Combinatorial Dominance and Heuristic Search

Steven Skiena

Department of Computer Science
State University of New York
Stony Brook, NY 11794–4400

http://www.cs.sunysb.edu/∼skiena

Motivation

Engineers and theoreticians have different ideas on how to solve combinatorial optimization problems.

The two groups speak completely different languages:

Engineer: “Genetic algorithms, simulated annealing, local optimization, energy landscapes. . . ”

Theoretician: “Approximation algorithms, PTAS, NP-completeness. . . ”

Dialog

Engineer: "Our heuristics work well in practice. *Your* heuristics are very problem-specific and can be difficult to implement."

Theoretician: "We get guarantees about our heuristics. Why do you think yours are any good?"

Dialog Breaks Down

Engineer: “Your Momma.”

Theoretician: “No, *your* Momma.”

Our proposal: Combinatorial dominance analysis as a possible way to help bridge this divide.

Outline of Talk

- Combinatorial Dominance Analysis…
- …Applied to Problem-Specific Heuristics
- …Applied to Black-Box Search Heuristics
- Building a General Combinatorial Search Engine

Letters of Reference

> "She is half as good as I am, but I am the best in the world…"

> "She finished first in my class of 75 students…"

The former is akin to an approximation ratio, the latter to a *combinatorial dominance* guarantee.
Which is a stronger endorsement? Not obvious.

Combinatorial Dominance Guarantees

The *solution space* $S_P(n)$ is the set of all possible solutions x to P, where $|x| = n$.

A heuristic $H_P(I)$ offers an $f(n)$ *combinatorial dominance guarantee* if for all instances I the solution $H_P(I)$ is provably at least as good as $f(n)$ elements of $S_P(n)$.

A specific instance I' yields a *blackball bound* of $b(n)$ if the solution $H_P(I')$ is provably worse than at least $b(n)$ elements of $S_P(n)$.

Good Things about Dominance Analysis

- Opens the possibility of analyzing practical heuristics such as k-opting, which do not offer interesting worst-case approximation ratios.
- Opens the possibility of analyzing heuristics for inapproximable problems like independent set or graph coloring.
- Yields cute, simple analysis for many popular heuristics.

Bad Things about Dominance Analysis

- It gives no absolute quality guarantee.
- David Johnson “hates this sort of thing”.
- Unnatural types of heuristics with messy analysis often results in high dominance bounds

Previous Work

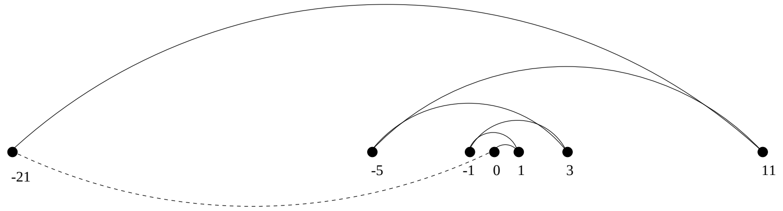
Combinatorial dominance has been independently discovered several times (including by us).

- First TSP heuristics with exponential domination number – Saranov and Doroshko (1981)
- First polynomial algorithm to dominate $\theta((n-1)!)$ TSP tours – Gutin, Yeo, and Zverovich (2001)
- Optimizing TSPs over natural exponentially-large neighborhoods – Deinieko and Woeginger (2000)

We seek to analyze popular heuristics for many problems.

Nearest Neighbor Heuristic for TSP

The nearest neighbor rule selects an arbitrary start point, and then proceed to the nearest unselected point. Repeat until all points have been selected.



Starting from the center point, the nearest neighbor tour repeatedly alternates taking points from the left and right. Since every vertical cut separating l points cuts exactly $2\min(l, n-l)$ tour edges, *this is the worst possible tour!*

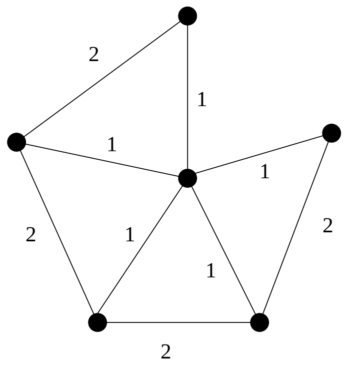
Doubling the Minimum Spanning Tree

The edge weights of a *metric* graph G satisfy the triangle inequality.

Doubling the minimum spanning tree of G, performing an Eulerian tour, and replacing redundant subtours with direct edges gives a 2-approximation for TSP tour on metric graphs.

But it can construct *the worst possible tour*!

A Bad Example for MST



G contains a single vertex distance 1 from all other vertices, a Hamiltonian path between all leaves with edges of weight 2, and all other edges of weight $1 + \epsilon$ (not shown).

The optimal TSP tour of G has weight $(n-2)(1+\epsilon)+2\cdot 1$. However, a possible Eulerian tour of this star traces all the weight-2 edges, yielding a TSP tour of weight $2(n-2)+2$.

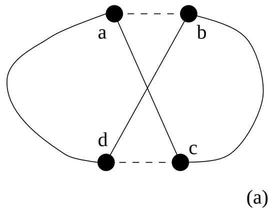
Two-Opting for TSP

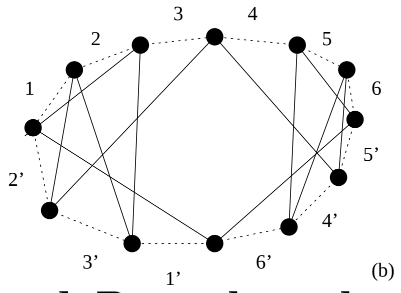
A TSP solution is k-*optimal* if there is no way to delete a subset of k edges and reconnect the resulting components to obtain a better solution.

Claim: Any 2-optimal TSP tour P dominates at least $(n/2-1)!$ other tours.

Because P is a 2-optimal tour, there does not exist an edge swap in P that leaves the tour both shorter and connected.

Thus any sequence of independent edge-swap operations must yield a tour whose cost is dominated by P.

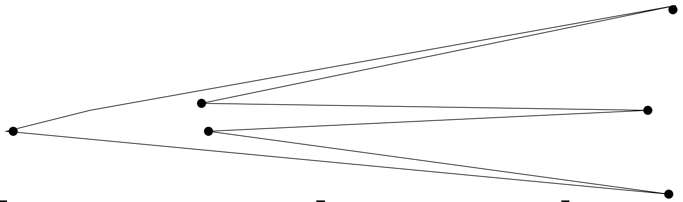




Partition P into two chains L and R, each with $n/2$ edges.
Map each edge of L to a distinct edge of R such that all pairs are an independent edge interchange operations.
Such a mapping is defined by any permutation of the edges of R which avoids having both $i \to j$ and $i+1 \to j+1$, since these two operations share an edge to be interchanged.
There are at least $(n/2-1)!$ permutations which avoid this, giving the result.

Simple Polygonization

The optimal Euclidean TSP tour in the plane is a simple polygon, i.e. non-crossing.



A zig-zag tour between two clusters can be very bad, but how *good* can it be?

What is the combinatorial dominance guarantee for any non-crossing TSP tour on n points?

Subset Problems

The hardness of subset problems results from imposing constraints which render certain elements of the search space to be infeasible solutions.

A solution x *positively dominates* solution y for a maximization (minimization) problem f if x and y are both feasible solutions and $C_p(I, x) \geq C_p(I, y)$ ($C_p(I, x) \leq C_p(I, y)$).

Any feasible solution dominates *every* infeasible solution with respect to combinatorial dominance guarantees, so the dominance number x is the positively dominated solutions plus the infeasible solutions.

Incremental Insertion

The incremental insertion heuristic successively inserts elements into the solution provided it remains feasible.

A maximization (minimization) problem P is *monotonically-constrained* if for any infeasible solution X of P, then X' is an infeasible solution for all $X \subset X'$ ($X' \subset X$).

Claim: Let P be a monotonically-constrained optimization problem. Then incremental insertion yields a $2^{n/2}$ combinatorial dominance guarantee for any instance of P.

Monotonically-constrained problems include clique, independent set, knapsack, and vertex cover.

Proof

Let S be the incremental insertion heuristic solution.
Any proper subset of S must be a feasible (and dominated) solution to P, since P is monotonically-constrained.
For *every* non-empty subset of $X = U - S$, $S \cup X$ is infeasible, or else the element of X would have also been selected by incremental insertion.
One way or the other, S must dominate $\geq 2^{n/2}$ subsets.

Local Improvement for Knapsack

In the knapsack problem, we are given a set of integers $S = \{s_1, \ldots, s_n\}$ and a capacity T. We seek $S' \subset S$ that maximizes $\Sigma_{s \in S'} s$ subject to the constraint that $\Sigma_{s \in S'} s \leq T$.

Consider an insert/exchange local improvement heuristic:

Insert operations add a new element x to the solution S' , provided the capacity T of the knapsack is not exceeded.

Exchange operations replace $x \in S'$ with $y \in S - S'$, provided the capacity is not exceeded and $y > x$.

The heuristic terminates when S' is locally optimal.

Analysis

Consider the elements of $S' = \{s_1, \ldots, s_k\}$ and $U' = S - S_k = \{u_1, \ldots, u_{n-k}\}$ where S' is locally optimal.
Assume that both S' and U' are sorted in increasing order.
We compare the median of S' with the median of U'

Case $s_{k/2} > u_{(n-k)/2}$

Then S' dominates (1) all $2^{n-k} - 1$ infeasible solutions by adding a proper subset of U' to S', and (2) all feasible but inferior solutions formed by taking a proper subset of $\{s_1 \ldots s_{k/2-1}\}$ and replacing a subset of $\{s_{k/2}, \ldots, s_k\}$ $j \leq k/2$ elements of with $i \leq j$ elements of $\{u_1, \ldots, u_{(n-k)/2}\}$.

$$f(n) \geq 2^k/2 \cdot \sum_{i=1}^{\min(n-k,k)/2} \binom{(n-k)/2}{i} \sum_{j=1}^{k/2} \binom{k/2}{j}$$

The maximum of (1) and (2) is minimized for $k = n/2$, which simplifies to

$$\begin{aligned} f(n) &\geq 2^{n/4} \cdot \sum_{i=1}^{n/4} \tbinom{n/4}{j} \geq 2^{n/4} \cdot \tbinom{n/4}{n/8} \cdot 2^{n/4-1} \\ &\geq 2^{n/2-1}\left(\frac{(n/4)}{(n/8)}\right)^{n/8} \geq 2^{5n/8}/2 \approx 1.542^n/2 \end{aligned}$$

$s_{k/2} < u_{(n-k)/2}$

Then S' dominates (1) all $2^k - 1$ feasible but inferior solutions by removing a proper subset of S' from S' and (2) all infeasible solutions formed by taking a proper subset of $\{s_{k/2}, \ldots, s_k\}$ and replacing a subset of $i \leq k/2$ elements of $\{s_1 \ldots s_{k/2-1}\}$ with $i \leq j$ elements of $\{u_{(n-k)/2}, \ldots, u_{n-k}\}$.

$$f(n) \geq 2^k/2 \cdot \sum_{i=1}^{\min(n-k,k)/2} \binom{k/2}{i} \sum_{j=1}^{(n-k)/2} \binom{(n-k)/2}{j}$$

The maximum of (1) and (2) is minimized for $k = n/2$, which simplifies as above.

Bad Example for Local Improvement

A bad example for this heuristic consists of $T = 4$, $S' = \{1, 1, 1/(n-3), \ldots, 1/(n-3)\}$ such that $\Sigma_{s \in S'} s = 3$, and $U' = \{3\}$.
This solution is locally optimal, but there are 2^{n-3} superior feasible solutions of the form 3 unioned with any proper subset of the $n-3$ elements of weight $1/(n-3)$.
Claim: The local exchange heuristic combinatorially dominates $2^{5n/8}/2 \approx 1.542^n/2$, and has a blackball bound of 2^{n-3}.

Our Results

Problem	Heuristic	Combinatorial Dominance Bounds $f(n)$ "good"	$b(n)$ "bad"	Approx Ratio
General TSP	dynasearch	$2^{n/2}$	$(n-1)!-(n/2)!$	none
General TSP	locally opt vertex insertion	$(n/3)!$	$(n-5)!$	none
General TSP	edge 2-opting	$(n/2-1)!$	$(n-7)!$	none
Metric TSP	min spanning tree	1	$(n-1)!-(n/2)!^2$	2
2D Euclidean TSP	nearest neighbor rule	1	$(n-1)!-(n/2)!^2$	$\lg n$
2D Euclidean TSP	min spanning tree	1	$(n-4)!$	2
2D Euclidean TSP	non-crossing tour	?	$(n/2-1)(n-2)!$	none
Clique/Ind. Set	incremental insertion	$2^{n/2}$	2^{n-2}	$\Omega(n^c)$
Vertex Cover	vertex insertion	$2^{n/2}$	2^{n-1}	none
Vertex Cover	edge insertion	$3^{n/3}$	$3^{n/2}$	2
Knapsack	incremental insertion	$2^{n/2}$	2^{n-3}	none
Knapsack	local exchange heuristic	$2^{5n/8-1}$	2^{n-3}	none
Knapsack	scaling PTAS	$2^{n/(c+1)}$	$2^{n/2-1}$	PTAS
Chromatic Number	incremental insertion	$\left\{ {n \atop 3n/4} \right\}$	$B(n-2)$	$\Omega(n^c)$

Black-Box Optimization

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.

Examples include simulated annealing, tabu search, genetic algorithms; each with their vocal adherents.

But which is best?

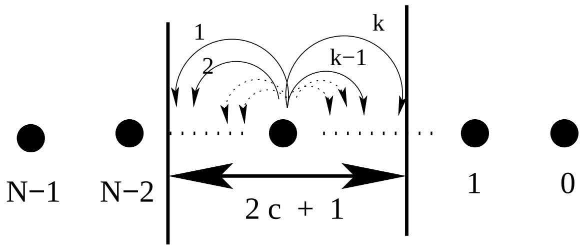
Comparing Black-Box Heuristics

Variants of search procedures are difficult to compare, even experimentally.

Wolpert and Macready's (1997) *no free lunch theorem* even claims that any heuristic that (1) extracts no information about the cost function, and (2) evaluates new solutions at the same rate performs no better than random sampling.

A Dominance Model for Search

We model the energy landscape for a given problem as a k-regular random directed graph $G = (V, E)$, where each vertex is assigned a unique rank $f(v_i) = i$.
The edge set E is constructed by drawing k edges from $v_i \in V$ to randomly chosen vertices in range $[v_{i-c}, v_{i+c}]$.



We seek a vertex of minimum rank while restricting the search to (1) generating an vertex uniformly at random, or (2) traversing an edge from a previously visited vertex.

Details

We pay unit cost to evaluate the objective function on any vertex.
All searches begin at vertex v_{N-1}.
We are interested in the expected performance of search heuristics on such graphs.
Although the model is simple, it is rich enough to study the impact of neighborhood size and structure on search performance.

Rationale

Why a regular graph of outdegree k? – most operators like vertex/edge swap are regular.

Why are edges of length bounded by c? – each local move changes only part of the solution.

Why are edge lengths uniformly distributed? – why not, especially since it simplifies analysis.

Understanding Heuristic Performance

Our graph naturally partitions into two regions by rank: the *transient* region $(v_c, v_{N-1}]$, and the *goal* region $[v_0, v_c]$.

- The range parameter c governs how fast we can descend from the start into the goal region. At least N/c local moves are necessary to enter the goal region.
- Greedy heuristics require k evaluations per node, where random search will use 2 in the transient region.
- In the transient region, the probability v is a sink is $1/2^k$.

  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 vertex is a sink increases rapidly as we move through the goal region.

  Thus it becomes progressively harder to make additional progress towards the optimum when 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 need exist for sufficiently small k.

Local Search Heuristics

- *Random hill climbing (RHC)*
- *Greedy hill climbing (GHC)*
- *Reverse greedy hill climbing (RGHC)*
- *Combined hill climbing (CHC)*

We also consider *random sampling*, which chooses solutions at random from S.

Results: Local Search

	$k < \lg N/c$		$k \geq \lg kN/c$	
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}	$N - c2^{k-1}$	$2(N-c)/c + ek$	$c/(ek + \epsilon_1)$

Backtracking Procedures

- *Greedy hill climbing with backtracking (GBT)*
- *Random hill climbing with backtracking (RBT)*
- *Reverse-greedy hill climbing with backtracking (RGBT)*
- *Combined hill climbing with backtracking (CBT)*

Results: Backtracking Procedures

	$k < \lg(N/c)/x$		$k \geq \lg N/c$, $m = 2(N-c)/c$	
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	$x'2^{k+1}$	$N - 2^k(c+1)$	$x'(m + ek)$	$c/(ekx')$

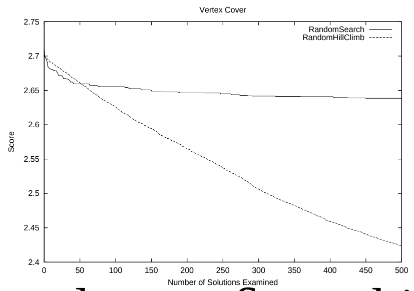
Results: Making Best Use of Time

Hill climbing heuristics make rapid initial progress toward improved solutions, but terminate quickly and thus do not use large amounts of search time effectively.

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

Experimental Results

Random sampling does better than hill-climbing for sufficiently short computations.



Our experiments grossly confirm this and several other predictions of this model.

Our theory breaks down on greedy backtracking, mostly because we assume undirected neighborhoods.

Building a General Heuristic Search Engine

The basic simulated annealing search algorithm can be implemented in roughly 30-50 lines of any modern programming language, so it is often rewritten from scratch for each new application rather than being reused.

Does it pay to develop a more sophisticated, general-purpose local-search-optimization engine?

The case for well-engineered optimization engines has clearly been made for linear programming, where commercial LP packages (such as CPLEX) significantly outperform homegrown implementations of the simplex algorithm.

Rationale for a General Search Engine

Many benefits of a general search system can result by amortizing the cost of building an infrastructure over many problems.

- *Awareness of Time*
- *Inclusion of Multiple Heuristics*
- *Parameter Tuning*
- *Testing and Evaluation Environment*
- *Visualization Environment*
- *Support for Parallelism*

Performance Improvements

Problem	Heuristic	1	5	10	15	20	25	30	45	60	90	120
VertCover	Early Version	2299	2260	2315	642	654	644	650	641	643	598	586
	Combination	**648**	**1100**	**571**	**554**	**541**	**546**	**528**	**531**	**521**	**521**	**523**
Bandwidth	Early Version	1970	**604**	**616**	**611**	**603**	**595**	622	620	609	**595**	**602**
	Combination	**693**	633	620	614	615	613	**609**	**607**	**608**	603	**602**
MaxCut	Early Version	**3972**	3972	3972	3972	3972	3969	3972	3972	3885	1373	1349
	Combination	**3972**	**1401**	**1385**	**1346**	**1462**	**1452**	**1441**	**1971**	**1689**	**1276**	**1240**
SCS	Early Version	3630	3688	1152	1158	1225	1120	1035	1013	994	902	982
	Combination	**1538**	**614**	**596**	**570**	**562**	**544**	**541**	**515**	**505**	**502**	**505**
TSP	Early Version	5101	1043	846	730	642	687	568	551	508	577	504
	Combination	**1133**	**933**	**633**	**472**	**411**	**406**	**397**	**378**	**381**	**375**	**374**
MaxSat	Early Version	27725	**572**	705	735	533	403	518	344	518	352	318
	Combination	**912**	636	**359**	**465**	**503**	**398**	**374**	**262**	**206**	**255**	**236**

A comparison between early (6 months old) and recent versions of the system.

Future Work

Use combinatorial dominance to analyze more heuristics, with tigher bounds.

Consider complexity-theoretic notions like randomized dominance guarantees and polynomial progress.

Analyze symmetric random graphs, with various edge length distributions.

Make our search engine better and more portable.

For Further Reading