A Model for Analyzing Black-Box Optimization (Extended Abstract)

Vinhthuy Phan Steven Skiena Pavel Sumazin
State University of New York at Stony Brook
Stony Brook, NY 11794-4400 USA
{phan,skiena,psumazin}@cs.sunysb.edu

1 Introduction

The design of heuristics for NP-hard problems is perhaps the most active area of research in the theory of combinatorial algorithms. However, practitioners more often resort to local improvement heuristics such as gradient descent search, simulated annealing, tabu search, or genetic algorithms. Properly implemented, these techniques can lead to short, efficient programs that yield reasonable solutions.

In this paper, we introduce a model for the landscape of combinatorial optimization problems, and analyze the behavior and expected performance of algorithms for black-box heuristic search on this model. Our model is simple enough to reason formally about, yet it is both intuitive and supported by experimental evidence. *Black-box* heuristics are problem-independent algorithms which assume only (1) the existence of local neighborhood operators which take an arbitrary element s of the solution space S and generates nearby elements of S, and (2) a cost function which evaluates the quality of a given element s.

We model the solution space S associated with a particular minimization problem instance as a random directed graph $G = (V, E)$. We rank the solutions from 0 to $N-1$ in order of increasing cost, breaking ties arbitrarily, so that $\text{rank}(i) < \text{rank}(j)$ implies $\text{cost}(i) \leq \text{cost}(j)$. Each of the N vertices in V represents a solution $s \in S$, and we say that vertex v_i corresponds to the solution of rank i. Each vertex v has an out-degree of k; with edges drawn from v to every vertex representing a neighbor of s. In this paper, we analyze the case where neighbors are chosen uniformly at random from a window of up to $2c$ vertices, centered at v and consisting of the vertices ranked immediately higher and lower than v, if they exist. The goal of our search, starting from vertex v_{N-1}, is to reach the lowest ranked vertex possible.

In summary, problem instances are parameterized by their solution space size N, local neighborhood size k, and rank edge span c. Although this model is simple, it is sufficiently rich to study such questions as the impact of neighborhood size and structure on search performance. Indeed, we [15] are building a general combinatorial optimization engine based on local search, and developed the theory in this paper to guide our implementation.

This model has proven very helpful in enabling us to reason about the relevant search algorithms in a rigorous way. For example, analysis of the model suggests that the advantage of gradient descent heuristics lies in their ability to isolate a subset of high-scoring solutions and to search within that subset. There is a cost for finding this subset, and repeated application of hill-climbing from different initial solutions multiplies this overhead. Thus certain forms of backtracking which avoid this penalty are preferred under our theory.

In this paper, we analyze our model to establish the following results:

- *There is a "Free Lunch"* – We show that local search cannot perform better than random sampling for unrestricted random local neighborhood operators (i.e. $c = N$), as suggested by Wolpert and Macready's [18] *no free lunch theorem*, which states that any heuristic that (1) does not extract information about the cost function, and (2) evaluates new solutions at the same rate is expected to perform no better than random sampling. Under our model, however, local search outperforms random sampling for restricted neighborhood operators (i.e. $c < N$), thus implying the choice of heuristic is indeed significant.

- *Comparison of Local Search Heuristics* – We analyze four different hill-climbing local search heuristics including greedy hill climbing (GHC), random hill climbing (RHC), reverse greedy hill climbing (RGHC) and combined hill climbing (CHC). See Section 4 for definitions. Our results, summarized in the table below, give the expected number of search operations and solution ranks for both small neighborhoods ($k < \lg(N/c)$) and large neighborhoods ($k > \lg(kN/c)$). Note that ϵ_1, ϵ_2, and ϵ_3 are all small constants.

	$k < \lg N/c$		$k \geq \lg kN/c$	
Algorithm	Operations	Expected Rank	Operations	Expected Rank
GHC	$k2^k$	$N - c2^k(k+2)/k - c/k$	$k(N-c)/c + ek$	$c/(ek + \epsilon_1)$
RHC	2^{k+1}	$N - c2^{k-1}$	$2(N-c)/c + ek$	$c/(ek + \epsilon_2)$
RGHC	$k2^k$	$N - c(2^{k+1}-1)/k$	$k^2(N+c)/2c + ek^2/2$	$2c/(k + \epsilon_3)$
CHC	2^{k+1}	$N - c2^{k-1}$	$2(N-c)/c + ek$	$c/(ek + \epsilon_1)$

- *Comparison of Backtracking Procedures* – We analyze five different backtracking heuristics including greedy backtracking (GBT) and random backtracking (RBT). These heuristics backtrack to random higher ranked neighbors when they reach a sink. We compare the performance of these backtracking heuristics when allowed a total of x backward moves against repeated runs of the random hill climbing heuristic (RHC) (allowed x' repetitions).

	$k < \lg(N/c)/x$		$k \geq \lg N/c$, $m = 2(N-c)/c$	
Algorithm	Operations	Expected Rank	Operations	Expected Rank
GBT	$2^{k+1} + kx2^k$	$N - c2^{k-1} - xc(2^k(1-2/k) - 1/2)$	$m + ek(x+1) + kx$	$c/(ek(x+1))$
RGBT	$2^{k+1} + kx2^k$	$N - c2^{k-1} - (xc/2)(2^{k+1}-1)$	$m + x(ek^2+1)$	$2c/(k(x+1))$
RBT	$2^{k+1}(x+1)$	$N - c(2^{k-1}(x+1) - x/2)$	$m + ek(x+1) + 2x$	$c/(ek(x+1))$
CBT	$2^{k+1}(x+1)$	$N - c(2^{k-1}(x+1) - x/2)$	$m + ek(x+1) + kx$	$c/(ek(x+1))$
RHC	$x'2^{k+1}$	$N - 2^k(c+1)$	$x'(m+ek)$	$c/(ekx')$

- *Making Best Use of Time* – A fundamental problem facing any general optimization strategy is making effective use of the full time allotted to completion. Hill climbing heuristics make rapid initial progress toward improved solutions, but terminate quickly and thus do not use large amounts of search time effectively.

  We use our model to investigate efficient time allocation policies. When the allotted search time is sufficiently short, random sampling is the most effective heuristic. As search times and local neighborhood sizes increase, hill climbing heuristics with increasing amount of backtracking become more effective. We identify the best heuristic as a function of allotted search time and local neighborhood size, with our results summarized below:

Allotted Time	$k < \lg(N/c)/x$	$\lg(N/c)/x \leq k < \lg N/c$	$\lg N/c \leq k$
$t < 4(N-c)/c$	random sampling	random sampling	random sampling
$4(N-c)/c \leq t < 4(N-c)/c + x$	random sampling	random sampling	backtracking
$4(N-c)/c + x \leq t$	random sampling	backtracking	backtracking

Assuming that t total number of operations and x number of backtracking moves are allowed, this table presents the best heuristic choice for different ranges of k.

- *Experimental Results* – We evaluate the basic validity of our abstract model through experimentation. Using the black-box combinatorial-optimization system described in [15], we evaluate the analyzed heuristics on instances of five classical combinatorial problems (TSP, shortest common superstring, maxcut, maximum satisfiability, and vertex cover).

  We compare the observed times and solution qualities with predictions of our model. To a great extent, the gross predictions of our theory are confirmed. Where our theory breaks down (most clearly, on the performance of greedy backtracking), clear avenues for further research are suggested – specifically, the study of undirected neighborhoods and non-uniform edge probability distributions.

  Finally, to validate our analytical results we evaluate the performance of the proposed heuristics on random graphs generated according to the model.

Organization

We begin this paper with a brief survey of related work on heuristic search in Section 2. We then describe our random graph model for heuristic search in Section 3. The local search and backtracking procedures we will analyze are described in Section 4. We then analyze local search heuristics for arbitrarily wide search neighborhoods ($c = N$) and the more interesting narrow search neighborhoods. We conclude in Section 6 with our experimental results.

2 Previous Work

Local search heuristics such as simulated annealing, tabu search, and genetic algorithms have been extensively studied. We refer to the relevant chapters of [1] for recent surveys.

Much of the analytical work on these algorithms concerns the rate to which these algorithms, particularly simulated annealing, converge on the global optimum. See [10, 13] for representative results. Aldous and Vazirani [3] model the combinatorial search landscape as walks on a tree to analyze the *go with the winners* heuristic, which breaks the search procedure into stages where the best solutions in the previous stage are used as initial solutions for new parallel searches. There has also been interest in analyzing the performance of local search on specific classes of problems, particularly graph bisection in random graph models with planted solutions [6, 12]. Here the goal is to identify problems and input classes where local optimization heuristics ensure good solutions with high probability.

Our model is akin to measures of the *combinatorial dominance* of heuristics, where the dominance number of a heuristic solution s counts the number of solutions which are provably no better than s. Problem specific heuristics which achieve high dominance numbers for the traveling salesman problem have been extensively studied [8, 9]. Combinatorial dominance provides a quality

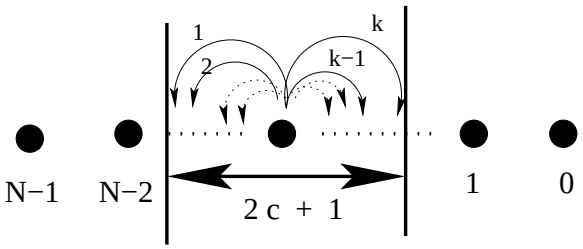

Figure 1: The landscape model, where each of N vertices has k randomly selected outgoing edges of span at most c. The goal of a search is to reach a vertex of lowest possible rank.

measure for heuristics which is sometimes orthogonal to approximation ratio. For example, a 2-factor approximation to metric TSP from the minimum spanning tree heuristic can in fact yield the worst of all $(n-1)!$ tours on certain instances [5].

Deineko and Woeginger [7] study the complexity of finding the solution which dominates certain well-defined neighborhoods for TSP. Finding a local optimum is difficult under certain problem formulations. The class *PLS* [11] consists of local search problems for which local optimality can be verified in polynomial time. Papadimitriou, Schafffer and Yannakakis [14] show that finding a local optimum for some problems in PLS is PSPACE-complete if we insist on a specific initial solution for the local search procedure. Other investigations into the relative difficulty of NP search problems include [2, 4].

3 A Model of Energy Landscapes

We model the energy landscape for a given problem as a random directed graph $G = (V, E)$, where $|V| = N$, $|E| = k \cdot N$, together with an objective function $f : V \longrightarrow \Re$ that imposes a total order among the vertices, so that $f(v_i) \leq f(v_j)$ for $1 \leq i < j \leq N$. We assume without loss of generality that $f(v_i) = i$. The edge set E is constructed by drawing k edges from $v_i \in V$ to randomly chosen vertices in range $[v_{i-c}, v_{i+c}]$; the target vertices are chosen with substitution, so that v_i may have more than one edge going to $v_j \in [v_{i-c}, v_{i+c}]$.

We seek to find a vertex of minimum rank while restricting our search algorithms to two operations, either (1) generating an element of V uniformly at random, or (2) traversing an edge (v_1, v_2) of G, where v_1 denotes a previously visited vertex of G. All searches begin at vertex v_{N-1}. We pay unit cost to evaluate the objective function on any vertex. We are interested in the expected performance of search heuristics on such graphs.

The performance of a heuristic depends on both the expected rank of the vertex that the heuristic reports (the solution quality) and the number of objective function evaluations it performs (the search cost). We say that one heuristic *dominates* another if it reports a vertex of the same or better expected rank as the other using fewer expected operations.

The graphs we study are parameterized by three constants: size N, out-degree k, and range c. Each graph has fixed out-degree k, with the vertices adjacent to v drawn uniformly at random from the set of vertices in a window of rank $\pm c$. For simplicity, we allow self-loops and multi-edges in G. See Figure 1 for a cartoon representation of our model.

We note that neighborhood operators such as *swap* and *2-opt* for permutation and subset problems have polynomial size-neighborhoods that are large enough to maintain connectivity. Furthermore, the length of the shortest path between any two permutation or subset solutions is in $O(\lg N)$, implying that $\frac{N}{c}$ is at most logarithmic in the size of the solution space.

To study the shape of our energy landscape, we partition G into two regions by rank: the *transient* region $(v_c, v_{N-1}]$, and the *goal* region $[v_0, v_c]$. The analysis of the behavior of a local-search heuristic depends on a proper understanding of its behavior in these regions.

- The range parameter c governs how fast we can hope to descend from the start vertex to the goal region. At least N/c local moves are necessary to enter the goal region.
- Within the transient region of the graph, the probability of a vertex being a sink is $1/2^k$, and hence depends completely on out-degree. Thus if k is sufficiently small relative to N and c, any gradient-descent search is likely to terminate in the transient region and not reach the goal region.
- The probability that a goal region vertex v is a sink depends on c, k, and $rank(v)$ this probability increases rapidly as we move down through the goal region. Thus it becomes progressively harder to make additional progress towards the optimum as we search through the goal region.
- Note that proximate vertices (v_i and v_{i+1}) represent similar scoring solutions. However, no short or even long path between them in the graph need exist for sufficiently small k.

4 Search Heuristics

Local search heuristics exploit the neighborhood structure for each solution. *Hill climbing* or *gradient descent* are the simplest local search heuristics. Starting from an initial solution, they proceed by visiting lower-scoring neighboring solutions of the current solution. These heuristics terminate when the vertex is a sink. In this paper, we consider four flavors of hill climbing:

- *Random hill climbing (RHC)* – Here, candidates are generated as random solutions from the set of neighboring solutions. We assume this is done without repeated selections, by traversing a random permutation of the neighborhood; the heuristic selects the first lower ranked neighbor discovered.
- *Greedy hill climbing (GHC)* – Here, the heuristic selects the lowest ranked solution from the entire set of neighboring solutions. Thus each step of GHC requires examining the entire set of k neighboring solutions.
- *Reverse greedy hill climbing (RGHC)* – Here, the heuristic selects the highest ranked solution from the set of all lower ranked neighbors. Thus it makes the slowest possible gradient descent through the search space. Again, every step of RGHC requires examining the entire set of k neighboring solutions.
- *Combined hill climbing (CHC)* – This hill climbing heuristic knows the value of c for the problem instance, which is sufficient insight to determine whether any solution s lies in the transient or goal regions of the solution space. CHC uses a combination of the random hill climbing and the greedy hill climbing heuristics: RHC-like traversal from an initial solution, and GHC-like traversal after entering the goal region.

We also consider the basic heuristic *random sampling*, which chooses solutions at random from S and visits as many solutions as possible within the allocated time.

Backtracking heuristics emerge from a given sink v_i by visiting some neighbor of inferior rank and continuing the search from there. We assume that backtrack heuristics remember the lowest ranking vertex encountered on this tour, so the additional search can only improve the quality of the resulting solution. We analyze four different backtracking heuristics based on variants of hill climbing; in all cases the search begins with a RHC search to a sink, followed by a backtracking move. All backtracking moves are made from a sink v to a randomly selected vertex in the neighborhood of v. Each are parameterized by the number of permissible backtrack moves x:

- *Greedy hill climbing with backtracking (GBT)* – Here, after the first backtracking move, downward moves are made to the lowest of the solution's k neighbors. Every step of GBT requires examining the entire set of k neighboring solutions.
- *Random hill climbing with backtracking (RBT)* – Here, all downward moves are made by selecting a random lower ranking neighbor.
- *Reverse-greedy hill climbing with backtracking (RGBT)* – Here, after the first backtracking move, downward moves are made to the highest ranking vertex from the set of lower ranking vertices. Every step of RGBT requires examining the entire set of k neighboring solutions.
- *Combined hill climbing with backtracking (CBT)* – This backtracking heuristic knows the value of c for the problem instance, and switches from a random to a greedy search strategy after reaching a sink of rank $\leq c$.

5 Analysis of Hill-Climbing and Backtracking Heuristics

The performance of each hill climbing variant depends on the relationship between $c < N$ and k. Proofs are omitted due to lack of space, and will appear in the full version of the paper.

The analysis of the behavior of random sampling is a baseline for our analysis of local search:

Theorem 1 *m iterations of random sampling will find a solution of expected rank $\frac{N}{m+1}$ for $m \ll N$.*

We are interested in estimating the solution quality as well as the cost of hill-climbing heuristics. We must establish they reach the goal region, and if so how deep into the lower region of the landscape, namely $[v_0, v_{c/k}]$. We begin by demonstrating that the heuristics reach the goal region if $k > \lg(N/c)$.

Lemma 1 *The expected number of steps made by a hill-climbing heuristic before encountering a sink in the transient region is $(\frac{2c+1}{c+1})^k$.*

The probability that a vertex in the transient region is a sink is $p = (\frac{c+1}{2c+1})^k$. The expected number of step to reach a sink is therefore $\frac{1}{p}$. This number of steps is the same for GHC and RHC. Assuming that $k > \ln N$, we see that this number of steps together with the next lemma implies that hill-climbing heuristics easily get past the transient region $(v_c, v_{N-1}]$.

Lemma 2 *The expected reduction in rank on every move made by each heuristic in the transient region is: (1) $\frac{c}{2}$ for RHC, and (2) $c - \frac{2c}{k+1}$ for GHC.*

Theorem 2 *With high probability GHC and RHC reach the goal region $[v_0, v_c]$ when $k > \lg(N/c)$.*

When $k < \ln N$, the landscape becomes highly disconnected, and hill-climbing searches stop well before the goal region:

Lemma 3 *G is increasingly connected with high probability for $k > \ln N$ and increasingly disconnected for $k < \ln N$.*

Lemma 4 *Let $r = N/c$. Then with high probability:*

- *Random hill climbing reaches the goal region if $k > \lg r + 1$. Otherwise, it will terminate after an expected 2^{k+1} operations and return a solution of expected rank $N - c2^k$.*
- *Greedy hill climbing reaches the goal region if $k > \lg r$. Otherwise, it will terminate after an expected $k2^k$ operations and return a solution of expected rank $N - c2^k(k+2)/k - c/k$*

The hill climbing heuristics are expected to reach the goal region if $k > \lg(N/c)$. Having reached the goal region, RHC and GHC find solutions within $[v_0, v_{c/k}]$, which contains vertices that are likely to be local optima.

Theorem 3 *The expected rank of the solution found by RHC and GHC is approximately $\frac{c}{ek}$; the expected cost is $\frac{2(N-c)}{c} + ek$ and $\frac{k(N-c)}{c} + ek$ respectively.*

5.1 Comparing the hill climbing heuristics and random sampling

Random sampling samples vertices in the range $[v_0, v_{N-1}]$, remembering the lowest-ranked vertex encountered. When $c = N$, RHC and GHC essentially perform random sampling until reaching a sink. When $c < N$, local search can make average rank improvements of at most $c/2$ per evaluation, thus random sampling dominates local search in the transient region. We formulate this below, Let t denote the number of operations allocated to a given search:

Theorem 4 *Random sampling dominates local search if $k < \lg(N/c)$ or $t < N/c$.*

5.2 Efficient time strategies

Black-box optimization systems are allotted a certain computation time, and repeated iterations may be performed if a single search completes before this time bound. We compare the performance of the repeated random hill climbing, random sampling and backtracking heuristics introduced in Section 4. We show that any reasonable backtracking heuristic will avoid traversing the transient region multiple times, and thus sample a larger number of vertices in the goal region. If $k \geq \lg(kN/c)$ all backtracking heuristics described in this section are expected to reach a sink in the goal region in $4N/c + ek$ operations.

Lemma 5 *Backtracking searches allowing x backtracking steps provide a solution of equal rank to $\lceil \frac{x3^k}{2^k+3^k} \rceil$ independent hill climbing runs in the goal region.*

Proof: The expected reduction in rank due to a backtracking move is $p = \frac{c-1}{2}$. Let a backtracking move lead from a vertex v_i to a vertex v_j. The rank j of v_j is expected to be $i + p$. The probability that v_j is a sink is smaller than

$$(1 - \frac{p}{c+1+p})^k = (1 - \frac{c-1}{3c+1})^k \approx (\frac{2}{3})^k$$

source	TSP	SCS	MaxCut	MaxSat	Vertex Cover
break-even point	47	36	158	25	60
neighborhood evaluations	45	40	210	25	90

Table 1: A comparison of the extrapolated N/c from (1) the number of evaluations required for RHC to outperform random sampling, and (2) the average number of evaluations per neighborhood performed by RHC.

□

If $k < \lg(N/c)/x$ the backtracking heuristics can expect to reach a vertex in the goal region. If $\lg(N/c)/x \leq k < \lg(N/c)$ the backtracking heuristics will reach a vertex in the goal region, but will sample fewer vertices there since some backtracking moves will be used to get out of local optima in the transient region.

6 Experimental Results

We conducted two classes of experiments to validate our results. The first (and more interesting class) evaluates instances of classical combinatorial optimization problems to establish the extent to which their adjacency graphs resemble the random graphs of our model. The second class constructs the random graphs of our model and performs simulated searches on them, to establish the accuracy of our analytical results, and will be discussed in the full paper.

All experiments were conducted on an AMD Athlon 1Ghz, 768Mb RAM PC running Linux. Detailed information about our experimental platform together with full experimental results are available at [15].

6.1 Model Validation

We report experimental results about the observed performance of the most interesting of our heuristics on the following five different combinatorial optimization problems:

- *Traveling Salesman* - given an edge-weighted graph, find a shortest tour visiting each vertex exactly once. TSP is well suited to local search heuristics, as move generation and incremental cost evaluation are inexpensive. The representative test case is a complete graph on 280 vertices from TSPlib [16] a280.

- *Shortest Common Superstring* - given a set of strings, find a shortest string which contains as substrings each input string. Our test case consists of 135 strings, each of length 50, randomly chosen from 15 concatenated copies of the length-500 prefix of the Book of Genesis.

- *Max Cut* - partition a graph into two sets of vertices which maximize the number of edges between them. Max cut is a subset problem which is well suited to heuristic search. The representative test case is the graph alb5000 from TSPlib [16] with 5000 vertices and 9998 edges.

- *Max Satisfiability* - given a set of weighted Boolean clauses, find an assignment that maximizes the weights of satisfiable clauses. Max sat is a difficult subset problem not particularly well suited to local search. The representative test case is the file jnh308 from [17], a set of 900 randomly weighted clauses of 100 variables.

- *Vertex Cover* - find a smallest set of vertices incident on all edges in a graph. Vertex cover is representative of constrained subset problems, where we seek the smallest subset satisfying a correctness constraint. The changing importance of size and correctness over the course of the search are a significant complications for any heuristic. The representative test case is a randomly generated graph of 1000 vertices and 1998 edges.

Although we limit the discussion to one instance for each problem due to lack of space, we assure the reader that the reported results are representative of the other instances in our test bed. Each experiment was repeated 15 times, and the reported results pertain to mean or median values; standard deviations are given when appropriate.

In practice, each move made by a local search heuristic is less costly than that of random sampling, since moving from a solution to its neighbor may require only a partial solution evaluation, while generating a new random solution always requires a complete solution evaluation. Here, however, we look at the number of operations and do not differentiate between partial and full solution evaluations.

Our analytical results on the abstract model make several predictions about the relative performance of search heuristics. Here we boldly compare these predictions to data on our five real-world problems:

- *Random sampling outperforms hill-climbing for short searches* – Our theory predicts that random sampling should outperform hill-climbing until $\approx N/c$ evaluations. This indeed appear to be the case. Table 1 presents the average break-even point (in terms of number of evaluations) between random sampling and hill-climbing for each of our five problems.

- *Edge length is bounded by c* – Our simple model assumes that all edges are drawn uniformly at random from a span at most c solutions. This is indeed simplistic (and demonstrably false) for many problems. However our results will remain applicable under the reasonable assumption that average edge span is $\theta(c)$.

  Our theory provides two relatively independent ways to estimate N/c for a given problem from observed data, by (1) observing when the number of evaluations per neighborhood for random hill-climbing increases, and (2) observing the break-even point between random sampling and hill-climbing. These events occur when the search heuristics cross from the transient region to the goal region. See Figure 2 for an example. As shown in Table 1, these predictions are in excellent agreement with each other on all problems.

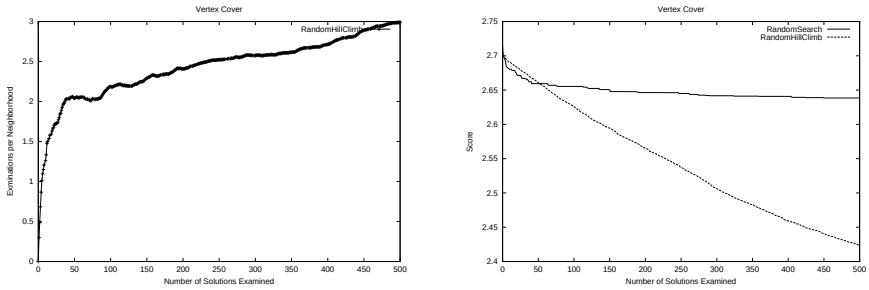

Figure 2: On the left we chart the average number of solution examinations per neighborhood as a function of the total number of solution examinations. On the right we compare the progress of the random sampling and the random hill climbing heuristics using the score of the best solution found as a function of the solutions examined.

Problem	RandHillClimb score	σ	GreedHillClimb score	σ	RandBackTrack score	σ	GreedBackTrack score	σ
TSP	6476.2	231.1	6688.5	584.0	5566.1	309.8	6200.2	318.9
SCS	2339.8	119.7	2436.5	135.1	2092.8	125.8	2239.8	165.4
MaxCut	2184.3	27.8	2119.5	27.1	2173.1	27.0	2131.5	23.3
MaxSat	3882.6	1101.2	4056.7	1060.0	2753.8	1065.3	3085.3	635.5
Vcover	2118.0	11.9	1795.4	454.7	647.5	7.4	648.6	5.5

Table 2: The performance of the heuristics. The backtracking heuristics were allowed to make ten backtracking moves.

Problem	RandHillClimb actual	predicted	GreedHillClimb actual	predicted	RandBackTrack actual	predicted	GreedBackTrack actual	predicted
TSP	758067	58635	17655120	937440	3266920	585976	1480500	1171856
SCS	92739	13605	1003995	180900	28902	135743	268836	271398
MaxCut	49950	7685	6915000	467500	119360	75215	6890010	150195
MaxSat	537	175	2800	1350	2253	1555	2410	3035
Vcover	8855	1575	341000	38500	21492	15105	24883	30085

Table 3: The number of evaluations made by the heuristics (median across the 15 experiments). The backtracking heuristics were allowed to make ten backtracking moves.

- *RHC and GHC return solutions of similar quality* – Our theory predicts that RHC and GHC yield solutions of similar quality. Table 2 gives the average scores of our heuristics on each of the five problems. Note that GHC does better than RHC on maxcut and vertex cover. These are the problems for which we least consistently predict c.

- *GHC is much more time consuming than RHC* – Our theory predicts that GHC requires roughly $k/2$ times more evaluations than RHC, where $k = n(n-1)/2$ for the swap operator employed in our simulations. Table 3 gives the median run-times of our heuristics on each of the five problems. Indeed, this is the case for all problems except vertex cover.

- *RBT and GBT yield similar search times and qualities* – Greedy backtracking performs much more poorly than our model predicts. This is because it repeatedly returns to the same local optima, a problem which occurs because our model assumed independent neighborhoods, while the *swap* is in fact symmetric. It is just this phenomenon that tabu search is designed to overcome.

- *Limitation of the model* – Table 3 shows that we underestimate the number of operations required for convergence, grossly in a few instances. We believe this is a result of the assumption of uniform distribution of the edges for each vertex. The problem appears to lie at the top of the goal region. For example, we predict that in less than 3 GHC moves, we will get from rank c to rank c/k. In fact, that data suggests that we make many more short moves in this region than predicted, almost as if c has shrank in the goal region. Well within the goal region, the probability that an edge will go down is still quite high. This is unlike what we've predicted. This observation motivates a more elaborate model with a non-uniform distribution of edges.

References

[1] E. Aarts and J. K. Lenstra. *The Traveling Salesman Problem: A Case Study*. Wiley, 1997.

[2] M. Agrawal, E. Allender, R. Impagliazzo, T. Pitassi, and S. Rudich. Reducing the complexity of reductions. In *ACM Symposium on Theory of Computing*, pages 730–738, 1997.

[3] D. Aldous and U. V. Vazirani. "Go with the Winners" algorithms. In *IEEE Symposium on Foundations of Computer Science*, pages 492–501, 1994.

[4] P. Beame, S. Cook, J. Edmonds, R. Impagliazzo, and T. Pitassi. The relative complexity of NP search problems. In *ACM Symposium on Theory of Computing*, pages 303–314, 1995.

[5] M. Bender and S. Skiena. Combinatorial dominance guarantees for heuristic algorithms. in preparation, 2001.

[6] T. Carson and R. Impagliazzo. Hill-climbing finds random planted bisections. In *Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 903–909, 2001.

[7] V. Deineko and G. Woeginger. A study of exponential neighborhoods for the traveling salesman problem and the quadratic assignment problem. *Mathematical Programming, Ser. A*, 87:519–542, 2000.

[8] G. Gutin and A. Yeo. TSP heuristics with large domination number. Technical Report PP-1998-13, Odense University, Denmark, Aug. 20, 1998.

[9] G. Gutin, A. Yeo, and A. Zverivich. Polynominal restriction approach for the atsp and stsp. In *The Traveling Salesman Problem*, to appear.

[10] B. Hajek. Cooling schedules for optimal simulated annealing. *Math. Operations Research*, 13:311–329, 1988.

[11] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? In *Proc. 26th Annual Symp. on Foundations of Computer Science*, pages 39–42, 1985. Also *J. Computer System Sci.*, 37(1), pp. 79-100, 1988.

[12] A. Juels. Topics in black box optimization. Ph.D. Thesis, University of California, Berkeley, 1996.

[13] O. Martin, S. Otto, and E. Felten. Large-step markov chains for the traveling salesman problem. *Complex Systems*, 5:299–326, 1991.

[14] C. H. Papadimitriou, A. Schaffer, and M. Yannakakis. On the complexity of local search. In *Proc. 22nd Annual ACM Symp. on Theory of Computing*, pages 438–445, 1990.

[15] V. Phan, P. Sumazin, and S. Skiena. A general black-box optimizer for combinatorial search. http://www.cs.sunysb.edu/~discropt, 2001.

[16] G. Reinelt. *TSPLIB*. University of Heidelberg, www.iwr.uni-heidelberg.de/groups/comopt/ software/TSPLIB95.

[17] M. Resende. *Max-Satisfiability Data*. Information Sciences Research Center, AT&T, www.research.att.com/ ~mgcr.

[18] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.