Blind No More: Constant Time Non-Random Improving Moves and Exponentially Powerful Recombination

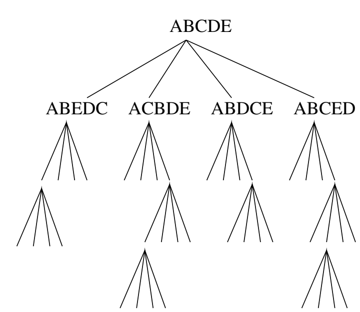
Darrell Whitley
Computer Science, Colorado State University

ACM 978-1-4503-2881-4/14/07.

What is a Landscape?

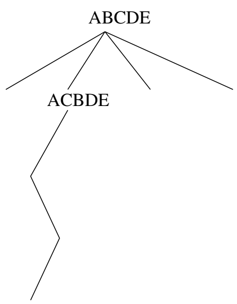
What is a Landscape?

Local Search as Tree Search



We often think of local search as greedy best-first search.
But the neighborhood really induces a connected graph.

Local Search as Tree Search



Goal: pick the best move at each level without search in O(1) time.

Introduction

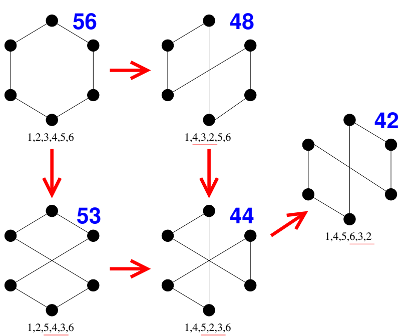
What is a landscape?

- Many different intuitive definitions
- A mathematical formalism of the *search space* of a combinatorial optimization problem

Definition: a landscape is a tuple (X, N, f)

A set of *states*	X
A *neighborhood* operator	$N : X \mapsto \mathcal{P}(X)$
A *fitness* function	$f : X \mapsto \mathbb{R}$

Introduction: a landscape



X	set of *states*
$N : X \mapsto \mathcal{P}(X)$	neighborhood operator
$f : X \mapsto \mathbb{R}$	objective function

Preliminaries

$$G(X, E)$$

is the underlying graph induced by N.
We assume G is regular with vertices of degree d.

$$\mathbf{A} \in \mathbb{R}^{|X|\times|X|}$$

is the *adjacency matrix* of G.
If x_1 and x_2 are neighbors, $A(x_1, x_2) = 1$.

$$\mathbf{\Delta} = \mathbf{A} - d\mathbf{I}$$

is the Laplacian of G.

The Wave Equation: definition 1

On an arbitrary landscape

- f and N are *unrelated*

On an elementary landscape

The wave equation

$$\mathbf{\Delta} f = \lambda f$$

- where λ is a scalar
- In other words, f is an eigenvector of the Laplacian

9

The Wave Equation: definition 1

Average change

$$\mathbf{\Delta} f = (\mathbf{A} - d\mathbf{I})f = k(\bar{f} - f)$$

$$\mathbf{\Delta} f(x) = \sum_{y \in N(x)} (f(y) - f(x)) = k(\bar{f} - f(x))$$

Average value

$$\begin{aligned}\operatorname*{avg}_{y \in N(x)} \{f(y)\} &= \frac{1}{d} \sum_{y \in N(x)} f(y) \\ &= f(x) + \frac{1}{d}\left(\sum_{y \in N(x)} f(y) - f(x)\right) \\ &= f(x) + \frac{1}{d}\Delta f(x) \\ &= f(x) + \frac{k}{d}\left(\bar{f} - f(x)\right)\end{aligned}$$

10

The Wave Equation: definition 2

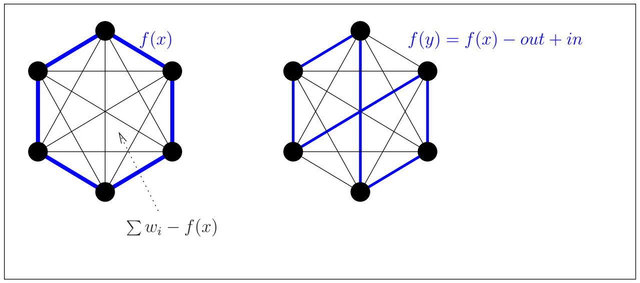
$$f(x) = \sum \text{ a subset of “components”}$$

Starting from average...

$$\operatorname*{avg}_{y \in N(x)} \{f(y)\} = f(x) + \operatorname*{avg}_{y \in N(x)} \{\text{components in} - \text{components out}\}$$

11

Example: TSP under 2-opt



- Components: set of edge weights $w_{i,j}$
- $f(x)$ = sum of edge weights induced by tour x
- There are $n(n-1)/2 - n$ weights not in tour x
- Average value of components out: $\frac{2}{n} f(x)$
- Average value of components in: $\frac{2}{n(n-3)/2} \left(\sum w - f(x)\right)$

12

The Components and $\bar{f}$

Let C denote the set of components

$0 < p_3 < 1$ is the proportion of the components in C that contribute to the cost function for any randomly chosen solution

$$\bar{f} = p_3 \sum_{c \in C} c$$

For the TSP:

$$\bar{f} = \frac{n}{n(n-1)/2} \sum_{w_{i,j} \in C} w_{i,j}$$

$$\bar{f} = \frac{2}{n-1} \sum_{w_{i,j} \in C} w_{i,j}$$

13

The Wave Equation: definition 2

$$\begin{aligned}
\operatorname*{avg}_{y \in N(x)} \{f(y)\} &= f(x) + \frac{2}{n(n-3)/2}\left(\sum w - f(x)\right) - \frac{2}{n} f(x) \\
&= f(x) + \frac{2}{n(n-3)/2}\left((n-1)/2\bar{f} - f(x)\right) - \frac{2}{n} f(x) \\
&= f(x) + \frac{(n-1)}{n(n-3)/2}(\bar{f} - f(x)) \\
&= f(x) + \frac{k}{d}(\bar{f} - f(x))
\end{aligned}$$

14

For a 5 city TSP

	ab	bc	cd	de	ae	ac	ad	bd	be	ce
ABCDE	1	1	1	1	1	0	0	0	0	0
ABEDC	1	0	1	1	0	1	0	0	1	0
ABCED	1	1	0	1	0	0	1	0	0	1
ABDCE	1	0	1	0	1	0	0	1	0	1
ACBDE	0	1	0	1	1	1	0	1	0	0
ADCBE	0	1	1	0	1	0	1	0	1	0

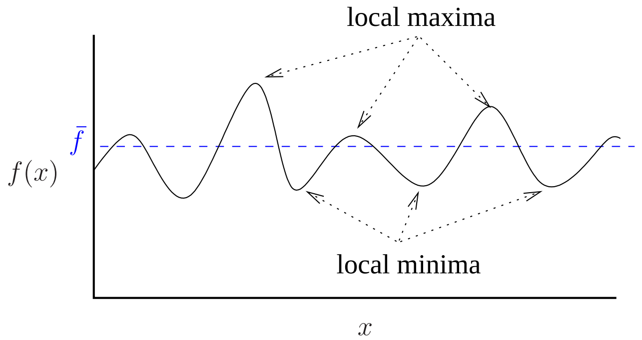
15

Looking at the neighbors in aggregate.

ab	bc	cd	de	ae	ac	ad	bd	be	ce
1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1

16

Properties



Walsh Functions

The Walsh Decomposition

$$f(x) = \sum_{j=0}^{2^n-1} w_j \psi_j(x)$$

Let $bc(j)$ count the number of 1 bits in string j:

$$\psi_j(x) = (-1)^{bc(j \wedge x)}$$

If $bc(j \wedge x)$ is odd, then $\psi_j(x) = -1$
If $bc(j \wedge x)$ is even, then $\psi_j(x) = 1$.

$$w_j = \frac{1}{2^n} \sum_{i=0}^{2^n-1} f(i)\psi_j(i)$$

Boolean Satisfiability

Given a logical expression consisting of Boolean variables, determine whether or not there is a setting for the variables that makes the expression TRUE.

Literal: a variable or the negation of a variable

Clause: a disjunct of literals

A 3SAT Example

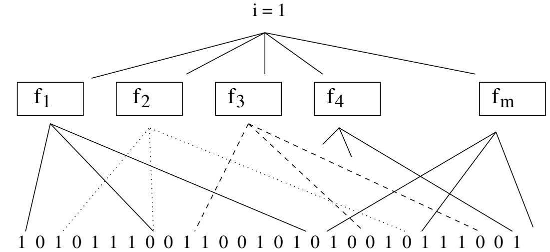
$$(\neg x_2 \vee x_1 \vee x_0) \wedge (x_3 \vee \neg x_2 \vee x_1) \wedge (x_3 \vee \neg x_1 \vee \neg x_0)$$

recast as a MAX3SAT Example

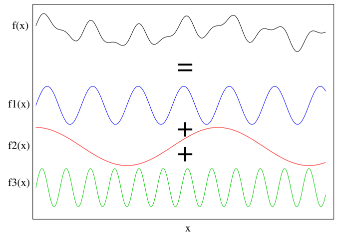
$$(\neg x_2 \vee x_1 \vee x_0) + (x_3 \vee \neg x_2 \vee x_1) + (x_3 \vee \neg x_1 \vee \neg x_0)$$

A General Model for all bounded Pseudo-Boolean Problems

$$\text{f(x)} = \sum_{i=1}^{m} f_i(\text{x, mask})$$



Why "elementary"? Components of more general landscapes



Walsh Analysis

Every n-bit MAXSAT or NK-landscape or P-spin problem is a sum of m subfunctions, f_i:

$$f(x) = \sum_{i=1}^{m} f_i(x)$$

The Walsh transform of f is is a sum of the Walsh transforms of the individual subfunctions.

$$W(f(x)) = \sum_{i=1}^{m} W(f_i(x))$$

Each subfunction f_i contributes only 2^K Walsh coefficients.
Assuming m is $O(n)$ then the number of Walsh coefficients is $O(n)$.

MAX-3SAT decomposition

MAX-3SAT is a superposition of 3 elementary landscapes

Walsh span of order p

$$\varphi^{(p)} = \sum_{\{i \,:\, bc(i)=p\}} w_i \psi_i$$

The p^{th} Walsh span is an elementary landscape

$$\Delta\varphi^{(p)} = -2p\varphi^{(p)}$$

MAX-3SAT decomposition

Recall that we can express f as:

$$f(x) = \sum_{i=1}^{m} \sum_{j=1}^{2^k} w_{m(i,j)} \psi_{m(i,j)}(x)$$

Grouping the Walsh decomposition results in

$$f(x) = \sum_{p=0}^{3} \varphi^{(p)}(x)$$

Thus MAX-3SAT is a superposition of 3-elementary landscapes

Superpositions of Elementary Landscapes

$$f(x) = f1(x) + f2(x) + f3(x) + f4(x)$$

$$f1(x) = f1_a(x) + f1_b(x) + f1_c(x)$$
$$f2(x) = f2_a(x) + f2_b(x) + f2_c(x)$$
$$f3(x) = f3_a(x) + f3_b(x) + f3_c(x)$$
$$f4(x) = f4_a(x) + f4_b(x) + f4_c(x)$$

$$\varphi^{(1)}(x) = f1_a(x) + f2_a(x) + f3_a(x) + f4_a(x)$$
$$\varphi^{(2)}(x) = f1_b(x) + f2_b(x) + f3_b(x) + f4_a(x)$$
$$\varphi^{(3)}(x) = f1_c(x) + f2_c(x) + f3_c(x) + f4_a(x)$$

$$f(x) = \varphi^{(1)}(x) + \varphi^{(2)}(x) + \varphi^{(3)}(x)$$

Constant Time Steepest Descent

Let vector w' store the Walsh coefficients including the sign relative to solution x.

$$w'_i(x) = w_i \psi_i(x)$$

Flip bit p such that $y_p \in N(x)$. Then

$$\text{if } \ p \subset i \ \text{ then } w'_i(y_p) = -w'_i(x)$$
$$\text{otherwise } \ w'_i(y_p) = w'_i(x)$$

For MAX-kSAT and NK-Landscapes flipping one bit changes the sign of only a constant number of Walsh coefficients.

Constant Time Steepest Descent

Construct a vector S such that

$$S_p(x) = \sum_{\forall b,\ p \subset b} w'_b(x)$$

In this way, all of the Walsh coefficients whose signs will be changed by flipping bit p are collected into a single number $S_p(x)$.

Constant Time Steepest Descent

Lemma 1.
Let $y_p \in N(x)$ be the neighbor of string x generated by flipping bit p.

$$f(y_p) = f(x) - 2(S_p(x))$$

If $p \subset b$ then $\psi_b(y_p) = -1(\psi_b(x))$ and otherwise $\psi_b(y_p) = \psi_b(x)$.

Corollary:
Because $f(x)$ is constant wrt p: **Maximizing $S_p(x)$ minimizes the neighborhood of $f(x)$.**

Constant Time Steepest Descent

To make this easy, assume such that **every variable occcurs exactly the same number of times.** Then each variable appears in $km/N = kc$ subfunctions.

This easy case analysis also exactly corresponds to the average complexity case (with mild restrictions on the frequency of bit flips).

29

Constant Time Steepest Descent

When one bit flips, it impacts kc subfunctions.

At most $ck(k-1)$ terms in vector S change.

When one bit flips, at most $ck(2^{k-2})$ nonlinear Walsh coefficients change, and only 1 linear term changes.

Thus, the update take $O(1)$ time.

30

The locations of the updates are obvious

$$
\begin{aligned}
S_1(y_p) &= S_1(x)\\
S_2(y_p) &= S_2(x)\\
S_3(y_p) &= S_3(x) + \sum_{\forall b,\ (p \wedge 3) \subset b} w_b'(x)\\
S_4(y_p) &= S_4(x)\\
S_5(y_p) &= S_5(x)\\
S_6(y_p) &= S_6(x)\\
S_7(y_p) &= S_7(x)\\
S_8(y_p) &= S_8(x) + \sum_{\forall b,\ (p \wedge 8) \subset b} w_b'(x)\\
S_9(y_p) &= S_9(x)
\end{aligned}
$$

31

"Old" and "New" improving moves

A "new" improving move must be a new updated location in S. Checking these takes $O(1)$ time on average.

There can be previously discovered "old" moves stored in a buffer.

For MAX-kSAT we use a fixed number of buffers to track "old" moves. This can be done (virtually always) in $O(1)$ time.

32

Next Ascent

If we want to do Next Ascent instead of Steepest Ascent, we just all of the improving moves into a buffer and pick one. Again, this takes $O(1)$ time.

33

Identifying Local Optima

If there are no improving moves, the point is a local optimum. The point is automatically identified: there are no "old" improving moves and no update is an improving move.

34

Speed Results for MAXSAT Solvers

	AdaptG2WSAT	GSAT	Walsh
UR-1000000	698.86	32.13	1.80
UR-2000000	3458.06	140.37	3.88
UR-3000000	8157.01	319.95	6.05
mem-ctrl2	4120.52	54.11	4.17
wb_4m8s-48	7339.77	83.16	6.06

Table: Time in seconds require to reach a Local Optima for several stochastic local search algorithms for MAX-kSAT problems.

35

Walsh and Hyperplane Information

Functions α and β are defined on a schema, h:

$$\alpha(h)[i] = \begin{cases} 0 & \text{if } h[i] = * \\ 1 & \text{if } h[i] = 0 \text{ or } 1 \end{cases}$$

$$\beta(h)[i] = \begin{cases} 0 & \text{if } h[i] = * \text{ or } 0 \\ 1 & \text{if } h[i] = 1 \end{cases}$$

36

Walsh and Hyperplane Information

$$f(h) = \frac{1}{|h|}\sum_{x \in h} f(x) = \sum_{j \subseteq \alpha(h)} w_j \psi_j(\beta(h))$$

$\alpha(h)$ is a mask used to select $2^{o(h)}$ relevant coefficients.
$\beta(h)$ extracts the 1 bits from the respective coefficients.
An odd number of 1 bits yields a negative sign.
Example: Let $h =$ **01** and compute $f(h)$

$$\alpha(\texttt{**01**}) = 001100 \text{ and } \beta(\texttt{**01**}) = 000100$$

$$j \in \{\texttt{000000, 000100, 001000, 001100}\}$$

$$f(\texttt{**01**}) = w_0 - w_4 + w_8 - w_{12}.$$

37

Hyperplane Initialization

The Hyperplane Advantages:
Start from 1) a good solutions and 2) in a good subspace
For each clause, select the order-3 hyperplane which yields the best average evaluation.
Use this hyperplane information to select a starting point for search (which is guaranteed to be below average).

$$f(\texttt{****010}) = w_0 + w_1 - w_2 + w_4 - w_3 + w_5 - w_6 - w_7.$$

Calculations take 23 additions (FFT/WFT Butterfly) per clause.

$$f(\texttt{****010}) = ((w_1) - (w_2 + w_4)) - ((w_3 - w_5) + (w_6 + w_7)).$$

38

Results for MAXSAT Solvers

	HyperWalsh	Walsh	GSAT	AdaptG2WSat
div-8	5467 $\pm$ 197	13761	13066 $\pm$ 187	10795 $\pm$ 93
c2-1	12819 $\pm$ 80	19524	19595 $\pm$ 116	16056 $\pm$ 251
b15	24517 $\pm$ 149	30803	31509 $\pm$ 149	27920 $\pm$ 176
mrisc	6435 $\pm$ 258	40851	39628 $\pm$ 588	34301 $\pm$ 768
rsd-37	22976 $\pm$ 167	92361	87911 $\pm$ 533	64162 $\pm$ 355
mem-c2	29649 $\pm$ 368	75729	73071 $\pm$ 584	38277 $\pm$ 535
3sat-1m	29249 $\pm$ 125	41511	40418 $\pm$ 165	30856 $\pm$ 134
3sat-2m	58415 $\pm$ 186	82898	80696 $\pm$ 220	61466 $\pm$ 182

Table: Mean and standard deviation of evaluations of solutions found after n bit flips by several algorithms.

39

Steepest Descent over Neighborhood Means

We have the vector S such that

$$S_p(x) = \sum_{\forall b,\ p \subset b} w'_b(x)$$

Also construct the vector Z such that

$$Z_p(x) = \sum_{\forall b,\ p \subset b} order(b)\ w'_b(x)$$

Note that S and Z and U all update at exactly the same locations.

Lemma 2.

$$Avg(N(y_p)) = Avg(N(x)) - 2(S_p(x)) + \frac{4}{N} Z_p(x)$$

40

Steepest Descent over Neighborhood Means

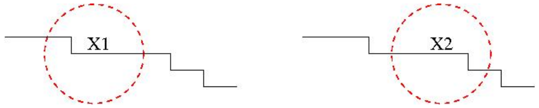
Let $U_p(x) = -2(S_p(x)) + \frac{4}{N} Z_p(x)$

$$Avg(N(y_p)) = Avg(N(x)) + U_p(x)$$

The vector $U(x)$ can now be used as a proxy for $Avg(N(x))$
Maximizing $U_p(x)$ minimizes the neighborhood of $Avg(N(y_p))$.

41

The Plateau Problem



42

The locations of the updates are obvious

$$\begin{aligned}
U_1(y_p) &= U_1(x)\\
U_2(y_p) &= U_2(x)\\
U_3(y_p) &= U_3(x) + Update\\
U_4(y_p) &= U_4(x)\\
U_5(y_p) &= U_5(x)\\
U_6(y_p) &= U_6(x)\\
U_7(y_p) &= U_7(x)\\
U_8(y_p) &= U_8(x) + Update\\
U_9(y_p) &= U_9(x)
\end{aligned}$$

43

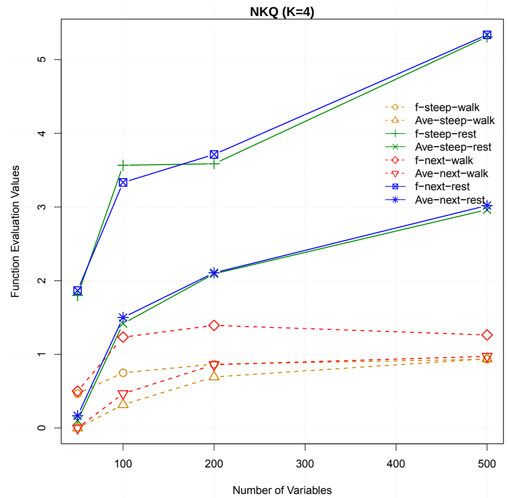
Search on an NKq-Landscape

And NKq-Landscape generates subfunctions using only q values. For $q = 2$ there are many plateaus and equal moves.

1. $f(x)$ versus $Avg(N(x))$
2. Steepest Ascent versus Next Ascent
3. Random Walk Restart (with $O(1)$ cost) versus Hard Random Restart (with $O(N)$ cost)

44

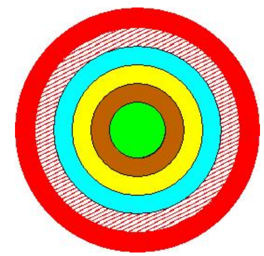
Search on an NKq-Landscape



45

Multiple Step Lookahead Local Search

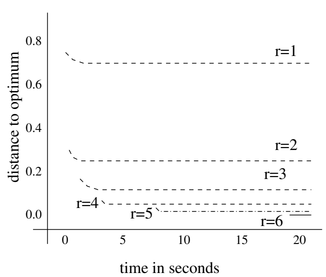
Let's return to simple local search.



What if we could look 3, 4, 5, or 6 moves ahead? WE CAN!

46

Multiple Step Lookahead Local Search



In this figure, N = 12,000, K = 2 (k=3), and q=4. The radius is 1, 2, 3, 4, 5, 6.

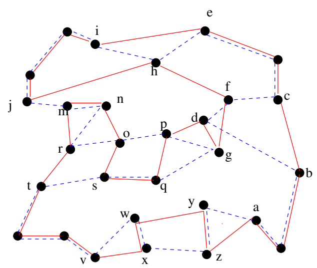
47

Conclusions

- Elementary landscapes provide an interesting tool for analyzing search in combinatorial optimization
- Linear algebraic approach to formalizing "landscape" concept for discrete problems
- These methods work for ALL k-bounded Pseudo-Boolean Functions

48

The Traveling Salesman Problem



How do we find reasonable solutions?

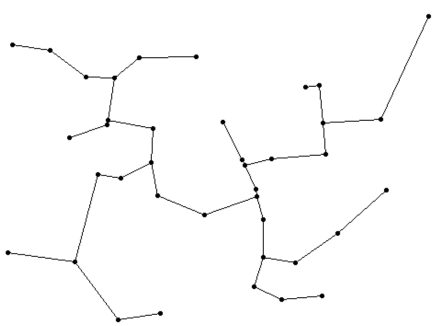
1. Construct the minimal spanning tree.

2. Then follow the minimal spanning tree cutting corners when possible.

3. No worst than 2.0 times (1.5 times) the true “optimum”
This assumes the triangle inequality holds.

4. Improve the solution using local search.

A Minimal Spannin Tree



2 Opt

A B C D E F G H I J K L M N O P Q S T U V W X Y Z

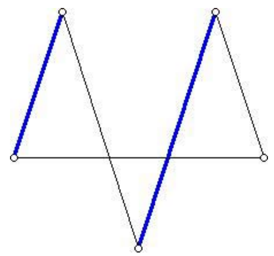
A B C D E F G ... H I J K L M N O P ... Q S T U V W X Y Z

A B C D E F G ... P O N M L K J I H ... Q S T U V W X Y Z

Do this forall pairs of break points.

$O(N^2)$

2 Opt uncrosses edges



53

3 Opt

A B C D E F G ... H I J K L M N O P ... Q S T U V ... W X Y Z

A B C D E F G ... V U T S Q ... H I J K L M N O P ... W X Y Z

Do this forall triples of break points.

$O(N^3)$

54

Lin-Kernighan K-Opt

Allows select k-opt moves so as to introduce a new "short" edge not currently being used.

Chained L-K .... uses soft restarts (double bridge moves) by allowing a series of non improving moves.

55

Partial Evaluation

Tour X: A B C D E F G...H I J K L M N O P...Q S T U V W X Y Z

Tour Y: A B C D E F G...P O N M L K J I H...Q S T U V W X Y Z

F(Y) = F(X) - edge(G,H) - edge(P,Q) + edge(G,P) + edge(H,Q)

$O(1)$ time evalution of $O(n^2)$ neighbors.

56

Branch and Bound on a Search Tree

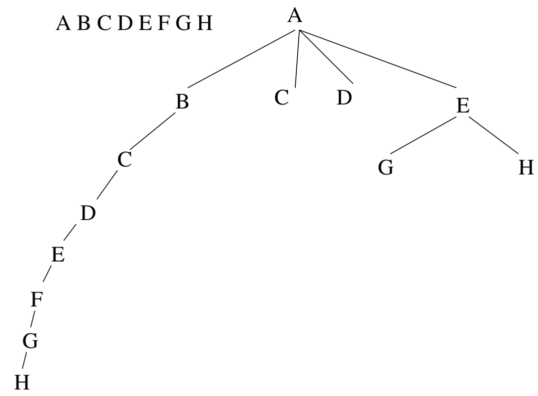
Branch and Bound

1) searches every solution

or

2) proves that all solutions in some part of the tree CANNOT be optimal.

Branch and Bound on a Search Tree



How can you prove a solution is NOT in a certain part of the tree?

The "Components" for TSP

	A	B	C	D	E
B	$w_{a,b}$				
C	$w_{a,c}$	$w_{b,c}$			
D	$w_{a,d}$	$w_{b,d}$	$w_{c,d}$		
E	$w_{a,e}$	$w_{b,e}$	$w_{c,e}$	$w_{d,e}$	
F	$w_{a,f}$	$w_{b,f}$	$w_{c,f}$	$w_{d,f}$	$w_{e,f}$

How can we speed up local search?

Only use the shortest edges. Usually only need 10 percent of the edges.

	A	B	C	D	E
B	XXX				
C	XXX	$w_{b,c}$			
D	$w_{a,d}$	$w_{b,d}$	XXX		
E	$w_{a,e}$	$w_{b,e}$	$w_{c,e}$	XXX	
F	$w_{a,f}$	XXX	XXX	$w_{d,f}$	$w_{e,f}$

On the famous ATT532 city problem, there are 532*531/2 edges.
Use the 25 shortest edges for each city (532*25).
This set includes the global optimum.

Evolutionary Algorithms

1. We generate a population of solutions

2. We take 2 parents out of the population and create 2 offspring

3. We evaluate the new offspring

4. Using "Truncation Selection" we keep the best solutions found so far (or apply some other form of selection).

Evolutionary Algorithms

For bit strings, recombination is easy.

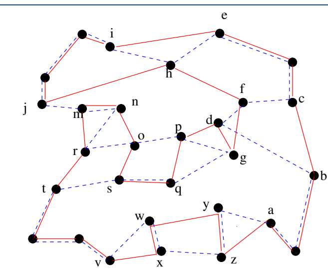
Parent One:	00000000000000000000 – 0000000000000
Parent Two:	11111111111111111111 – 1111111111111
Child:	00000000000000000000 – 1111111111111

Evolutionary Algorithms

But you can't just cut and paste permutations.

Parent One:	A B C D E F G – H I J K L
Parent Two:	D H A F I J B – A L C E G
Child??:	A B C D E F G – A L C E G

What if you could ...



... Perform Perfect Crossover for the TSP

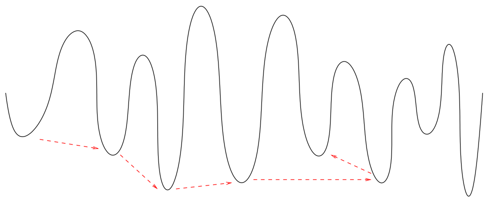
Respectful: All edges shared by the parents are inherited by the offspring.

Transmits Alleles: All edges in the offspring are inherited from parents.

Sometimes you can't.

But usually you can!

What if you could ...



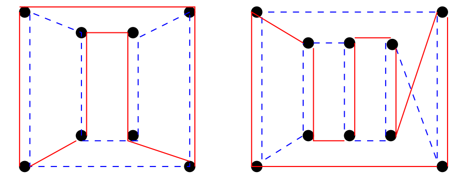
... "Tunnel" between local optima on a TSP.

Tunneling = jump from local optimum to local optimum

Sometimes you can't.

But usually you can!

Sometimes Perfect Recombination is Impossible



The Partition Crossover Theorem

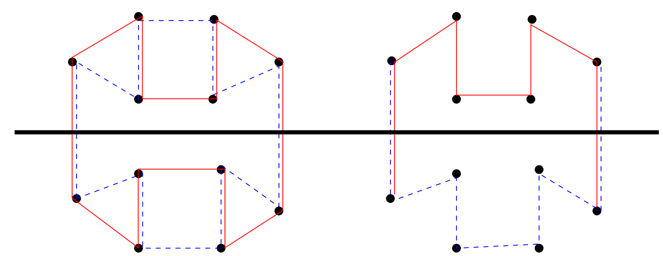
Let G be a graph produced by unioning 2 Hamiltonian Circuits.

Let G' be a reduced graph so that all common subtours are replaced by a single surrogate common edge.

If there is a partition of G' with cost 2, then the 2 Hamiltonian Circuits that make up G can be cut and recombined at this partition to create two new offspring.

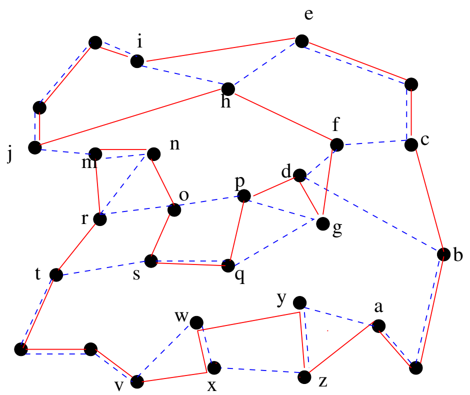
The resulting Partition Crossover is Respectful and Transmits alleles.

(Using G' makes the proof easier, but is not necessary.)

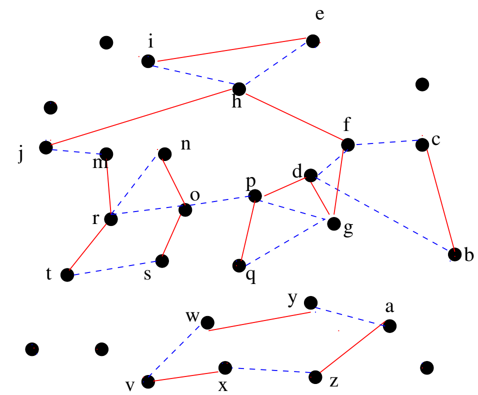


As a side effect: f(P1) + f(P2) = f(C1) + f(C2)

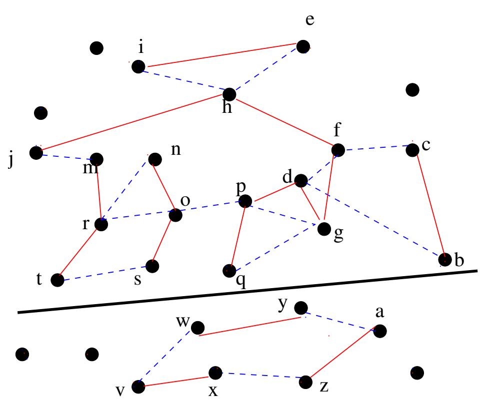
Partition Crossover

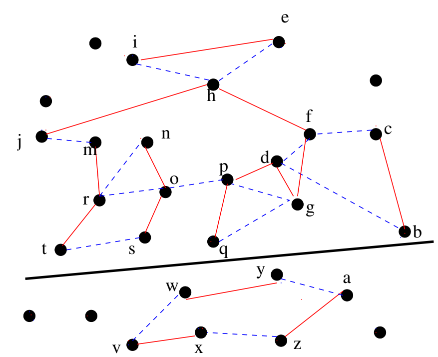


Partition Crossover



Partition Crossover in O(N) time



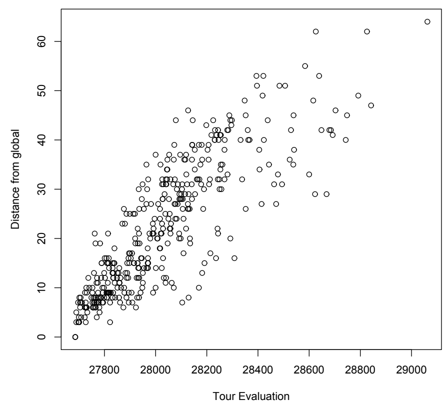


Why are the offspring usually local optima? Because the "pieces" that are recombined are already locally optimal, and they are inherited intact.

Only if 2-opt moves vertices across the partition is improvement possible.

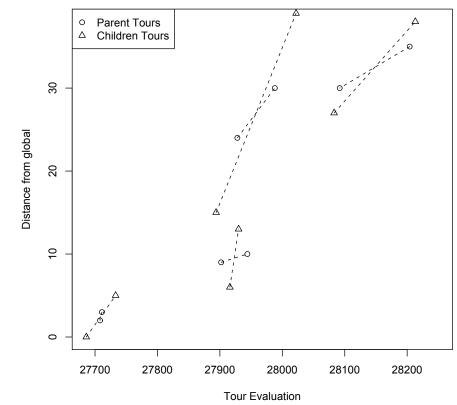
The Big Valley Hypothesis

is sometimes used to explain metaheuristic search

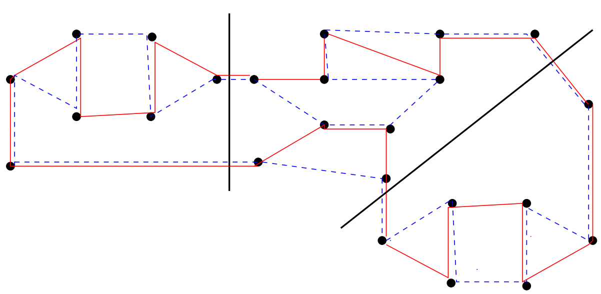


Tunneling Between Local Optima

Local Optima are "Linked" by Partition Crossover



Generalized Partition Crossover



All 1-point-crossovers are OK.

Generalize Partition Crossover is always feasible if the partitions have 2 exits (same color in and out). If a partition has more than 2 exits, the "colors" must match.

This will automatically happen if all of the partitions have cut two.

Instance	att532	nrw1379	rand1500	u1817
2-opt	3.3 ± 0.2	3.2 ± 0.2	3.7 ± 0.3	5.0 ± 0.3
3-opt	10.5 ± 0.5	11.3 ± 0.5	24.9 ± 0.2	26.2 ± 0.7
LK-search	5.3 ± 0.2	5.2 ± 0.3	10.6 ± 0.3	13.3 ± 0.4

Table: Average number of *partition components* used by GPX in 50 recombinations of random local optima found by 2-opt, 3-opt and LK-search.

Let $P1$ be a randomly generated population; Let $P2$ be a temporary child population; Forall $P1$: apply LK-search and evaluate;
1. Recombine best tour of $P1$ with the remaining $t-1$ tours; this generates a set of up to $2t$ offspring. 2. If recombination was not feasible mutate tour i and place in population $P2$; 3. Place the best solution found so far in population $P2$; 4. Select offspring to fill population $P2$; 5. For each member of $P2$: apply LK-search and evaluate; 6. $P1 = P2$; If stopping condition not met, goto 1.

Figure: The Hybrid GA; the GA is generational, but elitist.

77

	rand500	att532	nrw1379	rand1500	u1817
Hybrid GA	50/50	26/50	1/50	12/50	1/50
Chained-LK	38/50	16/50	1/50	2/50	0/50

Table: The number of times the global optimum is found by each algorithm after 1010 calls to LK-search over 50 experiments.

78

	Global Edges in Population	Global Edges in Minimum Tour	Unique Edges in Population
rand500	500 ± 0	449.68 ± 1.98	941.56 ± 1.56
att532	532 ± 0	464.1 ± 2.11	979.54 ± 1.47
nrw1379	1378.9 ± 0.04	1162.3 ± 3.44	2709.34 ± 2.25
rand1500	1500 ± 0	1301.02 ± 4.15	2871.9 ± 3.14
u1817	1815.12 ± 0.18	1562.44 ± 3.22	3616.92 ± 4.71

Table: Results obtained by running the hybrid GA for only 5 generations and *without* mutation.

79

Lin-Kernighan-Helsgaun-LKH

LKH is widely considered the best Local Search algorithm for TSP.

LKH uses deep k-opt moves, clever data structures and a fast implementation.

LKH-2 has found the majority of best known solutions on the TSP benchmarks at the Georgia Tech TSP repository that were not solved by complete solvers: http://www.tsp.gatech.edu/data/index.html.

80

LKH-2 and Clustered Instances

Empirical experiments show that LKH-2 performs significantly worse on random clustered instances than uniform random instances. We conjecture that its performance on clustered instances could be improved by exploiting crossover.

81

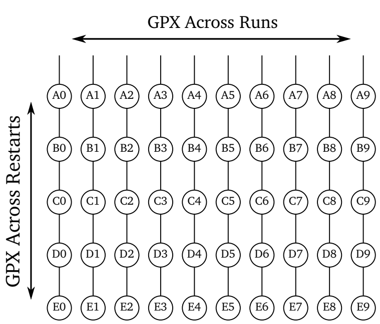
Iterative Partial Transcription and GPX

Instance	C3k.0	C3k.1	C10k.0	C10k.1	C31k.0	C31k.1
LKH-2 no x-over	0.660	0.863	1.143	1.009	1.489	1.538
LKH-2 w IPT	**0.622**	0.656	1.040	0.873	1.280	1.274
LKH-2 w GPX	**0.622**	**0.651**	**1.031**	**0.872**	**1.270**	**1.267**

The minimum percentage above the Held-Karp Bound for several clustered instances of the TSP of solutions found by ten random restarts of LKH-2 without crossover, with IPT and with GPX. Best values for each instance are in boldface. Sizes range from 3000 to 31,000 cities.

82

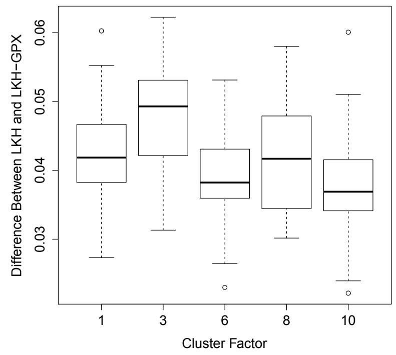
GPX Across Runs and Restarts



A diagram depicting 10 runs of multi-trial LKH-2 run for 5 iterations per run. The circles represent local optima produced by LKH-2. GPX across runs crosses over solutions with the same letters. GPX across restarts crosses over solutions with the same numbers.

83

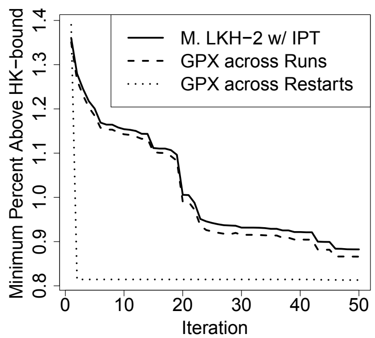
GPX on Clustered Problems



Improvement over LKH2 using GPX on Clustered Problems.

84

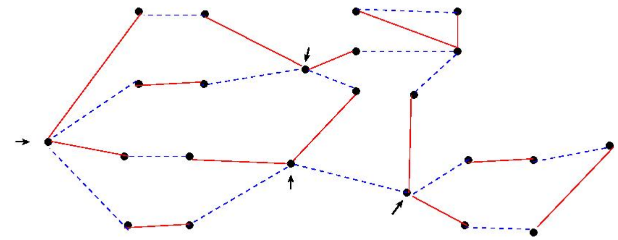
GPX on Clustered Problems



Improvement over time on a 31,000 city Dimacs Clustered Instance.

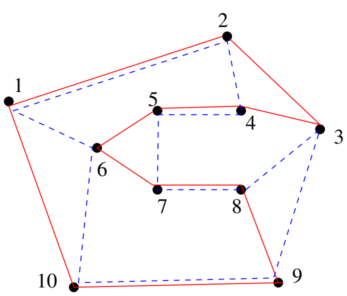
85

GPX, Cuts on Nodes of Degree 4



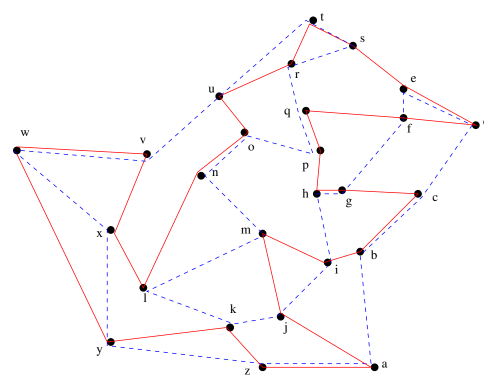
86

GPX, Cuts Crossing 4 Edges



87

GPX, Complex Cuts



88