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 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 single sub-optimal solution with new, random ones. Large fluctuations, or “avalanches”, ensue that efficiently explore many local optima. Drawing upon models used to simulate far-from-equilibrium dynamics, extremal optimization complements heuristics 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. Phase transitions are found in many combinatorial optimization problems, and have been conjectured to occur in the region of parameter space containing the hardest instances. We demonstrate how extremal optimization can be implemented for a variety of hard optimization problems. We believe that this will be a useful tool in the investigation of phase transitions in combinatorial optimization, thereby helping to elucidate the origin of computational complexity.

1 Emergence of Optimized Configurations

Natural systems are complex structures, and often optimize efficiency in surprisingly sophisticated ways. Biological evolution has resulted in a plethora of interdependent species, collectively competing for resources in such a way that these resources rarely go to waste [Bak96]. Geographic landscapes, in their minute intricacy, can serve a valuable purpose such as efficient drainage of water [Ro97]. Yet in spite of the complexity of the outcome, nature’s mechanisms are in general exceedingly simple. Evolution is driven merely by sunlight. Amazing as it may seem from an engineering standpoint, nature employs no form of central organization or intelligent design.

In an effort to understand these self-organizing qualities it is helpful to note, by way of abstraction from the concept of species, a feature common to many natural systems: the presence of a large number of strongly coupled entities with similar properties. At a coarse level, these may be modeled using a statistical description. One might consider evolution, or the development of landscapes, to be a stochastic process driven by an external force (the sun, flowing water, etc.). The precise elements that appear in nature are subject to the whims of chance, but the coupling of the elements insures that only the well-adapted ones can survive. The phenomenon of flexible adaptation thus emerges naturally, through the dynamics of adverse selection. “Good” species are not engineered. “Bad” ones are simply extinguished.

In recent years, statistical physicists have proposed models to describe this phenomenon [PacMB96]. One of the most successful has been the Bak-Sneppen model of biological evolution, based on precisely the extremal principle of eliminating the worst adapted elements in an ecosystem [BakS93, BoePa96]. Like nature’s own generic mechanism, it is devoid of any specificity as to the precise interactions between species. Nevertheless, it produces salient nontrivial features of paleontological data such as broadly distributed lifetimes of species, large extinction events, and punctuated equilibrium [GouE77].

In the Bak-Sneppen model, species are located on the sites of a lattice, and have an associated “fitness” value between 0 and 1. At each time step, the one species with the smallest value (poorest degree of adaptation) is selected for a random update, having its fitness replaced by a new value drawn randomly from a flat distribution on [0, 1]. But the change in fitness of one species impacts the fitness of interrelated species. Therefore, not only is the least-fit species updated, but all of the species at its neighboring lattice sites have their fitness replaced with new random numbers as well. No explicit definition is ever given of the mechanism by which these neighboring species are related. Yet after a sufficient number of steps, the system reaches a highly correlated state known as self-organized criticality (SOC) [BakTW87]. 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. This co-evolutionary activity gives rise to chain reactions called “avalanches”, large fluctuations that rearrange major parts of the system, potentially making any configuration accessible.

Although co-evolution does not have optimization as its exclusive goal, it serves as a powerful paradigm. We have used SOC as motivation for a new approach to optimization [BoePe00a]. The heuristic that we have introduced, called Extremal Optimization (EO), follows the spirit of the
Bak-Sneppen model in that it merely replaces the variables with the “worst” values in a solution by random ones, eliminating bad variables without ever explicitly creating good ones. In what follows, we show how this method may be implemented on a number of classic combinatorial optimization problems, and discuss key aspects of its performance.

2 Extremal Optimization

Extremal Optimization (EO) is inspired by previous attempts at using physical intuition to optimize. In much the same manner that simulated annealing (SA) [KiGV83] applies equilibrium statistical mechanics to optimization problems, EO opens the door to the systematic application of non-equilibrium processes, such as SOC. The result is a general method that appears to be a powerful addition to the canon of meta-heuristics [OK96]. In particular, the large fluctuations in EO provide significant hill-climbing ability, which enables it to perform well at the phase transitions “where the really hard problems are” [CKT91, AI96].

One popular hard optimization problem, to which we have applied EO successfully (see below and [BoePe00a, Boe99]), is the graph bi-partitioning problem (GBP) [Gaj79, KiGV83, JAMS89, MezP87]. 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 \( \Omega \) grows exponentially with \( n \), \( |\Omega| = \binom{n}{n/2} \), since all unordered divisions of the \( n \) vertices into two 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 [AaL97, V98], mapping \( S \rightarrow S' \in N(S) \subset \Omega \), is a “1-exchange” of one randomly chosen vertex from each subset.

EO performs a search starting from 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 \( \lambda_i \) for each variable \( x_i \), which correspond to (the inverse of) the “fitness” values in the Bak-Sneppen model above. Typically, \( \lambda_i \) depends on variable \( x_i \)’s state in relation to the other variables that it is connected to. Ideally,

\[
C(S) = \sum_{i=1}^{n} \lambda_i \tag{1}
\]

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 \( S_{\text{best}} = S \).
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 \( i \), 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 \( C(S') < C(S_{\text{best}}) \) then set \( S_{\text{best}} = S' \),
   (e) accept \( S \rightarrow S' \) always, regardless of whether \( C(S') < C(S) \).
3. Repeat at step (2) as long as desired.
4. Return \( S_{\text{best}} \) and \( C(S_{\text{best}}) \).

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 for \( S \). It implies that all other variables are “better” in the current \( S \). Those “better” variables only possess punctuated equilibrium: their memory gets erased when they happen to be connected to one of the variables forced to change. There is no parameter to be adjusted. The memory encapsulated in the ranking alone directs EO into the neighborhood of increasingly better solutions. Furthermore, in the choice of move to \( S' \), EO gives no consideration to the move’s outcome. Large fluctuations in the cost accumulate over many updates [BakS93], with merely the bias against “bad” fitnesses guiding EO back towards improved solutions. A typical “run” of this implementation of EO for the GBP on an \( n = 500 \) random graph is shown in Fig. 1.

A disadvantage of EO is that a reasonable definition of fitness for individual variables may be ambiguous or even impossible. Also, variables may be sufficiently connected that each update destroys more well-adapted variables than it could ever hope to improve [BoePe00a]. In highly connected systems, EO is slowed down considerably by reevaluating fitnesses [step (2a)]. For many problems, however, these disadvantages do not apply or are surmountable. In particular, problems in the important optimization class MAX-SNP [PapY91] 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 problems for the class have bounded connectivity in the worst case. MAX-SNP complete problems include MAX-2-SAT, MAX-3-SAT, \( K \)-COL, and MAXCUT (similar to the 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. EO’s capability appears to derive from its direct access to this local information. Superficially, EO’s ranking of fitnesses appears to resemble the rankings of candidate moves in some versions of SA [GreeS86, Re93] and in Tabu search [Gi86, Re93, AaL97]. But these moves are evaluated by their anticipated outcome, while EO’s fitnesses reflect the current configuration $S$ without biasing the outcome.

3.1 Simulated Annealing (SA)

SA [KiGV83] 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 with 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.

3.2 Genetic Algorithms (GA)

While similarly motivated, GA [Ho75, Goldb89] and EO algorithms have little in common. GAs, mimicking evolution on the genotype 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, whereas in GA cross-over operators perform global exchanges.

4 Applications of Extremal Optimization

4.1 Ground States of Spin Glasses

A simple version of a spin glass [MezPV87] 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 $J_{ij}$ drawn from some distribution $P(J)$ with 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 $S^\text{min}$ of

$$C(S) = -\frac{1}{2} \sum_i \sum_j J_{ij} \sigma_i \sigma_j - \sum_i \sigma_i h_i.$$  

(2)

Arranging the spins into an optimal configuration is hard due to “frustration” [T77]. To implement EO, we define fitness in terms of the local energy for each spin

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

(3)

and Eq. (2) turns into Eq. (1). Our implementation suggests that EO may be well suited for problems representable by a
4.2 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 find a configuration of the variables that maximizes the number of true clauses.

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 $K$ other variables. The phase transition in 2-SAT and 3-SAT has been investigated in [MoZKST99, AI96] on small instances using exact methods. We expect that EO would perform very well on those instances.

4.3 Graph Coloring ($K$-COL)

Given $K$ different colors with which we can label the vertices of a graph, we need to find a coloring that minimizes the number of edges connecting vertices of identical color [JAMS91]. We implement EO for $K$-COL by defining $\lambda_i$ for each vertex as the number of equally colored vertices connected to it. Similar to a spin glass, this problem is hard due to local frustration [T77], rather than to the global constraints found in the GBP. We will use EO to revisit the phase transition in 3- and 4-COL, which has been investigated in [CKT91].

5 Experimental Results

5.1 Simple EO Application on Graph Partitioning

Following the example of [JAMS89], we tested early implementations of EO [BoePe00a] 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 [JAMS89] 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 an enhancement that we call $\tau$-EO.

5.2 $\tau$-EO Implementation

$\tau$-EO is a general modification of EO which improves results and avoids “dead ends” occurring in some implementations. This comes, however, at the expense of a parameter. The method is as follows: Rank all variables according to fitness, the “worst” variable having rank 1 and the “best” variable having rank $n$. Consider a probability distribution over the ranks $i$,

$$P_i \propto i^{-\tau}, \quad 1 \leq i \leq n,$$

for a fixed value of the parameter $\tau$. On each update, for each independent variable to be moved, select distinct ranks $i_1, i_2, \ldots$ according to $P_i$. Then, move variables $x_{j_1}, x_{j_2}, \ldots$ where $i_1 = \text{rank}(\lambda_{j_1}), \ i_2 = \text{rank}(\lambda_{j_2}), \ldots$ While the “worst” variable (rank $i = 1$) will be chosen most often, higher ranks will sometimes be updated instead. The choice of a power-law distribution for $P_i$ maintains a bias against variables with “bad” fitness, while insuring that no rank gets completely excluded from further evolution. Note that the same would not be true if, say, an exponential distribution were used: higher ranks would then, in effect, never be selected.

Clearly, for $\tau \equiv 0$, $\tau$-EO is exactly a random walk through $\Omega$. Conversely, for $\tau \to \infty$, the process approaches a deterministic local search, only swapping the lowest-ranked vari-

![Figure 2: Comparison of 1000-run trials using various optimization methods, for the random graph $G_{500}$ (size $n = 500$, connectivity $c \approx 5$) from [JAMS89]. 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 $\tau$-EO with $\tau = 1.5$. The best cutsize ever found is 206. This result appeared only once over the 1000 SA runs, but occurred 80 times for $\tau$-EO.](image-url)
able, and is bound to reach a “dead end.” Not surprisingly, 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 \( \tau \) and a transition to “non-ergodic” behavior, in the sense that for larger values of \( \tau \) certain configurations in \( \Omega \) may be completely inaccessible over the course of an EO run. On the basis of that observation, in fact, we have developed a qualitative argument giving as the optimal choice \( \tau \approx 1 + \log(n) / \log(n) \) [BoePe00b], where \( t = An (1 \ll A \ll n) \) is the runtime. Typically, we use \( A \approx 10^2 \) for graphs of size \( n \approx 10^4 \), consistent with \( \tau \approx 1.5 \). Tests with longer runtimes indeed favor larger \( \tau \) values, while larger graphs require smaller \( \tau \) values.

### 5.3 Results on Large Graphs

Table 1 summarizes \( \tau \)-EO’s results on large-\( n \) mesh-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. EO’s average performance has, on the other hand, been inconsistent. For instance, half of the \( \text{Brack2} \) 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.

### 5.4 Phase Transitions in Graph Partitioning

In an extensive numerical study, we have shown [Boe99] that \( \tau \)-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 [GresSL86], roughly as \( C(S_{\text{best}}) \sim C(S_{\text{min}}) + A t^{-1/2} \) [BoePe00b]. Of course, it is not easy to verify for graphs of large \( n \) that those runs in fact converge closely to the optimum \( C(S_{\text{min}}) \), but finite-size scaling analysis seems to justify that expectation [BoePe00b].

In an even more impressive performance demonstration, we used EO to completely enumerate all optimal solutions \( S_{\text{min}} \) near the critical point for random graphs. Instances of random graphs typically have a high ground-state degeneracy, i.e., possess a large number of equally optimal solutions \( S_{\text{min}} \).

In [MoZKST99] it was shown that at the phase transition of 3-SAT the fraction of constrained variables, i.e., those that are found in an identical state in all \( S_{\text{min}} \), discontinuously jumps to a non-zero value. It was conjectured that the first-order phase transition [Gold92] in this “backbone” would exist for any NP-hard problem. To test those claims for the GBP, we generated a large number of random graphs and explored \( \Omega \)
The giant component. For connectivity Fig. 4. The backbone fraction rises monotonically with the distribution over all pairs of ground states, whose second moment is our backbone fraction. Averaged results are given in Fig. 4. The backbone fraction rises monotonically with the connectivity $c$, becoming non-zero already for $c > 1$, the percolation point [ER60]. In fact, for $1 \leq c \leq c_{\text{crit}} = 2 \ln 2$, the backbone fraction merely traces (the square of) the fraction of vertices belonging to the giant component, whose size is exactly known [Bol85]. At $c_{\text{crit}}$, the largest cluster becomes $> n/2$ and the backbone fraction begins to grow slower than the giant component. For $c > c_{\text{crit}}$, the cutsize becomes non-zero (see Fig. 4b) due to cuts forced within the giant component itself.

Our results indicate that critical phenomena in NP-hard problems have a more diverse phenomenology than expected in [MoZKST99]. Clearly, unlike the variables in $K$-SAT, vertices in the GBP do not experience frustration [T77]. Instead, almost all vertices happily attain a state identical to their neighbors, leaving merely a few along an interface to make tough choices. In addition, ground states in the GBP possess a global symmetry not found in SAT, typically an indication of differing critical behavior (or “universality class”) [Golde92].

Bibliography


