Editors: Harvey Gould, hgould@physics.clark.edu
Jan Tobochnik, jant@kzoo.edu

EXTREMAL OPTIMIZATION: HEURISTICS VIA COEVOLUTIONARY AVALANCHES

By Stefan Boettcher

IMAGINE THAT YOU WANT TO DESIGN SOME CIRCUITRY FOR A COMPUTER. THE LOGICAL FUNCTION YOU HAVE TO IMPLEMENT REQUIRES A KNOWN NETWORK OF n INTERRELATED LOGICAL GATES. UNFORTUNATELY, n IS TOO LARGE TO PUT ALL THE gates on a single integrated circuit, so you have to partition the gates between separate circuits. Let's assume that you are forced to place exactly $n/2$ gates each on two integrated circuits. The connections between the gates across the partition are slow, energy consuming, and heat producing, while the cost associated with connections inside an integrated circuit are negligible. So, you want to divide the network of gates such that the cost function C, the number of connections cutting across the partition, is minimized (see Figure 1). Because a million computers will be running almost nonstop for 10 years, removing even one costly connection would be worthwhile.

Fortunately, this (simplified) problem can be mapped onto the well-known graph-bipartitioning problem. In this problem, the n gates are the vertices of a graph with edges between two connected gates. Each vertex is a Boolean variable, with state "0" if placed on the left integrated circuit and state "1" if placed on the right integrated circuit. Although the graph of connections is fixed, the vertices can be moved so that we may obtain a good partition. Unfortunately, optimizing the equal partition is NP-hard; that is, the computations needed to find the global optimum with certainty for even the cleverest algorithm grow faster than any power of n. This computation would become unreasonable for about $n \gtrsim 10^3$.

Instead, we can "search" the space of all feasible (equal) partitions Ω. Because the configurations $S \in \Omega$ so far are unrelated, we need to define a "neighborhood" $N(S) \subset \Omega$ for each S, a way to proceed from the current configuration S to some neighboring configuration $S' \in N(S)$.[1] A simple neighborhood N for this problem is a "1-exchange," which consists of all $S' \in \Omega$ obtained from S by changing a 0-vertex to 1 and a 1-vertex to 0 (to maintain an equal partition). The neighborhood N provides Ω with a metric such that the cost function $C(S)$ exhibits local extrema, like a (high-dimensional) mountain landscape. Then, moving sequentially "downhill" to better configurations, we should reach a local minimum very quickly. However, in NP-hard optimization problems, the number of suboptimal minima of the cost function grows nearly as fast as the number of configurations, $|\Omega|$, which here grows like

$$|\Omega| = \binom{n}{n/2} \sim 2^n / \sqrt{n} \ .$$

Thus, in this approach there is no way to move the system from the current minimum to a better one; it's like trying to find the lowest point in a mountainous landscape at night.

In this case, our "heuristic" (derived from the Greek word for "find") produces merely approximate solutions: local minima of dubious quality. Can we do better? If we had more time, we could use an algorithm that alternates downhill moves with a small random change of the current local-minimum configuration. This change might take the system over a "mountain range" such that a subsequent descent would provide a new local minimum. A sequence of these stochastic updates can only increase our chances that our search passes through a better minimum.

A good stochastic search, while inherently slow, succeeds by controlling the right mixture of descending moves with "hill-climbing" perturbations. A particularly elegant stochastic optimization heuristic is simulated annealing.[2] To implement simulated annealing for the graph-bipartitioning problem, we can adopt the 1-exchange neighborhood by choosing two vertices at random at each update. If we accept only 1-exchanges that lower the cost, the system converges to a (likely poor) local minimum and no further improvement is possible for any pair of vertices that are chosen. In contrast, simulated annealing allows moves that raise the cost according to the Metropolis algorithm. In each update, a 1-exchange is accepted with probability $p = \min\{1, e^{-\Delta/T}\}$, where $\Delta = C(S') - C(S)$ is the difference in cost between the new and the old configuration.

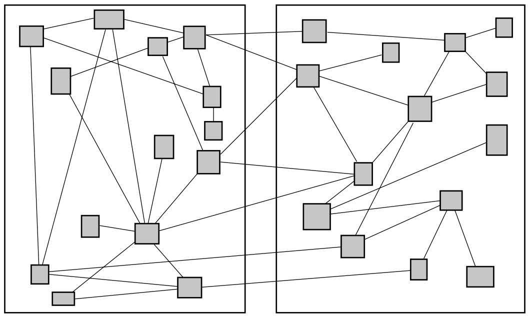


Figure 1. Schematic of two integrated circuits, each with an equal number of logical gates (represented by gray boxes). The graph of connections between gates is fixed, but the gates can be moved onto either integrated circuit. Find an equal partition of the gates such that there is a minimum of connections between the integrated circuits. How would you verify that a configuration is a global optimum?

Controlling the "temperature" T is crucial for simulated annealing to succeed: if T is too large, every uphill move is accepted and no minima are found at all, while for small T only downhill moves are accepted and the system quickly freezes into a local minimum. Instead, mimicking the annealing process designed to harden alloys, T is lowered slowly, allowing simulated annealing to explore the configuration space landscape widely at high T to reach the largest "valley." In turn, this valley may harbor a correspondingly lower minimum for the system to freeze into at smaller T. If T is changed slowly, the Metropolis algorithm ensures thermodynamic equilibrium (which means that each move is as likely to take place as its reverse, a state referred to as *detailed balance*).

Because equilibrium systems are well understood, we have an enormous amount of knowledge to guide simulated annealing. Theoretically, simulated annealing will always converge to the global optimum.[3] Unfortunately, this convergence requires vanishingly small decrements of T; about as many updates are needed as in an exhaustive search of Ω. Short of that, it is an art to devise a temperature schedule that balances computational efficiency with the quality of the minima that are found.[1] Despite its shortcomings, simulated annealing is conceptually elegant and often highly successful for practical problems about which little else is known. Depending on the structure of the graph, it may be useful for our graph-bipartitioning problem.[4,5] (Typically, simulated annealing works well for highly connected graphs, but can only get within an order of magnitude of the best-known minima for many geometrical graphs.)

But could we improve our procedure by throwing the equilibrium requirement overboard? The behavior of simulated annealing soon becomes difficult to predict for lack of any theoretical guidance. Although most processes in nature are out of equilibrium, our understanding of these processes is incomplete. Thus, researchers typically bypass physical considerations entirely and move to a more abstract conception of a natural process for inspiration of an optimization procedure.

This is clearly the case for genetic algorithms, which mimic evolution by natural selection.[6] The way that genetic material evolves and replicates is poorly understood in physical terms, but how genes selectively optimize themselves is readily transcribed into an optimization algorithm. Genetic algorithms consist of a collection of binary-encoded strings (the genotypes) and a mapping that takes each genotype to a configuration (the phenotype) such as a specific partitioning in the graph problem. During each step of the algorithm the population of genotypes is modified (by mutations and crossover operators that exchange sections of the strings representing two genotypes), and a few genotypes that lead to the best phenotypes (that is, the lowest cost function) are selected. Without the theoretical guidance that simulated annealing possesses, the values of the parameters that control the working of genetic algorithms are chosen mostly by trial and error. However, with some empirical knowledge, genetic algorithms also prove to be a powerful optimization procedure with many successful applications.[7]

Emergence and self-organized criticality

What can we learn by focusing on the physics of nonequilibrium processes? During the past decade, some physicists have become interested in systems exhibiting self-organized criticality, in which complex patterns emerge without the need to control any parameters.[8] For instance, biological evolution has developed, apparently by chance, efficient networks in which resources rarely go to waste. But 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 emerges naturally, without 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."

This coevolutionary process is the basis of the Bak-Sneppen model, where the high degree of adaptation of most species is obtained by the elimination of badly adapted ones instead of the engineering of better ones.[9,10] Species in the Bak-Sneppen model are sites of a lattice, and each is represented by a value between 0 and 1, indicating its fitness. At each update, 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]. Because the change in fitness of one species impacts the fitness of interrelated species, at each update of the Bak-Sneppen model, the fitness values on the sites neighboring the smallest value are replaced with new random numbers as well. After a certain number of updates, the system organizes itself into a highly correlated state characteristic of self-organized criticality.[11] In this state, almost all species have

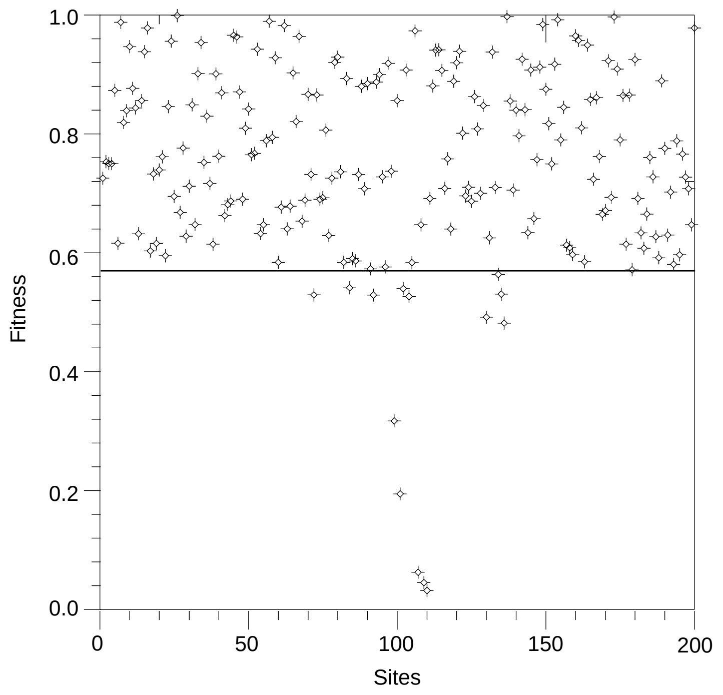


Figure 2. Snapshot during the evolution of the Bak-Sneppen model, showing the fitnesses of a 200-species system. Almost all species have developed fitnesses above a self-organized threshold (the horizontal line), while a small number of currently active species have fitnesses below.

reached a fitness above a certain threshold (see Figure 2). But chain reactions, called *avalanches*, produce large, nonequilibrium fluctuations in the configuration of fitness values. The result is that any possible configuration is accessible.

Extremal optimization

The extremal dynamics of the Bak-Sneppen model can be converted into an optimization algorithm called *extremal optimization*.[5] Attractive features of the model include the following:

- It is straightforward to relate the sum of all fitnesses to the cost function of the system.
- In the self-organized critical state to which the system inevitably evolves, almost all species have a much better than random fitness (see Figure 2).
- Most species preserve a good fitness for long times unless they are connected to poorly adapted species, providing the system with a long memory.[12]
- The system retains a potential for large, hill-climbing fluctuations at any stage.
- The model accomplishes these features without any control parameters.

To be precise, we define $S = (x_1, \ldots, x_n) \in \Omega$ to be a configuration of the n variables x_i in an optimization problem. For instance, in the graph-bipartitioning problem the variables x_i are the vertices, which can take on the values 0 or 1; a configuration S is one possible arrangement of $n/2$ 0's and $n/2$ 1's. The cost function $C(S)$ simply counts the number of bad edges that connect a 0 with a 1 in S. Finally, we define a neighborhood $N(S)$ that maps $S \to S' \in N(S) \subset \Omega$ to facilitate a local search, like the 1-exchange for the graph-bipartitioning problem.

Extremal optimization performs a search through sequential changes on a single configuration $S \in \Omega$. The cost $C(S)$ is assumed to consist of the individual cost contributions λ_i for each variable x_i, which correspond to the fitness values in the Bak-Sneppen model. Typically, the fitness λ_i of variable x_i depends on its state in relation to other variables to which x_i is connected. Ideally, it is possible to write the cost function as

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

For example, in the graph-bipartitioning problem, Equation 1 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 of x_i, whose cost is equally shared with the vertices on the other end of those edges.

For minimization problems, extremal optimization proceeds as follows:

```
1. Initialize a configuration $S$ at will and set $S_{\text{best}} = S$.
2. For the current configuration $S$,
   a. evaluate $\lambda_i$ for each variable $x_i$;
   b. find $j$ with $\lambda_j \geq \lambda_i$ for all $i$; that is, $x_j$ has the worst fitness;
   c. choose a random $S' \in N(S)$ such that $x_j$ must change;
   d. if $C(S') < C(S_{\text{best}})$, store $S_{\text{best}} = S'$;
   e. accept $S := S'$ unconditionally.
3. Repeat at step 2 as long as desired.
4. Return $S_{\text{best}}$ and $C(S_{\text{best}})$.
```

A typical run of this implementation of extremal optimization for the graph-bipartitioning problem on an $n = 500$ random graph is shown in Figure 3a.

The most apparent distinction between extremal optimization and other methods is the need to define local cost contributions λ_i for each variable, instead of merely a global cost. Extremal optimization's capability appears to derive from its ability to access this local information directly. Extremal optimization's ranking of fitnesses required for step 2b superficially appears like the ranking of possible moves in some versions of simulated annealing and other heuristics.[1] There, moves are evaluated by their *anticipated outcome*, while extremal optimization's fitnesses reflect the *current* configuration S without biasing the out-

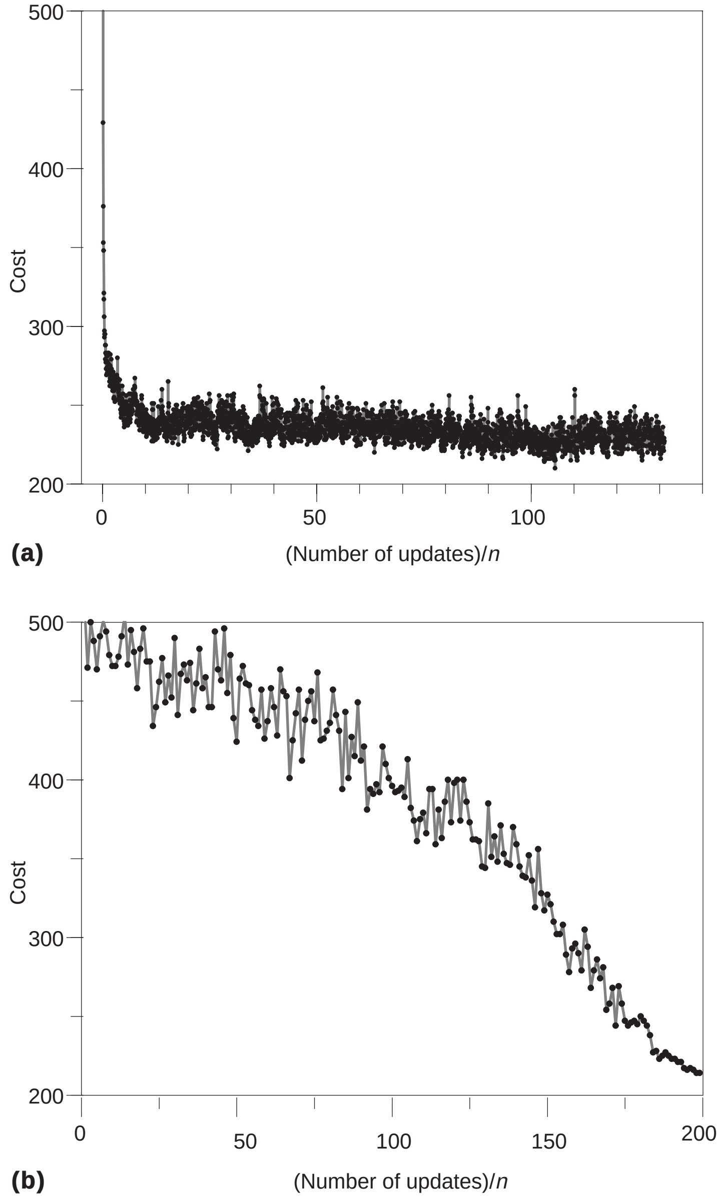


Figure 3. Evolution of the cost function $C(S)$ during a typical run of (a) extremal optimization and (b) simulated annealing, for the n = 500 vertices random graph G_{500}.[4] The lowest cost ever found for G_{500} is 206 (see Figure 4). In contrast to simulated annealing, which has large fluctuations 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 minima.

come. As a comparison of Figures 3a and 3b demonstrates, a sequence of these moves allows for much larger than equilibrium fluctuations in $C(S)$.

Although similarly motivated, genetic algorithms[6,7] and extremal-optimization algorithms have hardly anything in common. Genetic algorithms mimic evolution on the level of genes, and keep track of entire gene pools of configurations from which to select and breed an improved generation of solutions. In comparison, extremal optimization, based on evolutionary competition at the phenomenological level of species, operates only on a single configuration, with improvements achieved merely by the elimination of bad fitness values. Extremal optimization and simulated annealing perform a local search, but in genetic algorithms crossover operators perform global exchanges.

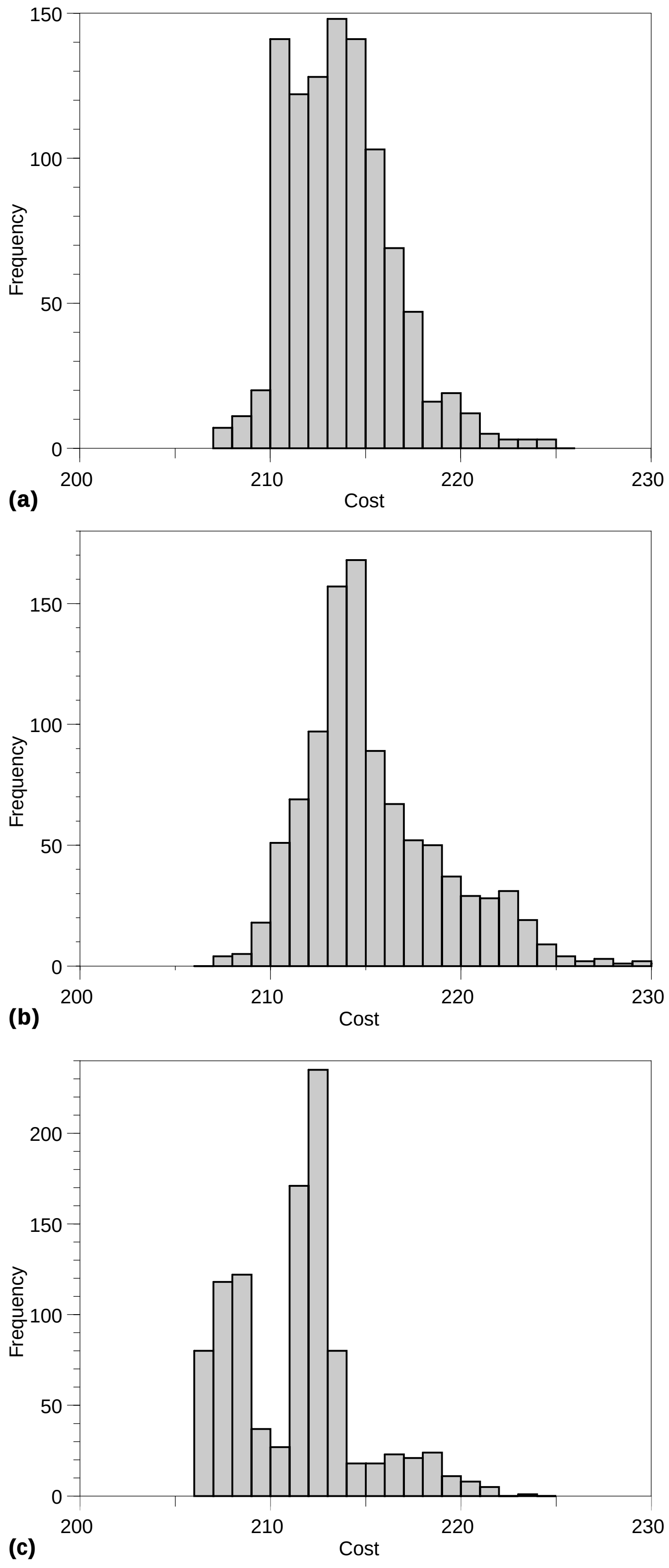
Simple extremal-optimization application to graph partitioning

Following the example of David S. Johnson and his colleagues[4] (see Figure 9 in the cited text), Allon Percus and I tested early implementations of extremal optimization[5] in a 1,000-run sample on their n = 500 random graph G_{500}. This version is based on a 1-exchange between the worst vertex (step 2b) and one randomly chosen vertex of the opposite state (step 2c). We first implemented Johnson's simulated-annealing algorithm[4] on the same data structure used by our extremal-optimization program (see Figure 4a). Then, we determined the frequency of solutions found by extremal optimization (see Figure 4b). We have used runtimes for extremal optimization that are about three times longer than the time it took for simulated annealing to freeze, because extremal optimization still yielded significant gains. We checked that neither the best of three runs of simulated annealing nor a three-times-longer temperature schedule improved the simulated-annealing results significantly. Although the basic, parameter-free version of extremal optimization considered so far is already competitive, the best results are obtained by τ-extremal optimization (see Figure 4c), which is discussed below.

The τ-extremal-optimization implementation

The τ-extremal-optimization implementation is a modification of extremal optimization that improves results and avoids dead ends that occur in some implementations at the expense of in-

Figure 4. Trials of 1,000 runs on G_{500} (see Figure 3) using (a) simulated annealing, (b) extremal optimization, and (c) extremal optimization with $\tau = 1.5$. The histograms give the frequency with which a particular cost has been obtained during the trial runs. The best cost ever found for this graph is 206.[4] This result appeared only once over the 1,000 simulated-annealing runs in Figure 4a. While the best result for extremal optimization in Figure 4b was 207, τ-extremal optimization in Figure 4c obtained 80 times a result of 206.



troducing a single parameter.[5] In general, the implementation of τ-extremal optimization proceeds as follows. Rank all the variables x_i according to their fitness λ_i; that is, find a permutation Π of the labels i such that

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

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,

$$P_k \propto k^{-\tau}, 1 \leq k \leq n \qquad (3)$$

for a fixed value of the parameter τ. At 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 $x_{i1}, x_{i2}, \ldots$ with $i_1 = \Pi(k_1)$, $i_2 = \Pi(k_2)$, … change. For example, in the bipartitioning 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 $k = 1$ will be chosen most often, sometimes (much) higher ranks will be updated instead. In fact, the choice of a power-law distribution for P_k (instead of, say, an exponential distribution with a cutoff scale excluding high ranks) ensures that no rank gets excluded from further evolution while maintaining a bias against variables with bad fitness.

Clearly, for $\tau = 0$, τ-extremal optimization is exactly a random walk through Ω. Conversely, for $\tau \to \infty$, the

Table 1. Best costs (and allowed runtime in seconds) for a testbed of large graphs. The value of *n* is given for each graph. Genetic algorithms results are the best reported[14] (using a 300-MHz CPU). The τ-extremal-optimization results are from our runs (200 MHz). Comparison data for three of the large graphs are due to spectral heuristics by Bruce Hendrickson and Robert Leland (50 MHz).[15] METIS is a partitioning program based on hierarchical reduction instead of local search,[16] and yields extremely fast, deterministic results (200 MHz).

Graph		Genetic algorithm		τ-extremal optimization		Spectral heuristics[15]		METIS	
Name	*n*	Cost	Time (sec.)	Cost	Time (sec.)	Cost	Time (sec.)	Cost	Time (sec.)
Hammond	4,720	90	1	90	42	97	8	92	0
Barth	15,606	139	44	139	64	146	28	151	0.5
Brack2	62,632	731	255	731	12	—	—	758	4
Ocean	143,437	464	1,200	464	200	499	38	478	6

Suggestions for further study

1. A simple model of a glass consists of a *d*-dimensional hypercubic lattice with a spin variable $\sigma_i \in \{-1, 1\}$ placed on each site *i*, $1 \le i \le n = L^d$.[1] Every spin *i* is connected to each of its nearest neighbors *j* via a fixed-bond variable $J_{i,j}$ drawn randomly from {–1, 1}. Spins may be coupled to an arbitrary external field h_i. The cost function to be minimized is the Hamiltonian

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

Arranging the spins into an optimal, lowest-energy configuration is hard because of frustration.[1] In fact, for $d > 2$ the problem is NP-hard.[2]

A. Find a definition of λ_i for each spin variable such that Equation 1 in the main article is satisfied.

B. Define a simple neighborhood *N* for this problem. Would your *N* lead to a stochastic or a deterministic sequence of update? Do you expect that your basic extremal-optimization implementation would be very successful?

C. Implement τ-extremal optimization for this spin glass in $d = 3$ for $h_i = 0$. Use single spin flaps as your neighborhood *N*, and sort your fitnesses in the hash table from the next exercise. A single run should have at least $t > n$ updates (why?); but $t \propto n^4$ appears to be necessary,[3] so keep *n* small (≤1,000). Try to find a good value for τ. Does it depend on *n* or *t*? For comparison, the cost function has been found with genetic algorithms to scale like $C(S_{\text{best}}) \sim -1.786n$ for $n = 3^3 \ldots 14^3$.[3]

2. The τ-extremal-optimization algorithm described in the main article requires a perfect ordering (see Equation 2 in the main article) of the λ_i ($1 \le i \le n$), which would produce a factor of $n \ln n$ for the computational time. In practice, it is sufficient to order the λ_i somewhat. (Most likely many λ_i will be degenerate.)

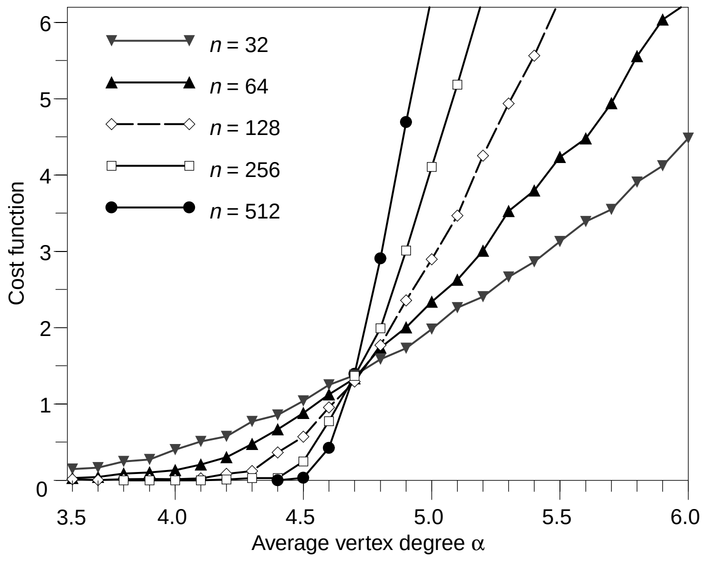
A. Devise a τ-extremal-optimization implementation in which the λ_i are sorted on a binary tree only (time factor ln *n*), where the λ_i are picked such that Equation 3 in the main article is roughly approximated. (For Allon Percus's and my humble attempt, see "Nature's Way of Optimizing."[4])

B. For some problems even the use of a hash table with a constant time factor may be useful. (Hash tables are explained in *The Practice of Programming*.[5]) For the spin glass problem in Problem 1, exploit the degeneracies between individual fitnesses to give a sorting algorithm using a hash table that leaves the λ_i perfectly ordered.

References

1. M. Mezard, G. Parisi, and M.A. Virasoro, *Spin Glass Theory and Beyond*, World Scientific, Singapore, 1987.
2. F. Barahona, "On the Computational Complexity of Ising Spin Glass Models," *J. Physics A: Mathematical and General*, Vol. 15, No. 10, Oct. 1982, pp. 3241–3253.
3. K.F. Pal, "The Ground State Energy of the Edwards-Anderson Ising Spin Glass with a Hybrid Genetic Algorithm," *Physica A*, Vol. 223, 1996, pp. 283–292.
4. S. Boettcher and A.G. Percus, "Nature's Way of Optimizing," *Artificial Intelligence*, Vol. 119, Nos. 1–2, May 2000, pp. 275–286.
5. B.W. Kernighan and R. Pike, *The Practice of Programming*, Addison-Wesley, Reading, Mass., 1999.

Figure 5. Plot of the average optimal cost as a function of the average vertex degree α for 3-COL of random graphs. Results were found by τ-extremal optimization for 2,300, 650, 330, 150, and 100 instances for graph sizes n = 32, 64, 128, 256, and 512 vertices, respectively, at each value of α. The critical point (where the costs intersect) is at $\alpha_{\text{crit}}(3) \approx 4.72$.



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 graph-bipartitioning problem, τ-extremal optimization obtained its best solutions for τ in the range of 1.4 to 1.6. Clearly, we have "tuned" away from the philosophy of the Bak-Sneppen model by inserting a single parameter for the sake of better results. To be successful, *every* heuristic has to allow for some adjustments to a particular problem.

Table 1 summarizes our τ-extremal-optimization results for some well-studied instances of graphs with large n, using $\tau = 1.4$ and the best of 10 runs. We obtained initial configurations from a simple clustering algorithm.[5] Then, we chose the number of updates t such that the results did not change much beyond t. The choice of t varied with the particularities of each graph, from $t = 2n$ to $t = 200n$, and the reported runtimes are of course influenced by the value of t. It is worth noting that the performance of extremal optimization varies. For instance, half of the runs on the graph known as Brack2 returned costs near 731, but the other half returned costs above 2,000. This variation may be a product of an unusual structure in this particular graph. In a systematic study of extremal optimization in comparison with simulated annealing by averaging over many graphs of increasing size, I found that extremal optimization finds near-optimal results for graphs of low connectivity and that the results of simulated annealing become worse with increasing graph size.[13]

Other extremal-optimization implementations

To demonstrate the generality of extremal optimization, Allon Percus and I are currently experimenting with its implementation for other NP-hard optimization problems such as graph coloring (K-COL), satisfiability (K-SAT), and spin glass Hamiltonians. In 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. A definition of fitness is as obvious as it was for the graph-bipartitioning problem: for each vertex x_i simply count the number b_i of equally colored vertices connected to it; setting $\lambda_i = b_i/2$ again satisfies Equation 1. The lack of a global constraint as for the graph-bipartitioning problem allows us to define a neighborhood by changing the state of only one, the worst, variable. However, this definition results in a deterministic search that quickly reaches a dead end. A τ-extremal-optimization implementation picking a single variable with $\tau \approx 2.7$ seems to work best for the graphs we have studied.

With this algorithm we have studied the "phase transition" of K-COL, "where the really hard instances are."[17] This transition arises as a function of the average vertex degree α for certain types of graphs. (The degree of a vertex is the number of edges emanating from it and may vary between vertices of a graph.) If α is small, almost all vertices have fewer than K neighbors, coloring becomes trivial, and the optimal solution has zero cost for almost all graph instances. But around a critical value $\alpha_{\text{crit}}(K)$, the cost becomes positive, with an ever sharper transition for $n \to \infty$. If we average the best solutions that extremal optimization finds over many instances of random graphs, we can show that $\alpha_{\text{crit}}(3) \approx 4.72$ (see Figure 5). The relationship between phase transitions, which occur in many NP-hard problems, and computational complexity is evolving into one of the hot topics in computer science.[17–20] In this interesting regime extremal optimization's large fluctuations appear to have the edge over simulated annealing's equilibrium requirements.[13]

Because extremal optimization is fairly new, there are many unanswered questions. It will not be difficult for the interested reader to think of research projects that, for example, compare extremal optimization to other methods. Clearly, like any other optimization method, extremal optimization will not be competitive for some problems (unfortunately, the traveling salesman problem seems to be one example[5]). But it can't hurt to have more alternative methods to choose from for tackling hard optimization problems.

Acknowledgments

I thank Allon Percus, with whom I developed extremal optimization, and Emory University's Research Committee for their support.

References

1. C.R. Reeves, ed., *Modern Heuristic Techniques for Combinatorial Problems*, John Wiley & Sons, New York, 1993.
2. S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. 220, No. 4,598, May 1983, pp. 671–680.
3. S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 6, No. 6, Nov. 1984, pp. 721–741.
4. D.S. Johnson et al., "Optimization by Simulated Annealing: An Experimental Evaluation; Part 1, Graph Partitioning," *Operations Research*, Vol. 37, No. 6, Nov. 1989, pp. 865–892.
5. S. Boettcher and A.G. Percus, "Nature's Way of Optimizing," *Artificial Intelligence*, Vol. 119, Nos. 1–2, May 2000, pp. 275–286.
6. J. Holland, *Adaptation in Natural and Artificial Systems*, Univ. of Michigan Press, Ann Arbor, Mich., 1975.
7. D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Mass., 1989.
8. P. Bak, *How Nature Works*, Springer-Verlag, New York, 1996.
9. P. Bak and K. Sneppen, "Punctuated Equilibrium and Criticality in a Simple Model of Evolution," *Physical Rev. Letters*, Vol. 71, No. 24, 13 Dec. 1993, pp. 4083–4086.
10. M. Newman, "Simple Models of Evolution and Extinction," *Computing in Science & Eng.*, Vol. 2, No. 1, Jan./Feb. 2000, pp. 80–86.
11. P. Bak, C. Tang, and K. Wiesenfeld, "Self-Organized Criticality: An Explanation of the 1/*f* Noise," *Physical Rev. Letters*, Vol. 59, No. 4, 27 July 1987, pp. 381–384.
12. S. Boettcher and M. Paczuski, "Ultrametricity and Memory in a Solvable Model of Self-Organized Criticality," *Physical Rev. E*, Vol. 54, No. 2, Aug. 1996, pp. 1082–1095.
13. S. Boettcher, "Extremal Optimization and Graph Partitioning at the Percolation Threshold," *J. Physics A: Mathematical and General*, Vol. 32, No. 28, 16 July 1999, pp. 5201–5211.
14. P. Merz and B. Freisleben, *Memetic Algorithms and The Fitness Landscape of the Graph Bi-Partitioning Problem*, *Lecture Notes in Computer Science*, No. 1,498, Springer-Verlag, Heidelberg, Germany, 1998, pp. 765–774.
15. B.A. Hendrickson and R. Leland, "A Multilevel Algorithm for Partitioning Graphs," *Proc. Supercomputing '95* (CD-ROM), IEEE Computer Soc. Press, Los Alamitos, Calif., 1995.
16. G. Karypis and V. Kumar, "METIS: Family of Multilevel Partitioning Algorithms," www-users.cs.umn.edu/~karypis/metis/main.shtml.
17. P. Cheeseman, B. Kanefsky, and W.M. Taylor, "Where the Really Hard Problems Are," *Proc. 1991 Int'l Joint Conf. Artificial Intelligence* (IJCAI '91), Morgan Kaufmann, San Francisco, 1991, pp. 331–337.
18. Special issue on Frontiers in Problem Solving: Phase Transitions and Complexity, *Artificial Intelligence*, Vol. 81, Nos. 1–2, Mar. 1996.
19. R. Monasson et al., "Determining Computational Complexity from Characteristic 'Phase Transitions,'" *Nature*, Vol. 400, No. 6,740, 8 July, 1999, pp. 133–137.
20. J. Machta and R. Greenlaw, "The Computational Complexity of Generating Random Fractals," *J. Statistical Physics*, Vol. 82, Nos. 5–6, Mar. 1996, pp. 1299–1326.

Stefan Boettcher is a member of the Department of Physics at Emory University. His research interests include self-organized criticality, optimization problems, and quantum field theory. He received his PhD from Washington University in St. Louis for research on perturbative methods in quantum field theory. Contact him at the Dept. of Physics, Emory Univ., Atlanta, GA 30322; stb@physics.emory.edu; www.physics.emory.edu/faculty/boettcher.