Optimizing through Co-Evolutionary Avalanches

Stefan Boettcher,[1]* Allon G. Percus,[2]** and Michelangelo Grigni[1]***

[1] Emory University, Atlanta, GA 30322
[2] Los Alamos National Laboratory, Los Alamos, NM 87545

Abstract. We explore a new general-purpose heuristic for finding high-quality solutions to hard optimization problems. The method, called *extremal optimization*, is inspired by "self-organized criticality," a concept introduced to describe emergent complexity in many physical systems. In contrast to *Genetic Algorithms* which operate on an entire "gene-pool" of possible solutions, *extremal optimization* successively replaces extremely undesirable elements of a sub-optimal solution with new, random ones. Large fluctuations, called "avalanches," ensue that efficiently explore many local optima. Drawing upon models used to simulate far-from-equilibrium dynamics, *extremal optimization* complements approximation methods inspired by equilibrium statistical physics, such as *simulated annealing*. With only one adjustable parameter, its performance has proved competitive with more elaborate methods, especially near phase transitions. Those phase transitions are found in the parameter space of most optimization problems, and have recently been conjectured to be the origin of some of the hardest instances in computational complexity. We will demonstrate how *extremal optimization* can be implemented for a variety of combinatorial optimization problems. We believe that *extremal optimization* will be a useful tool in the investigation of phase transitions in combinatorial optimization problems, hence valuable in elucidating the origin of computational complexity.

1 Natural Emergence of Optimized Configurations

Every day, enormous efforts are devoted to organizing the supply and demand of limited resources, so as to optimize their utility. Examples include the supply of foods and services to consumers, the scheduling of a transportation fleet, or the flow of information in communication networks within society or within a parallel computer. By contrast, *without* any intelligent organizing facility, many natural systems have evolved into amazingly complex structures that optimize the utilization of resources in surprisingly sophisticated ways [2]. For instance, biological evolution has developed efficient and strongly interdependent networks in which resources rarely go to waste. Even the inanimate morphology of natural landscapes exhibits patterns far from random that often seem to serve a purpose, such as the efficient drainage of water [31].

* e-mail: stb@physics.emory.edu
** e-mail: percus@lanl.gov
*** e-mail: mic@mathcs.emory.edu

Natural systems that exhibit such self-organizing qualities often possess common features: they generally consist of a large number of strongly coupled entities with very similar properties. Hence, they permit a statistical description at some coarse level. An external resource (sunlight in the case of evolution) drives the system which then takes its direction purely by chance. Like flowing water breaking through the weakest of all barriers in its wake, species are coupled in a global comparative process that persistently washes away the least fit. In this process, unlikely but highly adapted structures surface inadvertently. Optimal adaptation thus emerges naturally, without divine intervention, from the dynamics through a selection *against* the extremely "bad". In fact, this process prevents the inflexibility inevitable in a controlled breeding of the "good".

Certain models relying on extremal processes have been proposed to explain self-organizing systems in nature [28]. In particular, the Bak-Sneppen model of biological evolution is based on this principle [3, 10]. It is happily devoid of any specificity about the nature of interactions between species, yet produces salient nontrivial features of paleontological data such as broadly distributed lifetimes of species, large extinction events, and punctuated equilibrium.

In the Bak-Sneppen model, the high degree of adaptation of most species is obtained by the elimination of badly adapted ones instead of a particular "engineering" of better ones. Species in the Bak-Sneppen model are located on the sites of a lattice, and each is represented by a value between 0 and 1, indicating its "fitness". At each update step, the smallest value (representing the worst adapted species) is discarded and replaced with a new value drawn randomly from a flat distribution on $[0, 1]$. But the change in fitness of one species impacts the fitness of an interrelated species. Therefore, at each update step in the Bak-Sneppen model, the fitness values on the sites *neighboring* the smallest value are replaced with new random numbers as well. No explicit definition is given of the mechanism by which these neighboring species are related. Yet, after a certain number of updates, the system organizes itself into a highly correlated state known as self-organized criticality (SOC) [4]. In that state, almost all species have reached a fitness above a certain threshold. These species possess *punctuated equilibrium:* one's weakened neighbor can undermine one's own fitness. Co-evolutionary chain reactions called "avalanches" ensue; large fluctuations that make any possible configuration accessible.

2 Extremal Optimization

Extremal Optimization (EO) is inspired by previous attempts of using physical intuition to optimize. It opens the door to applying *non-equilibrium processes*, such as SOC, in the same manner simulated annealing (SA) [23] applies equilibrium statistical mechanics. The result is a general method that appears to be a powerful addition to the canon of meta-heuristics [27]. Its large fluctuations provide significant hill-climbing ability, which enables EO to perform well at phase transitions, "where the really hard problems are" [11, 1].

One popular hard optimization problem, to which we have applied EO successfully (see below and Refs. [8, 7]), is the graph bi-partitioning problem (GBP) [14, 23, 21]. In the GBP, we are given a set of n vertices, where n is even, and "edges" connecting certain pairs of vertices. The problem is to partition the vertices into two equal subsets, each of size $n/2$, with a minimal number of edges cutting across the partition. The size of the configuration space Ω grows exponentially with n, $|\Omega| = \binom{n}{n/2}$, since all unordered divisions of n vertices into to equal-sized sets are feasible configurations S. The cost function $C(S)$ (called "cutsize") counts the number of "bad" edges that cut across the partition. A typical neighborhood $N(S)$ for a *local search* [27, 30], mapping $S \to S' \in N(S) \subset \Omega$, is a "1-*exchange*" of one randomly chosen vertex from each subset.

EO performs in general a search on a single configuration $S \in \Omega$. S usually consists of a large number n of variables x_i. The cost $C(S)$ is assumed to consist of the individual cost contributions λ_i for each variable x_i, which correspond loosely to the "fitness" values in the Bak-Sneppen model above. Typically, the fitness λ_i of variable x_i depends on its state in relation to other variables that x_i is connected to. Ideally, it is

$$C(S) = \sum_{i=1}^{n} \lambda_i. \tag{1}$$

For example, in the GBP the variables x_i are the vertices, each being assigned to a set "0" or "1." Each vertex has edges connecting it to a certain number of other vertices. Eq. (1) for the cutsize $C(S)$ is satisfied, if we attribute to each vertex x_i a local cost $\lambda_i = b_i/2$, where b_i is the number of "bad" edges, whose cost is equally shared with the vertex on the other end of that edge.

For minimization problems in general, EO proceeds as follows:

1. Initialize a configuration S at will; set $S_{\rm best} = S$.
2. For the "current" configuration S,
   (a) evaluate λ_i for each variable x_i,
   (b) find j with $\lambda_j \geq \lambda_i$ for all i, i. e. x_j has the "worst fitness,"
   (c) choose at random a $S' \in N(S)$ such that the "worst" x_j must change its state,
   (d) if $C(S') < C(S_{\rm best})$ then set $S_{\rm best} = S'$,
   (e) accept $S \leftarrow S'$ *always, independent* of $C(S') - C(S)$.
3. Repeat at step (2) as long as desired.
4. Return $S_{\rm best}$ and $C(S_{\rm best})$.

The algorithm operates on a single configuration S at each step. All variables x_i in S have a fitness, of which the "worst" is identified. This *ranking* of the variables according to *individual* costs – unique to EO – provides the only measure of quality on S. It implies that all other variables are "better" in the current S. There is *no parameter to be adjusted* for the selection of better solutions aside from this ranking. In fact, it is only the memory encapsulated in this ranking that directs EO into the neighborhood of increasingly better solutions. Those "better" variables only possess punctuated equilibrium: their memory

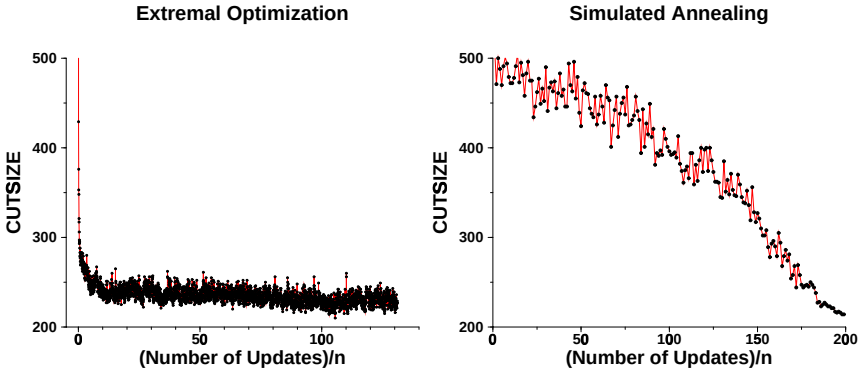

Fig. 1. Evolution of the cutsize $C(S)$ during a typical run of (a) EO and (b) SA for the $n = 500$, $c \approx 5$, random graph G_{500} introduced in Ref. [21]. The best cutsize ever found for G_{500} is 206 (see Fig. 2). In contrast to SA, which has large fluctuations in early stages of the run and then converges much later, extremal optimization quickly approaches a stage where broadly distributed fluctuations allow it to scale barriers and probe many local optima.

gets erased when they happen to be connected to one of the variables forced to change. On the other hand, in the choice of move to S', *no consideration to the outcome* of such a move is given, and not even the worst variable itself is guaranteed to improve its fitness. Large fluctuations in the cost accumulate over many updates [3], while merely the bias *against* "bad" fitnesses guides EO back towards improved solutions, see Fig. 1.

Disadvantages of EO are that a definition of fitness for individual variables may be ambiguous or even impossible. Also, variables may be strongly connected such that each update destroys more well-adapted variables than it could ever hope to improve [8]. In highly connected systems, EO is slowed down considerably by reevaluating fitnesses [step (2a)]. For many problems, these disadvantages do not apply or are surmountable. In particular, problems in the important optimization class MAX-SNP [29] fit naturally into the EO-framework. MAX-SNP problems have boolean variables and a collection of bounded-arity boolean terms and we seek an assignment satisfying as many (or as few) terms as possible. Such problems have a natural choice of fitness functions, and typically have low variable connectivity. Indeed, some complete problems for the class have bounded connectivity in the worst case. MAX-SNP complete problems include MAX-K-SAT, K-COL, and MAXCUT (similar to GBP), discussed below.

3 Comparison with other Heuristics

The most apparent distinction between EO and other methods is the need to define local cost contributions for each variable, instead of merely a global cost.[1]

[1] Apparently, local costs have previously been used in an otherwise unrelated ensemble Monte Carlo approach [12].

EO's capability appears to derive from its ability to access this local information directly. EO's ranking of fitnesses superficially appears like the rankings of possible moves in some versions of SA [17, 30] and in Tabu search [15, 30]. But these moves are evaluated by their *anticipated outcome*, while EO's fitnesses reflect the *current* configuration S without biasing the outcome.

Simulated Annealing (SA): SA [23] emulates the behavior of frustrated systems in *thermal equilibrium:* if one couples such a system to a heat bath of adjustable temperature, by cooling the system slowly one may come close to attaining a state of minimal energy (i. e. cost). SA accepts or rejects local changes to a configuration according to the Metropolis algorithm, requiring equilibrium conditions ("detailed balance") along a well-tuned "temperature schedule."

In contrast, EO drives the system *far from equilibrium:* aside from ranking, it applies no decision criteria, and all new configurations are accepted indiscriminately. Instead of tuning a whole schedule of parameters, EO often requires few choices. It may appear that EO's results should resemble an ineffective random search, similar to SA at a fixed but finite temperature. But in fact, by persistent selection against the worst fitnesses, one quickly approaches near-optimal solutions. Significant fluctuations still remain at late run-times (unlike in SA, see Fig. 1), crossing sizable barriers to access new regions in configuration space.

Genetic Algorithms (GA): While similarly motivated, GA [20, 16] and EO algorithms have hardly anything in common. GAs, mimicking evolution on the genotypical level, keep track of entire "gene pools" of configurations from which to select and "breed" an improved generation of solutions. By comparison, EO, based on evolutionary competition at the phenomenological level of "species," operates only on a single configuration, with improvements achieved merely by elimination of bad variables. EO, SA, and most other meta-heuristics perform a local search but in GA cross-over operators perform global exchanges.

4 Applications of Extremal Optimization

Ground States of Spin Glasses: A simple version of a spin glass [25] consists of a d-dimensional hyper-cubic lattice with a spin variable $\sigma_i \in \{-1, 1\}$ placed on each site i, $1 \leq i \leq n = L^d$. Every spin is connected to each of its nearest neighbors j via a bond variable $J_{i,j}$ drawn from some distribution $P(J)$ of zero mean and unit variance. Spins may be coupled to an arbitrary external field h_i. We try to find "ground states," i. e. lowest energy configurations $S_{\min}$ of

$$C(S) = H(\sigma_1, \ldots, \sigma_n) = -\frac{1}{2} \sum_i \sum_j J_{i,j} \sigma_i \sigma_j - \sum_i \sigma_i h_i. \tag{2}$$

Arranging the spins into optimal configurations is hard due to "frustration" [25]. To implement EO, we define as fitness the local energy for each spin

$$\lambda_i = -\sigma_i \left(\frac{1}{2} \sum_j J_{i,j} \sigma_j + h_i \right), \tag{3}$$

and Eq. (2) turns into Eq. (1). Our implementation suggests that EO may be well suited for problems representable as a spin-Hamiltonian [25].

Satisfiability (MAX-K-SAT): Instances of the satisfiability problem MAX-K-SAT consist of a formula composed of M clauses. Each clause contains K literals (i. e. x_i or $\neg x_i$), drawn randomly from a pool of n boolean variables x_i. A clause is verified, if at least one of its K literals is true (logical "or"), and the entire formula is verified only if every clause is true (logical "and"). Here, we try to maximize the number of true clauses by some configuration of the variables.

MAX-K-SAT has an obvious EO-implementation: For each variable we set $\lambda_i = 1/K \times \{\#$ of false clauses containing $x_i\}$. Again, Eq. (1) holds. Typically, $K = O(1)$ and $M = O(n)$ so that each variable appears only in a few ($\approx M/n$) clauses, each connecting it to $\sim K$ other variables. The phase transition in 2-SAT and 3-SAT has been investigated in Refs. [26, 1] on small instances using exact methods. We expect that EO would perform very well on those instances.

Graph Coloring (K-COL): Given K different colors to label the vertices of a graph, we need to find a coloring that minimizes the number of edges connecting vertices of identical color. We implement EO for K-COL by defining for each vertex the number of equally colored vertices connected to it as fitness. Similar to a spin glass, this problem is hard due to *local* frustration [25], in distinction to the global constraints in the GBP. A simple neighborhood consists of the re-coloring of a single vertex each update. Below, we present results of using EO in analyzing the phase transition in 3-COL, first investigated in Refs. [11, 1].

5 Experimental Results

Simple EO Application to Graph Partitioning: Following Ref. [21] (Fig. 9 there), we tested early implementations of EO [8] on their $n = 500$ random graph G_{500} of connectivity $c \approx 5$. In a 1000-run sample from different random initial conditions, we determined the frequency of solution obtained, see Fig. 2. For comparison, we have also implemented the SA algorithm as given in Ref. [21] on the same data structure used by our EO program. We have allowed runtimes for EO about three times longer than the time it took for SA to "freeze," since EO still obtained significant gains. We checked that neither the best-of-three runs of SA, or a three times longer temperature schedule, improved the SA results significantly. While the basic, parameter-free version of EO from Sec. 2 is already competitive, the best results are obtained by τ-EO.

τ-EO Implementation: τ-EO is a general modification of EO which improves results and avoids "dead ends" that occur in some implementations at the expense of introducing a single parameter [8]. We rank all the variables x_i according to fitness λ_i, i. e. find a permutation Π of the vertex labels i such that

$$\lambda_{\Pi(1)} \geq \lambda_{\Pi(2)} \geq \ldots \geq \lambda_{\Pi(n)}. \tag{4}$$

The worst variable x_j [see step (2b)] is of rank 1, $j = \Pi(1)$, and the best variable is of rank n. Consider a probability distribution over the *ranks* k,

$$P_k \propto k^{-\tau}, \qquad 1 \leq k \leq n, \tag{5}$$

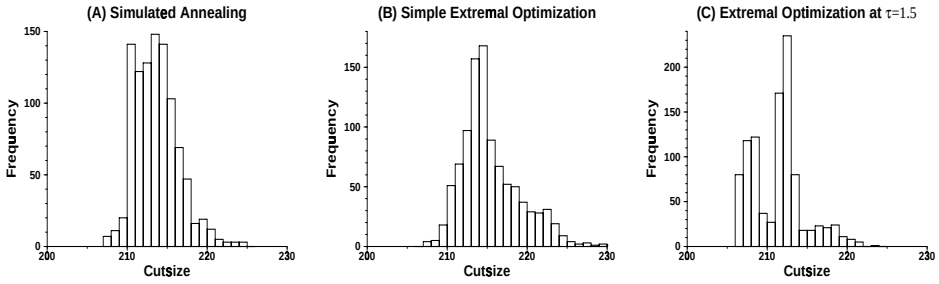

Fig. 2. Comparison of 1000-run trials using various optimization methods on a $n = 500$ random graph with $c = 5$. The histograms give the frequency with which a particular cutsize has been obtained during the trial runs for (A) SA, (B) basic EO, and (C) for τ-EO with $\tau = 1.5$. The best cutsize ever found for this graph is 206. This result appeared only once over the 1000 SA runs, but occurred 80 times for τ-EO.

for a fixed value of the parameter τ. On each update, for each independent variable x to be moved, select distinct ranks $k_1, k_2, \ldots$ according to P_k. Then, execute step (2c) such that all $x_{i_1}, x_{i_2}, \ldots$ with $i_1 = \Pi(k_1)$, $i_2 = \Pi(k_2), \ldots$ change. For instance, in the bi-partitioning problem, we choose *both* variables in the 1-*exchange* according to P_k, instead of the worst and a random one. Although the worst variable of rank $i = 1$ will be chosen most often, sometimes (much) higher ranks will be updated instead. In fact, the choice of a power-law distribution[2] for P_k ensures that no rank gets excluded from further evolution while maintaining a bias against variables with bad fitness.

Clearly, for $\tau = 0$, τ-EO is exactly a random walk through Ω. Conversely, for $\tau \to \infty$, the process approaches a deterministic local search, only swapping the lowest-ranked variables, and is bound to reach a "dead end." Indeed, tests of both, $\tau = 0$ and $\tau = \infty$, yield terrible results! In the GBP, we obtained our best solutions for $\tau \approx 1.4 - 1.6$. Under preliminary testing we find that there may be a link between the *optimal* choice for the parameter τ and a transition to "non-ergodic" behavior in the sense that for larger values of τ certain configurations in Ω may become inaccessible during the time of a complete EO-run. In fact, on the basis of that observation we have developed an argument to approximate $\tau \sim 1 + \ln(A)/\ln(n)$ [9] where $t = An$ with $1 \ll A \ll n$ is the runtime. (Typically, we use $A \approx 10^2$ for graphs of size $n \approx 10^4$, consistent with $\tau = 1.5$.) Tests with longer runtimes indeed favor larger τ values, while larger graphs require smaller values of τ.

Results on Large Graphs: In Tab. 1 we summarize τ-EO's results on large-n graphs, using $\tau = 1.4$ and best-of-10 runs. On each graph, we used as many update steps t as appeared productive for EO to reliably obtain stable results. This varied with the particularities of each graph, from $t = 2n$ to $200n$, and the reported runtimes are of course influenced by this. It is worth noting, though, that EO's *average* performance has been varied. For instance, half of the *Brack2*

[2] instead of, say, an exponential distribution with a cut-off scale excluding high ranks.

Table 1. Best cutsizes (and allowed runtime) for a testbed of large graphs. GA results are the best reported [24] (with a 300MHz CPU). τ-EO results are from our runs (200MHz). Comparison data for three of the large graphs are due to results from heuristics in Ref. [19] (50MHz). METIS is a partitioning program based on hierarchical reduction instead of local search [22], obtaining extremely fast deterministic results (200MHz). Runtimes comparisons here are at best qualitative.

Large Graph		GA	τ-EO	[19]	METIS
Hammond	($n = 4720$; $c = 5.8$)	90 (1s)	90 (42s)	97 (8s)	92 (0s)
Barth5	($n = 15606$; $c = 5.8$)	139 (44s)	139 (64s)	146 (28s)	151 (0.5s)
Brack2	($n = 62632$; $c = 11.7$)	731 (255s)	731 (12s)	—	758 (4s)
Ocean	($n = 143437$; $c = 5.7$)	464 (1200s)	464 (200s)	499 (38s)	478 (6s)

runs returned cutsizes near 731, but the other half returned cutsizes of above 2000. This may be a product of an unusual structure in these particular graphs.

Phase Transitions in Combinatorial Optimization: In extensive numerical studies [7] we have shown that τ-EO outperforms SA near phase transitions where graphs begin to "percolate" and cutsizes first become non-zero, see Fig. 3. Studies on the average rate of convergence towards better-cost configurations as a function of runtime t indicate power-law convergence [18], roughly like $C(S_{\rm best})_t \sim C(S_{\rm min}) + A\,t^{-\beta}$ with $\beta \approx 0.45$. Of course, it is not easy to assert for graphs of large n that those runs in fact converge close to the optimum $C(S_{\rm min})$, but finite-size scaling analysis seems to justify that expectation [9].

In an even more impressive performance, we used EO to completely enumerate *all* optimal solutions $S_{\rm min}$ near the critical point for random graphs in 3-COL. Instances of random graphs typically have a high ground-state degeneracy, i. e. possess a large number of equally optimal solutions $S_{\rm min}$. In Ref. [26] it was shown that at the phase transition of 3-SAT the fraction of *constrained vari-*

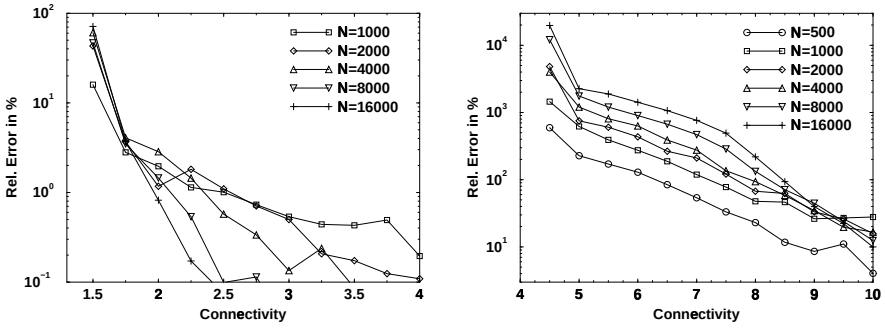

Fig. 3. Plot of the error in the best result of SA relative to EOs on identical instances of (a) random graphs and (b) geometric graphs as function of the mean connectivity c. The percolation points are at (a) $c = 1$ [13] and (b) $c \approx 4.5$ [5], the critical points for the GBP (the first time a component of size $> n/2$ appears) are slightly above that [e. g. at $c = 2\ln 2 = 1.386$ for (a)]. SA's error relative to EO near the critical point in each case rises with n.

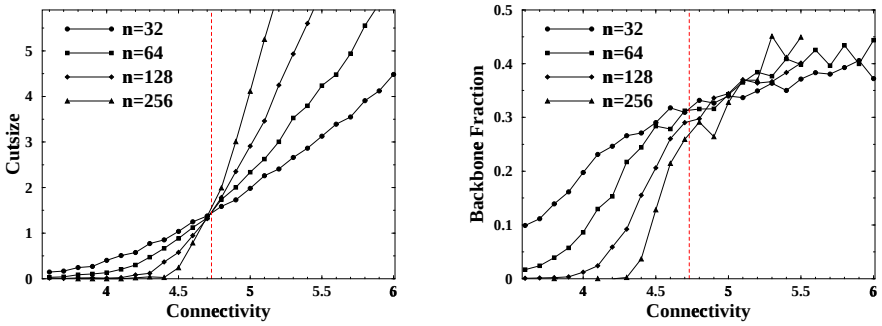

Fig. 4. Plot of the average (a) cutsize and (b) backbone fraction as a function of the connectivity c for random graph 3-COL. We have generated 2, 300, 500, 280 and 125 instances for $n = 32$, 64, 128, and 256, respectively, at each value of c. The prediction for the critical point of $c_{\rm crit}(3) \approx 4.73$ is indicated by a vertical line. The backbone fraction seems to develop a finite jump at the critical point for $n \to \infty$.

ables, i. e. those that are found in an identical state in *all* $S_{\rm min}$, discontinuously jumps to a non-zero value. It was conjectured that the *first-order* phase transition in this "backbone" would exist for any NP-hard problem. To test those claims for 3-COL, we generated a large number of random graphs and explored Ω for as many ground states as τ-EO could find. We fixed runtimes at $\approx 100n^2$, well above the times needed to saturate the set of all $S_{\rm min}$ in repeated trails on some test instances. Such long runtimes favored a large value of $\tau = 2.7$. For each instance, we measured the cutsize, entropy, and the "backbone." Due to the symmetry under interchanging colors, the backbone here consists of the fraction of constrained *pairs* of vertices, i. e. those which are in the same relative state (same or opposite color) in *all* ground states. Averaged results are given in Figs. 4a-b. As predicted in Ref. [26], asymptotically for large n the backbone fraction seems to jump discontinuously at the critical connectivity, $c_{\rm crit} \approx 4.73$.

This work was supported in part by the URC at Emory University, NSF grant CCR-9820931, and an LDRD grant from Los Alamos National Laboratory.

References

1. See *Frontiers in problem solving: Phase transitions and complexity,* Special issue of Artificial Intelligence **81**:1–2 (1996).
2. P. Bak *How Nature Works* (Springer, New York, 1996).
3. P. Bak and K. Sneppen, *Punctuated Equilibrium and Criticality in a simple Model of Evolution,* Phys. Rev. Lett. **71**, 4083-4086 (1993).
4. P. Bak, C. Tang, and K. Wiesenfeld, *Self-Organized Criticality,* Phys. Rev. Lett. **59**, 381 (1987).
5. I. Balberg, *Universal percolation-threshold limits in the continuum,* Phys. Rev. B **31**, R4053-4055 (1985).
6. S. Boettcher and A. G. Percus, *Extremal Optimization: Methods derived from Co-Evolution,* in *GECCO-99* (Morgan Kaufmann, San Francisco, 1999), 825-832.

7. S. Boettcher, *Extremal Optimization and Graph Partitioning at the Percolation Threshold,* J. Math. Phys. A: Math. Gen. **32**, 5201-5211 (1999).
8. S. Boettcher and A. G. Percus, *Nature's Way of Optimizing,* Artificial Intelligence **119**, 275-286 (2000).
9. S. Boettcher and A. G. Percus, (in preparation).
10. S. Boettcher and M. Paczuski, *Ultrametricity and memory in a solvable model of self-organized criticality,* Phys. Rev. E **54**, 1082-1095 (1996).
11. P. Cheeseman, B. Kanefsky, and W. M. Taylor, *Where the really hard Problems are,* in Proc. of IJCAI-91, eds. J. Mylopoulos and R. Rediter (Morgan Kaufmann, San Mateo, CA, 1991), 331-337.
12. F.-M. Dittes, *Optimization on Rugged Landscapes: A New General Purpose Monte Carlo Approach,* Phys. Rev. Lett. **76**, 4651-4655 (1996).
13. P. Erdös and A. Rényi, in: *The Art of Counting*, ed. J. Spencer (MIT, Cambridge, 1973).
14. M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness* (W. H. Freeman, New York, 1979).
15. F. Glover, *Future Paths for Integer Programming and Links to Artificial Intelligence,* Computers & Ops. Res. **5**, 533-549 (1986).
16. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning,* (Addison-Wesley, Reading, 1989).
17. J. W. Greene and K. J. Supowit, *Simulated Annealing without rejecting moves,* IEEE Trans. on CAD **5**, 221-228 (1986).
18. G. S. Grest, C. M. Soukoulis, and K. Levin, *Cooling-rate dependence for the spin-glass ground-state energy: Implications for optimization by simulated annealing,* Phys. Rev. Lett. **56**, 1148 (1986).
19. B. A. Hendrickson and R. Leland, *A multilevel algorithm for partitioning graphs,* in: *Supercomputing '95*, San Diego, CA (1995).
20. J. Holland, *Adaptation in Natural and Artificial Systems* (University of Michigan Press, Ann Arbor, 1975).
21. D. S. Johnson et al.,*Optimization by Simulated Annealing - an Experimental Evaluation. 1. Graph Partitioning,* Operations Research **37**, 865-892 (1989).
22. G. Karypis and V. Kumar, *METIS, a Software Package for Partitioning Graphs,* see http://www-users.cs.umn.edu/~karypis/metis/main.shtml,
23. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Optimization by simulated annealing,* Science **220**, 671-680 (1983).
24. P. Merz and B. Freisleben, *Memetic algorithms and the fitness landscape of the graph bi-partitioning problem,* Lect. Notes Comput. Sc. **1498** 765-774 (1998).
25. M. Mezard, G. Parisi, and M. A. Virasoro, *Spin Glass Theory and Beyond* (World Scientific, Singapore, 1987).
26. R. Monasson et al.,*Determining computational complexity from characteristic 'phase transitions,'* Nature **400**, 133-137 (1999).
27. *Meta-Heuristics: Theory and Application,* Eds. I. H. Osman and J. P. Kelly (Kluwer, Boston, 1996).
28. M. Paczuski, S. Maslov, and P. Bak, *Avalanche dynamics in evolution, growth, and depinning models,* Phys. Rev. E **53**, 414-443 (1996).
29. C. H. Papadimitriou and M. Yannakakis, *Optimization, Approximation, and Complexity Classes,* Journal of Computer and System Sciences **43**, 425-440 (1991).
30. *Modern Heuristic Techniques for Combinatorial Problems,* Ed. C. R. Reeves (Wiley, New York, 1993).
31. I. Rodriguez-Iturbe and A. Rinaldo, *Fractal river basins: chance and self-organization* (Cambridge, New York, 1997).