Combinatorics, Probability and Computing (2006) 15, 281–315. © 2006 Cambridge University Press
doi:10.1017/S096354830500725X Printed in the United Kingdom

Solving Sparse Random Instances of Max Cut and Max 2-CSP in Linear Expected Time

ALEXANDER D. SCOTT[1] and GREGORY B. SORKIN[2]

[1]Department of Mathematics, University College London, London WC1E 6BT, UK
(e-mail: scott@math.ucl.ac.uk)
[2]Department of Mathematical Sciences, IBM T.J. Watson Research Center,
Yorktown Heights NY 10598, USA
(e-mail: sorkin@watson.ibm.com)

Received 19 September 2004; revised 7 August 2005

For Béla Bollobás on his 60th birthday

We show that a maximum cut of a random graph below the giant-component threshold can be found in linear space and linear expected time by a simple algorithm. In fact, the algorithm solves a more general class of problems, namely binary 2-variable constraint satisfaction problems. In addition to Max Cut, such Max 2-CSPs encompass Max Dicut, Max 2-Lin, Max 2-Sat, Max-Ones-2-Sat, maximum independent set, and minimum vertex cover. We show that if a Max 2-CSP instance has an 'underlying' graph which is a random graph $G \in \mathcal{G}(n, c/n)$, then the instance is solved in linear expected time if $c \leqslant 1$. Moreover, for arbitrary values (or functions) $c > 1$ an instance is solved in expected time $n \exp(O(1 + (c-1)^3 n))$; in the 'scaling window' $c = 1 + \lambda n^{-1/3}$ with λ fixed, this expected time remains linear.

Our method is to show, first, that if a Max 2-CSP has a connected underlying graph with n vertices and m edges, then $O(n2^{(m-n)/2})$ is a deterministic upper bound on the solution time. Then, analysing the tails of the distribution of this quantity for a component of a random graph yields our result. Towards this end we derive some useful properties of binomial distributions and simple random walks.

1. Introduction

In this paper we prove that a maximum cut of a random graph below the giant-component threshold can be found in linear expected time.

Theorem 1.1. *For any $c \leqslant 1$, a maximum cut of a random graph $G \in \mathcal{G}(n, c/n)$ can be found in time whose expectation is $O(n)$, using space $O(m+n)$, where m is the size of the graph.*

We should point out the significance of requiring linear time 'in expectation' rather than just 'almost always'. With high probability, a random graph below the giant-component

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

threshold consists solely of trees and unicyclic components, and a maximum cut in such a graph is easy to find. (It cuts all edges except for one edge in each odd cycle.) However, exponential time can be spent on finding optimal cuts in the rare multicyclic graphs, which is what makes the proof of Theorem 1.1 difficult.

Our approach is, first, to give a deterministic algorithm and an upper bound on its running time as a function of the input graph's 'excess' of edges over vertices, $m - n$.

Theorem 1.2. *Let G be a connected graph with n vertices and m edges. A maximum cut of G can be found in time $O(n2^{(m-n)/2})$, using space $O(m+n)$.*

We then bound the distribution of the excess in a component of a sparse random graph. This enables us to bound the *expected* running time of our algorithm, and hence prove Theorem 1.1.

In fact, Theorems 1.1 and 1.2 are special cases of more general results (Theorems 7.1 and 3.3). Our algorithm employs local reductions that take us outside the class of Max Cut problems, forcing us to work with the larger class Max 2-CSP (defined and discussed in Section 2). Working in this broader class both simplifies our methods and means that our results apply not just to Max Cut but also to problems including Mix Dicut, Max 2-Lin, Max 2-Sat, Max-Ones-2-Sat, maximum independent set, minimum vertex cover, and weighted versions of these problems.

Throughout the paper, n and m are reserved for the number of vertices and edges of a graph G. By $G \in \mathscr{G}(n,p)$ as usual we denote a random graph with n vertices, where each potential edge is present with probability p, independently; we also write $G(n,p)$ as shorthand for such a graph.

1.1. Context

Our results are particularly interesting in the context of phase transitions for various maximum constraint-satisfaction problems. Since we are just situating our results, 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 $n \to \infty$; see Chvátal and Reed [8], Goerdt [16], and Fernandez de la Vega [14]. Indeed there is now a detailed picture of the scaling window; see Bollobás, Borgs, Chayes, Kim and Wilson [7]. Max 2-Sat has since been shown to exhibit similar behaviour, so for $c < 1$, only an expected $\Theta(1/n)$ clauses go unsatisfied, while for $c > 1$, an expected $\Theta(n)$ clauses must go unsatisfied; see Coppersmith, Gamarnik, Hajiaghayi and Sorkin [11].

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 $\Theta(n)$ (see for example Bollobás [6]). Again, [11] 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 $\Theta(n)$.

For both Max 2-Sat and Max Cut, it seems likely that the mostly satisfiable (or mostly cuttable) sparse instances are algorithmically easy, while the not-so-satisfiable

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

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 linear expected time; see Section 8, Conclusions, for further discussion.

More generally, our interest here is in solving random instances of hard problems in polynomial expected time, and of course there is a substantial body of literature on this subject. For example, results on colouring random graphs in polynomial expected time can be found in Krivelevich and Vu [23], Coja-Oghlan, Moore and Sanwalini [9], and Coja-Oghlan and Taraz [10].

As already remarked, our expected-linear-time result comes from analysing an algorithm which, for arbitrary connected graphs, runs in time $O(n2^{(m-n)/2})$, deterministically. This parametrization in terms of $m - n$ is efficient for random graphs up to the giant-component threshold or even slightly beyond, because a random graph $G(n, (1 + n^{-1/3})/n)$ typically has a giant component with $\Theta(n^{2/3})$ vertices and a similar number of edges, but excess only $\Theta(1)$. In a paper with a preliminary version of the present result [28], we also showed that, for any Max 2-CSP instance, the same Max 2-CSP algorithm has running time $\tilde{O}(2^{m/5})$, where the $\tilde{O}$ notation hides polynomial factors. Previously, Niedermeier and Rossmanith showed that Max 2-Sat could be solved in time $\tilde{O}(2^{0.347m})$ [27]; Hirsch improved this to $\tilde{O}(2^{m/4})$ [18]; Gramm, Hirsch, Niedermeier and Rossmanith improved it to $\tilde{O}(2^{m/5})$ and adapted it to solve Max Cut in time $\tilde{O}(2^{m/3})$ [17]; Fedin and Kulikov improved the Max Cut result to $\tilde{O}(2^{m/5})$ [24]; and in a forthcoming paper we refine our present algorithm and analysis to obtain time $\tilde{O}(2^{19m/100})$ for any Max 2-CSP [29].

1.2. Outline of proof

Our main result will be Theorem 1.1, generalized from $c \leqslant 1$ to a larger range, and from Max Cut to the class Max 2-CSP (to be defined in Section 2). Its proof has a few principal components. Since the maximum cut of a graph is the combination of maximum cuts of each of its connected components (and the same is true for any Max 2-CSP), it suffices to bound the expected time the algorithm spends on the component containing a fixed vertex.

In order to bound the expected running time of 'Algorithm A' (introduced in Section 3) we must control the distribution of the excess of a component of a random graph. This is done by 'exploring' the component as a branching process, dominating it with a similar process, and analysing the latter as a random walk. We obtain stochastic bounds on the component order u and, conditioned upon u, the 'width' w (defined later), and finally the excess, which is dominated by a binomial random variable $B(uw, c/n)$.

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 linear expected time.

We could also have tried to bound the expected running time of Algorithm A by characterizing the excess using estimates for $C(n, n + k)$, the number of connected graphs on n vertices with $n + k$ edges. Such estimates are given by Bollobás [5], Łuczak [25] and Bender, Canfield and McKay [3], but they seem not to be immediately suitable for our purposes; we discuss this more extensively in Section 6.

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

2. Max 2-CSP

The problem Max Cut is to partition the vertices of a given graph into two classes so as to maximize the number of edges 'cut' by the partition. Think of each edge as being a function on the classes or 'colours' of its endpoints, with value 1 if the endpoints are of different colours, 0 if they are the same: Max Cut is equivalent to finding a 2-colouring of the vertices which maximizes the sum of these edge functions. This view naturally suggests a generalization.

An *instance* (G, S) of Max 2-CSP is given by an 'underlying' graph $G = (V, E)$ and a set S of 'score' functions. Writing $\{R, B\}$ for the colours Red and Blue, for each edge $e \in E$ there is a 'dyadic' score function $s_e : \{R, B\}^2 \to \mathbb{R}$, for each vertex $v \in V$ there is a 'monadic' score function $s_v : \{R, B\} \to \mathbb{R}$, and finally there is a single 'niladic' score function $s_0 : \{R, B\}^0 \to \mathbb{R}$ which takes no arguments and is just a constant convenient for bookkeeping. We allow an instance to have parallel edges (and further such edges may be generated while the algorithm runs).

A potential *solution* is a 'colouring' of the vertices, *i.e.*, a function $\phi : V \to \{R, B\}$, and an optimum solution is one which maximizes

$$s(\phi) := s_0 + \sum_{v \in V} s_v(\phi(v)) + \sum_{uv \in E} s_{uv}(\phi(u), \phi(v)). \tag{2.1}$$

Without belabouring the notation for edges, we wish to take each edge just once, and (since s_{uv} need not be a symmetric function) with a fixed notion of which endpoint is 'u' and which is 'v'. We will typically assume that $V = [n]$ and any edge uv is really an ordered pair (u, v) with $1 \leqslant u < v \leqslant n$. We remark that the '2' in the name Max 2-CSP refers to the fact that the score functions take 2 or fewer arguments (3-Sat, for example, is out of scope); replacing 2 by a larger value would mean replacing the underlying graph with a hypergraph.

Our assumption of an undirected underlying graph is sound even for a problem such as Max Dicut (maximum directed cut). Here one normally thinks of a directed edge as cut only if its head has colour 0 and its tail has colour 1, but for a directed edge (v, u) with $v > u$ this may be expressed by the undirected (or, equivalently, canonically directed) edge (u, v) with score 1 if $(\phi(u), \phi(v)) = (1, 0)$, and score 0 otherwise. That is, instead of directing the *edges* we incorporate the direction into the score functions. (In cases like this we do not mention the monadic and niladic score functions; they are 'unused', *i.e.*, taken to be identically 0.)

An obvious computational-complexity issue is raised by allowing scores to be arbitrary *real* values. Our algorithm will add, subtract, and compare these values (never introducing an absolute value larger than the sum of those in the input) and we assume that each such operation can be done in time $O(1)$ and its result represented in space $O(1)$. If desired, scores may be limited to integers, and the length of the integers factored in to the algorithm's complexity, but this seems uninteresting and we will not remark on it further.

We can solve minimization problems by replacing each score function with its negation (there is no assumption of positivity) and solving the resulting maximization problem. Max 2-CSP also models *weighted* problems: assigning a weight to a constraint just means multiplying the score function by the weight. Generalization to problems like

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

Max-Ones-2-Sat can be achieved by adding, to the usual 2-Sat formulation, small monadic cost functions $s_v(\phi(v)) = \epsilon\phi(v)$ (for instance, $\epsilon = 1/2n$ will do); this rewards setting variables to 1, but not at the expense of satisfying even a single clause.

Max 2-CSP further includes problems that are not obviously structured around pairwise constraints. Our original example of Max Cut may fall into this category, as do maximum independent set and minimum vertex cover (a minimum set of vertices dominating all edges). To model the problem of finding a maximum independent set in a graph as a Max 2-CSP, let $\phi(v) = 1$ if vertex is to be included in the set and 0 otherwise, define vertex scores $s_v(\phi(v)) = \phi(v)$ (rewarding a vertex for being included in the set), and define edge scores $s_{uv}(\phi(u), \phi(v)) = -2$ if $\phi(u) = \phi(v) = 1$, and 0 otherwise (penalizing violations of independence, and outweighing the reward for inclusion). Similarly, for minimum dominating set we penalize vertices for inclusion, but more heavily penalize edges neither of whose endpoints is included.

3. Solving a maximum constraint-satisfaction instance

In this section we describe our algorithm, Algorithm A, and analyse its performance on an arbitrary Max 2-CSP instance. The implications for random instances are taken up in subsequent sections.

The algorithm uses three types of reductions and an additional, trivial 'pseudo-reduction'. 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 instance. This sequence defines a tree of instances, which can be solved bottom-up to solve the original one. Finally, we bound the algorithm's time and space requirements.

3.1. Reductions

Our first two reductions are 'transformations', each producing an equivalent problem with fewer vertices; the third is a 'splitting rule' producing 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 neighbour 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

$$s'_x(C) = s_x(C) + \max_D\{s_{xy}(C, D) + s_y(D)\},$$

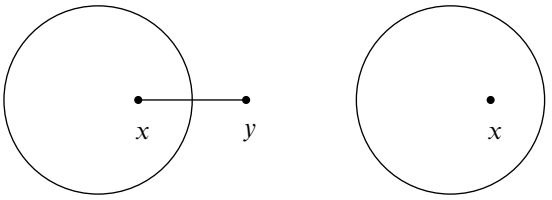
i.e., we set

$$s'_x(R) = s_x(R) + \max\{s_{xy}(R, R) + s_y(R), s_{xy}(R, B) + s_y(B)\},$$
$$s'_x(B) = s_x(B) + \max\{s_{xy}(B, R) + s_y(R), s_{xy}(B, B) + s_y(B)\}.$$

Note that any colouring ϕ' of V' can be extended to a colouring of V in two ways, namely ϕ_R and ϕ_B (corresponding to the two colourings of y), and the defining property of the reduction is that $S'(\phi') = \max\{S(\phi_R), S(\phi_B)\}$. In particular, $\max_{\phi'} S'(\phi') = \max_\phi S(\phi)$,

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

and an optimal colouring ϕ' for the instance (V', E', S') can be extended to an optimal colouring ϕ for (V, E, S).



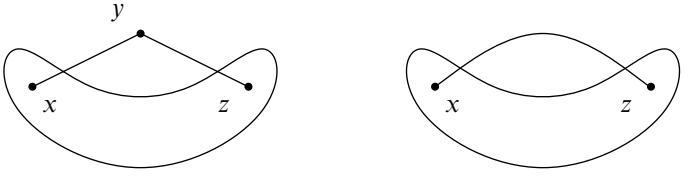
Reduction II. Let y be a vertex of degree 2, with neighbours x and z. If $x = z$ we have a pair of parallel edges: we combine the two edges and perform a type I reduction. Otherwise, reducing (V, E, S) on y results in a new problem (V', E', S') with $V' = V \setminus y$ and $E' = (E \setminus \{xy, yz\}) \cup \{xz\}$. S' is the restriction of S to V' and E', except that for $C, D, E \in \{R, B\}$ we set

$$s'_{xz}(C, D) = \max_E \{s_{xy}(C, E) + s_{yz}(E, D) + s_y(E)\},$$

i.e., we set

$$\begin{aligned}
s'_{xz}(R, R) &= \max\{s_{xy}(R, R) + s_{yz}(R, R) + s_y(R), s_{xy}(R, B) + s_{yz}(B, R) + s_y(B)\},\\
s'_{xz}(R, B) &= \max\{s_{xy}(R, R) + s_{yz}(R, B) + s_y(R), s_{xy}(R, B) + s_{yz}(B, B) + s_y(B)\},\\
s'_{xz}(B, R) &= \max\{s_{xy}(B, R) + s_{yz}(R, R) + s_y(R), s_{xy}(B, B) + s_{yz}(B, R) + s_y(B)\},\\
s'_{xz}(B, B) &= \max\{s_{xy}(B, R) + s_{yz}(R, B) + s_y(R), s_{xy}(B, B) + s_{yz}(B, B) + s_y(B)\}.
\end{aligned}$$

This reduction creates a new edge xy, which may be parallel to one or more existing edges, each such edge having its associated score function. (The only reason we do not immediately merge parallel edges is that, working within our linear-time constraint, there is not time to identify them! Unfortunately our notation fails to distinguish among parallel edges and their scores, but this is only to keep the notation manageable; there is no deeper issue.) As in Reduction I, any colouring 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 colouring ϕ' for the instance (V', E', S') can be extended to an optimal colouring ϕ for (V, E, S).



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, S), to (V', E', S^R) and (V', E', S^B), corresponding to assigning the colour R or B to y.

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

We define $V' = V \setminus y$, and E' as the restriction of E to V'. For $C, D \in \{R, B\}$, 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 neighbour x of y,

$$(s^C)_x(D) = s_x(D) + s_{xy}(D, C).$$

In other words, S^R is the restriction of S to $V \setminus y$, except that we set $(s^R)_0 = s_0 + s_y(R)$ and, for every neighbour x of y,

$$(s^R)_x(R) = s_x(R) + s_{xy}(R, R)$$
$$(s^R)_x(B) = s_x(B) + s_{xy}(B, R).$$

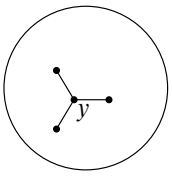
Similarly S^B is given by $(s^B)_0 = s_0 + s_y(B)$ and, for every neighbour x of y,

$$(s^B)_x(R) = s_x(R) + s_{xy}(R, B)$$
$$(s^B)_x(B) = s_x(B) + s_{xy}(B, B).$$

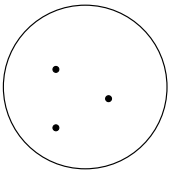
As in the previous reductions, any colouring ϕ' of $V \setminus y$ can be extended to V in two ways, ϕ_R and ϕ_B, corresponding to the colour given to y, and now (this is different!) $S^R(\phi') = S(\phi_R)$ and $S^B(\phi') = S(\phi_B)$. Furthermore,

$$\max\{\max_{\phi'} S^R(\phi'), \max_{\phi'} S^B(\phi')\} = \max_{\phi} S(\phi),$$

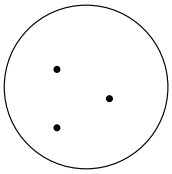
and an optimal colouring on the left can be extended to an optimal colouring on the right.



(V, E, S)



(V', E', S^R)



(V', E', S^B)

Reduction 0. We define one more 'pseudo-reduction'. If a vertex y has degree 0 (so it has no dyadic constraints), we simply delete it from the instance and incorporate its cost into the niladic score s_0. Specifically, reducing (V, E, S) on y results in a new problem (V', E', S') with $V' = V \setminus y$ and $E' = E$. S' is the restriction of S to V' and E', except that for $C \in \{R, B\}$ we set

$$s'_0 = s_0 + \max_C s_y(C).$$

As usual, $\max_{\phi'} S'(\phi') = \max_{\phi} S(\phi)$, and an optimal colouring ϕ' for (V', E', S') can be extended to an optimal colouring ϕ for (V, E, S).

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

3.2. Algorithm idea

A recursive algorithm for solving an input instance works as follows. Begin with the input problem instance. Given an instance $\mathscr{M} = (G, S)$:

(1) If any reduction of type 0, I or II is possible, apply it to reduce $\mathscr{M}$ to $\mathscr{M}'$, record certain information about the reduction, solve $\mathscr{M}'$ recursively, and use the recorded information to reverse the reduction and extend the solution to one for $\mathscr{M}$.
(2) If only a type III reduction is possible, reduce on a vertex of degree $\geqslant 4$ if one exists, a vertex of degree 3 otherwise. In either case, first recursively solve $\mathscr{M}^R$ (the 'red' version of the reduction), then solve $\mathscr{M}^B$ (the 'blue' version), select the solution with the larger score, and use the recorded information to reverse the reduction and extend the solution to one for $\mathscr{M}$.
(3) If no reduction is possible then the graph has no vertices, there is a unique colouring (the empty colouring), and the score is s_0 (from the niladic score function).

The algorithm makes two runs as above. In the first run, in step (3) save the score if it sets a new record; this returns the optimal score but not the corresponding colouring. In the second run, when the score matches the record value, stop and return the colouring, by using the information stored at the ancestor nodes of the current leaf.

That the algorithm returns an optimal solution, *i.e.*, that it is correct, follows from the definitions of the reductions. The run time is analysed in the following sections, dealing with implementation details and data structures. We shall speak in terms of a *computation tree* implicitly defined by the recursive computation. For clarity, we will speak of 'nodes' of this tree, as opposed to 'vertices' of the instance's underlying graph. The tree's root node is the original problem instance, and each node's children are the subinstances derived from reducing it; III-reducing a node produces two children and the other reductions give a single child. We will also use a *reduced tree* which collapses nodes with exactly one child, so that a series of reductions starting with any number of 0, I and II-reductions and ending with a III-reduction (or when the instance is empty) is represented by a single edge. The depth r of the reduced tree is a key parameter: Corollary 3.2 shows that an appropriate algorithm implementation runs in time $O(n2^r)$ (and space $O(m+n)$).

3.3. Implementation details and data structures

Limiting the algorithm to linear space excludes saving copies of the instance as we descend a branch of the tree; rather, when the algorithm is processing a node of the tree, the corresponding instance should be the only one explicitly maintained, while the ancestor instances should be reconstructible by compact information stored at the ancestor nodes.

In this subsection we establish the following claim.

Claim 3.1. *After linear-time preprocessing, we can do the following.*

(1) *Identify the next reduction to perform, in time* $O(1)$.
(2) *Perform a III-reduction on a vertex* y *in time* $O(n)$, *creating an* $O(\deg(y))$*-space annotation enabling the reduction to be reversed and its colouring optimally extended.*
(3) *Perform a series of 0-, I- or II-reductions corresponding to an edge of the reduced tree in time* $O(n)$, *creating an* $O(1)$*-space annotation for each individual reduction.*

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

The implementation details are unrelated to the main direction of the paper, the characterization of a random process. However, they are important since the linear-time result depends on these tight space and time bounds.

3.3.1. Data structure. We assume a RAM model, so that a given memory location can be accessed in constant time.

We presume that the input graph is given in a sparse representation, consisting of a vector of vertices, each with its monadic score function (a 2-element table) and a doubly linked list of incident edges, each edge with its dyadic score function (a 4-element table) and a pointer to the edge's twin copy indexed by the other endpoint. We also assume that there is a doubly linked list of all the vertices. As vertices are removed from an instance to create a subinstance, they are bridged over in the linked list, so that there is always a linked list of just the vertices in the subinstance. We maintain an indication of whether each vertex is still unset or has been set to Red or Blue.

In time $O(m+n)$ and space $O(n)$, we transform the input instance into an equivalent instance without multiple edges. The simple procedure relies on a pointer array of length n, initially empty. For each vertex u, we iterate through the incident edges. For an edge to vertex v, if the vth entry of the pointer array is empty, we put a pointer to the edge uv. If the vth entry is not empty, this is not the first uv edge, and we coalesce it with the original one. That is, using the pointer to the original edge, we add the redundant edge's score function to that of the original one. We use the link from the redundant uv edge to its 'vu' twin copy to delete the twin and bridge over it, then delete and bridge over the redundant uv edge itself. After processing the last edge for vertex u we run through its edges again, clearing the pointer array. The time to process a vertex u is of order the number of its incident edges (or $O(1)$ if it is isolated), so the total time is $O(m+n)$ as claimed. Henceforth we assume without loss of generality that the input instance has no multiple edges.

3.3.2. Vertex degrees. One of the trickier points is to maintain information about the degree of each vertex. The algorithm may introduce multiple edges, and by a vertex's 'degree' we mean the number of *distinct* neighbours. Rather than keeping the precise degree of each vertex, we maintain a 'low-degree stack' containing all vertices of degrees 0, 1, and 2; a stack of degree-3 vertices; and a 'high-degree stack' of vertices of degree $\geqslant 4$. The stacks themselves are maintained as doubly linked lists, and from each vertex v we keep a pointer to its 'marker' in the stack. The stacks can be created in linear time from the input, and can also be maintained efficiently. The only difficulty comes from the possibility of multiple edges.

The key subroutine is a *degree-checking* procedure for a vertex x. Iterate through x's incident edges, keeping track of the number of distinct neighbouring vertices seen, stopping when we run out of edges or find 4 distinct neighbours. If a neighbour is repeated, coalesce the two edges. The time spent on x is $O(1)$ plus the number of edge coalescences. Once the degree of x is determined as 3, less, or more, x's marker is removed from its old stack (using the link from x to delete the marker, and links from the marker to its predecessor and successor to bridge over it), and a marker to x is pushed onto the appropriate new stack.

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

Table 1. Example of a II-reduction replacing score functions for yx and yz with a score function for xz and the associated optimal values of y.

(a)			(b)			(c)			
y	x	score	y	z	score	x	z	y	score
R	R	0	R	R	0	R	R	B	3
R	B	2	R	B	1	R	B	B	2
B	R	2	B	R	1	B	R	R	2
B	B	0	B	B	0	B	B	R	3

When reducing on vertex y, run the degree-checking procedure on each neighbour x of y. Of each neighbour's degree-checking time of $O(1)$ plus the number of edge coalescences, we charge the $O(1)$ to y, and subsume it into the time for the reduction on y which anyway is $\Theta(\deg(y))$. We account for the edge coalescences separately, claiming that in any sequence of reductions (*i.e.*, within any branch of the recursion tree) there are at most n coalescences. It suffices to show that in any sequence of reductions, at most n duplicate edges are created; this is true because only a II-reduction can create such an edge, and even then it creates at most one.

Existence of the degree stacks assures the first part of Claim 3.1, that we can select a next reduction in time $O(1)$: simply pop a vertex, in preference order, from the low-degree stack, the high-degree stack, or the degree-3 stack. We have also shown that the stack maintenance can be performed within the times specified by the second and third parts of the claim. Thus we turn our attention to the remaining, more mathematical aspects of the reductions.

3.3.3. 0-, I- and II-reductions. We omit discussion of type 0 and I reductions and start with the slightly more complicated type II reductions. Suppose that the popped vertex y has two neighbours x and z. First we construct the score function s_{xz} replacing s_{xy} and s_{yz}. At the same time, we make a note of how to set y as a function of x and z. For example if xy is a 'cut' constraint of weight 2, and yz is a cut constraint of weight 1, these are replaced by a single anti-cut constraint on xz, with associated optimal values of y: Table (a) gives the score function for xy, Table (b) gives that for yz, and Table (c), gives the score function and the optimal value of y for xz. Table (c), mapping the colouring of xz to a score and an optimal colour for y, is associated with the instance $\mathscr{M}$ being reduced. The new instance $\mathscr{M}'$ is formed by deleting edge yx and its twin xy (bridging over them in the linked lists), deleting yz and its twin zy, and adding a new edge xz and twin zx with score function taken from Table (c). Finally, vertex y is deleted. Disregarding the degree-checking time accounted for in the previous subsection, the reduction takes time and space $O(1)$. Reversing the reduction is equally straightforward, if we associate x, y, z, and Table (c) with this step in the recursive calculation; this takes space $O(1)$ per node of the implicit computation tree, at most one root-to-leaf branch of which exists at any time.

3.3.4. III-reductions. A 'Red' III-reduction on y is performed by making a first sweep through the incident edges, for edge yx adding the score function $s_{yx}(R,.)$ to the monadic

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

score function $s_x()$, then making a second sweep and deleting each edge yx and twin xy. (As each edge is deleted, we save a pointer to it and its predecessor and successor; we also save a pointer to each neighbour x of y.) For each edge xy deleted, we run a degree check on x and place it on the appropriate stack. This defines an instance $\mathscr{M}^R$. The same procedure is of course applied for a reduction to $\mathscr{M}^B$. Reversing a reduction is straightforward, if before performing it we record y's neighbours and the corresponding dyadic score functions, as well as y's monadic score function. This takes space $O(\deg(y))$.

3.3.5. Backtracking. To undo a reduction of any type, we use the saved pointers to reconstruct the deleted edges, 'un-bridging' the pointer bridges we built around them, correcting the vertex degrees, and undoing the changes to the score functions.

This establishes Claim 3.1.

3.4. Algorithm implementation and analysis

Having established Claim 3.1, it is easy to analyse the algorithm's complexity first in terms of the depth of the reduced computation tree, and then in terms of m and n.

Corollary 3.2. *An n-vertex, m-constraint Max 2-CSP instance whose computation tree has at most r type III reductions in any root-to-leaf path can be solved in time $O(n2^r)$ using space $O(m+n)$.*

Proof. If the computation tree has at most r III-reductions in any root-to-leaf path, then by definition the reduced tree is a binary tree of depth at most r, with $O(2^r)$ nodes.

On its first pass, the recursive algorithm performs a depth-first search of the reduced tree; this takes time linear in the tree's size, multiplied by the time for an elementary step. Moving from a node to its child means performing a series of 0-, I- and II-reductions and a single III-reduction, which by Claim 3.1 takes time $O(n)$. Reversing a reduction is equally easy, so returning from a child to its parent also takes time $O(n)$. Thus the total run time for the first pass is $O(n2^r)$.

On its second pass, the algorithm repeats a portion of the depth-first search until it reaches a leaf with optimal score. It then reconstructs the corresponding colouring by using the information stored at ancestors of that leaf; since there are at most r III-reductions to reverse, this pass takes time at most $O(n2^r) + O(rn) + O(n) = O(n2^r)$.

At any stage of the recursion, the sub-problem being solved corresponds to a node of the reduction tree, and at each ancestor node is recorded information to reconstruct that instance. By Claim 3.1 the annotation for reducing any vertex y is of size $O(1+\deg(y))$, and so the space needed by the algorithm is $O(m+n)$. □

We can now bound the running time of Algorithm A in terms of the excess of the graph underlying the CSP. Note that Theorem 1.2 follows as a special case of Theorem 3.3.

Theorem 3.3. *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 $O(n2^{\min(\kappa/2, m/4)})$, using space $O(m+n)$.*

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

The '$m/4$' bound is used here only for Case 4 of the proof of Theorem 7.1, and we include a short proof to keep the argument self-contained. In fact a bound of $m/5$ holds for the same algorithm. A proof sketch was given in [28]; a more careful proof is given in [29], which also gives a more sophisticated algorithm and analysis resulting in a bound of $19m/100$.

Proof. In light of Corollary 3.2, it suffices to prove that the number of type III reduction steps $r(G)$ is bounded by both $\max\{0, \kappa/2\}$ and $m/4$.

We begin with the $\kappa/2$ bound. The proof is by induction on the order of G.[1] The base cases, order 0 and 1, is trivial. If the first reduction is of type I or II (it cannot be of type 0 as the graph is connected) then the induction is trivial.

Otherwise, the first type III reduction, from G to G', reduces the number of edges by at least 3 and the number of vertices by exactly 1, thus reducing the excess to $\kappa' \leqslant \kappa - 2$. 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 $\geqslant 3$ (that is, per Section 3.3.2, at least 3 distinct neighbours, independent of edge multiplicities), so G' has minimum degree $\geqslant 2$. Thus each component G'_i has minimum degree $\geqslant 2$, and so excess $\kappa'_i \geqslant 0$. Then, by induction, $r(G) = 1 + \sum_i r(G'_i) \leqslant 1 + \sum_i \max\{0, \kappa'_i/2\} = 1 + \sum_i \kappa'_i/2 = 1 + \kappa'/2 \leqslant 1 + (\kappa - 2)/2 = \kappa/2$.

For the $m/4$ bound, we simply argue that each III-reduction results in the destruction of at least 4 edges. A III-reduction on a high-degree vertex destroys at least 4 edges instantly. If the III-reduction is instead on a degree-3 vertex then its neighbours were also of degree 3. (If any were of higher degree we would have III-reduced on it instead; if any were of lower degree we would have 0-, I- or II-reduced on it.) The III-reduction converts these 3 neighbours to degree 2, they get pushed onto the (previously empty) low-degree stack, and the next step will be to II-reduce on the first of them, destroying a fourth edge following the 3 from the III-reduction. □

4. The binomial distribution and staircase random walks

Our analysis in the next section will centre on characterizing the order and excess of a component of a random graph, which we will do by showing how these quantities are dominated by parameters of a random walk. Characterization of the random walk itself requires a certain amount of work, and since it is independent of our Max CSP context we take it up in this separate section.

We would have expected the facts given here already to be known, but we have searched the literature and spoken to colleagues without turning up anything. Even if the main points of this section are not new (Definition 4.2, Theorem 4.5, Corollary 4.6, and Theorem 4.7) they seem not to be well known, and may be of independent interest.

[1] There is a simpler 'proof' from the fact that I- and II-reductions preserve excess, and III-reductions decrease it by at least 2. Unfortunately this overlooks 0-reductions, which increase excess, or equivalently, overlooks the fact that components consisting of an isolated vertex have negative excess.

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

Our aim parallels a well-known result for Brownian motion. Since we took both that result and its proof as our model, let us state it. Let $X : [0,1] \to \mathbb{R}$ be a standard Brownian motion with $X(0) = 0$ and $X(1) = s$. Then, where ϕ denotes the density of the standard normal $N(0,1)$, the following theorem is a classical result on the 'standard Brownian bridge' $X(t) - ts$.

Theorem 4.1. *For any* $b \geqslant 0$, $\Pr\big(\max_t(X(t) - ts) \geqslant b\big) = \phi(2b)/\phi(0) = \exp(-2b^2)$.

For a standard Brownian motion, an increment $X(t+\tau) - X(t)$ has Gaussian distribution $N(0,\tau)$, and the proof of the theorem applies the reflection principle to Brownian motion, using the symmetry $\phi(x) = \phi(-x)$ of the Gaussian density.

We require an analogous result for a simple random walk $X(t)$, by which we mean a walk which at each step increases by 1 with probability p, and stays the same with probability $1-p$; we will condition the walk on $X(0) = 0$ and $X(n) = s$. The increments $X(t+\tau) - X(t)$ for the (unconditioned) random walk have binomial distribution $B(\tau, p)$, and our proof of Theorem 4.7 (analogous to the Brownian-motion theorem above) applies the reflection principle to the random walk, using the *asymmetry* of the binomial distribution as in Theorem 4.5 and Corollary 4.6.

We begin by defining and characterizing a continuous extension of the binomial density function.

Definition 4.2. For any real values $n > 0$, $0 \leqslant p \leqslant 1$, and any real k, we define

$$B_{n,p}(k) := \frac{\Gamma(n+1)}{\Gamma(k+1)\Gamma(n-k+1)} p^k (1-p)^{n-k} \tag{4.1}$$

if $0 \leqslant k \leqslant n$, and $B_{n,p}(k) := 0$ otherwise.

For integers n and $k \in \{0, \ldots, n\}$ this is of course just the binomial density $\binom{n}{k} p^k (1-p)^{n-k}$. Although the continuous extension need not integrate to 1, we will still call it the 'continuous binomial density'.

It is a simple and well-known fact that the usual binomial density function (on integers) is unimodal with with a unique maximum lying at $k = \lfloor (n+1)p \rfloor$, or two maxima if, for that k, $B_{n,p}(k) = B_{n,p}(k-1)$. We first prove that the continuous extension is unimodal.

Theorem 4.3. *The continuous binomial density defined by* (4.1) *is unimodal;* $B_{n,p}((n+1)p - 1) = B_{n,p}((n+1)p)$*; every value of* $B_{n,p}$ *on the interval* $[(n+1)p - 1, (n+1)p] = [np - (1-p), np + p]$ *exceeds every value outside it; and thus the maximum lies in this interval.*

Note that the maximum need not occur at np, as shown for instance by $n = 3$, $p = 1/3$, where the maximum occurs at around 0.82 rather than at 1.

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

Proof. We use Gauss's representation $\Gamma(x) = x^{-1} \prod_{i=1}^{\infty} \left[(1+1/i)^x (1+x/i)^{-1}\right]$ [15, p. 450]. First, $B_{n,p}(k)$ is log-concave:

$$\begin{aligned}\ln B_{n,p}(k) = {} & k \ln p + (n-k)\ln(1-p) + \ln(k+1) + \ln(n-k+1) + \ln\Gamma(n+1) \\ & - \sum_{i \geqslant 1} [(k+1)\ln(1+1/i) - \ln(1+(k+1)/i)] \\ & - \sum_{i \geqslant 1} [(n-k+1)\ln(1+1/i) - \ln(1+(n-k+1)/i)],\end{aligned}$$

so

$$\frac{d}{dk} \ln B_{n,p}(k) = \ln p - \ln(1-p) + \sum_{i \geqslant 0} \left[\frac{1}{i+k+1} - \frac{1}{i+n-k+1}\right], \tag{4.2}$$

and

$$\frac{d^2}{dk^2} \ln B_{n,p}(k) = 0 + \sum_{i \geqslant 0} [-1/(i+k+1)^2 - 1/(i+n-k+1)^2] < 0.$$

Thus $B_{n,p}(k)$ is unimodal. Also,

$$B_{n,p}(k)/B_{n,p}(k-1) = \frac{n-k+1}{k} \frac{p}{1-p}, \tag{4.3}$$

so for $k = (n+1)p$, $B_{n,p}(k)/B_{n,p}(k-1) = 1$. Thus the maximum of $B_{n,p}(k)$ occurs for some k in the range $[(n+1)p - 1, (n+1)p]$, and moreover every value of $B_{n,p}(k)$ in this range is at least as large as every value outside it. □

We will need the following simple fact.

Remark 4.4. If a real-valued function f is convex on $[a-\lambda, b+\lambda]$, with $a < b$ and $\lambda \geqslant 0$, then

$$\frac{1}{b-a} \int_a^b f(x)dx \leqslant \frac{f(a-\lambda) + f(b+\lambda)}{2}.$$

Proof. Let $\bar{f}(x)$ be the linear interpolation at x from $f(a-\lambda)$ and $f(b+\lambda)$. By convexity, for all $x \in [a-\lambda, b+\lambda]$, $f(x) \leqslant \bar{f}(x)$. Integrating, $\frac{1}{b-a}\int_a^b f(x)dx \leqslant \frac{1}{b-a}\int_a^b \bar{f}(x)dx$. As the average value of a linear function, the latter quantity is $\frac{1}{2}\left(\bar{f}(a) + \bar{f}(b)\right) = \frac{1}{2}(\bar{f}(a-\lambda) + \bar{f}(b+\lambda)) = \frac{1}{2}(f(a-\lambda) + f(b+\lambda))$, concluding the proof. □

For a Gaussian distribution, the right and left tails are of course symmetric to one another. For fixed p and large n, a binomial distribution $B_{n,p}$ is approximately Gaussian and the two tails are nearly but not exactly symmetric. We use Claim 4.3 to show that, for $p \leqslant 1/2$, a binomial's right tail (slightly) dominates its left tail. (For $p > 1/2$ the opposite is true, by symmetry.)

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

Theorem 4.5. *For $p \in (0, 1/2)$, the continuous binomial density function $B_{n,p}(k)$ defined by* (4.1) *has the property that for all deviations $\delta \geqslant 0$,*

$$B_{n,p}((n+1)p - 1 - \delta) \leqslant B_{n,p}((n+1)p + \delta).$$

Proof. For notational convenience, let $N = n + 1$. The truth of the theorem for $\delta = 0$ is immediate from Claim 4.3's assertion that $B_{n,p}(Np - 1) = B_{n,p}(Np)$. It suffices, then, to prove the nonnegativity of

$$\frac{d}{d\delta} \ln\left(\frac{B_{n,p}(Np+\delta)}{B_{n,p}(Np-\delta-1)}\right) = \frac{d}{d\delta} \ln B_{n,p}(Np+\delta) - \frac{d}{d\delta} \ln B_{n,p}(Np-\delta-1).$$

(That is, the slope going forwards from the point $Np + \delta$ should 'outweigh' the slope going backwards from $Np - \delta - 1$.) Taking the derivatives from (4.2), then, we wish to show nonnegativity of

$$2(\ln(p) - \ln(1-p)) + \sum_{i \geqslant 0} \left[\frac{1}{i + Np + \delta + 1} - \frac{1}{i + N(1-p) - \delta} + \frac{1}{i + Np - \delta} - \frac{1}{i + N(1-p) + \delta + 1} \right]. \tag{4.4}$$

Before proving this, we note that if δ is fixed and we let $N \to \infty$, it can be seen (by approximating the sum by an integral) that (4.4) tends to 0; showing (4.4) to be positive will require a little care.

Let $f(x) = 1/[(p+x)(1-p+x)]$. Note that $(1-2p)\int_0^\infty f(x)dx = \ln(1-p) - \ln(p)$, so f will be used to address the first summand in (4.4). Also $f''(x) = 2/[(1-p+x)(p+x)^3] + 2/[(1-p+x)^2(p+x)^2] + 2/[(1-p+x)^3(p+x)]$, which is positive for $x > -p$, so f is convex on $(-p, \infty)$. Returning to the quantities in (4.4), then

$$\begin{aligned}
&\sum_{i \geqslant 0} \left[\frac{1}{i + Np + \delta + 1} - \frac{1}{i + N(1-p) - \delta} + \frac{1}{i + Np - \delta} - \frac{1}{i + N(1-p) + \delta + 1} \right] \\
&\quad = \sum_{i \geqslant 0} \left[\frac{N(1-2p)}{(i + Np - \delta)(i + N(1-p) - \delta)} + \frac{N(1-2p)}{(i + Np + \delta + 1)(i + N(1-p) + \delta + 1)} \right] \\
&\quad = \sum_{i \geqslant 0} \left[\frac{\frac{1}{N}(1-2p)}{(p + (i-\delta)/N)(1 - p + (i-\delta)/N)} \right. \\
&\qquad\qquad \left. + \frac{\frac{1}{N}(1-2p)}{(p + (i+\delta+1)/N)((1-p) + (i+\delta+1)/N)} \right] \\
&\quad = \frac{1}{N}(1-2p) \sum_{i \geqslant 0} \left[f((i-\delta)/N) + f((i+\delta+1)/N) \right] \\
&\quad \geqslant (1-2p) \sum_{i \geqslant 0} 2 \int_{i/N}^{(i+1)/N} f(x)dx \quad \text{(as explained below)} \\
&\quad = 2(1-2p) \int_0^\infty f(x)dx \\
&\quad = -2(\ln(p) - \ln(1-p)),
\end{aligned}$$

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

proving the nonnegativity of (4.4). The inequality follows from Remark 4.4, with $a = i/N$, $b = (i+1)/N$, and $\lambda = \delta/N$: f is convex on $(-p, \infty)$, which contains the relevant range because $a - \lambda = (i-\delta)/N \geqslant -\delta/N > -p$ as long as $\delta \leqslant np < Np$, while if $\delta > np$ then the theorem is true trivially, as $B_{n,p}((n+1)p - 1 - \delta)$ is 0 while $B_{n,p}((n+1)p + \delta)$ is positive. □

We note that this has the following corollary for binomial random variables.

Corollary 4.6. *For a binomially distributed random variable $X \sim B(n,p)$, $p \leqslant 1/2$, for any $\delta \geqslant 0$,* $\Pr(X = \lfloor (n+1)p - 1 - \delta \rfloor) \leqslant \Pr(X = \lfloor (n+1)p + \delta \rfloor)$.

Proof. From Claim 4.3, $\Pr(X = \lfloor (n+1)p - 1 - \delta \rfloor) = B_{n,p}(\lfloor (n+1)p - 1 - \delta \rfloor) \leqslant B_{n,p}((n+1)p - 1 - \delta)$. Also, $\Pr(X = \lfloor (n+1)p + \delta \rfloor) = B_{n,p}(\lfloor (n+1)p + \delta \rfloor) \geqslant B_{n,p}((n+1)p + \delta)$: if $\lfloor (n+1)p + \delta \rfloor \geqslant (n+1)p$ the last inequality follows from the fact that $B_{n,p}$ decreases above $(n+1)p$, while if $\lfloor (n+1)p + \delta \rfloor < (n+1)p$ it follows from the fact that every value of $B_{n,p}$ in the interval $[(n+1)p - 1, (n+1)p]$ is larger than any value outside it. By Theorem 4.5, $B_{n,p}((n+1)p - 1 - \delta) \leqslant B_{n,p}((n+1)p + \delta)$. Putting the three inequalities together proves the claim. □

Next we consider the deviation of a 'staircase' random walk above its linear interpolate. Our bound on the tail of this parameter is roughly the square of what would be obtained from a naive application of Hoeffding's inequality for sampling with replacement [19, Section 6].

As noted earlier, our result and proof are modelled on a classical equality (Theorem 4.1) for the Brownian bridge. Since the 'long-run' behaviour of a simple random walk converges to Brownian motion (in a sense we do not need to make precise), it is not surprising that we should be able to obtain a similar result.

Theorem 4.7. *Fix any positive integers n and $S \leqslant n/2$, and any integer discrepancy $b \geqslant 2$. Let $X_1, \ldots, X_n$ be a 0–1 sequence chosen uniformly at random from among all such sequences having sum $X_1 + \cdots + X_n = S$. Then*

$$\Pr\left(\max_i \left\{ X_1 + \cdots + X_i - \frac{i}{n} S \right\} \geqslant b \right) \leqslant \frac{B_{n,S/n}(S + 2b - 1)}{B_{n,S/n}(S)}.$$

Proof. First observe that a random 0–1 sequence with $X_1 + \cdots + X_n = S$ as above has precisely the same distribution as a sequence of n i.i.d. Bernoulli random variables conditioned on having sum S. For the remainder of the proof we adopt this view, in particular choosing to give each random variable the distribution $X_i \sim B(p)$ with $p = S/n \leqslant 1/2$. For notational convenience, let $X^\tau = \sum_{i=1}^{\tau} X_i$. Noting that $\mathbb{E}X^\tau = \tau p$, then, we are asking for the conditional probability that there is a time τ such that $X^\tau \geqslant \tau p + b$. If so, define the 'first crossing time' τ_b to be $\min\{\tau \leqslant n : X^\tau \geqslant \tau p + b\}$, and otherwise let $\tau_b = n + 1$. Because X^τ increases by at most 1 in a step, if $\tau_b \leqslant n$ then

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

$X^{\tau_b} = \lceil \tau_b p + b \rceil$. The event we are interested in is precisely that $\tau_b \leqslant n$, conditioned on $X^n = np$:

$$\Pr(\tau_b \leqslant n \mid X^n = np) = \frac{\Pr(\tau_b \leqslant n, X^n = np)}{\Pr(X^n = np)}. \tag{4.5}$$

The numerator of this expression is

$$\begin{aligned} \text{num} &= \sum_{\tau=1}^{n} \Pr(\tau_b = \tau,\ X^n = np) \\ &= \sum_{\tau=1}^{n} \Pr(\tau_b = \tau) \Pr(X^n = np \mid \tau_b = \tau) \end{aligned}$$

which by the Markovian nature of the process

$$\begin{aligned} &= \sum_{\tau=1}^{n} \Pr(\tau_b = \tau) \Pr(X^n = np \mid X^\tau = \lceil \tau p + b \rceil) \\ &= \sum_{\tau=1}^{n} \Pr(\tau_b = \tau) B_{n-\tau,p}(np - \lceil \tau p + b \rceil) \\ &= \sum_{\tau=1}^{n} \Pr(\tau_b = \tau) B_{n-\tau,p}((n - \tau + 1)p - 1 - \delta) \end{aligned}$$

where, using b's integrality, $\delta = ((n - \tau + 1)p - 1) - (np - \lceil \tau p + b \rceil) = b - 1 + \lceil \tau p \rceil - \tau p + p \geqslant 0$, and thus we may apply the inequality of Theorem 4.5:

$$\begin{aligned} &\leqslant \sum_{\tau=1}^{n} \Pr(\tau_b = \tau) B_{n-\tau,p}((n - \tau + 1)p + \delta) \\ &= \sum_{\tau=1}^{n} \Pr(\tau_b = \tau) B_{n-\tau,p}(np + b - 1 - 2\tau p + \lceil \tau p \rceil + 2p). \end{aligned}$$

For reasons that will shortly become clear, we wish to replace the binomial's argument by $np + 2b - 1 - \lceil \tau p + b \rceil$. We observe that the original argument is larger than $(n - \tau + 1)p$ (because $\delta \geqslant 0$); the new argument is smaller than the original one (because b is integral, and $-\lceil \tau p \rceil \leqslant -2\tau p + \lceil \tau p \rceil$); and the new argument is larger than $(n - \tau + 1)p - 1$ (because the difference is $b - \lceil \tau p \rceil + \tau p - p > b - 1 - p > 0$). Thus by Theorem 4.3, the binomial's value can only increase:

$$\begin{aligned} &\leqslant \sum_{\tau=1}^{n} \Pr(\tau_b = \tau) B_{n-\tau,p}(np + 2b - 1 - \lceil \tau p + b \rceil) \\ &= \sum_{\tau=1}^{n} \Pr(\tau_b = \tau) \Pr(X^n = np + 2b - 1 \mid X^\tau = \lceil \tau p + b \rceil), \end{aligned}$$

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

which again by the Markovian nature of the process

$$\begin{aligned}
&= \sum_{\tau=1}^{n} \Pr(\tau_b = \tau) \Pr(X^n = np + 2b - 1 \mid \tau_b = \tau) \\
&= \sum_{\tau=1}^{n} \Pr(\tau_b = \tau,\, X^n = np + 2b - 1) \\
&= \Pr(X^n = np + 2b - 1),
\end{aligned}$$

where the last equality holds because $X^n = np + 2b - 1$ means that X^n exceeds its expectation np by $2b - 1 \geqslant b$ and thus implies that $\tau_b \leqslant n$. Returning to (4.5) and substituting $S = np$ yields the claim. □

Remark 4.8. For positive integers b and np with $b \leqslant 2np$,

$$\frac{B_{n,p}(np + b)}{B_{n,p}(np)} \leqslant \exp\bigl(-(3\ln(3) - 2)/4 \cdot b^2/np\bigr).$$

Proof. The proof is by simple calculation:

$$\begin{aligned}
\frac{B_{n,p}(np + b)}{B_{n,p}(np)} &= \frac{(n - (np + b - 1)) \cdots (n - np)}{(np + b) \cdots (np + 1)} \left(\frac{p}{1 - p}\right)^b \\
&= \prod_{i=1}^{b} \frac{1 - (i - 1)/n(1 - p)}{1 + i/np} \\
&\leqslant \prod_{i=1}^{b} \frac{1}{1 + i/np} \\
&\leqslant \exp\left(-\int_0^b \ln(1 + i/np)\, di\right) \\
&= \exp\bigl(-np(1 + b/np)\ln(1 + b/np) + b\bigr). \qquad (4.6)
\end{aligned}$$

If $b = x \cdot np$, then the value of c for which (4.6) *equals* $\exp\bigl(-cb^2/np\bigr)$ is $c = \frac{(x+1)\ln(x+1) - x}{x^2}$. Since this is decreasing in x, the worst-case (smallest) value of c occurs for the largest allowed value of $x = b/np$. By hypothesis, this is $x = 2$, where $c = (3\ln(3) - 2)/4$. For smaller values of $x = b/np$, then, (4.6) is smaller than $\exp\bigl(-(3\ln(3) - 2)/4 \cdot b^2/np\bigr)$, completing the proof. □

5. Stochastic size and excess of a random graph

We stochastically bound the excess $\kappa = m - n$ of a component of a random graph G via the branching-process approach pioneered by Karp [21] (see also Kendall [22], von Bahr and Martin-Löf [2], and Martin-Löf [26]). Given a graph G and a vertex x_1 in G, together with a linear order on the vertices of G, the branching 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 plus 1).

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

At each step of the process, vertices are classified as 'living', 'dead', or '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 living 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 after the ith step and define the *width* $w = \max w(i)$. (Note that $w(i)$ and w are functions of the random process, not just the component, since they depend on the order in which the vertices are taken. Despite this, for convenience we will refer to the 'width of a component'.)

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 $w(1), \ldots, w(u)$, the number of excess edges is distributed as

$$B\left(\sum_{i=1}^{u}(w(i) - 1), c/n\right). \tag{5.1}$$

Since $\sum_{i=1}^{u}(w(i) - 1) \leqslant uw$, and also $\sum_{i=1}^{u}(w(i) - 1) \leqslant \sum_{i=1}^{u}(i - 1) = \binom{u}{2}$, the number of excess edges is dominated by the random variable $B(\min\{uw, \binom{u}{2}\}, c/n)$. 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 $B(n - i, c/n)$. This allows us to define a simpler random walk which dominates the graph edge-exposure branching process.

Definition 5.1. Given a constant $c > 0$ and integer $n > 0$, define the random walk RW by $X(0) = 1$ and, for $i > 0$, $X(i) = X(i - 1) + B(n - i, c/n)$, where the binomial increments are independent. Parametrize its time-i width by $W'(i) = X(i) - i$, its order by $U' = \min\{n, \min\{i : W'(i) = 0\}\}$, and its (maximum) width by $W' = \max_{i \leqslant U'} W'(i)$.

Claim 5.2. *The order U and width W of the component G_1 on vertex 1 of a random graph $G(n, c/n)$ are stochastically dominated by U' and W' of the random walk RW.*

Proof. Consider a variant of the branching process on the random graph in which at each step we add enough new special 'red' vertices to bring the number of unexplored vertices to $n - i$. This is equivalent to the random walk RW. It also dominates the original branching process: in the implicit coupling between the two, the variant has width at least as large at every step, and thus also has maximum width and order which are at least as large as those of the original process. □

Thus the excess κ_1 of G_1 is stochastically dominated by the same quantity for RW:

$$\kappa_1 \preceq B\left(\min\left\{U'W', \binom{U'}{2}\right\}, c/n\right). \tag{5.2}$$

Let $t(G_1)$ be the time spent by Algorithm A on G_1. We shall analyse the total running time by 'charging' $t(G_1)/|G_1|$ to each vertex of G_1; the running time is then the sum of these charges.

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

Claim 5.3. *The amortized running time $t(G_1)/|G_1|$ of Algorithm A on G_1, the component on vertex 1 of a random graph $G(n, c/n)$, satisfies*

$$\mathbb{E}(t(G_1)/|G_1|) = O(1)\,\mathbb{E}\exp\bigl(c(\sqrt{2}-1)\min\{U'W'/n, U'/2\}\bigr), \tag{5.3}$$

with U' and W' given by the random walk RW.

Proof. The running time of Algorithm A on a connected graph with n_1 vertices, m_1 edges and excess $\kappa_1 = m_1 - n_1$ is

$$O(n_1 2^{\kappa_1/2}). \tag{5.4}$$

The exponential moments of binomial random variables are simple and well known: if a random variable U has distribution $B(N, p)$, then

$$\mathbb{E}z^U = \sum_{i=0}^{N} \binom{N}{i} z^i p^i (1-p)^{N-i} = (pz + (1-p))^N = (1 + p(z-1))^N \leqslant \exp(p(z-1)N),$$

and in particular,

$$\mathbb{E}\sqrt{2}^{\,U} \leqslant \exp((\sqrt{2}-1)Np). \tag{5.5}$$

Setting $N = \min\{U'W', \binom{U'}{2}\}$ and combining (5.2), (5.4), and (5.5) gives

$$\mathbb{E}(t(G_1)/|G_1|) = O(1)\,\mathbb{E}(2^{\kappa/2}) \leqslant O(1)\,\mathbb{E}\exp((\sqrt{2}-1)N\,c/n). \tag{5.6}$$

Noting that $U' \leqslant n$, and so $\binom{U'}{2}/n \leqslant U'/2$, yields (5.3). □

In the following, we therefore focus on finding bounds on the probability $\Pr(U, W)$ that the 'first' component of a random graph has order U and width W, or, since $(U, W) \preceq (U', W')$ per Claim 5.2, the corresponding probability $\Pr(U', W')$ for the random walk RW.

We use a version of Chernoff's inequality (see Janson, Łuczak and Ruciński [20, Theorems 2.1 and 2.8]), which states that for a sum Z of independent 0–1 Bernoulli random variables with parameters $p_1, \ldots, p_n$ and expectation $\mu = \sum_{i=1}^{n} p_i$:

$$\Pr(Z \geqslant \mu + t) \leqslant \exp\bigl(-t^2/(2\mu + 2t/3)\bigr), \tag{5.7}$$

$$\Pr(Z \leqslant \mu - t) \leqslant \exp\bigl(-t^2/(2\mu)\bigr). \tag{5.8}$$

The next lemma describes the probability that U' is large, and has a corollary for the probability that $|G_1|$ is large. (We will not use the corollary, but we state it because it is natural and potentially useful.) Although the proof is framed in terms of the binomial increments $B(n-i, c/n)$ for RW (corresponding to vertex exposures in the random graph), it may also be helpful to think in terms of subdividing such an increment into $n-i$ Bernoulli increments $\mathrm{Be}(c/n)$ (corresponding to edge exposures in the random graph).[2]

[2] The edge-exposure model was previously used by Spencer in an elegant short paper [30]. Spencer was studying a related problem, calculating $C(n, n+k)$ (the number of connected graphs with k vertices and $n+k$ edges) for k fixed and $n \to \infty$. We will discuss $C(n, n+k)$ in Section 6.

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

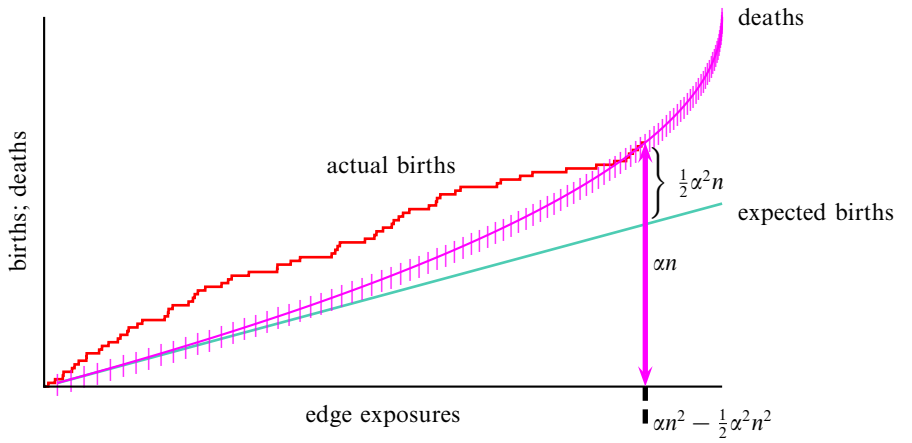


Figure 1. Birth-death process described by edge exposures in the augmented graph process, or equivalently by Bernoulli exposures in RW.

This view will be essential in proving Lemma 5.7, and is illustrated in Figure 1. The X axis indicates edge exposures j in the augmented graph model, or equivalently the number of Bernoulli random variables exposed in the random walk (a 'finer sampling' of the same RW). Since the number of edge exposures between successive deaths shrinks from $n-1$ to $n-2$ *etc.*, the cumulative number of deaths (call it $d(j)$) grows super-linearly. (As it happens, $d(j)$ is a parabola rotated sideways.) The expected cumulative number of births grows linearly, as $(c/n)j$, and (for $c=1$) is tangent to the 'death curve' at the origin. The event that the actual number of births equals deaths equals αn means that a corresponding sum of Bernoulli random variables $\mathrm{Be}(c/n)$ equals αn; the individual births comprising this sum describe a random walk, a sample of which is shown in the figure.

Lemma 5.4. *Let $n>0$ be an integer, $\Lambda>0$ a real, and $c>0$ a real with $c \leqslant 1+\Lambda$. For any integer $i>0$, setting $\alpha = i/n$, the time-i widths $W'(i)$ of the random walk RW with parameters c, n satisfy*

$$\Pr(W'(\alpha n) > 0) \leqslant \exp\left(\frac{-3\alpha^3 n(1-6\Lambda/\alpha)}{24-8\alpha}\right). \tag{5.9}$$

Proof. We assume $\Lambda/\alpha < 1/6$ since otherwise the inequality is trivial. We also assume without loss of generality that $c = 1+\Lambda$, since the corresponding process dominates that for any smaller c. Note that $W'(i)$ has distribution

$$\begin{aligned} W'(i) &\sim B\left((n-1)+\cdots+(n-i), \frac{1+\Lambda}{n}\right) - i + 1 \\ &= B\left(ni - \binom{i+1}{2}, \frac{1+\Lambda}{n}\right) - i + 1 \end{aligned}$$

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

and so $W'(i) > 0$ means that

$$B\left(ni - \binom{i+1}{2}, \frac{1+\Lambda}{n}\right) \geqslant i = \alpha n. \tag{5.10}$$

This binomial r.v. has expectation

$$\left(\alpha n^2 - \binom{\alpha n + 1}{2}\right)\frac{1+\Lambda}{n} \leqslant (1+\Lambda)(\alpha - \alpha^2/2)n. \tag{5.11}$$

For convenience, define $q = \Lambda/\alpha < 1/6$. Thus if (5.10) holds, the r.v. exceeds its expectation by at least

$$\alpha^2 n/2 - \Lambda(\alpha - \alpha^2/2)n = \frac{\alpha^2 n}{2}(1 - 2q + \alpha q) \geqslant 0. \tag{5.12}$$

Together with (5.11) and (5.12), (5.7) implies that (5.10) has probability at most

$$\exp\left(\frac{-\alpha^4 n^2 (1-2q+\alpha q)^2/4}{2(1+q\alpha)(\alpha - \alpha^2/2)n + \alpha^2 n(1-2q+\alpha q)/3)}\right). \tag{5.13}$$

A short calculation shows that this is at most (5.9). (More careful calculation can decrease the '6' in 6Λ a bit, but this is immaterial for our purposes.) □

We will use the lemma in the form of the following corollary.

Corollary 5.5. *Let $n > 0$ be an integer, $\Lambda > 0$ a real, and $c > 0$ a real with $c \leqslant 1 + \Lambda$. For any integers $i, j > 0$ with $i < j \leqslant 2i$, setting $\alpha = i/n$, we have*

$$\Pr(\exists k \in (i,j] : W'(k) = 0) \leqslant K \exp\left(\frac{-3\alpha^3 n(1 - 6\Lambda/\alpha)}{24 - 8\alpha}\right), \tag{5.14}$$

for some absolute constant K.

Proof. Assume without loss of generality that $j \leqslant n$. Let A be the event $(W'(i) > 0)$, let Z be the event $(\exists k \in (i,j] : W'(k) = 0)$, and for $k \in (i,j]$, define the event $B_k = (W'(k) = 0) \cap (W'(l) \neq 0 \,\forall l \in [k+1, j])$ (*i.e.*, the last zero of $W'(\cdot)$ in $[i,j]$ is at k). Then the events B_k are disjoint and $Z = \bigcup_{k=i+1}^{j} B_k$.

By Lemma 5.4, it is sufficient to show that $\Pr(A \mid Z) \geqslant 1/K$, since then $\Pr(A) \geqslant \Pr(A \cap Z) = \Pr(Z)\Pr(A \mid Z) \geqslant \Pr(Z)/K$, and the inequality follows from (5.9). As the B_k are disjoint, $\Pr(A \mid Z) = \sum_{k=i+1}^{j} \Pr(A \mid B_k)\Pr(B_k \mid Z)$. Since $\sum_{k=i+1}^{j} \Pr(B_k \mid Z) = 1$, it is enough to show that, for $k \in (i,j]$, $\Pr(A \mid B_k) \geqslant 1/K$.

Now consider the random walk $W'(\cdot)$. For $k > 0$, let $e(k) = nk - \binom{k+1}{2}$ (the number of edge exposures by time k). Then $W'(k) \sim B(e(k), (1+\Lambda)/n) - k + 1$. Thus B_k holds only if we have exactly $k-1$ successes in $e(k) = nk - \binom{k+1}{2}$ Bernoulli trials. Conditioning on B_k, $A \cap B_k$ is the event that $X \geqslant i$, where X is the number of these $k-1$ successes which occur within the first $e(i)$ trials. The $k-1$ successes are uniformly random over the $e(k)$ trials, so X is a hypergeometric random variable parametrized by $e(k)$ trials, $k-1$ successes, and $e(i)$ samples. We wish to show that $\Pr(X \geqslant i) \geqslant 1/K$.

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

The times of the $k-1$ successes may be simulated by drawing them randomly without replacement from $\{1,\ldots,n\}$, the rth success falling among the first $e(i)$ trials with probability at least $(e(i)-(r-1))/(e(k)-(r-1))$ (even if all the previous $r-1$ successes fell among the first $e(i)$ trials). Since $(e(i)-(r-1))/(e(k)-(r-1)) \geqslant e(i)/e(k)-(k-1)/e(k) \geqslant i/k-2/n \geqslant (i-2)/k$, X dominates a random variable $Y \sim B(k-1,\,(i-2)/k)$, and it suffices to show that $\Pr(Y \geqslant i) \geqslant 1/K$.

For any bounded k, $\Pr(Y \geqslant i) \geqslant \Pr(Y = k-1)$ is bounded away from 0, so it suffices to consider $k \geqslant 1001$. Since $i \geqslant k/2$, the binomial's probability parameter $(i-2)/k$ is at least 0.49. We now consider two cases. If $1-(i-2)/k \leqslant 100/(k-1)$ then $\Pr(Y \geqslant i) \geqslant \Pr(Y = k-1) \geqslant (1-100/(k-1))^{k-1} \geqslant 10^{-50}$. Otherwise, Y has variance $\sigma^2 \geqslant (k-1)\cdot 0.49\cdot 100/(k-1) = 49$. By the Berry–Esseen theorem (see for example [4]), the error in estimating Y from the corresponding Gaussian is at most $1.88/\sigma$. Since the Gaussian itself exceeds its mean by 3 with probability at least $1/2-3/(\sqrt{2\pi}\,\sigma)$, $\Pr(Y \geqslant i) \geqslant \Pr(Y \geqslant \mathbb{E}Y+3) \geqslant 1/2-3/(\sqrt{2\pi}\,\sigma)-1.88/\sigma > 0.05$. □

While we will not use it, we note that Lemma 5.4 has an immediate consequence for a component of a random graph.

Corollary 5.6. *Let $n>0$ be an integer, $\Lambda>0$ a real, and $c>0$ a real with $c \leqslant 1+\Lambda$. For any integer $i>0$, setting $\alpha = i/n$, the order of the component G_1 containing vertex 1 in a random graph $G(n,c/n)$ satisfies*

$$\Pr(|G_1| > \alpha n) \leqslant \exp\left(\frac{-3\alpha^3 n(1-6\Lambda/\alpha)}{24-8\alpha}\right).$$

Lemma 5.7. *Given an integer $n \geqslant 4$ and any value $c>0$, RW has the property that, for any $\alpha \in (0,1]$ and any $\beta \geqslant \frac{\alpha^2}{8-4\alpha}+\frac{4}{n}$,*

$$\Pr(W' \geqslant \beta n \mid W'(\alpha n)=0) \leqslant \exp\left(-(3\ln(3)-2)\left(\beta-\frac{\alpha^2}{8-4\alpha}-\frac{4}{n}\right)^2\frac{n}{\alpha}\right). \tag{5.15}$$

Proof. If $W'(\alpha n)=0$ then $W' \leqslant \max_{t\leqslant \alpha n} W'(t)$. Since $W'(t)$ decreases by at most 1 per step, and is 0 at αn, we have $W' \leqslant \alpha n$ and thus we may assume $\beta \leqslant \alpha$ (or else the result is trivial).

Switching from the vertex-exposure to the edge-exposure view for the random walk, up to and including the tth step of RW, the total number of edge exposures is

$$e(t) := (n-1)+\cdots+(n-t) = tn-\binom{t+1}{2},$$

and the total number of births is $Z_1+\cdots+Z_{e(t)}$, where the Zs are i.i.d. Bernoulli $\mathrm{Be}(c/n)$ random variables.

For the remainder of the proof, set $i := \alpha n$. Since the number of deaths by the tth vertex exposure is t, and the number of births is $Z_1+\cdots+Z_{e(t)}$, and we start with one live vertex, the condition $W'(i)=0$ means that the sequence $Z_1, Z_2, \ldots$ is conditioned by $Z_1+\cdots+Z_{e(i)} = i-1$.

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

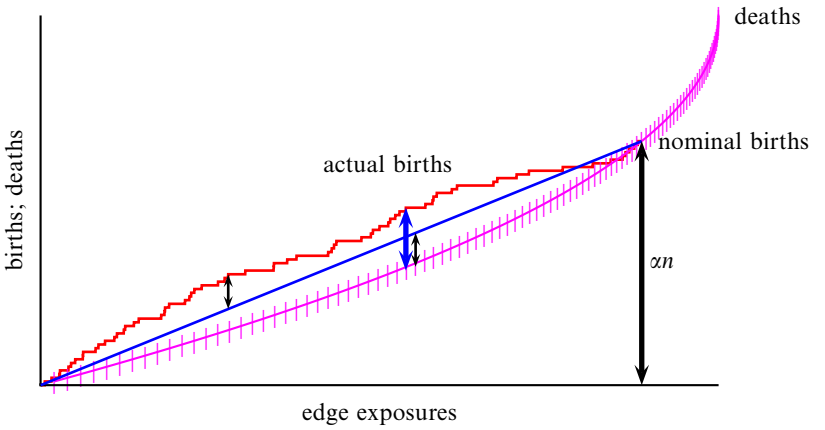


Figure 2. Birth/death process

For any time $t \leqslant i$, the number of live vertices is given by

$$W'(t) = Z_1 + \cdots + Z_{e(t)} - (t-1) \tag{5.16}$$

$$= \left[Z_1 + \cdots + Z_{e(t)} - \frac{e(t)}{e(i)}(i-1)\right] + \left[\frac{e(t)}{e(i)}(i-1) - (t-1)\right]; \tag{5.17}$$

that is, the gap between the actual number born and its expectation (conditioned on $Z_1 + \cdots + Z_{e(i)} = i-1$), plus the gap between the expectation and the number of deaths.

This view may be more easily apprehended with reference to Figure 2. The maximum of $W'(t)$, the number of live vertices at any time, is the largest gap between this random walk and the death curve, which can be bounded as the maximum gap between the random walk and a linear interpolation of it, plus the maximum gap between the linear interpolation and the death curve. The first of these two gaps is a random quantity governed by Claim 4.7, while the second is a deterministic function of α and n.

The second term in (5.17) may be bounded as

$$\max_{t \leqslant i}\left[\frac{e(t)}{e(i)}(i-1) - (t-1)\right] \leqslant 1 + \max_{t \leqslant i}\left[\frac{e(t)}{e(i)}i - t\right]$$
$$= 1 + i^2/[8n - 4i - 4]$$

(the maximum occurs at $t = i/2$), and with $i = \alpha n$ and $n \geqslant 2$, it is easily checked that this is

$$\leqslant \alpha^2 n/(8 - 4\alpha) + 2.$$

Thus, defining a 'discrepancy' δ by $\delta n = \lfloor \beta n - \alpha^2 n/(8-4\alpha) - 2 \rfloor \geqslant \beta n - \alpha^2 n/(8-4\alpha) - 3$, we may rewrite (5.17) as $W'(t) \leqslant \left[Z_1 + \cdots + Z_{e(t)} - \frac{e(t)}{e(i)}(i-1)\right] + \beta n - \delta n$

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

to obtain

$$\begin{aligned}\Pr\bigl(\max_{t\leqslant\alpha n} W'(t)\geqslant\beta n \mid W'(\alpha n)=0\bigr)&\\ \leqslant \Pr\left(\max_{t\leqslant\alpha n}\left[Z_1+\cdots+Z_{e(t)}-\frac{e(t)}{e(i)}(i-1)\right]\ \geqslant\ \delta n \mid W'(\alpha n)=0\right)&\\ \leqslant \Pr\left(\max_{j\leqslant e(i)}\left[Z_1+\cdots+Z_j-\frac{j}{e(i)}(i-1)\right]\ \geqslant\ \delta n \mid W'(\alpha n)=0\right).&\end{aligned}$$

Recalling that we have conditioned upon $Z_1+\cdots+Z_{e(i)}=i-1=\alpha n-1$, we apply Theorem 4.7, with $S=i-1=\alpha n-1$ and $b=\delta n$. (The theorem's 'n' is $e(i)$, so its '$p=S/n$' is $\frac{i-1}{e(i)}=\frac{i-1}{in-\binom{i+1}{2}}$, and its hypothesis '$p\leqslant 1/2$' is guaranteed by $n\geqslant 4$.) Using the notation $B(n,p;k)$ for $B_{n,p}(k)$, this shows the probability to be

$$\leqslant \frac{B\bigl(e(i),(i-1)/e(i);\ (i-1)+(2\delta n-1)\bigr)}{B\bigl(e(i),(i-1)/e(i);\ (i-1)\bigr)}. \tag{5.18}$$

By its definition, $\delta n\leqslant\beta n-2$, and we have already argued that the probability is 0 unless $\beta\leqslant\alpha$, so we may assume $\delta n\leqslant\beta n-2\leqslant\alpha n-2=i-2$, and in particular, $2\delta n-1<2(i-1)$: the deviation of $2\delta n-1$ in (5.18) is less than twice the mean. Thus we may apply Remark 4.8, showing the probability to be

$$\begin{aligned}&\leqslant \exp\bigl(-(3\ln(3)-2)/4\ \cdot(2\delta n-1)^2/(\alpha n-1)\bigr)\\ &\leqslant \exp\left(-(3\ln(3)-2)\left(\beta-\frac{\alpha^2}{8-4\alpha}-\frac{4}{n}\right)^2\frac{n}{\alpha}\right).\end{aligned}$$

□

We will use the lemma in the form of the following corollary.

Corollary 5.8. *Given an integer $n\geqslant 4$ and any value $c>0$, RW has the property that, for any $0<\alpha_0\leqslant\alpha_1\leqslant 1$ and any $\beta\geqslant\frac{\alpha_0^2}{8-4\alpha_0}+\frac{4}{n}$,*

$$\begin{aligned}\Pr(W'\geqslant\beta n \mid \exists\alpha\in[\alpha_0,\alpha_1]: W'(\alpha n)=0)&\\ \leqslant \exp\left(-(3\ln(3)-2)\left(\beta-\frac{\alpha_1^2}{8-4\alpha_1}-\frac{4}{n}\right)^2\frac{n}{\alpha_1}\right).&\end{aligned} \tag{5.19}$$

Proof. Let B_k and Z be the events defined in the proof of Corollary 5.5, where we take $i=\alpha_0 n$ and $j=\alpha_1 n$, and let C be the event $(W'\geqslant\beta n)$. Then,

$$\begin{aligned}\Pr(C\mid Z)&=\sum_{k=i}^{j}\Pr(C\mid B_k)\Pr(B_k\mid Z)\\ &=\sum_{k=i}^{j}\Pr(C\mid W'(k)=0)\Pr(B_k\mid Z)\qquad (W'(\cdot)\text{ is Markovian})\\ &\leqslant \max_{k\in[i,j]}\Pr(C\mid W'(k)=0).\end{aligned}$$

We are now done, by Lemma 5.7. □

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

6. Remarks on the bounds

Recall that we are aiming to bound κ, the excess of the component of a random graph containing the fixed vertex 1. That the tail bounds on κ must be done carefully – that the constants count – is illustrated by considering the probability that a large (linear-sized) excess arises simply because the random graph $G(n, 1/n)$ has many more edges than expected. Such a graph has $(n + \epsilon n)/2$ edges (rather than the expected $n/2$) with probability $\exp(-\Theta(\epsilon^2 n))$, and in this event the expected value of κ is $\Theta(\epsilon^3 n)$. Thus for $\epsilon = \Theta(1)$, the probability of excess $\kappa = \epsilon^3 n$ is at least $\exp(-\Theta(n))$, the running time is $\exp(\Theta(n))$, and it becomes critical which of the two constants hidden in the respective Thetas is larger. In particular, it is quite conceivable that it really might be essential to have our running-time bound of $2^{\kappa/2}$ rather than the more naive 2^κ that would be obtained by III-reducing on all vertices of degree 3 or more. This example also shows that good tail bounds on κ will be required even in the fantastically improbable regime $\kappa = \Theta(n)$.[3]

As remarked earlier, tail bounds on κ could also be computed by first-moment methods. Writing $C(u, u + \kappa)$ for the number of connected graphs with u vertices and $u + \kappa$ edges, and applying our algorithm to a random graph $G(n, p)$, the expected time spent on components of order u and excess κ is at most $2^{\kappa/2}$ times the expected number of them, namely

$$2^{\kappa/2} \binom{n}{u} C(u, u + \kappa) p^{u+\kappa} (1 - p)^{u(n-u)+\binom{u}{2}-u-\kappa}. \tag{6.1}$$

Motivated by the preceding paragraph, we will consider the value of this expression at $p = 1/n$, $u = \epsilon n$ and $\kappa = \epsilon^3 n$, with ϵ constant; we fix these values for the remainder of this section.

Wright [32] shows that $C(u, u + \kappa)$ is asymptotically equal to some explicit constant times

$$(A/\kappa)^{\kappa/2} u^{u+(3\kappa-1)/2} \tag{6.2}$$

where $A = e/12$ and the formula is valid for $1 \ll \kappa \ll u^{1/3}$, *i.e.*, for $\kappa \to \infty$ but $\kappa = o(u^{1/3})$. Since the algorithm's expected time is at most (6.1) summed over all u and κ, and Wright's formula for $C(u, u + \kappa)$ sacrifices only a constant factor, nothing is given up (outside of the $2^{\kappa/2}$ term in (6.1)), and this method of calculation must yield a suitable result (a value linear in n) *in the range* where Wright's formula is applicable. Unfortunately, this range does not include our $u = \epsilon n$, $\kappa = \epsilon^3 n$.

Similarly, Łuczak [25] shows that for $0 \ll \kappa \ll u$,

$$1/(e^8 \sqrt{\kappa})(A/\kappa)^{\kappa/2} u^{u+(3\kappa-1)/2} \;\leqslant\; C(u, u + \kappa) \;\leqslant\; \sqrt{u^3/\kappa}(A/\kappa)^{\kappa/2} u^{u+(3\kappa-1)/2}$$

where $A = e/12 + O(\kappa/u)$. With ϵ fixed, $u = \epsilon n$ and $\kappa = \epsilon^3 n$, this does not give the needed explicit bound on A. It might be possible to extract such a bound, and also to work with all parameter pairs (u, κ) (not just the demonstrative values we chose here), but even then we

[3] Were it not for the need to go up into the tails, we could capitalize on results such as those of Aldous [1], that the joint distribution of the largest component's order and excess is asymptotically normal. In Aldous's analysis, Brownian motion plays the same role as our random walk RW.

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

would be left with the polynomial leading factors: unless these can be banished, we would be unable to prove that the expected running time is linear rather than merely polynomial.

Bollobás [5] (see also Bollobás [6, V.4]) shows that (6.2) is a universal upper bound on $C(u, u+\kappa)$ for some universal constant A. Substituting (6.2) into (6.1) and evaluating at $p = 1/n$, $u = \epsilon n$ and $\kappa = \epsilon^3 n$ gives (up to small polynomial factors of n and ϵ) $[c(\epsilon)A]^{\epsilon^3 n/2}$, where $c(\epsilon)$ is an easily calculated explicit function. For this method to show that the expected time (for this u and κ) is polynomial in n, we would need a reasonably small upper bound on the constant A.

Bender, Canfield and McKay [3] (in the same journal issue as Łuczak's [25]) give extremely accurate estimates for $C(u, u+\kappa)$, whose substitution into expression (6.1) must in principle satisfy our needs. Their formula, though, is rather complex, involves an implicitly defined function, and appears difficult to work with.

In short, while suitable tail bounds could presumably be proved by a first-moment calculation, there are complications. We therefore chose to adopt the branching-process approach.

The branching-process approach also provides some intuition. It is a classical Erdős–Rényi result [13, 12] that a random graph at the critical density $1/n$ typically has a giant component whose size is $\Theta(n^{2/3})$. Since our component sizes are given as αn, the giant component (which is likely to be the most difficult component for our algorithm to solve, and thus the component we should focus on) would have $\alpha n = \Theta(n^{2/3})$, $\alpha = \Theta(n^{-1/3})$. The inequality (5.9) 'allows' such a component, giving its probability as $\exp(-\Theta(1))$.

Just above the scaling window, in a random graph $G(n,p)$ with $p = (1+\epsilon)/n$ where $n^{-1/3} \ll \epsilon \ll 1$ (or equivalently $p = (1+\lambda n^{-1/3})/n$ where $1 \ll \lambda \ll n^{1/3}$), up to constant factors the giant component's order is typically $\epsilon n = \lambda n^{2/3}$ and its excess typically $\epsilon^3 n = \lambda^3$, suggesting we consider $\alpha = \lambda n^{-1/3}$ and $\kappa = \lambda^3$. (See Bollobás [6, Chapter VI] and particularly Janson, Łuczak and Ruciński [20, Section 5.4] for this and related results.) Since our bound suggests that κ might be about $(\alpha n)(\beta n)(c/n)$, this would suggest that the typical 'width' (maximum number of live vertices) βn of the giant component would have $\beta = \lambda^2 n^{-2/3}$. Notice that (5.15) imposes no penalty on β until $\beta = \Theta(\alpha^2)$, and the above values $\alpha = \lambda n^{-1/3}$, $\beta = \lambda^2 n^{-2/3}$ fall just at this point.

In summary, we think that the branching-process analysis offers insight into the likelihood or unlikelihood of observing graph components of given order and excess. Especially if one takes as given the properties of binomial distributions and staircase random walks proved in Section 4, the branching process analysis is not unduly complicated, and is entirely self-contained.

7. Assembly

In this section we state and prove the main result, Theorem 7.1; note that Theorem 1.1 follows as a special case.

Theorem 7.1. *For any $\lambda = \lambda(n) > 0$ and $c \leqslant 1 + \lambda n^{-1/3}$, let $G \in \mathscr{G}(n, c/n)$ 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 $O(n)\exp(O(1+\lambda^3))$, and in space $O(m+n)$.*

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

Proof. Consider Algorithm A applied to the graph G. We calculate the running time as follows: for each component G' of G, let $t(G')$ be the time taken by Algorithm A to find an optimal assignment for G'. For each vertex v of G', we define $t(v) = t(G')/|G'|$. Then the total running time of Algorithm A is $O(m+n) + \sum_{v \in V(G)} t(v)$. Choosing any fixed vertex $v \in V(G)$ (say, $v = 1$), we see that the expected running time is at most

$$O((1+c)n) + n\mathbb{E}t(v). \tag{7.1}$$

It therefore suffices to prove that

$$\mathbb{E}\, t(v) \leqslant \exp(O(1+\lambda^3)) = \exp(O(1+\Lambda^3 n)), \tag{7.2}$$

where $\Lambda = \lambda n^{-1/3}$.

Recall the random walk RW of Definition 5.1, with order U' and maximum width W'. We examine the contribution of each possible pair (U', W') to the expectation. Specifically, for integers αn and βn, define $E(\alpha, \beta)$ to be the expected time that the algorithm spends on cases with $U' = \alpha n$ and $W' = \beta n$. Recalling that if $U' < n$ then $W' \leqslant U'$, we compute the contribution separately for $0 \leqslant \beta \leqslant \alpha < 1$ (this forms the bulk of the argument to come, including Cases 1–4 below), and for $\alpha = 1$ (Case 5).

As we are computing asymptotics in n, we may assume that n is large ($n \geqslant 10000$). It is enough to prove that

$$\sum_{\alpha n, \beta n} E(\alpha, \beta) \leqslant \exp(O(1+\lambda^3)). \tag{7.3}$$

To do so, we break $[0,1]^2$ into rectangular regions and bound the sum of $E(\alpha, \beta)$ separately over each region. Assuming $\alpha < 1$, the amortized running time of Algorithm A is bounded by Claim 5.3. Thus for any rectangle $R = [\alpha_0, \alpha_1] \times [\beta_0, \beta_1] \subseteq [0,1) \times [0,1]$,

$$\begin{aligned}
\sum_{(\alpha,\beta)\in R} E(\alpha,\beta) &\leqslant O(1) \exp\left\{ \max_R [(1+\Lambda)(\sqrt{2}-1)\alpha\beta] n \right\} \Pr((U'/n, W'/n) \in R) \\
&\leqslant O(1) \exp\left\{ \max_R [(1+\Lambda)(\sqrt{2}-1)\alpha\beta] n \right\} \\
&\quad \cdot \Pr(\exists \alpha \in [\alpha_0, \alpha_1] : W'(\alpha n) = 0) \\
&\quad \cdot \Pr(W'/n \in [\beta_0, \beta_1] \mid \exists \alpha \in [\alpha_0, \alpha_1] : W'(\alpha n) = 0).
\end{aligned}$$

By Corollaries 5.5 and 5.8,

$$\begin{aligned}
\sum_{(\alpha,\beta)\in R} E(\alpha,\beta) \leqslant O(1) \exp\Bigg\{ \Bigg(& \max_R \left[(1+\Lambda)(\sqrt{2}-1)\alpha\beta\right] - \left[\frac{3\tilde{\alpha}^3(1-6\Lambda/\tilde{\alpha})}{24-8\tilde{\alpha}} I(\tilde{\alpha}) \right] \\
& - \left[(3\ln(3)-2) \left(\beta_0 - \frac{\alpha_1^2}{8-4\alpha_1} - \frac{4}{n} \right)^2 \frac{1}{\alpha_1} J\left(\alpha_1, \beta_0 - \frac{4}{n} \right) \right] \Bigg) n \Bigg\},
\end{aligned} \tag{7.4}$$

where $\tilde{\alpha}$ is $\alpha_0 - 1/n$ (adjusting for the open interval of Corollary 5.5), $I(\tilde{\alpha})$ is the indicator function for $\tilde{\alpha} > 6\Lambda$, and $J(\alpha, \beta)$ is the indicator function for $\beta \geqslant \alpha^2/(8-4\alpha)$. We reiterate that (7.4) bounds the *sum* of $E(\alpha, \beta)$ over R, not just the maximum.

Without loss of generality we restrict Λ to $\Lambda > n^{-1/3}$, since for smaller Λ the quantity (7.4) decreases monotonically while the target bound $n \exp(O(1+\Lambda^3 n))$ does not decrease

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

below $n\exp(O(1))$. We may also restrict Λ to $\Lambda < 0.01$, since by then the target bound $n\exp(O(0.01^3 n))$ allows time for the algorithm to III-reduce on every vertex.

Given that $\Lambda < 0.01$ and $\alpha \leqslant 1$, in (7.4) we may replace each $\tilde{\alpha}$ by α at the expense of an $O(1)$ factor outside the whole expression: if $I(\tilde{\alpha}) = I(\alpha) = 1$ this is easy to check, while if $I(\tilde{\alpha}) = 0$ but $I(\alpha) = 1$ then $6\Lambda < \alpha \leqslant 6\Lambda + 1/n$, and a short calculation shows that the exponent (not forgetting the factor of n) changes by at most $O(1)$ and so the expression is multiplied by $\exp(O(1)) = O(1)$.

Also, given that $\Lambda < 0.01$ and $\alpha \leqslant 1$, on substituting $\beta = \beta' + \frac{4}{n}$ into (7.4), the first term (again taking into account the n inside the exponent) is $\leqslant \exp\{(1+\Lambda)(\sqrt{2}-1)\alpha\beta' n + 4\}$, and we may simply move the exp(4) outside and subsume it into the leading $O(1)$. That is, we may simply ignore the $\frac{4}{n}$ in (7.4), extending the range of summation to $\beta \in [-\frac{4}{n}, 1]$, and so summing, over rectangles $R = [\alpha_0, \alpha_1] \times [\beta_0, \beta_1]$, $O(1)$ times

$$\exp\left\{\left(\max_R\left[(1+\Lambda)(\sqrt{2}-1)\alpha\beta\right] - \left[\frac{3\alpha_0^3(1-6\Lambda/\alpha_0)}{24-8\alpha_0}I(\alpha_0)\right]\right.\right.$$
$$\left.\left. - \left[(3\ln(3)-2)\left(\beta_0 - \frac{\alpha_1^2}{8-4\alpha_1}\right)^2 \frac{1}{\alpha_1}J(\alpha_1,\beta_0)\right]\right)n\right\}. \tag{7.5}$$

We now consider five regimes of values (α, β), which together cover the space $[0,1] \times [-\frac{4}{n}, 1]$:

(1) α and β both small: $(\alpha, \beta) \in [0, 1000\Lambda] \times [0, 1000000\Lambda^2]$,
(2) $\beta \leqslant 0$,
(3) $\alpha < 1$ and $0 \leqslant \beta \leqslant 1.03\alpha^2/(8-4\alpha)$ (and excluding Case 1),
(4) $\alpha < 1$ and $\beta \geqslant 1.03\alpha^2/(8-4\alpha)$ (and excluding Case 1),
(5) $\alpha = 1$.

Only the first case contains likely and significant pairs (α, β), and it defines the bound in (7.3); the second case trivially makes a negligible contribution; the remaining cases also contribute negligibly but are a little trickier to analyse. Note that the $\alpha = 1$ case must be treated differently: if $U' = \alpha n = n$ we may have $W'(\alpha n) > 0$, Corollary 5.8 (with $\alpha_0 = \alpha_1 = 1$) does not apply, and thus neither does (7.5).

Case 1. $R = [0,\ 1000\Lambda] \times [0, 1000000\Lambda^2]$. This rectangle R contains likely pairs (α, β), and we simply estimate R's probability as 1. Thus (7.5) becomes

$$\sum_R E(\alpha,\beta) \leqslant \exp\{(1+\Lambda)(\sqrt{2}-1)\cdot 1000\Lambda \cdot 1000000\Lambda^2 \cdot n\} = \exp(O(\Lambda^3 n)).$$

Case 2. $R = [0,1] \times [-\frac{4}{n}, 1]$. Again we simply estimate R's probability as 1; since $\beta \leqslant 0$, the exponential's value is at most 1.

Case 3. Instead of considering only $\beta \leqslant 1.03\alpha^2/(8-4\alpha)$, we will treat a larger domain, $\beta \leqslant 1.03\alpha^2/4$. If $\alpha \leqslant 1000\Lambda$ this falls under Case 1, so we need only consider $\alpha \geqslant 1000\Lambda$. We cover the space $1000\Lambda \leqslant \alpha \leqslant 1$, $\beta \leqslant 1.03\alpha^2/4$ with rectangles

$$R_i = [\alpha_i^\star, 1.01\alpha_i^\star] \times [0, 1.03(1.01\alpha_i^\star)^2/4],$$

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

with $\alpha_i^\star = 1000\Lambda \cdot 1.01^i$, $i = 0, 1, \ldots, I-1$, where $\alpha_I^\star$ is the first term larger than 1. Within any such rectangle R, writing $\alpha^\star$ for $\alpha_i^\star$ (and noting that $\alpha^\star \leqslant 1$), (7.5) is at most

$$\begin{aligned}
&\exp\left\{\left(\left[(1+\Lambda)(\sqrt{2}-1)\cdot 1.01^3 \cdot 1.03\,\alpha^{\star 3}/4\right] - \left[\frac{3\alpha^{\star 3}(1-6\Lambda/\alpha^\star)}{24-8\alpha^\star}\right]\right)n\right\} \\
&\quad\leqslant \exp\left\{\left(\left[1.01\,(\sqrt{2}-1)\cdot 1.01^3 \cdot 1.03/4\right] - \left[\frac{3(1-6/1000)}{24}\right]\right)\alpha^{\star 3} n\right\} \\
&\quad\leqslant \exp\left(-0.013\alpha^{\star 3} n\right).
\end{aligned}$$

Recalling that $\alpha_i^\star = 1000\Lambda \cdot 1.01^i$ and that $\Lambda \geqslant n^{-1/3}$, the contribution to the overall sum (7.3) is at most

$$\sum_{i=0}^{\infty} \exp\left(-0.013 \cdot 1000^3 \Lambda^3 \cdot 1.01^{3i} n\right) \leqslant \sum_{i=0}^{\infty} \exp\left(-13000000 \cdot 1.03^i\right) = O(1).$$

Case 4. We divide this case into two sub-cases, $\alpha \leqslant 100\Lambda$ and $\alpha > 100\Lambda$.

Sub-case: $\alpha \leqslant 100\Lambda$. If $\beta \leqslant 1000000\Lambda^2$ then Case 1 applies, so we need only consider $\beta > 1000000\Lambda^2$. Break this domain into rectangles,

$$R_j = [0, 100\Lambda] \times [\beta_j^\star, 1.01\beta_j^\star],$$

with $\beta_j^\star = 1000000\Lambda^2 \cdot 1.01^j$, $j = 0, 1, \ldots$; it does no harm to let j run to infinity. For any such rectangle R, writing $\beta^\star$ for $\beta_j^\star$, a crude upper bound on (7.5) is given by

$$\exp\left\{\left(\left[(1+\Lambda)(\sqrt{2}-1)100\Lambda \cdot 1.01\beta^\star\right] - \left[(3\ln(3)-2)\,\left(\beta^\star - (100\Lambda)^2\right)^2 \frac{1}{100\Lambda}\right]\right)n\right\}.$$

Since $\Lambda \leqslant 0.01$ and, by definition of $\beta_0^\star$, $(100\Lambda)^2 \leqslant 0.01\beta^\star$, this is

$$\leqslant \exp\left\{\left(\left[(\sqrt{2}-1)1.01^2(100\Lambda)\beta^\star\right] - \left[(3\ln(3)-2)\,(0.99\beta^\star)^2/100\Lambda\right]\right)n\right\}.$$

Noting that $\beta^\star/100\Lambda \geqslant 10000\Lambda$, and factoring out $\Lambda\beta^\star$, this is

$$\begin{aligned}
&\leqslant \exp\left\{\left(\left[(\sqrt{2}-1)1.01^2 \cdot 100\right] - \left[(3\ln(3)-2)\,0.99^2 \cdot 10000\right]\right)\Lambda\beta^\star n\right\} \\
&\leqslant \exp\left(-12000\Lambda\beta^\star n\right).
\end{aligned}$$

Summing over the rectangles R_j with $\beta_j^\star = 1000000\Lambda^2 \cdot 1.01^j$ gives a contribution to (7.3) of at most

$$\sum_{j=0}^{\infty} \exp\left(-12000\Lambda^3 1.01^j n\right) \leqslant \sum_{j=0}^{\infty} \exp\left(-12000 \cdot 1.01^j\right) = O(1).$$

Sub-case: $\alpha > 100\Lambda$. This case, with $\alpha > 100\Lambda$ and $\beta > 1.03\alpha^2/(8-4\alpha)$, is the most delicate. Observations of such values α, and of β conditioned upon α, are both unlikely, and we need to keep all three terms in (7.5).

In this case, we break down (a superset of) the domain into rectangles

$$R_{ij} = [\alpha_i^\star, 1.001\alpha_i^\star] \times [\beta_{ij}^\star, 1.01\beta_{ij}^\star],$$

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

with $\alpha_i^\star = 100\Lambda \cdot 1.001^i$ and $\beta_{ij}^\star = 1.03(\alpha_i^\star)^2/(8-4\alpha_i^\star) \cdot 1.01^j$, over $i = 0,\ldots,I-1$ and $j = 0,\ldots,J(i)-1$, where I is the first value for which $\alpha_I > 1$, and $J(i)$ the first value for which $\beta_{i,J(i)} > 1$. Within any such rectangle R given by $\alpha^\star = \alpha_i^\star$ and $\beta^\star = \beta_{ij}^\star$, we have $J(1.001\alpha^\star, \beta^\star) = 1$ (from $1.03\alpha^{\star 2}/(8-4\alpha) \geqslant 1.001^2\alpha^{\star 2}/(8-4\cdot 1.001\alpha^\star)$), and thus (7.5) is at most

$$\begin{aligned}\exp\Bigg\{\Bigg(&\left[(1.01)(\sqrt{2}-1)\cdot 1.001\alpha^\star \cdot 1.01\beta^\star\right] - \left[\frac{3\alpha^{\star 3}(1-6/100)}{24-8\alpha^\star}\right] \\ &- \left[(3\ln(3)-2)\left(\beta^\star - \frac{1.001^2\alpha^{\star 2}}{8-4\cdot 1.001\alpha^\star}\right)^2 \frac{1}{1.001\alpha^\star}\right]\Bigg) n\Bigg\} \\ &=: \exp(-f(\alpha^\star,\beta^\star)\, n), \end{aligned} \tag{7.6}$$

and we claim that $f(\alpha,\beta) \geqslant 0.07\alpha\beta$ for all $0 \leqslant \alpha \leqslant 1$, $1.03\alpha^2/(8-4\alpha) \leqslant \beta \leqslant 1$.

Verifying this is fairly straightforward. $f(\alpha,\beta)/(\alpha\beta)$ is, for a given α, extremized either by an extreme value of β – namely $\beta = 1$ or $\beta = 1.03\alpha^2/(8-4\alpha)$ – or by the point where its derivative $\frac{\partial}{\partial\beta}$ vanishes. The derivative can be computed in closed form (a computer-algebra package helps), and is a rational expression whose numerator is a quadratic expression (with no linear term) in β, whose positive root may thus be written down as an explicit function $\beta = \beta(\alpha)$. Substituting $\beta = 1$, $\beta = 1.03\alpha^2/(8-4\alpha)$, or $\beta = \beta(\alpha)$ into $f(\alpha,\beta)/(\alpha\beta)$ yields in each case a function which is tractable over $\alpha \in [0,1]$. When the three are graphed, it is easily seen that the smallest value is achieved at $\alpha = 1$, $\beta = \beta(\alpha)$, where $f(\alpha,\beta)/(\alpha\beta) > 0.07$ (it is about 0.0704). This observation can be confirmed straightforwardly if tediously.

Now, summing the contributions of the rectangles R_{ij} to (7.3) is done as in the previous cases. It is at most

$$\begin{aligned}&\sum_{i=0}^{I}\sum_{j=0}^{J(i)} \exp(-f(\alpha_i^\star,\beta_{ij}^\star)n) \\ &\quad\leqslant \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \exp(-0.07\alpha_i^\star\beta_{ij}^\star n) \\ &\quad\leqslant \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \exp\left(-0.07\cdot(100\Lambda\cdot 1.001^i)\cdot(1.03\cdot[100\Lambda\cdot 1.001^i]^2)/8\cdot 1.01^j\cdot n\right) \\ &\quad\leqslant \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \exp(-9000\,\Lambda^3 1.003^i\, 1.01^j\, n) \\ &\quad\leqslant \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \exp(-9000\cdot 1.003^i\, 1.01^j) \\ &\quad= O(1).\end{aligned}$$

Case 5. If $U' = n$ then $W'(n) \geqslant 0$, and so we must have had at least $n-1$ births among the $\binom{n}{2}$ Bernoulli random variables comprising RW. Denote this (random) number of births by $M' \sim B\left(\binom{n}{2}, c/n\right)$. Let $M \sim B\left(\binom{n}{2}, c/n\right)$ be the number of edges in the whole

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

graph. (M' and M have the same distribution but represent different quantities.) From Theorem 3.3, the amortized running time for v is at most $2^{M/4}$. It suffices to show that $\mathbb{E}\big(2^{M/4}\cdot\mathbf{1}_{M'\geqslant n-1}\big)=o(1)$, where $\mathbf{1}$ denotes the indicator function of an event.

Recall that without loss of generality we are assuming $\Lambda\leqslant 0.01$ and $n\geqslant 10000$ (so $n-1\geqslant 0.9999n$). Thus it suffices to compute an upper bound on

$$\begin{aligned}&\mathbb{E}\big(2^{M/4}\cdot\mathbf{1}_{M'\geqslant 0.9999n}\big)\\&\quad=\mathbb{E}\big(2^{M/4}\cdot\mathbf{1}_{M<1.053n}\,\mathbf{1}_{M'\geqslant 0.9999n}\big)+\mathbb{E}\big(2^{M/4}\cdot\mathbf{1}_{M\geqslant 1.053n}\,\mathbf{1}_{M'\geqslant 0.9999n}\big)\\&\quad\leqslant\mathbb{E}\big(2^{1.053n/4}\cdot\mathbf{1}_{M'\geqslant 0.9999n}\big)+\mathbb{E}\big(2^{M/4}\cdot\mathbf{1}_{M\geqslant 1.053n}\big).\end{aligned}$$

We apply Chernoff's inequality (5.7) to both terms. Note that $\mathbb{E}M'=\mathbb{E}M=\binom{n}{2}\cdot c/n\leqslant\frac{1}{2}\cdot 1.01n=0.505n$.

For the first term, $M'\geqslant 0.9999n$ means that $M'-\mathbb{E}M'\geqslant 0.4949n$, so

$$\begin{aligned}\mathbb{E}\big(2^{1.053n/4}\cdot\mathbf{1}_{M'\geqslant 0.9999n}\big)&=2^{1.053n/4}\Pr(M'\geqslant 0.9999n)\\&\leqslant 2^{1.053n/4}\exp\left(-\frac{(0.4949n)^2}{2\cdot 0.505n+2/3\cdot 0.4949n}\right)\\&\leqslant\exp(1.053\ln(2)\,n/4)\cdot\exp(-0.1827n)\\&\leqslant\exp(-0.0002n)\\&=o(1).\end{aligned}$$

Noting that $\mathbb{E}M-0.505n\leqslant 0$, the second term is at most

$$\begin{aligned}&\sum_{m\geqslant 1.053n}2^{m/4}\Pr(M\geqslant\mathbb{E}M+m-0.505n)\\&\qquad=2^{0.505n/4}\sum_{t\geqslant 0.548n}2^{t/4}\Pr(M\geqslant\mathbb{E}M+t)\\&\qquad\leqslant 2^{0.505n/4}\sum_{t\geqslant 0.548n}2^{t/4}\exp\left(\frac{-t^2}{2\cdot 0.505n+2t/3}\right).\end{aligned}\tag{7.7}$$

Writing $t=\gamma n$, we see that the logarithm of the summand is

$$\frac{t}{4}\ln 2-\frac{t^2}{1.01n+2t/3}=\left(\frac{\gamma\ln 2}{4}-\frac{\gamma^2}{1.01+2\gamma/3}\right)n\leqslant -0.123n$$

for $\gamma\geqslant 0.548$. Since $\Pr(M\geqslant\mathbb{E}M+t)=0$ for $t>\binom{n}{2}$, expression (7.7) is at most

$$\begin{aligned}\tbinom{n}{2}\,2^{0.505n/4}\exp(-0.123n)&\leqslant\tfrac{1}{2}n^2\exp\big((0.505/4\,\ln 2-0.123)n\big)\\&\leqslant\tfrac{1}{2}n^2\exp(-0.03n)\\&=o(1).\end{aligned}$$

Thus, the case $U'=n$ contributes $o(1)$ to the amortized time.

It follows that, in each of the five cases above, the contributions of the corresponding $U'=\alpha n$, $W'=\beta n$ sum to $\exp(O(1+\lambda^3))$, and the result is proved. □

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

8. Conclusions

In the present paper we focus on Max Cut. Our result for random 'sparse' instances is strong in that it applies not only right up to $c = 1$, but through the scaling window $c = 1 + \lambda n^{-1/3}$.

We 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 2-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 Max 2-Sat instances with up to about n clauses can be solved efficiently on average (the Max 2-Sat phase transition occurs around n clauses), our present result for Max 2-Sat is relatively weak. As was the case with Max Cut, it is easy to see that with $m = cn$ and $c < 1$ it is almost always easy to solve Max 2-Sat for a random formula with m clauses on n variables; this follows immediately from the facts that decision 2-Sat is easy and that with high probability such a formula is completely satisfiable. The hard part is to show that it is easy not just with high probability but in expectation.

Since Max Cut is in general NP-hard (and even NP-hard to approximate to better than a 16/17 factor; see Trevisan, Sorkin, Sudan and Williamson [31]), it would be interesting to resolve whether random instances *above* the giant-component threshold can be solved in polynomial expected time (as we have shown for those below the threshold); we conjecture that they cannot. A related question is whether it is possible to solve Max Cut for a random cubic graph in polynomial expected time; again, we conjecture that it is not. It would also be interesting to know if there is a 'mildly exponential' algorithm for such graphs (*e.g.*, one with running time $\exp(O(\sqrt{n}))$. If not, because the 'kernel' of a random graph just above the giant-component threshold is close to being a random cubic graph (see Janson, Łuzcak and Ruciński [20, Section 5.4]), our running time of $O(n)\exp(O(1 + \lambda^3))$ may be the best possible. Questions about average-case hardness are of broad interest, but no quick resolution is in sight.

Acknowledgements

We are grateful to Don Coppersmith for helpful and enjoyable conversations. We also thank the referee for a thorough reading and many constructive comments.

References

[1] Aldous, D. (1997) Brownian excursions, critical random graphs and the multiplicative coalescent. *Ann. Probab.* **25** 812–854.

[2] von Bahr, B. and Martin-Löf, A. (1980) Threshold limit theorems for some epidemic processes. *Adv. Appl. Probab.* **12** 319–349.

[3] Bender, E. A., Canfield, E. R. and McKay, B. D. (1990) The asymptotic number of labeled connected graphs with a given number of vertices and edges. *Random Struct. Alg.* **1** 127–169.

[4] Berry, A. C. (1941) The accuracy of the Gaussian approximation to the sum of independent variates. *Trans. Amer. Math. Soc.* **49** 122–136.

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

[5] Bollobás, B. (1984) The evolution of sparse graphs. In *Graph Theory and Combinatorics* (Cambridge 1983), Academic Press, London, pp. 35–57.
[6] Bollobás, B. (2001) *Random Graphs*, Vol. 73 of *Cambridge Studies in Advanced Mathematics*, Cambridge University Press, Cambridge.
[7] Bollobás, B., Borgs, C., Chayes, J. T., Kim, J. H. and Wilson, D. B. (2001) The scaling window of the 2-SAT transition. *Random Struct. Alg.* **18** 201–256.
[8] Chvátal, V. and Reed, B. (1992) Mick gets some (the odds are on his side). In *33th Annual Symposium on Foundations of Computer Science* (Pittsburgh, PA, 1992), IEEE Comput. Soc. Press, Los Alamitos, CA, pp. 620–627.
[9] Coja-Oghlan, A., Moore, C. and Sanwalani, V. Max k-cut and approximating the chromatic number of random graphs. To appear in *Random Struct. Alg.*
[10] Coja-Oghlan, A. and Taraz, A. (2003) Colouring random graphs in expected polynomial time. In *Proc. STACS 2003*, Vol. 2607 of *Lecture Notes in Computer Science*, Springer, pp. 487–498.
[11] Coppersmith, D., Gamarnik, D., Hajiaghayi, M. T. and Sorkin, G. B. (2004) Random MAX SAT, random MAX CUT, and their phase transitions. *Random Struct. Alg.* **24** 502–545.
[12] Erdős, P. and Rényi, A. (1960) On the evolution of random graphs. *Magyar Tud. Akad. Mat. Kutató Int. Közl.* **5** 17–61.
[13] Erdős, P. and Rényi, A. (1961) On the evolution of random graphs. *Bull. Inst. Internat. Statist.* **38** 343–347.
[14] Fernandez de la Vega, W. (1992) On random 2-SAT. Manuscript.
[15] Gellert, W., Kästner, H., Küstner, H. and Hellwich, M., eds (1977) *The VNR Concise Encyclopedia of Mathematics*, Van Nostrand Reinhold Co., New York.
[16] Goerdt, A. (1996) A threshold for unsatisfiability. *J. Comput. System Sci.* **53** 469–486.
[17] Gramm, J., Hirsch, E. A., Niedermeier, R. and Rossmanith, P. (2003) New worst-case upper bounds for MAX-2-SAT with an application to MAX-CUT. *Discrete Appl. Math.* **130** 139–155.
[18] Hirsch, E. A. (2000) A new algorithm for MAX-2-SAT. In *Proc. STACS 2000* (Lille), Vol. 1770 of *Lecture Notes in Computer Science*, Springer, Berlin, pp. 65–73.
[19] Hoeffding, W. (1963) Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.* **58** 13–30.
[20] Janson, S., Łuczak, T. and Ruciński, A. (2000) *Random Graphs*, Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley-Interscience, New York.
[21] Karp, R. M. (1990) The transitive closure of a random digraph. *Random Struct. Alg.* **1** 73–93.
[22] Kendall, D. G. (1956) Deterministic and stochastic epidemics in closed populations. In *Proc. Third Berkeley Symposium on Mathematical Statistics and Probability, 1954–1955, Vol. IV* (Berkeley and Los Angeles), University of California Press, pp. 149–165.
[23] Krivelevich, M. and Vu, V. H. (2002) Approximating the independence number and the chromatic number in expected polynomial time. *J. Combin. Optim.* **6** 143–155.
[24] Kulikov, A. S. and Fedin, S. S. (2002) Solution of the maximum cut problem in time $2^{|E|/4}$. *Zap. Nauchn. Sem. S.-Peterburg. Otdel. Mat. Inst. Steklov.* (POMI) **293** 129–138, 183.
[25] Łuczak, T. (1990) On the number of sparse connected graphs. *Random Struct. Alg.* **1** 171–173.
[26] Martin-Löf, A. (1986) Symmetric sampling procedures, general epidemic processes and their threshold limit theorems. *J. Appl. Probab.* **23** 265–282.
[27] Niedermeier, R. and Rossmanith, P. (1999) New upper bounds for MaxSat. In *Automata, Languages and Programming* (Prague, 1999), Vol. 1644 of *Lecture Notes in Computer Science*, Springer, Berlin, pp. 575–584.
[28] Scott, A. D. and Sorkin, G. B. (2003) Faster algorithms for MAX CUT and MAX CSP, with polynomial expected time for sparse instances. In *Proc. Random 2003* (Baltimore, MD, 2003), Springer, pp. 382–395.

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press

[29] Scott, A. D. and Sorkin, G. B. Faster exponential-time algorithms for Max 2-Sat, Max Cut, and Max k-Cut. In preparation.
[30] Spencer, J. (1997) Enumerating graphs and Brownian motion. *Comm. Pure Appl. Math.* **50** 291–294.
[31] Trevisan, L., Sorkin, G. B., Sudan, M. and Williamson, D. P. (2000) Gadgets, approximation, and linear programming. *SIAM J. Comput*. **29** 2074–2097.
[32] Wright, E. M. (1980) The number of connected sparsely edged graphs III: Asymptotic results. *J. Graph Theory* **4** 393–407.

https://doi.org/10.1017/S096354830500725X Published online by Cambridge University Press