[6] C. Lund and M. Yannakakis, *On the Hardness of Approximating Minimization Problems*, 25th STOC, 1993.

[7] C. Lund and M. Yannakakis, *The Approximation of Maximum Subgraph Problems* 20th ICALP, 1993.

[8] C. Papadimitriou and K. Steiglitz, Combinatorial Optimization. Algorithms and Complexity, *Prentice Hall*, 1982.

[9] I. Tomescu, *Hypertrees and Bonferroni inequalities*, J. Combin. Theory Ser. B 41(1986), 209-217.

[10] I. Tomescu, *Ordered h-Hypertrees*, Discrete Mathematics 195(1992), 241-248.

[11] I. Tomescu and M. Zimand, *Optimal Spanning Hypertrees*, Discrete Appl. Mathematics (to appear).

Now observe that:

$$c(T) = \sum_{p,q,a \text{ graphic answer to } p,\ b \text{ graphic answer to } p)} c((p,1),(p,2),(p,a,q,b)) +$$
$$\sum_{p,q,a \text{ graphic answer to } p,\ b \text{ graphic answer to } p)} c((1,q),(2,q),(p,a,q,b)) =$$
$$\sum_{p,q} \pi^{-1}(p,q) = 2^{l(n)} \cdot ACC_{V,(A,B)}$$

It follows that $max_T c(T) \leq 2^{l(n)} ACC_V$.

Conversely, if (A, B) is a pair of prover strategies for which $ACC_V(x) = ACC_{V,(A,B)}(x)$, then we can order X as suggested above such that for each p, $a_p = A(p)$, and for each q, $b_q = B(q)$. For the corresponding hypertree T

$$c(T) = 2^{l(n)} \cdot ACC_{V,(A,B)} = 2^{l(n)} \cdot ACC_V(x)$$

In conclusion, $max_T c(T) = 2^{l(n)} \cdot ACC_V(x)$. □

References

[1] M. Bellare, *Interactive Proofs and Approximation*, IBM T.J. Watson Research Center, TR RC17969, 1992.

[2] M. Bellare and P. Rogovay, *The complexity of approximating a nonlinear program*, Report No. RC 17831, IBM Research, Yorktown heights, NY, (march 1992).

[3] U. Feige and L. Lovász, *Two-Prover One Round Systems: Their Power and Their Problems*, STOC 1992.

[4] U.Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy, *Approximating Clique is almost NP-complete*, FOCS 1991.

[5] D.S. Johnson, *The NP-Completeness Column: An Ongoing Result*, Journal of Algorithms 13 (1992), 502-524.

For each p, for each pair of 4-tuples u, v that are $p-distinct$, define:

$$c(u,v,y) = -M,\ \forall y \in X \setminus \{(p,2)\}$$

For each q, for each pair of 4-tuples u, v that are $q-distinct$ but not $p-distinct$ for any p, define:

$$c(u,v,y) = -M,\ \forall y \in X \setminus \{(2,q)\}$$

In this way between the two p-stakes there can be no pair of $p-distinct$ 4-tuples and between the two q-stakes there can be no pair of $q-distinct$ 4-tuples.

v) For the remaining triplets, $c(u,v,w) = 0$.

We take $M = \binom{N^4+4N+2}{2}N + 1$. Let $T = (X,E)$ be an 3-hypertree achieving the maximum cost and let $<$ be the induced order. Since any 3-hypertree over X has $\binom{N^4+4N+2}{2}$ edges, each of cost less than N, if T has an edge of cost $-M$, then $c(T) < 0$. So suppose that T does not have edges of cost $-M$. One can check that there is an ordering of X avoiding such edges. Indeed, order $s_3 < s_1 < s_2$, then place all the stakes in the "good" order, then for each $p, q \in \{0,1\}^{l(n)}$ fix some $a_p, b_q \in \{0,1\}^{l(n)}$ and place between the two q-stakes all the elements of the form (p, a_p, q, b_q) and finally throw all the remaining elements after the last stake. It follows that the maximum tree T possess such an ordering. By the commentaries following the definition of the costs, for any p, between the p-stakes there exist elements of the form $(p,*,*,*)$ (from iii)), and there is no pair of p-distinct elements between the p-stakes (from iv)). If element $(p,a,*,*)$ is between the p-stakes, call a the graphic answer to p. It follows that each p has a unique graphic answer a. Similarly, each q has a unique graphic answer b. By the property 3 of the position of stakes, for any 4-tuple (p,a,q,b), a is the graphic answer to p if and only if b is the graphic answer to q. These graphic answers define a pair of prover strategies (A,B) such that $A(p) =$ the graphic answer to p, and $B(q) =$ the graphic answer to q.

$u_1 < u_2$. The relations ii.3) imply the natural order on u_i's. The important fact to retain is that if a tree T does not have edges of cost $-M$, then, in the order induced by that tree, we have:

$$s_1 < s_2 < u_1 < u_2 < \ldots < u_{4N}$$

and

$$s_3 < u_1$$

In consequence, property 3 is respected by the induced order of the tree.

iii) For all $p, q, a, b \in \{0,1\}^{l(n)}$, define:

iii.1) $c(s_1, (p,a,q,b), y) = -M, \ \forall y \in X \setminus \{(p,1)\}$, if $p \neq 0^{l(n)}$(i.e. $(p,1) \neq u_1$)

iii.2) $c(s_3, (p,a,q,b), y) = -M, \ \forall y \in X \setminus \{(p,1)\}$, if $p = 0^{l(n)}$(i.e. $(p,1) = u_1$)

iii.3) $c((p,1),(p,2),y) = -M, \ \forall y \in X \setminus \{(p,a',q',b') \mid a', q', b' \in \{0,1\}^{l(n)}\}$

iii.4) $c((p,1),(p,2),(p,a,q,b)) = \begin{cases} (1/2)|\pi^{-1}(p,q)|, & \text{if } \rho(x,paqb) = 1 \\ 0, & \text{otherwise} \end{cases}$

iii.5) $c((1,q),(2,q),y) = -M, \ \forall y \in X \setminus \{(p',a',q,b') \mid p', a', b' \in \{0,1\}^{l(n)}\}$

iii.6) $c((1,q),(2,q),(p,a,q,b)) = \begin{cases} (1/2)|\pi^{-1}(p,q)|, & \text{if } \rho(x,paqb) = 1 \\ 0, & \text{otherwise} \end{cases}$

where $\pi^{-1} = \{r \in \{0,1\}^{l(n)} \mid \pi(x,r) = (p,q)\}$. Relations iii.1) and iii.2), in conjunction with the ordering induced on X by definitions ii), imply that no vertex of the form (p,a,q,b) can be placed in front of the first p-stake. From iii.3). we deduce that, in trees avoiding edges of cost $-M$, for each p, at least one element of the form (p,a,q,b) is placed between the p-stakes. The cost of the corresponding edge is equal to $(1/2)2^{l(n)} \cdot$ Prob $\{$ V accepts $x \mid \pi(x) = (p,q)\}$. A similar property holds for each q.

iv) We call a pair of 4-tuples $(p_1,a_1,q_1,b_1),(p_2,a_2,q_2,b_2)$ $p-$*distinct* if $p_1 = p_2 = p$ and $a_1 \neq a_2$, and $q-$*distinct* if $q_1 = q_2$ and $b_1 \neq b_2$.

i.1) $c(s_2, s_3, y) = -M,\ \forall y \in X \setminus \{s_1\}$

i.2) $c(s_2, s_1, y) = -M,\ \forall y \in X \setminus \{s_3, s_4\}$

i.3) $c(s_2, s_4, y) = -M,\ \forall y \in X \setminus \{s_1\}$

If T does not have edges of cost $-M$, then $s_1 < s_2$ in the order induced by T. Indeed, from the first definition, either $s_1 < s_2$ or $s_1 < s_3$. Suppose that $s_2 < s_1$ and $s_1 < s_3$. From i.2), it follows that $s_4 < s_1$. But i.3) implies $s_1 < s_2$ or $s_1 < s_4$. So, the only possibility is $s_1 < s_2$.

ii) Let $S = \bigcup_{p \in \{0,1\}^{l(n)}} S_p \ \cup \ \bigcup_{q \in \{0,1\}^{l(n)}} S_q$ be the set of all $4N$ stakes. They have either the form (p, i) or (i, q) with $i \in \{1, \ldots, N\}$ and $p, q \in \{0,1\}^{l(n)} = \{x_1 < x_2 < \ldots x_N\}$ (i.e., we have ordered $\{0,1\}^{l(n)}$). We order S in the following way:

$$(x_1, 1) < (x_2, 1) < \ldots < (x_N, 1) < (1, x_1) < (1, x_2) \ldots (1, x_N) <$$

$$(2, x_1) < (2, x_2) < \ldots < (2, x_N) < (x_1, 2) < (x_2, 2) \ldots (x_N, 2) <$$

Note that for all $p, q \in \{0,1\}^{l(n)}$

$$(p, 1) < (1, q) < (2, q) < (p, 2) \tag{3}$$

(i.e., the two p-stakes and the two q-stakes are interleaved.)

For the sake of the next definition, rename the elements of S by $u_1 < u_2 \ldots < u_{4N}$. Define:

ii.1) $c(s_1, u_1, y) = -M,\ \forall y \in X \setminus \{s_2\}$

ii.2) $c(s_3, u_2, y) = -M,\ \forall y \in X \setminus \{u_1\}$,

ii.3) $c(u_{i-2}, u_i, y) = -M,\ \forall y \in X \setminus \{u_{i-1}\},$ for $i = 3, \ldots, 4N$

Observe that u_1 coincides with s_4 from the previous group of definitions and thus there is no conflict among the relations defining c. Note that from ii.2), we deduce that $s_1 < s_2 < u_1$. Since $u_1 = s_4$, from i.2), we get that $s_3 < s_1$ or $s_3 < s_2$. In both cases, we have $s_3 < u_1$. So, from ii.2), we deduce that

we further force each two 4-tuples that contain distinct answers to p to not be situated both in front of $(p, 2)$. In this way, the answer to p is uniquely determined by the order of T, since there can be just one type of 4-tuples $(p, a, *, *)$ between the two p-stakes, i.e., the a is unique. We make a similar construction for each q, fixing a question to the second prover. Since we need to make the cost of the maximum tree equal to ACC_V, we also want, for each fixed pair $p, q \in \{0,1\}^{l(n)}$, to have a unique 4-tuple (p, a, q, b) that is placed simultaneosly between the p-stakes and the q-stakes. This condition is realized by forcing the two q-stakes to be placed between the two p-stakes, for all $p, q \in \{0,1\}^{l(n)}$. Forcing specific order relations between the nodes of the maximal tree is done using different combinations of the following gadget. If for a triplet of nodes, u, v, w we have $c(u, v, x) = -M$ (here M is a very large value) for all nodes x except w, then w must not be the last of the triplet u, v and w in the order induced by the maximum tree, because otherwise we cannot avoid having a 3-edge of the form (u, v, y) with $y \neq w$, and which therefore has cost $-M$ (the only possibility could be to place u, v and w in the first three positions, but in this case we can permute them in order to have, say w between u and v, without affecting the cost of the tree).

We proceed to formally define $H = (X, E)$. The set X of nodes consists of nodes of different types:

$$X = X_1 \cup \bigcup_{p \in \{0,1\}^{l(n)}} S_p \cup \bigcup_{q \in \{0,1\}^{l(n)}} S_q \cup \{s_1, s_2, s_3\}$$

where

(a) $X_1 = \{(p, a, q, b) \mid p, a, q, b \in \{0,1\}^{l(n)}\}$, (there are N^4 such nodes).

(b) For each $p \in \{0,1\}^{l(n)}$, $S_p = \{(p, 1), (p, 2)\}$ (these are the two p-stakes),

(c) For each $q \in \{0,1\}^{l(n)}$, $S_q = \{(1, q), (2, q)\}$ (these are the two q-stakes),

(d) s_1, s_2 and s_3 are three distinct special nodes.

In total, there are $N^4 + 4N + 3$ nodes in X.

The function cost is defined as follows:

i) Let s_4 be the first stake in lexicographical order, i.e.$s_4 = (0^{l(n)}, 1)$. Define:

Theorem 4 *Suppose $MIP(2,1)$ reduces to the NP-optimization problem (S,g). Then this problem is hard to approximate in the following sense*

1. *There is a constant $\epsilon > 0$ such that the existence of a $DTIME[n^{\log^\epsilon n}]$ approximation algorithm A with $r_A(I) = 2^{\log^\epsilon n}$ for (S,g) implies $NP \subseteq DTIME[n^{\log^\epsilon n}]$.*

2. *There is a constant $\mu > 0$ such that the existence of a polynomial time approximation algorithm A with $r_A(I) = \mu$ for (S,g) implies $P = NP$.*

Note that in case of a maximization problem, the approximation ratio is defined by

$$r_A(I) = \frac{opt(I)}{A(I)}$$

We will show that MIP(2,1) reduces to the *max* version of the optimal spanning hypertree problem. By the above result, we conclude that if $P \neq NP$, then $r_A(I) \geq k$ (ct.), and if $NP \not\subseteq DTIME(n^{\log^{O(1)} n})$, then $r_A(I) \geq 2^{\log^\epsilon n}$, for all polynomial time , respectively quasi-polynomial time approximation algorithms A for this problem.

Theorem 5 *MIP(2,1) reduces to maximum - 3 - Spanning Hypertree (MAX-3-SHT).*

Proof. Let $V = (\rho, \pi, l)$ be a verifier strategy and let $x \in \{0,1\}^n$. We construct a complete 3 - hypergraph H whose maximum spanning hypertree T, verifies $C(T) = 2^{l(n)} \cdot ACC_V(x)$.

We first overview the construction, in the hope that it will become clearer. In the description below, all spatial references (i.e., in front of, after, between) refer to the order induced by the maximum hypertree. Let $N = 2^{l(n)}$. For each $p \in \{0,1\}^{l(n)}$ we want to specify a unique $a \in \{0,1\}^{l(n)}$, which will represent the answer of the first prover to question p. Similarly, for the second prover, we want to specify for each possible question $q \in \{0,1\}^{l(n)}$, a unique answer $b \in \{0,1\}^{l(n)}$. In order to simulate the $MIP(2,1)$ system, we represent the pairs *(question, answer)* by 4-tuples (p,a,q,b). To achieve the above uniqueness, we introduce for which p the *stakes* $(p,1),(p,2)$, we force them to be ordered by the maximum tree in the natural order $(p,1) < (p,2)$, we force at least one 4-tuple which has p as the first component to be placed between the two p-stakes,and

the input and l is a polynomial function. More formally, a *verifier strategy* is a triple of polynomial time computable functions (ρ, π, l) satisfying

1. $l : N \rightarrow N$ is polynomially bounded;

2. $\pi : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^* \times \{0,1\}^*$ is a polynomial time computable function which on inputs $x \in \{0,1\}^n$ and $R \in \{0,1\}^n$ returns a pair (p, q) of strings, each of length $l(n)$;

3. $\rho : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}$ is a polynomial time computable function giving the final verdict.

A *prover strategy* is a function $A : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ whose output is of length equal to the length of its input.

Let $V = (\rho, \pi, l)$ be a verifier strategy and (A, B) a pair of prover strategies. For each $x \in \{0,1\}^n$, let $ACC_{V,(A,B)}(x)$ denote the probability that $\rho(x, p.A(x,p).q.B(x,q)) = 1$, when R is chosen randomly in $\{0,1\}^{l(n)}$ and (p, q) is set to $\pi(x, R)$. The *value of the verifier strategy V at x* is the maximum of $ACC_{V,(A,B)}(x)$ over all possible pairs (A, B) of prover strategies, and we denote it by ACC_V. We say that V has error-probability ϵ with respect to L (where $\epsilon : N \rightarrow R$ and $L \subseteq \{0,1\}^*$) if

1. $x \in L$ implies $ACC_V(x) = 1$, and

2. $x \notin L$ implies $ACC_V(x) < \epsilon(|x|)$.

We say that L has a two-prover, one-round proof ($MIP(2,1)$)if there exists a verifier strategy V which has error-probability $1/2$ with respect to L.

Let (S, g) be an NP-optimization problem. We say that $MIP(2,1)$ reduces to (S, g) if for any verifier strategy $V = (\rho, \pi, l)$ there is a function Φ_V which maps a common input x for the verifier strategy into an instance $\Phi_V(x)$ of the optimization problem and has the following properties:

1. For all $x \in \{0,1\}^*$, $g^*(\Phi_V(x)) = 2^{l(|x|)} \cdot ACC_V(x)$;

2. $\Phi_V(x)$ is computable in time polynomial in $2^{l(|x|)}$.

Based on a result due to Feige and Lovász [3] which shows that SAT has $MIP(2,1)$ with small error-probability and polylogarithmic communication complexity $l(n)$, the following theorem is stated in [1]:

There exists a polynomial q such that $|I| \leq q(m)$. Clearly, for sufficiently large m it holds that $2^m \geq p(q(m)) \geq p(|I|)$. If we would have a polynomial time algorithm A with

$$A(I) \leq p(|I|)opt(I), \ \forall I$$

then by applying it to the above instance I, we could decide if $C \in$ SAT or not in polynomial time. It would follow SAT $\in$ P, and, consequently, P = NP. $\square$

The analysis of approximation algorithms for the max version of the optimum spanning hypertree problem requires a distinct reduction. We use the multi prover interactive approach which has been first discovered by Feige et. al in their seminal paper [4]. The method has been used afterwards with big success in establishing the complexity of approximation for many problems (see [1], [2], [6], [7]; for an early survey, see [5]). We use the formalization of the method as synthesized in [1].

An *NP-optimization problem* is a pair (S, g), where $S : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}$ and $g : \{0,1\}^* \times \{0,1\}^* \rightarrow R$ are functions computable in time polynomial in the length of their first input. We call $y \in \{0,1\}^*$ a solution to w if $S(w, y) = 1$. We denote $g^*(w) = max_{y \in S(w)} g(w, y)$, where $S(w)$ is the set of solutions to w. For example, if H is a complete hypergraph, $S(H)$ is the set of all hypertrees over H and $g^*(H)$ is the cost of the maxmum such hypertree.

A two-prover, one-round interactive proof system , $MIP(2,1)$, involves three parties: a probabilistic, polynomial time *verifier* and a pair of arbitrarily powerful, deterministic *provers*, A and B, that cannot communicate to one another during the following protocol. The parties share a common input x, and it is the provers (joint) goal to convince V to accept x. The interaction between V, A and B runs as follows. The verifier computes $\pi(x, R) = (p, q)$ a pair of questions, based on the common input x and his private random bits R, and he sends question p to A and question q to B. The provers respond with the answers $a = A(x, p)$, and respectively $b = B(x, q)$. After the verifier receives the answers a and b, he computes $\rho(x, paqb)$ which tells him whether to accept or not. We suppose that p, q, a, b all have length $l(n)$, where n is the length of

an h-hypertree T'' such that $c^*(T'') \leq c^*(T') \leq k$, and $d_{T''}(y) = \binom{n-2}{h-2}$, i.e. y is a terminal vertex for T''. Denote by $E_1, \ldots, E_q$ all edges containin y in T''. Then $E_1 \backslash \{y\}, \ldots, E_q \backslash \{y\}$ induce an h-hypertree T with vertex set X having $c(T) = c^*(T'') \leq k$. □

3 Approximation Algorithms

The proofs of Theorems 1 and 2 show that MSH, for $h \geq 3$ is NP-complete even if we restrict the cost function to take values only in $\{1, 2\}$ (or $\{0, 1\}$). It follows that MSH is strongly NP-complete, which implies that no fully polynomial time approximation schema exists for MSH, $h \geq 3$ (for example see [8] for the appropriate definitions).

In fact the situation is much worse. We consider first the *min* version of MSH, $h = 3$. Let I be an instance of this problem and denote $opt(I)$ the cost of the minimum spanning 3-hypertree of I. If A is an algorithm for the same problem, we denote by $A(I)$ the cost of the spanning tree found by A. The approximation ratio for the algorithm A is defined as

$$r_A(I) = \frac{A(I)}{opt(I)}$$

Observe that $r_A(I) \geq 1$ for all I and all A. We would like $r_A(I)$ to be exactly 1 (or as close as possible).

Theorem 3 *Let p be a polynomial with natural coefficients. There exists no polynomial time algorithm A for MSH, $h = 3$, such that $r_A(I) \leq p(|I|)$ for all instances I, unless $P = NP$ ($|I|$ is the length size of the instance I).*

Proof. In the proof of Theorem 1, we modify the value 2 by $2^m\binom{8m+2}{2}$ +1 in cases i) - v) in the definition of the cost function c. Note that, when encoded in binary, the length of $2^m\binom{8m+2}{2}$ +1 is still polynomial in m, hence polynomial in the length of C, (recall that C is the instance of 3-SAT that we transform in an instance I of MSH, $h = 3$). As in Theorem 1, we get that

$$C \in \text{SAT} \implies opt(I) = \binom{8m+2}{2}$$

$$C \notin \text{SAT} \implies opt(I) \geq (1 + 2^m)\binom{8m+2}{2}$$

In order to prove the converse, we need the following Lemma. (For other similar exchange properties of hypertrees, see [11].)

Lemma 2 *Let T be an h-hypertree having vertex set X. Then for any non-terminal vertex $x \in V(T)$ there exist $u \in E(T), v \in \binom{X}{h} \setminus E(T)$ such that $x \in u$ and $x \notin v$ and $(E(T)\setminus\{u\})\bigcup\{v\}$ is the edge set of an h-hypertree over X.*

Proof. The property is obvious for trees. Let $h \geq 3$ and suppose that the property holds for every $(h-1)$-hypertree. Since x is not a terminal vertex it follows that $x \neq x_{i_n}$ (x_{i_n} is defined by the order (1) for T). Also it follows that x is not a terminal vertex in at least one of the $(h-1)$-hypertrees $T_2, T_3, \ldots, T_{n-h+1}$ from (1). Indeed, if this property does not hold, then we deduce that

$$d_T(x) \leq \sum_{i=h-3}^{n-3} \binom{i}{h-3} = \binom{n-2}{h-2}$$

Hence, by Lemma 1, one has $d_T(x) = \binom{n-2}{h-2}$ and x is a terminal vertex of T, which contradicts the hypothesis. Hence there exists $p \geq h+1$ such that $x \in V(T_{p-h+1})$ and x is not a terminal vertex for this $(h-1)$-hypertree. By the induction hypothesis there exist edges $u_1 \in E(T_{p-h+1})$, $v_1 \in \binom{V(T_{p-h+1})}{h-1} \setminus E(T_{p-h+1})$ such that $x \in u_1$, $x \notin v_1$ and $(E(T_{p-h+1}) \setminus \{u_1\}) \cup \{v_1\}$ is the edge set of an $(h-1)$-hypertree U_{p-h+1} having vertex set $V(T_{p-h+1})$. Let $u = u_1 \cup \{x_{i_p}\}$ and $v = v_1 \cup \{x_{i_p}\}$. Clearly $x \in u$ and $x \notin v$. We deduce that $v \notin E(T)$ since the family of edges $\{w \cup \{x_{i_{h+r-1}}\} \mid w \in E(T_r)\}$ has only edges that do not contain x_{i_p} for $2 \leq r \leq p-h$, and for $r \geq p-h+2$ only edges containing vertices x_{i_s} with $s \geq p+1$. Hence $(E(T) \setminus \{u\}) \cup \{v\}$ is the edge set of an h-hypertree with vertex set X, which is generated by the order (1) of T with the exception that T_{p-h+1} is replaced by U_{p-h+1} . □

(*Continuation of Proof of Theorem 2*) For the converse direction, let T' be an $(h+1)$-hypertree such that $v(T') = Y$ and $c^*(T') \leq k$. If y is not a terminal vertex for T', then by Lemma 2, we can obtain another h-hypertree U such that $c^*(U) \leq c^*(T')$ and $d_U(y) = d_T(y) - 1$. By repeating this procedure we arrive to

$$
\begin{aligned}
< \quad & \{x_1, k_2\} < \{x_1, k_2^{'}\} < \{\bar{x}_2, k_2\} < \{\bar{x}_2, k_2^{'}\} < \{x_1, k_3\} < \{x_1, k_3^{'}\} \\
< \quad & \{\bar{x}_4, k_3\} < \{\bar{x}_4, k_3^{'}\} < \{\bar{x}_4, k_4\} < \{\bar{x}_4, k_4^{'}\} < k_1 < k_1^{'} < k_2 < k_2^{'} < \\
< \quad & k_3 < k_3^{'} < k_4 < k_4^{'} < \{x_3, k_2\} < \{x_3, k_2^{'}\} < \{x_2, k_3\} < \{x_2, k_3^{'}\} < \\
< \quad & \{\bar{x}_1, k_4\} < \{\bar{x}_1, k_4^{'}\} < \{x_3, k_4\} < \{x_3, k_4^{'}\}. \ \square
\end{aligned}
$$

Theorem 2 *The problem MSH is NP-complete for every $h \geq 4$.*

Proof. We first argue that MSH is in NP for every $h \geq 4$. Indeed an h-hypertree is completely specified if the order (1) is given and each $(h-1)$-hypertree T_r, $r = 2, \ldots, n-h+1$ of (1) is also completely specified. If $S(n,h)$ denotes the size of a complete specification of an h-hypertree over n vertices, then we get the following recurrece:

$$S(n,h) = S(h,h-1) + S(h+1,h+1) + \ldots S(n-1,h-1) + O(n \log n)$$

Now it is easy to see by induction on h that $S(n,h) \leq n^h$ for every naturals n and h. Since h is considered to be fixed, the size of a "witness" for (h)- MSH is polynomial in the size of the input instance. In consequence, MSH is in NP for all h.

In order to prove the NP-completeness, we show that for $h \geq 3$ the problem MSH for h-hypertrees can be reduced to the MSH for $(h+1)$-hypertrees. For a given function $c : \binom{X}{h} \rightarrow R_+$ and a threshold value $k \in R_+$, we denote the corresponding problem by MSH(X,h,c,k). Let $y \notin X$, $Y = X \cup \{y\}$ and $c^* : \binom{Y}{h+1} \rightarrow R_+$ be defined by $c^*(u) = 0$ if $u \subseteq X$ and $c^*(u) = c(v)$ otherwise, where $v = u \backslash \{y\}$. Then there exists an h-hypertree T such that $c(T) \leq k$ for the problem MSH(X,h,c,k) if and only if there exists an $(h+1)$-hypertree T' such that $c^*(T') \leq k$ for the problem MSH$(Y,h+1,c^*,k)$. Indeed, suppose that there exists T such that $c(T) \leq k$ and let the order $\mu : x_1 < x_2 < \ldots < x_n < y$ be defined on Y. For all $(h+1)$-hypertrees generated by μ the vertex y is a terminal vertex and we consider an $(h+1)$-hypertree H such that $V(H) = X = \{x_1, x_2, \ldots, x_n\}$. By the definition of c^* it follows that $c^*(H) = 0$. Let T' be the $(h+1)$-hypertree generated by μ such that $E(T') = E(H) \cup \{z \cup \{y\} \mid z \in E(T)\}$. We have $c^*(T') = c^*(H) + c(T) = c(T) \leq k$.

Relatively to v), let us remark that for each triplet (c_i, c_j, α), with $i < j$, and disjunctions c_i, c_j, we have to avoid the edges of cost 2 $(\{\alpha, k_i\}, \{\bar{\alpha}, k_j\}, x)$, with $x \in X \backslash \{k_j\}$ This can be achieved if k_j is not the greatest in the set $\{\{\alpha, k_i\}, \{\bar{\alpha}, k_j\}, k_j\}$. Similarly, $k_j^{'}$ should not be the greatest in the set $\{\{\alpha, k_i^{'}\}, \{\bar{\alpha}, k_j^{'}\}, k_j^{'}\}$. Consequently, if T does not contain edges of cost equal to 2, then: under hypothesis I, it is not possible that a literal α satisfies graphically k_i and $\bar{\alpha}$ satisfies graphically k_j, and under the hypothesis II it is not possible that α satisfies graphically $k_i^{'}$ and $\bar{\alpha}$ satisfies graphically $k_j^{'}$.

Suppose now that there exists a 3-hypertree T with vertex set X such that $c(T) \leq \binom{8m+2}{2}$. It follows that T contains only edges u such that $c(u) = 1$. Hence we have two cases I: $s_1 < s_2$ or II: $s_1 < s_3$. If $s_1 < s_2$ we have seen that each vertex k_i is satisfied graphically by a literal α contained in c_i for $i = 2, \ldots, m$, hence α may satisfy c_i as a propositional variable. Recall that by our stipulation, c_1 can be satisfied independently from the other clauses c_i, $i = 2, \ldots, m$. Since α and $\bar{\alpha}$ do not satisfy simultaneously c_i and c_j having α in conflict, it follows that there exists an assignment of truth values to propositional variables such that all clauses of C are satisfiable. A similar conclusion holds if $s_1 < s_3$.

Suppose now that there is an assignment A of truth values to propositional variables such that all clauses $c_i, \ldots c_m$ have logical value true. We consider the following total order on X:

$$\begin{aligned} s_1 < s_2 < s_3 < S_1 < S_2 < \ldots < S_m < k_1 < k_1^{'} < k_2 < k_2^{'} < \ldots \\ < k_m < k_m^{'} < S_1^{*} < S_2^{*} < \ldots < S_m^{*}, \end{aligned} \tag{2}$$

where S_i (respectively S_i^{*}) denote the sequence of pairs $\{\alpha, k_i\}$ and $\{\alpha, k_i^{'}\}$ such that α appears in c_i and α satisfies (respectively, does not satisfy) c_i relative to the assignment A. It is easy to check that there is a 3-hypertree T induced by the total order (2) such that $c(T) = \binom{8m+2}{2}$. For example, for $m = 4$ and $c_1 = u \vee y \vee z$, $c_2 = x_1 \vee \bar{x_2} \vee x_3$, $c_3 = x_1 \vee x_2 \vee \bar{x}_4$ and $c_4 = \bar{x}_1 \bigvee x_3 \bigvee \bar{x}_4$ an assignment A satisfying c_1, c_2, c_3, c_4 is $u = T$, $y = T$, $z = T$, $x_1 = T$, $x_2 = F$, $x_3 = F$, $x_4 = F$. In this case the total order on X is :

$$s_1 \quad < \quad s_2 < s_3 < \{u, k_1\} < \{u, k_1^{'}\} < \{y, k_1\} < \{y, k_1^{'}\} < \{z, k_1\} < \{z, k_1^{'}\}$$

v) for any triplet (c_i, c_j, α) with $i < j$ and c_i, c_j having literal α in conflict (say, $\alpha \in c_i, \bar{\alpha} \in c_j$) we define

$$c(\{\alpha, k_i\}, \{\bar{\alpha}, k_j\}, x) = 2 \text{ for any } x \in X \backslash \{k_j\},$$

$$c(\{\alpha, k_i^{'}\}, \{\bar{\alpha}, k_j^{'}\}, x) = 2 \text{ for any } x \in X \backslash \{k_j^{'}\},$$

vi) for the remaining triplets (u, v, w), we define $c(u, v, w) = 1$.

The threshold is k = $\binom{8m+2}{2}$ (note that $|X| = 8m + 3$). Consider now a 3-hypertree $T = (X, E)$ defined by (1) and a total order $<$ on X induced by (1). From i) in order to avoid edges (s_2, s_3, x) where $x \neq s_1$ we must have

$$I: \ s_1 < s_2$$

or

$$II: \ s_1 < s_3$$

If $E_1 = \{s_1, s_2, s_3\}$ then these vertices may be permuted in any order and I and II are satisfied. Under the hypothesis I, we can avoid edges whose cost is defined by ii) only if $s_1 < s_2 < k_1 < k_2 < \ldots < k_m$ and similarly, under the hypothesis II, edges whose cost is defined by iii) can be avoided only if $s_1 < s_3 < k_1^{'} < k_2^{'} < \ldots < k_m^{'}$.

Relatively to the definition iv), under the hypothesis I, to avoid edges of cost equal to 2 it is necessary that for all $i = 2, \ldots, m$ each vertex k_i to be greater (relatively to $<$) than at least one of the vertices $\{\alpha_{i,1}, k_i\}, \{\alpha_{i,2}, k_i\}$ and $\{\alpha_{i,3}, k_i\}$ if at ii) all edges of cost 2 are avoided (i.e. $s_1 < k_i$ for all $i = 1, \ldots, m$). We say that the vertex k_i is satisfied graphically by the literal α if $\{\alpha, k_i\} < k_i$. It follows that in this case, each vertex k_i is satisfied graphically by at least a literal α contained in c_i for $i = 2, \ldots, m$. Similarly, under the hypothesis II, if all edges of cost 2 defined by iii) and iv) are avoided, then, for all $i = 2, \ldots, m$, each vertex $k_i^{'}$ is satisfied graphically by at least a literal α contained in the disjunction c_i since $\{\alpha, k_i^{'}\} < k_i^{'}$, for $i = 2, \ldots, m$.

2 NP-Completeness of MSH

Since the proofs of the NP-completeness of MSH are different for $h = 3$ and $h > 3$, we state these results in two separate theorems.

Theorem 1 *The problem of the minimum spanning hypertree (MSH) is NP-complete for h=3.*

Proof It is not difficult to see that for $h = 3$, MSH is in NP. Indeed, for a fixed total order μ defined on X by 1, we can decide in polynomial time if there exists a spanning 3-hypertree T_μ relative to μ such that $C(T_\mu) \leq k$ and the length of an encoding of μ is polynomial in the length of an instance of MSH.
Now we prove that for $h = 3$ the problem 3-SAT is reducible to the problem MSH. Let $C = \{c_1, c_2, \ldots, c_m\}$ be an instance of the problem 3-SAT where $|c_i| = 3$ for $1 \leq i \leq m$. W.l.o.g we impose $c_1 \cap c_i = \emptyset$, for $2 \leq i \leq m$. We define an instance of the problem MSH as follows:

$$X = \{\{s_1, s_2, s_3, k_1, k_1^{'}, k_2, k_2^{'}, \ldots, k_m, k_m^{'}\}$$
$$\cup \bigcup_{i=1}^{m} \{\{\alpha, k_i\}, \{\alpha, k_1^{'}\} \mid \alpha \text{ is a literal in } c_i\}\}$$

and the cost function $c : \binom{X}{3} \rightarrow R_+$ by the following rules:

i) $c(s_2, s_3, x) = 2$ for any $x \in X \backslash \{s_1\}$;

ii) $c(s_1, k_1, x) = 2$ for any $x \in X \backslash \{s_2\}$ and for every $i = 2, \ldots, m$, $c(s_2, k_i, x) = 2$ for any $x \in X \backslash \{k_{i-1}\}$;

iii) $c(s_1, k_1^{'}, x) = 2$ for any $x \in X \backslash \{s_3\}$ and for every $i = 2, \ldots, m$, $c(s_3, k_i^{'}, x) = 2$ for any $x \in X \backslash \{k_{i-1}^{'}\}$;

iv) for every disjunction

$$c_i = \alpha_{i,1} \vee \alpha_{i,2} \vee \alpha_{i,3} \ (2 \leq i \leq m)$$

we define

$$c(s_1, k_i, x) = 2 \text{ for any } x \in X \backslash \{\{\alpha_{i,1}, k_i\}, \{\alpha_{i,2}, k_i\}, \{\alpha_{i,3}, k_i\}\}$$

and

$$c(s_1, k_i^{'}, x) = 2 \text{ for any } x \in X \backslash \{\{\alpha_{i,1}, k_i^{'}\}, \{\alpha_{i,2}, k_i^{'}\}, \{\alpha_{i,3}, k_i^{'}\}\}$$

We consider an equivalent problem. Namely, for a cost function $c:\binom{X}{h} \rightarrow R$, find an h-hypertree T on vertex set X (hence T is a spanning tree of the complete h-hypergraph K_n^h) such that its cost $c(T) = \sum_{u \in E(T)} c(u)$ is optimum (minimum or maximum).

Without loss of generality we can consider $c : \binom{X}{h} \rightarrow R_+$, since if there exists $u \in \binom{X}{h}$ having $c(u) < 0$, then by denoting $p = min_u c(u)$ and $c^*(u) = c(u) - p$ for every $u \in \binom{X}{h}$, we have $c^*(u) \geq 0$ and $c^*(T) = c(T) - p\binom{n-1}{h-1}$ by Lemma 1. In a similar manner, we can reduce the maximum problem to the minimum problem. Indeed, if $r = max_u c(u)$ and we define $c'(u) = r - c(u)$ for all $U \in \binom{X}{h}$, then it is easy to see that

$$max_T \ c(T) = r\binom{n-1}{h-1} - min_T \ c'(T)$$

We define the problem MSH (Minimum Spanning Hypertree) as follows:

Instance: Let $h \geq 2$, X be a finite set of cardinality $n > h$, $c:\binom{X}{h} \rightarrow R_+$, $k \in R_+$.

Question: There exists an h-hypertree T on vertex set T such that its cost satisfies $c(T) \leq k$?

It is well known that the problem MSH can be solved in polynomial time for $h = 2$ (see for example the algorithms by Kruskal, Borůvka, Prim). We show here that for $h \geq 3$, MSH is NP-complete. The proof of this result is in Section 2. In Section 3 we show that it is highly improbable that MSH admits polynomial time approximation algorithms, even if we allow an as bad approximation ratio as $p(|I|)$, where p is any polynomial and $|I|$ is the length of the instance of the problem. The existence of such an algorithm would imply $P = NP$. For the *maximum* variant of this problem, not even a quasipolynomial algorithm achieving the rather bad approximation ratio $2^{\log^{\epsilon} n}$ can exist, unless $NP \subseteq DTIME[n^{\log^{O(1)} n}]$

not belong to E_1 and $E(T)$ is composed from E_1 and the family of h-sets:

$$(\{Y \ \cup \ \{x_{i_{h+r-1}}\} \mid Y \in E(T_r)\})_{2 \leq r \leq n-h+1}.$$

If $E_1 = \{x_{i_1}, \ldots, x_{i_h}\}$, we say that T is a μ-ordered h-hypertree (see [9]) where $\mu : x_{i_1} < x_{i_2} < \ldots < x_{i_n}$ is a total order on X induced by (1). It is clear that every h-hypertree is a μ-ordered h-hypertree for many total orders μ on X which may be obtained e.g., by permuting the elements in E_1 or by considering other terminal vertices of T instead of x_{i_n}.

Consider a measure m defined on a finite set Z, $m : Z \rightarrow R_+$ such that $m(A) = \sum_{x \in A} m(x)$ for every $A \subseteq Z, A \neq \emptyset$ and $m(\emptyset) = 0$. Improved Bonferroni inequalities are obtained in [9] as follows:

Let T_n^h denote an h-hypertree with vertex set $\{1, \ldots, n\}$. The following inequalities hold:

$$m(\cup_{i=1}^n A_i) \leq \sum_{k=1}^{2p-1} (-1)^{k-1} S_k - \sum_{E_n^{2p}} m(A_{i_1} \cap \ldots \cap A_{i_{2p}})$$

for every $1 \leq p \leq n/2$, where the second sum is over all edges $\{i_1, \ldots i_{2p}\} \in E(T_n^{2p})$;

$$m(\cup_{i=1}^n A_i) \geq \sum_{k=1}^{2p} (-1)^{k-1} S_k - \sum_{E_n^{2p+1}} m(A_{i_1} \cap \ldots \cap A_{i_{2p+1}})$$

for every $1 \leq p \leq (n-1)/2$, where the second sum is over all edges $\{i_1, \ldots i_{2p+1}\} \in E(T_n^{2p+1})$; here $S_k = \sum m(A_{i_1} \cap A_{i_2} \cap \ldots \cap A_{i_k})$.

When compared to the classical Bonferroni inequalities, the above inequalities involve an additional term of the form

$$(-1)^{(h-1)} \sum_{E(T_n^h)} m(A_{i_1} \cap \ldots \cap A_{i_h})$$

where $h \leq n$ and the sum is over all edges $\{i_1, \ldots, i_h\}$ of an h-hypertree T such that $V(T) = X\{1, \ldots, n\}$. For a fixed h, in order to obtain the best upper (resp. lower) bounds for $m(\cup_{i=1}^n A_i)$ we must find an h-hypertree T such that $\sum_{u \in E(T)} m(\cap_{i \in u} A_i)$ is maximum (minimum) if we know all values $(m(\cap_{i \in u} A_i))_{u \in E(K_n^h)}$.

has $E(K_n^h) = \binom{X}{h}$, where vertex set X contains n vertices.

As in the case of standard graphs (i.e. $h = 2$) a proper notion of a hyper-tree is an important topic in the study of hypergraphs. The most appealing generalization (i.e. acyclic and connected) does not make sense in all situations. In [9] the combinatorial structure of a h-hypertree was introduced as a natural tool for obtaining Bonferroni type inequalities. It turned out that this notion is the correct generalization of the standard notion of a tree (see Th.1 in [10]).

An h-hypertree is an h-uniform hypergraph $T = (X, E)$ such that for $h = 2$, T is a tree with vertex set X and for $h \geq 3$, T is defined recursively by the following two rules:

i) If $X = \{x_1, \ldots, x_h\}$ then T has a unique edge $\{x_1, \ldots, x_h\}$.

ii) If $|X| \geq h + 1$ then there exists a vertex x_i such that if $E_1, \ldots, E_q$ denote all edges containing x_i then $E_1 \setminus \{x_i\}, \ldots, E_q \setminus \{x_i\}$ induce an (h-1)-hypertree with vertex set $X \setminus \{x_i\}$ and the remaining edges of T induce an h-hypertree with vertex set $X \setminus \{x_i\}$.

For $h = 2$ (ii) expresses a basic property of trees if we consider that every 1-tree is a singleton and we ignore the condition that the $(h-1)$-hypertree has vertex set $X \setminus \{x_i\}$: For any tree T there is at least one vertex x such that by deleting x and the unique edge incident to x the resulting graph is also a tree.

If $X = \{x_1, \ldots, x_h\}$ then all vertices $x_1, \ldots, x_h$ are called terminal vertices of T; otherwise $|X| \geq h + 1$ and any vertex $x_i \in X$ having property (ii) will be called a terminal vertex of T. The following result was proved in [9]

Lemma 1 *Every h-hypertree T of order n has $\binom{n-1}{h-1}$ edges and for any vertex $x \in V(T)$, $d_T(x) \geq \binom{n-2}{h-2}$ holds. Moreover $d_T(x) = \binom{n-2}{h-2}$ if and only if X is a terminal vertex of T.* $\square$

It follows that every h-hypertree $T = (X, E)$ is defined by a sequence

$$E_1, (T_2, x_{i_{h+1}}), (T_3, x_{i_{h+2}}), \ldots, (T_{n-h+1}, x_{i_n}) \tag{1}$$

where $E_1 \in \binom{X}{h}$, T_2 is an $(h-1)$-hypertree with vertex set E_1, and T_r is an $(h-1)$ -hypertree with vertex set $V(T_{r-1}) \cup \{x_{i_{h+r-2}}\}$ for any $r = 3, \ldots, n-h+1$. Vertices $x_{i_{h+1}}, \ldots, x_{i_n}$ are pairwise distinct and do

The Complexity of the Optimal Spanning Hypertree Problem

Marius Zimand *

University of Rochester

Computer Science Department

October 1, 1993

Abstract

The notion of h-hypertree was defined in [9] in order to improve the Bonferroni inequalities. We prove that the problem of finding the optimal spanning h-hypertree of a complete hypergraph is NP complete, for $h \geq 3$. Moreover, no polynomial time approximate algorithm even with poor approximation ratio seems to exist for this problem. The existence of such an algorithm for the minimum version achieving an approximation ratio of $poly(n)$ implies $P = NP$, and, for the maximum version, the existence of a quasipolynomial algorithm achieving the ratio $2^{log^{\epsilon} n}$ implies $NP \subseteq DTIME[n^{\log^{O(1)} n}]$.

1 Basic Notions

An h-uniform hypergraph H is a pair $H = (X, E)$, where X is a set of vertices and E is a family of h-sets, i.e. $E = (E_1, \ldots, E_m)$ with $|E_i| = h$ for all i and $h \geq 2$. The sets E_i are called the edges of E. The order of the hypergraph H is $|X|$. For a hypergraph H the vertex set and edge set are denoted by $V(H)$ and $E(H)$, respectively. The degree of a vertex $x \in X$, denoted by $d_H(x)$, is the number of edges of H containing it. The family of all h-subsets of X is denoted by $\binom{X}{h}$. The complete h-hypergraph of order n, denoted by K_n^h,

*Supported in part by the National Science Foundation under grant NSF-CCR-8957604. Some results are contained also in "Optimal Spanning Hypertrees", I. Tomescu and Marius Zimand, Discrete Applied Mathematics (to appear)