Energy Minimization via Graph Cuts: Settling What is Possible

Daniel Freedman and Petros Drineas
Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY 12180

Abstract

The recent explosion of interest in graph cut methods in computer vision naturally spawns the question: what energy functions can be minimized via graph cuts? This question was first attacked by two papers of Kolmogorov and Zabih [23, 24], in which they dealt with functions with pairwise and triplewise pixel interactions. In this work, we extend their results in two directions. First, we examine the case of k-wise pixel interactions; the results are derived from a purely algebraic approach. Second, we discuss the applicability of provably approximate algorithms. Both of these developments should help researchers best understand what can and cannot be achieved when designing graph cut based algorithms.

Keywords: energy minimization, graph cuts.

1 Introduction

Since the early papers of Boykov *et al.* [9], there has been an explosion of interest in using combinatorial methods for energy minimization within computer vision. In particular, the techniques used are based on solving for the minimum cut across a graph. These techniques have spawned so much interest because they are effective: they allow for high quality optimization of a fairly wide variety of functions, in polynomial time. As a result, they have seen use in a number of different computer vision applications, including stereo [22], motion [10], and segmentation [8].

A natural and important question arises: what functions can be minimized via graph cuts? The papers by Kolomogorov and Zabih [23, 24] made good headway in attacking this problem. These papers showed necessary and sufficient conditions for the exact minimization of energy functions where there are terms depending on pairs of pixels (the so-called $\mathcal{F}^2$ class) and where there are terms depending on triples of pixels (the so-called $\mathcal{F}^3$ class). There are two natural extensions to this work. The first is characterize what is known about k-wise pixel interactions; the second is to characterize when provably approximate energy minimization can be achieved. More specifically, the main theoretical contributions of this paper are the following:

1. What is possible in the case of exact minimization.
   (a) A purely algebraic, and hence simplified, proof of the regularity results of Kolmogorov and Zabih [23, 24], for the cases $\mathcal{F}^2$ and $\mathcal{F}^3$.
   (b) An extension of these algebraic methods to the $\mathcal{F}^k$ case, and a discussion of the relationship of these results to known results on submodular functions.[1]
2. What is possible in the case of provably approximate minimization. It is well known that minimization of an arbitrary function of n discrete variables is an NP-complete problem. However, NP-completeness doesn't tell the whole story: provably approximate solutions are also acceptable in computer vision. Indeed, in many applications in which the label set is of size greater than 2, one has to resort to the use of iterations of "large moves" such as α-expansions or α-β swaps [10], which are in fact approximations to the energy minimization. The idea is to sequentially minimize a number of functions of binary variables exactly; such a binary minimization is called a large move. However, while the binary minimizations are exact, the overall minimization is approximate (with provable bound – see [10] for details). As algorithms based on these large moves are approximate in any case, adding in an extra level of approximation for the large move itself (i.e. the binary minimization) is not unreasonable.

The remainder of the paper is organized as follows. In Section 2, we investigate what can be said about exact algorithms. In Section 3, we turn to the problem of provably approximate algorithms.

2 Exact Algorithms

In this section, we extend the results of Kolmogorov and Zabih [23, 24] on exact minimization of energy functions

[1] Submodular functions were discussed in [24], though not [23]. A more thorough discussion of their relation to the problem at hand is given here.

via graph cuts. We begin, in Section 2.1, by laying out a useful theorem on the types pairwise functions that can be minimized via graph cut constructions. Using this simple result, the rest of the section is posed in an entirely algebraic manner, without explicit reference to graphs. In Section 2.2, we show how the regularity conditions can be derived very easily in the case of pairwise functions; in Section 2.3, we use a somewhat longer proof for the triplewise case. Although both of these results have already been proven in [23, 24], our proofs serve two purposes: they are simplified, purely algebraic proofs, making them easier to parse; and they show how these ideas can be generalized to the k-wise case, which we do in Section 2.4. In Section 2.5, we discuss the relationship of our conditions to the submodularity conditions.

2.1 A Useful Theorem

We begin by stating a theorem which is very important in the subsequent work. We note that this theorem is generally considered to be part of combinatorial optimization folklore, and a version of it may be found in [28].

Theorem 1 *Let $x_i \in \{0,1\}$ and let $E(x_1,\ldots,x_n) = \sum_{i,j} a_{ij}x_ix_j + L$, where L represents terms that are linear in the x_i plus any constants (i.e. $L = \sum_i a_ix_i + c$). Then E can be minimized via graph cut techniques if and only if $a_{ij} \leq 0$ for all i,j.*

Proof: We only prove the "if" direction; a proof of the "only if" direction may be found in [28]. With a little manipulation, such an E can be rewritten as

$$E = \sum_{i,j} a'_{ij}x_i(1-x_j) + L'$$

where $a'_{ij} = -a_{ij}$ and the linear term L' is altered. Ignoring the linear term L' for the moment, it is easy to see that minimizing E over the binary variables x_i is the same as finding a minimum cut in a complete graph with n vertices, one vertex corresponding to each x_i, and edge weights given by $w_{ij} = a'_{ij}$. The cut itself splits those vertices with $x_i = 0$ from those with $x_i = 1$; this is because choosing $x_i = 1$ and $x_j = 0$ adds a'_{ij} to the energy, whereas any other setting of x_i and x_j does not add a'_{ij} to the energy.[2] It is well known, from the theory of combinatorial optimization [28], that solving min-cut in polynomial time is possible if and only if the edge weights are non-negative. Thus, we must have that $a'_{ij} \geq 0$, so that $a_{ij} \leq 0$.

[2]Note that we would usually set $a'_{ji} = a'_{ij}$, so that $x_i = 0$ and $x_j = 1$ also yields the same result; this is the distinction between cuts across directed and undirected graphs.

We may now turn to the issue of the linear terms L'. Note that

$$\begin{aligned} L' &= \sum_i a'_ix_i + c' \\ &= \sum_{i:a'_i\geq 0} a'_ix_i + \sum_{i:a'_i<0} |a'_i|(1-x_i) + c'' \end{aligned}$$

Thus, we can add such terms into the graph formulation by simply adding in source (S) and sink (T) nodes, where S corresponds to 0 and T corresponds to 1. In this case, for each i for which $a'_i \geq 0$, we add in an edge from the node i to S with weight a'_i; and for each i for which $a'_i < 0$, we add in an edge from the node i to T with weight $|a'_i|$. All of these weights are non-negative, and thus we can apply graph cut techniques to optimize in polynomial time. □

2.2 Recasting the $\mathcal{F}^2$ Case

Before going on to discuss the k-wise case, we will discuss the simpler pairwise and triplewise cases. Of course, the results for these cases have already been demonstrated in [23, 24]; however, we use the same approach here as we do for the k-wise case, so it is worth reviewing these cases. (We also believe that the proofs presented here, which are purely algebraic, are simpler than those in [23, 24].)

The class of energy functions belonging to $\mathcal{F}^2$ includes all those with pairwise pixel interactions, i.e.

$$E(x_1,\ldots,x_n) = \sum_i E_i(x_i) + \sum_{i,j} E_{ij}(x_i,x_j) \quad (1)$$

We may now reprove the regularity results of [23, 24] very simply using Theorem 1. Note that we may write

$$\begin{aligned} E_{ij}(x_i,x_j) &= E_{ij}^{00}(1-x_i)(1-x_j) + E_{ij}^{01}(1-x_i)x_j \\ &\quad + E_{ij}^{10}x_i(1-x_j) + E_{ij}^{11}x_ix_j \end{aligned}$$

where $E_{ij}^{\alpha\beta} = E_{ij}(x_i = \alpha, x_j = \beta)$. Similarly, we may write

$$E_i(x_i) = E_i^0(1-x_i) + E_i^1x_i$$

Putting these terms back into equation (1) gives

$$E(x_1,\ldots,x_n) = \sum_{i,j}(E_{ij}^{00} + E_{ij}^{11} - E_{ij}^{01} - E_{ij}^{10})x_ix_j + L$$

where again L includes terms that are linear in the x_i, as well as any constants. Applying Theorem 1 says that such an energy can be minimized via graph cuts if and only if

$$E_{ij}^{00} + E_{ij}^{11} - E_{ij}^{01} - E_{ij}^{10} \leq 0 \quad \forall i,j$$

which is precisely the regularity condition of [23, 24].

2.3 Recasting the $\mathcal{F}^3$ Case

The class of energy functions belonging to $\mathcal{F}^3$ includes all those with triplewise pixel interactions, i.e.

$$E(x_1, \ldots, x_n) = \sum_i E_i(x_i) + \sum_{i,j} E_{ij}(x_i, x_j) + \sum_{i,j,k} E_{ijk}(x_i, x_j, x_k)$$

Before proving any results, let us introduce some notation. Greek letters, such as α and β, will typically refer to subsets of $\{1, \ldots, n\}$. We define $\mathbf{x}_\alpha$ to be $\prod_{\ell \in \alpha} x_\ell$. Also, we let $E_\beta^{ijk} = E_{ijk}(x_i = 1, i \in \beta)$.

Let us begin by expanding the function $E_{ijk}(x_i, x_j, x_k)$ in a polynomial series:

$$E_{ijk}(x_i, x_j, x_k) = \sum_{\alpha \subset \{i,j,k\}} a_\alpha^{ijk} \mathbf{x}_\alpha$$

To solve for the coefficients of the expansion, a_α^{ijk}, we can plug in all values of the binary values, leading to 8 equations in 8 unknowns. After some algebra, these equations can be solved to yield

$$a_\alpha^{ijk} = \sum_{\beta \subset \alpha} (-1)^{|\alpha| - |\beta|} E_\beta^{ijk}$$

The function E_{ijk} may therefore be written

$$E_{ijk}(x_i, x_j, x_k) = a_{ij}^{ijk} x_i x_j + a_{ik}^{ijk} x_i x_k + a_{jk}^{ijk} x_j x_k + a_{ijk}^{ijk} x_i x_j x_k + L$$

where L is a subquadratic term.

The key step is to convert E_{ijk}, which is an $\mathcal{F}^3$ function, into an $\mathcal{F}^2$ function via the introduction of an extra binary variable y_{ijk}. In particular, note that

$$x_i x_j x_k = \max_{y_{ijk} \in \{0,1\}} [(x_i + x_j + x_k - 2) y_{ijk}]$$

If $a_{ijk}^{ijk} \leq 0$, we may write

$$a_{ijk}^{ijk} x_i x_j x_k = \min_{y_{ijk} \in \{0,1\}} [a_{ijk}^{ijk} (x_i + x_j + x_k - 2) y_{ijk}]$$

which therefore gives

$$E_{ijk}(x_i, x_j, x_k) = \min_{y_{ijk}} [a_{ij}^{ijk} x_i x_j + a_{ik}^{ijk} x_i x_k + a_{jk}^{ijk} x_j x_k + a_{ijk}^{ijk} x_i y_{ijk} + a_{ijk}^{ijk} x_i y_{ijk} + a_{ijk}^{ijk} x_i y_{ijk} + L]$$

(L is a modified subquadratic term from the L introduced above.) Thus, E_{ijk} is written as a pairwise ($\mathcal{F}^2$) function, where we have introduced the extra variable y_{ijk}. In fact, we must take the minimum over y_{ijk}; however, since the entire function E will ultimately be minimized, this step simply introduces some extra variables to minimize over.

Now, what if $a_{ijk}^{ijk} > 0$? In a similar manner to the above, we can introduce an expansion

$$E_{ijk} = \sum_{\alpha \subset \{i,j,k\}} \bar{a}_\alpha^{ijk} \bar{\mathbf{x}}_\alpha$$

where $\bar{x}_i = 1 - x_i$ (and following the previous convention, $\bar{\mathbf{x}}_\alpha = \prod_{\ell \in \alpha} \bar{x}_\ell$.) It can be shown that

$$\bar{a}_\alpha^{ijk} = \sum_{\beta \subset \alpha} (-1)^{|\alpha| - |\beta|} \bar{E}_\beta^{ijk}$$

where $\bar{E}_\beta^{ijk} = E_{ijk}(x_i = 0, i \in \beta)$. In this case, some inspection shows that $\bar{a}_{ijk}^{ijk} = -a_{ijk}^{ijk}$. Therefore, if $a_{ijk}^{ijk} > 0$, then $\bar{a}_{ijk}^{ijk} < 0$, and (after some manipulation) we can write

$$E_{ijk}(x_i, x_j, x_k) = \min_{y_{ijk}} [\bar{a}_{ij}^{ijk} x_i x_j + \bar{a}_{ik}^{ijk} x_i x_k + \bar{a}_{jk}^{ijk} x_j x_k + \bar{a}_{ijk}^{ijk} x_i y_{ijk} + \bar{a}_{ijk}^{ijk} x_i y_{ijk} + \bar{a}_{ijk}^{ijk} x_i y_{ijk} + \bar{L}]$$

Note that the variables above are x_i and not $\bar{x}_i$. This is due to the fact that

$$\begin{aligned} \bar{x}_i \bar{x}_j &= (1 - x_i)(1 - x_j) \\ &= x_i x_j + \text{linear term} + \text{constant} \end{aligned}$$

so that any terms of the form $\bar{x}_i \bar{x}_j$ can be effectively replaced by $x_i x_j$ without affecting the expression (except through the precise forms of the subquadratic terms, which we do not care about).

Finally, let

$$b_\alpha^{ijk} = \begin{cases} a_\alpha^{ijk} & \text{if } a_{ijk}^{ijk} \leq 0, \\ \bar{a}_\alpha^{ijk} & \text{otherwise.} \end{cases}$$

Then we have that

$$E = \min_{\text{all } y_{ijk}} \left\{ \sum_{i,j,k} [b_{ij}^{ijk} x_i x_j + b_{ik}^{ijk} x_i x_k + b_{jk}^{ijk} x_j x_k + b_{ijk}^{ijk} x_i y_{ijk} + b_{ijk}^{ijk} x_i y_{ijk} + b_{ijk}^{ijk} x_i y_{ijk} + L_{ijk}] \right\}$$

Due to Theorem 1, we can ignore the linear terms L_{ijk}. We also note that since $\bar{a}_{ijk}^{ijk} = -a_{ijk}^{ijk}$, we must have that $b_{ijk}^{ijk} \leq 0$. Thus, we know that the terms involving the y_{ijk} variables satisfy the conditions of Theorem 1 (namely, that their coefficients be non-positive). Thus, we can look at the remainder of the function, i.e.

$$\begin{aligned} E' &= \sum_{i,j,k} b_{ij}^{ijk} x_i x_j + b_{ik}^{ijk} x_i x_k + b_{jk}^{ijk} x_j x_k \\ &= \sum_{ij} q_{ij} x_i x_j \end{aligned}$$

where $q_{ij} = \sum_k b_{ij}^{ijk}$. In this case, according to Theorem 1, the conditions under which $q_{ij} \leq 0$ are identical to the conditions under which the energy can be minimized via graph cut methods.

Using the expressions for a_α^{ijk} and $\bar{a}_\alpha^{ijk}$, a little algebra shows that

$$\begin{aligned} b_{ij}^{ijk} &= E_{ijk}(0,0,x_k) + E_{ijk}(1,1,x_k) \\ &\quad - E_{ijk}(0,1,x_k) - E_{ijk}(1,0,x_k) \end{aligned}$$

where $x_k = 0$ if $b_{ij}^{ijk} = a_{ij}^{ijk}$ and $x_k = 1$ otherwise. Thus,

$$\begin{aligned} q_{ij} = \sum_k [&E_{ijk}(0,0,x_k) + E_{ijk}(1,1,x_k) \\ &- E_{ijk}(0,1,x_k) - E_{ijk}(1,0,x_k)] \end{aligned} \quad (2)$$

It turns out that the condition that $q_{ij} \leq 0$ is precisely the regularity condition of [23, 24]. To see this, let us introduce the notation $x_{-ij} = \{x_\ell\}_{\ell \neq i,j}$, and

$$E_{ij}^{proj}(x_i, x_j) = E(x_i, x_j, x_{-ij})$$

where we have assumed the x_{-ij} are fixed, and therefore have suppressed them on the left-hand side. Then

$$E_{ij}^{proj}(0,0) = \sum_k E_{ijk}(0,0,x_k) + \sum_{i' \neq i, j' \neq j, k'} E_{i'j'k'}(x_{i'}, x_{j'}, x_{k'})$$

The second term does not depend on x_i or x_j. Thus using equation (2), $q_{ij} \leq 0$ becomes

$$E_{ij}^{proj}(0,0) + E_{ij}^{proj}(1,1) - E_{ij}^{proj}(0,1) - E_{ij}^{proj}(1,0) \leq 0$$

which is exactly the regularity condition of [23, 24].

2.4 The Generic $\mathcal{F}^k$ Case

We now come to the most generic case of energy functions with k-wise pixel interactions, labelled $\mathcal{F}^k$. We may use similar, though perhaps simpler, arguments as in the case of $\mathcal{F}^3$ to establish sufficient conditions for a function in $\mathcal{F}^k$ to be minimized via graph cut methods.

The first step is to realize that any function in $\mathcal{F}^k$ can be written as

$$E(x_1, \ldots, x_n) = \sum_{\alpha \subset \{1,\ldots,n\}, |\alpha| \leq k} a_\alpha \mathbf{x}_\alpha \quad (3)$$

where again $\mathbf{x}_\alpha = \prod_{\ell \in \alpha} x_\ell$. This fact can easily be proven, though we do not do so here. As described in Section 2.3, we can solve for the coefficients a_α by means of a linear system of 2^k equations in 2^k unknowns; the result (whose precise derivation is omitted here) is the same as in the case of $\mathcal{F}^3$ functions, i.e.

$$a_\alpha = \sum_{\beta \subset \alpha} (-1)^{|\alpha| - |\beta|} E_\beta \quad (4)$$

The second step is to convert an $\mathcal{F}^k$ function to an $\mathcal{F}^2$ function through the introduction of extra variables; this is precisely analogous to what was done in Section 2.3. Note that if $|\alpha| > 2$

$$\mathbf{x}_\alpha = \max_{y_\alpha \in \{0,1\}} \left[\left(\sum_{\ell \in \alpha} x_\ell - (|\alpha| - 1) \right) y_\alpha \right]$$

where y_α is the extra binary variable. If $a_\alpha \leq 0$, we may write

$$a_\alpha \mathbf{x}_\alpha = \min_{y_\alpha \in \{0,1\}} \left[a_\alpha \left(\sum_{\ell \in \alpha} x_\ell - (|\alpha| - 1) \right) y_\alpha \right] \quad (5)$$

The final step is to use the above fact to note that if $a_\alpha \leq 0$ for all α, we can combine equations (3) and (5) to yield

$$\begin{aligned} E(x_1, \ldots, x_n) &= L + \sum_{i,j} a_{ij} x_i x_j + \\ &\sum_{\alpha : |\alpha| > 2} \min_{y_\alpha \in \{0,1\}} \left[a_\alpha \left(\sum_{\ell \in \alpha} x_\ell - (|\alpha| - 1) \right) y_\alpha \right] \end{aligned}$$

where as usual, L represents linear terms and the constant. Thus, the minimization of E can be rewritten as follows:

$$\min_{x_i} E(x_i) = \min_{x_i, y_\alpha} \tilde{E}(x_i, y_\alpha)$$

where

$$\tilde{E}(x_i, y_\alpha) = \sum_{\alpha : |\alpha| > 2} \sum_{\ell \in \alpha} a_\alpha x_\ell y_\alpha + \sum_{i,j} a_{ij} x_i x_j + L$$

We can apply Theorem 1 to this function to discover that E can be minimized by graph cut techniques if

$$a_\alpha \leq 0 \quad \forall \alpha : 2 \leq |\alpha| \leq k$$

Plugging in the expression for a_α from equation (4) leads to the following sufficient conditions for minimization of E via graph cut methods:

$$\boxed{\sum_{\beta \subset \alpha} (-1)^{|\alpha| - |\beta|} E_\beta \leq 0 \quad \forall \alpha : 2 \leq |\alpha| \leq k \quad (6)}$$

where as before $E_\beta = E(x_i = 1, i \in \beta)$. The inequalities of (6) represent the main result of this section of the paper.

Note that an extra argument is invoked in the case of $\mathcal{F}^3$, to eliminate the condition that $a_\alpha \leq 0$ for $|\alpha| = 3$. Such an argument relied on the fact that an expansion could also be performed on the $\bar{x}_i$ variables, where $\bar{x}_i = 1 - x_i$; it was then shown that $\bar{a}_\alpha = -a_\alpha$ for $|\alpha| = 3$, so that in this case either $a_\alpha \leq 0$ or $\bar{a}_\alpha \leq 0$. Unfortunately, this is not true for $k > 3$; indeed, for $k = 4$ we have that $\bar{a}_\alpha = a_\alpha$.

2.5 Submodularity

A well known fact from the theory of combinatorial optimization is that the class of submodular functions can be optimized in polynomial time [26]. This fact was noted in [24] (though not [23]), but we wish to add some further discussion of these functions here.

Suppose S is a set with n elements. A set-valued function $f : S \to \mathbb{R}$ is said to be submodular if

$$f(X \cap Y) + f(X \cup Y) \leq f(X) + f(Y) \quad \forall X, Y \subset S$$

One can, of course, easily move from set-valued functions to binary-valued functions, by letting inclusion of element i in a set correspond to $x_i = 1$, and exclusion to $x_i = 0$. We wish to make two comments regarding the relationship between submodularity and the conditions described here:

1. The relationship between the conditions for $\mathcal{F}^k$ derived in (6) and submodularity is unknown, but the conditions are not the same. This can be clearly seen from the fact that the submodularity conditions always involve exactly 4 terms, whereas the inequalities in (6) can involve more.

2. It is not obvious from inspection as to how to specialize the submodularity conditions to classes of functions like $\mathcal{F}^k$; these conditions will look the same, no matter how many pixel are allowed to interact. (Of course, the number of such conditions applying may decrease, but the way in which this takes place is also not obvious from inspection.) The new conditions, by contrast, relate to precisely the function classes $\mathcal{F}^k$ which are relevant for computer vision; in many vision applications, the number of interacting pixels k is fixed. Thus, from a computer vision point-of-view, these conditions are important. For example, it is clear from the inequalities of (6) precisely which new inequalities get added as k increases.

3 Provably Approximate Algorithms

Perhaps the most straight-forward question that arises from the above discussion is how to address optimization problems on n boolean variables *if* the conditions for exact optimization stated in Sections 2.2-2.5 are *not* satisfied. Not surprisingly, general formulations of such optimization problems have been extensively studied in Theoretical Computer Science and, in certain cases, polynomial time approximation schemes are known. In this section, we shall define the most general formulation of optimization problems on n boolean variables and we shall briefly review recent, state-of-the-art, *provable* algorithmic results for tackling this general formulation. Our goal is to communicate these recent developments to the Computer Vision community, since we believe that some of the algorithmic ideas developed in the Theoretical Computer Science community are quite useful in practical settings. Our exposition should serve as a roadmap to researchers seeking to bridge the gap between theory and practical applications.

3.1 Defining Max-r-CSP optimization problems on n Boolean variables

Consider the following optimization problem. We are given n boolean variables $x_1, x_2, \ldots, x_n \in \{0, 1\}$ and a set of m functions $f_1, f_2, \ldots, f_m$, where each function depends on *at most* r (out of n) Boolean variables. Let $\mathcal{Z}$ be the set of all 0-1 strings of length r – thus, $|\mathcal{Z}| = 2^r$, and let $A^{(z)}$ be r-dimensional $\underbrace{n \times n \ldots \times n}_{r}$ arrays for all 0-1 strings $z \in \mathcal{Z}$. To understand what these arrays represent, consider a specific r-dimensional array $A^{(z)}$, $z = \underbrace{0 \ldots 0}_{r}$. The $(i_1, \ldots, i_r)$ element of this array represents the value of a function on the variables $x_{i_1}, \ldots, x_{i_r}$ when $x_{i_1} = 0, \ldots, x_{i_r} = 0$; if none of the $f_1, \ldots, f_m$ is a function of these variables, then this entry of $A^{(z)}$ is zero. Let $P(x) = P(x_1, \ldots, x_n)$ be the polynomial

$$P(x) = \sum_{z \in \mathcal{Z}} \sum_{i_1, \ldots, i_r = 1}^{n} A^{(z)}(i_1, \ldots, i_r) \prod_{x_i : z_i = 1} x_i \prod_{x_i : z_i = 0} (1 - x_i)$$

Notice that the term $\prod_{x_i : z_i = 1} x_i \prod_{x_i : z_i = 0} (1 - x_i)$ determines whether the entries of an r-dimensional array $A^{(z)}$ contribute to the value of $P(x)$, depending on the values assumed by the x_i's. We now define the following optimization problem:

$$\text{OPT} = \max_{x \in \{0,1\}^n} P(x). \tag{7}$$

It is easy to see that the above definition is a generalization of the optimization problems defined for the energy functions of Sections 2.2-2.4, where restrictive constraints were placed on the entries of the r-dimensional arrays $A^{(z)}$. The *unconstrained* optimization problem (7) is NP-hard, as well as Max-SNP hard [21]. Thus, unless $P = NP$, no polynomial time approximation schemes (PTAS) exist for this problem. We also note that any problem in the class Max-SNP can be formulated as a Max-r-CSP problem for some constant r. (Recall that a PTAS is an algorithm that for every fixed $\epsilon > 0$ achieves an approximation ratio of $1 - \epsilon$ in time which is $poly(n)$ but perhaps exponential in $1/\epsilon$. Such a scheme is a fully polynomial time approximation scheme (FPTAS) if the running time is $poly(n, 1/\epsilon)$.)

3.2 An easy algorithm for dense instances of Max-r-CSP problems

In an important paper, Alon *et. al.* [1, 2] presented a simple PTAS for a special class of Max-r-CSP problems. Given the optimization problem of equation (7) on n variables, sample *uniformly at random* c variables, thus creating a new, *induced*, optimization problem. We notice that c is a constant, independent of n, to be specified shortly. Let

$$P_c(x) = \sum_{z \in \mathcal{Z}} \sum_{i_1,\ldots,i_r=1}^{c} A_c^{(z)}(i_1,\ldots,i_r) \prod_{x_i : z_i = 1} x_i \prod_{x_i : z_i = 0} (1-x_i)$$

In the above formula, $A_c^{(z)}$ is the induced $\underbrace{c \times \ldots \times c}_{r}$ submatrix of $A^{(z)}$ that emerges by keeping the elements that correspond to the c sampled variables. [1, 2] prove that solving the optimization problem

$$Z = \max_{x \in \{0,1\}^c} P_c(x), \tag{8}$$

returns a solution Z such that, with high probability,

$$\left| \frac{n^r}{q^r} Z - \text{OPT} \right| \leq \epsilon n^r W_{\max}, \tag{9}$$

for any $\epsilon > 0$, if $c = poly(1/\epsilon)$, independent of n. Here we assume that $W_{\max}$ is a constant – independent of n – that denotes the maximal entry in the arrays $A^{(z)}$. A simple application of the probabilistic method (see, e.g., [27]) shows that if all the $A^{(z)}$ are *dense* and their entries are $\Omega(1)$, i.e., each $A^{(z)}$ contains at least $\Omega(n^r)$ non-zero elements, then OPT $= \Omega(n^r)$, and thus equation (9) proves the existence of a FPTAS for *dense* instances of Max-r-CSP problems. Finally, we note that solving the optimization problem of equation (8) is straight-forward, since c is independent of n; for more details see [1, 2].

3.3 Other PTAS for dense instances of Max-r-CSP problems

To make the following discussion more concrete, we shall have as a running example a specific, well-known, Max-2-CSP problem, the Max-Cut problem. The Max-Cut problem is one of the most well studied problems in Theoretical Computer Science, and, along with numerous other graph and combinatorial problems, is contained in the Max-2-CSP class of problems [21]. In the weighted version of the Max-Cut problem, the input consists of the $n \times n$ adjacency matrix A of an undirected graph $G = (V, E)$ with n vertices, and the objective of the problem is to find a cut, i.e., a partition of the vertices into two subsets V_1 and V_2, such that the sum of the weights of the edges of E that have one endpoint in V_1 and one endpoint in V_2 is maximized. In the format of equation (7),

$$\textbf{MAX-CUT}\,[G] = \max_{x \in \{0,1\}^n} \sum_{i,j=1}^{n} A^{(10)}(i,j) x_i (1 - x_j)$$

The Max-Cut problem has applications in such diverse fields as statistical physics and circuit layout design [7] and has been extensively studied theoretically [29, 19]. It is known to be NP-hard, both for general graphs and when restricted to dense graphs [3], where a graph on n vertices is dense if it contains $\Omega(n^2)$ edges. Thus, much effort has gone into designing and analyzing approximation algorithms for the Max-Cut problem. It is known that there exists a 0.878-approximation algorithm [19]; it is also known from the PCP results of Arora *et. al.* [4] that (unless $P = NP$) there exists a constant α, bounded away from 1, such that there does not exist a polynomial time α-approximation algorithm. In particular, this means that there does not exist a polynomial time approximation scheme (PTAS) for the general Max-Cut problem.

Work originating with [3] has focused on designing PTASs for the Max-Cut problem, as well as larger classes of NP-hard optimization problems, such as the Max-2-CSP or the Max-r-CSP class of problems, when the problem instances are *dense* [3, 11, 17, 20, 18, 1, 2]. An instance of a Max-r-CSP problem is considered dense if $\Omega(n^r)$ entries in $A^{(z)}$ are non-zero for *all* $z \in \mathcal{Z}$. Intuitively, all the matrices $A^{(z)}$ must be dense.

[3] and [11], using quite different methods, designed approximation algorithms for Max-Cut that achieve an additive error of $\epsilon n^2 W_{\max}$ (where $\epsilon > 0$, $\epsilon \in \Omega(1)$ is an error parameter, and $W_{\max}$ is the maximal edge weight in the graph G) in time $poly(n)$ (and exponential in $1/\epsilon$). This results implies relative error for dense instances of such problems, since, using the probabilistic method [27], it is easy to prove that a graph with $\Omega(n^2)$ edges has a cut containing at least $\Omega(n^2)$ edges. [3] can be extended to solve all Max-r-CSP optimization problems with an additive error $\epsilon n^r W_{\max}$, where $W_{\max}$ is the maximal entry in any of the matrices $A^{(z)}$.

In [20] it was shown that a constant-sized (with respect to n) sample of a graph is sufficient to determine whether a graph has a cut close to a certain value. This work investigated dense instances of NP-hard problems from the viewpoint of query complexity and property testing and yielded an $O(1/\epsilon^5)$ time algorithm to approximate, among other problems, dense instances of Max-Cut. [17] and [18] examined the regularity properties of dense graphs and developed a new method to approximate matrices; this led to a PTAS for dense instances of all Max-2-CSP, and more generally for dense instances of all Max-r-CSP, problems, assuming that all entries in the $A^{(z)}$'s are $\Omega(W_{\max})$. Finally, we note that it has been recently shown that there does exist a

PTAS for Max-Cut and all Max-2-CSP problems restricted to slightly subdense, i.e., $\Omega(n^2/\log n)$ edges, graphs [12].

3.4 Uniform vs. importance sampling

All the approximation algorithms of Section 3.3 involve sampling elements of the input uniformly at random in order to construct a sub-problem, which is then used to compute an approximation to the original problem with additive error at most $\epsilon n^r W_{\max}$ [3, 11, 17, 20, 18, 1, 2]. Such methods are clearly not useful for nondense graphs, since with such an error bound a trivial approximate solution (zero) would always suffice. Uniform sampling does have the advantage that it can be carried out "blindly" since the "coins" can be tossed before seeing the data. Subsequently, given either random access or one pass, i.e., one sequential read, through the data, samples from the data may be drawn and then used to compute. Such uniform sampling is appropriate for problems that have nice uniformity or regularity properties [17].

With the additional flexibility of several passes over the data, we may **use one pass to assess the "importance" of a piece (or set of pieces) of data** and determine the probability with which it (or they) should be sampled, and a second pass to actually draw the sample. Such importance sampling has a long history [25]. The power of using information to construct nonuniform sampling probabilities has also been demonstrated in recent work examining so-called oblivious versus so-called adaptive sampling [5, 6]. For instance, it was proven that in certain cases approximation algorithms (for matrix problems such as those discussed in [13, 14, 15]) which use oblivious uniform sampling cannot achieve the error bounds that are achieved by adaptively constructing nonuniform sampling probabilities.

In [16], a PTAS is presented for all dense Max-2-CSP problems, using nonuniform sampling probabilities in the construction of the sub-problem to be solved. The use of these methods improved previous results [20, 2]. To understand this result, we again focus on the Max-Cut problem. Let A be the $n \times n$ adjacency matrix of a graph $G = (V, E)$, let ϵ be a constant independent of n, and recall that $\|A\|_F^2 = \sum_{ij} A_{ij}^2$. The algorithm of [16], upon being input A, returns an approximation Z to the Max-Cut of A such that with high probability

$$|Z - \textbf{MAX-CUT}\,[A]| \leq \epsilon n \|A\|_F \,. \tag{10}$$

The algorithm makes a small constant number of passes, i.e., sequential reads, through the matrix A and then uses additional space and time that is independent of n in order to compute the approximation. Judiciously-chosen and data-dependent nonuniform probability distributions are used in the sampling process in order to obtain bounds of the form (10). The approach of [16] is quite intuitive and seems to provide a first step towards a simple, practical algorithm: pick a constant number – polynomially dependent in $1/\epsilon$ – of boolean variables, and maximize the induced polynomial. However, the boolean variables should be picked with non-uniform probabilities, i.e., they they should be picked with probabilities that depend on the influence of the variables on the function output. To make this point clearer, it is useful to think of Max-Cut: instead of solving Max-Cut on an induced subgraph of vertices that are chosen uniformly at random, we should pick the vertices with non-uniform probabilities that depend on their *degrees*.

To assess the quality of the error bound (10), we again focus on the Max-Cut problem; our observations generalize to all problems in Max-2-CSP. Notice that in general $\sqrt{2\,|E|} = \|A\|_F < n$, where $|E|$ is the cardinality of the edge set of an unweighed graph G. In this case, the error bound (10) becomes $\epsilon n \sqrt{2\,|E|}$. This is an improvement over the previous results of ϵn^2 [20, 2].

Acknowledgments

This work was supported in part by the U.S. National Science Foundation, under Award IIS-0133144.

References

[1] N. Alon, W. de la Vega, R. Kannan, and M. Karpinski. Random sampling and approximation of MAX-CSP problems. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 232–239, 2002.

[2] N. Alon, W. de la Vega, R. Kannan, and M. Karpinski. Random sampling and approximation of MAX-CSPs. *Journal of Computer and System Sciences*, 67:212–243, 2003.

[3] S. Arora, D. Karger, and M. Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 284–293, 1995.

[4] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. In *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[5] Z. Bar-Yossef. *The Complexity of Massive Data Set Computations*. PhD thesis, University of California, Berkeley, 2002.

[6] Z. Bar-Yossef. Sampling lower bounds via information theory. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 335–344, 2003.

[7] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36:493–513, 1988.

[8] Y. Boykov and V. Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *Proc. ICCV*, 2003.

[9] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *Proc. ICCV*, volume 1, pages 377–384, 1999.

[10] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Machine Intell.*, 23(11):1222–1239, 2001.

[11] W. de la Vega. MAX-CUT has a randomized approximation scheme in dense graphs. *Random Structures and Algorithms*, 8(3):187–198, 1996.

[12] W. de la Vega and M. Karpinski. A polynomial time approximation scheme for subdense MAX-CUT. Technical Report TR02-044, Electronic Colloquium on Computational Complexity, 2002.

[13] P. Drineas, R. Kannan, and M. Mahoney. Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. Technical Report YALEU/DCS/TR-1269, Yale University Department of Computer Science, New Haven, CT, February 2004.

[14] P. Drineas, R. Kannan, and M. Mahoney. Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. Technical Report YALEU/DCS/TR-1270, Yale University Department of Computer Science, New Haven, CT, February 2004.

[15] P. Drineas, R. Kannan, and M. Mahoney. Fast Monte Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition. Technical Report YALEU/DCS/TR-1271, Yale University Department of Computer Science, New Haven, CT, February 2004.

[16] P. Drineas, R. Kannan, and M. Mahoney. Sampling subproblems of Max-Cut problems and approximation algorithms. *22nd Annual Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science 3404*, pages 57–68, 2005.

[17] A. Frieze and R. Kannan. The regularity lemma and approximation schemes for dense problems. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 12–20, 1996.

[18] A. Frieze and R. Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19(2):175–220, 1999.

[19] M. X. Goemans and D. P. Williamson. .878-approximation algorithms for MAX CUT and MAX 2SAT. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 422–431, 1994.

[20] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 339–348, 1996.

[21] S. Khanna, M. Sudan, L. Trevisan, and D. Williamson. The approximability of constraint satisfaction problems. *SIAM Journal on Computing*, 30(6):1863–1920, 2001.

[22] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *Proc. ECCV*, 2002.

[23] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts. In *Proc. ECCV*, 2002.

[24] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Anal. Machine Intell.*, 26(2):147–159, 2004.

[25] J. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer-Verlag, New York, 2001.

[26] L. Lovasz. *Combinatorial optimization : algorithms and complexity*. Dover Publications, 1998.

[27] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, 1995.

[28] C. Papadimitriou and K. Steiglitz. *An algorithmic theory of numbers, graphs, and convexity*. Soc for Industrial and Applied Math, 1986.

[29] S. Poljak and Z. Tuza. Maximum cuts and large bipartite subgraphs. In W. Cook, L. Lovász, and P. Seymour, editors, *Combinatorial Optimization*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 20, pages 181–244. American Mathematical Society, 1995.