Approximate Pure Equilibria in Cut Games

Arash Asadpour
Management Science and Engineering Department, Stanford University,
Stanford, CA 94305, USA
asadpour@stanford.edu,

Tomás Feder
268 Waverley St., Palo Alto, CA 94301, USA
tomas@theory.stanford.edu,

Amin Saberi
Management Science and Engineering Department, Stanford University,
Stanford, CA 94305, USA
saberi@stanford.edu

April 24, 2009

1 Introduction

The problem of finding the maximum cut in an undirected weighted graph G has been studied vastly in the computer science community. The seminal work of Goemans and Williamson [2] provide a 0.879 approximation for this problem, while Khot, et. al prove the optimality of that algorithm under the assumption of unique games conjecture and P$\neq$NP [4].

One can easily define a local search algorithm for the Max-Cut problem. In the local search, each vertex changes its side if the total weight of its adjacent edges in the cut increase. It is straightforward to see that this local search will eventually find a local optimum. Finding such a local optimum is PLS-complete [3, 8].

This local search however, defines a game called *cut games* [1]. These games are motivated by the *party affiliation game*: there are n players who have two actions $\{-1, 1\}$ available. The weights $w_{i,j}$ are known which are attributed to the heat of relationship between enemies, i.e. those pairs who strongly prefer to stay in different sides have higher $w_{i,j}$. The payoff of a person i is $\text{sgn}(\sum_j s_i.s_j.w_{i,j})$.

In this paper we study the following different variations of cut games. In the general setting, a *directed* weighted graph $G = (V, E)$ is given. Each vertex is controlled by a selfish player. Each player can adopt one of the k strategies $\{1, 2, \cdots, k\}$ for some fixed $k \geq 2$. In other words, we are interested in the pure strategies of this game. We represent the set of players who adopted strategy s_i by P_i. Also, let $P_{-s_i} = V - P_{s_i}$. Different variations of cut games can be defined by assuming different payoff functions.

- **Symmetric Cut Game.** For all pairs i, j, we have $w_{i,j} = w_{j,i} \geq 0$. The payoff function for a player i who adopts strategy s_i is $\sum_{j \in P_{-s_i}} w_{i,j}$.
- **The Asymmetric Cut Game.** The weight $w_{i,j}$ can be different from $w_{j,i}$. The payoff function is similar to that in symmetric cut games.
- **Love and Hate Game.** When there are love and hate relationships, i.e. when there are two types of edges, L and H and the payoff function for a vertex i is $\sum_{\substack{j \in P_{-s_i}\\(i,j)\in H}} w_{i,j} + \sum_{\substack{j \in P_{s_i}\\(i,j)\in L}} w_{i,j}$.

The *list* version of the above games is defined where each vertex i is only allowed to lie in one of the r_i parts listed for her a priori. Clearly, the case $r_i = k$ for all i is equivalent to the original game.

In each of the above games, an α-approximate solution is an assignment of vertices to the k parts so that the payoff of each vertex i is at least $\alpha \sum_j w_{i,j}$. Note that the payoff of a vertex i cannot be more than $\sum_j w_{i,j}$. Hence, the approximate solutions are in correspondence with approximate pure Nash equilibria of the game.

2 Symmetric Cut Games

We first consider the well-known local search for the max cut in Theorem 2.1. Although it ensures a good approximation factor at any local optimum, the convergence time might be exponentially large. We resolve this issue by achieving different weaker approximation bounds in Theorems 2.3, 2.5, and 2.7.

Theorem 2.1 *Suppose G has n vertices and the largest edge weight in G is W. Then the local search for the list symmetric cut game achieves $\alpha = 1 - 1/r$ for each $2 \leq r \leq k$ in time polynomial in n and W. In fact, we can simultaneously achieve $\alpha_i = 1 - 1/r_i$ for each vertex v_i if each v_i has list of size $r_i \geq 1$.*

Proof. Assing the vertices of G to parts P_j from their list arbitrarily. Then repeatedly select vertices v_i so that moving v_i from P_j to a different part $P_{j'}$ in the list of v_i increases the weight coming out of v_i to a part different from the part of v_i. As the weight out of a vertex v_i is at most $(n-1)W$, this moving of vertices v_i from a part P_j to a part $P_{j'}$ will happen at most $n(n-1)W$ times, so the algorithm runs in time polynomial in n and W. Once the moving is over, if a vertex v_i is in part P_j and has weight of edges coming out v_i and staying in P_j adding up to U_i, then the edges coming out of v_i to a part $P_{j'}$ other than P_j in the list of v_i have weights adding to at least U_i, for a total over all such $P_{j'}$ of $(r_i - 1)U_i$. Thus $\alpha \geq (r_i - 1)/r_i = 1 - 1/r_i$ is achieved. □

Corollary 2.2 *Suppose G has n vertices and the largest edge weight in G is W. Then the local search for symmetric cut games achieves $\alpha = 1 - 1/k$ for each $k \geq 2$ in time polynomial in n and W.*

The difficulty with the local search is that W may be exponential in n. In the next sections, we shall achieve weaker α in time polynomial in n. However, some previous results ensure polynomial upperbound on the running time of local search [5]. Here, we take a different approach towards sparse instances.

Theorem 2.3 *Suppose G has n vertices and vertex v_i has degree d_i. Then the local search in list symmetric cut games achieves $\alpha = 1/d_i$ in time polynomial in n, for any $2 \leq r \leq k$.*

Proof. Select for each vertex v_i an edge of maximum weight coming out of v_i. The selected edges must form a forest because a cycle would have outgoing edges of successively higher weight along the cycle, which is not possible when we traverse the cycle back to a starting vertex. We may then examine each tree in the forest, assign a part P_j to a chosen root v_i for the root, and then go down the tree so that if some vertex $v_{i'}$ is assigned part $P_{j'}$ then each child $v_{i''}$ of $v_{i'}$ is assigned a part $P_{j''}$ different from $P_{j'}$. We always can do that because $r \geq 2$. Thus for each vertex v_i we will have that at least $\alpha = 1/d_i$ is achieved, as the largest weight goes to a different part. □

Corollary 2.4 *Suppose G has n vertices and vertex v_i has degree d_i. Then the local search for the symmetric cut game achieves $\alpha = 1/d_i$ for v_i in time polynomial in n, for any $k \geq 2$.*

2.1 Achieving $\alpha = 1/(\lceil \log m/\log(k-1)\rceil + 1)$ for $k \geq 3$ by Randomization

Theorem 2.5 *Suppose G has n vertices and m edges, and $k \geq 3$. Then the local search for the symmetric cut game achieves $\alpha = 1/(\lceil \log m/\log(k-1)\rceil + 1)$ in randomized time polynomial in n.*

Proof. As in the previous section, find the corresponding forest of maximum weights and proceed to assign parts starting at roots of trees so that children vertices $v_{i''}$ get a different part from the parent $v_{i'}$. We will always have at least $k-1$ choices of parts for the child. If we order the edges out of a vertex by increasing weight, then if a vertex has an edge e coming out but staying in the same part, then the probability that the successive $t = \lceil \log m/\log(k-1)\rceil$ edges in the ordering will also stay in the same part is at most $(k-1)^{-t} \leq 1/m$, so with probability at least $(m-1)/m = 1-1/m$ this will not happen. Thus the edges out a vertex can be partitioned into groups of size at most $t+1$ whose largest weight goes to a different part, giving $\alpha = 1/(t+1) = 1/(\lceil \log m/\log(k-1)\rceil + 1)$ as required. □

Corollary 2.6 *Suppose G has n vertices and m edges, and $3 \leq r \leq k$. Then the local search for list symmetric cut game achieves $\alpha = 1/(\lceil \log m/\log(r-1)\rceil + 1)$ in randomized time polynomial in n.*

Proof. Follows as in the preceeding theorem, as each vertex in a tree of the forest of largest weights has at least $r-1$ choices. □

2.2 Achieving $\alpha = 1 - 1/\lfloor k/2\rfloor - O(1/n^c)$ for $k \geq 4$ Deterministically

Theorem 2.7 *Suppose G has n vertices, and $k \geq 4$. Then the local search for the symmetric cut game achieves $\alpha = 1 - 1/\lfloor k/2\rfloor - O(1/n^c)$ in deterministic time polynomial in n, for any constant $c \geq 1$.*

Proof. Let $N = n^{c+1}$, let W be the largest edge weight, and let W_i be the largest edge weight coming out of v_i. Let S be the set of vertices v_i such that $W/N^{2f} \geq W_i > W/N^{2f+1}$ for some integer $f \geq 0$. Let T be the set of vertices v_i such that $W/N^{2f+1} \geq W_i \geq W/N^{2f+2}$ for some integer $f \geq 0$. This partitions the vertices v_i into the two sets S and T.

We shall assign the vertices in S to k' parts and the vertices in T to different k' parts, for $k' = \lfloor k/2\rfloor$. This subdivides the problem into two subproblems, one for the parts for S and one to the parts for T. This two subproblems are similar, so we only show how to solve the subproblem for the k' parts for S. Consider each $f \geq 0$ in turn, and solve the problem for vertices v_i having $W/N^{2f} \geq W_i > W/N^{2f+1}$ by ignoring all weights at most W/N^{2f+2} and replacing each weight u by $\lfloor u/(W/N^{2f+3}\rfloor$, as weights shall be polynomial in N and thus in n, by the algorithm in section 2. This achieves $\alpha = 1-1/k' - O(1/N)$ by the rounding and the bound in section 2 for a given f, and the ignored weights are at most nW/N^{2f+2} which gives an additional $O(n/N)$ error with respect to the largest weight, which is at least W/N^{2f+1}. We thus achieve $\alpha = 1 - 1/k' - O(n/N) = 1 - 1/\lfloor k/2\rfloor - O(1/n^c)$, as desired. □

Corollary 2.8 *Suppose G has n vertices, and $k \geq 4(k-r+1)$. Then the local search for list symmetric cut game achieves $\alpha = 1 - 1/\lfloor k/2(k-r+1)\rfloor - O(1/n^c)$ in deterministic time polynomial in n, for any constant $c \geq 1$.*

Proof. Divide the k parts into $k' = \lfloor k/(k-r+1)\rfloor$ groups of size at least $k-r+1$. Each group intersects all subsets of r parts and will be considered as a single part. Obtain $\alpha = 1 - 1/\lfloor k'/2\rfloor - O(1/n^c)$ from the preceding theorem. □

2.3 NP-Completeness of $\alpha > 1/2$ with $k = 2$, Unit Weights, Degrees 2 and 3

Consider the bipartite symmetric cut game (i.e. $k = 2$) with unit edge weights, and vertices of degrees 2 and 3. We know we can achieve $\alpha = 1/2$. The question of whether we can achieve $\alpha > 1/2$ is NP-complete, even for this very special case of the problem.

Theorem 2.9 *The question of whether we can achieve $\alpha > 1/2$ for bipartite symmetric cut game is NP-complete, even with unit edge weights and maximum degree 3.*

Proof. Consider the not-all-equal satisfiability problem with at most three occurences per variable, which is NP-complete [7]. An instance of this problem has variables $x_1, x_2, \ldots, x_n$, and a collection of triples of variables (x_i, x_j, x_k) so that each variable occurs in at most three triples. The aim is to assign to each variable value 0 or 1 so that no triple gets $(0, 0, 0)$ or $(1, 1, 1)$

Represent each triple by a triangle (x_i, x_j, x_k), where all triangles are made disjoint by making multiple copies of the at most three occurences of each variable. If x_i occurs in three triples as x_i^1, x_i^2, x_i^3, introduce auxiliary variables y_i^1, y_i^2, y_i^3, z_i, and edges $x_i^1y_i^1$, $x_i^2y_i^2$, $x_i^3y_i^3$, $y_i^1z_i$, $y_i^2z_i$, $y_i^3z_i$, so that the copies are joined by a small tree.

Suppose the two sides of the partition with $k = 2$ are named 0 and 1. For the vertices y_i^j, of degree 2, we must have $x_i^j \neq y_j^i \neq z_i$ since otherwise $\alpha \leq 1/2$. For the triangles (x_i, x_j, x_k) we must not get $(0, 0, 0)$ or $(1, 1, 1)$ since otherwise $\alpha \leq 1/3$. So a solution with $\alpha > 1/2$ solves the not-all-equal satisfiability problem. Conversely, a solution to the not-all equal satisfiability problem achieves $\alpha = 1$ at the y_i^j and z_i, while it also achieves if say (x_i, x_j, x_k) is $(0, 1, 1)$ the fraction $\alpha = 1$ at x_i and the fraction $\alpha = 2/3$ at x_j, x_k because of the edges joining each x_i^j to a y_i^j. This proves we achieve $\alpha = 2/3$ when the not-all-equal satisfiability problem is solved, proving NP-completeness. □

3 Asymmetric Cut Games

In this section we consider the case where the underlying graph G is directed (i.e. $w_{i,j}$ and $w_{j,i}$ can be different), and each vertex wishes to achieve a fraction α of the sum of the weights of its outgoing edges.

For this problem, we shall only be interested in whether any $\alpha > 0$ can be achieved for an instance. A *strong component* in G is a maximal induced subgraph H such that there is a directed path from u to v for any pair of vertices u, v in H. The graph G can be decomposed into strong components H_i by partitioning the vertices. A *sink strong component* is a strong component that has no edges coming out of it. A sink strong component H_i is *good* if H_i contains only one vertex or H_i contains a cycle of even length.

Theorem 3.1 *One can achieve some $\alpha > 0$ for the asymmetric cut game with $k = 2$ if and only if every sink strong component in the instance G is good. This property can be decided in polynomial time.*

Proof. Suppose every sink strong component is good. Then we may choose an even length cycle in each sink strong component with at least two vertices, and choose for the remaining vertices in-trees that end in the single vertex or even length cycles of sink strong components. These in-trees ending in even cycles form a bipartite graph, so the bipartition of this bipartite graph can be used to define the two parts, with $\alpha > 0$.

Suppose some sink strong component H_i is not good, but we have a solution achieving $\alpha > 0$ for H_i. Selecting an edge going out of each vertex in H_i to the other part will give in-trees for H_i ending in an even cycle, contrary to the assumption that H_i is not good.

The property of whether a directed graph has an even length directed cycle can be decided in polynomial time [6], so it suffices to test all sink strong components with more than one vertex for this property in polynomial time. □

For larger k, the results follow as from before. Although, some of the results need a slightly different lowerbound for k.

Theorem 3.2 *Suppose G has n vertices and vertex v_i has out-degree d_i. Then $\alpha = 1/d_i$ for each vertex i can be achieved for list asymmetric cut game in time polynomial in n, for any $3 \leq r \leq k$.*

Proof. If we select the out-edge of largest degree for each vertex, we obtain in-forests ending in directed cycles, whose vertices can be 3-colored and thus put in 3 parts. $\square$

Theorem 3.3 *Suppose G has n vertices and m edges, and $4 \leq r \leq k$. Then $\alpha = 1/(\lceil \log m/\log(r-2) \rceil + 1)$ can be achieved for list asymmetric cut game in randomized time polynomial in n.*

Proof. As two neighbors are previously chosen, we have now $r-2$ choices instead of $r-1$ before. $\square$

4 Love and Hate Games

Consider the cut game with $k = 2$. Suppose that instead of considering that we benefit the weight of an edge only if the two endpoints are in different parts, we benefit from an edge if the two endpoints are in the same part for some edges and if they are in different parts for some other edges. More generally, for arbitrary k, we shall assume that each edge e has an associated *incompatibility* graph G_e on k vertices corresponding to the k parts, possibly with loops, and we benefit w_e from edge e only if its two ends lie in two *non*-adjacent parts in G_e. In this case we say these two parts are "compatible" for edge e. Any two parts whose corresponding vertices in G_e are adjacent, are called "incompatible" for edge e.

We emphasize that this is a generalization of love and hate games. To see that, it suffice to define G_e as the union of k loops for all hate edges e, and define it to be K_k for love edges. We shall assume that each part in G_e is adjacent with at most D parts in G_e, so that each part has at most D parts. Note that in love and hate games $D = k - 1$. In symmetric cut games however, no love edge exists and hence $D = 1$. Moreover, to have a well-defined notion of approximation, we assume that every vertex in any G_e is neither isolated nor connected to all vertices (including itself). Since, in the former case we could never benefit from e and in the latter we always do. Hence, we assume that always $0 < D < k$. The results obtained before generalize as follows. We omit most of the proofs.

Theorem 4.1 *Suppose G has n vertices and the largest edge weight in G is W. Then the local search achieves $\alpha = 1 - D/r$ in list love and hate game for each $D < r \leq k$ in time polynomial in n and W. In fact, it simultaneously achieve $\alpha_i = 1 - D/r_i$ for each vertex v_i if each v_i has list of size $r_i \geq D$.*

Proof. During local moves an end vertex of some edge e can benefit from this edge if it goes to any of $r - D$ parts that are not incompatible for e, so if the edges incident to a vertex v have total weight W, then the average of the benefit we would get by moving to each part is at least $(r-D)W/r \geq (1-D/r)W$, giving $\alpha = 1 - D/r$ for some part. $\square$

Theorem 4.2 *Suppose G has n vertices and vertex v_i has degree d_i. Then $\alpha = 1/d_i$ can be achieved for list love and hate games in time polynomial in n, for any $D < r \leq k$.*

Corollary 4.3 *Suppose G has n vertices and vertex v_i has out-degree d_i. Then we can achieve $\alpha = 1/d_i$ in the list asymmetric love and hate game in time polynomial in n, for any $k > 2D$.*

Proof. When each component of largest weights is an in-tree into a cycle, each vertex has two previously chosen neighbors, so it has at least $k - 2D$ choices. $\square$

Theorem 4.4 *Suppose G has n vertices and m edges, and $2D < r \leq k$. Then the we can achieve $\alpha = 1/(\lceil \log m/(\log(r - D) - \log D) \rceil + 1)$ for the list love and hate game in randomized time polynomial in n.*

Proof. We proceed as in Theorem 2.5. The number of choices for each vertex of each tree in the forest of maximum weights is $r - D$, and at most D will fail, so we must have $(D/r - D)^t \leq 1/m$, so $t = \lceil \log m/(\log(r - D) - \log D) \rceil$ suffices to give $\alpha = 1/(t + 1) = 1/(\lceil \log m/(\log(r - D) - \log D) \rceil + 1)$ as required. $\square$

Theorem 4.5 *Suppose G has n vertices. Suppose all G_e are the same, all vertices of G_e have loops so that each part is incompatible with itself, and G_e contains two disjoint subsets of respective sizes $s, s', D < s \leq s'$, such that no edge in G_e joins the s parts with the s' parts. Then the list love and hate game achieves $\alpha = 1 - D/s - O(1/n^c)$ in deterministic time polynomial in n, for any constant $c \geq 1$.*

Proof. We proceed as in Theorem 2.7, but instead of using sizes $\lfloor k/2 \rfloor$ we use size s for the partition of the problem into two parts. $\square$

Corollary 4.6 *Suppose G has n vertices and m edges, and $k > 3D$. Then we can achieve $\alpha = 1/(\lceil \log m/(\log(k - 2D) - \log D) \rceil + 1)$ the list asymmetric love and hate game in randomized time polynomial in n.*

Proof. As two neighbors are previously chosen, we now have $k - 2D$ choices instead of $k - D$. $\square$

References

[1] A. Fabrikant, C. Papadimitriou, and K. Talwar. *On the complexity of pure equilibria*, Proc. 36th ACM Symp. on Theory of Computing (2004) 604–612.

[2] M. Goemans and D. Williamson. *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, J. ACM, 42:11151145, 1995.

[3] D. Johnson, C.H. Papadimitriou, and M. Yannakakis. *How easy is local search?*, J. Computer and System Sciences, 37:79100, 1988.

[4] S. Khot, G. Kindler, E. Mossel, and R O'Donnell. *Optimal Inapproximability Results for Max-Cut and Other 2-Variable CSPs?*, Proc. 45th IEEE Symp. on Foundations of Computer Science (2002) 767–775.

[5] S. Poljak. *Integer linear programs and local search for max-cut*, Siam Journal of Computing, 24(4):822839, 1995.

[6] N. Robertson, P. Seymour and R. Thomas. *Permanents, Pfaffian orientations, and even directed circuits*, Annals of Math., 150 (1999), 929-975.

[7] T. J. Schaefer. *The complexity of satisfiability problems*, Proc. 10th ACM Symp. on Theory of Computing (1978) 216–226.

[8] A. Schaffer and M. Yannakakis. *Simple local search problems that are hard to solve*, SIAM journal on Computing, 20(1):5687, 1991.