Finding a maximum-weight vertex-weighted triangle is not harder than matrix multiplication [*†]

Artur Czumaj
Department of Computer Science
University of Warwick
czumaj@dcs.warwick.ac.uk

Andrzej Lingas
Department of Computer Science
Lund University
Andrzej.Lingas@cs.lth.se

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 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 recently established by Vassilevska et al. (STOC 2006, $\mathcal{O}(n^{2.688})$) and (ICALP 2006, $\mathcal{O}(n^{2.575})$). Its asymptotic time complexity matches that of the fastest known algorithm for finding *a* triangle (not necessarily a maximum-weight one) in a graph.

By applying or extending our algorithm, we can also 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 established in the papers by Vassilevska et al. For example, 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 $\mathcal{O}(m^{1.41})$.

[*]A preliminary version of this paper appeared in the proceedings of the 18th annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2007).

[†]Research supported in part by NSF ITR grant CCR-0313219, by the *Centre for Discrete Mathematics and its Applications (DIMAP)*, University of Warwick, EPSRC grant EP/D063191/1, and by VR grant 621-2005-4085.

1 Introduction

We consider a classical graph problem of finding a fixed subgraph in a graph. The most basic version of that problem, that of finding a triangle (a cycle of length three, C_3), is related to the shortest path problem. It is well known that the asymptotic time complexity of finding a triangle in a graph does not exceed that of matrix multiplication (cf. [10]) [1], that is, $\mathcal{O}(n^{\omega})$, where $\omega < 2.376$ is the exponent of the fastest matrix multiplication algorithm [4] (see also [2]).

The more general problem of finding a maximum-weight triangle in a graph with vertex or edge weights has been widely open for long time. Interestingly, its variant with vertex weights can be regarded as a natural generalization of the 3-Sum problem and it is also related to a buyer-seller problem in computational economics [12].

The first substantially sub-cubic upper time-bound for finding a maximum-weight triangle in vertex weighted graphs has been established only recently by Vassilevska et al. in [12]. It has been later improved by Vassilevska et al. [13], who observed that the problem of finding a maximum weight triangle in a vertex-weighted graph immediately reduces to the problem of finding the so-called maximum witnesses of Boolean matrix product studied in [6, 11]. Hence, by the upper time-bound $\mathcal{O}(n^{2.616})$ for the latter problem established in [11] (improved to $\mathcal{O}(n^{2.575})$ by rectangular matrix multiplication in [6]), they could obtain the same upper time-bounds for finding a maximum-weight triangle in a vertex weighted graph.

Generally, the triangle detection problem and its weighted variants constitute the most basic instance of the subgraph isomorphism problem [8] and its weighted optimization variants , respectively. Typically, an improved upper time bound on such a triangle problem yields an improved upper time bound on the corresponding subgraph isomorphism problem (cf. [13]).

In this paper, 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 and on contrary, it strongly utilizes the fact that the output to the problem is a single triangle. By applying a recursive elimination scheme and 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 *a* 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). Previously, Vassilevska et al. [13] designed an algorithm that finds a maximum-weight triangle in time $\mathcal{O}(m^{\frac{18-4\omega}{13-3\omega}}) = \mathcal{O}(m^{1.45})$. 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 matches that of the fastest algorithm for finding *any* triangle in a graph, due to Alon et al. [1].

The problems of finding fixed cliques and more generally subgraphs isomorphic to fixed graph are natural generalizations of the problem of finding a triangle in a graph. In [12, 13], Vassilevska et al. considered the vertex-weighted variants of these problems, where the task is to find a maximum (or, equivalently minimum) weight subgraph isomorphic to a fixed graph. The weight of a subgraph

[1]For sparse graphs, in particular, planar graphs, there are known more efficient algorithms for triangle detection [10].

Problem	Source	Running-time	Numerical running-time/Comments
maximum-weight triangle	[12]	$\mathcal{O}(B \cdot n^{\frac{3+\omega}{2}})$	$\mathcal{O}(B \cdot n^{2.688})$
	[12]	$\mathcal{O}(n^{\frac{3+\omega}{2}} \log n)$	$\mathcal{O}(n^{2.688})$; randomized
	[13]	$\mathcal{O}(n^{2+1/(4-\omega)})$	$\mathcal{O}(n^{2.616})$
	[13]		$\mathcal{O}(n^{2.575})$
	this paper	$\mathcal{O}(n^{\omega^*})$	$\mathcal{O}(n^{2.376})$
maximum-weight triangle graph with m edges	[13]	$\mathcal{O}(m^{\frac{18-4\omega}{13-3\omega}})$	$\mathcal{O}(m^{1.45})$
	this paper	$\mathcal{O}(m^{\frac{2\omega^*}{1+\omega^*}})$	$\mathcal{O}(m^{1.41})$

Table 1: Summary of results for the problem of finding a maximum-weight triangle. In all results, n denotes the number of vertices, m 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 [4], and $\omega^* = \max\{\omega, 2 + o(1)\}$ (hence, $\omega^* < 2.376$).

is defined as the total weight of its vertices. Vassilevska et al. [13] obtained non-trivial time upper bounds for these variants by applying their algorithm for a maximum-weight triangle. We improve these bounds by using or extending our algorithm for a maximum-weight triangle.

First, we observe that our algorithm for finding a maximum-weight triangle can be easily extended to find a maximum-weight clique K_{3k} (or, in general, to find a maximum-weight fixed subgraph (either induced or not) with $3k$ vertices) in time $\mathcal{O}(n^{\omega^* k}) = \mathcal{O}(n^{2.376k})$, for any constant k. For 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 [13]) for all values of $h \geq 6$.

Our second algorithm uses fast rectangular matrix multiplication (instead of that for square matrices) and improves the running time even further (for values $h \bmod 3 \neq 0$). And so, if $h = 3f + 1$, 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 $h = 3f + 2$, 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 known result about $\omega(1, r, 1)$ [3, 4, 9], this yields in particular running times of $\mathcal{O}(n^{2.376f})$ for $h = 3f$, $\mathcal{O}(n^{2.376 \cdot f + 1})$ for $h = 3f + 1$, and $\mathcal{O}(n^{2.376f + 1.844})$ for $h = 3f + 2$.

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 H on h vertices in a vertex-weighted graph on n vertices. A 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 fastest algorithm for determining the existence of a subgraph isomorphic to H. By combining this method with that of Alon et al. [1] for detecting simple fixed cycles, we obtain better bounds on finding maximum-weight simple cycles on four and five vertices (C_4 and C_5) than those yielded by the aforementioned algorithm using rectangular matrix multiplication.

The bounds listed above subsume the corresponding ones from [13] (see Table 2).

Organization. In the next section, 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 maximum-weight triangle and extend it in order to derive upper time-bounds on finding maximum-weight cliques and subgraphs in a vertex weighted graph. In the next section, 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 maximum-weight triangle to include maximum-weight subgraph isomorphic to a fixed subgraph and present its applications.

2 Finding a maximum-weight triangle in $\mathcal{O}(n^{\omega^*})$ time

In this section, we present our recursive algorithm $\mathsf{HT}_\lambda(G, I, K, J)$ for finding a maximum-weight triangle in $\mathcal{O}(n^{\omega^*})$ time. It starts from a search region specified by three sets of vertices, each of size n and sorted in weight increasing order, where the maximum weight triangle is supposed to have precisely one vertex from each set. Letting λ to be a large integer (possibly depending on n), the region is divided into λ^3 subregions, where each subregion contains three sets of $\mathcal{O}(n/\lambda)$ vertices. The algorithm determines for each subregion whether or not it contains a triangle in time $\mathcal{O}((n/\lambda)^\omega)$. Even if $\Omega(\lambda^3)$ of the subregions can contain a triangle, only $\mathcal{O}(\lambda^2)$ among them can contain a maximum-weight triangle. These $\mathcal{O}(\lambda^2)$ subregions can be determined in $\mathcal{O}(\lambda^3)$ time. Our algorithm recurses on each of these $\mathcal{O}(\lambda^2)$ subregions. By choosing appropriately λ, the running time of our algorithm becomes $\mathcal{O}(n^{\omega^*})$.

The algorithm $\mathsf{HT}_\lambda(G, I, K, J)$ is presented in details on the next page. Now, we will begin with its analysis.

Lemma 1 *The procedure* HT_λ *is correct, that is, it returns a maximum-weight triangle* (i, j, k)*, if any, such that* $i \in I$, $j \in J$*, and* $k \in K$.

Proof. The correctness of the algorithm follows from the two following observations:

1. just before pruning T, $(p, r, q) \in T$ if and only if there is a triangle (i, k, j) with $i \in [i_0 + (p-1)\ell + 1, i_0 + p\ell]$, $k \in [k_0 + (r-1)\ell + 1, k_0 + r\ell]$, and $j \in [j_0 + (q-1)\ell + 1, j_0 + q\ell]$;
2. (p, r, q) is then removed from T if and only if there is a $(p', r', q') \in T$ with $p < p'$, $r < r'$, $q < q'$.

Indeed, the latter property implies that if there is a triangle (i, k, j) with $i \in [i_0 + (p-1)\ell + 1, i_0 + p\ell]$, $k \in [k_0 + (r-1)\ell + 1, k_0 + r\ell]$, and $j \in [j_0 + (q-1)\ell + 1, j_0 + q\ell]$, then the triple (p, q, r) representing such triangles (i, k, j) is removed from T only if there is another triangle (i', k', j') such that for some $(p', r', q') \in T$ with $p < p'$, $r < r'$, $q < q'$, we have $i' \in [i_0 + (p'-1)\ell + 1, i_0 + p'\ell]$, $k' \in [k_0 + (r'-1)\ell + 1, k_0 + r'\ell]$, and $j' \in [j_0 + (q'-1)\ell + 1, j_0 + q'\ell]$.

```
procedure $HT_\lambda(G, I, K, J)$

Input: A graph $G = (V, E)$ with vertex weights ($G$ is given as adjacency matrix)
       vertices are numbered in non-decreasing weight order from $1$ to $n$
       subintervals $I$, $K$, and $J$ of $[1, \ldots, n]$, of the same length $\kappa$ assumed to be a power of $\lambda$

Output: Maximum-weight triangle $(i, j, k)$, if any, such that $i \in I$, $j \in J$, and $k \in K$

  let $i_0$, $j_0$, and $k_0$ be such that
     $I = [i_0 + 1, i_0 + \kappa]$, $K = [k_0 + 1, k_0 + \kappa]$, and $J = [j_0 + 1, j_0 + \kappa]$
  $\ell = \kappa/\lambda$
  if $\kappa = 1$ then
    if $(i_0 + 1, j_0 + 1, k_0 + 1)$ is a (non-degenerate) triangle in $G$ then return $(i_0 + 1, j_0 + 1, k_0 + 1)$;
    stop
  for all $p, r = 1, \ldots, \lambda$ do
    form an $\ell \times \ell$ Boolean matrix $A_{pr}$ such that for every $1 \le i' \le \ell$ and $1 \le k' \le \ell$:
      $A_{pr}[i', k'] = \begin{cases} 1 & \text{if } (i_0 + (p-1)\ell + i', k_0 + (r-1)\ell + k') \in E \\ 0 & \text{otherwise.} \end{cases}$
  for all $r, q = 1, \ldots, \lambda$ do
    form an $\ell \times \ell$ Boolean matrix $B_{rq}$ such that for every $1 \le k' \le \ell$ and $1 \le j' \le \ell$:
      $B_{rq}[k', j'] = \begin{cases} 1 & \text{if } (k_0 + (r-1)\ell + k', j_0 + (q-1)\ell + j') \in E \\ 0 & \text{otherwise.} \end{cases}$
  for all $p, r, q = 1, \ldots, \lambda$ do
    compute $C^r_{pq} = A_{pr} \times B_{rq}$ (using the fast Boolean matrix multiplication algorithm)
  $T = \emptyset$
  for all $p, r, q = 1, \ldots, \lambda$ do
    for all $i', j' = 1, \ldots, \ell$ do
      if $C^r_{pq}(i', j') = 1$ and $(i_0 + (p-1)\ell + i', j_0 + (q-1)\ell + j') \in E$ then
        $T = T \cup \{(p, r, q)\}$
  { Observation 1: $(p, r, q) \in T$ iff there is a triangle $(i, k, j)$ with $i \in [i_0 + (p-1)\ell + 1, i_0 + p\ell]$,
                   $k \in [k_0 + (r-1)\ell + 1, k_0 + r\ell]$, and $j \in [j_0 + (q-1)\ell + 1, j_0 + q\ell]$ }
  for every $(p, r, q) \in T$ do
    if there is a $(p', r', q') \in T$ with $p < p'$, $r < r'$, $q < q'$ then
      remove $(p, r, q)$ from $T$
  { Observation 2: $(p, r, q)$ is removed from $T$ iff there is a $(p', r', q') \in T$ with $p < p'$, $r < r'$, $q < q'$}
  for every $(p, r, q) \in T$ do
    call $HT_\lambda(G, [i_0 + (p-1)\ell + 1, i_0 + p\ell], [k_0 + (r-1)\ell + 1, k_0 + r\ell], [j_0 + (q-1)\ell + 1, j_0 + q\ell])$
  Return the maximum-weight triangle among the triangles returned by these calls
```

It follows by the properties of the initial vertex numbering that then such a triangle (i, k, j) cannot be any maximum-weight triangle. $\square$

To estimate the running time of the procedure HT_λ we shall use 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 t' has each coordinate greater than the corresponding one in t. The cardinality of X is at most $3\lambda^2 - 3\lambda + 1$.*

Proof. Define the relation $\prec$ such that $(i, k, j) \prec (i', k', j')$ iff $i < i'$, $k < k'$, and $j < j'$. 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 $\mathtt{chain}((t_1, t_2, t_3))$ to be the set of all triples 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 $\mathtt{chain}(t)$ is indeed a chain in the poset $(\{1, 2, \ldots, \lambda\}^3, \prec)$ and the chains $\mathtt{chain}(t)$ cover all the elements in $\{1, 2, \ldots, \lambda\}^3$. It follows now from Dilworth's lemma [7] that the cardinality of the largest anti-chain in the aforementioned poset does not exceed the number of the triples with at least one coordinate equal to 1, which in turn, is at most $\lambda^3 - (\lambda - 1)^3 = 3\lambda^2 - 3\lambda + 1$. $\square$

Lemma 3 *The running time of HT_λ satisfies the recurrence*

$$\tau(\kappa) \le (3\lambda^2 - 3\lambda + 1) \cdot \tau(\kappa/\lambda) + \mathcal{O}(\lambda^{3-\omega} \cdot \kappa^\omega + \lambda^6) \ . \tag{1}$$

Proof. Forming the Boolean matrices A_{pr} and B_{rq}, computing their products C^r_{pq}, and computing T, take time $\mathcal{O}(\lambda^2 \cdot \ell^2)$, $\lambda^3 \cdot \mathcal{O}(\ell^\omega)$ and $\mathcal{O}(\lambda^3 \cdot \ell^2 + \lambda^6)$, respectively. By Lemma 2, the final size of T is at most $3\lambda^2 - 3\lambda + 1$. Hence, we obtain the following recurrence for the running time of HT_λ:

$$\begin{aligned} \tau(\kappa) &\le \mathcal{O}(\lambda^2 \cdot \ell^2) + \lambda^3 \cdot \mathcal{O}(\ell^\omega) + \mathcal{O}(\lambda^3 \cdot \ell^2 + \lambda^6) + (3\lambda^2 - 3\lambda + 1) \cdot \tau(\ell) \\ &= \mathcal{O}(\lambda^{3-\omega} \cdot \kappa^\omega + \lambda^6) + (3\lambda^2 - 3\lambda + 1) \cdot \tau(\kappa/\lambda) \ . \end{aligned}$$

$\square$

As an immediate corollary, 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. Let n be the initial size of the input instance and let $\lambda = \lambda(n) \ge 1$ be a (possibly constant) function of n. Then, Lemma 3 implies that there is a positive constant c such that for all N being powers of λ with $\lambda \le N \le n$:

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

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

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

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

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

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 $c^{\log n/\log\lambda} = n^{\log c/\log\lambda} = n^{\frac{\log c}{\log(2c)/\epsilon}} \leq n^\epsilon$ and

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

  Hence,

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

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

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

□

3 Improved bounds for sparse graphs

By arguing analogously as in the proof of Theorem 2 in [13], 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 $|V \setminus X| \leq 2m/\delta$. 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. 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. $\square$

4 Finding max-weight clique K_h and H-subgraphs

In this section, we present a construction that extends the algorithm for finding a maximum-weight triangle to include finding a maximum-weight clique or any maximum-weight subgraph isomorphic to a given graph.

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 an K_{2f} clique in G. Observe that G' has $\mathcal{O}(n^f)$ vertices and it 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 the 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, whereas 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 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. $\square$

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 case $H = K_h$, 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 super-graphs 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*

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

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

5 Refinement by using fast rectangular matrix multiplication

The algorithms and the bounds from Theorems 6 and 7 can be improved for $h \bmod 3 \neq 0$ if we use fast rectangular matrix multiplication algorithms (instead of fast square matrix multiplication).

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 need to generalize the procedure $HT_\lambda(G, I, K, J)$ to include the case where the sizes of the intervals I, K and J are not necessarily equal. We also relax the requirement that vertices are numbered in non-decreasing weight order by requiring solely that within each of the three input intervals I, K and J, the numbering has this property.

For any σ, $\frac{1}{2} \leq \sigma \leq 2$, let $HT_\lambda^{\langle\sigma\rangle}(G, I, K, J)$ denote such an analogous generalized procedure, where the sizes of I, K, and J (besides being powers of λ) satisfy $|I| = |J|$ and $|K| = |I|^\sigma$. The key difference in the body of $HT_\lambda^{\langle r\rangle}$ compared with that of HT_λ is that since the matrices A_{pr} and B_{rq} are now of sizes $\kappa/\lambda \times (\kappa/\lambda)^\sigma$ and $(\kappa/\lambda)^\sigma \times \kappa/\lambda$, respectively, we use the fast rectangular Boolean matrix multiplication algorithm with the exponent $\omega(1, \sigma, 1)$ instead of the fast square one with the exponent ω.

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

```
procedure $HT_\lambda^{\langle\sigma\rangle}(G, I, K, J)$
Input: A graph $G = (V, E)$ with vertex weights ($G$ is given as adjacency matrix)
        vertices are numbered in non-decreasing weight order from $1$ to $n$
        subintervals $I$, $K$, and $J$ of $[1, \dots, n]$, of the length $\kappa$, $\kappa^\sigma$, and $\kappa$, respectively,
                where $\kappa$, $\kappa^\sigma$, and $(\kappa/\lambda)^\sigma$ are powers of $\lambda$
Output: Maximum-weight triangle $(i, j, k)$, if any, such that $i \in I$, $j \in J$, and $k \in K$
  let $i_0$, $k_0$, and $j_0$ be such that $I = [i_0 + 1, i_0 + \kappa]$, $K = [k_0 + 1, k_0 + \kappa^\sigma]$, and $J = [j_0 + 1, j_0 + \kappa]$
  if $\kappa = 1$ then
    if $(i_0 + 1, j_0 + 1, k_0 + 1)$ is a (non-degenerate) triangle in $G$ then return $(i_0 + 1, j_0 + 1, k_0 + 1)$;
    stop
  for all $p = 1, \dots, \lambda$, $r = 1, \dots, \lambda^\sigma$ do
    form an $\frac{\kappa}{\lambda} \times (\frac{\kappa}{\lambda})^\sigma$ Boolean matrix $A_{pr}$ such that for every $1 \le i' \le \frac{\kappa}{\lambda}$ and $1 \le k' \le (\frac{\kappa}{\lambda})^\sigma$:
        $A_{pr}[i', k'] = \begin{cases} 1 & \text{if } (i_0 + (p-1)\frac{\kappa}{\lambda} + i', k_0 + (r-1)(\frac{\kappa}{\lambda})^\sigma + k') \in E \\ 0 & \text{otherwise.} \end{cases}$
  for all $r = 1, \dots, \lambda^\sigma$, $q = 1, \dots, \lambda$ do
    form an $(\frac{\kappa}{\lambda})^\sigma \times \frac{\kappa}{\lambda}$ Boolean matrix $B_{rq}$ such that for every $1 \le k' \le (\frac{\kappa}{\lambda})^\sigma$ and $1 \le j' \le \frac{\kappa}{\lambda}$:
        $B_{rq}[k', j'] = \begin{cases} 1 & \text{if } (k_0 + (r-1)(\frac{\kappa}{\lambda})^\sigma + k', j_0 + (q-1)\frac{\kappa}{\lambda} + j') \in E \\ 0 & \text{otherwise.} \end{cases}$
  for all $p, q = 1, \dots, \lambda$ and $r = 1, \dots, \lambda^\sigma$ do
    compute $C^r_{pq} = A_{pr} \times B_{rq}$ (using the fast rectangular Boolean matrix multiplication algorithm)
  $T = \emptyset$
  for all $p, q = 1, \dots, \lambda$ and $r = 1, \dots, \lambda^\sigma$ do
    for all $i', j' = 1, \dots, \frac{\kappa}{\lambda}$ do
      if $C^r_{pq}(i', j') = 1$ and $(i_0 + (p-1)\frac{\kappa}{\lambda} + i', j_0 + (q-1)\frac{\kappa}{\lambda} + j') \in E$ then
        $T = T \cup \{(p, r, q)\}$
   {Observation 1: $(p, r, q) \in T$ iff there is a triangle $(i, k, j)$ with $i \in [i_0 + (p-1)\frac{\kappa}{\lambda} + 1, i_0 + p\frac{\kappa}{\lambda}]$,
                   $k \in [k_0 + (r-1)(\frac{\kappa}{\lambda})^\sigma + 1, k_0 + r(\frac{\kappa}{\lambda})^\sigma]$, and $j \in [j_0 + (q-1)\frac{\kappa}{\lambda} + 1, j_0 + q\frac{\kappa}{\lambda}]$ }
  for every $(p, r, q) \in T$ do
    if there is a $(p', r', q') \in T$ with $p < p'$, $r < r'$, $q < q'$ then
      remove $(p, r, q)$ from $T$
   {Observation 2: $(p, r, q)$ is removed from $T$ iff there is a $(p', r', q') \in T$ with $p < p'$, $r < r'$, $q < q'$. }
  for every $(p, r, q) \in T$ do
    call $HT_\lambda^{\langle\sigma\rangle}(G, [i_0 + \frac{(p-1)\kappa}{\lambda} + 1, i_0 + \frac{p\kappa}{\lambda}], [k_0 + \frac{(r-1)\kappa^\sigma}{\lambda^\sigma} + 1, k_0 + \frac{r\kappa^\sigma}{\lambda^\sigma}], [j_0 + \frac{(q-1)\kappa}{\lambda} + 1, j_0 + \frac{q\kappa}{\lambda}])$
  return the maximum-weight triangle among the triangles returned by these calls
```

Lemma 8 *One choose the parameter* λ *to ensure that the procedure* $\mathrm{HT}_\lambda^{\langle\sigma\rangle}(I,K,J)$ *returns a maximum-weight triangle* (i,j,k)*, if any, where* $i \in I$*,* $j \in J$*, and* $k \in K$*, 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, but this time to compute the matrices C^r_{pq} we use fast rectangular matrix multiplication. Forming the Boolean matrices A_{pr} and B_{rq} takes time $\mathcal{O}(\lambda^{1+\sigma} \cdot \frac{\kappa}{\lambda} \cdot (\frac{\kappa}{\lambda})^\sigma) = \mathcal{O}(\kappa^{1+\sigma})$, computing their products C^r_{pq} takes time $\mathcal{O}(\lambda^{2+\sigma} \cdot (\kappa/\lambda)^{\omega(1,\sigma,1)})$, and computing T takes time $\mathcal{O}(\lambda^\sigma \cdot \kappa^2 + \lambda^{4+2\sigma})$. Using the arguments from Lemma 2, the final size of T 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 $\mathrm{HT}_\lambda^{\langle\sigma\rangle}$:

$$\begin{aligned}\tau(\kappa) &\leq \mathcal{O}(\kappa^{1+\sigma}) + \mathcal{O}(\lambda^{2+\sigma} \cdot (\kappa/\lambda)^{\omega(1,\sigma,1)}) + \mathcal{O}(\lambda^\sigma \cdot \kappa^2 + \lambda^{4+2\sigma}) + (2\lambda^{1+\sigma} + \lambda^2) \cdot \tau(\kappa/\lambda)\\ &\leq \mathcal{O}(\lambda^{2+\sigma}\kappa^{\omega(1,\sigma,1)} + \lambda^6) + (2\lambda^{1+\sigma} + \lambda^2)\,\tau(\kappa/\lambda)\ ,\end{aligned}$$

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

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 Appendix A) analogously as in the proof of Theorem 4 to conclude that the running time of the algorithm is $\tau(\kappa) = \mathcal{O}(\kappa^{\omega(1,\sigma,1)} + \kappa^{2+o(1)} + \kappa^{1+\sigma+o(1)})$. □

Equipped with Lemma 8, we can improve the bound form 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 non-decreasing weight order as well as those corresponding to the K_f cliques occur in a continuous interval in non-decreasing weight order.

Set I and J to the first of the aforementioned intervals and K to the second one, and run the procedure $H_\lambda^{\langle\sigma\rangle}(I,K,J)$ for G'' and $\sigma = \log_{|I|}|K|$, where $\frac{1}{2} \leq \sigma < 1$.[2] Observe that by Lemma 8 and by the definitions of G'', I, K and J, the triangle returned by $H_\lambda^{\langle\sigma\rangle}(I,K,J)$, 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, $H_\lambda^{\langle\sigma\rangle}(I,K,J)$ 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 J are set to the interval of vertices corresponding to K_f cliques whereas K 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.

[2]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.

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

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

Coppersmith [3] and Huang and Pan [9] proved the following facts.

Fact 11 [3, 9] *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 [9, 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 [9] 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, 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 weight increasing order, where the maximum weight subgraph isomorphic to H is supposed to have precisely one vertex from each set. Letting λ to be a large constant, the region is divided into λ^h subregions, where each 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.*

While this result is weaker than those from the previous two sections for $h \geq 6$, it leads to some interesting improvements for $h = 4, 5$. Alon et al. [1] showed that for any fixed h, if a graph on n vertices contains a copy of C_h (a simple cycle on exactly h vertices) then one such a copy can be detected in time $O(n^\omega \log n)$. This fact together with Theorem 13 yield the following corollary, which subsumes Theorem 10 for C_4 and C_5.

Corollary 14 *For $h \in \{4, 5\}$, if a vertex weighted graph on n vertices contains a copy of C_h then such a maximum-weight cycle C_h can be found in the graph in time $\mathcal{O}(n^{h-1+o(1)})$.*

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 whether or not our results for vertex weighted cliques are asymptotically optimal.

References

[1] N. Alon, R. Yuster, and U. Zwick. Color-coding. *Journal of the ACM*, 42: 844-856, 1995.

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

[3] D. Coppersmith. Rectangular matrix multiplication revisited. *Journal of Symbolic Computation*, 13: 42–49, 1997.

[4] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progression. *Journal of Symbolic Computation*, 9: 251–290, 1990.

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

[6] A. Czumaj, M. Kowaluk, and A. Lingas. Faster algorithms for finding lowest common ancestors in directed acyclic graphs. *Theoretical Computer Science*, Vol. 380 1-2 (21), June 2007, pp. 37-46. Also in *Electronic Colloquium on Computational Complexity (ECCC)*, Report No. 111, 2006.

[7] R. P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics* 51(1):161–166, January 1950.

[8] M.R. Garey, D.S. Johnson. Computers and Intractability. A Guide to the Theory of NP-completeness. Freeman, San Francisco, 1979.

[9] X. Huang and V. Y. Pan. Fast rectangular matrix multiplications and applications. *Journal of Complexity*, 14: 257–299, 1998.

[10] A. Itai and M. Rodeh. Finding a minimum circuit in a graph. *SIAM Journal on Computing*, 7(4): 413–423, 1978.

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

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

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

A Appendix: Solving the recurrence from Lemma 8

Let us remind that in the proof of Lemma 8, we want to show that for an appropriate choice of parameter λ, the solution to the recurrence

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

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

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:

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

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

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

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

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

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

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

Now, let us consider two cases.

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

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

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

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

This yields the proof of Lemma 8. □

Problem	Source	Running-time	Numerical runtime
maximum-weight fixed	[13]	$\mathcal{O}(n^{\omega+1})$	$\mathcal{O}(n^{3.376})$
subgraph with 4 vertices	*this paper*	$\mathcal{O}(n^{\omega(1,2,1)} + n^{3+o(1)})$	$\mathcal{O}(n^{3.334})$
maximum-weight C_4	*this paper*	$\mathcal{O}(n^{3+o(1)})$	$\mathcal{O}(n^{3+o(1)})$
maximum-weight fixed	[13]	$\mathcal{O}(n^{\omega+2})$	$\mathcal{O}(n^{4.376})$
subgraph with 5 vertices	*this paper*	$\mathcal{O}(n^{2\cdot\omega(1,\frac{1}{2},1)} + n^{4+o(1)})$	$\mathcal{O}(n^{4.220})$
maximum-weight C_5	*this paper*	$\mathcal{O}(n^{4+o(1)})$	$\mathcal{O}(n^{4+o(1)})$
maximum-weight fixed	[13]	$\mathcal{O}(n^{4+2/(4-\omega)})$	$\mathcal{O}(n^{5.232})$
subgraph with 6 vertices	*this paper*	$\mathcal{O}(n^{2\,\omega^*})$	$\mathcal{O}(n^{4.752})$
maximum-weight fixed	[13]	$\mathcal{O}(n^{4+3/(4-\omega)})$	$\mathcal{O}(n^{5.848})$
subgraph with 7 vertices	*this paper*	$\mathcal{O}(n^{2\,\omega(1,\frac{3}{2},1)} + n^{5+o(1)})$	$\mathcal{O}(n^{5.752})$
maximum-weight fixed	[13]	$\mathcal{O}(n^{2\,\omega+2} + n^{4+o(1)})$	$\mathcal{O}(n^{6.752})$
subgraph with 8 vertices	*this paper*	$\mathcal{O}(n^{3\,\omega(1,\frac{2}{3},1)} + n^{6+o(1)})$	$\mathcal{O}(n^{6.596})$
maximum-weight fixed	[13]	$\mathcal{O}(n^{2\,\omega+3})$	$\mathcal{O}(n^{7.752})$
subgraph with 9 vertices	*this paper*	$\mathcal{O}(n^{3\,\omega^*})$	$\mathcal{O}(n^{7.128})$
maximum-weight fixed	[12]	$\mathcal{O}(n^{\frac{(3+\omega)\,f}{2}})$; randomized	$\mathcal{O}(n^{2.688\,f})$
subgraph with $3f$ vertices	[13]		$\mathcal{O}(n^{2.575\cdot f})$
	this paper	$\mathcal{O}(n^{\omega^*\cdot f})$	$\mathcal{O}(n^{2.376\cdot f})$
maximum-weight fixed			
subgraph with $3f+1$ 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 with $3f+2$ vertices	*this paper*	$\mathcal{O}(n^{(f+1)\,\omega(1,\frac{f}{f+1},1)} + n^{(f+1)\,(2+o(1))})$	$\mathcal{O}(n^{2.376\cdot f+1.844})$

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, m 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 [4], $\omega^* = \max\{\omega, 2 + o(1)\}$ (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 [3, 9].