Math. Program., Ser. A
DOI 10.1007/s10107-007-0139-z

FULL LENGTH PAPER

The capacitated max k-cut problem

Daya Ram Gaur · Ramesh Krishnamurti ·
Rajeev Kohli

Received: 25 July 2005 / Accepted: 12 April 2007

Abstract We consider a capacitated max k-cut problem in which a set of vertices is partitioned into k subsets. Each edge has a non-negative weight, and each subset has a possibly different capacity that imposes an upper bound on its size. The objective is to find a partition that maximizes the sum of edge weights across all pairs of vertices that lie in different subsets. We describe a local-search algorithm that obtains a solution with value no smaller than $1 - 1/k$ of the optimal solution value. This improves a previous bound of $1/2$ for the max k-cut problem with fixed, though possibly different, sizes of subsets.

1 Introduction

Consider a graph with non-negative weights associated with the edges. The capacitated max k-cut problem concerns the partitioning of vertices into k subsets, each of which has a capacity that specifies the maximum number of vertices it can contain. The capacities can vary from one subset to another, but their sum must be no smaller than the total number of vertices in the graph. The objective of the problem is to partition

We thank an anonymous referee for extensive and constructive comments. The first and second authors are grateful for the support provided by the Natural Sciences and Engineering Research Council of Canada.

D. R. Gaur
University of Lethbridge, Lethbridge, Canada
e-mail: gaur@cs.uleth.ca

R. Krishnamurti
Simon Fraser University, Burnaby, Canada
e-mail: ramesh@cs.sfu.ca

R. Kohli (✉)
Columbia University, Uris Hall, 3022 Broadway, New York, NY 10027, USA
e-mail: rk35@columbia.edu

the set of vertices into subsets such that the sum of edge weights across all pairs of vertices that lie in different subsets is maximized.

The proposed algorithm begins with an arbitrary partitioning of the vertices, with each subset size no greater than its capacity. It searches in a neighborhood comprising all pairs of vertices that lie in different subsets and evaluates the value of a switch by comparing the weighted sum of edge weights with and without flipping a pair of vertices. The algorithm stops when no further improvements can be attained in this value across all possible pairs of flips. We refer to this procedure as "local search" in the rest of the paper. Similar local-search procedures have been considered by Papadimitriou and Stieglitz [18] for the maxcut problem. The proposed local-search procedure terminates in a finite number of steps. However, it is not clear if it terminates in polynomial time.

In general, local-search algorithms that run in polynomial time have proved elusive for combinatorially-hard problems. Johnson et al. [14] introduce polynomial-time local search (PLS), a complexity class containing problems that are equally hard in the sense that if a polynomial-time locally optimal solution can be found for one problem, then such a solution can be found for all problems in the class. Problems in this class are called PLS-complete. Among these is the problem of finding a locally optimal solution for the weighted version of the maxcut problem, obtained by setting $k = 2$ and removing the capacity constraints in the capacitated max k-cut problem. Thus, not only is capacitated max k-cut NP-Hard, it also appears unlikely that it admits a polynomial-time local-search procedure. The complexity of finding a local optimum is an important open problem [14]. Orlin et al. [17] provide a fully polynomial-time locally optimal approximation scheme for every combinatorial optimization problem for which the neighbourhood is "efficiently searchable." In particular, they provide such a locally optimal approximation scheme for every problem in PLS. We use this result to show that the proposed local-search algorithm can be used to obtain a solution within $(1 - 1/k)(1 - \epsilon)$ of the optimal within time polynomial in the input instance size and $1/\epsilon$.

Related problems. Ageev, Hassin and Sviridenko [2] give a $1/2$ approximation algorithm for the capacitated max 2-cut problem. Ageev and Sviridenko [1] consider a generalization of capacitated max k-cut for hypergraphs. They give a $1-(1-\frac{1}{r})^r-(\frac{1}{r})^r$-factor approximation for general weighted hypergraphs where r is the cardinality of the smallest edge, and a $(1 - 1/e)$-factor approximation for the case when each hyperedge has at least three vertices. This implies an approximation ratio of $1/2$ for the max k-cut problem with fixed, possibly different, subset sizes. Both results use a pipage-rounding technique.

For equal capacities, a $1 - 1/k + \Omega(1/k^3)$ performance guarantee is obtained by using an algorithm described by Andersson [3]. Feige and Langberg [9] allow unequal capacities when $k = 2$. They use semi-definite programming to construct an approximation algorithm with a lower bound of $1/2 + \epsilon$, where $\epsilon > 0$ is a universal constant. Karloff [15] considers the problem of partitioning the vertices of a graph into two sets such that at least half the edges from each vertex go across the cut. He shows that a simple local search procedure produces such a partition.[1]

[1] We thank an anonymous reviewer for drawing our attention to this result.

Methods for solving the max k-cut problem without capacity constraints include an algorithm by Frieze and Jerrum [10] that extends Goemans and Williamson's [12] randomized-rounding procedure using semi-definite programming. The algorithm obtains solutions with expected value no smaller than $\alpha_k \sim (2 \log k/)k^2$. Frieze and Jerrum's algorithm gives a 0.65 approximation for max bisection. Ye [19] describes an algorithm for the same problem with a performance guarantee of 0.699. Goemans and Williamson [13] use an extension of semi-definite programming to complex space and device an approximation algorithm for the max 3-cut problem. They obtain a performance guarantee of $\frac{7}{12} + \frac{3}{4\pi^2} \arccos^2(-1/4) \approx 0.83601$; the same lower bound is also obtained by de Klerk et al. [8]. Kann et al. [16] show that unless P=NP, the best possible performance ratio attainable by a heuristic for the max k-cut problem is $1 - 1/(34k)$. This result also applies to the more general capacitated version of the problem we consider in this paper. We obtain a lower bound of $1 - 1/k$ for the proposed local-search procedure for solving the capacitated max k-cut problem. This generalizes the lower bound of $1 - 1/k$ for local-search for the special case where the capacities are equal [11].

Applications. The capacitated max k-cut problem has several applications. We describe below the four problems that motivated our consideration of the problem.

The first application concerns the placement of television commercials in program breaks [7]. The objective is to obtain a commercial schedule so that competing products are not advertised in the same break. A special case of the capacitated max k-cut problem is attained by representing each commercial by a vertex, an edge connecting a pair of vertices representing commercials for competing products. The capacity constraints for the subsets of vertices correspond to the number of commercials that can be shown in each of the program breaks. A key factor in obtaining the max k-cut representation is that commercials are typically 30 seconds long and most breaks last 2 or 3 minutes (and so have a capacity for four or six commercials). Observe that all edge weights are equal in this problem; allowing unequal edge weights corresponds to allowing differences in the extent to which pairs of commercials are considered to be in conflict with each other.

The second application considers the placement of containers on a ship with k bays. Each bay has a capacity for a certain number of containers. The ship uploads and downloads containers at each of a sequence of ports [4–6]. Let the vertices of the graph represent containers. Let an edge connect two vertices if placing the associated containers in the same bay requires moving one or both containers during uploading or downloading. The solution to the capacitated max k-cut problem identifies a stowage plan that assigns the maximum number of conflicting containers to different bays.

The objective in the third application is to partition a set of items (e.g., books sold by Amazon.com) into k subsets. Items that are least often purchased by the same customer (or items that are least often purchased together) are placed into different subsets. The max k-cut is obtained by associating the vertices of a graph with items and assigning to an edge a weight equal to the number of customers who buy at most one (but not both) of the pair of items identified by the vertices of an edge. The capacity constraints arise because one wishes to place a small number of similar items—for example, four or five books—into each subset, so that the most promising items are offered for cross-purchase to a customer.

The fourth application concerns the design of product modules. Each vertex represents a component of a product. An edge connects a pair of vertices if the corresponding pair of components do not interact (e.g., are not connected) with each other. The problem is to find a partitioning of components into k modules so that the maximum number of non-interacting components are placed in different modules. The capacity restriction arises because of the need to construct "balanced" modules.

2 A local-search algorithm

Let $G(V, E)$ denote a graph with $|V| = n$ vertices. Let edge $e \in E$ connect vertices $u, v \in V$. We associate a weight $w(u, v)$ with each edge. We consider the problem of partitioning the vertices into k subsets, $V_1, \ldots, V_k$, where the i-th subset V_i contains at most s_i vertices, and where $|V| \leq s_1 + \cdots + s_k$. The objective is to find a partition of the vertices so that the sum of the edge weights across all pairs of subsets in the partition is as large as possible. We call this the capacitated max k-cut problem. Observe that equal edge weights corresponds to counting the number of edges that connect vertices in different subsets in the partition.

Let

$$w_{uV_i} = \sum_{v \in V_i,\ v \neq u} w(u, v)$$

denote the sum of the weights of the edges from a vertex u to the vertices in set V_i. We examine the performance of the following local-search algorithm.

Initialization. Partition the vertices into k sets, $V_1, \ldots, V_k$, arbitrarily assigning $|V_i| \leq s_i$ vertices to set V_i, for all $i = 1, \ldots, k$.

Iterative step. Determine if there is a pair of vertices $u \in V_i$ and $v \in V_l$, $i \neq l$, for which

$$|V_i| w_{uV_i} + |V_l| w_{vV_l} > |V_i| w_{uV_l} + |V_l| w_{vV_i}.$$

If such a pair of vertices exist, reassign vertex u to V_l, and vertex v to V_i.

Termination. Stop when

$$|V_i| w_{uV_i} + |V_l| w_{vV_l} \leq |V_i| w_{uV_l} + |V_l| w_{vV_i}, \quad \text{for all } u \in V_i \text{ and } v \in V_l.$$

If each set in the partition has the same size $|V_1| = \cdots = |V_k|$, then the condition in the iterative step becomes

$$w_{uV_i} + w_{vV_l} > w_{uV_l} + w_{vV_i}.$$

However, when the sets in the partitions have different sizes, we assign different weights to switches into and out of subsets, as is seen by writing the condition in the iterative step as

$$|V_i|(w_{uV_i} - w_{uV_l}) > |V_l|(w_{vV_i} - w_{vV_l}).$$

The effect of this, as will become evident below, is to eliminate the influence of the subset sizes on the performance of the algorithm.

If all edge weights are equal (i.e., if the objective is to maximize the number of edges across all pairs of subsets), then the above local-search procedure runs in time $O(n^3m)$, where n is the number of vertices and m is the number of edges in the graph. However, the problem of obtaining such a locally optimal solution for the general weighted version is PLS-complete.

3 Worst-case analysis

The following theorem characterizes the worst-case bound of the local-search algorithm for all $k \geq 2$.

Theorem 1 *The solution obtained using the local-search algorithm has a value no smaller than* $1 - \frac{1}{k}$ *of the optimal solution value.*

Proof Let OPT denote the value of the optimal solution. Let $V_1, \ldots, V_k$ be an arbitrary solution obtained using the local-search algorithm. Let ALG denote the value of the solution $V_1, \ldots, V_k$.

Let w_{ii} denote the sum of the weights of the edges with both the vertices in V_i. Let w_{il} denote the sum of the weights of the edges that connect a vertex in V_i to a vertex in V_l. Let

$$W_s = \sum_{i=1}^{k} w_{ii} \quad \text{and}$$

$$\text{ALG} = \sum_{i=1}^{k} \sum_{l=i+1}^{k} w_{il}.$$

Recall that w_{uV_i} denotes the sum of the edge weights from a vertex u onto set V_i. As the solution returned by the local-search algorithm is locally optimal,

$$|V_i|w_{uV_i} + |V_l|w_{vV_l} \leq |V_i|w_{uV_l} + |V_l|w_{vV_i}, \quad \text{for all } \ u \in V_i \text{ and } v \in V_l.$$

We sum both sides of the above inequality over all $u \in V_i$ and obtain

$$|V_i| \sum_{u \in V_i} w_{uV_i} + |V_i||V_l|w_{vV_l} \leq |V_i| \sum_{u \in V_i} w_{uV_l} + |V_i||V_l|w_{vV_i}.$$

Next, we sum both sides of the above expression over all $v \in V_l$ and get

$$|V_l||V_i| \sum_{u \in V_i} w_{uV_i} + |V_i||V_l| \sum_{v \in V_l} w_{vV_l} \leq |V_l||V_i| \sum_{u \in V_i} w_{uV_l} + |V_i||V_l| \sum_{v \in V_l} w_{vV_i}.$$

Setting

$$\sum_{u \in V_i} w_{uV_i} = 2w_{ii}, \quad \sum_{v \in V_l} w_{vV_l} = 2w_{ll}, \quad \sum_{u \in V_i} w_{uV_l} = w_{il}, \quad \sum_{v \in V_l} w_{vV_i} = w_{li},$$

gives

$$2|V_i||V_l|(w_{ii} + w_{ll}) \leq |V_i||V_l|(w_{il} + w_{li}).$$

Cancelling the common term $|V_i||V_l|$ from both sides, we get

$$w_{ii} + w_{ll} \leq w_{il}.$$

Summing both sides of the above inequality over all $i, l = 1, \ldots, k,\ i \neq l$, gives

$$\sum_{i=1}^{k} \sum_{l=1, l \neq i}^{k} (w_{ii} + w_{ll}) \leq \sum_{i=1}^{k} \sum_{l=1, l \neq i}^{k} w_{il}.$$

We simplify the right-hand side by setting $w_{il} = w_{li}$ and obtain

$$\sum_{i=1}^{k} \sum_{l=1, l \neq i}^{k} w_{il} = 2\text{ALG}.$$

Thus,

$$\sum_{i=1}^{k} \sum_{l=1, l \neq i}^{k} w_{ii} + \sum_{i=1}^{k} \sum_{l=1, l \neq i}^{k} w_{ll} \leq 2\text{ALG}.$$

The above inequality can be written as

$$(k-1) \sum_{i=1}^{k} w_{ii} + (k-1) \sum_{i=1}^{k} w_{ii} \leq 2\text{ALG},$$

or

$$2(k-1) \sum_{i=1}^{k} w_{ii} \leq 2\text{ALG}.$$

It follows that

$$W_s = \sum_{i=1}^{k} w_{ii} \leq \frac{\text{ALG}}{k-1}.$$

The optimal solution can contain at most all of the edges. Therefore

$$\text{OPT} \leq W_s + \text{ALG} \leq \text{ALG} + \frac{\text{ALG}}{k-1},$$

or equivalently

$$\frac{\text{ALG}}{\text{OPT}} \geq 1 - \frac{1}{k}.$$

□

We note in closing that the neighbourhood specified by the present local-search algorithm is efficiently searchable. In other words, it is possible to verify that a solution is locally optimal, and also to improve a locally non-optimal solution, in polynomial time. Theorem 1, together with the result of Orlin et al. [17], implies that a solution within $(1 - 1/k)(1 - \epsilon)$ of the optimal solution can be obtained in time that is a polynomial in the input size and $1/\epsilon$.

References

1. Ageev, A.A., Sviridenko, M.I.: An approximation algorithm for hypergraph max k-cut with given sizes of parts. Lecture Notes in Computer Science (Proceedings of ESA '00), vol. 1879, pp. 32–41 (2000)
2. Ageev, A., Hassin, R., Sviridenko, M.: A 0.5-approximation algorithm for max dicut with given sizes of parts. SIAM J. Discret. Math. **14**(2), 246–255 (2001)
3. Andersson, G.: An approximation algorithm for max p-section. Lecture Notes in Computer Science (Proceedings of STACS'99), vol. 1563, pp. 237–247 (1999)
4. Aslidis, A.: Minimizing of overstowage in container ship operations. Oper. Res. **90**, 457–471 (1990)
5. Avriel, M., Penn, M.: Exact and approximate solutions of the container ship stowage problem. Comput. Ind. Eng. **25**, 271–274 (1993)
6. Avriel, M., Penn, M., Shpirer, N., Witteboon, S.: Stowage planning for container ships to reduce the number of shifts. Ann. Oper. Res. **76**, 55–71 (1997)
7. Bollapragada, S., Garbiras, M.: Scheduling commercials on broadcast television. Oper. Res. **52**(3), 337–345 (2004)
8. de Klerk, E., Pasechnik, D.V., Warners, J.P.: On approximate graph colouring and MAX k-CUT algorithms based on the θ-function. J. Comb. Optim. **8**(3), 267–294 (2004)
9. Feige, U., Langberg, M.: Approximation algorithms for maximization problems arising in graph partitioning. J. Algorithms **41**, 174–211 (2001)
10. Frieze, A., Jerrum, M.: Improved approximation algorithms for max k-cut and max bisection. Algorithmica **18**(1), 67–81 (1997)
11. Gaur, D., Krishnamurti, R.: The capacitated max k-cut problem. In: Proceedings of the International Conference on Computational Science and its Applications (2005), LNCS 3483, pp. 670–679
12. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. Assoc. Comput. Mach. **42**, 1115–1145 (1995)
13. Goemans, M.X., Williamson, D.P.: Approximation algorithms for MAX 3-CUT and other problems via complex semidefinite programming. J. Comput. Syst. Sci. **68**(2), 442–470 (2004)

14. Johnson, D.S., Papadimitriou, C.H., Yannakakis, M.: How easy is local search? J. Comput. Syst. Sci. **37**, 79–100 (1988)
15. Karloff, H.J.: Fast parallel algorithms for graph-theoretic problems: matching, coloring and partitioning. Ph.D. Thesis, UC Berkeley (1985)
16. Kann, V., Khanna, S., Lagergren, J., Panconesi, A.: On the hardness of approximating max k-cut and its dual. Chicago J. Theor. Comput. Sci. **2**, 1–18 (1997)
17. Orlin, J.B., Punnen, A.P., Schulz, A.S.: Approximate local search in combinatorial optimization. SIAM J. Comput. **33**(5), 1201–1214 (2004)
18. Papadimitriou, C.H., Steiglitz, K.: Combinatorial optimization: algorithms and complexity. Prentice-Hall, Englewood Cliffs (1982)
19. Ye, Y.: A .699-approximation algorithm for max-bisection. Math. Program. (A) **90**, 101–111 (2001)