Faster Algorithms for MAX CUT and MAX CSP, with Polynomial Expected Time for Sparse Instances

Alexander D. Scott[1] and Gregory B. Sorkin[2]

[1] Department of Mathematics, University College London, London WC1E 6BT, UK.
scott@math.ucl.ac.uk

[2] IBM T.J. Watson Research Center, Department of Mathematical Sciences, Yorktown Heights NY 10598, USA.
sorkin@watson.ibm.com

Abstract. We show that a random instance of a weighted maximum constraint satisfaction problem (or MAX 2-CSP), whose clauses are over pairs of binary variables, is solvable by a deterministic algorithm in polynomial expected time, in the "sparse" regime where the expected number of clauses is half the number of variables. In particular, a maximum cut in a random graph with edge density $1/n$ or less can be found in polynomial expected time.

Our method is to show, first, that if a MAX 2-CSP has a connected underlying graph with n vertices and m edges, the solution time can be deterministically bounded by $2^{(m-n)/2}$. Then, analyzing the tails of the distribution of this quantity for a component of a random graph yields our result. An alternative deterministic bound on the solution time, as $2^{m/5}$, improves upon a series of recent results.

1 Introduction

In this paper we prove that a maximum cut of a sparse random graph can be found in polynomial expected time.

Theorem 1. *For any $c \le 1$, a maximum cut of a random graph $G(n, c/n)$ can be found in time whose expectation is* $\text{poly}(n)$*, and using space $O(m+n)$, where m is the size of the graph.*

Our approach is to give a deterministic algorithm and bound its running time on any graph in terms of size and cyclomatic number. We then bound the expected running time for random instances by bounding the distribution of cyclomatic number in components of a sparse random graph.

Theorem 2. *Let G be a connected graph with m edges and n vertices. There is an algorithm that finds a maximum cut of G in time $O(m+n)\min\{2^{m/5}, 2^{(m-n)/2}\}$, and in space $O(m+n)$.*

We remark that the bound in Theorem 2 is of independent interest, and improves on previous algorithms giving bounds of $2^{m/4}\operatorname{poly}(m+n)$ [KF02] and $2^{m/3}\operatorname{poly}(m+n)$ [GHNR].

In fact, the algorithm employs several local reductions that take us outside the class of MAX CUT problems. We therefore work with the larger class MAX 2-CSP of weighted maximum constraint satisfaction problems consisting of constraints on pairs (and singletons) of variables, where each variable may take two values. Theorems 1 and 2 are then special cases of the more general Theorems 3 and 5 below.

1.1 Context

Our results are particularly interesting in the context of phase transitions for various maximum constraint-satisfaction problems. Since the technicalities are not relevant to our result, but only help to put it into context, we will be informal. It is well known that a random 2-SAT formula with density $c < 1$ (where the number of clauses is c times the number of variables) is satisfiable with probability tending to 1, as the number n of variables tends to infinity, while for $c > 1$, the probability of satisfiability tends to 0 as $n \to \infty$ [CR92, Goe96, FdlV92]; for more detailed results, see [BBC$^+$01]. More recently, MAX 2-SAT has been shown to exhibit similar behavior, so for $c < 1$, only an expected $\Theta(1/n)$ clauses go unsatisfied, while for $c > 1$, $\Theta(n)$ clauses are unsatisfied [CGHS03, CGHS].

For a random graph $G(n, c/n)$, with $c < 1$ the graph almost surely consists solely of small trees and unicyclic components, while for $c > 1$, it almost surely contains a "giant", complex component, of order $\Theta(n)$ [Bol01]. Again, [CGHS] proves the related facts that in a maximum cut of such a graph, for $c < 1$ only an expected $\Theta(1)$ edges fail to be cut, while for $c > 1$ it is $\Theta(n)$.

Theorem 3 is concerned with algorithms that run in polynomial expected time. Results on coloring random graphs in polynomial expected time can be found in [KV02, COMS, TCO03]. For both MAX CUT and MAX 2-SAT, it seems likely that the mostly-satisfiable (or mostly-cuttable) sparse instances are algorithmically easy, while the not-so-satisfiable dense instances are algorithmically hard. While, as far as we are aware, little is known about the hardness of dense instances, our results here confirm that not only are typical sparse MAX CUT instances easy, but even the atypical ones can be accommodated in polynomial expected time; see the Conclusions for further discussion.

1.2 Outline of Proof

Our proof of Theorem 3 has a few main parts. Since the maximum cut of a graph is the combination of maximum cuts of each of its connected components, it suffices to bound the expected time to partition the component containing a fixed vertex.

In Theorem 5 we show that Algorithm A's running time on a component is bounded by a function of the component's cyclomatic number, the number of edges less the number of vertices plus one. For brevity we will call this the "excess" (a slight abuse of the standard meaning, which is just edges minus vertices). Theorem 5 also gives a $2^{m/5}\,\mathrm{poly}(m+n)$ bound on the running time.

In the randomized setting, Lemma 8 provides a bound on the exponential moments of the excess of a component. It does so by "exploring" the component as a branching process, dominating it with a similar process, and analyzing the latter as a random walk. This gives stochastic bounds on the component order u and, conditioned upon u, the "width" w (to be defined later); the excess is easily stochastically bounded in terms of u and w.

Finally, we combine the running times, which are exponentially large in the excess, with the exponentially small large-deviation bounds on the excess, to show that Algorithm A runs in polynomial expected time.

2 Solving a Maximum Constraint-Satisfaction Instance

We begin by defining a class of weighted maximum constraint satisfaction problems, or MAX CSPs, generalizing MAX CUT, and (in Theorem 5) bounding their running time in terms of parameters of an instance.

2.1 Weighted Maximum Constraint-Satisfaction Problems

We may think of MAX CUT as a MAX CSP in which the constraints simply prefer opposite "colors" on the endpoints of each edge, and all constraints have the same "weight". We generalize this not only for the sake of a more general result but because we need to: intermediate steps of Algorithm A, applied to a MAX CUT instance, generate instances of more general type.

For our purposes, a general instance of a (weighted) MAX 2-CSP consists of a graph $G = (V, E)$, and a score function consisting of: a sum of "monadic constraint" scores of each vertex and its color, "dyadic" scores of each edge and the pair of colors at its endpoints, and (for notational convenience) a single "niladic" score (a constant). Specifically, there is a (niladic) score s_0; for each $x \in V$ (monad) there is a pair of scores s^x_R, s^x_B corresponding to the two ways that the vertex could be colored; and for each edge $e = \{x, y\} \in E$ (dyad) there is a 4-tuple of scores $s^{xy}_{BB}, s^{xy}_{BR}, s^{xy}_{RB}, s^{xy}_{RR}$ corresponding to the four ways that the edge could be colored, and the score of a coloring $\phi : V \to \{R, B\}$ is

$$S(\phi) := s_0 + \sum_{x\in V} s^x_{\phi(x)} + \sum_{\{x,y\}\in E} s^{xy}_{\phi(x)\phi(y)}.$$

(Note that for any $C, D \in \{R, B\}$, s^{xy}_{CD} and s^{yx}_{DC} refer to the same score, and thus must be equal.) Let S refer to the full collection of scores s^x_C and s^{xy}_{CD} as above. Then $\mathrm{MAX}(V, E, S)$ is the computational problem of finding a coloring ϕ achieving $\max_\phi S(\phi)$.

As one quick example, MAX 2-SAT is such a MAX CSP. Using colors T (true) and F (false), a SAT constraint $\bar{X} \vee Y$ is modelled as a dyadic constraint mapping (T, F) to score 0 (unsatisfied) and any other coloring to score 1 (satisfied).

Another example is MAX DICUT, the problem of partitioning a directed graph to maximize the number of edges passing from the left side to the right.

Our main result is that a weighted MAX 2-CSP on a random graph $G(n, c/n)$, $c < 1$, can be solved in polynomial expected time, per the following theorem.

Theorem 3. *For any $c \leq 1$ and any n, let $G(n, c/n)$ be a random graph, and let (G, S) be any weighted* MAX *2-*CSP *instance over this graph. Then (G, S) can be solved exactly in expected time* $\operatorname{poly}(n)$*, and in space $O(m + n)$.*

2.2 Algorithm A

In this section we give an algorithm for solving instances of weighted MAX 2-CSP. The algorithm will use 3 types of reductions. We begin by defining these reductions. We then show how the algorithm fixes a sequence in which to apply the reductions by looking at the underlying graph of the CSP. This sequence defines a tree of CSPs, which can be solved bottom-up to solve the original CSP. Finally, we bound the algorithm's time and space requirements.

Reductions The first two reductions each produce equivalent problems with fewer vertices, while the third produces a pair of problems, both with fewer vertices, one of which is equivalent to the original problem.

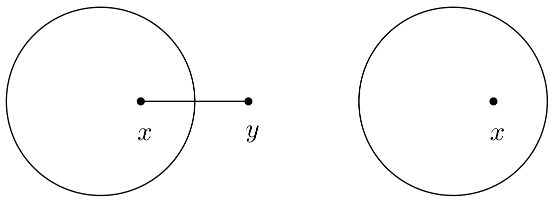
Reduction I Let y be a vertex of degree 1, with neighbor x. Reducing (V, E, S) on y results in a new problem (V', E', S') with $V' = V \setminus y$ and $E' = E \setminus xy$. S' is the restriction of S to V' and E', except that for $C, D \in \{R, B\}$ we set

$$s'^x_C = s^x_C + \max_D \{s^{xy}_{CD} + s^y_D\},$$

i.e., we set

$$s'^x_R = s^x_R + \max\{s^{xy}_{RR} + s^y_R, s^{xy}_{RB} + s^y_B\}$$
$$s'^x_B = s^x_B + \max\{s^{xy}_{BB} + s^y_B, s^{xy}_{BR} + s^y_R\}.$$

Note that any coloring ϕ' of V' can be extended to a coloring of V in two ways, namely ϕ_R and ϕ_B (corresponding to the two colorings of x); and the defining property of the reduction is that $S'(\phi') = \max\{S(\phi_R), S(\phi_B)\}$. In particular, $\max_{\phi'} S'(\phi') = \max_\phi S(\phi)$, and an optimal coloring ϕ' for the problem $\text{MAX}(V', E', S')$ can be extended to an optimal coloring ϕ for $\text{MAX}(V, E, S)$, in constant time.



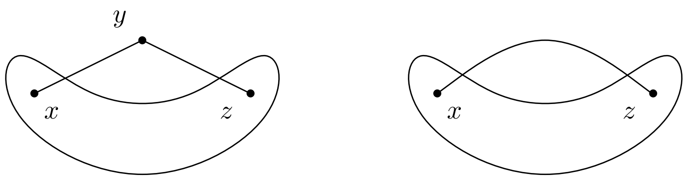
Reduction II Let y be a vertex of degree 2, with neighbors x and z. Reducing (V, E, S) on y results in a new problem (V', E', S') with $V' = V \setminus y$ and $E' = (E \setminus \{xy, yz\}) \cup \{xz\}$. S' is the restriction of S to V' and E', except that for $C, D, E \in \{R, B\}$ we set

$$s'^{xz}_{CD} = s^{xz}_{CD} + \max_E \{s^{xy}_{CE} + s^{yz}_{ED} + s^y_E\}$$

i.e., we set

$$\begin{aligned} s'^{xz}_{RR} &= s^{xz}_{RR} + \max\{s^{xy}_{RR} + s^{yz}_{RR} + s^y_R, s^{xy}_{RB} + s^{yz}_{BR} + s^y_B\} \\ s'^{xz}_{RB} &= s^{xz}_{RB} + \max\{s^{xy}_{RR} + s^{yz}_{RB} + s^y_R, s^{xy}_{RB} + s^{yz}_{BB} + s^y_B\} \\ s'^{xz}_{BR} &= s^{xz}_{BR} + \max\{s^{xy}_{BR} + s^{yz}_{RR} + s^y_R, s^{xy}_{BB} + s^{yz}_{BR} + s^y_B\} \\ s'^{xz}_{BB} &= s^{xz}_{BB} + \max\{s^{xy}_{BR} + s^{yz}_{RB} + s^y_R, s^{xy}_{BB} + s^{yz}_{BB} + s^y_B\}, \end{aligned}$$

where our notation presumes that if xz was not an edge in E, then $s^{xz}_{CD} = 0$ for all colors C and D. As in Reduction I, any coloring ϕ' of V' can be extended to V in two ways, ϕ_R and ϕ_B, and S' picks out the larger of the two scores. Also as in Reduction I, $\max_{\phi'} S'(\phi') = \max_\phi S(\phi)$, and an optimal coloring ϕ' for $\text{MAX}(V', E', S')$ can be extended to an optimal coloring ϕ for $\text{MAX}(V, E, S)$, in constant time. (Note that neither multiple edges nor loops are created by this reduction, nor the next one.)



Reduction III Let y be a vertex of degree 3 or higher. Where reductions I and II each had a single reduction of (V, E, S) to (V', E', S'), here we define a pair of reductions of (V, E, R), to (V', E', S^R) and (V', E', S^B), corresponding to assigning the color R or B to y. We define $V' = V \setminus y$, and E' as the restriction of E to $V \setminus y$. For $C, D, E \in \{R, B\}$, S^C is the restriction of S to $V \setminus y$, except that we set

$$(s^C)_0 = s_0 + s^y_C,$$

and, for every neighbor x of y,

$$(s^C)^x_D = s^x_D + s^{xy}_{DE}.$$

In other words, S^R is the restriction of S to $V \setminus y$, except that we set $(s^C_0) = s_0 + s^y_C$ and, for every neighbor x of y,

$$\begin{aligned} (s^R)^x_R &= s^x_R + s^{xy}_{RR} + s^y_R \\ (s^R)^x_B &= s^x_B + s^{xy}_{BR} + s^y_R. \end{aligned}$$

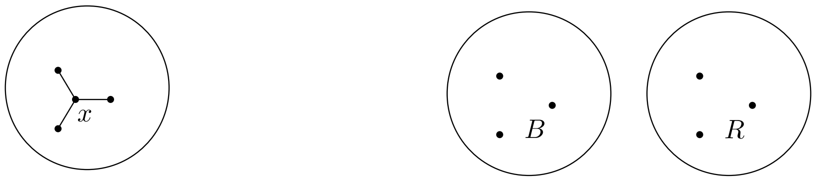
Similarly S^B is given by $(s^B)_0 = s_0 + s^y_B$ and, for every neighbor x of y,

$$(s^B)^x_R = s^x_R + s^{xy}_{RB} + s^y_B$$
$$(s^B)^x_B = s^x_B + s^{xy}_{BB} + s^y_B.$$

As in the previous reductions, any coloring ϕ' of $V \setminus y$ can be extended to V in two ways, ϕ_R and ϕ_B, corresponding to the color given to y, and now (this is different!) $S_R(\phi') = S(\phi_R)$ and $S_B(\phi') = S(\phi_B)$. Furthermore,

$$\max\{\max_{\phi'} S_R(\phi'), \max_{\phi'} S_B(\phi')\} = \max_{\phi} S(\phi),$$

and an optimal coloring on the left can be extended to an optimal coloring on the right in time $O(\deg(y))$.



Defining Algorithm A in terms of these reductions is straightforward, and it should come as no surprise that the running time is polynomial in n and m, times 2 raised to the power of the number of times reduction III is employed. We now detail this.

Setup Phase: Choosing a Sequence of Reductions First, observe that the two problems generated by reduction III have different score sets, but the same underlying graph. Thus each of the three reductions, considering only the graphs and ignoring the scores, reduces a graph to a subgraph of smaller order.

Given an input graph G of order n, Algorithm A begins by constructing a sequence $G_1, G_2, \ldots, G_i$, of at most n graphs, where $G_1 = G$ is the input graph, each subsequent graph is a reduction of its predecessor graph (ignoring scores), and the final graph G_i has no edges.

Specifically, with an ordering on the vertices of G: if G has minimum degree 1, apply reduction I to the first vertex of degree 1; if G has minimum degree 2, apply reduction II to the first vertex of degree 2; and otherwise, apply reduction III to the first vertex of maximum degree.

The precise running time of this setup procedure clearly depends on the data structures employed, but it is clearly polynomial. Maintaining a list of vertices of each degree, and the neighbors of each vertex, and storing only the changes at each step rather than the new graph, the time can be limited to $O(n+m)$ in the RAM model (where the length of an integer's binary representation is ignored).

Solving the Tree of csps The sequence of graphs, along with another sequence specifying one binary value for each type-III reduction, determines a sequence of CSPs; the collection of all 2^r binary sequences (where r is the number of type-III reductions) naturally defines a tree of CSPs, having depth i (we generate a child even for type-I and -II reductions) and 2^r leaves (each type-III reduction producing 2 children for each CSP in the current generation). Given an optimal solution to a CSP's child/children, an optimal solution to the CSP can be found by trying both extensions to the vertex "y", in time $O(\deg(y))$.

Starting from the leaf problems, and propagating their solutions upwards, solves the original problem.

Analysis The foregoing procedure runs in time $O(m+n)2^r$. Moreover, the tree can be stored and traversed implicitly, as a path with nodes corresponding to the graph reductions, and at each type-III node a state corresponding to which of the two reductions is currently being explored, yielding a space bound of $O(m+n)$. Thus we have the following lemma.

Lemma 4. *Given a weighted* MAX 2-CSP *whose underlying graph G is connected, and an order on the vertices of G, Algorithm A returns an optimal solution in time $O(m+n)2^r$ and space $O(m+n)$, where $r(G)$ is the (order-dependent) number of type-III reductions taken for G.*

3 Parametric Complexity

The following theorem bounds the running time of Algorithm A in terms of parameters of the graph underlying the CSP.

Theorem 5. *Given a weighted* MAX 2-CSP *whose underlying graph G is connected, has order n, size m, and excess $\kappa = m - n$, Algorithm A returns an optimal solution in time $O(m+n)2^{\min\{m/5,\kappa/2\}}$.*

We remark that to prove our expected-time result (Theorem 3), we use only the $2^{\kappa/2}$ bound. However, the $2^{m/5}O(m+n)$ bound, for arbitrary MAX 2-CSPs, is of independent interest. For MAX CUT it improves on the $2^{m/4}\,\mathrm{poly}(m+n)$ of [KF02], and for MAX 2-SAT it matches the $2^{m/5}\,\mathrm{poly}(m+n)$ bound of [GHNR] (which also gave a $2^{m/3}\,\mathrm{poly}(m+n)$ bound for MAX CUT). These works also used algorithms based on reductions.

In light of Lemma 4, it suffices to prove that (for any order on the vertices of G), the number of type-III reduction steps $r(G)$ is bounded by both $m/5$ and $\kappa/2$. These two claims are proved in the following two subsections.

3.1 Bounding in Terms of Excess

Claim 6. *For a connected graph G with excess κ, the number of type-III reduction steps of Algorithm A is $r \le \max\{0, \kappa/2\}$.*

Proof. The proof is by induction on the order of G. If G has excess 0 (it is unicyclic) or excess -1 (it is a tree), then type-I and -II reductions destroy all its edges, so $r = 0$.

Otherwise, the first type-III reduction reduces the number of edges by at least 3 and the number of vertices by exactly 1, thus reducing the excess to $\kappa' \leq \kappa - 2$. If G' has components $G'_1, \ldots, G'_I$, then $r(G) = 1 + \sum_i r(G'_i)$. Given that we applied a type-III reduction, G had minimum degree ≥ 3, so G' has minimum degree ≥ 2. Thus each component G'_i has minimum degree ≥ 2, and so excess $\kappa'_i \geq 0$. Then, by induction, $r(G) = 1 + \sum_i r(G'_i) \leq 1 + \sum_i \kappa'_i/2 \leq 1 + \kappa'/2 \leq \kappa/2$. Note that the inductive step $r(G'_i) \leq \kappa'_i/2$ used the fact that $\kappa'_i \geq 0$. □

3.2 Bounding in Terms of Size

Claim 7. *For a graph G with m edges, the number of type-III reduction steps of Algorithm A is at most $m/5$.*

Proof. Since type-I and type-II steps cannot increase the number of edges, it is enough to show that each type-III step, on average, reduces the number of edges by 5 or more. As long as the maximum degree is $d \geq 5$ this is clear, since each type-III reduction immediately destroys d edges. Thus it suffices to consider graphs of maximum degree $d \leq 4$; since the reductions never increase the degree of any vertex, the maximum degree will then remain at most 4.

Given a graph of maximum degree at most 4, suppose that Algorithm A performs r type-III reduction steps, consisting of r_3 reductions on vertices of degree 3, and r_4^k reductions on vertices of degree 4 having k neighbors of degree 3 and $r - k$ neighbors of degree 4. (If a neighbor had degree more than 4 we should have chosen it in preference to y; degree 2 or less and we should have applied a type-I or -II reduction instead.)

How many edges are destroyed by the $r = r_3 + \sum_{k=0}^{r} r_4^k$ type-III reductions? Each "r_3-reduction" deletes the 3 edges incident on y, each of which went to a vertex also of degree 3 (4 or more and we would have chosen it in preference to y, 2 or less and we would have applied a type-I or -II reduction), changing their degrees to 2 and subjecting each to a type-II reduction, and so destroying 3 more edges. (A type-II reduction destroys edges yx and yz, and if edge xz was not previously present it creates it, thus reducing the number of edges by at least 1, and possibly 2.) Similarly, each "r_4^k reduction", on a degree-4 vertex adjacent to k degree-3 vertices, along with the k type-II reductions it sets up, destroys $4 + k$ edges. Thus the average number of edges destroyed per step is at least

$$\frac{6r_3 + \sum_{k=0}^{4}(4+k)r_4^k}{r_3 + \sum_{k=0}^{4} r_4^k}. \tag{1}$$

Clearly this ratio is at least 5 unless the value of r_4^0 can be made large, but we now show that the r_4^k values must satisfy an additional condition which effectively prohibits this.

Note that each r_3-reduction decreases the number of degree-3 vertices by 4 (itself and its 3 neighbors), while each r_4^k-reduction decreases it by $2k-4$ (destroying k degree-3 neighbors, but also turning $4-k$ old degree-4 neighbors into new degree-3 vertices). Type-I and -II reductions do not affect the number of degree-3 vertices. Since the number of degree-3 vertices is initially non-negative, and finally 0, the decrease must be non-negative, i.e.,

$$\sum_k r_4^k(2k-4) + 4r_3 \geq 0. \tag{2}$$

Subject to the constraint given by (2), how small can the ratio (1) be? To be (slightly) pessimistic, we may let the values r_3 and r_4^k range over the non-negative reals. Multiplying the set of values by any constant affects neither the constraint nor the ratio, so without loss of generality we may set the denominator of (1) to 1. That is, we add a constraint

$$r_3 + \sum r_4^k = 1, \tag{3}$$

and minimize

$$6r_3 + \sum_{k=0}^{4}(4+k)r_4^k. \tag{4}$$

This is simply a linear program (LP) with objective function (4) and the two constraints (2) and (3). The LP's optimal objective value is 5, and the LP dual solution of $(\frac{1}{4}, 5)$ establishes 5 as a lower bound. That is, adding $\frac{1}{4}$ times constraint (2) to 5 times constraint (3) gives

$$\frac{1}{4}\left(\sum(2k-4)r_4^k + 4r_3\right) + 5\left(r_3 + \sum r_4^k\right) = 6r_3 + \sum(4+k/2)r_4^k \geq 5,$$

so (4), which is $6r_3 + \sum(4+k)r_4^k$, must be at least this large.

This establishes that the number of edges destroyed by type-III reductions is at least 5 times the number of such reductions, concluding the proof. □

We note that the upper bound of $m/5$ is achievable; that is, $m/5$ type-III reductions are needed by some graphs. An easy example is K_5, with 10 edges, reduced by two type-III reductions to K_4 and K_3, the latter reduced to the empty graph by type-I and -II reductions.

4 Stochastic Size and Excess of a Random Graph

We stochastically bound the excess κ of a component of a random graph G through a standard "exposure" process. Given a graph G and a vertex x_1 in G, together with a linear order on the vertices of G, the exposure process finds a spanning tree of the component G_1 of G that contains x_1 and, in addition, counts the number of non-tree edges of G_1 (i.e., calculates the excess).

At each step of the process, vertices are classified as "living", "dead", and "unexplored", beginning with just x_1 living, and all other vertices unexplored. At the ith step, the process takes the earliest living vertex x_i. All edges from x_i to unexplored vertices are added to the spanning tree, and the number of non-tree edges is increased by 1 for each edge from x_i to a living vertex. Unexplored vertices adjacent to x_i are then reclassified as living, and x_i is made dead. The process terminates when there are no live vertices.

Now suppose G is a random graph in $\mathcal{G}(n, c/n)$, with the vertices ordered at random. Let $w(i)$ be the number of live vertices at the ith step and define the *width* $w = \max w(i)$. Let $u = |G_1|$, so that $w(0) = 1$ and $w(u) = 0$. The number of non-tree edges uncovered in the ith step is binomially distributed as $B(w(i) - 1, c/n)$, and so, conditioning on u and $w(1), \dots, w(u)$, the number of excess edges is distributed as $B(\sum_{i=1}^{u}(w(i)-1), c/n)$. Since $\sum_{i=1}^{u}(w(i)-1) \leq uw$, the (conditioned, and therefore also the unconditioned) number of excess edges is dominated by the random variable $B(uw, c/n)$.

At the ith stage of the process, there are at most $n - i$ unexplored vertices, and so the number of new live vertices is dominated by $B(n - i, 1/n)$. Consider now a variant of the exposure process in which at each step we add enough special "red" vertices to bring the number of unexplored vertices to $n - i$. Let $h(i)$ be the number of living vertices at the ith stage. Then $h(0) = 1$, and $h(i)$ is distributed as $h(i-1) + B(n-i, c/n) - 1$. Let $X = n \wedge \min\{t : h(t) = 0\}$ and $H = \max_{i \leq X} h(i)$.

By considering the second process as an extension of the first (and exploring the added vertices in the second process only when no other vertices remain), we obtain a coupling between the two processes such that $u \leq X$ and $w \leq H$. Thus the excess of G_1 is dominated by $B(XH, 1/n)$.

Since the running time of Algorithm A is at most $\mathbb{E}(O(m + n)2^{\kappa/2})$, it can be bounded by the quantity $O(n^2)\mathbb{E}(\sqrt{2}^{(B(XH,1/n)})$. It is useful to note that

$$\mathbb{E}z^{B(n,p)} = \sum_{i=0}^{n} \binom{n}{i} z^i p^i (1-p)^{n-i} = (pz + (1-p))^n = (1 + p(z-1))^n \leq \exp(p(z-1)n).$$

In particular, $\mathbb{E}\sqrt{2}^{B(n,p)} \leq \exp((\sqrt{2}-1)np)$. In the following, we therefore focus on bounding quantities of form $\Pr(X = x, H = h)\exp((\sqrt{2}-1)xh/n)$.

Lemma 8. *With $h(t)$ the random process defined above, for all times $i = 1, 2, \dots$ parametrized as $\alpha n = i$,*

$$\Pr(h(\alpha n) \geq 0) \leq \exp\left(-3\alpha^3 n/(24 - 8\alpha)\right). \tag{5}$$

Furthermore, for any height h parametrized as $h = \beta n$, with $\alpha^2/(8-4\alpha) \leq \beta \leq \alpha$,

$$\Pr(\max_{t \leq \alpha n} h(t) \geq \beta n \mid h(\alpha n) = 0) \leq O(n^{3/2}) \exp\left(-\left(\beta - \frac{\alpha^2/4}{2-\alpha}\right)^2 \frac{7n}{8\alpha}\right). \tag{6}$$

In order to prove the lemma, we shall make use of the following fairly standard bound.

Claim 9. *With $N = ni - \binom{i+1}{2}$, let $Z_1, Z_2, \dots, Z_N$, be a random sequence of binomial random variables conditioned upon $\sum_{j=1}^{N} Z_j = i - 1$. Parametrize $i = \alpha n$. Suppose that β is in the range $\alpha^2/(8-4\alpha) \le \beta \le \alpha$, and $t \le i$. Then, writing $N' = nt - \binom{t+1}{2}$,*

$$\Pr(\sum_{i=1}^{N'} Z_i \ge \beta n + (t-1)) \le O(\sqrt{n}) \exp\left(-\left(\beta - \frac{\alpha^2}{8-4\alpha}\right)^2 \frac{7n}{8\alpha}\right). \tag{7}$$

We omit the proof.

Proof (of Lemma 8). We first prove (5). Note that

$$h(i) = B\left((n-1) + \cdots + (n-i), 1/n\right) - i + 1 = B\left(ni - \binom{i+1}{2}, 1/n\right) - i + 1$$

and so $h(i) \ge 0$ means that

$$B\left(ni - \binom{i+1}{2}, 1/n\right) \ge i + 1 = \alpha n + 1. \tag{8}$$

This binomial r.v. has expectation

$$\left(\alpha n^2 - \binom{\alpha n + 1}{2}\right) \frac{1}{n} \le (\alpha - \alpha^2/2)n. \tag{9}$$

Thus if (8) holds, the r.v. differs from its expectation by at least $\alpha^2 n/2$.

We use the inequality that for a sum of independent 0-1 Bernoulli random variables with parameters $p_1, \dots, p_n$ and expectation $\mu = \sum_{i=1}^{n} p_i$, $\mathbb{P}(X \ge \mu + t) \le \exp\left(-t^2/(2\mu + 2t/3)\right)$. Together with (9) this implies that (8) has probability at most $\exp\left(-(\alpha^4 n^2/4)/(2\alpha n(1-\alpha/2) + \alpha^2 n/3)\right) = \exp\left(-3\alpha^3 n/(24 - 8\alpha)\right)$.

To prove (6), we bound the conditional probability

$$\Pr(\max_{t \le \alpha n} h(t) \ge \beta n \mid h(\alpha n) = 0). \tag{10}$$

In this part, rather than thinking of $h(i)$ as $B(ni - \binom{i+1}{2}, 1/n) - i + 1$, we think of it as a sum of $N = ni - \binom{i+1}{2}$ independent Bernoulli random variables Z_i each with distribution $B(1/n)$, plus $-i+1$. Note that, conditional on the sum of the Z_is, any particular assignment of 0s and 1s is equally likely: the collection of Z_is is a random binomial sequence conditioned upon $h(\alpha n) = 0$, i.e., upon having sum $\alpha n - 1$. We apply Claim 9 to show that for any given t, the probability of each of the events comprising that in (10) is bounded by (7), namely $\Pr(h(t) \ge \beta n \mid h(\alpha n) = 0) \le O(\sqrt{n}) \exp\left(-\left(\beta - \frac{\alpha^2}{8-4\alpha}\right)^2 \frac{7n}{8\alpha}\right)$. Summing over $1 \le t = \gamma n \le \alpha n$, the required bound (6) follows. □

Recall the random process h defined before Lemma 8, with stopping time X and maximum height H.

Lemma 10.
$$\mathbb{E}\left[\exp\left((\sqrt{2}-1)XH/n\right)\right] \leq n^{9/2}.$$

Proof. We show that each possible pair $X \in \{1, \ldots, n-1\}$ and $H \in \{1, \ldots, \frac{1}{2}n^2 + O(1)\}$ contributes at most $O(n^{3/2})$ to the expectation. Specifically, we show that for all α and β, $\exp\left((\sqrt{2}-1)\alpha\beta n\right)\Pr(X = \alpha n)\Pr(Y = \beta n) = O(n^{3/2})$.

Case 1. If $\beta < \alpha^2/(8-4\alpha)$ then, from Lemma 8,
$$\Pr(X = \alpha n) \leq \Pr(h(\alpha n) = 0) \leq \Pr(h(\alpha n) \geq 0) \leq \exp\left(-3\alpha^3 n/(24-8\alpha)\right) \tag{11}$$
and so
$$\exp\left((\sqrt{2}-1)\alpha\beta n\right)\Pr(X = \alpha n) \leq \exp\left((\sqrt{2}-1)\frac{\alpha^3 n}{8-4\alpha} - \frac{3\alpha^3 n}{24-8\alpha}\right).$$
This is less than 1 provided that
$$\frac{\sqrt{2}-1}{8-4\alpha} \leq \frac{3}{24-8\alpha},$$
which is easily verified to hold for all $\alpha \in [0,1]$.

Case 2. If $\beta \geq \alpha^2/(8-4\alpha)$ then, from Lemma 8, in addition to (11), we have that
$$\begin{aligned}\Pr(H = \beta n \mid X = \alpha n) &\leq \Pr(H \geq \beta n \mid X = \alpha n)\\ &\leq O(n^{3/2})\exp\left(-\left(\beta - \frac{\alpha^2/4}{2-\alpha}\right)^2 \frac{7n}{8\alpha}\right).\end{aligned}$$

So in this case it suffices to show that
$$\exp\left(\left[(\sqrt{2}-1)\alpha\beta n\right] - \left[3\alpha^3 n/(24-8\alpha)\right] - \left[\left(\beta - \frac{\alpha^2/4}{2-\alpha}\right)^2 \frac{7n}{8\alpha}\right]\right) \leq 1, \tag{12}$$
i.e., that
$$\left[(\sqrt{2}-1)\alpha\beta\right] - \left[3\alpha^3/(24-8\alpha)\right] - \left[\left(\beta - \frac{\alpha^2/4}{2-\alpha}\right)^2 \frac{7}{8\alpha}\right] \tag{13}$$
is at most 0.

For fixed $a \in (0,1]$, (13) is maximized by
$$\beta = \frac{4}{7}(\sqrt{2}-1)\alpha^2 + \frac{\alpha^2}{8-4\alpha}.$$

Substituting this value of β into (13), and multiplying by the (positive) quantity $(\alpha-2)(\alpha-3)/\alpha^3$ gives a quadratic which is easily seen to be negative on $(0,1]$.

Thus, in both Case 1 and Case 2, for any α and β, the contribution of the $X=\alpha n$, $H=\beta n$ term to the expectation of $(\sqrt{2}-1)^{XH/n}$ is at most $O(n^{3/2})$, and the sum of all $O(n^3)$ such contributions (recalling that X and H may take on $O(n)$ and $O(n^2)$ possible values, respectively) is $O(n^{9/2})$. □

We can now prove Theorem 3.

Proof (of Theorem 3). By Theorem 5, and the remarks before Lemma 8, Algorithm A runs in expected time $\mathbb{E}(O(m+n)\sqrt{2}^{\kappa} \leq O(n^2)\mathbb{E}(\sqrt{2}^{B(XH)}) \leq O(n^2)\mathbb{E}(\exp((\sqrt{2}-1)XH/n))$. But it follows from Lemma 10 that this is $O(n^{13/2})$. □

5 Conclusions

In the present paper we focus on MAX CUT. Our result for "sparse" instances is strong in that it applies right up to $c=1$, and we expect it could be extended through the scaling window, to $c=1+\lambda n^{-1/3}$ (at the expense of a constant factor depending on λ in the run time, and additional complication in the analysis). We also believe that our methods can be extended to MAX 2-SAT, but the analysis is certainly more complicated. In fact our results already apply to any MAX CSP, and in particular to MAX 2-SAT, but only in the regime where there are about $n/2$ clauses on n variables; since it is likely that random instances with up to about n clauses can be solved efficiently on average (the 2-SAT phase transition occurs around n clauses), our present result for MAX 2-SAT is relatively weak.

Since MAX CUT is in general NP-hard (and even NP-hard to approximate to better than a 16/17 factor [TSSW00]), it would be interesting to resolve whether dense instances of MAX CUT as well as sparse ones can be solved in polynomial expected time (thus separating the average-case hardness from the worst-case hardness) or whether random dense instances are hard. Precisely the same questions can be asked about MAX 2-SAT, and in both cases we would guess that dense instances are hard, even on average.

References

[BBC+01] Béla Bollobás, Christian Borgs, Jennifer T. Chayes, Jeong Han Kim, and David B. Wilson, *The scaling window of the 2-SAT transition*, Random Structures Algorithms **18** (2001), no. 3, 201–256.

[Bol01] Béla Bollobás, *Random graphs*, Cambridge Studies in Advanced Mathematics, vol. 73, Cambridge University Press, Cambridge, 2001.

[CGHS] Don Coppersmith, David Gamarnik, Mohammad Hajiaghayi, and Gregory B. Sorkin, *Random* MAX SAT*, random* MAX CUT*, and their phase transitions*, Submitted for publication. 49 pages.

[CGHS03] Don Coppersmith, David Gamarnik, Mohammad Hajiaghayi, and Gregory B. Sorkin, *Random* MAX SAT*, random* MAX CUT*, and their phase transitions*, Proceedings of the 14th Annual ACM–SIAM Symposium on Discrete Algorithms (Baltimore, MD, 2003), ACM, New York, 2003.

[COMS] Amin Coja-Oghlan, C. Moore, and V. Sanwalani, *Max k-cut and approximating the chromatic number of random graphs*, To appear.

[CR92] Vasěk Chvátal and Bruce Reed, *Mick gets some (the odds are on his side)*, 33th Annual Symposium on Foundations of Computer Science (Pittsburgh, PA, 1992), IEEE Comput. Soc. Press, Los Alamitos, CA, 1992, pp. 620–627.

[FdlV92] Wenceslas Fernandez de la Vega, *On random* 2*-SAT*, Manuscript, 1992.

[GHNR] Jens Gramm, Edward A. Hirsch, Rolf Niedermeier, and Peter Rossmanith, *New worst-case upper bounds for MAX-2-SAT with an application to MAX-CUT*, Discrete Applied Mathematics, In Press.

[Goe96] Andreas Goerdt, *A threshold for unsatisfiability*, J. Comput. System Sci. **53** (1996), no. 3, 469–486.

[KF02] A. S. Kulikov and S. S. Fedin, *Solution of the maximum cut problem in time* $2^{|E|/4}$, Zap. Nauchn. Sem. S.-Peterburg. Otdel. Mat. Inst. Steklov. (POMI) **293** (2002), no. Teor. Slozhn. Vychisl. 7, 129–138, 183.

[KV02] Michael Krivelevich and Van H. Vu, *Approximating the independence number and the chromatic number in expected polynomial time*, J. Comb. Optim. **6** (2002), no. 2, 143–155.

[TCO03] Anusch Taraz and Amin Coja-Oghlan, *Colouring random graphs in expected polynomial time*, Proceedings of STACS 2003, LNCS 2607, 2003, pp. 487–498.

[TSSW00] Luca Trevisan, Gregory B. Sorkin, Madhu Sudan, and David P. Williamson, *Gadgets, approximation, and linear programming*, SIAM J. Comput. **29** (2000), no. 6, 2074–2097.