SIAM J. COMPUT.
Vol. 39, No. 2, pp. 431–444

FINDING A HEAVIEST VERTEX-WEIGHTED TRIANGLE IS NOT HARDER THAN MATRIX MULTIPLICATION*

ARTUR CZUMAJ† AND ANDRZEJ LINGAS‡

Abstract. We show that a maximum-weight triangle in an undirected graph with n vertices and real weights assigned to vertices can be found in time $\mathcal{O}(n^\omega + n^{2+o(1)})$, where ω is the exponent of the fastest matrix multiplication algorithm. By the currently best bound on ω, the running time of our algorithm is $\mathcal{O}(n^{2.376})$. Our algorithm substantially improves the previous time-bounds for this problem, and its asymptotic time complexity matches that of the fastest known algorithm for finding *any* triangle (not necessarily a maximum-weight one) in a graph. We can extend our algorithm to improve the upper bounds on finding a maximum-weight triangle in a sparse graph and on finding a maximum-weight subgraph isomorphic to a fixed graph. We can find a maximum-weight triangle in a vertex-weighted graph with m edges in asymptotic time required by the fastest algorithm for finding *any* triangle in a graph with m edges, i.e., in time $\mathcal{O}(m^{1.41})$. Our algorithms for a maximum-weight fixed subgraph (in particular any clique of constant size) are asymptotically as fast as the fastest known algorithms for a fixed subgraph.

Key words. time complexity, graph algorithms, vertex-weighted graph, graph triangle, matrix multiplication

AMS subject classifications. 68W01, 68W40, 68Q25, 68R10, 05C50

DOI. 10.1137/070695149

1. Introduction.

We consider a classical graph problem of finding a heaviest fixed subgraph in a vertex-weighted graph. The weight of a subgraph is defined as the total weight of its vertices.

The most basic unweighted version of that problem is detecting a *triangle* (a cycle of length three, C_3) in a graph. It is well known that the asymptotic time complexity of triangle detection in a graph on n vertices does not exceed that of $n \times n$ matrix multiplication [12], that is, $\mathcal{O}(n^\omega)$, where $\omega < 2.376$ is the exponent of the fastest matrix multiplication algorithm [5] (see also [3]).

The more general problem of finding a *maximum-weight triangle* in a graph with vertex or edge weights in $\mathcal{O}(n^{3-\epsilon})$ time has been widely open for a long time. Only recently, Vassilevska and Williams [17] resolved positively the variant with vertex weights.

The problems of detecting fixed cliques and, more generally, subgraphs isomorphic to fixed graphs and their weighted optimization variants, can be regarded as natural generalizations of the triangle detection problem [9, 15]. They constitute the basic instances of the fixed subgraph isomorphism problem [10] and its weighted optimization variants, respectively. In the weighted variants of these problems, the task is to find a maximum (or, equivalently, minimum) weight subgraph isomorphic to a

*Received by the editors June 22, 2007; accepted for publication (in revised form) March 12, 2009; published electronically June 3, 2009. A preliminary version of this paper appeared in *Proceedings of the* 18*th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007, pp. 986–994. This research was supported in part by NSF ITR grant CCR-0313219, by EPSRC grant EP/D063191/1, by the Centre for Discrete Mathematics and Its Applications (DIMAP), and by VR grant 621-2005-4085.

http://www.siam.org/journals/sicomp/39-2/69514.html

†Department of Computer Science and Centre for Discrete Mathematics and Its Applications (DIMAP), University of Warwick, Coventry, CV4 7AL, UK (A.Czumaj@warwick.ac.uk).

‡Department of Computer Science, Lund University, S-22 100 Lund, Sweden (Andrzej.Lingas@cs.lth.se).

fixed graph [16, 17, 18]. The weight of a subgraph is defined as the total weight of its vertices or/and edges, respectively. Typically, an improved upper time-bound on such a triangle problem yields an improved upper time-bound on the corresponding fixed subgraph isomorphism problem (cf. [15, 18]).

1.1. Related work.

Unweighted graphs. Itai and Rodeh [12] were the first to observe that a *shortest cycle*, in particular a triangle (i.e., a clique on three vertices), can be detected in an undirected graph on n vertices in time $\mathcal{O}(n^\omega)$. To determine if there is a triangle, we first compute the Boolean square of the adjacency matrix A and then check if there is a pair of indices $\langle i,j\rangle$ with $A[i,j]=1$ and $A^2[i,j]=1$. (For sparse graphs, Itai and Rodeh [12] proposed more efficient algorithms for triangle detection.)

Nešetřil and Poljak [15] extended the algorithm of Itai and Rodeh to include the detection of cliques of fixed size ℓ in time $\mathcal{O}(n^{\alpha(\ell)})$, where $\alpha(\ell)=\lfloor \ell/3\rfloor\cdot\omega+\ell \bmod 3$.

Alon, Yuster, and Zwick [2] considered the sparse version of the triangle detection problem and showed how to find a triangle in time $\mathcal{O}(m^{\frac{2\omega}{\omega+1}})$, where m is the number of edges of the input graph. Subsequently, Kloks, Kratch, and Müller [14] and Eisenbrand and Grandoni [9] generalized the result of [2] to include ℓ-cliques. The running time of their combined algorithms is $\mathcal{O}(m^{\alpha(\ell)/2})$ for $\ell\geq 6$.

Eisenbrand and Grandoni [9] also provided an improved algorithm for finding an ℓ-clique and, even more generally, for detecting a fixed (induced or not) subgraph on ℓ vertices. They used the fast algorithm for the *rectangular matrix multiplication* [4, 11] to improve the running time of the algorithm due to Nešetřil and Poljak in the case when $\ell \bmod 3\neq 0$. If $\omega(r,s,t)$ denotes the exponent of the fastest algorithm for the multiplication of an $n^r\times n^s$ matrix by an $n^s\times n^t$ matrix, then the algorithm of Eisenbrand and Grandoni [9] runs in time $\mathcal{O}(n^{\omega(\lfloor l/3\rfloor,\lceil (l-1)/3\rceil,\lceil l/3\rceil)})$ on a graph with n vertices.

Weighted graphs. The first $\mathcal{O}(n^{3-\epsilon})$-time algorithm for a maximum-weight triangle in vertex-weighted graphs has been proposed recently by Vassilevska and Williams [17]. Soon after, Vassilevska, Williams, and Yuster [18] proposed a faster solution to this problem by a straightforward reduction to the problem of computing the so-called *maximum witnesses* of Boolean matrix product.

A *maximum witness* of an entry $C[i,j]$ of the Boolean product C of two square Boolean matrices A and B is the maximum index k such that $A[i,k]=1$ and $B[k,j]=1$. It is sufficient to number the vertices of the input graph in nondecreasing weight order and compute the maximum witnesses for the Boolean square C of the adjacency matrix. For each edge (i,j), the maximum witness k of $C[i,j]$ (if any) forms the heaviest triangle including this edge. Hence, using the $\mathcal{O}(n^{2.616})$ time-bound for the maximum witness problem established in [13] and improved to $\mathcal{O}(n^{2.575})$ by rectangular matrix multiplication [7], Vassilevska, Williams, and Yuster [18] obtained the same upper time-bounds for finding a maximum-weight triangle in a vertex-weighted graph.

Vassilevska, Williams, and Yuster [18] studied also the problem for sparse graphs and presented an $\mathcal{O}(m^{\frac{18-4\omega}{13-3\omega}})=\mathcal{O}(m^{1.45})$-time algorithm for finding a maximum-weight triangle in a vertex-weighted graph. The same authors [18] considered also the vertex-weighted variants of the problems of finding a fixed clique, and more generally, a subgraph isomorphic to a fixed graph. The objective was to find such a clique or subgraph maximizing the total vertex weight. Their approach generalized the reductions of the fixed clique and fixed subgraph problems to the triangle and fixed clique problems from [9, 15] to include the weighted case; see Table 1.

TABLE 1

Summary of results for the problem of finding a maximum-weight triangle. In all results, n denotes the number of vertices, m the number of edges, B is the number of bits of precision of the input, $\omega < 2.376$ is the exponent of the fastest matrix multiplication algorithm [5]*, and $\omega^* = \max\{\omega, 2 + o(1)\}$ (hence, $\omega^* < 2.376$).*

Problem	Source	Running time	Numerical runtime/comments
Maximum-weight triangle	[17]	$\mathcal{O}(B \cdot n^{\frac{3+\omega}{2}})$	$\mathcal{O}(B \cdot n^{2.688})$
	[17]	$\mathcal{O}(n^{\frac{3+\omega}{2}} \log n)$	$\mathcal{O}(n^{2.688})$; randomized
	[18]	$\mathcal{O}(n^{2+1/(4-\omega)})$	$\mathcal{O}(n^{2.616})$
	[18]		$\mathcal{O}(n^{2.575})$
	this paper	$\mathcal{O}(n^{\omega^*})$	$\mathcal{O}(n^{2.376})$
Maximum-weight triangle	[18]	$\mathcal{O}(m^{\frac{18-4\,\omega}{13-3\,\omega}})$	$\mathcal{O}(m^{1.45})$
Graph with m edges	this paper	$\mathcal{O}(m^{\frac{2\,\omega^*}{1+\omega^*}})$	$\mathcal{O}(m^{1.41})$

1.2. Our contributions. We present a new algorithm for finding a maximum-weight triangle in a vertex-weighted graph. It does not rely on computing maximum witnesses of Boolean matrix product. Instead, it strongly utilizes the fact that the output to the problem is a single triangle. By applying a recursive elimination scheme and the fast matrix multiplication algorithm, we obtain an algorithm whose running time is $\mathcal{O}(n^{\omega^*}) = \mathcal{O}(n^{2.376})$, where $\omega^* = \max\{\omega, 2 + o(1)\}$. The running time of our algorithm matches that of the fastest algorithm for finding *any* triangle (not necessarily one with the maximum weight) in a graph.

Next, we study the same problem for sparse graphs with m edges (with the running time being a function of m). We use our $\mathcal{O}(n^{\omega^*})$-time algorithm for finding a maximum-weight triangle to design an algorithm running in time $\mathcal{O}(m^{\frac{2\,\omega^*}{1+\omega^*}}) = \mathcal{O}(m^{1.41})$. The running time of this algorithm also matches that of the fastest algorithm for finding *any* triangle in a graph, due to Alon, Yuster, and Zwick [1].

Following [18], we also extend our algorithm for finding a maximum-weight triangle to detecting a maximum-weight clique $K_{3\,h}$ (or, in general, to detecting a maximum-weight fixed subgraph, either induced or not, with $3\,h$ vertices) in time $\mathcal{O}(n^{\omega^*\,h}) = \mathcal{O}(n^{2.376\,k})$, for any constant h. To extend it further to other values of the size of the graphs, we design two algorithms. The first algorithm finds a maximum-weight clique K_h (or, in general, any fixed subgraph with h vertices) in time $\mathcal{O}(n^{\lfloor h/3 \rfloor \cdot \omega^* + (h \bmod 3)})$. This algorithm improves the running time upon the fastest previously existing algorithms (see [18]) for all values of $h \geq 6$. Our second algorithm uses fast rectangular matrix multiplication (instead of that for square matrices) similar to that due to Eisenbrand and Grandoni [9] for the unweighted $\mathcal{O}(1)$-clique problem, improving the running time even further (for values $h \bmod 3 \neq 0$). So, if $h = 3\,f + 1$, then the second algorithm runs in time $\mathcal{O}(n^{f \cdot \omega(1, \frac{f+1}{f}, 1)} + n^{f \cdot (2 + \frac{1}{f} + o(1))})$, where $\omega(1, r, 1)$ is the exponent of the multiplication of an $n \times n^r$ matrix by an $n^r \times n$ matrix. For $h = 3\,f + 2$, the running time is $\mathcal{O}(n^{(f+1) \cdot \omega(1, \frac{f}{f+1}, 1)} + n^{(f+1) \cdot (2 + o(1))})$. By a known result about $\omega(1, r, 1)$ [4, 5, 11], this yields in particular running times of $\mathcal{O}(n^{2.376\,f})$ for $h = 3f$, $\mathcal{O}(n^{2.376 \cdot f + 1})$ for $h = 3f + 1$, and $\mathcal{O}(n^{2.376 f + 1.844})$ for $h = 3f + 2$. These bounds subsume the corresponding ones from [18] (see Table 2) and match those for unweighted graphs from [9].

We also present a direct generalization of our algorithm for finding a maximum-

weight triangle to include the problem of finding a maximum-weight subgraph isomorphic to a fixed graph K on k vertices in a vertex-weighted graph on n vertices. The direct generalization solves the maximum-weight subgraph problem in time $\mathcal{O}(n^{\delta} + n^{h-1+o(1)})$, where δ is the exponent of the fastest algorithm for determining the existence of a subgraph isomorphic to H. In the context of all our results, it is useful solely for $h \in \{3, 4, 5\}$.

Paper organization. In section 2, we describe our algorithm for finding a maximum-weight triangle in a vertex-weighted graph. In section 3, we derive the upper time-bound on finding a maximum-weight triangle for sparse graphs. In section 4, we use our algorithm for a heaviest triangle and extend it in order to derive upper time-bounds on finding maximum-weight cliques and subgraphs in a vertex-weighted graph. In section 5, we improve the bounds of section 4 by using rectangular matrix multiplication. Finally, in section 6 we outline our direct generalization of the algorithm for a maximum-weight triangle to include a maximum-weight subgraph isomorphic to a fixed subgraph.

2. Finding a maximum-weight triangle in $\mathcal{O}(n^{\omega^*})$ time. We present here a new recursive algorithm for finding a maximum-weight triangle in $\mathcal{O}(n^{\omega^*})$ time.

We start by sorting the vertices of the input graph $G = (V, E)$ in nondecreasing order with respect to their weights. Therefore from now on, we will always assume that $V = \{1, \ldots, n\}$ with the weight of vertex i to be smaller than or equal to the weight of vertex $i + 1$ for all i, $1 \le i \le n - 1$.

We will use a parameter λ, where λ may be a function of n; the value of λ will be set up later, in the proof of Theorem 4. For simplicity of presentation, we assume that n is a power of λ. This assumption can be achieved by increasing n by a multiplicative factor less than λ via adding dummy vertices.

For any triplet (X, Y, Z) of intervals $X,\ Y,\ Z \subset [1,\ n]$, we say (X, Y, Z) *has a triangle* if there are three vertices $x \in X, y \in Y, z \in Z$ such that (x, y, z) is a triangle in G.

Observe that if $|X| = |Y| = |Z|$, then one can use fast matrix multiplication (cf. [12]) to test in $\mathcal{O}(|X|^{\omega})$ time if (X, Y, Z) has a triangle. Namely, one can compute the Boolean product C of two submatrices of the adjacency matrix A of G corresponding to the edges between vertices in X and Y, and vertices in Y and Z, respectively, and then check if there are $x \in X$ and $y \in Z$ such that $A[x, z] = 1$ and $C[x, z] = 1$.

If (X, Y, Z) has a triangle, then a *maximum-weight triangle in* (X, Y, Z) is a maximum-weight triangle (x, y, z) in G with $x \in X, y \in Y, z \in Z$.

Algorithm HT_λ. Now, we can describe our recursive algorithm $HT_\lambda(G, I, J, K)$. In each recursive call, the algorithm has three intervals (I, J, K), $I, J, K \subseteq V$, and the goal is to find a maximum-weight triangle in (I, J, K). Initially, $I = J = K = V$ and in all recursive instances $|I| = |J| = |K|$.

1. To find a maximum-weight triangle for the triplet (I, J, K), we first partition each interval I, J, and K into λ subintervals of the same size. That is, for every i, $1 \le i \le \lambda$, we define subintervals I_i, J_i, and K_i of I, J, and K, respectively, such that $|I_i| = \frac{|I|}{\lambda}$, $|J_i| = \frac{|J|}{\lambda}$, and $|K_i| = \frac{|K|}{\lambda}$. For example, if $I = \{s, s + 1, \ldots, s + r\lambda - 1\}$, then $I_i = \{s + (i - 1)\lambda, \ldots, s + i\lambda - 1\}$.
2. For each triplet of subintervals of the form (I_i, J_j, K_k) with $1 \le i, j, k \le \lambda$, determine if (I_i, J_j, K_k) has a triangle.
3. Next, we use some basic properties of the weights in the intervals. Observe that for every $1 \le i < i' \le \lambda$, if $s \in I_i$ and $r \in I_{i'}$, then the weight of vertex s is smaller than or equal to the weight of vertex r; the same property holds

for the subintervals of J and K. Therefore, if for a given triplet (I_i, J_j, K_k) there is another triplet $(I_{i'}, J_{j'}, K_{k'})$ that has a triangle and such that $i < i'$, $j < j'$, and $k < k'$, then we can ignore all triangles in the triplet (I_i, J_j, K_k) because their weight will not be larger than the weight of any triangle in the triplet $(I_{i'}, J_{j'}, K_{k'})$. By this observation, if for a given triplet (I_i, J_j, K_k) there is another triplet $(I_{i'}, J_{j'}, K_{k'})$ that has a triangle and such that $i < i'$, $j < j'$, and $k < k'$, then we call (I_i, J_j, K_k) *overshadowed*.

The next step of our algorithm is to consider all triplets of subintervals (I_i, J_j, K_k) having a triangle that are *not overshadowed* and recursively call for them $HT_\lambda(G, I_i, J_j, K_k)$ to find a maximum-weight triangle in (I_i, J_j, K_k).

4. The final step of the algorithm is to collect the maximum-weight triangles computed in the recursive calls and to return one with the maximum weight.

2.1. Analysis of $HT_\lambda(G, I, J, K)$. We begin with the following lemma that follows immediately from our discussion above.

LEMMA 1. *The procedure $HT_\lambda(G, I, J, K)$ returns a maximum-weight triangle (if any) in (I, J, K).*

Now, we will show that by choosing the value of λ appropriately, the running time of our algorithm is $\mathcal{O}(n^{2+o(1)} + n^\omega) < \mathcal{O}(n^{2.376})$. The main idea in our analysis is to observe that while in each call the algorithm considers λ^3 triplets of subintervals, the number of recursive calls will be only $\mathcal{O}(\lambda^2)$ because if many triplets have a triangle, then many of them must be overshadowed. To formalize this intuition, we prove the following lemma.

LEMMA 2. *Let λ be any positive integer. Let X be any subset of $\{1, 2, \ldots, \lambda\}^3$ which does not contain any two t, t' such that each coordinate in t is greater than the corresponding one in t. Then the cardinality of X is at most $3\lambda^2 - 3\lambda + 1$.*

Proof. Define the relation $\prec$ such that $(i, j, k) \prec (i', j', k')$ if and only if $i < i'$, $j < j'$, and $k < k'$. The relation $\prec$ defines a partial order on $\{1, 2, \ldots, \lambda\}^3$. For each $(t_1, t_2, t_3) \in \{1, 2, \ldots, \lambda\}^3$ that has at least one coordinate equal to 1, define $\mathsf{chain}((t_1, t_2, t_3))$ to be the set of all triplets in $\{1, 2, \ldots, \lambda\}^3$ of the form (t_1+i, t_2+i, t_3+i) for $i = 0, 1, \ldots$. Observe that $\mathsf{chain}(t)$ is indeed a chain in the poset $(\{1, 2, \ldots, \lambda\}^3, \prec)$ and the chains $\mathsf{chain}(t)$ cover all the elements in $\{1, 2, \ldots, \lambda\}^3$. It follows now from Dilworth's lemma [8] that the cardinality of the largest antichain in the aforementioned poset does not exceed the number of the triplets with at least one coordinate equal to 1, which, in turn, is at most $\lambda^3 - (\lambda - 1)^3 = 3\lambda^2 - 3\lambda + 1$. ∎

LEMMA 3. *The running time of HT_λ satisfies the recurrence*

$$T(N) \leq (3\lambda^2 - 3\lambda + 1) \cdot T(N/\lambda) + \mathcal{O}(\lambda^{3-\omega} \cdot N^\omega + \lambda^6) \ . \tag{1}$$

Proof. Steps 1–2 of the algorithm can be performed in time $\mathcal{O}(\lambda^{3-\omega} \cdot N^\omega)$. For each triplet (I_i, J_j, K_k), we can determine if it is overshadowed in $\mathcal{O}(\lambda^3)$ time; hence for all triplets this will take $\mathcal{O}(\lambda^6)$ time. By Lemma 2, we will make at most $3\lambda^2 - 3\lambda + 1$ recursive calls, and in each recursive call, the size of each interval is N/λ. The final step can be performed in $\mathcal{O}(\lambda^2)$ time. This gives us recurrence (1) for the running time of HT_λ. ∎

Using Lemma 3 and by an appropriate choice of λ, we obtain our main result.

THEOREM 4. *A maximum-weight triangle in a vertex-weighted graph on n vertices can be found in time $\mathcal{O}(n^{2+o(1)} + n^\omega) < \mathcal{O}(n^{2.376})$.*

Proof. Let us consider the recurrence (1) for the running time of HT_λ. Our goal is to choose λ to minimize it. For a given λ, the solution of (1) depends on the relationship between $n^{\log_\lambda(3\lambda^2-3\lambda+1)}$ and $\lambda^{3-\omega} \cdot n^\omega + \lambda^6$: if the first term is significantly

larger, then the solution is $\mathcal{O}(n^{\log_\lambda(3\lambda^2-3\lambda+1)})$; if the second term is significantly larger, then the solution is $\mathcal{O}(\lambda^{3-\omega}\cdot n^\omega+\lambda^6)$ (e.g., see Master theorem in [6]). However, the fact that we do not know ω, and hence have to parameterize our analysis, requires special care in the selection of λ that minimizes (1), as shown in the manipulations that follow.

Let $\lambda=\lambda(n)\geq 1$ be a (possibly constant) function of n. By Lemma 3, there is a positive constant c such that for all N being powers of λ with $\lambda\leq N\leq n$,

$$T(N)\leq c\cdot\lambda^2\cdot T(N/\lambda)+c\cdot N^\omega\cdot\lambda^{3-\omega}+c\cdot\lambda^6\ . \tag{2}$$

With the inequality (2), easy induction on k implies that for all k with $\lambda^k\leq n$ we have

$$T(n)\leq c^k\cdot\lambda^{2k}\cdot T(n/\lambda^k)+n^\omega\cdot\sum_{i=1}^{k}c^i\,\lambda^{1+(2-\omega)\,i}+\sum_{i=1}^{k}c^i\,\lambda^{4+2\,i}\ .$$

Hence, if we set $k=\log_\lambda n$, then we obtain

$$\begin{aligned}T(n)&\leq c^k\cdot\lambda^{2k}\cdot T(n/\lambda^k)+n^\omega\cdot\lambda\cdot\sum_{i=1}^{k}(c\,\lambda^{2-\omega})^i+\lambda^4\cdot\sum_{i=1}^{k}(c\,\lambda^2)^i\\&\leq c^{\log_\lambda n}\cdot\lambda^{2\,\log_\lambda n}\cdot\mathcal{O}(1)+n^\omega\cdot\lambda\cdot\sum_{i=1}^{\log_\lambda n}(c\,\lambda^{2-\omega})^i+c^{1+\log_\lambda n}\cdot\lambda^{6+2\log_\lambda n}\\&=\mathcal{O}(c^{\log_\lambda n}\cdot\lambda^6\cdot n^2)+n^\omega\cdot\lambda\cdot\sum_{i=1}^{\log_\lambda n}(c\,\lambda^{2-\omega})^i\ .\end{aligned}$$

Now, we know that c is a small constant and we have the freedom of choosing λ. We choose λ depending on whether $\omega=2+o(1)$ or not.

- If there is a positive constant ϵ with $\omega\geq 2+\epsilon$, then we set $\lambda=(2c)^{1/\epsilon}$. Let us first observe that $c^{\log_\lambda n}=n^{\log c/\log\lambda}=n^{\frac{\log c}{\log(2c)/\epsilon}}\leq n^\epsilon$ and

$$\sum_{i=1}^{\log_\lambda n}(c\,\lambda^{2-\omega})^i\leq\sum_{i=1}^{\log_\lambda n}(c\,\lambda^{-\epsilon})^i=\sum_{i=1}^{\log_\lambda n}2^{-i}\leq 1\ .$$

  Hence,

$$T(n)\leq\mathcal{O}(c^{\log_\lambda n}\,\lambda^6\,n^2)+n^\omega\,\lambda\sum_{i=1}^{\log_\lambda n}(c\,\lambda^{2-\omega})^i\ \leq\ \mathcal{O}(n^{2+\epsilon})+\mathcal{O}(n^\omega)\ \leq\ \mathcal{O}(n^\omega)\ .$$

- If $\omega\leq 2+o(1)$, then we will set $\lambda=n^{1/\log\log n}$ and the running time becomes

$$\begin{aligned}T(n)&\leq\mathcal{O}(c^{\log_\lambda n}\cdot\lambda^6\cdot n^2)+n^\omega\cdot\lambda\cdot\sum_{i=1}^{\log_\lambda n}(c\,\lambda^{2-\omega})^i\\&\leq\mathcal{O}((\log n)^{\log c}\cdot\lambda^6\cdot n^2)+n^\omega\cdot\lambda\cdot\sum_{i=1}^{\log_\lambda n}c^i\\&\leq\mathcal{O}((\log n)^{\log c}\cdot n^{2+6/\log\log n})+n^{\omega+\frac{1}{\log\log n}}\cdot\mathcal{O}((\log n)^{\log c})\\&=n^{2+o(1)}.\qquad\square\end{aligned}$$

3. Improved bounds for sparse graphs. By arguing analogously as in the proof of Theorem 2 in [18], we can refine the upper bound from Theorem 4 in the case of sparse graphs as follows.

COROLLARY 5. *A maximum-weight triangle in a vertex-weighted graph with m edges and no isolated vertices can be found in time $\mathcal{O}(m^{\frac{2\,\omega^*}{1+\omega^*}}) < \mathcal{O}(m^{1.41})$.*

Proof. Let $G = (V, E)$ be the input graph and let X be the set of all vertices in V of degree at most δ. It follows that $|V \setminus X| \leq 2\,m/\delta$. In time $\mathcal{O}(m\,\delta)$, we can enumerate all triangles in G that contain a vertex in X and find a maximum-weight one among them. (To achieve this, it is sufficient to list all edges in G having at least one endpoint of degree at most δ, and then for each such edge to examine all the at most δ triangles formed by it and a neighbor of its endpoint of degree at most δ.) On the other hand, a maximum-weight triangle in G that has all vertices in $V \setminus X$ can be found in time $\mathcal{O}((m/\delta)^{\omega^*})$ by Theorem 4. By setting $\delta = m^{\frac{\omega^*-1}{1+\omega^*}}$, we obtain the corollary. □

4. Finding max-weight clique K_h and H-subgraphs. Vassilevska, Williams, and Yuster [18] presented a reduction of the problem of finding a maximum-weight $\mathcal{O}(1)$-clique to that of finding a maximum-weight triangle (generalizing that for unweighted graphs due to Nešetřil and Poljak [15]).

In this section, we use this approach to extend our algorithm for finding a maximum-weight triangle to include finding a maximum-weight fixed clique or any maximum-weight subgraph isomorphic to a fixed graph. To keep our paper self-contained, we provide a complete analysis.

THEOREM 6. *A maximum-weight clique K_h in a vertex-weighted graph on n vertices can be found in time*

$$\mathcal{O}(n^{\lfloor h/3 \rfloor \cdot \omega^* + (h \bmod 3)}) \ .$$

Proof. Let $f = \lfloor h/3 \rfloor$. Suppose first that $h = 3f$. Form a new graph G' in which each vertex corresponds one to one to a K_f in the original graph G and the weight of such a vertex equals the total weight of the vertices in this K_f. Two vertices in G' are connected by an edge if and only if the corresponding cliques form a K_{2f} clique in G. Observe that G' has $\mathcal{O}(n^f)$ vertices and can be constructed in time $\mathcal{O}(n^{2f})$. Furthermore, a maximum-weight triangle in G' corresponds to a maximum-weight K_h clique in G. Therefore, by using Theorem 4 to find a maximum-weight triangle in G', we can find a maximum-weight clique in G in time $\mathcal{O}(n^{\omega^*})$.

Next, let us consider the case $h = 3f + 2$. Find all cliques K_f and K_{f+1} that are subgraphs of G. Divide the K_{f+1} subgraphs into $\mathcal{O}(n)$ groups of size $\mathcal{O}(n^f)$. For each such two groups a and b (a can be equal to b) and the K_f subgraphs form a tripartite graph $G_{a,b}$ whose vertices in the first part, second part, and third part are in one-to-one correspondence with the K_{f+1} subgraphs in the first group, the K_{f+1} subgraphs in the second group, and the K_f subgraphs of G, respectively. The weights of the vertices in $G_{a,b}$ are equal to the total weights of the corresponding cliques in G. There is an edge between two vertices in $G_{a,b}$ if the corresponding cliques are disjoint and induce the clique in G whose size equals the sum of their sizes. Observe that all the constructions can be easily done in total time $\mathcal{O}(n^{2f+2})$. Now note that a maximum-weight triangle among the maximum-weight triangles in the graphs $G_{a,b}$ yields a maximum-weight K_h in G. By Theorem 4, it takes time $\mathcal{O}(n^2 \times (n^f)^{\omega^*})$.

The proof of case $h = 3f + 1$ is analogous. For each group a of the K_{f+1} subgraphs we form a tripartite graph G_a whose vertices in the first part are in one-to-one correspondence with the K_{f+1} subgraphs in a, and the vertices in each of the

two remaining parts are in one-to-one correspondence with the K_f subgraphs of G. The vertex weights and edges are specified analogously as in the case of $G_{a,b}$, and all the constructions take time $\mathcal{O}(n^{f+1} \times n^f)$. Analogously, a maximum-weight triangle among the maximum-weight triangles in the graphs G_a yields a maximum-weight K_h in G. □

It is easy to extend the result from Theorem 6 to arbitrary induced subgraphs isomorphic to a given graph H on h vertices. We use an analogous construction to that in the proof of Theorem 6. We decompose H into three induced subgraphs H_i (possibly isomorphic), $i = 1, 2, 3$, and for each isomorphism between an induced subgraph of G and H_i, we form a separate node in an auxiliary graph. Two such nodes are connected by an edge if the union of the corresponding isomorphisms yields an isomorphism between the subgraph induced by the vertices of the two underlying subgraphs and the subgraph of H induced by the vertices of the H_i images (required to be different) of the two isomorphisms. (In the case when $H = K_h$, it has not been necessary to have separate nodes for different isomorphisms between an induced subgraph of G and a clique which is a subgraph of K_h because of the symmetry between the vertices in the clique with respect to K_h.)

Furthermore, since any subgraph (not necessarily induced) of G on h vertices which is isomorphic to H is a subgraph of the induced subgraph of G on the same h vertices, finding (not necessarily induced) subgraphs reduces to finding induced subgraphs of the same size. The induced subgraphs correspond to all possible supergraphs of H on h vertices. This yields the following.

THEOREM 7. *Let H be a fixed graph on h vertices. A maximum-weight induced subgraph of a vertex-weighted graph on n vertices that is isomorphic to H can be found in time*

$$\mathcal{O}(n^{\lfloor h/3 \rfloor \cdot \omega^* + (h \bmod 3)}) \ .$$

5. Refinement by using fast rectangular matrix multiplication. The algorithms and the bounds from Theorems 6–7 can be improved for $h \bmod 3 \neq 0$ if we use fast rectangular matrix multiplication algorithms (instead of fast square matrix multiplication), similarly to how Eisenbrand and Grandoni [9] improved the bounds for finding a fixed subgraph.

Let $\omega(1, \sigma, 1)$ denote the exponent of the multiplication of an $n \times n^\sigma$ matrix by an $n^\sigma \times n$ matrix. In order to improve Theorems 6 and 7 in terms of $\omega(1, \sigma, 1)$, we generalize the procedure $HT_\lambda(G, I, J, K)$ to include the case where the sizes of the intervals I, J, and K are not necessarily equal.

For any σ, $\frac{1}{2} \leq \sigma \leq 2$, let $HT_\lambda^{\langle\sigma\rangle}(G, I, J, K)$ denote such an analogous generalized procedure, where the sizes of I, J, and K (besides being powers of λ) satisfy $|I| = |K|$ and $|J| = |I|^\sigma$. We modify only steps 1–2 of the algorithm HT_λ. In step 1, instead of partitioning each interval into λ subintervals, we partition now interval J into λ^σ subintervals $J_1, \ldots, J_{\lambda^\sigma}$ (intervals I and K are partitioned into λ subintervals as in HT_λ). Observe that this gives us $\lambda^{2+\sigma}$ triplets to consider. Then, in step 2, since the intervals I, J, and K have different sizes, we will use the fast rectangular Boolean matrix multiplication algorithm to determine if a given triplet of subintervals (I_i, J_j, K_k) has a triangle. Since in each triplet we have $|I_i| = |K_k| = |I|/\lambda$ and $|J_j| = |J|/\lambda^\sigma = (|I|/\lambda)^\sigma$, step 2 can be performed in time $\mathcal{O}(\lambda^{2+\sigma} \cdot (|I|/\lambda)^{\omega(1,\sigma,1)})$.

By performing an analysis of $HT_\lambda^{\langle\sigma\rangle}$ analogous to that of HT_λ, we obtain the following lemma.

LEMMA 8. *One can choose the parameter λ to ensure that the procedure $HT_\lambda^{\langle\sigma\rangle}(I,K,J)$ returns a maximum-weight triangle (i,j,k), if any, in time $\mathcal{O}(|I|^{\omega(1,\sigma,1)} + |I|^{2+o(1)} + |I|^{1+\sigma+o(1)})$.*

Proof. We proceed as in the proof of Lemma 3. The correctness of $HT_\lambda^{\langle\sigma\rangle}(I,K,J)$ follows easily from the discussion above. To determine the running time of $HT_\lambda^{\langle\sigma\rangle}(I,K,J)$, we have to modify the analysis from Lemma 3 to incorporate the changes in the runtime of step 2 and the new bound for the number of recursive calls performed.

Step 2 of $HT_\lambda^{\langle\sigma\rangle}(I,K,J)$ requires $\mathcal{O}(\lambda^{2+\sigma} \cdot (|I|/\lambda)^{\omega(1,\sigma,1)})$ time. Using arguments from Lemma 2, the number of recursive calls is at most $\lambda^{2+\sigma} - (\lambda-1)^2(\lambda^\sigma - 1) = 2\lambda^{1+\sigma} + \lambda^2 - 2\lambda - \lambda^\sigma + 1$. Since we consider $\lambda \geq 1$, this is always bounded from above by $2\lambda^{1+\sigma} + \lambda^2$. Hence, we obtain the following recurrence for the running time of $HT_\lambda^{\langle\sigma\rangle}$ (with $|I| = N$):

$$T(N) \leq \mathcal{O}(\lambda^{2+\sigma}\, N^{\omega(1,\sigma,1)} + \lambda^6) + (2\lambda^{1+\sigma} + \lambda^2)\, T(N/\lambda) \ ,$$

with the base case $T(1) = \mathcal{O}(1)$.

Since we have assumed that $\frac{1}{2} \leq \sigma \leq 2$, and since λ is a parameter that we can set as an arbitrary integer, one can solve this recurrence (see the appendix) analogously as in the proof of Theorem 4 to conclude that the running time is $T(N) = \mathcal{O}(N^{\omega(1,\sigma,1)} + N^{2+o(1)} + N^{1+\sigma+o(1)})$. ∎

Equipped with Lemma 8, we can improve the bound from Theorem 6 for $h \neq 3f$ as follows.

Consider first the case when $h = 3f + 2$. Find all cliques K_f and K_{f+1} that are subgraphs of G. Next, form a new graph G'', where vertices are in one-to-one correspondence with the found cliques and their weight equals the total weight of the corresponding clique. Two vertices in G'' are adjacent if the corresponding cliques induce a clique in G'' whose size equals the sum of their sizes. Next, number the vertices of G'' such that the vertices corresponding to the K_{f+1} cliques occur in a continuous interval in nondecreasing weight order and such that those corresponding to the K_f cliques occur in a continuous interval in nondecreasing weight order.

Set I and K to the first of the aforementioned intervals and J to the second one, and run the procedure $HT_\lambda^{\langle\sigma\rangle}(I,J,K)$ for G'' and $\sigma = \log_{|I|} |J|$, where $\frac{1}{2} \leq \sigma < 1$.[1] Observe that by Lemma 8 and by the definitions of G'', I, J, and K, the triangle returned by $HT_\lambda^{\langle\sigma\rangle}(I,J,K)$, if any, corresponds to a maximum-weight K_h in G. By Lemma 8, monotonicity of the time taken by the multiplication of an $n \times n^\sigma$ matrix by an $n^\sigma \times n$ matrix with respect to n and σ, and straightforward calculations, $HT_\lambda^{\langle\sigma\rangle}(I,J,K)$ takes time $\mathcal{O}((n^{f+1})^{\omega(1,\frac{f}{f+1},1)} + (n^{f+1})^{2+o(1)})$ for sufficiently large λ.

The proof in case $h = 3f + 1$ is analogous with the exception that now I and K are set to the interval of vertices corresponding to K_f cliques, whereas J is set to the interval of vertices corresponding to K_{f+1} cliques. By analogous arguments, we conclude that in this case we can find a maximum-weight K_h in time $\mathcal{O}((n^f)^{\omega(1,\frac{f+1}{f},1)} + n^{f\,(2+\frac{1}{f}+o(1))})$.

By combining our improvements with Theorem 6, we obtain the following theorem.

THEOREM 9. *Let h be a positive integer, and let $f = \lfloor h/3 \rfloor$. A maximum-weight*

[1] The simplifying assumption about the sizes of the intervals being the power of λ can be achieved by increasing the sizes by a multiplicative factor less than λ via adding dummy vertices.

clique K_h in a vertex-weighted graph on n vertices can be found in time $T_h(n)$, where

$$T_h(n) = \begin{cases} \mathcal{O}(n^{f\,\omega^*}), & h \bmod 3 \equiv 0, \\ \mathcal{O}(n^{f\cdot\omega(1,\frac{f+1}{f},1)} + n^{f\,(2+\frac{1}{f}+o(1))}), & h \bmod 3 \equiv 1, \\ \mathcal{O}(n^{(f+1)\omega(1,\frac{f}{f+1},1)} + n^{(f+1)(2+o(1))}), & h \bmod 3 \equiv 2. \end{cases}$$

Similarly as in the previous section, the results from Theorem 9 can be extended to finding a maximum-weight fixed graph.

THEOREM 10. *For any fixed integer h, let H be any graph on h vertices. A maximum-weight induced subgraph of a vertex-weighted graph on n vertices that is isomorphic to H can be found in time $T_h(n)$, where the function $T_h(n)$ is as defined in Theorem* 9.

In asymptotically the same time complexity one can find a maximum-weight subgraph (not necessarily induced) isomorphic to H.

Coppersmith [4] and Huang and Pan [11] proved the following facts.

FACT 11 (see [4, 11]). *Let $\omega = \omega(1,1,1) < 2.376$ and let $\alpha = \sup\{0 \le r \le 1 : \omega(1,r,1) = 2 + o(1)\} > 0.294$. Then $\omega(1,r,1) \le \beta(r)$, where $\beta(r) = 2 + o(1)$ for $r \in [0,\alpha]$ and $\beta(r) = 2 + \frac{\omega-2}{1-\alpha}(r-\alpha) + o(1)$ for $r \in [\alpha,1]$.*

(Observe a useful fact that, if our goal is to compute $(f+1)\cdot\omega(1,\frac{f}{f+1},1)$, then the bounds in Fact 11 simplify it to $(f+1)\cdot\omega(1,\frac{f}{f+1},1) = (f+1)\cdot(2+\frac{\omega-2}{1-\alpha}(\frac{f}{f+1}-\alpha)+o(1)) = 2 - \frac{(\omega-2)\,\alpha}{1-\alpha} + f\cdot\omega + o(f) < 1.844 + f\cdot\omega + o(f)$.)

FACT 12 (see [11, section 8.1]). *$\omega(1,2,1) < 3.334$, and for every $r > 1$, we have $\omega(1,r,1) \le \omega + r - 1$.*

(Section 8.1 in [11] contains some discussion about stronger bounds for $\omega(1,r,1)$ for other values $r > 2$.)

Therefore, for example, by using the bounds from Facts 11 and 12, we have (see also Table 2)

$$\begin{aligned} T_3(n) &= \mathcal{O}(n^{\omega^*}) < \mathcal{O}(n^{2.376}) \ , \\ T_4(n) &= \mathcal{O}(n^{\omega(1,2,1)} + n^{3+o(1)}) < \mathcal{O}(n^{3.334}) \ , \\ T_5(n) &= \mathcal{O}(n^{2\cdot\omega(1,\frac{1}{2},1)} + n^{4+o(1)}) < \mathcal{O}(n^{4.220}) \ , \\ T_6(n) &= \mathcal{O}(n^{2\,\omega^*}) < \mathcal{O}(n^{4.752}) \ , \\ T_{3f}(n) &= \mathcal{O}(n^{f\,\omega^*}) < \mathcal{O}(n^{2.376\,f}) \ , \\ T_{3f+1}(n) &= \mathcal{O}(n^{f\,\omega^*+1}) < \mathcal{O}(n^{2.376\,f+1}) \ , \\ T_{3f+2}(n) &= \mathcal{O}(n^{f\,\omega^*+1.844}) < \mathcal{O}(n^{2.376\,f+1.844}) \ . \end{aligned}$$

Note that, for example, this bound subsumes the upper bounds of Theorem 6 for K_4 and K_5, and for K_{3f+2} for every $f \ge 1$.

6. Further extensions. Finally, we directly generalize our method for finding a maximum-weight triangle to include the problem of finding a maximum-weight subgraph isomorphic to a fixed graph H on h vertices in a vertex-weighted graph on n vertices.

The generalized algorithm starts from a search region specified by h sets of vertices, each of size n and sorted in increasing weight order, where the maximum weight subgraph isomorphic to H is supposed to have precisely one vertex from each set. Letting λ be a large constant, the region is divided into λ^h subregions, where each

TABLE 2

Summary of results for finding maximum-weight cliques of size greater than 3 *and fixed induced subgraphs. In all results,* n *denotes the number of vertices,* $\omega < 2.376$ *is the exponent of the fastest matrix multiplication algorithm* [5], $\omega^* = \max\{\omega, 2 + o(1)\}$ *(hence,* $\omega^* < 2.376$*), and* $\omega(1, r, 1)$ *is the exponent of the multiplication of an* $n \times n^r$ *matrix by an* $n^r \times n$ *matrix* [4, 11].

Problem	Source	Running time	Numerical runtime
Maximum-weight fixed subgraph on 4 vertices	[18]	$\mathcal{O}(n^{\omega+1})$	$\mathcal{O}(n^{3.376})$
	this paper	$\mathcal{O}(n^{\omega(1,2,1)} + n^{3+o(1)})$	$\mathcal{O}(n^{3.334})$
Maximum-weight fixed subgraph on 5 vertices	[18]	$\mathcal{O}(n^{\omega+2})$	$\mathcal{O}(n^{4.376})$
	this paper	$\mathcal{O}(n^{2\cdot\omega(1,\frac{1}{2},1)} + n^{4+o(1)})$	$\mathcal{O}(n^{4.220})$
Maximum-weight fixed subgraph on 6 vertices	[18]	$\mathcal{O}(n^{4+2/(4-\omega)})$	$\mathcal{O}(n^{5.232})$
	this paper	$\mathcal{O}(n^{2\,\omega^*})$	$\mathcal{O}(n^{4.752})$
Maximum-weight fixed subgraph on 7 vertices	[18]	$\mathcal{O}(n^{4+3/(4-\omega)})$	$\mathcal{O}(n^{5.848})$
	this paper	$\mathcal{O}(n^{2\,\omega(1,\frac{3}{2},1)} + n^{5+o(1)})$	$\mathcal{O}(n^{5.752})$
Maximum-weight fixed subgraph on 8 vertices	[18]	$\mathcal{O}(n^{2\,\omega+2} + n^{4+o(1)})$	$\mathcal{O}(n^{6.752})$
	this paper	$\mathcal{O}(n^{3\,\omega(1,\frac{2}{3},1)} + n^{6+o(1)})$	$\mathcal{O}(n^{6.596})$
Maximum-weight fixed subgraph on 9 vertices	[18]	$\mathcal{O}(n^{2\,\omega+3})$	$\mathcal{O}(n^{7.752})$
	this paper	$\mathcal{O}(n^{3\,\omega^*})$	$\mathcal{O}(n^{7.128})$
Maximum-weight fixed subgraph on $3f$ vertices	[17]	$\mathcal{O}(n^{\frac{(3+\omega)\,f}{2}})$; randomized	$\mathcal{O}(n^{2.688\,f})$
	[18]		$\mathcal{O}(n^{2.575\cdot f})$
	this paper	$\mathcal{O}(n^{\omega^*\cdot f})$	$\mathcal{O}(n^{2.376\cdot f})$
Maximum-weight fixed subgraph on $3f+1$ vertices	this paper	$\mathcal{O}(n^{f\,\omega(1,\frac{f+1}{f},1)} + n^{f\,(2+\frac{1}{f}+o(1))})$	$\mathcal{O}(n^{2.376\cdot f+1})$
Maximum-weight fixed subgraph on $3f+2$ vertices	this paper	$\mathcal{O}(n^{(f+1)\omega(1,\frac{f}{f+1},1)} + n^{(f+1)(2+o(1))})$	$\mathcal{O}(n^{2.376\cdot f+1.844})$

subregion contains h sets of $\mathcal{O}(n/\lambda)$ vertices. The algorithm determines for each subregion whether it contains a subgraph isomorphic to H or not in time $\mathcal{O}((n/\lambda)^\delta)$. By a straightforward generalization of Lemma 2 to include h-tuples instead of triplets, only $\mathcal{O}(\lambda^{h-1})$ among the subregions can contain a maximum-weight subgraph isomorphic to H. These $\mathcal{O}(\lambda^{h-1})$ subregions can be determined in constant time (but being a function of h and λ). The generalized algorithm recurses on each of these $\mathcal{O}(\lambda^{h-1})$ subregions. It runs in time $\mathcal{O}(n^\delta + n^{h-1+o(1)})$, where δ is the exponent of fastest algorithm for determining the existence of a subgraph isomorphic to H.

THEOREM 13. *If a vertex-weighted graph* G *on* n *vertices contains a subgraph isomorphic to a fixed graph* H *on* h *vertices, then such a maximum-weight subgraph can be found in* G *in time* $\mathcal{O}(n^\delta + n^{h-1+o(1)})$*, where* δ *is the exponent of fastest algorithm for determining the existence of a subgraph isomorphic to* H*.*

Observe that this result is weaker than those from the previous two sections for $h \geq 6$. Hence, it is useful solely for $h \in \{3, 4, 5\}$. In the special case when G is a cycle, the old result of Plehn and Voigt [16] yields the better upper time-bound of $\mathcal{O}(n^3)$ for $h = 4, 5$.

7. Conclusions. We have shown that finding a maximum-weight triangle is asymptotically not more difficult than matrix multiplication. Consequently, we could substantially improve prior upper time-bounds on finding a maximum-weight clique of a constant size and a maximum-weight subgraph isomorphic to a fixed graph.

A natural question arises of whether or not our results for vertex-weighted cliques as well as those known for unweighted cliques are asymptotically optimal.

Appendix. Solving the recurrence from Lemma 8. Note that in the proof of Lemma 8, we want to show that for an appropriate choice of parameter λ, the solution to the recurrence

$$T(N) \leq \mathcal{O}(\lambda^{2+\sigma}\, N^{\omega(1,\sigma,1)} + \lambda^6) + (2\lambda^{1+\sigma} + \lambda^2)\, T(N/\lambda)\ , \tag{3}$$

with the base case $T(1) = \mathcal{O}(1)$ and $\frac{1}{2} \leq \sigma \leq 2$, is $T(n) = \mathcal{O}(n^{\omega(1,\sigma,1)} + n^{2+o(1)} + n^{1+\sigma+o(1)})$.

The proof of this fact follows the arguments used in the proof of Theorem 4. Let us first introduce the notation $A = 2\lambda^{1+\sigma} + \lambda^2$, $B = c \cdot \lambda^{2+\sigma}$, and $C = c \cdot \lambda^6$, where c is a positive constant for which we can rewrite the recurrence (3) to obtain

$$T(N) \leq A \cdot T(N/\lambda) + B\, N^{\omega(1,\sigma,1)} + C\ .$$

With this recurrence, an easy proof by induction yields for all integers k, $k \leq \log n / \log \lambda$,

$$T(n) \leq A^k \cdot T(n/\lambda^k) + B \cdot n^{\omega(1,\sigma,1)} \cdot \sum_{i=0}^{k-1} (A/\lambda^{\omega(1,\sigma,1)})^i + C \cdot \sum_{i=0}^{k-1} A^i\ .$$

If we set $k = \log n / \log \lambda$, then we will obtain

$$\begin{aligned} T(n) &\leq \mathcal{O}(A^{\log n/\log\lambda}) + B \cdot n^{\omega(1,\sigma,1)} \cdot \sum_{i=0}^{\log n/\log\lambda - 1} (A/\lambda^{\omega(1,\sigma,1)})^i + C \cdot A^{\log n/\log\lambda} \\ &\leq \mathcal{O}(C \cdot A^{\log n/\log\lambda}) + B \cdot n^{\omega(1,\sigma,1)} \cdot \sum_{i=0}^{\log n/\log\lambda - 1} (A/\lambda^{\omega(1,\sigma,1)})^i\ . \end{aligned}$$

Before we continue, let us define $\mu_\sigma = \max\{2, 1+\sigma\}$ so that $A \leq 3\,\lambda^{\mu_\sigma}$. With this notation, the bound above can be simplified to the following:

$$T(n) \leq \mathcal{O}(\lambda^6 \cdot n^{\mu_\sigma + \log 3/\log\lambda}) + \mathcal{O}(\lambda^{2+\sigma} \cdot n^{\omega(1,\sigma,1)}) \cdot \sum_{i=0}^{\log n/\log\lambda - 1} (3 \cdot \lambda^{\mu_\sigma - \omega(1,\sigma,1)})^i\ .$$

Now, let us consider two cases.

- If there is a positive constant ϵ with $\omega(1,\sigma,1) \geq \mu_\sigma + \epsilon$, then we will choose $\lambda = 6^{1/\epsilon}$ to obtain the inequalities $\log 3/\log\lambda \leq \epsilon$ and $3 \cdot \lambda^{\mu_\sigma - \omega(1,\sigma,1)} \leq 3 \cdot \lambda^{-\epsilon} = \frac{1}{2}$. The latter inequality implies also that $\sum_{i=0}^{k-1} (3 \cdot \lambda^{\mu_\sigma - \omega(1,\sigma,1)})^i \leq 2$. Hence, we obtain

$$\begin{aligned} T(n) &\leq \mathcal{O}(\lambda^6 \cdot n^{\mu_\sigma + \log 3/\log\lambda}) + \mathcal{O}(\lambda^{2+\sigma} \cdot n^{\omega(1,\sigma,1)}) \cdot \sum_{i=0}^{k-1} (3 \cdot \lambda^{\mu_\sigma - \omega(1,\sigma,1)})^i \\ &\leq \mathcal{O}(n^{\mu_\sigma + \epsilon} + n^{\omega(1,\sigma,1)}) \\ &= \mathcal{O}(n^{\omega(1,\sigma,1)})\ . \end{aligned}$$

- If $\omega(1,\sigma,1) \le \mu_\sigma + o(1)$, then we will choose $\lambda = n^{1/\log\log n}$.

$$\begin{aligned}
T(n) &\le \mathcal{O}(\lambda^6 \cdot n^{\mu_\sigma + \log 3/\log\lambda}) + \mathcal{O}(\lambda^{2+\sigma} \cdot n^{\omega(1,\sigma,1)}) \cdot \sum_{i=0}^{k-1} (3 \cdot \lambda^{\mu_\sigma - \omega(1,\sigma,1)})^i \\
&\le \mathcal{O}\Big(n^{\mu_\sigma + \frac{\log 3 \log\log n}{\log n} + \frac{6}{\log\log n}} \\
&\qquad + n^{\omega(1,\sigma,1) + \frac{2+\sigma}{\log\log n}} \cdot 3^{\frac{\log n}{\log\lambda}} \cdot (\lambda^{\max\{\mu_\sigma - \omega(1,\sigma,1),0\}})^{\frac{\log n}{\log\lambda}}\Big) \\
&\le \mathcal{O}(n^{\mu_\sigma + o(1)} + n^{\omega(1,\sigma,1) + o(1) + \max\{\mu_\sigma - \omega(1,\sigma,1),0\}}) \\
&\le \mathcal{O}(n^{\mu_\sigma + o(1)} + n^{\omega(1,\sigma,1) + o(1)}) \\
&\le \mathcal{O}(n^{\mu_\sigma + o(1)}) \\
&= \mathcal{O}(n^{2+o(1)} + n^{1+\sigma+o(1)}) \ .
\end{aligned}$$

This yields the proof of Lemma 8. □

Acknowledgments. The authors are grateful to the anonymous referees for valuable comments.

REFERENCES

[1] N. Alon, R. Yuster, and U. Zwick, *Color-coding*, J. ACM, 42 (1995), pp. 844–856.

[2] N. Alon, R. Yuster, and U. Zwick, *Finding and counting given length cycles*, Algorithmica, 17 (1997), pp. 209–223.

[3] H. Cohn, R. Kleinberg, B. Szegedy, and C. Umans, *Group-theoretic algorithms for matrix multiplication*, in Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05), IEEE Press, Piscataway, NJ, 2005, pp. 379–388.

[4] D. Coppersmith, *Rectangular matrix multiplication revisited*, J. Symbolic Comput., 13 (1997), pp. 42–49.

[5] D. Coppersmith and S. Winograd, *Matrix multiplication via arithmetic progression*, J. Symbolic Comput., 9 (1990), pp. 251–290.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed., McGraw-Hill, Boston, MA, 2001.

[7] A. Czumaj, M. Kowaluk, and A. Lingas, *Faster algorithms for finding lowest common ancestors in directed acyclic graphs*, Theoret. Comput. Sci., 380 (2007), pp. 37–46.

[8] R. P. Dilworth, *A decomposition theorem for partially ordered sets*, Ann. of Math. (2), 51 (1950), pp. 161–166.

[9] F. Eisenbrand and F. Grandoni, *On the complexity of fixed parameter clique and dominating set*, Theoret. Comput. Sci., 326 (2004), pp. 57–67.

[10] M. R. Garey and D. S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, 1979.

[11] X. Huang and V. Y. Pan, *Fast rectangular matrix multiplications and applications*, J. Complexity, 14 (1998), pp. 257–299.

[12] A. Itai and M. Rodeh, *Finding a minimum circuit in a graph*, SIAM J. Comput., 7 (1978), pp. 413–423.

[13] M. Kowaluk and A. Lingas, *LCA queries in directed acyclic graphs*, in Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05), Lecture Notes in Comput. Sci. 3580, Springer, Berlin, 2005, pp. 241–248.

[14] T. Kloks, D. Kratch, and H. Müller, *Finding and counting small induced subgraphs efficiently*, Inform. Process. Lett., 74 (2000), pp. 115–121.

[15] J. Nešetřil and S. Poljak, *On the complexity of the subgraph problem*, Comment. Math. Univ. Carolin., 26 (1985), pp. 415–419.

[16] J. Plehn and B. Voigt, *Finding minimally weighted subgraphs*, in Proceedings of the 16th Workshop on Graph-Theoretic Concepts in Computer Science (WG'90), Lecture Notes in Comput. Sci. 484, Springer, Berlin, 1991, pp. 18–29.

[17] V. Vassilevska and R. Williams, *Finding a maximum weight triangle in $n^{3-\delta}$ time, with applications*, in Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC'06), ACM Press, New York, 2006, pp. 225–231.

[18] V. Vassilevska, R. Williams, and R. Yuster, *Finding the smallest H-subgraph in real weighted graphs and related problems*, in Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP'06), Lecture Notes in Comput. Sci. 4051, Springer, Berlin, 2006, pp. 262–273.

[19] R. Yuster and U. Zwick, *Detecting short directed cycles using rectangular matrix multiplication and dynamic programming*, in Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), ACM, New York, SIAM, Philadelphia, 2004, pp. 254–260.

[20] R. Yuster and U. Zwick, *Finding even cycles even faster*, SIAM J. Discrete Math., 10 (1997), pp. 209–222.