On the Hardness of Equal Shortest Path Routing

Frédéric Giroire, Stéphane Pérennes, Issam Tahiri

On the Hardness of Equal Shortest Path Routing

Frédéric Giroire, Stéphane Pérennes, Issam Tahiri [1]

MASCOTTE
INRIA, I3S (CNRS/UNSA)
Sophia Antipolis, France

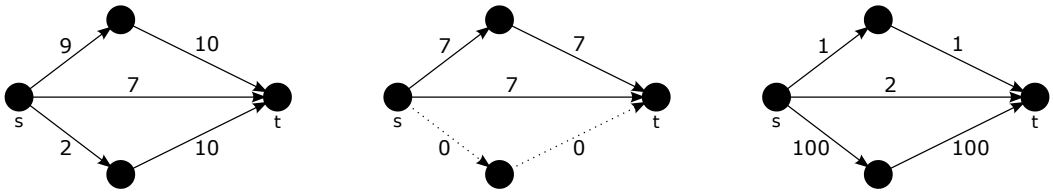
Abstract

In telecommunication networks packets are carried from a source s to a destination t on a path that is determined by the underlying routing protocol. Most routing protocols belong to the class of shortest path routing protocols. In such protocols, the network operator assigns a length to each link. A packet going from s to t follows a shortest path according to these lengths. For better protection and efficiency, one wishes to use multiple (shortest) paths between two nodes. Therefore the routing protocol must determine how the traffic from s to t is distributed among the shortest paths. In the protocol called OSPF-ECMP (for Open Shortest Path First -Equal Cost Multiple Path) the traffic incoming at every node is uniformly balanced on all outgoing links that are on shortest paths. In that context, the operator task is to determine the "best" link lengths, toward a goal such as maximizing the network throughput for given link capacities.

In this work, we show that the problem of maximizing *even a single* commodity flow for the OSPF-ECMP protocol cannot be approximated within any constant factor ratio. Besides this main theorem, we derive some positive results which include polynomial-time approximations and an exponential-time exact algorithm. We also prove that despite their weakness, our approximation and exact algorithms are, in a sense, the best possible.

Keywords: OSPF-ECMP, max flow, **NP**-Hard, approximation.

[1] Corresponding author. Email: issam.tahiri@inria.fr



(a) capacities (b) Max-Ospf-Flow (c) corresponding lengths

Fig. 1. Example of maximum ospf-flow of value 14.

1 Introduction

In this paper we study the complexity of routing according to the *Open Shortest Path First protocol* (OSPF) [1].

The goal of a routing protocol is to make every router capable of deciding, whenever it receives a packet, the next hop router. The decision must be taken locally and quickly, while allowing an efficient usage of the network resources. When OSPF is the routing protocol used in the network, all packets follow shortest paths, according to the link lengths set by the network administrator.

When there are several shortest paths between two nodes u and v, the routing depends on the rule that is used for load balancing the traffic among the shortest paths. There are several rules. One of the most used is ECMP (Equal Cost Multiple Path). According to this rule, a router which has several outgoing links on shortest paths toward a destination v, balances the incoming traffic directed to v evenly among all of them.

In order to understand the algorithmic difficulty of OSPF routing, we look into the most essential problem in which the goal is to maximize the throughput when only *a single pair* of nodes is communicating over the network. We call this problem *Max-Ospf-Flow*:

- INSTANCE: a directed capacitated graph $D = (V, A, c)$ where V is the set of nodes, A is the set of arcs, and c is the capacity function that assigns to each arc (u, v) a capacity $c(u, v)$, and two vertices $s, t \in V$, respectively the source and the sink of the flow.
- QUESTION: find the *maximum flow* going from s to t along with its corresponding arc-length assignment under "Open Shortest Path First" protocol with "Equal Cost Multiple Path" strategy.

We give an example of maximum ospf-flow in Figure 1 and a formal description of the problem in Section 2. Let us point out that there may be a large gap between the Max-Ospf-Flow and the standard maximum flow: Figure 2 shows a flow graph with $n + 2$ vertices in which the value of the Max-Ospf-Flow is 2 while the value of the standard maximum flow is n. To the best of our knowledge, this problem has been only proved to be NP-Hard [11].

In this paper, we show that the Max-Ospf-Flow cannot be approximated within any constant factor ratio. Nevertheless, we derive several positive results. First, the problem can be ρ-approximated. This implies that, when the capacities of all arcs are equal, the problem can even

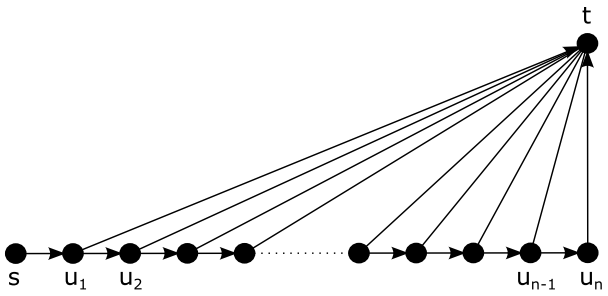


Fig. 2. A flow graph with a big gap between Max-Ospf-Flow and the standard maximum flow. Arcs on the path $s - u_1 - u_2 - \cdots - u_n$ have all capacities equal to n while arcs entering t have all capacities equal to 1.

be solved *exactly* in polynomial time. We then present an exact exponential algorithm of complexity $\tilde{O}(2^{|A|})$ [†] and show that any exact algorithm is likely to cost $2^{\Theta(|A|)}$. Finally, we provide an approximation for networks with small longest distance of factor $\frac{1}{2}\mathcal{H}(\lfloor \frac{c_{\max}}{c_{\min}} \rfloor)^{1-L}$ where L is the length of a longest simple path between the source s and the destination t of the flow.

This study shows that the difficulty of Max-Ospf-Flow highly depends on the range of the capacity function and of the "depth" of the network.

Related Work

Some previous works [8], [10], [4] showed that finding an OSPF routing that optimizes some criterias (load, capacity, ...) of the network performance is a difficult task; and this holds for both unsplittable and splittable (with ECMP rule) routings. When the traffic is unsplittable, the lengths have to be set in a way that for every pair of nodes there is a unique shortest path. In [8], the authors studied the problem of optimizing OSPF-ECMP lengths so as to minimize the total cost of the network when the cost function on arcs is convex and increasing with the congestion. They showed that this problem is **NP**-Hard. In particular, they defined the maximum utilization of the network as $\max_{a \in A} \frac{l(a)}{c(a)}$ where $l(a)$ is the load of arc a, and they proved that minimizing the maximum utilization is **NP**-Hard. They also provided worst-case results about the performance of OSPF-ECMP routing vs. an optimal multi-commodity flow routing in term of congestion. In [4], the authors studied the unsplittable OSPF that minimizes the maximum utilization of the network. They showed that this problem is hard to approximate within a factor of $O(|V|^{1-\varepsilon})$.

As these problems are difficult, the methods to solve them in practice usually follow a two-phase approach based on linear and integer programming: In the first phase a routing that is not necessarily feasible with OSPF is found and in the second phase lengths that achieve this routing through OSPF are computed, when possible. This second phase attempts to solve what is called *the inverse shortest path problem (ISP)*. This problem can be formulated as a linear program and therefore can be solved in polynomial time. Even if the weights are constrained to take non-negative integer values, ISP remains polynomial thanks to a rounding scheme presented in [2]. However finding a solution to ISP minimizing the maximum length over all arcs is **NP**-Hard [3].

[†] $\tilde{O}(g(n)) = O(g(n) \cdot \log^k g(n))$, for $k \in \mathbb{N}$. Logarithmic factors are ignored.

In this paper, we consider the problem of approximating the maximum ospf-flow for a network with a single commodity. This problem has been proved to be **NP**-Hard in [11], using a reduction to set cover. To the best of our knowledge, no approximation with non trivial guaranteed ratio has been suggested before for solving this problem.

To conclude, note that, more generally, the results presented here are valid with minor modifications for other shortest path routing protocols as IS-IS [7] or PNNI [10]. For more information, a survey on the optimization of OSPF routing can be found in [5].

2 Problem Definition

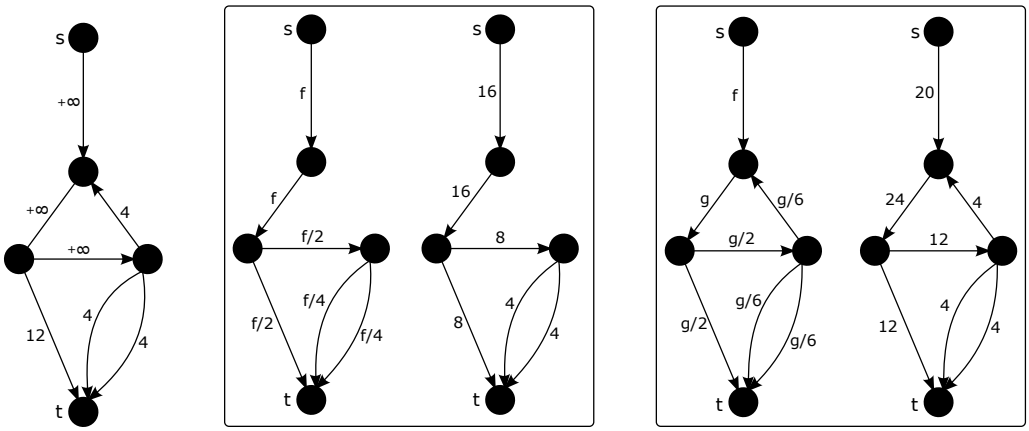
In this section, we show that in the context of our study, which assumes a single commodity, the inverse shortest path problem is guaranteed, under some conditions on the routing, to have a solution and can be easily solved. Therefore, even though Max-Ospf-Flow remains difficult, we can focus on finding an adequate routing function while forgetting about arc-length assignment.

In the case of a single commodity flow, solving the inverse shortest path problem is trivial for any flow f going from s to t. For this we define $A' = \{a \in A \mid f(a) > 0\}$, and set $\forall a \in A', l(a) = 0$ and $\forall a \notin A', l(a) = 1$. Then the shortest paths from the source to the sink are all the paths using only arcs in A'. One may wish for a better length function, such that $\forall a \in A$, $l(a) > 0$, but it exists iff A' is acyclic. First, if A' contains a cycle then $l(a)$ must be null on all the arcs of the cycle. Second, if A' is acyclic, one easily gets a length function since there exists then a potential function $p : V \to \mathbb{R}$ such that $p(u) < p(v)$ if there is a path from u to v (this potential function is given by the dual problem, see [6]). Then, one sets $\forall a = (u, v) \in A, l(u, v) = p(v) - p(u)$. Notice that all paths from s to t have length $p(t)$ - $p(s)$. Combining those two facts, one can hence always define lengths such that $\forall (u, v) \in V^2$, all the paths from u to v in A' have the same length and with $l(a) > 0$ on all arcs a not contained in a cycle. Note that, even if it may seem strange, there exist digraphs for which any solution of Max-Ospf-Flow contains a flow loop (see Figure 3).

Therefore, for any flow function, there always exist lengths for which all the paths used by the flow are shortest paths, and this lengths can be non-negative if the flow is acyclic. Nevertheless, it does not mean that any flow function can be achieved using OSPF-ECMP. Indeed f must also fulfill the "Equally Balanced Condition". This means that we must have: at any node $a \in V$ and $\forall (u, v) \in A$, either $f(u, v) = 0$ or $f(u, v) = out(u)$ where $out(u)$ depends only on u. This allows us to reformulate the Max-Ospf-Flow in a simpler way, without using metric explicitly. But before we need to introduce a few definitions.

Definition 2.1 [flow graph] Given a directed capacitated graph $D = (V, A, c)$ and two particular nodes s and t (to be respectively seen as the source and the sink), their corresponding flow graph is defined as the 5-uplet $\mathcal{D} = (V, A, c, s, t)$.

Definition 2.2 [flow function/value] Given a flow graph $\mathcal{D} = (V, A, c, s, t)$, a function $f : A \to \mathbb{R}^+$ is a flow function if $\forall a \in A, f(a) \leq c(a)$ and $\forall v \in V \backslash \{s, t\}, \sum_{(u,v)\in A} f(u, v) = \sum_{(v,u)\in A} f(v, u)$.



(a) capacities (b) loop-free flow (c) flow with loop

Fig. 3. Example of an instance where the optimal flow can only be achieved using a loop: when avoiding the loop, one can easily check that the limitation comes from the constraint $f/4 \leq 4$ which means that the flow sent from s to t is at most 16; while using a loop the same constraint allows g to be equal to 24 which leads to a flow going from s toward t that is equal to $f = g - g/6 = 20$. Other configurations with less arcs are not depicted but lead to even lower ospf-flow value.

The *flow value* is $\text{val}(f) = \sum_{(s,v)\in A} f(s,v) - \sum_{(v,s)\in A} f(v,s)$.

Definition 2.3 [regular function] A function over a set S is said to be regular if it takes only two values, 0 and another possible one.

Definition 2.4 [balanced function] Given graph $G = (V, A)$ and $V' \subset V$, a function f is said to be balanced on V' iff $\forall u \in V'$ the function f_u, defined by $\forall (u, v) \in A$, $f_u(v) = f(u, v)$, is regular.

Definition 2.5 [ospf-flow function] For a flow graph $\mathcal{D} = (V, A, c, s, t)$, a function $f : A \to \mathbb{R}^+$ is an ospf-flow if it is a flow function and if it is balanced on V.

Now, we give an equivalent formulation of the problem **Max-Ospf-Flow**:

- **Instance : a flow graph $\mathcal{D} = (V, A, c, s, t)$.**
- **Question : find the maximum flow value of an ospf-flow function in A.**

The metric problem has only apparently vanished, since choosing on which arc the flow is null and on which arcs it is strictly positive actually determines the underlying hidden length function.

Some notations: For f a function from a set S to $\mathbb{R}$, we denote $f(S) = \sum_{s\in S} f(s)$. Given a digraph with vertex set V and arc set A we also abusively denote (U, V) the set of couples $\{(u, v) \in A \mid u \in U, v \in V\}$. We also note the set of integers $\{1, \ldots, N\}$ by $[N]$.

3 Inapproximability results

We use a reduction to the MAX-3-SAT problem (which cannot be approximated within a factor $\frac{7}{8}+\varepsilon$ for any constant $\varepsilon > 0$) in order to prove first that Max-Ospf-Flow is hard to approximate, in a polynomial time, within an approximation ratio of $\frac{31}{32}$. Then we use a self-amplifying method to prove that this problem is not in APX. The proofs of following results can be found in Appendix A.1 or in the research report [].

Proposition 3.1 *It is NP-hard to approximate Max-Ospf-Flow within a factor* $1-\frac{l-3}{8l}+\varepsilon, \forall \varepsilon > 0$, *even if* $\forall a \in A, c(a) \in \{1, l\}$. *As example, it is NP-hard to approximate Max-Ospf-Flow within a factor* $\frac{31}{32}+\varepsilon, \forall \varepsilon > 0$, *even if* $\forall a \in A, c(a) \in \{1, 4\}$,

Theorem 3.2 *It is* NP*-Hard to approximate Max-Ospf-Flow within any constant factor.*

Sketch of the Proof. To prove the result, we use a self-amplifying error technic. Indeed, the flow graph $\mathcal{D}_I$ associated to an m-clause 3-SAT instance I (in the reduction of Proposition 3.1) connects s to t is like a virtual arc (s,t) with capacity m, but for which it is NP-Hard to use a capacity larger than $\frac{31m}{32}$. Considering a flow graph $\mathcal{D}$, we will replace each of its edges by those virtual arcs. Two factors of error will then get multiplied: one because no polynomial algorithm can use the full capacity of the virtual arcs, the other because of the error on the original network.□

4 Positive results

Even though it is hard to find the optimal solution to Max-Ospf-Flow, we can still get some approximation algorithms.

Theorem 4.1 *Given a flow graph* $\mathcal{D} = (V, A, c, s, t)$. *Let* $c_{\min}$ *and* $c_{\max}$ *be respectively the smallest and the largest value of the capacity function* c. *Then Max-Ospf-Flow can be approximated on* $\mathcal{D}$ *in polynomial time within a factor* $c_{\min}/c_{\max}$.

Proof. Let OPT be the maximum value of an ospf-flow on $\mathcal{D}$, and let $\mathcal{D}_i$ to be the directed graph that have the same elements as $\mathcal{D}$ but where the capacity function is the constant function that assigns i to every arc. We compute a maximum integral flow function F_1 on $\mathcal{D}_1$. On the one hand, the value of an ospf-flow on $\mathcal{D}$ cannot exceed the maximum value of a flow on $D_{c_{\max}}$, which is $c_{\max}v(F_1)$, so $\mathrm{OPT} \leq c_{\max}v(F_1)$. On the other hand, $c_{\min}F_1$ is a feasible ospf-flow for $\mathcal{D}$ with value $c_{\min}v(F_1) \geq \frac{c_{\min}}{c_{\max}}\mathrm{OPT}$. □

When $c_{\min} = c_{\max}$ we get the next corollary :

Corollary 4.2 *Any instance of Max-Ospf-Flow where all capacities of the flow graph are equal can be solved in polynomial-time.*

We now give an exact algorithm with exponentional solving time.

Theorem 4.3 *Given a flow graph $\mathcal{D} = (V, A, c, s, t)$, we have an exact algorithm for solving Max-Ospf-Flow on $\mathcal{D}$ which runs in $\tilde{O}(2^{|A|})$.*

Proof. Consider an ospf-flow function f_x with a flow value x and remark that the knowledge of the *configuration* $A' = \{a \in A \mid f_x(a) > 0\}$ entirely determines f_x in the following sense: For a given configuration, there exists at most *one* ospf-flow of value x. Indeed if $d(v)$ denotes the out-degree in A' of a vertex v the following equations are satisfied:

$$\forall v \notin \{s,t\}, \forall (v,w) \in A' \; f_x(v,w) = \tfrac{1}{d(v)} \textstyle\sum_{(u,v)\in A'} f_x(u,v)$$
$$\forall (s,w) \in A' \qquad f_x(s,w) = \tfrac{x}{d(s)}$$

Note that the above system is well-defined iff all nodes adjacent to some edge in A' belong to a path in A' from s to t. Since the configuration associated to the Max-Ospf-Flow satisfies this condition, we can discard any degenerated set A' violating the above condition.

If f_1 is the function that solves the above system for $x = 1$, then for $x \in \mathbb{R}$ the solution is given by the function $f_x = x \cdot f_1$. Therefore, the maximum flow value of an ospf-flow using the configuration A', noted $\text{val}(A')$, expresses as the maximum among all $x \in \mathbb{R}$ that respects the following capacity constraints:

$$\forall (v,w) \in A', x f_1(v,w) \leq c(v,w)$$

Hence, we have $\text{val}(A') = \min_{(v,w)\in A'} \frac{c(v,w)}{f_1(v,w)}$.

Finally, to compute the max-ospf-flow, we can simply compute $\text{val}(A')$, in polynomial time, for each non degenerated configuration and take the maximum. □

We now give a more efficient approximation algorithm when $\frac{c_{\max}}{c_{\min}}$ is large and when the longest path from s to t is short. We call the length of the latter $L_{\mathcal{D}}(s,t)$. The proof is in Appendix A.2 and in the research report []. The algorithm is based on a sampling argument and on the monotonicity of a (multi-source) ospf-flow, meaning that if a ospf-flow exists, ospf-flows of smaller values always exist.

Theorem 4.4 *Let $\mathcal{D} = (V, A, c, s, t)$ be a flow graph, and assume that $c : A \to [c_{\min}, c_{\max}]$, then Max-Ospf-Flow can be approximated in polynomial time within a factor $\frac{1}{2}\mathcal{H}(\lfloor \frac{c_{\max}}{c_{\min}} \rfloor)^{1-L_{\mathcal{D}}(s,t)}$*

5 Conclusion & Open problems

In this study, our main incentive was to investigate the difficulty of routing the data "optimally" in networks that have opted for OSPF with ECMP strategy, as a dynamic routing protocol. To do so, we concentrated on the fondamental problem of maximizing the flow from a single source to a single destination on a network where this protocol is assumed to be running.

The main result shows that there exists no constant factor approximation if the capacities are not bounded. When all capacities are integers in $[c]$, There is a c-factor approximation, however, we have been unable to find an algorithm with ratio better than c So we conjecture that for any μ Max-Ospf-Flow cannot be approximated within a factor better than $c^{1-\mu}$. Nevertheless, if the length of the longest path from s to t is bounded by L, then we derive an approximation algorithm with the ratio $\frac{1}{2}\mathcal{H}(\lfloor\frac{c_{\max}}{c_{\min}}\rfloor)^{1-L}$.

Additionally, we suggest an exact algorithm that has a complexity of $\tilde{O}(2^{|A|})$. It is an open problem to find an algorithm with a better exponential base. However, there is a limitation on the efficiency of such algorithm, since it would be able to solve MAX-3-SAT (see the reduction of Theorem 3.1) and the best known algorithm solving an m-clause instance has complexity $\tilde{O}(1.3^m)$.

References

[1] *Ospf version 2, rfc2328*, http://www.ietf.org/rfc/rfc2328.txt (1998).

[2] Ben-Ameur, W. and E. Gourdin, *Internet routing and related topology issues*, SIAM J. Discret. Math. **17** (2004), pp. 18–49.

[3] Bley, A., *Inapproximability results for the inverse shortest paths problem with integer lengths and unique shortest paths*, in: *Proceedings of the Second International Network Optimization Conference*, 2005.

[4] Bley, A., *On the approximability of the minimum congestion unsplittable shortest path routing problem*, in: *Proceedings of the 11th international conference on Integer Programming and Combinatorial Optimization* (2005), pp. 97–110.

[5] Bley, A., B. Fortz, E. Gourdin, K. Holmberg, O. Klopfenstein, M. Pioro, A. Tomaszewski and H. Umit, "Optimization of OSPF routing in IP networks," Springer, 2009 pp. 199–240, in Graphs and Algorithms in Communication Networks: Studies in Broadband, Optical, Wireless and Ad Hoc Networks.

[6] Cook, W. J., W. H. Cunningham, W. R. Pulleyblank and A. Schrijver, "Combinatorial optimization," Wiley, 1998.

[7] Fortz, B. and M. Thorup, *Optimizing ospf/is-is weights in a changing world*, IEEE Journal on Selected Areas in Communications **20** (2002), pp. 756–767.

[8] Fortz, B. and M. Thorup, *Increasing internet capacity using local search*, Computational Optimization and Applications **29** (2004), pp. 13–48.

[9] Håstad, J., *Some optimal inapproximability results*, Journal of the ACM **48** (2001), pp. 798–859.

[10] Izmailov, R., B. Sengupta and A. Iwata, *Administrative weight allocation for pnni routing algorithms*, in: *IEEE Workshop on High Performance Switching and Routing*, 2001, pp. 347–352.

[11] Pióro, M., Á. Szentesi, J. Harmatos, A. Jüttner, P. Gajowniczek and S. Kozdorowski, *On open shortest path first related network optimisation problems*, Journal of Performance Evaluation **48** (2002), pp. 201–223.

A Appendix

A.1 Inapproximability Results

We give here the proofs of Proposition 3.1 and Theorem 4.4.

We use a reduction to the MAX-3-SAT problem in order to prove first that Max-Ospf-Flow is hard to approximate, in a polynomial time, within an approximation ratio of $\frac{31}{32}$. Then we use a self-amplifying method to prove that this problem is not in APX. We recall that MAX-3-SAT is the problem defined by: a set $F = \{B_j\}_{j\in[m]}$ of clauses, built on a finite set $\{x_i\}_{i\in[n]}$ of binary variables, such that each clause contains exactly three literals; and the question is to find a truth assignment of the variables $\{x_i\}_{i\in[n]}$ that maximizes the number of satisfied clauses in F.

Unless $\mathbf{P} = \mathbf{NP}$, Problem MAX-3-SAT cannot be approximated within a factor $\frac{7}{8} + \varepsilon$ for any constant $\varepsilon > 0$, see [9]. In particular, even when the optimum value is m, it is still $\mathbf{NP}$-Hard to find an assignment that satisfies more than $\frac{7}{8} + \varepsilon$ clauses of F, $\forall\varepsilon > 0$.

Proposition 3.1: *It is NP-hard to approximate Max-Ospf-Flow within a factor* $1 - \frac{l-3}{8l} + \varepsilon, \forall\varepsilon > 0$*, even if* $\forall a \in A, c(a) \in \{1, l\}$*. As example, it is NP-hard to approximate Max-Ospf-Flow within a factor* $\frac{31}{32} + \varepsilon, \forall\varepsilon > 0$*, even if* $\forall a \in A, c(a) \in \{1, 4\}$.

Proof. For seek of simplicity, our reduction uses a multigraph. To get a graph from it one can simply insert a node on every arc. This will make the graph simple and will only double its initial size. Given an instance I of MAX-3-SAT, defined by the set of binary variables $\{x_i\}_{i\in[n]}$ and the set of clauses $\{B_j\}_{j\in[m]}$, we build the following flow graph $D_I = (V, A, c, s, t)$:

- For each $i \in [n]$, we create a vertex a_i representing variable x_i.
- For each $j \in [m]$, we create a vertex b_j representing clause B_j.
- For each $i \in [n]$, we add k_i parallel arcs with capacity 1 going from s to a_i, where k_i is the number of times the variable x_i appears in the clauses.
- We add an arc with capacity 1 from every b_j to t, with $j \in [m]$.

In addition to those arcs, we add arcs that depend on the structure of the clauses :

- $\forall(i, j) \in [n] \times [m]$, if B_j contains the positive literal x_i we add an arc with capacity 1 from a_i to b_j ;
- $\forall(i, j) \in [n] \times [m]$, if B_j contains a negative literal $\bar{x}_i$ we add l parallel arcs with capacity $\frac{1}{l}$ from a_i to b_j

An example of the construction is displayed in Figure A.1. Note that the size of $\mathcal{D}_I$ is linear in the number of clauses.

The idea of the reduction is to emulate binary variables as follows: For $(i, j) \in [n] \times [m]$ we define the virtual link $vl(a_i, b_j)$ as the set of arcs connecting a_i to b_j, we say that the virtual link

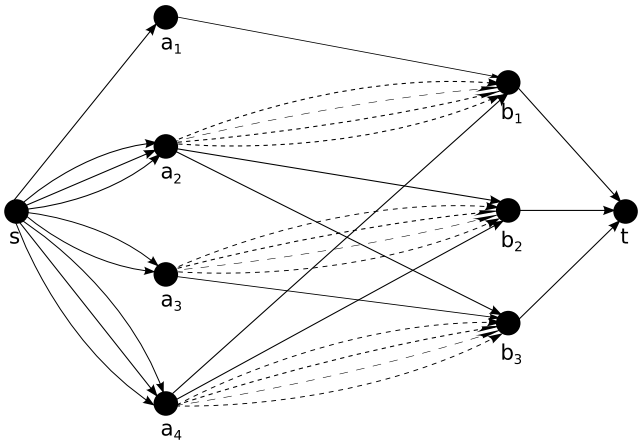


Fig. A.1. Flow graph $\mathcal{D}_I$ constructed from a set F containing the clauses $B_1 = (x_1 \vee \bar{x}_2 \vee x_4)$, $B_2 = (x_2 \vee \bar{x}_3 \vee x_4)$ and $B_3 = (x_2 \vee x_3 \vee \bar{x}_4)$. [The example assumes $l = 4$. Dashed arcs have capacity $\frac{1}{4}$. The others have capacity 1].

$vl(a_i, b_j)$ is *positive* (resp. *negative*) *virtual link* if x_i (resp. $\overline{x_i}$) is a literal of B_j. Then, one must choose at a_i either to send significative flow on the positive virtual links or the negative ones, and this is exclusive. This formalizes with the following property:

Property 1 *Any ospf-flow function must satisfy*

a) If for some $(i, j) \in [n] \times [m]$ the flow on a positive virtual link $vl(a_i, b_j)$ is strictly greater than $\frac{1}{l}$ then any flow on a negative virtual link $vl(a_i, b_k)$ $\forall b_k, k \in [m]$ is at most 0.

b) If for some $(i, j) \in [n] \times [m]$ the flow on a negative virtual link $vl(a_i, b_j)$ is strictly positive then any flow on a positive virtual link $vl(a_i, b_k)$ $\forall b_k, k \in [m]$ is at most $\frac{1}{l}$.

To prove (a) remark that positive virtual links are made of a single arc with capacity 1, so if (a) premise happen we must have $f(a_i, b_j) > \frac{1}{l}$ and since f_{a_i} is regular no flow can leave a_i on the arcs with capacity $\frac{1}{l}$. Case (b) is similar, since (b) premise means that the arcs with capacity $\frac{1}{l}$ are used, and since f_{a_i} is regular, so $f(a_i, b_j) \leq \frac{1}{l}$ (since there is a unique arc for the positive virtual links).

We first prove that the Max-Ospf-Flow on $\mathcal{D}_I$ is m if I has a truth assignment that satisfies all clauses. To do so, we associate to this truth assignment a flow function f on $\mathcal{D}_I$ as follows: for each satisfied clause B_j, we choose one of its satisfied literals, and if it is associated to the variable x_i we send one unit of flow on the path $s - a_i - b_j - t$ using the virtual virtual link corresponding to the literal. The flow f has then a value m and is balanced since:

- $\forall i \in [n]$ with x_i true all arcs leaving a_i have flow 0 or 1.
- $\forall i \in [n]$ with x_i false all arcs leaving a_i have flow 0 or $\frac{1}{l}$.

Therefore f is an ospf-flow. Moreover, this flow is maximum since the cut $(\{t\}, V \setminus \{t\})$ has a

capacity m. Consequently, the value of the Max-Ospf-Flow on $\mathcal{D}_I$ is m.

We now assume that we are able to find (using a polynomial algorithm) an ospf-flow function f. To that ospf-flow flow we associate a binary assignment for each $i \in [n]$ as follows:

a) If a vertex a_i sends strictly more than $\frac{1}{l}$ units of flow along a positive virtual link we set x_i to true.

b) If a vertex a_i sends strictly more than $\frac{1}{l}$ units of flow along negative virtual link we set x_i to false.

According to Property 1 this assignment is consistent. Note that the above rules may let some variables unassigned, in which case we simply discard them (or let them unassigned).

Consider now one of the nodes $b_j, j \in [m]$ for which the flow $f(b_i, t)$ is greater than $\frac{3}{l}$. Since there are at most three virtual links entering the node b_j one virtual link $(a_i, b_j), i \in [n]$ carries a flow strictly greater than $\frac{1}{l}$. Then, either $vl(a_i, b_j)$ is a positive virtual link and x_i was set to true (by rule (a)) and the clause B_j is satisfied. Or $vl(a_i, b_j)$ is a negative virtual link and x_i is set to false (by rule (b)) and the clause B_j is satisfied. So such a node b_j corresponds a clause B_j satisfied by our assignment. So (unless $\mathbf{P} = \mathbf{NP}$) we have at most $(\frac{7}{8} + \varepsilon)m$ such nodes. But then, the flow is at most $\left(\frac{7}{8} + \varepsilon\right) m \cdot 1 + \left(1 - \left(\frac{7}{8} + \varepsilon\right)\right) m \cdot \frac{3}{l} = m - \frac{l-3}{l}\left(\frac{1}{8} - \varepsilon\right) m = \left(1 - \frac{l-3}{8l}\right) m + \varepsilon \left(\frac{l-3}{l}\right) m$.

So we conclude that $\forall l \geq 3$ and $\forall \varepsilon'$, (unless $\mathbf{P} = \mathbf{NP}$) no polynomial algorithm computes a flow with value greater than $(1 - \frac{l-3}{8l} + \varepsilon')m$. Note that the constructed graph uses capacities $\{\frac{1}{l}, 1\}$ so we multiply all the capacities by l to get an equivalent graph with capacities in $\{1, l\}$. □

Theorem 3.2: *It is* $\mathbf{NP}$*-Hard to approximate Max-Ospf-Flow within any constant factor.*

Proof. *To prove the result, we use a self-amplifying error technic. Indeed, the flow graph $\mathcal{D}_I$ associated to a 3-SAT instance I connects s to t is like a virtual arc (s, t) with capacity m, but for which it is $\mathbf{NP}$-Hard to use a capacity larger than $\frac{31m}{32}$. Considering a flow graph $\mathcal{D}$, we will replace each of its edges by those virtual arcs. Two factors of error will then get multiplied: one because no polynomial algorithm can use the full capacity of the virtual arcs, the other because there is another error on the original network itself. We note $\mathrm{OPT}(\mathcal{D})$ the value of Max-Ospf-Flow on $\mathcal{D}$. Let us introduce two definitions:*

- *For a flow graph $\mathcal{D} = (V, A, c, s, t)$, we define $x \otimes \mathcal{D} = (V, A, c_x, s, t)$ where c_x is defined by $\forall a \in A, c_x(a) = x \cdot c(a)$. For any positive number x, there is a trivial bijection which relates every flow function f for $\mathcal{D}$ to a flow function g for $x \otimes \mathcal{D}$, defined by $\forall e \in A, g(a) = x \cdot f(a)$.*
- *Given two flow graphs $\mathcal{D}_1$ and $\mathcal{D}_2$, we define $\mathcal{D}_1 \circ \mathcal{D}_2$ as the graph in which every arc $a = (u, v) \in A$ (A is in $\mathcal{D}_1$) with capacity $c(a)$ is substituted with the following sequence (see Figure A.2): a copy of $c(a) \otimes \mathcal{D}_2$ with source $s(u, v)$ and sink $t(u, v)$, two arcs $(u, s(u, v))$ and $(t(u, v), v)$ with capacities $c(a)M$, where M is an upper bound of $\mathrm{OPT}(\mathcal{D}_2)$.*

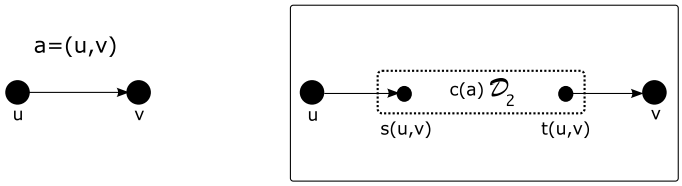


Fig. A.2. A virtual arc of $\mathcal{D}_1 \circ \mathcal{D}_2$ (Right) substituting the single arc (u, v) of $\mathcal{D}_1$ (Left).

Suppose that one had proven that the Max-Ospf-Flow is hard to approximate within ρ_1 and ρ_2; namely we can associate to any instance I of an $\mathbf{NP}$*-Complete problem a flow digraph $\mathcal{D}_1$ (resp. $\mathcal{D}_2$) such that a ρ_1(resp. ρ_2) approximation of the Max-Ospf-Flow allows to solve I. We build the flow graph $\mathcal{D}_1 \circ \mathcal{D}_2$. Note first that the size of $\mathcal{D}_1 \circ \mathcal{D}_2$ is the product of $\mathcal{D}_1$ and $\mathcal{D}_2$ sizes (hence if $\mathcal{D}_1$ and $\mathcal{D}_2$ have polynomial size then so does $\mathcal{D}_1 \circ \mathcal{D}_2$). The maximum ospf-flow of $\mathcal{D}_1 \circ \mathcal{D}_2$ has value $\mathrm{OPT}(\mathcal{D}_1)\mathrm{OPT}(\mathcal{D}_2)$. Suppose now that we can find in polynomial time an ospf-flow of value strictly greater than $\rho_1\rho_2\mathrm{OPT}(\mathcal{D}_1)\mathrm{OPT}(\mathcal{D}_2)$ for every instance I. For any instance I, two different cases can happen:*

- *Either, there exists at least one virtual arc with a flow value strictly larger than $\rho_2\mathrm{OPT}(\mathcal{D}_2)$. In this case, we use reduction 2 and solve the NP-problem for instance I.*
- *Or, for all virtual arcs, the flow is less than $\rho_2\mathrm{OPT}(\mathcal{D}_2)$. In this case, we can define a valid ospf-flow for the graph $\mathrm{OPT}(\mathcal{D}_2) \otimes \mathcal{D}_1$ by simply taking as flow value of arc (u, v) the flow value of arc $(u, s(u, v))$ in $\mathcal{D}_1 \circ \mathcal{D}_2$. We can build a valid ospf-flow for $\mathcal{D}_1$ by dividing the value of the flow on each arc by $\mathrm{OPT}(\mathcal{D}_2)$. This ospf-flow is of value greater than $\rho_1\mathrm{OPT}(\mathcal{D}_1)$. We then use reduction 1 and we solve the NP-problem for instance I.*

Hence, by contradiction, it is not possible to find in polynomial time an ospf-flow of value greater than $\rho_1\rho_2\mathrm{OPT}(\mathcal{D}_1)\mathrm{OPT}(\mathcal{D}_2)$ for every instance I. To complete the proof we start with $\rho_1 = \rho_2 = \rho = \frac{31}{32}$. And starting from hardness to approximate within ρ we prove hardness to approximate within ρ^2 it follows that Max-Ospf-Flow cannot be approximated within any constant unless $\mathbf{P} = \mathbf{NP}$. □

A.2 Positive Results

We give here the proof of Theorem 4.4. This theorem gives a more efficient approximation algorithm when c is large. The algorithm is based on a sampling argument given in Lemma A.2 and on the monotonicity of a (multi-source) ospf-flow, meaning that if a ospf-flow exists, ospf-flows of smaller values always exist in the sense of Lemma A.4.

Definition A.1 Given two functions f, g from $S \to \mathbb{R}$, we say that g is f-dominated if $\forall x \in S, f(x) \leq g(x)$.

Lemma A.2 *Let S be a finite set, and a function $p : S \to [MAX]$, then there exists an integral regular p-dominated function q with $q(S) \geq \lceil \frac{p(S)}{\mathcal{H}(MAX)} \rceil$.*

Proof. *Assume that the unique non zero value that we pick for q is i. Then, since $q \leq p$, the best choice for q (in order to maximize $q(S)$) is to simply set $q(x) = 0$ iff $p(x) < i$ and $q(x) = i$ iff $p(x) \geq i$. So, let us define the two sets: $\Delta(i) = p^{-1}(i)$ and $G(i) = \cup_{j \in [MAX], j \geq i} \Delta(j)$. Then, from definition, for any $i \in [MAX]$, we can build a regular function q, p-dominated, such that $q(S) = i \cdot |G(i)|$. It follows that the best value that we can get for $q(S)$ is:*

$$\mathrm{OPT} = \max_{i \in [MAX]} i \cdot |G(i)|.$$

Note also that, with those notations (using a standard double couting argument), we have:

$$p(S) = \sum_{i \in [MAX]} i|\Delta(i)| = \sum_{i \in [MAX]} |G(i)|.$$

We have $\forall i \in [MAX], |G(i)| \leq \frac{\mathrm{OPT}}{i}$. So $p(S) = \sum_{i \in [MAX]} |G(i)| \leq \mathrm{OPT} \sum_{i \in [MAX]} \frac{1}{i} = \mathcal{H}(MAX)\mathrm{OPT}$. It follows that $\mathrm{OPT} \geq \frac{p(S)}{\mathcal{H}(MAX)}$. Note that by construction the optimal function q is integral and OPT is integral, so we indeed have $\mathrm{OPT} \geq \lceil \frac{p(S)}{\mathcal{H}(MAX)} \rceil$. □

Definition A.3 [Multisource ospf-flow] Consider a digraph $G = (V, A)$ and a capacity function on the arcs c, a multisource ospf-flow with n sources $\{s_i\}_{i \in [n]}$ and a sink t is a function $f : A \to \mathbb{R}^+$ such that

- $\forall a \in A, f(a) \leq c(a)$ and
- $\forall v \in V \setminus \{s_i\}_{i \in [n]} \cup \{t\}, \sum_{uv \in A} f(u, v) = \sum_{vu \in A} f(v, u)$.
- balanced on V.

For $i \in [n]$, $\mathrm{val}_i = \sum_{(s_i, v) \in A} f(s_i, v) - \sum_{(v, s_i) \in A} f(v, s_i)$ is called the *ouput value* of source i The *flow value* of f is defined as $\mathrm{val}(f) = \sum_{i \in [n]} \mathrm{val}_i$.

Lemma A.4 (monotonicity) *Consider a flow graph with k sources $\{s_i\}_{i \in [k]}$ and a sink t. If it exists a multisource ospf-flow in which the source s_i has an output value val_i, then, for any $\{val'_i\}_{i \in [k]}$ such that $\forall i \in [k], val'_i \leq val_i$, there exists a multisource ospf-flow in which each source has an output value val'_i.*

Proof. *We take the configuration A' associated to the initial ospf-flow, i.e. $A' = \{a \in A; f(a) > 0\}$. We then want to determine if an ospf-flow with output values $val'_i, i \in [k]$ on the sources is feasible with configuration A'. Such a flow is entirely determined by the values $val'_i, i \in [k]$ (see the proof of Theorem 4.3), and the flow on an arc a is of the form $\sum_{i \in [k]} \lambda_{a,i} val'_i$ where $\lambda_{a,i}$ are all positive or null. So the load of an arc is increasing with the source outputs and the result follows (the set of feasible ospf-flows, with configuration A', is defined by $\forall a \in A, \sum \lambda_{a,i} val'_i \leq c(a)$).* □

Definition A.5 For $l \in \mathbb{N}$, we say that a flow graph $\mathcal{D} = (V, A, c, s, t)$ is l-layered if its vertex set can be partitionned into $V = \cup_{i\in[l]}\text{val}_i$ such that $V_1 = \{s\}$, $V_l = \{t\}$ and the arc set A satisfies $A = \cup_{i\in[l-1]}A_i$ where $\forall i \in [l-1], A_i \subseteq (\text{val}_i, V_{i+1})$.

Proposition A.6 *Given an integer $l \geq 3$, let $\mathcal{D} = (V, A, c, s, t)$ be an l-layered flow graph. Given an integral flow function F that has values in $[c]$, we can compute in polynomial-time an ospf-flow of value $val(f)$ greater than $\mathcal{H}(c)^{-l}val(F)$.*

Proof. *The idea is to sample the original flow function F level per level. Let $\rho = \mathcal{H}(c)^{-1}$ be our sampling factor. We build by induction on t a flow function g_t that has the following properties:*

- *g_t is balanced on the t last levels (i.e. on the vertex set $B_t = \cup_{i\in[l-t,l]}V_i$).*
- *g_t is F-dominated, that is $\forall a \in A, g_t(a) \leq F(a)$.*
- *$v(g_t) \geq \rho^t \cdot val(F)$.*
- *g_t is integral till level t (i.e $\forall a \in \cup_{i\in[t]}A_i$, $g_t(a)$ is an integer).*

For $t = 0$, we take $g_0 = F$. Since B_0 is reduced to the sink, the inductive hypothesis holds.

We now build g_t from g_{t-1} for $t \geq 1$. We start setting $g_t = g_{t-1}$. By induction hypothesis, g_t is balanced on all the levels in $[l-t+1, l]$ (i.e on B_{t-1}). We want to make g_t balanced on level V_{l-t}. So, we look at the situation of the outgoing flow from level V_{l-t} to level V_{l-t+1}. We consider the flow function g_{t-1} on the set A_{l-t}. By induction hypothesis, g_{t-1} is integral with values in $[c]$. So we can sample g_{t-1} on the set A_{l-t} using Lemma A.2 and get an integral, regular, g_{t-1} dominated function q_t such that $q(A_{l-t}) \geq \rho g_{t-1}(A_{l-t})$. We then set $\forall a \in A_{l-t}, g_t(a) = q_t(a)$. Hence, the flow from level $l-t$ to level $l-t+1$ is at least $\rho v(g_{t-1}) = \rho^t val(F)$.

The function g_t is not anymore a flow since it does not respect the conservation anymore on levels V_{l-t} and V_{l-t+1}. Nodes in V_{l-t} receive too much flow, and nodes in V_{l-t+1} send too much flow.

- *For nodes in V_{l-t+1}, we use monotony (Lemma A.4) which allows us to decrease the flow sent toward the sink while keeping the ospf constraints satisfied.*
- *For nodes in V_{l-t} we can easily decrease the flow since the flow function is not required to fulfill the regularity condition. Nevertheless, we have to keep the flow integral. But since q_t is an integral function, the excess flow received at any vertex $v \in V_{t-l}$ is an integer. So we can decrease the flow unit per unit and keep g_t integral on all levels till $t-l$.*

So all the inductive conditions are fulfilled for t, the result follows by induction.

□

We define $L_{\mathcal{D}}(u, v)$ as the length of the longest simple path between u and v in a flow graph $\mathcal{D}$.

Theorem A.7 *Let $\mathcal{D} = (V, A, c, s, t)$ be a flow graph, and assume that $c : A \to [c]$, then Max-Ospf-Flow can be approximated in polynomial time within a factor $\mathcal{H}(c)^{1-L_{\mathcal{D}}(s,t)}$*

Proof. *We first compute a maximum integral flow F without flow loops (such always exists).*

The arcs used by the flow then form an acyclic digraph, and we only consider this digraph. We now assign to each vertex a level as follows: at step $i \geq 0$, we mark any vertex for which all the in-neighbors were marked at previous steps and assign to such a vertex the Level i. The procedure ensures that the arcs all go from a level to a greater level. Note that the source is the only vertex at Level 0 and that the sink is the only vertex at Level $L_{\mathcal{D}}(s,t)$, since there is a path going from the sink to all vertices and that all paths go to the sink. So we have almost layered our digraph, to complete the process we add $k-1$ intermediary nodes on any arc going from level i to level $i+k$ in order to get only arcs between consecutive levels. See Figure A.3 for an example.

Then we apply Proposition A.6 to the layered digraph with $l = L_{\mathcal{D}}(s,t)+1$. We obtain in polynomial time an ospf-flow f on the layered digraph of value greater than $\mathcal{H}(c)^{-l}val(F)$. We can easily build an ospf-flow of same value on $\mathcal{D}$ (the flow on all arcs of a path with intermediary nodes is equal, we assigne this flow value to the corresponding arc of the original digraph). This ends the proof since the value of a maximum flow F is an upper bound on the value of a maximum ospf-flow. □

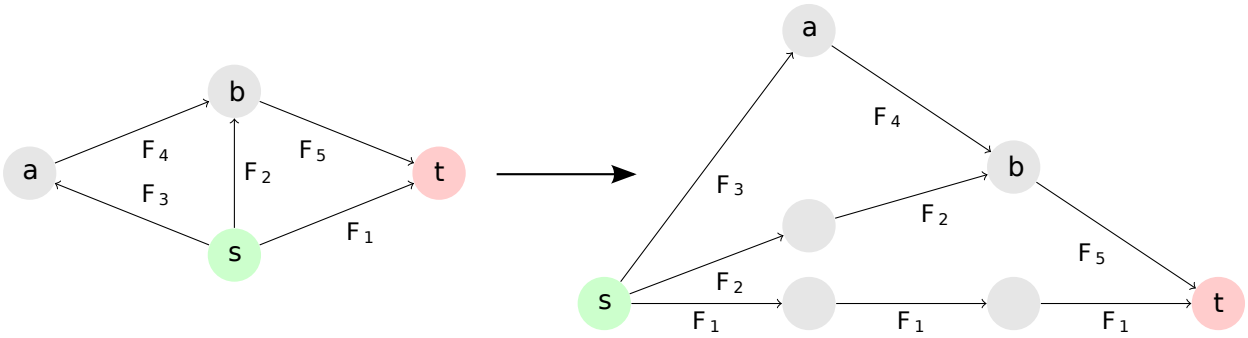


Fig. A.3. Example of the construction of Theorem 4.4 Left: $\mathcal{D}'$. Right: $\mathcal{D}$. A vertex u is placed at the level $L_{\mathcal{D}}(s,v)$, with $L_{\mathcal{D}}(s,v)$ the length of a longest path between s and v. Each arc $(u,v) \in A$ is then subdivided into a path of length $L_{\mathcal{D}}(u,v)$.

Theorem 4.4: Let $\mathcal{D} = (V, A, c, s, t)$ be a flow graph, and assume that $c : A \to [c_{\min}, c_{\max}]$, then the maximum value of an ospf-flow be approximated in polynomial time within a factor $\frac{1}{2}\mathcal{H}(\lfloor \frac{c_{\max}}{c_{\min}} \rfloor)^{1-L_{\mathcal{D}}(s,t)}$

Proof. Let f_0 be the maximum value of a (standard) flow on $\mathcal{D}$. We consider the flow graph $\mathcal{D}' = (V, A, c', s, t)$ with capacities $\forall a \in A, c'(a) = \lfloor \frac{1}{c_{\min}} c(a) \rfloor$. We have $\forall a \in A, c'(a) \in [\lfloor \frac{c_{\max}}{c_{\min}} \rfloor]$. Since $\forall a \in A$, $c'(a) \geq \frac{c(a)}{2}$, the maximum value of a flow on $\lfloor \mathcal{D} \rfloor$ is at least $\frac{1}{2}\frac{c_{\max}}{c_{\min}} f_0$. We then apply Theorem 4.4 to $\lfloor \mathcal{D} \rfloor$ wich uses integral capacities in $[\lfloor \frac{c_{\max}}{c_{\min}} \rfloor]$ and get an ospf-flow with value at least $\mathcal{H}(\lfloor \frac{c_{\max}}{c_{\min}} \rfloor)^{1-L_D(s,t)} \frac{1}{2}\frac{c_{\max}}{c_{\min}} f_0$. We then multiply this flow by $c_{\min}$ and get the result. □