A Local Search Algorithm for the Constrained Max Cut Problem on Hypergraphs

Nasim Samei[1] **and Roberto Solis-Oba**[*2]

[1]**Department of Computer Science, Western University, London, Ontario**
[2]**Department of Computer Science, Western University, London, Ontario**

Corresponding author:
Roberto Solis-Oba[2]

Email address: solis@csd.uwo.ca

ABSTRACT

In the constrained max k-cut problem on hypergraphs, we are given a weighted hypergraph $H = (V, E)$, an integer k and a set c of constraints. The goal is to divide the set V of vertices into k disjoint partitions in such a way that the sum of the weights of the hyperedges having at least two endpoints in different partitions is maximized and the partitions satisfy all the constraints in c. In this paper we present a local search algorithm for the constrained max k-cut problem on hypergraphs and show that it has approximation ratio $1-\frac{1}{k}$ for a variety of constraints c, such as for the constraints defining the max Steiner k-cut problem, the max multiway cut problem and the max k-cut problem. We also show that our local search algorithm can be used on the max k-cut problem with given sizes of parts and on the capacitated max k-cut problem, and it has approximation ratio $1-\frac{|V_{max}|}{|V|}$, where $|V_{max}|$ is the cardinality of the biggest partition. In addition, we present a local search algorithm for the directed max k-cut problem that has approximation ratio $\frac{k-1}{3k-2}$.

1 INTRODUCTION

A weighted hypergraph $H = (V, E)$ consist of a set V of nodes, a set E of hyperedges and a function w that assigns a non-negative weight to every edge. A hyperedge e consist of a non-empty set of nodes (called its endpoints). Graphs are special cases of hypergraphs where each hyperedge has exactly two nodes. The size of a hyperedge e is the number of nodes in e and the rank of a hypergraph $H = (V, E)$ is the size of the hyperedge $e \in E$ with the biggest cardinality.

In the max k-cut problem on hypergraphs we are given a weighted hypergraph $H = (V, E)$ and an integer k, and the goal is to partition V into k non-empty sets in such a way that the sum of the weights of the hyperedges having at least two endpoints in different partitions is maximized.

In the related max multiway cut problem on hypergraphs, besides having a weighted hypergraph $H = (V, E)$ and integer k, we are also given a set $T = \{t_1, t_2, \ldots, t_k\} \subseteq V$ of terminals and the goal is to divide V into k partitions so as to maximize the sum of the weights of the hyperedges having at least two endpoints in different partitions and such that each partition has exactly one terminal. Some other related problems are max Steiner k-cut, max cut with given sizes of parts (Ageev and Sviridenko, 1999) and capacitated max k-cut (Gaur et al., 2008).

All above problems involve grouping the vertices of a weighted hypergraph $H = (V, E)$ into k non-empty partitions that satisfy some additional set c of constraints and the goal is to maximize the sum of the weights of the hyperedges having at least two endpoints in different partitions. We call this problem the constrained max k-cut problem on hypergraphs. For the aforementioned problems the sets c of constraints that their solutions need to satisfy are as follows.

*The research of the third author was partially supported by the Natural Sciences and Engineering Research Council of Canada, grant 04667-2015 RGPIN.

- Max k-cut: No additional constraints, just divide V into k disjoint non-empty partitions.
- Max multiway cut: Each partition must include one vertex from a given set $T = \{t_1, t_2, \ldots, t_k\} \subseteq V$ of terminals.
- Max Steiner k-cut: Each partition must include at least one vertex from a given set $T = \{t_1, t_2, \ldots, t_l\} \subseteq V$ of terminals, where $l \geq k$. Note that this is a generalization of the max multiway cut problem.
- Capacitated max k-cut problem: Given a set $\{s_1, s_2, \ldots, s_k\}$ of sizes, a valid partition $V_1, \ldots, V_k$ of V must satisfy $|V_i| \leq s_i$, for all $1 \leq i \leq k$.
- Max k-cut with given sizes of parts: Given a set $\{s_1, s_2, \ldots, s_k\}$ of sizes, a valid partition $V_1, V_2, \ldots, V_k$ of V must satisfy $|V_i| = s_i$, for all $1 \leq i \leq k$. This is a special case of the capacitated max k-cut problem.

In this paper we present a general local search algorithm for the constrained max k-cut problem on hypergraphs that finds approximate solutions for all aforementioned problems. Our local search algorithm starts with an arbitrary feasible solution for the problem that partitions V into k disjoint sets. The algorithm then tries to improve the current solution by either moving one node from its current partition to another partition or by swapping two nodes from different partitions.

Our algorithm can be modified so it can be used also on the directed max k-cut problem on hypergraphs. A directed hypergraph $H = (V, E)$ consist of a set V of nodes and a set E of directed hyperedges. A directed hyperedge is an ordered pair (t, h) formed by two disjoint sets of nodes: t (the tail set) and h (the head set).

Given a directed hypergraph $H = (V, E)$ and a partition $V_1, V_2, \ldots, V_k$ of V, the weight of the partition is the total weight of the hyperedges having at least one head in some partition i and at least one of their tails in some partition j, where $i > j$. In the directed max k-cut problem on hypergraphs, the goal is to find a maximum weight partition $V_1, V_2, \ldots, V_k$ of V.

The approximation ratio of our algorithm for max k-cut, max multiway cut and max Steiner k-cut is $1 - \frac{1}{k}$. For the max k-cut problem with given sizes of parts and the capacitated max cut problem our algorithm has approximation ratio $1 - \frac{|V_{max}|}{|V|}$, where $|V_{max}|$ is the size of the largest partition. The approximation ratio of our algorithm for the directed max k-cut problem on hypergraphs is $\frac{k-1}{3k-2}$.

Related Work: There has been a significant amount of research on max k-cut and related problems on graphs. Papadimitriou (1994) presented a local search algorithm for the unweighted max cut problem, a special case of the max k-cut problem when $k = 2$, and showed that the approximation ratio of his algorithm is $\frac{1}{2}$. This is a simple algorithm that starts with two arbitrary partitions and then repeatedly improves the solution by moving one node to the other partition. Goemans and Williamson (1995) introduced a randomized rounding approximation algorithm based on a semidefinite relaxation of the max cut problem with expected approximation ratio 0.8785. They later designed an algorithm for the max 3-cut problem with approximation ratio 0.8360 (Goemans and Williamson, 2004). An algorithm with the same approximation ratio was presented by de Klerk et al. (2004).

Vazirani (2001) designed a simple greedy $(1 - \frac{1}{k})$-approximation algorithm for the max k-cut problem. Frieze and Jerrum (1997) generalized the randomized approximation algorithm of Goemans and Williamson and designed a randomized algorithm for the max k-cut problem with expected approximation ratio $1 - \frac{1}{k} + 2\,\frac{\ln k}{k^2}$. Kann et al. (1997) show that no approximation algorithm for the max k-cut problem can have approximation ratio better than $1 - \frac{1}{34k}$ unless $P = NP$.

Frieze and Jerrum (1997) also designed a randomized algorithm for the max bisection problem, where we have to partition V into two sets of equal size, and showed that the approximation ratio of their algorithm is 0.65. Ye (2001) improved on this result by designing an algorithm with approximation ratio 0.699. Later, Halperin and Zwick (2002), Feige and Langberg (2006), Raghavendra and Tan (2012) designed algorithms with approximation ratios 0.7016, 0.7028 and 0.85 respectively, for the same problem. Finally, Austrin et al. (2013) improved the approximation ratio to 0.8776.

Currently the best known approximation algorithm for max k-Section (in this problem $|V_1| = |V_2| = \ldots = |V_k| = \frac{|V|}{k}$) is by Andersson (1999) with approximation ratio $1 - \frac{1}{k} + \Theta(\frac{1}{k^3})$ based on semidefinite programming, generalizing the algorithm in (Frieze and Jerrum, 1997) for the max bisection problem.

Liu et al. (2006) designed a greedy local search algorithm for the generalized max k-multiway cut problem with approximation ratio $1 - \frac{1}{k}$. In the generalized max k-multiway cut problem besides having a weighted graph $G = (V,E)$ and integer k, we are also given p disjoint subsets U_i of V of size k. The goal is to divide V into k partitions such that each partition includes exactly one node from U_i for all $1 \leq i \leq p$.

For the max cut problem with given sizes of parts Ageev and Sviridenko (1999) desiged a $\frac{1}{2}$-approximation algorithm using pipage rounding. Feige and Langberg (2001) designed a semi-definite programming-based algorithm with approximation ratio $\frac{1}{2} + \varepsilon$ for the same problem for any $\varepsilon > 0$. For the capacitated max k-cut problem Wu and Zhu (2014) modified the local search algorithm by Gaur et al. (2008) and show that the approximation ratio of their algorithm is $\frac{|V_{min}|(k-1)}{2(|V_{max}|-1)+|V_{min}|(k-1)}$, where $|V_{min}|$ and $|V_{max}|$ are sizes of the minimum and the maximum partitions returned by the algorithm. Our algorithm for the capacitated max k-cut problem has approximation ratio $1 - \frac{|V_{max}|}{|V|} \geq 1 - \frac{|V_{max}|}{|V_{max}|+|V_{min}|(k-1)}$. Therefore, our algorithm is better than the algorithm of Wu and Zhu when $|V_{max}| \geq 2$. Furthermore, our algorithm works on hypergraphs and not just on graphs.

For the directed max cut problem Goemans and Williamson (1995) designed a 0.796-approximation algorithm that uses a semidefinite programming based technique. Feige and Goemans (1995) used a similar technique and improved the ratio to 0.859. Also, a $\frac{1}{2}$-approximation algorithm for the max directed cut problem with given sizes of parts was designed by Ageev et al. (2001) based on pipage rounding.

For the max cut problem on hypergraphs Andersson and Engebretsen (1998) designed a 0.72-approximation algorithm. For the max k-cut problem on hypergraphs with given sizes of parts Ageev and Sviridenko (2000) designed an approximation algorithm based on pipage rounding with approximation ratio $1-(1-\frac{1}{r})^r - (\frac{1}{r})^r$, where r is the number of nodes in the smallest hyperedge. For the case when all the hyperedges have at least 3 nodes they gave a $(1-\frac{1}{e})$-approximation algorithm. If we compare our $(1-\frac{|V_{max}|}{|V|})$-approximation algorithim for the max k-cut problem with given sizes of parts on hypergraphs with that of Ageev and Sviridenko (2000), since $1-(1-\frac{1}{r})^r - (\frac{1}{r})^r \leq 0.7$ our algorithm has better approximation ratio when $|V_{max}| < \frac{3}{10}|V|$, where $|V_{max}|$ is the size of the biggest partition.

Zhu and Guo (2011) used local search to design a $\frac{k-1}{\Delta+k-1}$-approximation algorithm for the max k-cut problem on hypergraphs, where $\Delta = min\{\frac{s(s-1)}{2}, \frac{k(k-1)}{2}\}$ and s is the size of the largest hyperedge. They also gave a local search $(1-\frac{1}{k})$-approximation algorithm for the max k- cut problem on graphs. We note that our $(1-\frac{1}{k})$-approximation algorithm for hypergraphs has a much better approximation ratio than that of Zhu and Guo.

2 THE LOCAL SEARCH ALGORITHM

Given a hypergraph $H = (V,E)$, let $V_1, V_2, \ldots, V_k$ be an arbitrary partition of V into k non-empty sets. We denote a hyperedge e as $(u_1, u_2, \ldots, u_{r_e})$, where $u_1, u_2, \ldots, u_{r_e}$ are the endpoints of e. We define H_i to be the set of hyperedges whose endpoints are all in partition V_i and $H_i(u)$ to be the set of hyperedges from H_i incident on u:

$$H_i = \{(u_1,u_2,...,u_{r_e}) \mid u_1,u_2,...,u_{r_e} \in V_i, \, (u_1,u_2,...,u_{r_e}) \in E\}, \text{and} \tag{1}$$

$$H_i(u) = \{(u_1,u_2,...,u_{r_e}) \mid u_j = u \text{ for some } 1 \leq j \leq r, \, (u_1,u_2,...,u_{r_e}) \in H_i\}. \tag{2}$$

Let H_{ij} be the set of hyperedges that have one endpoint in V_i and all other endpoints in V_j, and let $H_{ij}(u)$ be the set of hyperedges from H_{ij} incident on u. Note that in general $H_{ij} \neq H_{ji}$. Our algorithm for the constrained max k-cut problem on hypergraphs is described below.

Algorithm Local Search(H,w,c)
Input: Hypergraph $H = (V,E)$, weight function $w: E \to Z^+$, constraints c.
Output: A partition of the set V satisfying c.

1. Start with an arbitrary partition, V_1, ..., V_k that satisfies the constraints c.

2. If there is a node $u \in V_i$ such that there is a partition V_l, $i \neq l$ for which

   $\sum_{e \in H_i(u)} w(e) > \sum_{e \in H_{il}(u)} w(e)$

   and moving u to V_l creates a partition that satisfies the constraints in c, then move u from V_i to V_l.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.27434v1 | CC BY 4.0 Open Access | rec: 18 Dec 2018, publ: 18 Dec 2018

3. If there are nodes $u \in V_i$ and $v \in V_l$, $i \neq l$ for which
   $\sum_{e \in H_i(u)} w(e) + \sum_{e \in H_l(v)} w(e) > \sum_{e \in H_{il}(u)} w(e) + \sum_{e \in H_{li}(v)} w(e)$
   and moving u to V_l and v to V_i creates a partition that satisfies the constraints in c, then move u to V_l and v to V_i.

4. If a node u as specified in Step 2 exists or if nodes u, v as specified in Step 3 exist then repeat Steps 2 and 3, otherwise output the partition $V_1, V_2, \ldots, V_k$.

Schaffer and Yannakakis (1991) proved that given a weighted graph, the problem of finding a partition of its vertices so the weight of the cut cannot be increased by moving a vertex from one side to the other (same operation as described in Step 2 of our algorithm) is polynomial time local search (PLS)-complete. The class PLS-complete introduced by Johnson et al. (1988) is formed by those problems for which a polynomial time local search algorithm for one implies such an algorithm for all of them. Therefore, it is unlikely that our local search algorithm has polynomial running time.

The running time of our local search algorithm is dominated by the time complexity of Step 2 and Step 3 and by the number of times that Step 2 and Step 3 are repeated. Step 2 can be easily implemented to run in $O(k|V|(|V||E| + f(c)))$ time, where the time needed to verify if a partition of V satisfies the constraints in c is $f(c)$, and Step 3 can be implemented to run in $O(|V|^2(|V||E| + f(c)))$ time. The number of iterations of Steps 2 and 3 is at most $\sum_{e \in E} w(e)$ since at each step of the algorithm the weight of the solution increases by at least one unit, but this is not polynomial in the size of the input. Using the result by Orlin et al. (2004) we can transform our algorithm into an ε-local search algorithm for any $\varepsilon > 0$ with approximation ratio $(1 - \varepsilon)$ times the approximation ratio of the local search algorithm. The running time of the ε- local search algorithm is $O(|V|^4(|V||E| + f(c)))$, which is polynomial for any constant value $\varepsilon > 0$ when $f(c)$ is polynomial. We note that $f(c)$ is polynomial for all problems mentioned above. In the sequel we will analyze the performance of the local search algorithm knowing that we can modify it to achieve polynomial running time at the expense of a small loss in the quality of the approximation ratio.

3 MAX K-CUT, MAX MULTIWAY CUT, AND MAX STEINER K-CUT PROBLEMS

In this section we analyze the local search algorithm described in the previous section and compute its approximation ratio for the max k-cut, the max multiway cut, and the max Steiner k-cut problems on hypergraphs.

Let $P = (V_1, V_2, ..., V_k)$ be the partition computed by the local search algorithm. We define E' as the set of hyperedges that have at least two endpoints in different partitions:

$$E' = \{(u_1, u_2, ..., u_{r_e}) \mid \text{partition containing } u_i \neq \text{partition containing } u_j,\ (u_1, u_2, ..., u_{r_e}) \in E\}. \quad (3)$$

Then the cost S of the local optimum solution computed by our algorithm is,

$$S = \sum_{e \in E'} w(e). \quad (4)$$

Note that the only hyperedges that do not contribute to S are those whose endpoints are all in the same partition. Since P is a local optimal solution, for any nodes $u \in V_i$ and $v \in V_l$, $V_l \neq V_i$, according to the conditions stated in Steps 2 and 3 of the local search algorithm either one or both of the following inequalities hold:

- $$\sum_{e \in H_i(u)} w(e) \leq \sum_{e \in H_{il}(u)} w(e). \quad (5)$$

  The above inequality holds if u can be moved to V_l while satisfying the set c of constraints.

- $$\sum_{e \in H_i(u)} w(e) + \sum_{e \in H_l(v)} w(e) \leq \sum_{e \in H_{il}(u)} w(e) + \sum_{e \in H_{li}(v)} w(e). \quad (6)$$

The above inequality holds if u and v can swap partitions while satisfying the set c of constraints.

To make the analysis of the algorithm uniform when applied to any one of the 3 problems considered in this section, for each partition V_i, $i = 1,2,\ldots,k$, we try to choose a node p_i so that inequality (6) holds for all pairs of nodes p_i, p_l, $i \neq l$: We choose (i) $p_i = t_i \in T$ for the max multiway cut problem, (ii) p_i does not exist for the max k-cut problem, and (iii) $p_i = t'_i$ for the max Steiner k-cut problem, where t'_i is a terminal from V_i. Note that inequality (5) holds for all nodes $V_i \setminus p_i$, $1 \leq i \leq k$, for all three problems.

Consider partitions $V_l \neq V_i$. If we add inequality (5) for all nodes in $V_i \setminus p_i$ we get,

$$\sum_{u \in V_i \setminus p_i} \sum_{e \in H_i(u)} w(e) \leq \sum_{u \in V_i \setminus p_i} \sum_{e \in H_{il}(u)} w(e). \tag{7}$$

Observe that in the term $\sum_{u \in V_i \setminus p_i} \sum_{e \in H_i(u)} w(e)$ the weight of each hyperedge $e \in H_i$ is counted r_e times, except the weight of the hyperedges e incident on the terminals p_i whose weights are counted $r_e - 1$ times. In addition, $\sum_{u \in V_i \setminus p_i} \sum_{e \in H_{il}(u)} w(e)$ includes the weight of all the hyperedges in H_{il} except those incident on terminal p_i. Since $r_e \geq 2$ for each hyperedge e, we can rewrite inequality (7) as follows,

$$2 \sum_{e \in H_i} w(e) - \sum_{e \in H_i(p_i)} w(e) \leq \sum_{e \in H_i} r_e\, w(e) - \sum_{e \in H_i(p_i)} w(e) \leq \sum_{e \in H_{il}} w(e) - \sum_{e \in H_{il}(p_i)} w(e). \tag{8}$$

Where $H_i(p_i)$ and $H_{il}(p_i)$ are empty if p_i does not exist. Adding the above inequality over all partitions $V_l \neq V_i$ we get,

$$2(k-1) \sum_{e \in H_i} w(e) - \sum_{\substack{1 \leq l \leq k \\ l \neq i}} \sum_{e \in H_i(p_i)} w(e) \leq \sum_{\substack{1 \leq l \leq k \\ l \neq i}} \sum_{e \in H_{il}} w(e) - \sum_{\substack{1 \leq l \leq k \\ l \neq i}} \sum_{e \in H_{il}(p_i)} w(e). \tag{9}$$

Adding this last inequality over all partitions V_i we get,

$$2(k-1) \sum_{1 \leq i \leq k} \sum_{e \in H_i} w(e) - \sum_{1 \leq i \leq k} \sum_{\substack{1 \leq l \leq k \\ l \neq i}} \sum_{e \in H_i(p_i)} w(e) \leq \sum_{1 \leq i \leq k} \sum_{\substack{1 \leq l \leq k \\ l \neq i}} \sum_{e \in H_{il}} w(e) - \sum_{1 \leq i \leq k} \sum_{\substack{1 \leq l \leq k \\ l \neq i}} \sum_{e \in H_{il}(p_i)} w(e). \tag{10}$$

Since (6) holds for all the nodes p_i then,

$$\sum_{e \in H_i(p_i)} w(e) + \sum_{e \in H_l(p_l)} w(e) \leq \sum_{e \in H_{il}(p_i)} w(e) + \sum_{e \in H_{li}(p_l)} w(e), \text{ for each } 1 \leq i \neq l \leq k. \tag{11}$$

We now add up this last inequality over all $i, l = 1, .., k$, $i \neq l$, to get

$$\sum_{1 \leq i \leq k} \sum_{\substack{1 \leq l \leq k \\ l \neq i}} \left(\sum_{e \in H_i(p_i)} w(e) + \sum_{e \in H_l(p_l)} w(e) \right) \leq \sum_{1 \leq i \leq k} \sum_{\substack{1 \leq l \leq k \\ l \neq i}} \left(\sum_{e \in H_{il}(p_i)} w(e) + \sum_{e \in H_{li}(p_l)} w(e) \right). \tag{12}$$

We can rewrite the above inequality as follows,

$$2 \sum_{1 \leq i \leq k} \sum_{\substack{1 \leq l \leq k \\ l \neq i}} \sum_{e \in H_i(p_i)} w(e) \leq 2 \sum_{1 \leq i \leq k} \sum_{\substack{1 \leq l \leq k \\ l \neq i}} \sum_{e \in H_{il}(p_i)} w(e). \tag{13}$$

Dividing the above inequality by 2 and adding it to (10), we get

$$2(k-1) \sum_{1 \leq i \leq k} \sum_{e \in H_i} w(e) \leq \sum_{1 \leq i \leq k} \sum_{\substack{1 \leq l \leq k \\ l \neq i}} \sum_{e \in H_{il}} w(e). \tag{14}$$

Since $\sum_{1 \leq i \leq k} \sum_{\substack{1 \leq l \leq k \\ l \neq i}} \sum_{e \in H_{il}} w(e) \leq 2S$, then by (14)

$$\sum_{1 \leq i \leq k} \sum_{e \in H_i} w(e) \leq \frac{1}{k-1} S. \tag{15}$$

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.27434v1 | CC BY 4.0 Open Access | rec: 18 Dec 2018, publ: 18 Dec 2018

Since an optimum solution can at most include the weights of all the edges, the cost O of an optimum solution can be bounded by

$$O \leq S + \sum_{1 \leq i \leq k} \sum_{e \in H_i} w(e) \leq \left(1 + \frac{1}{k-1}\right) S. \tag{16}$$

Therefore,

$$\frac{S}{O} \geq 1 - \frac{1}{k}. \tag{17}$$

THEOREM 1. *There is a $(1-\frac{1}{k})$-approximation algorithm for the max k-cut, max multiway cut, and max Steiner k-cut problems on hypergraphs.*

4 MAX CAPACITATED K-CUT PROBLEM AND MAX K-CUT PROBLEM WITH GIVEN SIZES OF PARTS

In this section we analyse our local search algorithm for the max capacitated k-cut problem and the max k-cut problem with given sizes of parts and show that its approximation ratio is $1 - \frac{|V_{max}|}{|V|}$, where $|V_{max}|$ is the size of the biggest partition returned by the algorithm.

We proceed similarly as in Section 3. Since $P = (V_1, V_2, \ldots, V_k)$ is a local optimal solution, for any nodes $u \in V_i$ and $v \in V_l$, $V_l \neq V_i$, either one or both of inequalities (5) and (6) must hold. Observe that in the max k-cut problem with given sizes of parts only swaps are allowed, therefore only inequality (6) is true for all the nodes. On the other hand, in the capacitated max k-cut problem the condition in Step 2 of the algorithm is true for a node $u \in V_i$ only if there is a partition $V_l \neq V_i$ of size $|V_l| < s_l$ and such that $\sum_{e \in H_i(u)} w(e) > \sum_{e \in H_{il}(u)} w(e)$. Since swaps are allowed for all pairs of nodes in the capacitated max k-cut problem inequality (6) is true for all of them; hence in the analysis we will only use this inequality.

Adding inequality (6) for all $u \in V_i$ we get,

$$\sum_{e \in H_i} r_e w(e) + |V_i| \sum_{e \in H_l(v)} w(e) \leq \sum_{e \in H_{il}} w(e) + |V_i| \sum_{e \in H_{li}(v)} w(e). \tag{18}$$

Notice that the first term in the left side of this inequality is $\sum_{e \in H_i} r_e w(e)$ because each hyperedge e in H_i is counted exactly r_e times in $\sum_{u \in V_i} \sum_{e \in H_i(u)} w(e)$ and the first term in the right side of the inequality is $\sum_{e \in H_{il}} w(e)$ since each hyperedge in H_{il} is counted exactly one time in $\sum_{u \in V_i} \sum_{e \in H_{il}(u)} w(e)$. Next, we sum inequality (18) for all $v \in V_l$ to get

$$|V_l| \sum_{e \in H_i} r_e w(e) + |V_i| \sum_{e \in H_l} r_e w(e) \leq |V_l| \sum_{e \in H_{il}} w(e) + |V_i| \sum_{e \in H_{li}} w(e). \tag{19}$$

Since $r_e \geq 2$ for each hyperedge then,

$$2|V_l| \sum_{e \in H_i} w(e) + 2|V_i| \sum_{e \in H_l} w(e) \leq |V_l| \sum_{e \in H_i} r_e w(e) + |V_i| \sum_{e \in H_l} r_e w(e) \leq |V_l| \sum_{e \in H_{il}} w(e) + |V_i| \sum_{e \in H_{li}} w(e). \tag{20}$$

We sum this inequality for all $i, l = 1, 2, \ldots, k$, $i \neq l$:

$$\sum_{1 \leq i \leq k} \sum_{\substack{1 \leq l \leq k \\ i \neq l}} 2(|V_l| \sum_{e \in H_i} w(e) + |V_i| \sum_{e \in H_l} w(e)) \leq \sum_{1 \leq i \leq k} \sum_{\substack{1 \leq l \leq k \\ i \neq l}} (|V_l| \sum_{e \in H_{il}} w(e) + |V_i| \sum_{e \in H_{li}} w(e)). \tag{21}$$

The left side of the above inequality can be simplified as follows,

$$\begin{aligned} \sum_{1 \leq i \leq k} \sum_{\substack{1 \leq l \leq k \\ i \neq l}} 2(|V_l| \sum_{e \in H_i} w(e) + |V_i| \sum_{e \in H_l} w(e)) &= 2 \sum_{1 \leq i \leq k} \sum_{e \in H_i} w(e) \sum_{\substack{1 \leq l \leq k \\ i \neq l}} |V_l| + \\ 2 \sum_{1 \leq l \leq k} \sum_{e \in H_l} w(e) \sum_{\substack{1 \leq i \leq k \\ i \neq l}} |V_i| &= 2 \sum_{1 \leq i \leq k} (|V| - |V_i|) \sum_{e \in H_i} w(e) + \\ 2 \sum_{1 \leq l \leq k} (|V| - |V_l|) \sum_{e \in H_l} w(e) &= 4 \sum_{1 \leq i \leq k} (|V| - |V_i|) \sum_{e \in H_i} w(e). \end{aligned}$$

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.27434v1 | CC BY 4.0 Open Access | rec: 18 Dec 2018, publ: 18 Dec 2018

Similarly, the right side of inequality (21) can be simplified as follows,

$$\sum_{1\le i\le k}\sum_{\substack{1\le l\le k\\ i\ne l}}(|V_l|\sum_{e\in H_{il}}w(e)+|V_i|\sum_{e\in H_{li}}w(e))=\sum_{1\le i\le k}\sum_{\substack{1\le l\le k\\ i\ne l}}|V_l|\sum_{e\in H_{il}}w(e)+ \sum_{1\le l\le k}\sum_{\substack{1\le i\le k\\ i\ne l}}|V_i|\sum_{e\in H_{li}}w(e)=2\sum_{1\le i\le k}\sum_{\substack{1\le l\le k\\ i\ne l}}|V_i|\sum_{e\in H_{il}}w(e).$$

Therefore, we can re-write inequality (21) as follows,

$$2\sum_{1\le i\le k}(|V|-|V_i|)\sum_{e\in H_i}w(e)\le\sum_{1\le i\le k}\sum_{\substack{1\le l\le k\\ i\ne l}}|V_i|\sum_{e\in H_{il}}w(e). \tag{22}$$

Let $|V_{max}|=max\{|V_i|,i=1,2,\dots,k\}$, then

$$2(|V|-|V_{max}|)\sum_{1\le i\le k}\sum_{e\in H_i}w(e)\le|V_{max}|\sum_{1\le i\le k}\sum_{\substack{1\le l\le k\\ i\ne l}}\sum_{e\in H_{il}}w(e)\le 2|V_{max}|\,S \tag{23}$$

Therefore,

$$\sum_{1\le i\le k}\sum_{e\in H_i}w(e)\le\frac{|V_{max}|}{|V|-|V_{max}|}S. \tag{24}$$

Since,

$$O\le S+\sum_{1\le i\le k}\sum_{e\in H_i}w(e)\le(1+\frac{|V_{max}|}{|V|-|V_{max}|})S, \tag{25}$$

then,

$$\frac{S}{O}\ge\frac{1}{1+\frac{|V_{max}|}{|V|-|V_{max}|}}=1-\frac{|V_{max}|}{|V|} \tag{26}$$

THEOREM 2. *There is a $(1-\frac{|V_{max}|}{|V|})$-approximation algorithm for the max capacitated k-cut problem and max k-cut problem with given sizes of parts on hypergraphs.*

Corollary 1. *There is a $\frac{1-|V_{max}|}{2|V|-|V_{max}|}$-approximation algorithm for the max capacitated k-cut problem and the max k-cut problem with given sizes of parts restricted to hypergraphs where every hyperedge has at least 3 endpoints.*

Proof. Note that if every hyperedge has at least three endpoints then inequality (23) becomes $2(|V|-|V_{max}|)\sum_{1\le i\le k}\sum_{e\in H_i}\le|V_{max}|S$ and thus in this case $\frac{S}{O}\ge 1-\frac{|V_{max}|}{2|V|-|V_{max}|}$. □

5 DIRECTED MAX K-CUT PROBLEM

A directed hypergraph $H=(V,E)$ consist of set V of nodes and set E of hyperedges. Each hyperedge $e=(u_1,u_2,\dots,u_{r_e})\in E$ has a set t_e of tails and, a set h_e of heads and a weight $w(e)$. We call a hyperedges e, a *B-arc* if e has only one head h_e and a *F-arc* if e has only one tail t_e. A *BF-hypergraph* is a directed hypergraph in which all the hyperedges are B-arcs or F-arcs. In this section we deal with BF-hypergraphs, so in the sequel hypergraph means BF-hypergraph.

Given a directed hypergraph $H=(V,E)$ and a partition $V_1,V_2,\dots,V_k$ of V, the weight of the partition is the total weight of the hyperedges having at least one head in some partition i and at least one of their tails in some partition j, where $i>j$. In the directed max k-cut problem on hypergraphs, the goal is to find a maximum weight partition $P=V_1,V_2,\dots,V_k$ of V.

In Figure 1 a hypergraph $H=(V,E)$ with 8 vertices and 5 hyperedges is shown. The sets of tails and heads for each hyperedge are as follows, $t_{e_1}=\{v_1\}$, $h_{e_1}=\{v_2\}$, $t_{e_2}=\{v_4\}$, $h_{e_2}=\{v_2,v_3\}$, $t_{e_3}=\{v_1\}$, $h_{e_3}=\{v_5\}$, $t_{e_4}=\{v_4\}$, $h_{e_4}=\{v_6,v_7,v_8\}$, $t_{e_5}=\{v_5\}$ and $h_{e_5}=\{v_4,v_8\}$. Let 3, 4, 1, 5, 1 be the weights

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.27434v1 | CC BY 4.0 Open Access | rec: 18 Dec 2018, publ: 18 Dec 2018

of hyperedges e_1, e_2, e_3, e_4 and e_5 respectively. Consider partition $P = V_1, V_2, V_3$. The weight of this partition is 1+5+1=7.

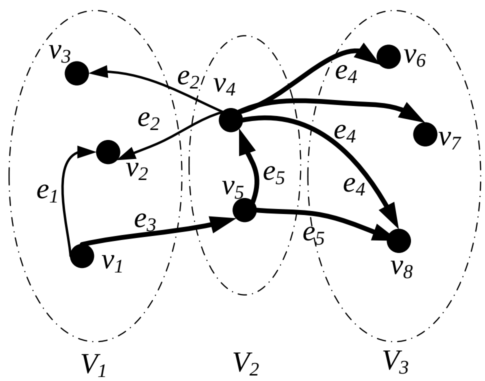

Figure 1. Example of a directed Hypergraph.

Given a hypergraph $H = (V,E)$, and a partition $P = V_1, V_2, \ldots, V_k$ of V we define sets H_i, $H_i(u)$, $T_i(u)$, as follows,

$$H_i = \{(u_1, u_2, \ldots, u_{r_e}) \mid u_1, u_2, \ldots, u_{r_e} \in V_i,\ (u_1, u_2, \ldots, u_{r_e}) \in E\},$$

$$H_i(u) = \{e = (u_1, u_2, \ldots, u_{r_e}) \mid (u_1, u_2, \ldots, u_{r_e}) \in H_i,\ u \in h_e\},$$

$$T_i(u) = \{e = (u_1, u_2, \ldots, u_{r_e}) \mid (u_1, u_2, \ldots, u_{r_e}) \in H_i,\ u \in t_e\}.$$

We define additional sets of hyperedges T_{ij} and H_{ij} as follows.

- T_{ij}, $i < j$, is a set of B-arcs and F-arcs that contribute to the weight of the partition P such that if we move one of the tails of any of these hyperedges from V_i to V_j then that hyperedge will no longer contribute to the weight of the partition. The hyperedges of T_{ij} have the following properties:

  (i) each B-arc e in T_{ij} has exactly one tail in V_i and every other tail in $\bigcup_{j \le q \le k} V_q$, and its head is in V_j,

  (ii) each F-arc e in T_{ij} has its tail in V_i, at least one head in V_j and no head in $\left(\bigcup_{j < q \le k} V_q\right)$.

  Let $T_{ij}(u)$, $u \in V_i$, be the set of hyperedges e from T_{ij} for which $u \in t_e$.

- H_{ij}, $i > j$, is a set of B-arcs and F-arcs that contribute to the weight of partition P such that if we move one of the heads of any of these hyperedges from partition V_i to partition V_j then that hyperedge will no longer contribute to the weight of P. The hyperedges of H_{ij} have the following properties:

  (i) each B-arc e in H_{ij} has its head in V_i, no tail in $\bigcup_{1 \le q < j} V_q$, and at least one tail in V_j,

  (ii) each F-arc e in H_{ij} has exactly one head in V_i and all other heads in $\bigcup_{1 \le q \le j} V_q$, and its tail in V_j.

  Let $H_{ij}(u)$, $u \in V_i$, be the set of hyperedges e from H_{ij}, where $u \in h_e$.

Our algorithm for the directed max k-cut problem is described below.

Algorithm Max DICUT (H, w)
Input: Directed hypergraph $H = (V,E)$, hyperedge weight function $w : E \to Z^+$.
Output: A partition of the set V.

1. Start with an arbitrary partition, V_1, ..., V_k, where $V_i \neq \emptyset$ for $i = 1, 2, \ldots, k$.
2. If there is a node $u \in V_i$ and a partition V_l, $i < l$, such that

$$\sum_{e \in H_i(u)} w(e) > \sum_{i<j\leq l} \sum_{e \in T_{ij}(u)} w(e),$$

then move u from V_i to V_l.
3. If there is a node $u \in V_i$ and a partition V_l, $i > l$, such that

$$\sum_{e \in T_i(u)} w(e) > \sum_{l \leq j < i} \sum_{e \in H_{ij}(u)} w(e),$$

then move u from V_i to V_l.
4. If a node u as specified in Step 2 or Step 3 exists then repeat Step 2 and Step 3. Otherwise, compare the cost of the solution induced by the ordered partition $P = V_1, V_2, \ldots, V_k$ and the cost of the solution induced by the reverse partition $P_r = V_k, V_{k-1}, \ldots, V_1$ and return the partition with the bigger cost.

Using the local search property specified in Step 2 of the algorithm, for each node $u \in V_i,\ i,l \in \{1, 2, \ldots, k\}$ and $i < l$ we have,

$$\sum_{e \in H_i(u)} w(e) \leq \sum_{e \in T_{ij}(u)} w(e). \tag{27}$$

Adding up inequality (27) for all nodes in V_i we get,

$$\sum_{u \in V_i} \sum_{e \in H_i(u)} w(e) \leq \sum_{u \in V_i} \sum_{i<j\leq l} \sum_{e \in T_{ij}(u)} w(e). \tag{28}$$

Observe that each hyperedge e in the term $\sum_{u \in V_i} \sum_{e \in H_i(u)} w(e)$ is counted $|h_e|$, times therefore $\sum_{e \in H_i} w(e) \leq \sum_{e \in H_i} |h_e| w(e) = \sum_{u \in V_i} \sum_{e \in H_i(u)} w(e)$. In the term $\sum_{u \in V_i} \sum_{i<j\leq l} \sum_{e \in T_{ij}(u)} w(e)$ each hyperedge e is counted once because in this expression e is counted only when $u \in t_e \cap V_i$ and from the definition of $T_{ij}(u)$ we know that u must be a tail of e, at least one head of e must be in V_j and no head of e can be in V_q for $j < q \leq k$. Therefore, inequality (28) can be simplified as follows,

$$\sum_{e \in H_i} w(e) \leq \sum_{i<j\leq l} \sum_{e \in T_{ij}} w(e). \tag{29}$$

Adding (29) over all $1 \leq i < l \leq k$, we get

$$\sum_{1\leq i\leq k} \sum_{i<l\leq k} \sum_{e \in H_i} w(e) \leq \sum_{1\leq i\leq k} \sum_{i<l\leq k} \sum_{i<j\leq l} \sum_{e \in T_{ij}} w(e). \tag{30}$$

Similarly, using the local search property specified in Step 3 of the algorithm, for each node $u \in V_i,\ i,l \in \{1, 2, \ldots, k\}$ and $l < i$, we have,

$$\sum_{e \in T_i(u)} w(e) \leq \sum_{l\leq j<i} \sum_{e \in H_{ij}(u)} w(e). \tag{31}$$

Adding up inequality (31) for all nodes in V_i we get,

$$\sum_{u \in V_i} \sum_{e \in T_i(u)} w(e) \leq \sum_{u \in V_i} \sum_{l\leq j<i} \sum_{e \in H_{ij}(u)} w(e). \tag{32}$$

Observe that by a similar argument as above $\sum_{e \in T_i} w(e) \leq \sum_{u \in V_i} \sum_{e \in T_i(u)} w(e)$. Also, in the term $\sum_{u \in V_i} \sum_{l\leq j<i} \sum_{e \in H_{ij}(u)} w(e)$ in the right side of (32) each hyperedge e is counted once. To see this consider the following two cases: If e is a B-arc then e has its head in V_i, at least one tail in V_j and no tail in $\bigcup_{1\leq q<j} V_q$; hence, in the right side of (32) e is counted only once when j is the smallest index of a partition containing a tail of e. If e is an F-arc then it has exactly one head in V_i, its tail in V_j and all other heads in $\cup_{1\leq q\leq j} V_q$; therefore, in the right side of (32) e is only counted once when j is the index of the partition containing the tail of e.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.27434v1 | CC BY 4.0 Open Access | rec: 18 Dec 2018, publ: 18 Dec 2018

Therefore, inequality (32) can be simplified as follows,

$$\sum_{e\in H_i} w(e) \leq \sum_{l\leq j<i} \sum_{e\in H_{ij}} w(e). \tag{33}$$

Adding inequality (33) over all $1 \leq l < i \leq k$, we get

$$\sum_{1\leq i\leq k} \sum_{1\leq l<i} \sum_{e\in H_i} w(e) \leq \sum_{1\leq i\leq k} \sum_{1\leq l<i} \sum_{l\leq j<i} \sum_{e\in H_{ij}} w(e). \tag{34}$$

Now we add inequalities (30) and (34):

$$\sum_{1\leq i\leq k} \sum_{\substack{1\leq l\leq k\\ i\neq l}} \sum_{e\in H_i} w(e) \leq \sum_{1\leq i\leq k} \sum_{i<l\leq k} \sum_{i<j\leq l} \sum_{e\in T_{ij}} w(e) + \sum_{1\leq i\leq k} \sum_{1\leq l<i} \sum_{l\leq j<i} \sum_{e\in H_{ij}} w(e). \tag{35}$$

Each term $\sum_{e\in T_{ij}} w(e)$ is counted $k-j+1$ times in $\sum_{1\leq i\leq k} \sum_{i<l\leq k} \sum_{i<j\leq l} \sum_{e\in T_{ij}} w(e)$ because for each pair i, j, $i<j$, the value of l must be such that $j \leq l$ and $l \leq k$; since there are $k-j+1$ such values, the term $\sum_{e\in T_{ij}}$ appears $k-j+1$ times. Similarly, the term $\sum_{e\in H_{ij}} w(e)$, $1 \leq j < i \leq k$, is counted j times in $\sum_{1\leq i\leq k} \sum_{1\leq l<i} \sum_{l\leq j<i} \sum_{e\in H_{ij}} w(e)$, because for each pair i, j, $i<j$, the value of l must be such that $l \geq 1$ and $l \leq j$; since there are j such values, the term $\sum_{e\in H_{ij}} w(e)$ appears j times. Therefore, we can rewrite the right hand side of (35) as follows,

$$\begin{aligned} &\sum_{1\leq i\leq k} \sum_{i<l\leq k} \sum_{i<j\leq l} \sum_{e\in T_{ij}} w(e) + \sum_{1\leq i\leq k} \sum_{1\leq l<i} \sum_{l\leq j<i} \sum_{e\in H_{ij}} w(e) = \\ &\sum_{1\leq i\leq k} \sum_{i<j\leq k} (k-j+1) \sum_{e\in T_{ij}} w(e) + \sum_{1\leq i\leq k} \sum_{1\leq j<i} j \sum_{e\in H_{ij}} w(e). \end{aligned} \tag{36}$$

Observe that in the term $\sum_{1\leq i\leq k} \sum_{1\leq j<i} j \sum_{e\in H_{ij}} w(e)$ if we replace i with j and j with i then we get,

$$\sum_{1\leq i\leq k} \sum_{1\leq j<i} j \sum_{e\in H_{ij}} w(e) = \sum_{1\leq j\leq k} \sum_{1\leq i<j} i \sum_{e\in H_{ji}} w(e). \tag{37}$$

Note that in the term $\sum_{1\leq j\leq k} \sum_{1\leq i<j} i \sum_{e\in H_{ji}} w(e)$, i can get values from 1 to $k-1$ and j can get values from $i+1$ to k, therefore,

$$\sum_{1\leq j\leq k} \sum_{1\leq i<j} i \sum_{e\in H_{ji}} w(e) = \sum_{1\leq i<k} \sum_{i<j\leq k} i \sum_{e\in H_{ji}} w(e) = \sum_{1\leq i\leq k} \sum_{i<j\leq k} i \sum_{e\in H_{ji}} w(e). \tag{38}$$

The second equality in (38) is true since, if $i=k$ there is no value j such that $i<j\leq k$. Let $E_{ij} = T_{ij} \cup H_{ji}$, for each $i<j$. Using (37) and (38) in the right hand side of (36) we get,

$$\begin{aligned} &\sum_{1\leq i\leq k} \sum_{i<j\leq k} (k-j+1) \sum_{e\in T_{ij}} w(e) + \sum_{1\leq i\leq k} \sum_{1\leq j<i} j \sum_{e\in H_{ij}} w(e) = \\ &\sum_{1\leq i\leq k} \sum_{i<j\leq k} (k-j+1) \sum_{e\in T_{ij}} w(e) + \sum_{1\leq i\leq k} \sum_{i<j\leq k} i \sum_{e\in H_{ji}} w(e) \leq \\ &\sum_{1\leq i\leq k} \sum_{i<j\leq k} (k-j+1) \sum_{e\in E_{ij}} w(e) + \sum_{1\leq i\leq k} \sum_{i<j\leq k} i \sum_{e\in E_{ij}} w(e) = \\ &\sum_{1\leq i\leq k} \sum_{i<j\leq k} (k+i-j+1) \sum_{e\in E_{ij}} w(e) \leq k \sum_{1\leq i\leq k} \sum_{i<j\leq k} \sum_{e\in E_{ij}} w(e). \end{aligned} \tag{39}$$

The last inequality holds because $i<j$. Now we show that all sets E_{ij}, for all $1 \leq i < j \leq k$, are disjoint. Suppose that there are sets E_{ij} and E_{lq}, $E_{ij} \neq E_{lq}$, $1 \leq i < j \leq k$ and $1 \leq l < q \leq k$, such that $E_{ij} \cap E_{lq} \neq \emptyset$.

- Let E_{ij} and E_{lq} share a B-arc e. Recall that by the definition of B-arcs, e has one head. Without loss of generality assume $l<i$. Since $E_{ij} = T_{ij} \cup H_{ji}$, by the definition of T_{ij} and H_{ji} if $e \in E_{ij}$ then e has its head in V_j, at least one tail in V_i, and no tails in $\bigcup_{1\leq t<i} V_t$ (observe that if $e \in T_{ij}$ then e has exactly one tail in V_i and all other tails are in $\bigcup_{j\leq t\leq k} V_t$, and since $i<j$ then there is no tail in $\bigcup_{1\leq t<i} V_t$). Similarly if $e \in E_{lq}$, then e should have its head in V_q, and since e has only one head then it must be that $V_j = V_q$; furthermore e has at least one tail in V_l, however since $l<i$ this contradicts the fact that e has no tails in $\bigcup_{1\leq t<i} V_t$.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.27434v1 | CC BY 4.0 Open Access | rec: 18 Dec 2018, publ: 18 Dec 2018

- Now suppose that E_{ij} and E_{lq} share a F-arc e. Recall that F-arcs have only one tail. Without loss of generality assume that $j < q$. Since $E_{ij} = T_{ij} \cup H_{ji}$, by the definition of T_{ij} and H_{ji} if $e \in E_{ij}$ then it has its tail in V_i, at least one head in V_j and no head in $\bigcup_{j<t\leq k} V_t$. Similarly if $e \in E_{lq}$, e has its tail in V_l and since e has only one tail then $V_i = V_l$; moreover e has at least one head in V_q, however since $j < q$ and e cannot have any heads in $\bigcup_{j<t\leq k} V_t$ this is a contradiction.

Let A_{ij}, $i < j$ be the set of hyperedges that have at least one tail in V_i and at least one head in V_j; note that $E_{ij} \subseteq A_{ij}$ for each $i < j$. The weight of the local optimal partition P is the weight of all the hyperedges in $\bigcup_{1\leq i<j\leq k} A_{ij}$, and since $E_{ij} \subseteq A_{ij}$ then $\bigcup_{1\leq i<j\leq k} E_{ij} \subseteq \bigcup_{1\leq i<j\leq k} A_{ij}$. Given a set C of hyperedges, let $w(C)$ denote the weight of the hyperedges of C. Then $w(\bigcup_{1\leq i<j\leq k} E_{ij}) \leq w(\bigcup_{1\leq i<j\leq k} A_{ij}) = S$, where S is the weight of the local optimal solution. Since all the sets E_{ij}, $1 \leq i,j \leq k$, are disjoint then $w(\bigcup_{1\leq i<j\leq k} E_{ij}) = \sum_{1\leq i\leq k} \sum_{i<j\leq k} \sum_{e\in E_{ij}} w(e)$, and so the right side of the last inequality in (39) can be bounded as follows,

$$k \sum_{1\leq i\leq k} \sum_{i<j\leq k} \sum_{e\in E_{ij}} w(e) \leq kS. \tag{40}$$

We can simplify the left side of inequality (35):

$$\sum_{1\leq i\leq k} \sum_{\substack{1\leq l\leq k \\ i\neq l}} \sum_{e\in H_i} w(e) = (k-1) \sum_{1\leq i\leq k} \sum_{e\in H_i} w(e) \tag{41}$$

Therefore, by inequalities (35), (36), (39), (40) and (41) we have,

$$(k-1) \sum_{1\leq i\leq k} \sum_{e\in H_i} w(e) \leq kS, \text{ or } \sum_{1\leq i\leq k} \sum_{e\in H_i} w(e) \leq \frac{k}{k-1} S. \tag{42}$$

Let B be the set of hyperedges in $E - S_L - \bigcup_{1\leq i\leq k} \bigcup_{e\in H_i} e$, where S_L is the set of hyperedges that contribute to the weight of the local optimal solution. Let S_r be the set of hyperedges that contribute to the weight of the reverse partition $P_r = V_k, V_{k-1}, \ldots, V_1$ as described in Step 4 of the algorithm. Note that because of the last step of the algorithm, $S \geq w(S_r)$, and since $w(B) \leq w(S_r)$ then $w(B) \leq S$. Let O be the weight of an optimal solution. Adding $w(B) + S$ to left side of inequality (42) and $2S$ to the right side we get,

$$O \leq w(B) + S + \sum_{1\leq i\leq k} \sum_{e\in H_i} w(e) \leq 2S + \frac{k}{k-1} S. \tag{43}$$

Therefore,

$$\frac{k-1}{3k-2} \leq \frac{S}{O}. \tag{44}$$

THEOREM 3. *There is a $\frac{k-1}{3k-2}$-approximation algorithm for the directed max k-cut problem on hypergraphs.*

REFERENCES

Ageev, A. A., Hassin, R., and Sviridenko, M. I. (2001). A 0.5-approximation algorithm for max dicut with given sizes of parts. *SIAM Journal on Discrete Mathematics*, 14(2):246–255.

Ageev, A. A. and Sviridenko, M. I. (1999). Approximation algorithms for maximum coverage and max cut with given sizes of parts. *Integer Programming and Combinatorial Optimization*, 1610:17–30.

Ageev, A. A. and Sviridenko, M. I. (2000). An approximation algorithm for hypergraph max k-cut with given sizes of parts. *European Symposium on Algorithms*, 1879:32–41.

Andersson, G. (1999). An approximation algorithm for max p-section. *Annual Symposium on Theoretical Aspects of Computer Science*, 1563:237–247.

Andersson, G. and Engebretsen, L. (1998). Better approximation algorithms for set splitting and not-all-equal sat. *Information Processing Letters*, 65(6):305–311.

Austrin, P., Benabbas, S., and Georgiou, K. (2013). Better balance by being biased: A 0.8776- approximation for max bisection. *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 277–294.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.27434v1 | CC BY 4.0 Open Access | rec: 18 Dec 2018, publ: 18 Dec 2018

de Klerk, E., Pasechnik, D., and Warners, J. (2004). On approximate graph colouring and max k-cut algorithms based on the θ-function. *Journal of Combinatorial Optimization*, 8(3):267–294.

Feige, U. and Goemans, M. (1995). Approximating the value of two prover proof systems, with applications to max 2sat and max dicut. *Proceedings of the Third Israel Symposium on Theory of Computing and Systems*, pages 182–189.

Feige, U. and Langberg, M. (2001). Approximation algorithms for maximization problems arising in graph partitioning. *Journal of Algorithms*, 41(2):174–211.

Feige, U. and Langberg, M. (2006). The rpr2 rounding technique for semidefinite programs. *Journal of Algorithms*, 60(1):1–23.

Frieze, A. and Jerrum, M. (1997). Improved approximation algorithms for max-k-cut and max bisection. *Algorithmica*, 18(1):67–81.

Gaur, D. R., Krishnamurti, R., and Kohli, R. (2008). The capacitated max k-cut problem. *Mathematical Programming*, 115(1):65–72.

Goemans, M. X. and Williamson, D. (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145.

Goemans, M. X. and Williamson, D. P. (2004). Approximation algorithms for max-3-cut and other problems via complex semidefinite programming. *Journal of Computer and System Sciences*, 68(2):442–470.

Halperin, E. and Zwick, U. (2002). A unified framework for obtaining improved approximation algorithms for maximum graph bisection problems. *Random Structures Algorithms*, 20(3):382–402.

Johnson, D. S., Papadimitriou, C. H., and Yannakakis., M. (1988). How easy is local search? *Journal of Computer and System Sciences*, 37:79–100.

Kann, V., Khanna, S., Lagergren, J., and Panconesi, A. (1997). On the hardness of approximating max k-cut and its dual. *Chicago Journal of Theoretical Computer Science*, 2:1–18.

Liu, J., Peng, Y., and Zhao, C. (2006). Generalized k-multiway cut problems. *Journal of Applied Mathematics and Computing*, 21:69–82.

Orlin, J. B., Punnen, A. P., and Schulz, A. S. (2004). Approximate local search in combinatorial optimization. *SIAM Journal on Computing*, 33(5):1201–1214.

Papadimitriou, C. H. (1994). Computational complexity. Addison-Wesley.

Raghavendra, P. and Tan, N. (2012). Approximating csps with global cardinality constraints using sdp hierarchies. *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete algorithms*, pages 373–387.

Schaffer, A. A. and Yannakakis, M. (1991). Simple local search problems that are hard to solve. *SIAM Journal on Computing*, 20(1):56–87.

Vazirani, V. (2001). Approximation algorithms. Springer.

Wu, J. and Zhu, W. (2014). On a local search algorithm for the capacitated max-k-cut problem. *Kuwait Journal of Science*, 41(3):129–137.

Ye, Y. (2001). A 699-approximation algorithm for max-bisection. *Mathematical Programming*, 90(1):101–111.

Zhu, W. and Guo, C. (2011). A local search approximation algorithm for max-k-cut of graph and hypergraph. *Fourth International Symposium on Parallel Architectures Algorithms and Programming*, pages 236–240.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.27434v1 | CC BY 4.0 Open Access | rec: 18 Dec 2018, publ: 18 Dec 2018