega/\varepsilon \log(M/\varepsilon))$ time $(1+\varepsilon)$-approximation algorithm for the minimum cycle in undirected graphs with integral weights in $[1, M]$.*

2. Preliminaries

Let $G(V, E, w)$ be a weighted graph, where V is its set of *vertices* or *nodes* (we use these terms interchangeably), $E \subseteq V \times V$ is its set of edges, and $w : E \to \{1, \ldots, M\}$ is a weight function. The function $w(\cdot, \cdot)$ can be extended to the entire $V \times V$ by setting $w(u, v) = \infty$ for every $(u, v) \notin E$. Unless otherwise noted, n refers to the number of nodes in the graph.

An edge can be directed or undirected. An *undirected* graph is a graph with undirected edges only and a directed graph is a graph with directed edges only. A graph is *simple* if it does not contain self loops or multiple copies of the same edge. In a directed graph, two directed edges (x, y) and (y, x) in opposite directions are allowed since they are considered distinct. All graphs considered in this paper are simple.

We define a cycle C in a graph $G(V, E, w)$ to be an ordered set of vertices $\{v_1, v_2, \ldots, v_\ell\}$, such that $(v_i, v_{i+1}) \in E$ for every $i < \ell$ and $(v_\ell, v_1) \in E$. Let $w(C)$ be the sum of the weights of the edges of C and let $w_{\max}(C)$ be the weight of the heaviest edge. We denote with $d_C[v_i, v_j]$ the weight of the path that traverses the cycle from v_i to v_j by passing from v_i to v_{i+1} and so on. In the case that $j < i$ we traverse from v_ℓ to v_1 and continue until we reach v_i. Let $n(C)$ denote the number of vertices/edges in C. A cycle C is *simple* if no node or edge appears twice in C. With this definition, an undirected graph cannot have a simple cycle on 2 nodes, whereas directed graphs can, provided the two cycle edges are in opposite directions.

3. Our approach

Our reductions are based on a combinatorial property of cycles in weighted directed, undirected and mixed graphs that might be of independent interest. This

property is extremely useful as it shows that crucial portions of the minimum weight cycle are shortest paths. We present this property in the following lemma.

Lemma 1 (Critical edge)**.** *Let $G(V, E, w)$ be a weighted graph, where $w : E \to \mathbb{R}$, and assume that G does not contain a negative cycle. Let $C = \{v_1, v_2, \ldots, v_\ell\}$ be a cycle in G of weight $w(C) \geq 0$ and let $s \in C$. There exists an edge (v_i, v_{i+1}) on C such that $\lceil w(C)/2 \rceil - w(v_i, v_{i+1}) \leq d_C[s, v_i] \leq \lfloor w(C)/2 \rfloor$ and $\lceil w(C)/2 \rceil - w(v_i, v_{i+1}) \leq d_C[v_{i+1}, s] \leq \lfloor w(C)/2 \rfloor$. Furthermore, if C is a minimum weight cycle in G then $d[s, v_i] = d_C[s, v_i]$ and $d[v_{i+1}, s] = d_C[v_{i+1}, s]$.*

Proof: We can assume, wlog, that $s = v_1$. We start to traverse along C from v_1 until we reach the first edge (v_i, v_{i+1}) that satisfies $d_C[v_1, v_i] \leq \lfloor w(C)/2 \rfloor$ and $d_C[v_1, v_i] + w(v_i, v_{i+1}) \geq \lceil w(C)/2 \rceil$. Since $d_C[v_1, v_1] = 0 \leq \lfloor w(C)/2 \rfloor$ either we find an edge (v_i, v_{i+1}) that satisfies the requirement, where $i < \ell$ or we reach v_ℓ without finding such an edge. In the latter case $d_C[v_1, v_\ell] \leq \lfloor w(C)/2 \rfloor$ and since $d_C[v_1, v_\ell] + w(v_\ell, v_1) = w(C) \geq \lceil w(C)/2 \rceil$ the edge (v_ℓ, v_1) satisfies the requirement.

It follows immediately from the properties of the edge (v_i, v_{i+1}) that $d_C[v_1, v_i] \geq \lceil w(C)/2 \rceil - w(v_i, v_{i+1})$ and hence we get that $\lceil w(C)/2 \rceil - w(v_i, v_{i+1}) \leq d_C[v_1, v_i] \leq \lfloor w(C)/2 \rfloor$ as required.

We now bound $d_C[v_{i+1}, v_1]$. Recall, $d_C[v_{i+1}, v_1] = w(C) - (d_C[v_1, v_i] + w(v_i, v_{i+1}))$. Since $d_C[v_1, v_i] + w(v_i, v_{i+1}) \geq \lceil w(C)/2 \rceil$ we get that $d_C[v_{i+1}, v_1] \leq \lfloor w(C)/2 \rfloor$. Also, since $d_C[v_1, v_i] \leq \lfloor w(C)/2 \rfloor$ we get that $d_C[v_{i+1}, v_1] \geq \lceil w(C)/2 \rceil - w(v_i, v_{i+1})$.

It remains to show that if C is a minimum weight cycle, then $d[v_1, v_i] = d_C[v_1, v_i]$ and $d[v_{i+1}, v_1] = d_C[v_{i+1}, v_1]$. If G is a directed graph, then it is straightforward to see that the minimality of C implies that $d_C[u, v] = d[u, v]$ for every $u, v \in C$ and in particular $d[v_1, v_i] = d_C[v_1, v_i]$ and $d[v_{i+1}, v_1] = d_C[v_{i+1}, v_1]$ as required. Thus, we only need to consider the case that G is an undirected graph. From the first part of the proof we know that $d_C[v_{i+1}, v_1] \leq \lfloor w(C)/2 \rfloor$. If $d[v_{i+1}, v_1] < d_C[v_{i+1}, v_1]$, then let P be the path from v_{i+1} to v_1 of weight $d[v_{i+1}, v_1]$ and let C_2 be the portion of C from v_1 to v_{i+1}. The union of P and C_2 is a walk in G whose weight is strictly less than $w(C)$. Furthermore, since $d[v_{i+1}, v_1] < d_C[v_{i+1}, v_1] \leq \lfloor w(C)/2 \rfloor \leq w(C_2)$, P and C_2 must differ by at least one edge and hence $P \cup C_2$ contains some simple cycle of weight less than $w(C)$, a contradiction to the minimality of C. The argument for showing that $d[v_1, v_i] = d_C[v_1, v_i]$ is symmetric. ∎

Lemma 1 shows that it is possible to decompose every cycle into three portions: a single edge of weight at most $O(M)$ and two pieces whose weight differs by at most $O(M)$, and which are shortest paths if the cycle is of minimum weight. This observation is crucial for our efficient reductions. Another important piece of Lemma 1 is that *every* vertex on the cycle has a critical edge. This is crucial in the directed graph case.

Armed with Lemma 1 we can describe the general framework of our approach. Suppose that we have some way to compute a function $D : V \times V \to \mathbb{R}$ that satisfies:

- For every $u, v \in V$, $d[u, v] \leq D[u, v]$
- There exists a vertex v on the minimum cycle C whose critical edge (x, y) endpoints satisfy $D[v, x] = d[v, x]$ and $D[y, v] = d[y, v]$.

In Section 4 we show how to compute a function D in $O(n^2 \log n \log Mn)$ time for undirected graphs with integral weights from $[1, M]$, and in Section 5 we show how to compute a function D in $\tilde{O}(Mn^\omega)$ time for directed graphs with integral weights from $[-M, M]$ and no negative cycles.

Now consider the following (multi-)graph $G'(V', E', w')$ where $V' = V^1 \cup V^2$ and V^1, V^2 are disjoint copies of V. For every $D[a, b]$ which was computed we place an edge between $a^1 \in V^1$ and $b^2 \in V^2$ and (for directed graphs) also one between $a^2 \in V^2$ and $b^1 \in V^1$. These edges get weight $D[a, b]$ and correspond to the two large portions of the minimum cycle. Further, for every edge (a, b) in G, we add an edge from $a^2 \in V^2$ to $b^2 \in V^2$ with weight $w(a, b)$, i.e. V^2 induces a copy of G; these edges correspond to the critical edge of the minimum cycle. In our reduction for directed graphs we further transform G' into a simple undirected tripartite graph.

Consider v, x, y from the second bullet above. By Lemma 1, $D[v, x] + w(x, y) + D[y, v] = d_C(v, x) + w(x, y) + d_C(y, v) = w(C)$. Hence G' will contain $\{v^1, x^2, y^2\}$ as a triangle of weight $w(C)$. Our reductions in the next two sections give transformations which ensure that every triangle in G' corresponds to a simple cycle in G and that $\{v^1, x^2, y^2\}$ is preserved as a triangle. Since the values $D[\cdot, \cdot]$ are upper bounds on the distances, $\{v^1, x^2, y^2\}$ is a minimum weight triangle in G'. The graph G' however can have really large weights; $D[\cdot, \cdot]$ can be as large as Mn in general. Thus our transformations also apply a weight reduction technique which reduces all edge weights to the interval $[-O(M), O(M)]$. This technique is different for undirected and directed graphs.

Finding a minimum cycle: Our reductions show that the minimum cycle problem can be efficiently reduced to the minimum triangle problem in a different

graph with roughly the same number of nodes and weight sizes. Here we briefly discuss how one can actually find a minimum weight triangle in an n-node graph $G(V, E, w)$ with integral edge weights in the interval $[-M, M]$. With our reductions, this will give algorithms for the minimum cycle problem as well.

Let A be the $n \times n$ adjacency matrix of G defined as $A[i, j] = w(i, j)$ whenever $(i, j) \in E$ and $A[i, j] = \infty$ otherwise. A well known approach to finding a minimum weight triangle mimics Itai and Rodeh's algorithm for unweighted triangle finding [10]. It first computes the distance product of A with itself, $(A \star A)[i, j] = \min_k A[i, k] + A[k, j]$, to find for every pair of nodes i, j the minimum length of a path with at most 2 edges between them. Then, the weight of a minimum triangle is exactly $\min_{i,j} A[j, i]+(A\star A)[i, j]$. Finding the actual triangle takes only $O(n)$ time after one finds i, j minimizing the above expression. Thus the running time is dominated by the time for computing $A \star A$. The algorithm of Alon, Galil and Margalit [1] (following Yuval [21]) does this in $\tilde{O}(Mn^\omega)$ time, whenever the entries of A are in $\{-M, \ldots, M\}$. Hence a minimum triangle, can be found in $\tilde{O}(Mn^\omega)$ time.

4. Minimum weight cycle in undirected graphs with weights in $\{1, \ldots, M\}$

Let $G(V, E, w)$ be an undirected graph with integral edge weights from the range $[1, M]$. In this section we show that in $\tilde{O}(n^2 \log Mn)$ time we can compute a cycle whose weight is at most twice the weight of the minimum weight cycle and a new undirected graph $G'(V', E', w')$ with integral weights from the range $[-M, M]$ whose minimum triangle if exists corresponds to the minimum weight cycle in G, with constant probability. (To boost the probability of success, we actually create $O(\log n)$ graphs G'.) If G' does not have a triangle then the cycle that we have computed is the minimum weight cycle of G. We start by presenting an $\tilde{O}(n^2)$ time algorithm that given an integer t either reports a cycle of length $2t$ or computes all distances up to t. The computed distances are used to form G'.

Cycle or Distance Computation: The algorithm works in iterations and in each iteration it repeats the same procedure from a new vertex of the graph. This procedure is a simple adaptation of Dijkstra's algorithm. The input in each iteration is a source vertex s and an integer value t. The algorithm either reports a cycle of length at most $2t$ or computes the distances from s to every vertex that is within distance t from s. Lingas and Lundell [11] used a similar approach in order to compute a 2-approximation of the minimum weight cycle. Their algorithm, however, either returns a cycle of length at most $2t$ or computes the distances from s to every vertex that is within distance $2t$ from s. This small difference between the two algorithms is crucial for our needs. Pseudocode is given in Algorithm 1. The algorithm repeats the procedure Cycle? n times, each time with a different vertex. Every run of Cycle? takes at most $O(n \log n)$ time since it stops with the first cycle it detects, and thus the algorithm runtime is $O(n^2 \log n)$.

In the next Lemma we prove an important property of the algorithm.

Lemma 2. *For any integer t, Min-Cycle(G, t) either finds a cycle of weight at most $2t$, or computes all distances of length at most t.*

Proof: A cycle is reported when a vertex u is extracted from the priority queue Q and for one of its edges (u, v) that is being relaxed the value of $d[v]$ before the relaxation is not infinity. As any distance and any distance estimation are at most t, if a cycle is reported it must be of length at most $2t$. If a cycle is not reported, then our algorithm is almost identical to Dijkstra's algorithm. The only difference is that our algorithm relaxes an edge (u, v) when u is extracted from the priority queue if and only if $d[u]+w(u, v) \leq t$, while Dijkstra's algorithm relaxes all edges of u with no restriction. This implies that our algorithm computes all distances that are smaller or equal t. ■

Algorithm 1: Min-Cycle(G, t)

```
foreach s ∈ V do
    C' ← Cycle?(G, s, 2t);
    if w(C') < w(C*) then C* ← C'
return C*
```

Algorithm 2: Cycle?(G, s, t')

```
foreach v ∈ V do d[v] ← ∞;
d[s] = 0;
Q ← {s};
while Q ≠ ∅ do
    u ← Extract-Min(Q);
    Controlled-Relax(u, t'/2);
```

The reduction to minimum triangle: Our goal is to prove Theorem 1. The main part of the proof is describing an algorithm that computes an upper bound for the minimum weight cycle and an instance G' of minimum triangle, such that either the girth of the graph is exactly the upper bound, or with constant probability the minimum triangle weight in G' is the girth of G. Below we only present G' as having large weights. Later

Algorithm 3: Controlled-Relax(u, w_u)

```
(u, v) ← Extract-Min(Q_u);
while d[u] + w(u, v) ≤ w_u do
    if d[v] ≠ ∞ then
        report a cycle and stop;
    else
        Relax(u, v);
    (u, v) ← Extract-Min(Q_u);
```

on, we find a value t with which we use Lemma 2, so that $2t$ is a bound on the minimum cycle weight that is tight within $O(M)$. As mentioned in Section 3, this value allows us to reduce the weights of G' so that they fall in the interval $[-O(M), O(M)]$.

Reminder of Theorem 1 *Let $G(V, E, w)$ be an undirected graph with $w : E \to \{1, \ldots, M\}$ and let C be a minimum cycle in G. There is an $O(n^2(\log nM)\log n)$ time deterministic algorithm that computes a cycle $\hat{C}$ and constructs $O(\log n)$ graphs $G'_1, \ldots, G'_k$ on $\Theta(n)$ nodes and edge weights in $\{1, \ldots, O(M)\}$ such that either $w(\hat{C}) = w(C)$ or the minimum out of all weights of triangles in the graphs G'_i is exactly $w(C)$.*

The weight of the minimum cycle is an integer value from the range $[1, nM]$. From Lemma 2 it follows that we can use algorithm Min-Cycle to perform a binary search over this range in order to find the largest value $t \in [1, nM]$ for which Min-Cycle(G, t) does not report a cycle but computes all distances of length at most t. This implies that by running Min-Cycle$(G, t + 1)$ we obtain a cycle of weight at most $2t+2$. Hence, we only need to show that it is possible to detect the minimum cycle in the case that its weight $w(C)$ is $2t+1$ or less. Let us first prove some consequences of the fact that Min-Cycle(G, t) does not report a cycle.

Lemma 3. *Let $C = \{v_1, v_2, \ldots, v_\ell\}$ be a minimum cycle in $G(V, E, w)$. Suppose that Min-Cycle(G, t) does not report a cycle. There are three **distinct** vertices $v_i, v_{i+1}, v_j \in C$ such that $d_C[v_j, v_i] + w(v_i, v_{i+1}) > t$ and $d_C[v_{i+1}, v_j] + w(v_i, v_{i+1}) > t$.*

Proof: Let (v_i, v_{i+1}) be the critical edge for v_1 given by Lemma 1. Assume first that $v_1 \neq v_i$ and $v_1 \neq v_{i+1}$. If either $d_C[v_1, v_i] + w(v_i, v_{i+1}) \leq t$ or $d_C[v_{i+1}, v_1] + w(v_i, v_{i+1}) \leq t$ then the edge $w(v_i, v_{i+1})$ is relaxed. Assume that we are in the case that $d_C[v_1, v_i] + w(v_i, v_{i+1}) \leq t$. Then after (v_i, v_{i+1}) is relaxed $d[v_{i+1}] \leq t$. If $d[v_{i+1}] < \infty$ before the relaxation of (v_i, v_{i+1}) the algorithm stops and reports a cycle. If $d[v_{i+1}] = \infty$ before the relaxation of (v_i, v_{i+1}) then a cycle will be detected as well but only when the edge (v_{i+2}, v_{i+1}) is relaxed. This edge must be relaxed since $d_C[v_{i+1}, v_1] \leq \lfloor w(C)/2 \rfloor \leq t$ which implies that v_{i+2} will be extracted and its edge (v_{i+2}, v_{i+1}) will satisfy the relaxation requirement and will be relaxed. We conclude that if either $d_C[v_1, v_i] + w(v_i, v_{i+1}) \leq t$ or $d_C[v_{i+1}, v_1] + w(v_i, v_{i+1}) \leq t$ then Min-Cycle(G, t) must report a cycle, giving a contradiction.

We now turn to the case that either $v_1 = v_i$ or $v_1 = v_{i+1}$. Assume wlog that $v_1 = v_i$, that is, $(v_i, v_{i+1}) = (v_1, v_2)$. From Lemma 1 we know that $w(v_1, v_2) \geq \lceil w(C)/2 \rceil$ and $d_C[v_2, v_1] \leq \lfloor w(C)/2 \rfloor$. We also know that there is at least one additional vertex v_ℓ between v_2 and v_1 on the cycle C. We now apply Lemma 1 on the vertex v_ℓ. It is easy to see that in that case the edge (v_1, v_2) will be the critical edge of v_ℓ as well. We now have three different vertices and the rest of this case is identical to the first case. ∎

As a first attempt, we create the new graph $G'(V', E', w')$ as follows. The vertex set V' contains two copies V^1 and V^2 of V. For $i = 1, 2$, let E^i be the set of edges with both endpoints in V^i. The set E^1 is empty and the set E^2 is E, that is, $(u^2, v^2) \in E^2$ if and only if $(u, v) \in E$. Let E^{12} be the set of edges with one endpoint in V^1 and one endpoint in V^2. Let $u^1 \in V^1$ and $v^2 \in V^2$. If the distance between u and v was computed by Min-Cycle(G, t) then we add an edge (u^1, v^2) to E^{12} with weight $d[u, v]$. We show that there is triangle in $G'(V', E', w')$ that corresponds to the minimum cycle of G and has the same weight.

Claim 5. *Let $C = \{v_1, v_2, \ldots, v_\ell\}$ be a minimum cycle in $G(V, E, w)$, $w(C) \leq 2t + 1$. There exists a triangle in $G'(V', E', w')$ on vertices of C of weight $w(C)$.*

Proof: Without loss of generality, let v_1 be the vertex v_j from Lemma 3, and let v_i and v_{i+1} be the other two vertices. From Lemma 3 we know that all three vertices are distinct and that $d_C[v_1, v_i] + w(v_i, v_{i+1}) > t$ and $d_C[v_{i+1}, v_1] + w(v_i, v_{i+1}) > t$. Combining this with the fact that C is a minimum cycle and $w(C) \leq 2t + 1$ we get that $d[v_1, v_i] = d_C[v_1, v_i] \leq t$ and $d[v_{i+1}, v_1] = d_C[v_{i+1}, v_1] \leq t$. When Cycle? is run from v_1 it computes $d[v_1, v_i]$ and $d[v_1, v_{i+1}]$. Hence, there must be a triangle of weight $w(C)$ in G' on the vertices v_1^1,v_i^2 and v_{i+1}^2. ∎

The claim above shows only one direction, that is, if there is a minimum cycle C of weight at most $2t+1$ in G then there is a corresponding triangle in G' on vertices $y^2, z^2 \in V^2$ and $x^1 \in V^1$, that correspond to vertices of C with the same weight. To complete the reduction we must show that there are no false positives: triangles in G' of smaller weight which do not correspond to

a minimum cycle of G. Unfortunately, this is not the case and G' may have such false triangles with smaller weight, as in the following example: Let $x, y, z \in V$. If there is a shortest path of length at most t from x to z whose last edge is (y, z) then there is a triangle in G'. To see that, notice that there are two different shortest paths one from x to z and one from x to y, both of length at most t. In such a case, the graph G' includes the edges (x^1, y^2) and (x^1, z^2) and together with the edge (y^2, z^2) they form a triangle. Moreover, such a triangle has the same structure as a valid triangle and might be of smaller weight. Thus, a triangle detection algorithm cannot distinguish between a valid and a false triangle. In what follows we first show that this is the only situation in which a false triangle is formed, and then show a construction that avoids such false triangles.

In the above pathological case the only reason that the triangle x^1, y^2, z^2 did not correspond to a cycle, was because we had two different paths P_1 and P_2 that both start in the same vertex and the last vertex of one of these paths was the vertex right before the last vertex of the other path. In the next lemma we show that this is the *only* bad case.

Lemma 4. *Let $x, y, z \in V$ be three distinct vertices. Let $P_1 = y \to y' \to \ldots \to x$ and $P_2 = x \to \ldots \to z' \to z$ be simple shortest paths between y and x and x and z respectively. Let $y' \neq z$ and $z' \neq y$ and let $(z, y) \in E$. Then, $P_1 \cup P_2 \cup \{(z, y)\}$ contains a simple cycle of weight at most $w(P_1) + w(P_2) + w(z, y)$.*

Proof: Let P_1^{-1} be P_1 with its edges reversed. Look at P_1^{-1} and P_2. There are two options. Either one path is a subpath of the other, or there is a node x' such that $x \to \ldots \to x'$ is a subpath of both, and x' is followed by q_1 in P_1^{-1} and by $q_2 \neq q_1$ in P_2.

Consider the first case. Wlog, P_2 is a subpath of P_1^{-1} (the other inclusion is symmetric). Since $y' \neq z$, the subpath between y and z on P_1 has at least 2 edges, and adding edge (z, y) produces a simple cycle of weight less than the sum of the two original path weights.

Consider the second case when x', q_1, q_2 exist as above. If there is some node between x' and y on P_1^{-1} which also appears in P_2 after x', then let q be the first such node. Then no node on P_2 between x' and q appears between x' and q in P_1^{-1}. The two disjoint simple paths between x' and q form a simple cycle on at least 3 nodes since $q_1 \neq q_2$. The weight of this cycle is less than the sum of the two original path weights.

Finally, suppose no such q exists. Then the subpaths of P_1^{-1} and P_2 between x' and y and x' and z share no vertices and hence adding edge (z, y) closes a simple cycle of weight at most $w(P_1) + w(P_2) + w(z, y)$. ■

Lemma 4 implies that our reduction to minimum triangle will work, provided that we can ensure that for every triangle x^1, y^2, z^2 in G', the last node z' before z on the shortest path from x to z in G is distinct from y. To do this, we use the color-coding method from the seminal paper of Alon, Yuster and Zwick [2]. The idea is as follows. Let $\{C_1, C_2\}$ be two distinct colors. Assign to every node of G one of these colors independently and uniformly at random. Fix four vertices y, y', z, z'. The probability that $\text{color}(y') = \text{color}(z') = C_1$ and $\text{color}(y) = \text{color}(z) = C_2$ is $1/2^4 = O(1)$.

Now we will modify $G'(V', E', w')$ from before. Recall that $V' = V^1 \cup V^2$. For every node x of G we add a copy x^1 to V^1, so that V^1 is a copy of V. Furthermore, if $\text{color}(x) = C_2$ we also add a copy x^2 to V^2. We now define the set of edges E'. Let E^{ij} be the set of edges between V^i and V^j, for $i, j \in \{1, 2\}$. The edge set E^{11} is empty, so $E' = E^{12} \cup E^{22}$. Let $x, z', z \in V$ such that (z', z) is the last edge of the shortest path from x to z. The sets E^{12} and E^{22} are defined as follows:

$$E^{12} = \{(x^1, z^2) \mid \text{color}(z') = C_1 \wedge \text{color}(z) = C_2\},$$

$$E^{22} = \{(x^2, z^2) \mid \text{color}(x) = C_2 \wedge \text{color}(z) = C_2\}.$$

The weight of an edge $(x^1, z^2) \in E^{12}$ is $d[x, z]$. The weight of an edge $(x^2, z^2) \in E^{22}$ is $w(x, z)$. We now prove that G' does not contain false triangles.

Lemma 5. *If $T = \{x, y, z\}$ is triangle in G' then there exists a simple cycle C in G such that $\{x, y, z\} \subseteq C$ and $w(C) \leq w(T)$.*

Proof: Any triangle in G' either have one vertex from V^1 and two vertices from V^2 or all three vertices from V^2. In the latter case the triangle is also in G so we focus in the former case, that is, $T = \{x^1, y^2, z^2\}$ is a triangle in G' such that $x^1 \in V^1$ and $y^2, z^2 \in V^2$. Let $x, y, z \in V$ be the vertices that correspond to x^1, y^2 and z^2 in G. Let y' (z') be the last vertex before y (z) on the shortest path P_1 (P_2) between x and y (z) in G. The fact that $(x^1, y^2) \in E^{12}$ and $(x^1, z^2) \in E^{12}$ implies that $\text{color}(y') = \text{color}(z') = C_1$ and $\text{color}(y) = \text{color}(z) = C_2$. Hence we get that $y' \neq z$ and $z' \neq y$. Combining this with the fact that $E^{22} \subseteq E$ we get that the paths P_1, P_2 and the edge (y, z) satisfy the requirements of Lemma 4, and there is a simple cycle of weight at most $w(P_1) + w(P_2) + w(y, z) = w(T)$ in G. ■

Now that we have shown that G' does not contain false triangles we prove that the minimum weight cycle in G corresponds to a triangle in G'. (This can be viewed as proving Claim 5 for the new G').

Claim 6. *Let $C = \{v_1, v_2, \ldots, v_\ell\}$ be a minimum cycle in $G(V, E, w)$. Assume that $w(C) \leq 2t + 1$. Then there exists a triangle in $G'(V', E', w')$ on vertices of C of weight $w(C)$, with constant probability.*

Proof: Without loss of generality, let v_1 be the vertex v_j from Lemma 3, and let v_i and v_{i+1} be the other two vertices. As in the proof of Claim 5, $d[v_1, v_i] = d_C[v_1, v_i] \leq t$ and $d[v_{i+1}, v_1] = d_C[v_{i+1}, v_1] \leq t$ and these values are computed by Min-Cycle(G, t) . The random coloring is successful when color$(v_{i-1}) = C_1$, color$(v_i) = C_2$, color$(v_{i+1}) = C_2$ and color$(v_{i+2}) = C_1$. The probability that this happens is $1/2^4 = O(1)$. The triangle $\{v_1^1, v_i^2, v_{i+1}^2\}$ is in G' exactly when the coloring is successful, and hence C is represented by that triangle in G' with constant probability. The weight of the triangle $\{v_1^1, v_i^2, v_{i+1}^2\}$ is $d[v_1, v_i] + w(v_i, v_{i+1}) + d[v_{i+1}, v_1] = w(C)$. ∎

Weight reduction: Currently, the maximum edge weight in G' can be as large as $\Omega(nM)$ as the weights of edges in E^{12} are distances in G. To complete the reduction, we show that it is possible to reweight the edges of G' without changing the minimum triangle so that the edge weights will be integers from $[-M, M]$.

The key idea is to use Lemma 3 in two different ways. As we previously mentioned, Lemma 3 implies that $d_C[v_j, v_i] \leq t$ and $d_C[v_{i+1}, v_j] \leq t$. Moreover, the bounds $d_C[v_j, v_i] + w(v_i, v_{i+1}) > t$ and $d_C[v_{i+1}, v_j] + w(v_i, v_{i+1}) > t$ imply that $d_C[v_j, v_i] > t - M$ and $d_C[v_{i+1}, v_j] > t - M$. Thus, we can remove from E^{12} every edge of weight strictly more than t and every edge of weight $t - M$ or smaller with no effect on the minimum triangle in G'. We now decrease the weights of all the edges that were left in E^{12} by t. The weight of every triangle in G' with a node from V^1 has decreased by exactly $2t$. Hence, the minimum triangle out of those with a node in V^1 remains the same. The weights of edges in E^{12} are now integers from the interval $[-M, 0]$, and the rest of the edge weights are still in $[1, M]$. If the minimum weight triangle in G' now has nodes only from V^2, then this triangle was also the minimum weight one in G' before the reweighting, and hence corresponds to a minimum weight cycle, with high probability. Otherwise, the minimum weight triangle in G' has a node from V^1. The minimum out of these triangles was also the minimum one among the triangles with a node in V^1 also before the reweighting. Hence it also corresponds to a minimum weight cycle, with high probability. This completes the description of our construction.

Derandomization: The reduction can be made deterministic, just as in the color-coding paper of Alon *et al.* [2], by using a k-perfect hash family, a family $F = \{f_1, \ldots, f_{|F|}\}$ of hash functions from $\{1, \ldots, n\}$ to $\{1, \ldots, k\}$ so that for every $V' \subset V$ with $|V'| = k$, there exists some i so that f_i maps the elements of V' to distinct colors. In our case, $k = 2$. By enumerating through the functions of F, and using each f_i in place of the random coloring, our reduction runs in $O(n^2(\log nM) \log n + |F|n^2)$ time, provided each f_i can be evaluated in constant time. Our reduction produces $O(|F|)$ instances of minimum triangle.

Schmidt and Siegel [15] (following Fredman, Komlos and Szemeredi [7]) gave an explicit construction of a k-perfect family in which each function is specified using $O(k) + 2 \log \log n$ bits. For our case of $k = 2$, the size of the family is therefore $O(\log^2 n)$. The value of each one of the hash functions on each specified element of V can be evaluated in $O(1)$ time. Alon, Yuster and Zwick [2], reduced the size of the hash family to $O(\log n)$. Using this family we can derandomize our reduction so that it runs in deterministic $O(n^2(\log nM) \log n)$ time.

5. Minimum cycle in directed graphs with weights in $\{-M, \ldots, M\}$

In this section we consider directed graphs graphs with possibly negative weights but no negative cycles. In contrast to the situation in undirected graphs it is relatively easy to reduce the minimum cycle problem in directed graphs to the problem of computing all pairs shortest paths. If D is the distance matrix of a directed graph then its minimum cycle has weight $\min_{i,j} D[i, j] + w(j, i)$. Hence, using Zwick's APSP algorithm [22] we can compute the minimum cycle in $O(M^{0.681} n^{2.575})$ time. In this section we show that the minimum cycle problem in directed graphs can be reduced to the problem of finding a minimum triangle in an undirected graph. This also implies that the minimum weight cycle in directed graphs can be computed in $\tilde{O}(Mn^\omega)$ time.

Similarly to before, our approach will be to compute upper bounds on the distances in the graph so that for some node s on the minimum cycle C and its critical edge (v_i, v_{i+1}) we obtain the exact distances $d[s, v_i] = d_C[s, v_i]$ and $d[v_{i+1}, s] = d_C[v_{i+1}, s]$.

Computing cycle distances: The Dijkstra-like approach in the previous section does not work for directed graphs. It also only applies when the edge weights are nonnegative. Here we utilize a new approach that allows us to reduce the minimum cycle problem in directed graphs with integral weights in the interval $[-M, M]$ to the minimum triangle problem in undirected graphs with weights in $[-M, M]$. Our result is more general than before. However this comes at a

cost: the reduction no longer takes nearly quadratic time, but consumes $\tilde{O}(Mn^\omega)$ time.

Our approach uses the fact that Lemma 1 applies for *every* vertex of a cycle, together with a result by Yuster and Zwick [20] given in Theorem 7 below.

Theorem 7 (Yuster and Zwick '05)**.** *Given an n-node directed graph with integral edge weights in the interval $[-M, M]$, in $\tilde{O}(Mn^\omega)$ time one can compute an $n \times n$ matrix D such that the i, j entry of the distance product $D \star D$ contains the distance between i and j.*

The matrix D can contain entries with values as large as $\Omega(Mn)$ and so $D \star D$ is not known to be computable in truly subcubic time, even when M is small. Nevertheless, the theorem applies to general graphs with positive or negative weights. It also gives an $\tilde{O}(Mn^\omega)$ time algorithm for detecting a negative cycle in a graph, and is extremely useful in computing minimum cycles.

The Yuster-Zwick algorithm proceeds in stages. In each stage ℓ, a node subset sample B_ℓ is maintained so that each node is in B_ℓ with probability at least $\min\{1, 9(2/3)^\ell \ln n\}$. They prove the following lemma.

Lemma 6 (Yuster and Zwick '05)**.** *For every stage ℓ and any node $s \in B_\ell$ and node $v \in V$, the algorithm has estimates $D[s, v]$ and $D[v, s]$, so that if the shortest path from s to v has at most $(3/2)^\ell$ edges then $D[s, v] = d[s, v]$, with high probability. Similarly, if the shortest path from v to s has at most $(3/2)^\ell$ edges then $D[v, s] = d[v, s]$ with high probability.*

The Yuster-Zwick algorithm also provides a matrix Π of *predecessors* so that if $k = \Pi[i, j]$, then k is the predecessor of j on a simple path from i to j of weight $D[i, j]$. Similarly, one can obtain a matrix Π' of *successors* so that if $k = \Pi'[i, j]$, then k is the successor of i on a simple path from i to j of weight $D[i, j]$.

Now, first use the algorithm to check whether the given graph has a negative cycle. If it does not, then let C be the minimum weight cycle, $w(C) \geq 0$. Recall that $n(C)$ is the number of vertices/edges on C. Let ℓ be the minimum value so that $n(C) \leq (3/2)^\ell$. Note that then $n(C) \geq (3/2)^{\ell-1}$. The probability that a particular node s of C is not in B_ℓ is at most $1 - (2/3)^\ell(9 \ln n)$. The events are independent for all s in C, and since $n(C) \geq (3/2)^{\ell-1}$, the probability that no node of C is in B_ℓ is at most $(1 - (2/3)^\ell(9 \ln n))^{(3/2)^{\ell-1}} \leq 1/n^6$.

Thus the probability that some node s of C is in B_ℓ is $1 -$ poly$^{-1}(n)$; furthermore by Lemma 6 (with high probability) for all $x \in C$, the Yuster-Zwick algorithm has computed $D[s, x] = d[s, x]$ and $D[x, s] = d[x, s]$, since the number of edges on the respective shortest paths are at most $n(C) \leq (3/2)^\ell$.

In particular, this means that $D[s, v_i] = d[s, v_i]$ and $D[v_{i+1}, s] = d[v_{i+1}, s]$ for the critical edge (v_i, v_{i+1}) for s on C as Lemma 1 applies for every vertex of C. Moreover, since C is a minimum weight cycle with $w(C) \geq 0$, by Lemma 1, the paths between s and v_i and v_{i+1} and s on C are both shortest paths. Thus, with high probability, $d_C[s, v_i] = d[s, v_i] = D[s, v_i]$ and $d_C[v_{i+1}, s] = d[v_{i+1}, s] = D[v_{i+1}, s]$.

Creating the minimum triangle instance G': G' will still be undirected, but unlike the construction for undirected graphs, G' will now be tripartite. The vertex set V' of G' has partitions V^1, V^2, V^3 which are all copies of V.

The construction is as follows. For every directed edge (u, v) of G, add an edge from $u^2 \in V^2$ to $v^3 \in V^3$ with weight $w(u, v)$. Furthermore, for every two nodes x, y so that $D[x, y] < \infty$ add an edge from $x^1 \in V^1$ to $y^2 \in V^2$ and one from $x^3 \in V^3$ to $y^1 \in V^1$, each with weight $D[x, y]$. Hence the edges between $x^1 \in V^1$ and $y^2 \in V^2$ correspond to directed paths from x to y, and the edges between $x^3 \in V^3$ and $y^1 \in V^1$ correspond to directed paths from y to x. Hence any triangle in G' corresponds to a directed closed walk in G. However, any such closed walk must contain a simple cycle of no larger weight: If the walk is not simple, find a closest pair of copies of a node v on the walk. These copies enclose a simple cycle C'' in G. Now, either C'' has no larger weight than the walk, or removing it from the walk produces a smaller closed walk of negative weight, and hence G contains a negative cycle, which we assumed is not the case. Since G' contains no false positives, the minimum triangle of G' corresponds exactly to C.

Weight reduction: As in the construction for undirected graphs, the maximum edge weight in G' can be as large as $\Omega(nM)$. Here we give a different way to reduce them to the interval $[-M, M]$.

Let t be a parameter that we will be changing. Intuitively, our goal will be to set t to roughly $\lfloor w(C)/2 \rfloor$; it will be sufficient for t to be $\leq \lfloor w(C)/2 \rfloor$.

Initially, $t = Mn$. Now, check whether there is a triangle $a^1 \in V^1$, $b^2 \in V^2$, $c^3 \in V^3$ in G' so that $D[a^1, b^2], D[c^3, a^1] \leq t$. We run a binary search on t in the interval $[0, Mn]$, until we find the smallest t such that there is such a triangle. Each search can be done using Boolean matrix product: create a matrix A which is 1 wherever D is $\leq t$ and 0 otherwise; multiply A by itself and check for a triangle closed by an edge (b^2, c^3), $b^2 \in V^2, c^3 \in V^3$. This takes $O(n^\omega \log w(C))$ time.

Let (whp) $\{s^1, v_i^2, v_{i+1}^3\}$ be the triangle in G'

that corresponds to the minimum cycle C of G. Since $\{s^1, v_i^2, v_{i+1}^3\}$ is a valid triangle, and $d_C[s, v_i], d_C[v_{i+1}, s] \leq \lfloor w(C)/2 \rfloor$ by Lemma 1, then after the completion of the binary search, $t \leq \lfloor w(C)/2 \rfloor$. Furthermore, since C is a minimum cycle and by the definition of t, $w(C) \leq 2t + w(e)$, where e is some edge in G, which implies that $w(C) \leq 2t + M$. Hence, $t \leq \lfloor w(C)/2 \rfloor$ and $w(C)/2 \leq t + M/2$.

Now, remove from G' every edge $(c^3, a^1) \in V^3 \times V^1$ with $D[c^3, a^1] > t + M/2$ and every $(a^1, b^2) \in V^1 \times V^2$ with $D[a^1, b^2] > t + M/2$. If an edge has weight $\leq \lfloor w(C)/2 \rfloor \leq t + M/2$, it is not removed. In particular, (s^1, v_i^2) and (v_{i+1}^3, s^1) are still edges, by Lemma 1.

Remove every $(c^3, a^1) \in V^3 \times V^1$ with $D[c^3, a^1] < t - M$ and every $(a^1, b^2) \in V^1 \times V^2$ with $D[a^1, b^2] < t - M$. If an edge has weight $\geq \lfloor w(C)/2 \rfloor - M \geq t - M$, then it is not removed. Hence again (s^1, v_i^2) and (v_{i+1}^3, s^1) are not removed because their weight is at least $t - M$ as follows from Lemma 1.

All remaining edges in $(V^3 \times V^1) \cup (V^1 \times V^2)$ have integral weights in $[t - M, t + M/2]$, and C is still represented by the minimum triangle $\{s^1, v_i^2, v_{i+1}^3\}$. Now, for every remaining edge $(a, b) \in (V^1 \times V^2) \cup (V^3 \times V^1)$, change its weight to $D[a, b] - t$. The weights of the edges of G' are now in the interval $[-M, M]$. Furthermore, since the weights of all triangles have decreased by $2t$, the minimum triangle of G' is still the same. This completes the construction of G'.

Derandomization: The only randomized part of our reduction is our use of Yuster and Zwick's result. Their algorithm can be derandomized (see [20]) without affecting our use of their result. Hence, we obtain a deterministic reduction which runs in $O(Mn^\omega \log n + n^\omega (\log Mn) \log n)$ time and does not increase the graph size or the edge weights by more than a constant factor.

Acknowledgments: The second author would like to thank Satish Rao and Ryan Williams for their valuable comments. The first author was supported by ISF grant no. 822/10. The second author was supported by NSF Grant #0963904 and NSF Grant #0937060 to the CRA for the CIFellows Project. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or the CRA.

REFERENCES

[1] N. Alon, Z. Galil, and O. Margalit, "On the exponent of the all pairs shortest path problem," *J. Comput. Syst. Sci.*, vol. 54, no. 2, pp. 255–262, 1997.

[2] N. Alon, R. Yuster, and U. Zwick, "Color-coding," *JACM*, vol. 42, no. 4, pp. 844–856, 1995.

[3] E. Arkin and C. H. Papadimitriou, "On negative cycles in mixed graphs," *Operations Research Letters*, vol. 4, no. 3, pp. 113–116, 1985.

[4] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," *J. Symbolic Computation*, vol. 9, no. 3, pp. 251–280, 1990.

[5] D. Dor, S. Halperin, and U. Zwick, "All-pairs almost shortest paths," *SIAM J. Comput.*, vol. 29, no. 5, pp. 1740–1759, 2000.

[6] R. W. Floyd, "Algorithm 97: shortest path," *Comm. ACM*, vol. 5, p. 345, 1962.

[7] M. Fredman, J. Komlós, and E. Szemerédi, "Storing a sparse table with $O(1)$ worst case access time," *JACM*, vol. 31, pp. 538–544, 1984.

[8] Z. Galil and O. Margalit, "All pairs shortest paths for graphs with small integer length edges," *JCSS*, vol. 54, pp. 243–254, 1997.

[9] A. Goldberg, "Scaling algorithms for the shortest paths problem," in *Proc. SODA*, 1993, pp. 222–231.

[10] A. Itai and M. Rodeh, "Finding a minimum circuit in a graph," *SIAM J. Computing*, vol. 7, no. 4, pp. 413–423, 1978.

[11] A. Lingas and E.-M. Lundell, "Efficient approximation algorithms for shortest cycles in undirected graphs," *Inf. Process. Lett.*, vol. 109, no. 10, pp. 493–498, 2009.

[12] S. Pettie, "A new approach to all-pairs shortest paths on real-weighted graphs," *Theor. Comput. Sci.*, vol. 312, no. 1, pp. 47–74, 2004.

[13] L. Roditty and R. Tov, "Approximating the girth," in *Proc. SODA*, 2011, pp. 1446–1454.

[14] L. Roditty and V. V. Williams, "Minimum weight cycles and triangles: Equivalences and algorithms," *CoRR*, vol. abs/1104.2882, 2011.

[15] J. Schmidt and A. Siegel, "The spatial complexity of oblivious k-probe hash functions," *SIAM J. Comput.*, vol. 19, no. 5, pp. 775–786, 1990.

[16] R. Seidel, "On the all-pairs-shortest-path problem in unweighted undirected graphs," *JCSS*, vol. 51, pp. 400–403, 1995.

[17] A. Shoshan and U. Zwick, "All pairs shortest paths in undirected graphs with integer weights," in *Proc. FOCS*, 1999, pp. 605–614.

[18] S. Warshall, "A theorem on Boolean matrices," *J. ACM*, vol. 9, no. 1, pp. 11–12, 1962.

[19] V. V. Williams and R. Williams, "Subcubic equivalences between path, matrix and triangle problems," in *Proc. FOCS*, 2010, pp. 645–654.

[20] R. Yuster and U. Zwick, "Answering distance queries in directed graphs using fast matrix multiplication," in *Proc. FOCS*, 2005, pp. 389–396.

[21] G. Yuval, "An algorithm for finding all shortest paths using $N^{2.81}$ infinite-precision multiplications," *Inf. Proc. Letters*, vol. 4, pp. 155–156, 1976.

[22] U. Zwick, "All pairs shortest paths using bridging sets and rectangular matrix multiplication," *JACM*, vol. 49, no. 3, pp. 289–317, 2002.