Conditional Lower Bounds for All-Pairs Max-Flow[*]

Robert Krauthgamer[1] and Ohad Trabelsi[2]

1 **Weizmann Institute of Science, Rehovot, Israel**
`robert.krauthgamer@weizmann.ac.il`

2 **Weizmann Institute of Science, Rehovot, Israel**
`ohad.trabelsi@weizmann.ac.il`

Abstract

We provide evidence that computing the maximum flow value between every pair of nodes in a directed graph on n nodes, m edges, and capacities in the range $[1..n]$, which we call the All-Pairs Max-Flow problem, cannot be solved in time that is faster significantly (i.e., by a polynomial factor) than $O(n^2m)$. Since a single maximum st-flow in such graphs can be solved in time $\tilde{O}(m\sqrt{n})$ [Lee and Sidford, FOCS 2014], we conclude that the all-pairs version might require time equivalent to $\tilde{\Omega}(n^{3/2})$ computations of maximum st-flow, which strongly separates the directed case from the undirected one. Moreover, if maximum st-flow can be solved in time $\tilde{O}(m)$, then the runtime of $\tilde{\Omega}(n^2)$ computations is needed. This is in contrast to a conjecture of Lacki, Nussbaum, Sankowski, and Wulf-Nilsen [FOCS 2012] that All-Pairs Max-Flow in general graphs can be solved faster than the time of $O(n^2)$ computations of maximum st-flow.

Specifically, we show that in sparse graphs $G = (V, E, w)$, if one can compute the maximum st-flow from every s in an input set of sources $S \subseteq V$ to every t in an input set of sinks $T \subseteq V$ in time $O((|S||T|m)^{1-\varepsilon})$, for some $|S|$, $|T|$, and a constant $\varepsilon > 0$, then MAX-CNF-SAT (maximum satisfiability of conjunctive normal form formulas) with n' variables and m' clauses can be solved in time $m'^{O(1)}2^{(1-\delta)n'}$ for a constant $\delta(\varepsilon) > 0$, a problem for which not even $2^{n'}/\mathsf{poly}(n')$ algorithms are known. Such runtime for MAX-CNF-SAT would in particular refute the Strong Exponential Time Hypothesis (SETH). Hence, we improve the lower bound of Abboud, Vassilevska-Williams, and Yu [STOC 2015], who showed that for every fixed $\varepsilon > 0$ and $|S| = |T| = O(\sqrt{n})$, if the above problem can be solved in time $O(n^{3/2-\varepsilon})$, then some incomparable (and intuitively weaker) conjecture is false. Furthermore, a larger lower bound than ours implies strictly super-linear time for maximum st-flow problem, which would be an amazing breakthrough.

In addition, we show that All-Pairs Max-Flow in *uncapacitated* networks with every edge-density $m = m(n)$, cannot be computed in time significantly faster than $O(mn)$, even for acyclic networks. The gap to the fastest known algorithm by Cheung, Lau, and Leung [FOCS 2011] is a factor of $O(m^{\omega-1}/n)$, and for acyclic networks it is $O(n^{\omega-1})$, where ω is the matrix multiplication exponent.

1998 ACM Subject Classification G.2.2 Graph Theory, F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases Conditional lower bounds, Hardness in P, All-Pairs Maximum Flow, Strong Exponential Time Hypothesis

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.20

[*] This work was partially supported by the Israel Science Foundation grant #897/13 and by a Minerva Foundation grant.

1 Introduction

The maximum flow problem is one of the most fundamental problems in combinatorial optimization. This classic problem and its variations such as minimum-cost flow, integral flow, and minimum-cost circulation, were studied extensively over the past decades, and have become key algorithmic tools with numerous applications, in theory and in practice. Moreover, techniques developed for flow problems were generalized or adapted to other problems, see for example [6, 3, 5]. The maximum *st*-flow problem, which we shall denote Max-Flow, asks to ship the maximum amount of flow from a source node s to a sink node t in a directed edge-capacitated graph $G = (V, E, w)$, where throughout, we denote $n = |V|$ and $m = |E|$, and assume integer capacities bounded by U. After this problem was introduced in 1954 by Harris and Ross (see [22] for a historical account), Ford and Fulkerson [12] devised the first algorithm for Max-Flow, which runs in time $O((n + m)F)$, where F is the maximum value of a feasible flow. Ever since, a long line of generalizations and improvements was studied, and the current fastest algorithm for Max-Flow with arbitrary capacities is by Lee and Sidford [20], which takes $O(m\sqrt{n} \log U)$ time. For the case of small capacities and sufficiently sparse graphs, the fastest algorithm, due to Mądry [21], has a running time $\tilde{O}(m^{10/7}U^{1/7})$. Here and throughout, $\tilde{O}(f)$ denotes $O(f \log^c f)$ for unspecified constant $c > 0$.

A very natural problem is to compute the maximum *st*-flow for multiple source-sink pairs in the same graph G. The seminal work of Gomory and Hu [14] shows that in undirected graphs, Max-Flow for all $\binom{n}{2}$ source-sink pairs requires at most $n - 1$ executions of Max-Flow (see also [15], where the $n - 1$ computations are all on the input graph), and a lot of research aimed to extend this result to directed graphs, with several partial successes, see details in Section 1.1. However, it is still not known how to solve Max-Flow for multiple source-sink pairs faster than solving it separately for each pair, even in special cases like a single source and all possible sinks. We shall consider the following problems involving multiple source-sink pairs, where the goal is always to report the value of each flow (and not an actual flow attaining it).

▶ **Definition 1.1.** (Single-Source Max-Flow) Given a directed edge-capacitated graph $G = (V, E, w)$ and a source node $s \in V$, output, for every $t \in V$, the maximum flow that can be shipped in G from s to t.

▶ **Definition 1.2.** (All-Pairs Max-Flow) Given a directed edge-capacitated graph $G = (V, E, w)$, output, for every pair of nodes $u, v \in V$, the maximum flow that can be shipped in G from u to v.

▶ **Definition 1.3.** (ST-Max-Flow) Given a directed edge-capacitated graph $G = (V, E, w)$ and two subsets of nodes $S, T \subseteq V$, output, for every pair of nodes $s \in S$ and $t \in T$, the maximum flow that can be shipped in G from s to t.

▶ **Definition 1.4.** (Global Max-Flow) Given a directed edge capacitated graph $G = (V, E, w)$, output the maximum among all pairs $u, v \in V$, of the maximum flow value that can be shipped in G from u to v.

▶ **Definition 1.5.** (Maximum Local Edge Connectivity) Given a directed graph $G = (V, E)$, output the maximum among all pairs $u, v \in V$, of the maximum number of edge-disjoint uv-paths in G.

Note that in a graph with all edge capacities equal to 1, the problem of finding the maximum local edge connectivity is equivalent to finding the global maximum flow.

Table 1 Known algorithms for multiple-pairs Max-Flow. In this table, $T(n,m)$ is the fastest time to compute maximum st-flow in an undirected graph, ω is the matrix multiplication exponent, and $\gamma = \gamma(G)$ is a topological property of the input network that varies between 1 and $\Theta(n)$. In planar graphs, γ is the minimum number of faces required to cover all the nodes (i.e., every node is adjacent to at least one such face) over all possible planar embeddings [13].

Directed	Class	Problem	Runtime	Reference
No	General	All-Pairs (G-H Tree)	$(n-1)T(n,m)$	[14]
No	Uncapacitated Networks	All-Pairs (G-H Tree)	$\tilde{O}(mn)$	[7]
No	Genus bounded by g	All-Pairs (G-H Tree)	$2^{O(g^2)} n \log^3 n$	[8]
Yes	Sparse	All-Pairs	$O(n^2 + \gamma^4 \log \gamma)$	[4]
Yes	Constant Treewidth	All-Pairs	$O(n^2)$	[4]
Yes	Uncapacitated	All-Pairs	$O(m^\omega)$	[11]
Yes	Uncapacitated DAG	Single-Source	$O(n^{\omega-1} m)$	[11]
Yes	Planar	Single-Source	$O(n \log^3 n)$	[19]

1.1 Prior Work

We start with undirected graphs, where the All-Pairs Max-Flow values can be represented in a very succint manner, called nowdays *a Gomory-Hu* tree [14]. In addition to being very succint, it allows the flow values and the corresponding cuts (vertex partitions) to be quickly retrieved. See Table 1 for a list of previous algorithms for multiple pairs maximum st-flow, see Table 1. For directed graphs, no current algorithm computes the maximum flow between any $k = \omega(1)$ given pairs of nodes faster than the time of $O(k)$ separate Max-Flow computations. However, some results are known in special settings. It is possible to compute Max-Flow for $O(n)$ pairs in the time it takes for a single Max-Flow computation [16] and this result is used to find a global minimum cut. However, these pairs cannot be specified in the input.

For directed planar graphs, there is an $O(n \log^3 n)$ time algorithm for the Single-Source Max-Flow problem [19], which immediately yields an $O(n^2 \log^3 n)$ time algorithm for the All-Pairs version, that is much faster than the time of $O(n^2)$ computations of planar Max-Flow, a problem that can be solved in time $O(n \log n)$ [9]. Based on these results, it was conjectured in [19] that also in general graphs, All-Pairs Max-Flow can be solved faster than the time required for computing $O(n^2)$ separate maximum st-flows.

Several hardness results are known for multiple-pairs variants of Max-Flow [2]. For ST-Max-Flow in sparse graphs ($m = O(n)$) and $|S| = |T| = O(\sqrt{n})$, there is an $n^{3/2-o(1)}$ lower bound assuming at least one of the Strong Exponential Time Hypothesis (SETH), 3SUM, and All-Pairs Shortest-Paths (APSP) conjectures is correct (for a comprehensive survey on them, see [23]). In addition, they show that Single-Source Max-Flow on sparse graphs requires $n^{2-o(1)}$ time, unless MAX-CNF-SAT can be solved in time $2^{(1-\delta)n}\mathsf{poly}(m)$ for some fixed $\delta > 0$, and in particular SETH is false.

We will mostly rely on SETH, a conjecture introduced by [17], and on some weaker assumption related to its maximization version, MAX-CNF-SAT. In more detail, SETH states that for every fixed $\varepsilon > 0$ there is an integer $k \geq 3$ such that k-SAT on n variables and m clauses cannot be solved in time $2^{(1-\varepsilon)n}\mathsf{poly}(m)$, where $\mathsf{poly}(m)$ refers to $O(m^c)$ for unspecified constant c. By the sparsification lemma [18], in order to refute SETH it can be assumed that the number of clauses is $O(n)$. The MAX-CNF-SAT problem asks for the maximum number of clauses that can be satisfied in an input CNF formula. Most of our conditional lower bounds are based on the assumption that for every fixed $\delta > 0$, MAX-CNF-SAT cannot be solved in time $2^{(1-\delta)n}\mathsf{poly}(m)$, where currently even $2^n/\mathsf{poly}(n)$ algorithms are not known for this problem [2]. Note that this is a weaker assumption than SETH, since a faster algorithm for MAX-CNF-SAT would imply a faster algorithm for CNF-SAT and refute SETH. Different assumptions regarding the hardness of CNF-SAT have been the basis for many lower bounds, including for the runtime of solving NP-hard problems exactly, parametrized complexity, and problems in P. See the Introduction in [1] and the references therein.

1.2 Our Contribution

We present conditional runtime lower bounds for both uncapacitated and capacitated networks. The proofs appear in sections 2 and 3, respectively, where the order reflects increasing level of complication. All our lower bounds hold even when the input G is a DAG and has a constant diameter, and in the case of general capacities, they can be easily modified to apply also for graphs with constant maximum degree. In addition, for integer $k \geq 1$ we use the notation $[k]$ to denote the range $\{1, ..., k\}$.

Capacitated Networks

Our main result is that for every set sizes $|S|$ and $|T|$, the ST-Max-Flow cannot be solved significantly faster than $O(|S||T|m)$ (i.e., polynomially smaller runtime), unless a breakthrough in MAX-CNF-SAT is achieved, and consequently in SETH.

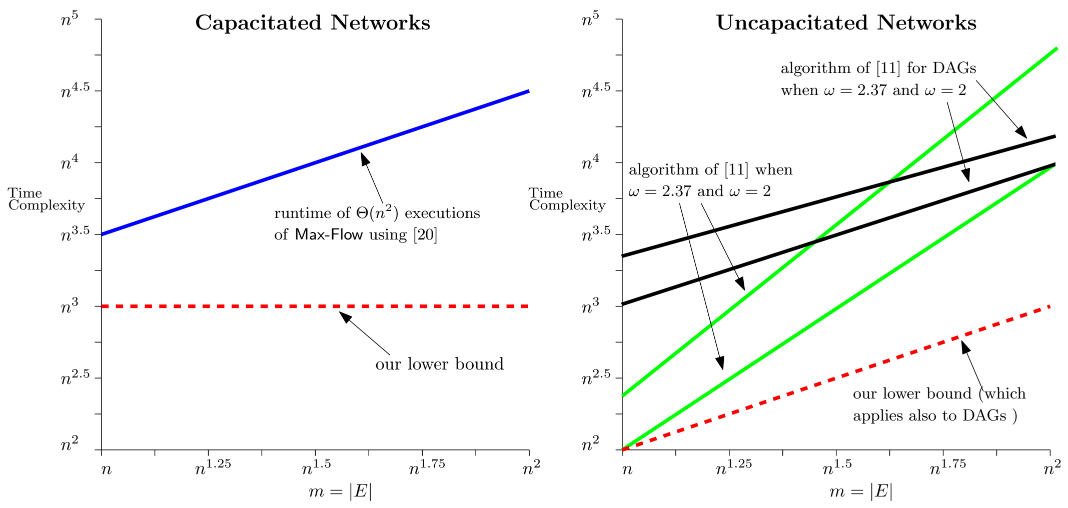
► **Theorem 1.6.** *If for some fixed $\varepsilon > 0$ and some (possibly functions of n) set sizes $|S|$ and $|T|$, ST-Max-Flow can be solved in graphs with n nodes, $m = O(n)$ edges and capacities in $[n]$ in time $O((|S||T|m)^{1-\varepsilon})$, then for some $\delta(\varepsilon) > 0$, MAX-CNF-SAT on n' variables and $O(n')$ clauses can be solved in time $2^{(1-\delta)n'}\mathsf{poly}(n')$, and in particular SETH is false.*

This result improves the aforementioned $n^{3/2-o(1)}$ lower bound of [2], as for their setting of $|S| = |T| = O(\sqrt{n})$ our lower bound is $n^{2-o(1)}$, although their lower bound is based on an incomparable (and intuitively weaker) conjecture, that at least one of the SETH, 3SUM, and APSP conjectures is correct. In fact, if there was a reduction from SETH that implied a larger runtime lower bound for ST-Max-Flow, then the (single-pair) Max-Flow problem would require a strictly super-linear time under it, but such a reduction is not possible unless the non-deterministic version of SETH (abbreviated NSETH) is false [10]. And anyway, such a lower bound for Max-Flow would be an amazing breakthrough.

The next theorem is an immediate corollary of Theorem 1.6, by assigning $|S|, |T| = \Theta(n)$.

► **Theorem 1.7.** *If for some fixed $\varepsilon > 0$, All-Pairs Max-Flow in graphs with n nodes, $m = O(n)$ edges, and capacities in $[n]$ can be solved in time $O((n^2m)^{1-\varepsilon})$, then for some $\delta(\varepsilon) > 0$, MAX-CNF-SAT on n' variables and $O(n')$ clauses can be solved in time $2^{(1-\delta)n'}\mathsf{poly}(n')$, and in particular SETH is false.*

This conditional lower bound (see Figure 1) shows that All-Pairs Max-Flow requires time that is equivalent to $\Omega(n^{3/2})$ computations of Max-Flow, which strongly separates the directed case from the undirected one (where a Gomory-Hu tree can be constructed in the time of $n-1$ computations). If Max-Flow takes $\tilde{O}(m)$ time, which is currently open but plausible, then the running time of $\tilde{\Omega}(n^2)$ computations of Max-Flow is needed. This is in contrast to the aforementioned conjecture of Lacki, Nussbaum, Sankowski, and Wulf-Nilsen [19] that All-Pairs Max-Flow in general graphs can be solved faster than the time of $O(n^2)$ computations of maximum st-flow.



■ **Figure 1** State of the art bounds for All-Pairs Max-Flow in directed networks. Conditional lower bounds are depicted in dashed lines, and known algorithms in solid lines.

Uncapacitated Networks

For the case of uncapacitated networks, we show that for every $m = m(n)$, All-Pairs Max-Flow cannot be solved significantly faster than $O(mn)$. Here we introduce a new technique to design reductions from SETH to graphs with varying edge densities, rather than the usual reductions that only deal with sparse graphs. Our technique is based on partitioning the variables set of CNF-SAT to different sizes.

▶ **Theorem 1.8.** *If for some fixed $\varepsilon > 0$ and some $m = m(n) \in [n, n^2]$, All-Pairs Max-Flow in uncapacitated graphs with n nodes and m edges can be solved in time $O((nm)^{1-\varepsilon})$, then for some $\delta(\varepsilon) > 0$, MAX-CNF-SAT on n' variables and $O(n')$ clauses can be solved in time $2^{(1-\delta)n'}\mathsf{poly}(n')$, and in particular SETH is false.*

Hence, a certain additional improvement to the $O(m^\omega)$ time algorithm of [11] (and similarly to the $O(n^\omega m)$ time for DAGs, where our lower bounds apply too) is not likely. We now present conditional lower bounds for ST-Max-Flow, which are functions of $|S|$ and $|T|$.

▶ **Theorem 1.9.** *If for some fixed $\varepsilon > 0$ and some (possibly functions of n) set sizes $|S|$ and $|T|$, ST-Max-Flow on uncapacitated graphs with n nodes and $O((|S| + |T|)n)$ edges can be solved in time $O((|S||T|n)^{1-\varepsilon})$, then for some $\delta(\varepsilon) > 0$, MAX-CNF-SAT on n' variables and $O(n')$ clauses can be solved in time $2^{(1-\delta)n'}\mathsf{poly}(n')$, and in particular SETH is false.*

Finally, we present a conditional lower bound for computing the Maximum Local Edge Connectivity of a sparse graph, which is the same as Global Max-Flow if all the capacities are 1, which is indeed the case in our reduction. In the Orthogonal Vectors problem the input is two sets U and V, each of n vectors from $\{0,1\}^d$, and the goal is to determine whether there are $u \in U$ and $v \in V$ such that $\sum_{i=1}^{d} u_i \cdot v_i = 0$. An equivalent version of the problem has $U = V$. For $d = \omega(\log n)$, Williams [24] proved that SETH implies the non-existence of a truly subquadratic (in n) algorithm for the problem. The next result, proved in Section 4, was obtained together with Bundit Laekhanukit and Rajesh Chitnis, and we thank them for their permission to include it here.

▶ **Theorem 1.10.** *If for some fixed $\varepsilon > 0$, the Maximum Local Edge Connectivity in graphs with n nodes and $\tilde{O}(n)$ edges can be found in time $O(n^{2-\varepsilon})$, then for some $\delta(\varepsilon) > 0$, the Orthogonal Vectors problem on n' vectors and every $d = \mathsf{polylog}(n')$ can be solved in time $O(n'^{2-\delta})$, and in particular SETH is false.*

2 Reduction to Multiple-Pairs Max-Flow with Unit Capacity

In this section we prove Theorems 1.8 and 1.9. We start with a general lemma which is the heart of the proofs.

▶ **Lemma 2.1.** *Let $a \in [0, 1]$ and $b \in [0, 1 - a]$. Then MAX-CNF-SAT on n variables and m clauses can be reduced to $O(m)$ instances of ST-Max-Flow with $|S| = 2^{an}$ and $|T| = 2^{bn}$ in graphs with $\Theta(2^{an} + 2^{(1-a-b)n}m + 2^{bn})$ nodes, $\Theta((2^{an} + 2^{bn}) \cdot 2^{(1-a-b)n}m)$ edges, and capacities in $\{0, 1\}$.*

Proof. Given a CNF-formula F on n variables and m clauses $\{C_i\}_{i\in[m]}$ as input for MAX-CNF-SAT, $a \in [0, 1]$, and $b \in [0, 1 - a]$, we split the variables into three sets U_1, U_2, and U_3, where U_1 is of size an, U_2 is of size $(1 - a - b)n$, and U_3 is of size bn, and enumerate all their 2^{an}, $2^{(1-a-b)n}$, and 2^{bn} partial assignments (with respect to F), respectively, when the objective is to find a triple α, β, γ of assignments to U_1, U_2, and U_3 respectively, that satisfies the maximal number of clauses. We will have an instance G_p of ST-Max-Flow for each value $p \in [m]$, in which by one call to ST-Max-Flow we check if there exists a triple α, β, and γ that satisfies at least p clauses, as follows.

We construct a graph G_p for every $p \in [m]$ on N nodes $V_1 \cup V_2 \cup V_3$, where V_1 contains a node α for every assignment α to U_1, V_2 contains $2m + 1 + (p - 1) = 2m + p$ nodes for every assignment β to U_2, that are β_i^l and β_i^r for every $i \in [m]$, β', and the set $\{\beta'^i\}_{i\in[p-1]}$, and V_3 contains a node γ for every assignment γ to U_3. We use the notation α for nodes in V_1 and for assignments to U_1, β for assignments to U_2, and γ for nodes in V_3 and assignments to U_3. However, it will be clear from the context. Now, we have to describe the edges in the network. In order to simplify the reduction, we partition the edges into blue and red colors, as follows.

For every α, β, and $i \in [m]$, we add a blue edge from α to β_i^l if both of α and β do not satisfy the clause C_i (do not set any of the literals to true), and otherwise we add a red edge from α to β_i^r. We further add, for every β, γ, and $i \in [m]$, a blue edge from β_i^l to γ if γ does not satisfy C_i. For every β, γ, and $j \in [p - 1]$, we add a red edge from every β'^j to every γ. For every β and $i \in [m]$, we add a red edge from β_i^l to β_i^r and from β_i^r to β', and finally for every β and $j \in [p - 1]$, we add a red edge from β' to β'^j, where all edges are of capacity 1.

The graph we built has $2^{an} + 2 \cdot 2^{(1-a-b)n}m + 2^{(1-a-b)n} + 2^{(1-a-b)n}(p - 1) + 2^{bn} = \Theta(2^{an} + 2^{(1-a-b)n}m + 2^{bn})$ nodes, $2^{an} \cdot 2^{(1-a-b)n}m + 2^{bn} \cdot 2^{(1-a-b)n}m + 2 \cdot 2^{(1-a-b)n}m + (p-1)2^{(1-a-b)n} + 2^{bn} \cdot (p-1)2^{(1-a-b)n} = \Theta((2^{an} + 2^{bn}) \cdot 2^{(1-a-b)n}m)$ edges, with capacities in $\{0, 1\}$ (see Figure 2), and its construction time is asymptotically the same as the time it takes to construct its edges set.

For every α, β, and γ, we denote by $G_p^{\alpha,\beta,\gamma}$ the graph induced from G_p on the nodes

$$(\alpha, \beta', \gamma) \cup \left(\bigcup_{\substack{y\in\{l,r\} \\ i\in[m]}} \beta_i^y \right) \cup \left(\bigcup_{j\in[p-1]} \beta'^j \right).$$

We claim that for every α and γ, the maximum flow from α to γ can be bounded by the sum, over all β, of the maximum flow between them in $G_p^{\alpha,\beta,\gamma}$. This claim follow easily

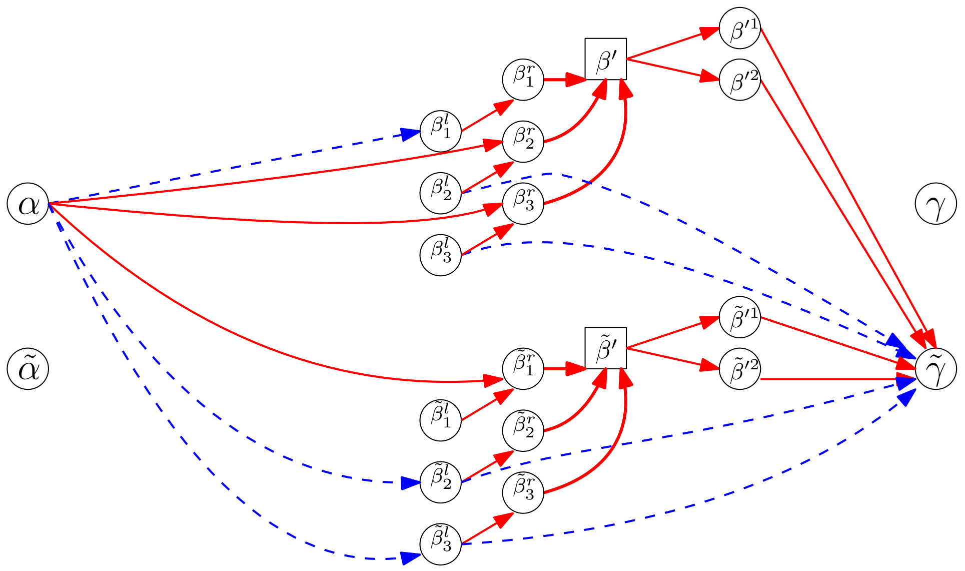


Figure 2 An illustration of part of the reduction. Here, U_1, U_2, and U_3 have 2 assignments each, α and $\tilde{\alpha}$ to U_1, β and $\tilde{\beta}$ to U_2, and γ and $\tilde{\gamma}$ to U_3. Blue edges are dashed. For simplicity, only the edges of $G_3^{\alpha,\beta,\tilde{\gamma}} \cup G_3^{\alpha,\tilde{\beta},\tilde{\gamma}}$ are presented. In this illustration, α does not satisfy anything, β satisfies C_2 and C_3, $\tilde{\beta}$ satisfies C_1, and $\tilde{\gamma}$ satisfies C_1. Note that the assignment comprised of α, β, and $\tilde{\gamma}$ satisfies all the clauses, and indeed the maximum flow from α to γ is $2 \cdot 3 - 1 = 5$.

because the intersection $G_p^{\alpha,\beta_1,\gamma} \cap G_p^{\alpha,\beta_2,\gamma}$ for $\beta_1 \neq \beta_2$ is exactly the source and the sink $\{\alpha,\gamma\}$, no edge passes between these two graphs, and $\left(\bigcup_\beta G_i^{\alpha,\beta,\gamma}\right)$ consists of all nodes that are both reachable from α and γ is reachable from them.

We now prove that if there is an assignment to F that satisfies at least p clauses then the graph G_p we built has a triple α, β, γ with maximum flow from α to γ in $G_p^{\alpha,\beta,\gamma}$ at most $m-1$. Since for every $\tilde{\beta}$, m is the number of outgoing edges from α in $G_p^{\alpha,\tilde{\beta},\gamma}$, m is also an upper bound for the maximum flow from α to γ in it, and hence in G_p it is at most $2^{(1-a-b)n}m - 1$. Otherwise, we will show that every triple α, β, γ has a maximum flow from α to γ in $G_p^{\alpha,\beta,\gamma}$ of size at least m, and so in G_p it is at least $2^{(1-a-b)n}m$. Hence, by simply picking the maximal $j \in [m]$ such that the maximum flow in G_j of some pair α, γ is at most $2^{(1-a-b)n}m - 1$, and then iterating over all assignments β to U_2 with α and γ fixed as the assignments to U_1 and U_3, we can also find the required triple α, β, γ.

For the first direction, assume that F has an assignment that satisfies at least p clauses, and denote such assignment by Φ. Let α_Φ, β_Φ, and γ_Φ be the assignments to U_1, U_2, and U_3, respectively, that are induced from Φ. Since a blue path from α_Φ through ${\beta_\Phi}_i^l$ for some $i \in [m]$ to γ_Φ corresponds to α_Φ, β_Φ, and γ_Φ all do not satisfy C_i, in $G_p^{\alpha_\Phi,\beta_\Phi,\gamma_\Phi}$ there are at most $m-p$ (internally) disjoint blue paths from α to γ. As the only way to ship flow in $G_p^{\alpha_\Phi,\beta_\Phi,\gamma_\Phi}$ that is not through a blue path is through the node β'_Φ, and the total number of edges going out of this node is $p-1$, we conclude that the total maximum flow in $G_p^{\alpha_\Phi,\beta_\Phi,\gamma_\Phi}$ from α_Φ to β_Φ is bounded by $m - p + (p-1) = m-1$. Since for every β, the maximum amount of flow that can be shipped in $G_p^{\alpha_\Phi,\beta,\gamma_\Phi}$ from α_Φ to γ_Φ is at most m, summing over all β we get that the total flow in G_p from α_Φ to γ_Φ is bounded by $(2^{(1-a-b)n} - 1)m + (m-1) \leq 2^{(1-a-b)n}m - 1$, as required.

For the second direction, assume that every assignment to F satisfies at most $p-1$ clauses. In order to show that the maximum flow from every α to every γ is at least $2^{(1-a-b)n}m$, we first fix α, β, and γ. Then, by passing flow in two phases we show that m units of flow can be passed in $G_p^{\alpha,\beta,\gamma}$ from α to γ. As this argument applies for every β, we can add up the respective flows without violating capacities, concluding the proof. By the assumption, there exist $m-(p-1) = m-p+1$ i's, such that α, β, and γ do not satisfy C_i, and we denote a set with this amount of such i's by I_β. Each of these i's induces a blue path $(\alpha \to \beta_i^l \to \gamma)$ from α to γ in $G_p^{\alpha,\beta,\gamma}$, and so we ship a unit of flow through every one of them according to I_β, in what we call the first phase. In the second phase, we ship additional $m-(m-p+1) = p-1$ units in the following way. Let $A_1 := \{i \in [m] \setminus I_\beta : \alpha \nvDash C_i \wedge \beta \nvDash C_i\}$, and $A_2 := ([m] \setminus I_\beta) \setminus A_1 = \{i \in [m] \setminus I_\beta : \alpha \vDash C_i \vee \beta \vDash C_i\}$, where $\alpha \vDash C_i$ denotes that the assignment α satisfies C_i (as defined earlier), and $\alpha \nvDash C_i$ denotes that it does not satisfy C_i. Let $f : A_1 \cup A_2 \to [m-|I_\beta|]$ be a bijective function such that the range of A_1 is $[|A_1|]$ and the range of A_2 is $[m-|I_\beta|] \setminus [|A_1|]$. Clearly, there exists such bijection and it is easy to find one. For every $i \in A_1$ we ship flow through the path $(\alpha \to \beta_i^l \to \beta_i^r \to \beta' \to \beta'^j \to \gamma)$, and for every $i \in A_2$ through the path $(\alpha \to \beta_i^r \to \beta' \to \beta'^j \to \gamma)$, in both cases with $j = f(i)$.

Since we defined the flow in paths, we only need to show that the capacity requirements hold, and we start with blue edges. Indeed, edges of the form (α, β_i^l) are used in the first phase, with flow that is determined uniquely by β and $i \in I_\beta$, and in the second phase uniquely according to β and $i \in [m] \setminus I_\beta$, and so they cannot be used twice. Edges of the form (β_i^l, γ) are only used in the first phase, and their flow is uniquely determined according to β and $i \in I_\beta$, and so are good too. We now proceed to red edges, which were used only in the second phase.

Edges of the forms (α, β_i^r), (β_i^l, β_i^r) and (β_i^r, β') have flow that is uniquely determined by β and $i \in [m] \setminus I_\beta$, and so are not used more than once. Edges of the form (β', β'^j) have flow that is uniquely determined by β and $j = f(i) \in [p-1]$, and since f is a bijection, every j has at most one i such that $f(i) = j$, and so these edges are also used at most once. As a byproduct, and since every edge of the form (β'^j, γ) has only the edge (β', β'^j) as its source for flow, edges of the form (β'^j, γ) are also used at most once. Altogether, we have bounded the total flow in all edges that were used in both phases, and so the capacity requirements follow, which completes the proof of the second direction and of Lemma 2.1. ◀

Proof of Theorem 1.8. We apply Lemma 2.1 in the following way. By setting $a = b \in [1/3, 1/2]$ we get graphs $G = (V, E, w)$ with $|V| = \Theta(2^{an})$ ($|V| = \Theta(2^{an})m$ if $a = 1/3$) and $|E| = 2^{(1-a)n}m$. Hence, $|E| = O(|V|^{1/a-1})$ and we get our desired bound for every $|E|$ between $|V|$ and $|V|^2$ and Theorem 1.8 follows. ◀

Proof of Theorem 1.9. Here we apply Lemma 2.1 a bit differently. By setting $a, b \leq 1/3$ we get graphs $G = (V, E, w)$ with $|V| = \Theta(2^{(1-a-b)n}m)$ and $|E| = \Theta((2^{an} + 2^{bn})2^{(1-a-b)n}m)$. By setting $|S| = |V|^{a/(1-a-b)}$ and $|T| = n^{b/(1-a-b)}$ we get our lower bound for $|E| = O((|S| + |T|)|V|)$ and Theorem 1.9 follows. ◀

3 Reduction to Multiple-Pairs Max-Flow in Capacitated Networks

In this section we prove Theorems 1.6 and 1.7. We proceed to prove our main technical lemma.

▶ **Lemma 3.1.** *MAX-CNF-SAT on n variables and m clauses $\{C_i\}_{i\in[m]}$ can be reduced to $O(m)$ instances of ST-Max-Flow, each with the property that $S \cap T = \emptyset$, and all of them are in graphs with $N = \Theta(2^{n/3}m)$ nodes, $O(2^{n/3}m) = O(N)$ edges, and with capacities in $[N]$.*

Proof. Given a CNF-formula F on n variables and m clauses as input for MAX-CNF-SAT, we split the variables into three sets U_1, U_2, and U_3 of size $n/3$ each and enumerate all $2^{n/3}$ partial assignments (with respect to F) to each of them, when the objective is to find a triple (α, β, γ) of assignments to U_1, U_2, and U_3, that satisfy the maximal number of clauses. We will have an instance G_p of ST-Max-Flow with the mentioned property for each value $p \in [m]$, in which by one call to ST-Max-Flow we check if there exists a triple (α, β, γ) that satisfies at least p clauses, as follows.

We construct the graph G_p on N nodes $V_1 \cup V_2 \cup V_3 \cup A \cup B \cup \{v_B\}$, where V_1 contains a node α for every assignment α to U_1, V_2 contains $3m+1$ nodes for every assignment β to U_2, that are β_i^l, β_i^c, β_i^r, for every $i \in [m]$, and β', V_3 contains a node γ for every assignment γ to U_3, A contains two nodes $C_i^{\vDash}$ and $C_i^{\nvDash}$ for every clause C_i, and B contains a node C_i for every clause C_i. We use the notation α for nodes in V_1 and assignments to U_1, β to assignments to U_2, γ for nodes in V_3 and assignments to U_3, and C_i for nodes in B and clauses. However, it will be clear from the context. Now, we have to describe the edges in the network. In order to simplify the reduction, we partition the edges into red and blue colors, as follows.

For every α and $i \in [m]$ we add a red edge of capacity $2^{n/3}$ from α to $C_i^{\vDash}$ if $\alpha \vDash C_i$, and a blue edge of the same capacity from α to $C_i^{\nvDash}$ otherwise. We further add, for every β, a red edge of capacity 1 from $C_i^{\vDash}$ to β_i^c, a blue edge of capacity 1 from $C_i^{\nvDash}$ to β_i^l, a blue edge of capacity 1 from β_i^l to β_i^r if $\beta \nvDash C_i$, a red edge of capacity 1 from β_i^c to β', and a blue edge of capacity 1 from β_i^r to C_i. For every β we add a red edge of capacity $p-1$ from β' to v_B. For every γ we add a red edge of capacity $2^{n/3}(p-1)$ from v_B to $\gamma \in V_3$, and finally, for every γ and $i \in [m]$ we add a blue edge of capacity $2^{n/3}$ from C_i to γ if $\gamma \nvDash C_i$.

The graph we built has $N = 2^{n/3} + 2m + 2^{n/3} \cdot 3m + 2^{n/3} + 1 + m + 2^{n/3} = \Theta(2^{n/3}m)$ nodes, at most $2^{n/3}m + 2^{n/3} \cdot 2m + 2^{n/3} \cdot 2m + 2^{n/3}m + 2^{n/3} + 1 + 2^{n/3}m + 2^{n/3}m = O(2^{n/3}m)$ edges, all of its capacities are in $[N]$, and its construction time is $O(Nm)$ (see Figure 3).

We proceed to prove that if there is an assignment to F that satisfies at least p clauses then the graph G_p we built has a pair α, γ with maximum flow from α to γ at most $2^{n/3}m - 1$, and otherwise, every α, γ has a maximum flow of size at least $2^{n/3}m$. Hence, by simply picking the maximal $j \in [m]$ such that the maximum flow in G_j of some pair α, γ is at most $2^{n/3}m - 1$, and then iterating over all assignments β to U_2 with α and γ fixed as the assignments to U_1 and U_3, we can also find the required triple α, β, γ.

For the first direction, assume that F has an assignment that satisfies at least p clauses, and denote such assignment by Φ. Let α_Φ, β_Φ, and γ_Φ be the assignments to U_1, U_2, and U_3, respectively, that are induced from Φ. We will show that there exists an $(\alpha_\Phi, \gamma_\Phi)$ cut whose capacity is at most $2^{n/3}m - 1$, hence by the Min-Cut Max-Flow theorem, the maximum flow from α_Φ to γ_Φ is bounded by this number, concluding the proof of the first direction. We define the cut in a way that for every $\beta \neq \beta_\Phi$, the cut will have m cut edges that are contributed from nodes related to β, and nodes related to β_Φ will be carefully added to either side of the cut so that they will contribute capacity of only $m-1$ to the cut. To be more precise, we define a suitable cut as follows.

$$\begin{aligned} S = &\{\alpha_\Phi, \beta'_\Phi\} \cup \{C_i^{\vDash} : \alpha_\Phi \vDash C_i\} \cup \{C_i^{\nvDash} : \alpha_\Phi \nvDash C_i\} \cup \{{\beta_\Phi}_i^c : i \in [m]\} \\ &\cup \{C_i, {\beta_\Phi}_i^l, {\beta_\Phi}_i^r : \gamma_\Phi \vDash C_i\} \cup \{{\beta_\Phi}_i^l : \gamma_\Phi \nvDash C_i \wedge \beta_\Phi \vDash C_i\} \end{aligned}$$

▶ **Claim 3.2.** *The cut $(S, V \setminus S) = (S, T)$ has capacity $2^{n/3}m - 1$.*

Proof of Claim. We will go over all the nodes in S, and count the total capacity leaving to nodes in T for each of them. $\alpha_\Phi \in S$ and all nodes $C_i^{\vDash}$ and $C_i^{\nvDash}$ that are adjacent to it are in S too, hence it does not contribute anything. For every $i \in [m]$, we have two cases for

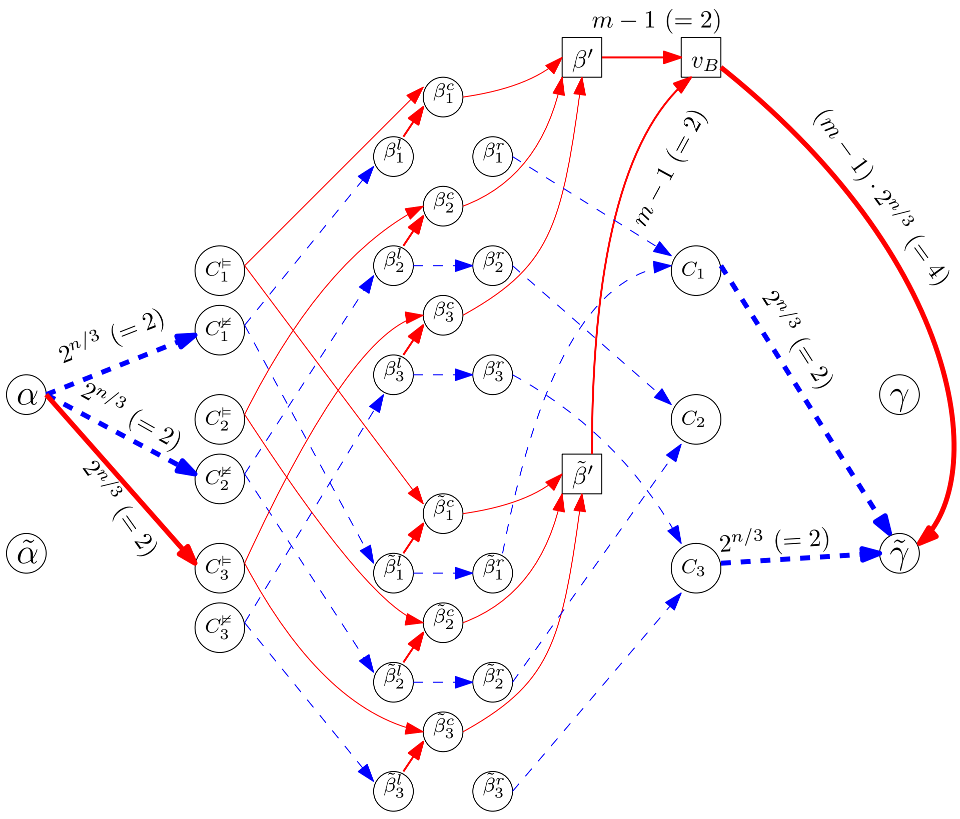


Figure 3 An illustration of part of the reduction, with $p = m$. Here, U_1, U_2, and U_3 have 2 assignments each; α and $\tilde{\alpha}$ to U_1, β and $\tilde{\beta}$ to U_2, γ and $\tilde{\gamma}$ to U_3. Bolder edges correspond to edges of higher capacity (specified wherever they are bigger than 1), and blue edges are dashed. For simplicity, only the edges relevant to α and $\tilde{\gamma}$ are presented. In this illustration, α satisfies C_3, β satisfies C_1, $\tilde{\beta}$ satisfies C_3, and $\tilde{\gamma}$ satisfies C_2. Note that the assignment comprised of α, β, and $\tilde{\gamma}$ satisfies all the clauses, and indeed the maximum flow from α to γ is $2 \cdot 3 - 1 = 5$.

nodes in A. If $\alpha_\Phi \models C_i$ then $C_i^{\not\models} \in T$ and hence $C_i^{\not\models}$ does not contribute anything. However, $C_i^{\models}$ has $2^{n/3}$ outgoing edges, where all except ${\beta_\Phi}_i^c$ are in T. Hence, it contributes $2^{n/3} - 1$ to the cut. Else, if $\alpha_\Phi \not\models C_i$ then $C_i^{\models} \in T$ and hence $C_i^{\models}$ does not contribute anything. But $C_i^{\not\models}$ has $2^{n/3}$ outgoing edges, of which $2^{n/3} - 1$ are cut edges as their targets are in T, and the one incoming to ${\beta_\Phi}_i^l$ is a cut edge if and only if ${\beta_\Phi}_i \not\models C_i$ and also $\gamma_\Phi \not\models C_i$ (equivalently, ${\beta_\Phi}_i^l \in T$), and in our current case it means that $\Phi \not\models C_i$. Hence, for every $i \in [m]$, the nodes in $\{C_i^{\models}, C_i^{\not\models}\}$ contribute $2^{n/3} - 1$ to the cut if $\Phi \models C_i$, and $2^{n/3}$ otherwise. Since there are at most $m - p$ clauses that are not satisfied by Φ, summing over all $i \in [m]$ would yield a total of at most $p(2^{n/3} - 1) + (m - p)(2^{n/3}) = 2^{n/3}m - p$ cut edges for vertices with origin in A.

For every $\beta \neq \beta_\Phi$, all nodes in V_2 that are associated with β, v_B, and γ_Φ, are in T and hence will not contribute anything to the cut. However, the node ${\beta_\Phi}'$ is always in S, with v_B its sole target, and hence the edge $({\beta_\Phi}', v_B)$ is in the cut and ${\beta_\Phi}'$ contributes an additional amount of $p - 1$, to a current total of at most $2^{n/3}m - p + (p - 1) = 2^{n/3}m - 1$. In addition, ${\beta_\Phi}'$ is the only target of ${\beta_\Phi}_i^c$, and thus ${\beta_\Phi}_i^c$ will not contribute to the cut.

We will show that the rest of the nodes, i.e., nodes in V_2 that are associated with β_Φ, and the nodes in B, contribute nothing to the cut. For every $i \in [m]$, ${\beta_\Phi}_i^l \in S$ if and only if either $\beta_\Phi \models C_i$ or $\gamma_\Phi \models C_i$. It always happens that ${\beta_\Phi}_i^c \in S$, and ${\beta_\Phi}_i^r \in T$ if and only if

$\gamma_\Phi \not\models C_i$, but in such case it must be that $\beta_\Phi \models C_i$, which implies that the edge $({\beta_\Phi}_i^l, {\beta_\Phi}_i^r)$ is not in the graph, thus the total contribution of ${\beta_\Phi}_i^l$ is zero.

For every $i \in [m]$, it is easy to verify that each of the following four implies the rest. ${\beta_\Phi}_i^r \in S$, $\gamma_\Phi \models C_i$, $C_i \in S$, and the edge (C_i, γ_Φ) is not in the graph. In the case where C_i and ${\beta_\Phi}_i^r$ are in T it is clear that they do not contribute anything, so we will focus on the other case. Since C_i is the only target of ${\beta_\Phi}_i^r$, ${\beta_\Phi}_i^r$ will not increase the cut capacity. In addition, since the edge (C_i, γ_Φ) is not in the graph, C_i does not increase the capacity of the cut either. Altogether we have bounded the total capacity of the cut by $2^{n/3}m - 1$, finishing the proof of Claim 3.2. ◀

Proceeding with the proof of Lemma 3.1, we now focus on the second direction. Assume that every assignment to F satisfies at most $p-1$ clauses. We remind that we need to prove that the maximum flow from every α to every γ is at least $2^{n/3}m$, and to do this we first fix α and γ. By the assumption, for every β there exist $m-(p-1) = m-p+1$ i's, such that α, β, and γ do not satisfy C_i, and we denote a set with this amount of such i's by I_β. Each of these i's induces a blue path $(\alpha \to C_i^{\not\models} \to \beta_i^l \to \beta_i^r \to C_i \to \gamma)$ from α to γ, and so we pass a unit of flow through every one of them according to I_β, and for all β, in what we call the first phase. We note that so far, the flow sums up to $2^{n/3}(m-p+1)$, and so we carry on with shipping the second phase of flow through paths that are not entirely blue.

We claim that for every β, we can pass an additional amount of $m-(m-p+1) = p-1$ units through β', which would add up to a total flow of $2^{n/3}(m-p+1)+2^{n/3}(p-1) = 2^{n/3}m$, concluding the proof. Indeed, for every β, we ship flow in the following way. For every $i \in [m] \setminus I_\beta$, if $\alpha \not\models C_i$ then send a unit through $(\alpha \to C_i^{\not\models} \to \beta_i^l \to \beta_i^c \to \beta' \to v_B \to \gamma)$, and otherwise send a unit through $(\alpha \to C_i^{\models} \to \beta_i^c \to \beta' \to v_B \to \gamma)$.

Since we defined the flow in paths, we only need to show that the capacity constraints are satisfied, starting with edges of color blue. Edges of the forms (β_i^l, β_i^r), (β_i^r, C_i), and (C_i, γ) are only used in the first phase, where the flow in the first two is uniquely determined by β and $i \in I_\beta$, and so at most 1 unit of flow is passed through them, and the flow in the latter kind is determined by $i \in I_\beta$, and the same $i \in I_\beta$ can have at most $|\{\beta_i^r\}_\beta| = 2^{n/3}$ units of flow passing in (C_i, γ), and so the flow in it is also bounded. The flow in edges of the form $(C_i^{\not\models}, \beta_i^l)$ in the first phase is uniquely determined by β and $i \in I_\beta$, and in the second phase uniquely according to β and $i \in [m] \setminus I_\beta$, and so will not be used twice, and the flow in edges of the form $(\alpha, C_i^{\not\models})$ is determined in the first phase by $i \in I_\beta$ and in the second phase by $i \in [m] \setminus I_\beta$, and so will be used at most $\sum_\beta |I_\beta \cap \{i\}| + \sum_\beta |([m] \setminus I_\beta) \cap \{i\}| \le 2^{n/3}$ times.

We now proceed to prove that red edges too do not have more flow than their capacity, and for this we only need to consider the second phase. Edges of the forms $(C_i^{\models}, \beta_i^c)$, (β_i^l, β_i^c), and (β_i^c, β') has flow that is uniquely determined by β and $i \in [m] \setminus I_\beta$ and so are not used more than once, edges of the form (β', v_B) has flow that is determined by β and thus have flow $|\{\beta_i^c\}_{i \in [m] \setminus I_\beta}| = |[m] \setminus I_\beta| = p-1$ and hence are properly bounded, and edges of the form (v_B, γ) have flow of size $(p-1)|\{\beta'\}_\beta| 2^{n/3} = (p-1)2^{n/3}$. Finally, edges of the form $(\alpha, C_i^{\models})$ have flow that is determined by $i \in [m] \setminus I_\beta$ and so are used at most $|\{\beta_i^c\}_\beta| = 2^{n/3}$ times. Altogether, we have bounded the total flow in all the edges that were used in both phases, and so the capacity requirements follow, which completes the proof of the second direction and of Lemma 3.1. ◀

Proof of Theorem 1.6. We use Lemma 3.1 in the following way. Assume that for some $|S'|$ and $|T'|$ there is an algorithm for ST-Max-Flow that runs in time $O((|S'||T'|m)^{1-\varepsilon})$, and consider the version of ST-Max-Flow with S'' and T'' such that $S'' \cap T'' = \emptyset$. Applying such an algorithm repeatedly with S' and T' iterate over respective partitions of S'' and T'' of sizes

$|S''|/|S'|$ and $|T''|/|T'|$, respectively, would solve S'' and T'''s version of ST-Max-Flow and take time $(|S''|/|S'|)(|T''|/|T'|)O(|S'||T'|m)^{1-\varepsilon}) = O((|S''||T''|m)^{1-\varepsilon'})$ for some $\varepsilon' = \varepsilon'(\varepsilon)$, and Theorem 1.6 follows. ◀

4 Global Max-Flow

Proof of Theorem 1.10. Let U and V be an instance of the Orthogonal Vectors problem, where $|U| = |V| = n'$, and all the vectors are from $\{0,1\}^d$ for some $d = \mathsf{polylog}(n')$. We construct a graph $G = (U', V', D)$ as follows. U' contains a node u for every vector $u \in U$, V' contains a node v for every $v \in V$, and D contains three nodes $c_{0,0}$, $c_{0,1}$, and $c_{1,0}$ for every coordinate $c \in [d]$. For every $u \in U'$ and $c \in D$, we add an edge from u to $c_{0,0}$ and $c_{0,1}$ if $u[c] = 0$, and an edge from u to $c_{1,0}$ otherwise. Similarly, for every $v \in U'$ and $c \in D$, we add an edge from v to $c_{0,0}$ and $c_{1,0}$ if $v[c] = 0$, and an edge from v to $c_{0,1}$ otherwise. This graph has $n = n' + n' + 3d = O(n')$ nodes and at most $n' \cdot 2d + n' \cdot 2d = \tilde{O}(n')$ edges. For every $u \in U'$, $v \in V'$, and $c \in [d]$, there is exactly one path (of length 2) from u to v through nodes associated with c if and only if $u[c] \cdot v[c] = 0$, and no paths through them otherwise. Hence, the number of edge disjoint paths from u to v is d if their inner product is 0, and less than d otherwise, and so an algorithm for Maximum Local Edge Connectivity with strictly subquadratic runtime implies an algorithm for the Orthogonal Vectors problem with similar runtime, completing the proof. ◀

Acknowledgements. We thank Rajesh Chitnis and Bundit Laekhanukit for some useful conversations, and for their part in achieving the result on Global Max-Flow.

References

1 Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. SETH-based lower bounds for subset sum and bicriteria path. *CoRR*, 2017. URL: `http://arxiv.org/abs/1704.04546`.

2 Amir Abboud, Virginia Vassilevska-Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC'15, pages 41–50. ACM, 2015. `doi:10.1145/2746539.2746594`.

3 Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows – theory, algorithms and applications.* Prentice Hall, 1993.

4 Srinivasa R. Arikati, Shiva Chaudhuri, and Christos D. Zaroliagis. All-pairs min-cut in sparse networks. *J. Algorithms*, 29(1):82–110, 1998. `doi:10.1006/jagm.1998.0961`.

5 Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012. `doi:10.4086/toc.2012.v008a006`.

6 Mokhtar S. Bazaraa, John J. Jarvis, and Hanif D. Sherali. *Linear Programming and Network Flows.* John Wiley & Sons, Inc., Hoboken, NJ, fourth edition, 2010.

7 Anand Bhalgat, Ramesh Hariharan, Telikepalli Kavitha, and Debmalya Panigrahi. An $\tilde{O}(mn)$ Gomory-Hu tree construction algorithm for unweighted graphs. In *39th Annual ACM Symposium on Theory of Computing*, STOC'07, pages 605–614. ACM, 2007. `doi:10.1145/1250790.1250879`.

8 Glencora Borradaile, David Eppstein, Amir Nayyeri, and Christian Wulff-Nilsen. All-pairs minimum cuts in near-linear time for surface-embedded graphs. In *32nd International Symposium on Computational Geometry (SoCG 2016)*, volume 51 of *Leibniz International*

Proceedings in Informatics (LIPIcs), pages 22:1–22:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.SoCG.2016.22.

9 Glencora Borradaile and Philip Klein. An $\tilde{O}(n \log n)$ algorithm for maximum *st*-flow in a directed planar graph. *J. ACM*, 56(2):9:1–9:30, 2009. doi:10.1145/1502793.1502798.

10 Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, ITCS'16, pages 261–270. ACM, 2016. doi:10.1145/2840728.2840746.

11 Ho Yee Cheung, Lap Chi Lau, and Kai Man Leung. Graph connectivities, network coding, and expander graphs. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, FOCS'11, pages 190–199. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.55.

12 L. R. Ford, Jr. and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. doi:10.4153/CJM-1956-045-5.

13 Greg N. Frederickson. Using cellular graph embeddings in solving all pairs shortest paths problems. *J. Algorithms*, 19(1):45–85, 1995. doi:10.1006/jagm.1995.1027.

14 R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9:551–570, 1961. doi:10.1137/0109047.

15 Dan Gusfield. Very simple methods for all pairs network flow analysis. *SIAM J. Comput.*, 19(1):143–155, 1990. doi:10.1137/0219009.

16 Jianxiu Hao and James B. Orlin. A faster algorithm for finding the minimum cut in a directed graph. *J. Algorithms*, 17(3):424–446, 1994. doi:10.1006/jagm.1994.1043.

17 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

18 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

19 Jakub Lacki, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen. Single source – all sinks max flows in planar digraphs. In *Proceedings of the 53rd IEEE Annual Symposium on Foundations of Computer Science*, FOCS'12, pages 599–608. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.66.

20 Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $\tilde{O}(\sqrt{\text{rank}})$ iterations and faster algorithms for maximum flow. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, FOCS'14, pages 424–433. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.52.

21 Aleksander Mądry. Computing maximum flow with augmenting electrical flows. In *Proceedings of the 57th IEEE Annual Symposium on Foundations of Computer Science*, FOCS'16, pages 593–602. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.70.

22 Alexander Schrijver. On the history of the transportation and maximum flow problems. *Math. Program.*, 91(3):437–445, 2002. doi:10.1007/s101070100259.

23 Virginia Vassilevska-Williams. Hardness of Easy Problems: Basing Hardness on Popular Conjectures such as the Strong Exponential Time Hypothesis (Invited Talk). In *10th International Symposium on Parameterized and Exact Computation (IPEC 2015)*, volume 43 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17–29. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.IPEC.2015.17.

24 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2):357–365, 2005. doi:10.1016/j.tcs.2005.09.023.