Theoretical Foundations of Evolutionary Computation for Beginners and Veterans

GECCO '21 Companion, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8351-6/21/07□$15.00
https://doi.org/10.1145/3449726.3461407

1

Theory For Beginners and Veterans

Darrell Whitley

Computer Science
Colorado State University

GECCO 2021

2

TUTORIAL

Next Generation Genetic Algorithms:
A User's Guide and Tutorial

Handbook of Metaheuristics
Springer, 2019

Email me:

darrell.whitley@gmail.com

SUBJECT: Tutorial

3

ROAD-MAP

1. No Free Lunch
2. An Infinite Population Model of Genetic Algorithms
3. The Genetic Algorithm Fractal
4. K-bounded Functions
5. Deterministic Improving Moves and Recombination
6. Quadratization of Boolean Functions
7. MAXSAT
8. The Traveling Salesman Problem

4

No Free Lunch

All search algorithms are equivalent when compared over all possible discrete functions.

Wolpert, Macready (1995)
No free lunch theorems for search. Santa Fe Institute.

Radcliffe, Surry (1995)
Fundamental Limitations on Search Algorithms: Springer Verlag LNCS 1000.

5

Variations on No Free Lunch

Consider any algorithm A_i applied to function f_j.

On(A_i, f_j) outputs the order in which A_i visits the elements in the codomain of f_j. For every pair of algorithms A_k and A_i and for any function f_j, there exist a function f_l such that

$$On(A_i, f_j) \equiv On(A_k, f_l)$$

Consider a "BestFirst" local search with restarts.

Consider a "WorstFirst" local search with restarts.

For every j there exists an l such that

$$On(BestFirst, f_j) \equiv On(WorstFirst, f_l)$$

6

No Free Lunch BASICS

Consider $\mathbf{f(x_i) = y_j}$

An Algorithm is modeled as a permutation indicating the order in which new points in the domain, $\mathbf{x}_i$, are sampled.

Search behavior is also modeled as a permutation, indicating the order in which values in the co-domain, $\mathbf{y}_i$, are sampled.

7

$\mathbf{f(x_i)} \quad = \quad \mathbf{y_j}$

A1:	1 2 3	F1:	A B C
A2:	1 3 2	F2:	A C B
A3:	2 1 3	F3:	B A C
A4:	2 3 1	F4:	B C A
A5:	3 1 2	F5:	C A B
A6:	3 2 1	F6:	C B A

8

$f(x_i)$ = y_j

On(A1 = 1 2 3, F1 = A B C) = *On*(A1 = 3 1 2, F4 = B C A)
On(A2 = 1 3 2, F6 = C B A) = *On*(A6 = 3 2 1, F5 = B A C)

$$On(A_i, f_j) \equiv On(A_k, f_l)$$

If any 3 components are known, the fourth is determined!

9

Sharpened No Free Lunch

Let $P(f_i)$ denote the permutation of function f_i

General **NFL holds**

if an only if

the set of function is closed under permutation (C.U.P.).

Whitley 2000, Functions as Permutations, PPSN.

Schumacher, Whitley, Vose 2001. No Free Lunch and Problem Description Length. GECCO

10

It's a Zero Sum Game:

If algorithm A1 is better than A2 on f_i

then A2 is better than A1 on $(P(f_i) - f_i)$

11

Theorem: Given a finite set of M unique codomain values,
NFL holds over a set of M! functions,
where every function has an average description length of
O(M log M).

Sketch of Proof. (Very similar to Sorting complexity).
Construct a binary tree with functions located at the leaf nodes.
To uniquely label every function requires log(M!) bits.
Each label has average length O(M log M).

Note: enumeration also has O(M log M) cost:
We have M unique evaluations,
and each evaluation requires log(M) bits.

12

THE OBJECTION TO "NO FREE LUNCH"

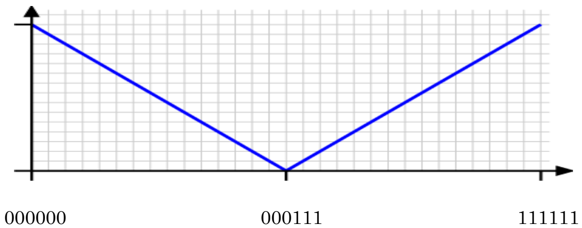
Functions in the Permutation Closure
are mostly uncompressible if every codomain value is unique.

But if there N distinct evaluations,
where N is the number of variables,
the functions in the Permutation Closure *might be* compressible.

And we don't always need a
permutation closure.

13

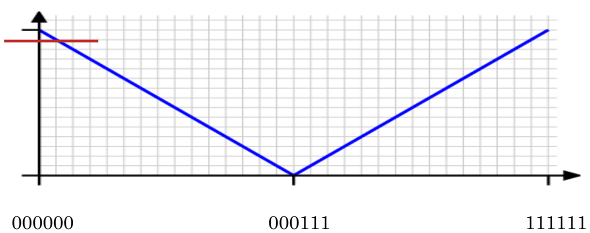
Multiple Local Optima: The *Lethals* Problem (1987).



A simple TWOMAX problem.

14

Multiple Local Optima: The *Lethals* Problem (1987).

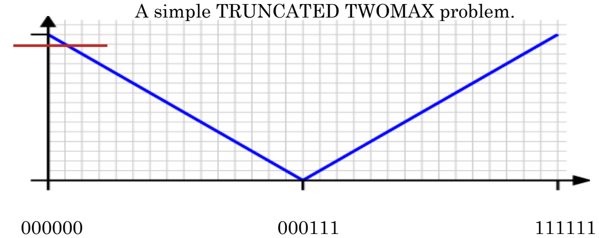


A simple TRUNCATED TWOMAX problem.

15

My Algorithm A1 has better complexity than Algorithm A2 on Truncated TwoMax.

A simple TRUNCATED TWOMAX problem.



Ha! A2 has better complexity than A1 on the Permutation Closure Excluding Truncated TwoMax!

16

BEWARE OF COMPLEXITY COMPARISONS

	G1		G2			G3			
Ux=0	1	1	1	1	1	1	1	1	1
Ux=1	4	4	3	3	3	2	2	2	2
Ux=2	1	0	2	1	0	3	2	1	0
Ux=3	0	1	0	1	2	0	1	2	3
	6	4	60	96	24	120	360	216	24

	G4					G5				
Ux=0	1	1	1	1	1	1	1	1	1	1
Ux=1	1	1	1	1	1	0	0	0	0	0
Ux=2	4	3	2	1	0	5	4	3	2	1
Ux=3	0	1	2	3	4	0	1	2	3	4
	60	320	360	96	4	6	60	120	60	6

	G6			G7				G8				
Ux=0	0	0	0	0	0	0	0	0	0	0	0	0
Ux=1	4	4	4	3	3	3	3	2	2	2	2	2
Ux=2	2	1	0	3	2	1	0	4	3	2	1	0
Ux=3	0	1	2	0	1	2	3	0	1	2	3	4
	15	24	6	80	240	144	16	90	480	540	144	6

	G9					G10				
Ux=0	0	0	0	0	0	0	0	0	0	0
Ux=1	1	1	1	1	1	0	0	0	0	0
Ux=2	5	4	3	2	1	6	5	4	3	2
Ux=3	0	1	2	3	4	0	1	2	3	4
	24	240	480	240	24	1	24	90	80	15
						Trunc				
						Two				
						MAX				

The Permutation Closure for n=4 includes approximately 208 functions
(exactly 5005 functions including symmetries),
not 29,227,900,000,000 functions as you might mistakenly think.

17

GRAY CODES VERSUS BINARY ENCODINGS.

BINARY		GRAY
000	0	000
001	1	001
010	2	011
011	3	010
100	4	110
101	5	111
110	6	101
111	7	100

	Gray Matrix	Degray Matrix
3-bits	$\begin{vmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{vmatrix}$	$\begin{vmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{vmatrix}$
5-bits	$\begin{vmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{vmatrix}$	$\begin{vmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{vmatrix}$

18

GRAY CODES VERSUS BINARY ENCODINGS.

```
R1: 000   001   010   011   100   101   110   111
R2: 000   001   011   010   110   111   101   100
R3: 000   001   010   011   101   100   111   110
R4: 000   001   011   010   111   110   100   101

R5: 000   001   010   011   100   101   110   111
```

Gray codings form a ***Group*** with an ***orbit*** of 2L or less.
A No Free Lunch result holds over at most 2L functions, where L is the length of the bit encoding.

19

1. Sharpened No Free Lunch:

NFL holds if and only if
the set of function is closed under permutation (C.U.P.).

2. Focused No Free Lunch: An example

No Free Lunch holds over at most 2L functions, where L is the length of the bit encoding.
The set of 2L functions is NOT C.U.P.

WAIT: Isn't this a contradiction?

20

1. Sharpened No Free Lunch:
requires a permutation closure.

2. Focused No Free Lunch for Bit Representations:
does not require permutation closure.

WAIT: Isn't this a contradiction?

Proofs have assumptions and conditions.

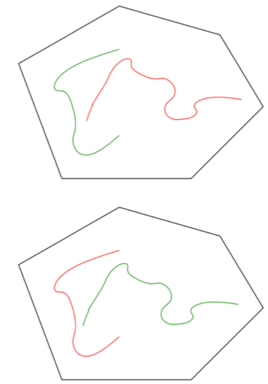
Sharpened NFL holds FORALL arbitrary search algorithms.

Focused NFL uses group theory to compare the orbits of two SPECIFIC algorithms.

These not the same, but if you are not alert,
you can draw *extremely misguided conclusions.*

21

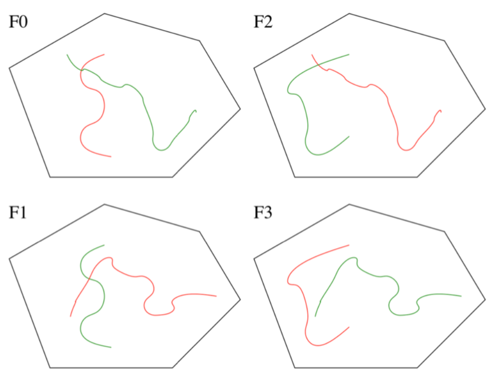
Focused No Free Lunch



Can hold over just 2 functions,
And if one is compressible,
Both are compressible..

22

Focused No Free Lunch



23

Theory can tell you what it true.

But Theory can also blind you to what is true.

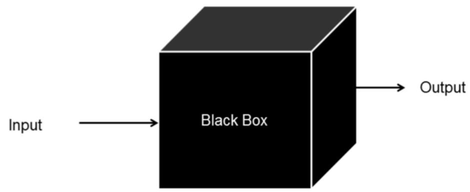
Beware of: Assumptions
Existential proofs
and Universal proofs.

For example: "The Halting Problem"

Many "Complexity Results" are Existential.

24

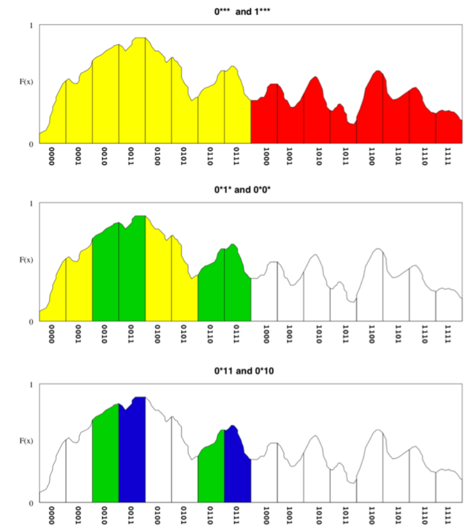
No Free Lunch is Black Box



Break the Black Box, Break No Free Lunch.

(This is almost always true.)

25



Genetic Algorithms and Hyperplane Sampling

John Holland characterized the GA as a near perfect global sampling method.

26

THE SCHEMA THEOREM

Selection Only: $P(H, t + intermediate) = P(H, t)\frac{f(H)}{\bar{f}}$.

An Exact Calculation:

$$P(H,t+1) = (1-p_c)P(H,t)\frac{f(H)}{\bar{f}} + p_c\left[P(H,t)\frac{f(H)}{\bar{f}}(1-losses) + gains\right]$$

$$P(H,t+1) = P(H,t)\frac{f(H)}{\bar{f}}(1 - p_c\, losses) + p_c gains$$

A Common Version of the "Schema Theorem":

$$P(H,t+1) \geq P(H,t)\,\frac{f(H)}{\bar{f}}\,\left[1 - p_c\frac{\Delta(H)}{L-1}\left(1 - P(H,t)\,\frac{f(H)}{\bar{f}}\right)\right](1-p_m)^{o(H)}$$

27

Hyperplane Sampling:

Approximate Claim:

A GA will achieve near optimal hyperplane sampling to
guide search toward globally competitive regions of the search space.

Two Problems:

$$P(H, t + intermediate) = P(H, t)\frac{f(H)}{\bar{f}}.$$

The population average changes every GENERATION.
This equation only looks one GENERATION into the future,
And ignores secondary interactions over time.

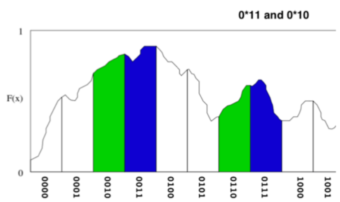
28

Hyperplane Sampling:

Approximate Claim:

A GA will achieve near optimal hyperplane sampling to guide search toward globally competitive regions of the search space.

Two Problems:



Small populations do not sample small hyperplanes.

This inherently yields a course grain sample, that is changing over time.

29

Hyperplane Sampling:

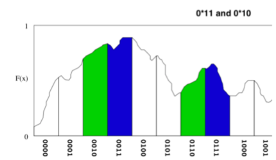
Again, you need to be careful with THEORY.

Theoreticans are human like the rest of us,
And are just as prone to "hyping results" as anyone else.

What are the ASSUMPTIONS?

Are the results Universal or Existential?

Are the results being "stretched" beyond the proof.



30

An Infinite population GA model (Whitley 1990):

An Executable Model

$$P(Z,t+1) = P(Z,t)\frac{f(Z)}{\bar{f}}(1 - \{P_c \text{ losses}\}) + \{P_c \text{ gains.}\}$$

$$\text{losses} = p_{I0}\frac{f(111)}{\bar{f}}P(111,t) + p_{I0}\frac{f(101)}{\bar{f}}P(101,t)$$
$$+p_{I1}\frac{f(110)}{\bar{f}}P(110,t) + p_{I2}\frac{f(011,t)}{\bar{f}}P(011,t).$$

$$\text{gains} = p_{I0}\frac{f(001)}{\bar{f}}P(001,t)\frac{f(100)}{\bar{f}}P(100,t)$$
$$+p_{I1}\frac{f(010)}{\bar{f}}P(010,t)\frac{f(100)}{\bar{f}}P(100,t)+p_{I1}\frac{f(011)}{\bar{f}}P(011,t)\frac{f(100)}{\bar{f}}P(100,t)$$
$$+p_{I2}\frac{f(001)}{\bar{f}}P(001,t)\frac{f(110)}{\bar{f}}P(110,t)+p_{I2}\frac{f(001)}{\bar{f}}P(001,t)\frac{f(010)}{\bar{f}}P(010,t)$$

31

An Infinite population GA model:

An Executable Model

$$P(Z,t+1) = P(Z,t)\frac{f(Z)}{\bar{f}}(1 - \{P_c \text{ losses}\}) + \{P_c \text{ gains.}\}$$

$$\text{losses} = p_{I0}\frac{f(111)}{\bar{f}}P(111,t) + p_{I0}\frac{f(101)}{\bar{f}}P(101,t)$$
$$+p_{I1}\frac{f(110)}{\bar{f}}P(110,t) + p_{I2}\frac{f(011,t)}{\bar{f}}P(011,t).$$

$$\text{gains} = p_{I0}\frac{f(001)}{\bar{f}}P(001,t)\frac{f(100)}{\bar{f}}P(100,t)$$
$$+p_{I1}\frac{f(010)}{\bar{f}}P(010,t)\frac{f(100)}{\bar{f}}P(100,t)+p_{I1}\frac{f(011)}{\bar{f}}P(011,t)\frac{f(100)}{\bar{f}}P(100,t)$$
$$+p_{I2}\frac{f(001)}{\bar{f}}P(001,t)\frac{f(110)}{\bar{f}}P(110,t)+p_{I2}\frac{f(001)}{\bar{f}}P(001,t)\frac{f(010)}{\bar{f}}P(010,t)$$

This model can be executed to track the GA behavior on strings up to about 10 to 20 bits.

We can easily add mutation.

Compute the probability of generating the string of all ZEROs, 0^L

Use XOR to remap the equation to every string in the search space.

32

An Infinite population model:

Vose/Liepins: Crossover and Mutation Probabilities can be expressed as a matrix.

SAME COMPUTATION

Compute the probability of generating the string of all ZEROs, 0^L

33

An Infinite population model:

An Exact Calculation:

$$P(i,t+1) = P(i,t)\frac{f(i)}{\bar{f}}(1-p_c\, losses) + p_c gains$$

The Vose/Liepins Model:

$$s_i^t = P(S_i,t)f(S_i)/\bar{f}$$

$$p_i^{t+1} = (\rho_i\ s)^T M(\rho_i\ s) = P(i,t)\frac{f(i)}{\bar{f}}(1-p_c\, losses) + p_c gains$$

Mutation and Crossover are in matrix *M.*

34

An Infinite population model:

A permutation function, ρ, is defined as follows:

$$\rho_j < s_0, ..., s_{N-1} >^T = < s_{j\oplus 0}, ..., s_{j\oplus(N-1)} >^T$$

A general operator $\mathcal{M}$ can now be defined over s which remaps $s^T M s$ to cover all strings in the search space.

$$\mathcal{M}(s) = < (\rho_0\ s)^T M \rho_0\ s, ..., (\rho_{N-1}\ s)^T M \rho_{N-1}\ s >^T$$

35

An Infinite population model:

SO WHAT?

We learned about the dynamics of Genetic Algorithms.

We learned that is the GA has it own modes of failure.

Dynamics are more complex than "Hyperplane Sampling."

But we also learned some amazing things too.

36

An Infinite population GA model:

An Executable Model

$$P(Z,t+1) = P(Z,t)\frac{f(Z)}{\bar{f}}(1 - \{P_c \text{ losses}\}) + \{P_c \text{ gains.}\}$$

$$\text{losses} = p_{I0}\frac{f(111)}{\bar{f}}P(111,t) + p_{I0}\frac{f(101)}{\bar{f}}P(101,t)$$
$$+p_{I1}\frac{f(110)}{\bar{f}}P(110,t) + p_{I2}\frac{f(011,t)}{\bar{f}}P(011,t).$$

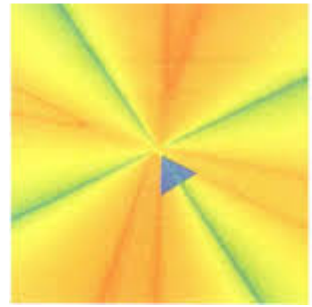
$$\text{gains} = p_{I0}\frac{f(001)}{\bar{f}}P(001,t)\frac{f(100)}{\bar{f}}P(100,t)$$
$$+p_{I1}\frac{f(010)}{\bar{f}}P(010,t)\frac{f(100)}{\bar{f}}P(100,t)+p_{I1}\frac{f(011)}{\bar{f}}P(011,t)\frac{f(100)}{\bar{f}}P(100,t)$$
$$+p_{I2}\frac{f(001)}{\bar{f}}P(001,t)\frac{f(110)}{\bar{f}}P(110,t)+p_{I2}\frac{f(001)}{\bar{f}}P(001,t)\frac{f(010)}{\bar{f}}P(010,t)$$

The Physic community, about 1993:
This equation is potentially (probably) chaotic

37

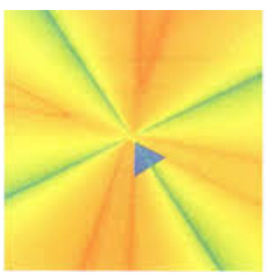
The genetic algorithm **fractal**

J Juliany, MD **Vose** - Evolutionary Computation, 1994 - MIT Press



38

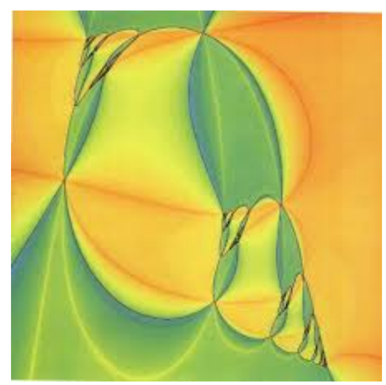
The Genetic Algorithm Fractal:



The blue triangle is the feasible population space.
The colors represent convergence speed.
The colors are inverted inside the feasible population space.

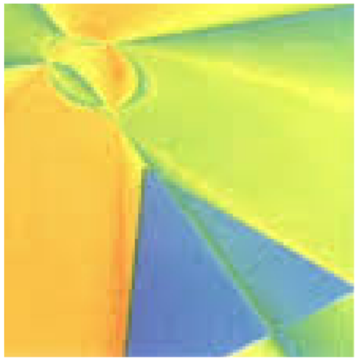
39

The Genetic Algorithm Fractal:



40

The Genetic Algorithm Fractal:



The dynamics appear to show that Evolutionary Algorithms can be influenced by attractors that are OUTSIDE the feasible population representation space.

Note how the pattern outside the feasible population extends through the feasible population.

41

Limitations of the Infinite Population Model:

An Exact Calculation:

$$P(i,t+1) = P(i,t)\frac{f(i)}{\bar{f}}(1-p_c\, losses) + p_c gains$$

The Vose/Liepins Model:

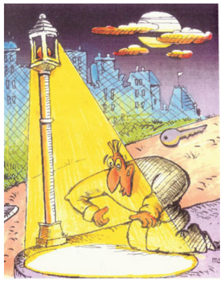
$$s_i^t = P(S_i,t)f(S_i)/\bar{f}$$

$$p_i^{t+1} = (\rho_i\ s)^T M(\rho_i\ s) = P(i,t)\frac{f(i)}{\bar{f}}(1-p_c\, losses) + p_c gains$$

Uses fitness proportional selection,
Because it is easy to model mathematically.

42

Limitations of the Infinite Population Model:



Another risk with theory:

Looking where you have a mathematical flashlight.

Perhaps sometimes OK.

But Be Aware,
and Beware.

Uses fitness proportional selection,
Because it is easy to model mathematically.

43

Elementary Landscapes:

A "Wave Equation" holds for certain problem classes and operators:

- Traveling Salesman Problem
- Graph Coloring, Number Partitioning, …
- MAXSAT, NK-Landscapes, …
- *all k-bounded pseudo-Boolean functions (thanks, Andrew Sutton).*

- Lov Grover (1992) and Peter Stadler (1996)

44

The Wave Equation for Neighborhoods

Average value

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

45

The Wave Equation: A Simpler Explanation.

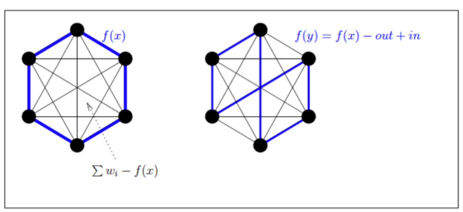
$$f(x) = \sum \text{ a subset of ``components''}$$

Starting from average...

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

46

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: $\frac{2}{n(n-3)/2}\left(\sum w - f(x)\right)$

47

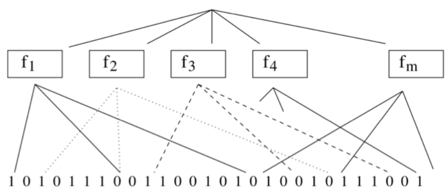
For the Traveling Salesman Problem

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

48

K-bounded Pseudo-Boolean Functions

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



49

K-bounded Pseudo-Boolean Functions

MAXSAT

NK Landscapes

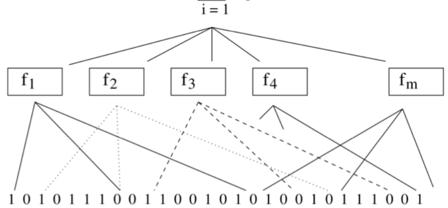
Spin Glass Problems

Bit Representations
that are not K_bounded
can be transformed into K bounded functions.

50

K-bounded Pseudo-Boolean Functions

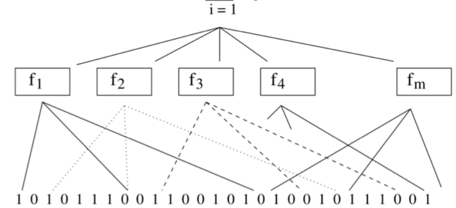
$$f(x) = \sum_{i=1}^{m} f_i(x, \text{mask})$$



51

K-bounded Pseudo-Boolean Functions

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

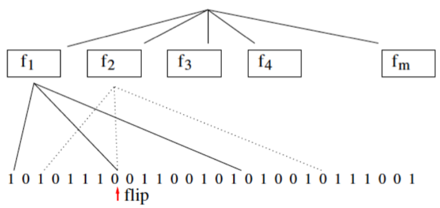


Why is this important?

52

K-bounded Pseudo-Boolean Functions

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



The location of *Improving Moves* can be computed on average in *constant* time. Special versions of this are known from 1992.
A general proof is given by: Whitley et al. 2013 AAAI.

53

Constructing the Multilinear form.

$$f(000) = w_0$$
$$f(001) = w_0 + w_1$$
$$f(010) = w_0 + w_2$$
$$f(011) = w_0 + w_1 + w_2 + w_3$$
$$f(100) = w_0 + w_4$$
$$f(101) = w_0 + w_4 + w_1 + w_5$$
$$f(110) = w_0 + w_4 + w_2 + w_6$$
$$f(111) = w_0 + w_1 + w_2 + w_4 + w5 + w_6 + w_7$$

There is only one unknown at very step.

$$f(x_3, x_2, x_1) = w_0 + w_1x_1 + w_2x_2 + w_3x_1x_2 + w_4x_3 + w_5x_3x_1 + w_6x_3x_2 + w_7x_3x_2x_1$$

54

The Multilinear form

$w_0 = f(000)$
$w_1 = f(001) - w_0$
$w_2 = f(010) - w_0$
$w_3 = f(011) - w_0 - w_1 - w_2$
$w_4 = f(100) - w_0$
$w_5 = f(101) - w_0 - w_4 - w_1$
$w_6 = f(110) - w_0 - w_4 - w_2$

$f(x) = w_0 + w_1x_1 + w_2x_2 + w_4x_4 + w_3x_1x_2 + w_5x_1x_4 + w_6x_2x_4$

55

The Multilinear form

$w_0 = f(000)$
$w_1 = f(001) - w_0$
$w_2 = f(010) - w_0$
$w_3 = f(011) - w_0 - w_1 - w_2$
$w_4 = f(100) - w_0$
$w_5 = f(101) - w_0 - w_4 - w_1$
$w_6 = f(110) - w_0 - w_4 - w_2$

$f(x) = w_0 + w_1x_1 + w_2x_2 + w_4x_4 + w_3x_1x_2 + w_5x_1x_4 + w_6x_2x_4$

The multilinear form can also tell us about Nonlinear "LINKAGE" between bits in $O(n^2)$ time.

56

HOW TO SOLVE ONEMAX IN O(N) TIME BY SAMPLING

$f(0000) = 0, w_0 = 0$

$f(0001) = 1, w_1 = 1$

$f(0010) = 1, w_2 = 1$

$f(0100) = 1, w_4 = 1$

$f(1000) = 1, w_8 = 1$

Construct the linear terms of the multilinear form.

Test the hypothesis that the function is linear.

$$f(x_4, x_3, x_2, x_1) = w_0 + w_1 x_1 + w_2 x_2 + w_4 x_3 + w_8 x_4$$

57

THE WALSH/DISCRETE FOURIER POLYNOMIAL

$$\vec{w} = \frac{1}{2^L} \vec{f}^{\,T} M$$

$$\begin{bmatrix} 4.5 \\ -0.50 \\ 0.25 \\ 1.25 \\ 0.00 \\ -0.50 \\ 1.25 \\ -1.25 \end{bmatrix} = \frac{1}{8} \begin{bmatrix} 5 \\ 7 \\ 2 \\ 4 \\ 6 \\ 1 \\ 3 \\ 8 \end{bmatrix}^T \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

58

THE WALSH/DISCRETE FOURIER POLYNOMIAL

$$\vec{w} = \frac{1}{2^L} \vec{f}^{\,T} M$$

$$\begin{bmatrix} 4.5 \\ -0.50 \\ 0.25 \\ 1.25 \\ 0.00 \\ -0.50 \\ 1.25 \\ -1.25 \end{bmatrix} = \frac{1}{8} \begin{bmatrix} 5 \\ 7 \\ 2 \\ 4 \\ 6 \\ 1 \\ 3 \\ 8 \end{bmatrix}^T \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

$$\begin{matrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{matrix} \qquad \begin{matrix} 1 & 1 \\ 1 & -1 \end{matrix}$$

The decomposition pattern of the Walsh Matrix.

M M
M -M

59

THE WALSH/DISCRETE FOURIER POLYNOMIAL

$$\vec{w} = \frac{1}{2^L} \vec{f}^{\,T} M$$

$$\begin{bmatrix} 4.5 \\ -0.50 \\ 0.25 \\ 1.25 \\ 0.00 \\ -0.50 \\ 1.25 \\ -1.25 \end{bmatrix} = \frac{1}{8} \begin{bmatrix} 5 \\ 7 \\ 2 \\ 4 \\ 6 \\ 1 \\ 3 \\ 8 \end{bmatrix}^T \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

$$\begin{matrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{matrix} \qquad \begin{matrix} 1 & 1 \\ 1 & -1 \end{matrix}$$

M M
M -M

AND, *YES* THERE IS A FAST DISCRETE FOURIER TRANSFORM with O(n log n) complexity.

60

The Walsh/Discrete Fourier Polynomial

x	w_i	$W(f_1)$	$W(f_2)$	$W(f_3)$	$W(f(x))$
0000	w_0	0.875	0.875	0.875	2.625
0001	w_1	-0.125	0	0.125	0
0010	w_2	-0.125	-0.125	0.125	-0.125
0011	w_3	-0.125	0	-0.125	-0.250
0100	w_4	0.125	0.125	0	0.250
0101	w_5	0.125	0	0	0.125
0110	w_6	0.125	0.125	0	0.250
0111	w_7	0.125	0	0	0.125
1000	w_8	0	-0.125	-0.125	-0.250
1001	w_9	0	0	0.125	0.125
1010	w_{10}	0	-0.125	0.125	0
1011	w_{11}	0	0	-0.125	-0.125
1100	w_{12}	0	0.125	0	0.125
1101	w_{13}	0	0	0	0
1110	w_{14}	0	0.125	0	0.125
1111	w_{15}	0	0	0	0

$$f_1 = (\neg x_2 \vee x_1 \vee x_0)$$
$$f_2 = (x_3 \vee \neg x_2 \vee x_1)$$
$$f_3 = (x_3 \vee \neg x_1 \vee \neg x_0)$$

61

The Walsh/Discrete Fourier Polynomial

THEOREM:

For a k-bounded function,
if there are O(n) subfunctions (m=O(n)),
The Walsh polynomial is O(n) in size.

PROOF: by construction there are at most 2^km coefficients

62

The Walsh/Discrete Fourier Polynomial

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

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

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

63

The Walsh/Discrete Fourier Polynomial

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

$\alpha(h)$ 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 $h = \star\star 01 \star\star$ and compute $f(h)$

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

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

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

64

We can compute one HYPERPLANE average in O(N) time for K-bounded problems.

For K-bounded problems,
we can exactly compute
the hyperplane averages that Holland
was trying to estimate!

This can yield the global optimum, BUT ONLY
IF there is no deception, or only low order deception.

THE ONLY CHALLENGING
K-bounded Problems Are DECEPTIVE.

(SAT Solvers have used this trick for decades!)

65

Gray Box Optimization:

From the Fourier/Walsh Polynomial we can compute the location of improving moves on average in O(1) time

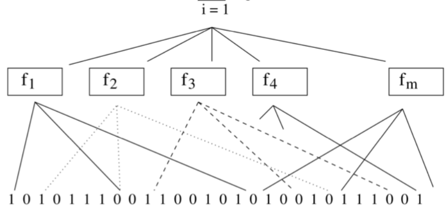
$$W(f(x)) = \sum_{i=1}^{m} W(f_i(x))$$

The Walsh
Polynomial
Is zero
centered.

66

K-bounded Pseudo-Boolean Functions

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

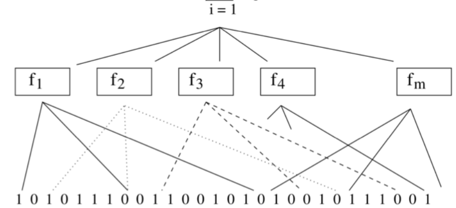


Each f_i with k=3 has only 4 nonlinear coefficients

67

K-bounded Pseudo-Boolean Functions

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

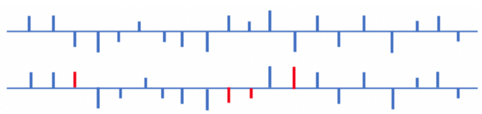


$$f(x_3, x_2, x_1) = w_0 + w_1x_1 + w_2x_2 + w_3x_1x_2 + w_4x_3 + w_5x_3x_1 + w_6x_3x_2 + w_7x_3x_2x_1$$

68

Gray Box Optimization:

$$f(x) = \psi_x w_1 + \psi_x w_2 + \psi_x w_3 + \psi_x w_4 + \psi_x w_5 + \psi_x w_6 + \psi_x w_7 + \psi_x w_8$$
$$+ \psi_x w_{1,2} + \psi_x w_{2,3} + \psi_x w_{3,4} + \psi_x w_{1,4} + \psi_x w_{3,5} + \psi_x w_{5,6}$$
$$+ \psi_x w_{6,7} + \psi_x w_{5,7} + \psi_x w_{7,8} + \psi_x w_{8,4}$$
$$+ \psi_x w_{5,6,7} + \psi_x w_{4,7,8}$$



(WARNING: I shortened the notation for compactness.)

Constant Time Improving Moves

Assume we flip bit p to move from x to $y_p \in N(x)$. Construct a vector $Score$ such that

$$Score(x, y_p) = f(y_p) - f(x)$$

$$Score(x, y_p) = -2 \left\{ \sum_{\forall b,\ p \subset b} -1^{b^T x} w_b(x) \right\}$$

All Walsh coefficients whose signs will be changed by flipping bit p are collected into a single number $Score(x, y_p)$.

See Hoos and Stützle, Stochastic Local Search, 2005

Constant Time Improving Moves

IMPROVING_MOVE_LIST: y_3, y_5

$$\begin{aligned}
Score(y_p, y_1) &= Score(x, y_1) \\
Score(y_p, y_2) &= Score(x, y_2) \\
Score(y_p, y_3) &= Score(x, y_3) - 2\Big(\sum_{\forall b,\ (p \wedge 3) \subset b} w'_b(x)\Big) \\
Score(y_p, y_4) &= Score(x, y_4) \\
Score(y_p, y_5) &= Score(x, y_5) \\
Score(y_p, y_6) &= Score(x, y_6) \\
Score(y_p, y_7) &= Score(x, y_7) \\
Score(y_p, y_8) &= Score(x, y_8) - 2\Big(\sum_{\forall b,\ (p \wedge 8) \subset b} w'_b(x)\Big) \\
Score(y_p, y_9) &= Score(x, y_9)
\end{aligned}$$

IMPROVING_MOVE_LIST: y_8, y_5

Constant Time Improving Moves

OTHER THEORETICAL RESULTS:

No difference in runtime cost
For BEST-FIRST, and NEXT-FIRST.

You can flip 2 or 3 or 10 bits at once, still in O(1) time.
(We will come back to this.)

A MAX-3SAT INSTANCE

-1 9 -8
-1 9 8
1 -9 8
-1 -9 -8
-5 -4 -6
-5 4 6
-5 5 -6
5 -4 6
2 1 7
2 -1 7
-2 -1 7
-2 -1 7
3 6 8
3 6 -8
3 -6 8
-3 6 -8

73

A MAX-3SAT INSTANCE

-1 9 -8	T
-1 9 8	F
1 -9 8	T
-1 -9 -8	T
-5 -4 -6	T
-5 4 6	T
-5 5 -6	T
5 -4 6	T
2 1 7	T
2 -1 7	T
-2 -1 7	T
-2 -1 7	T
3 6 8	T
3 6 -8	T
3 -6 8	F
-3 6 -8	T

74

A MAX-3SAT INSTANCE

-1 9 -8	T
-1 9 8	F
1 -9 8	T
-1 -9 -8	T
-5 -4 -6	T
-5 4 6	T
-5 5 -6	T
5 -4 6	T
2 1 7	T
2 -1 7	T
-2 -1 7	T
-2 -1 7	T
3 6 8	T
3 6 -8	T
3 -6 8	F
-3 6 -8	T

Randomly flipping bits would be really silly.

75

A MAX-3SAT INSTANCE

-1 9 -8	T
-1 9 8	F
1 -9 8	T
-1 -9 -8	T
-5 -4 -6	T
-5 4 6	T
-5 5 -6	T
5 -4 6	T
2 1 7	T
2 -1 7	T
-2 -1 7	T
-2 -1 7	T
3 6 8	T
3 6 -8	T
3 -6 8	F
-3 6 -8	T

Randomly flipping bits would be really silly.

Enumerative Local Search would be really silly.

76

A MAX-3SAT INSTANCE

-1 9 -8
-1 9 8 F <<
1 -9 8
-1 -9 -8
-5 -4 -6
-5 4 6
-5 5 -6
5 -4 6
2 1 7
2 -1 7
-2 -1 7
-2 -1 7
3 6 8
3 6 -8
3 -6 8 F << Flip 8 to -8?
-3 6 -8

77

A MAX-3SAT INSTANCE

-1 9 -8
-1 9 8
1 -9 8
-1 -9 -8

78

A MAX-3SAT INSTANCE

-1 9 -8
-1 9 8
1 -9 8
-1 -9 -8

x1 IFF x9 and ~x8

Construct the Truth Table and the DNF form

Convert to CNF using DeMorgan's Rule:

-1 9 -8
-1 9 8
1 -9 8
-1 -9 -8

79

REAL WORLD INDUSTRIAL SAT PROBLEMS FROM THE SAT COMPETITIONS

Typically have less than 2N nonlinear terms over N variables.

80

In Modern Combinatorial Optimization,
Uniform Random Mutation
Makes no Sense

81

In Modern Combinatorial Optimization,
Uniform Random Mutation
Makes no Sense

If we want to solve real world Problems
Uniform Random Mutation
Makes no Sense

82

In Modern Combinatorial Optimization,
Uniform Random Mutation
Makes no Sense

If we want to solve real world Problems
Uniform Random Mutation
Makes no Sense

In Modern Biology
Uniform Random Mutation
Makes no Sense

83

How Genes are Encoded is Important

CTAGTCGATTCGTAATCATCCGACGTACGTACTGAGTT

Neighbor-Dependent Mutation: "a cytosine followed by a guanine is about 10 times more mutable than a cytosine in any other dinucleotide."

CpG mutations account for 1/3
of genetic based diseases in humans.

Single Nucleotide Polymorphism is associated with "mutation hotspots" and the most common type of genetic variation.

84

Limiting Mutation in DNA

There are *ultraconserved* segments of DNA that have survived unchanged for 100 million years in vertebrates.

There are highly sophisticated error correction mechanisms.

Gene Duplication and Genetic Redundancy is common.

Most "mutations" have small effects or no effect.

85

Back to K bounded Functions

a: 1 -0 6	l: -6 10 13	q: -11 16 17	v: -15 -7 -13
b: 2 -1 6	m: 8 -3 6	r: 12 -10 17	w: 16 -9 -11
c: -1 2 4	n: 7 -12 -15	s: -13 -12 15	x: 17 -5 -16
d: -4 1 14	o: 9 11 14	t: 14 -4 16	y: -3 -7 13
e: -5 4 2	p: -10 -2 17	u: -9 14 16	z: 0 6 -14

86

What about Recombination?

a: 1 -0 6	l: -6 10 13	q: -11 16 17	v: -15 -7 -13
b: 2 -1 6	m: 8 -3 6	r: 12 -10 17	w: 16 -9 -11
c: -1 2 4	n: 7 -12 -15	s: -13 -12 15	x: 17 -5 -16
d: -4 1 14	o: 9 11 14	t: 14 -4 16	y: -3 -7 13
e: -5 4 2	p: -10 -2 17	u: -9 14 16	z: 0 6 -14

We could consider a MAX-3SAT problem

87

What about Recombination?

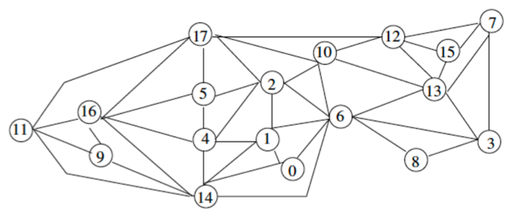
$f_a(1,0,6)$	$f_l(6,10,13)$	$f_q(11,16,17)$	$f_v(15,7,13)$
$f_b(2,1,6)$	$f_m(8,3,6)$	$f_r(12,10,17)$	$f_w(16,9,11)$
$f_c(1,2,4)$	$f_n(7,12,15)$	$f_s(13,12,15)$	$f_x(17,5,16)$
$f_d(4,1,14)$	$f_o(9,11,14)$	$f_t(14,4,16)$	$f_y(3,7,13)$
$f_e(5,4,2)$	$f_p(10,2,17)$	$f_u(9,14,16)$	$f_z(0,6,14)$

We could consider an NK-Landscape or MK Landscape

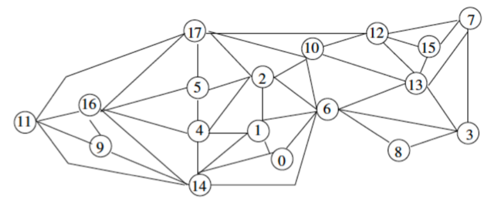
The variables interactions are the same.

88

The Variable Interaction Graph

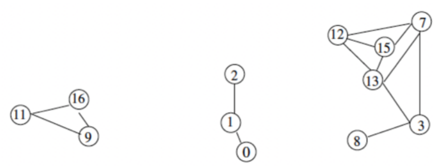


The Variable Interaction Graph



LOCAL OPTIMUM P1: 000000000000000000
LOCAL OPTIMUM P2: 111100011101110110

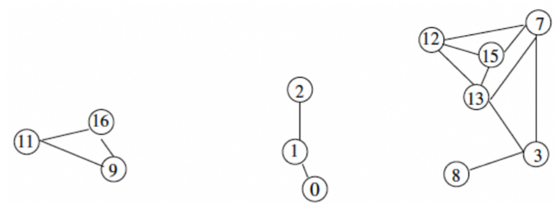
The Recombination Graph: The Decomposed VIG.



When recombining the solutions $S_{P1} = 000000000000000000$ and $S_{P2} = 111100011101110110$, the vertices and edges associated with shared variables 4, 5, 6, 10, 14, 17 are deleted to yield the **recombination graph**.

Thanks to Renato Tinos

The Recombination Graph: The Decomposed VIG.

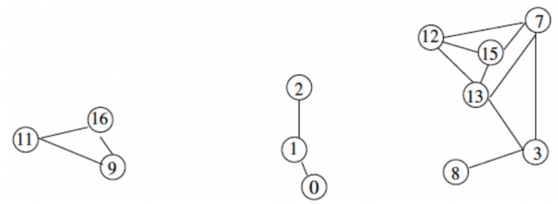


q, o, u, w, x, t a, b, c, d, e, p, z l, m, n, r, s, v, y

This decomposition partitions the variables **and** the subfunctions.

Partition Crossover deterministically returns the *best* of 2^q offspring.

The Recombination Graph:
Be GREEDY



q, o, u, w, x, t   a, b, c, d, e, p, z   l, m, n, r, s, v, y

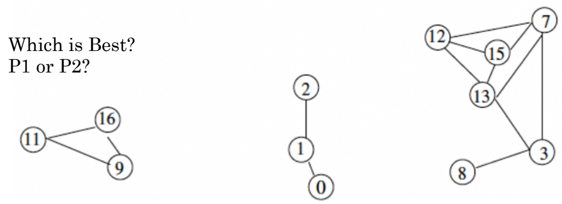
This decomposition partitions the variables **and** the subfunctions.

Partition Crossover deterministically returns the *best* of 2^q offspring.

93

The Recombination Graph:
Be GREEDY

Which is Best?
P1 or P2?



q, o, u, w, x, t   a, b, c, d, e, p, z   l, m, n, r, s, v, y

This decomposition partitions the variables **and** the subfunctions.

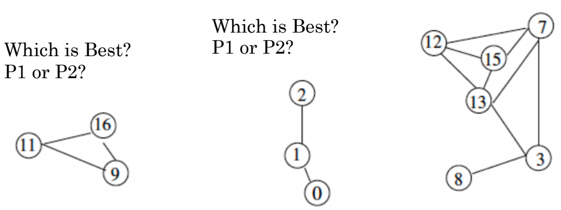
Partition Crossover deterministically returns the *best* of 2^q offspring.

94

The Recombination Graph:
Be GREEDY

Which is Best?
P1 or P2?

Which is Best?
P1 or P2?



q, o, u, w, x, t   a, b, c, d, e, p, z   l, m, n, r, s, v, y

This decomposition partitions the variables **and** the subfunctions.

Partition Crossover deterministically returns the *best* of 2^q offspring.

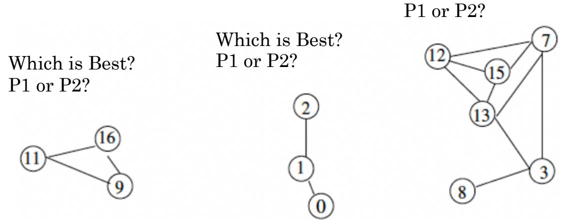
95

The Recombination Graph:
Be GREEDY

Which is Best?
P1 or P2?

Which is Best?
P1 or P2?

Which is Best?
P1 or P2?



q, o, u, w, x, t   a, b, c, d, e, p, z   l, m, n, r, s, v, y

Partition Crossover deterministically returns the *best* of 2^q offspring.

96

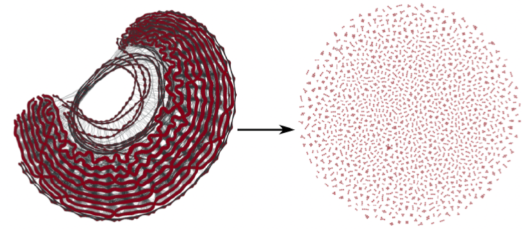
Partition Crossover and Local Optima

The Subspace Optimality Theorem: For any k-bounded pseudo-Boolean function f, if Parition Crossover is used to recombine two parent solutions that are locally optimal, then the offspring must be a local optima in the hyperplane subspace defined by the bits shared in common by the two parents.

Example: if the parents 0000000000 and 1100011101
are locally optimal, then the best offspring
is locally optimal in the hyperplane subspace **000***0*.

97

Decomposed Evaluation for MAXSAT



atco_enc3_opt1_13_48

Air traffic controller shift scheduling problem: 1087 components.

PX returns the best of 2^{1087} offsprings. $N = 1{,}067{,}657$

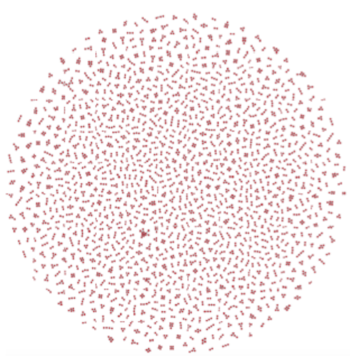
(Thanks to Wenxiang Chen)

98

Decomposed Evaluation for MAXSAT

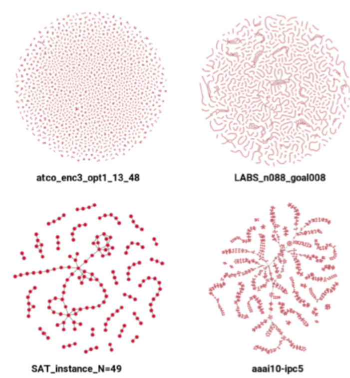
Crossover returns the Best of 2^{1087} offspring.

All offspring are Local Optima in this subspace.



99

"Optimal Recombination by Dynamic Programming"



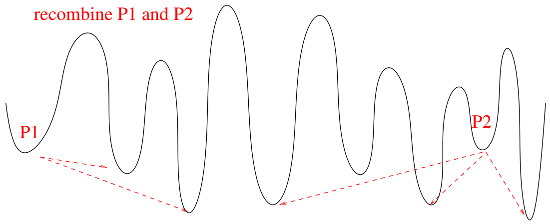
These subproblems have a tree decomposition with low width.

Thanks to Francisco Chicano.

These subproblems can be solved by Dynamic Programming!

100

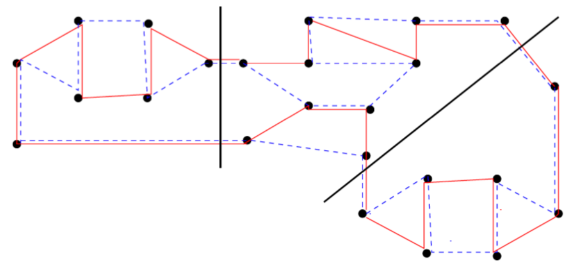
Partition Crossover *Deterministically* "Tunnels" Between Optima



We can remove randomness from Crossover

101

Partition Crossover and TSP



102

PART 2: Transforms

103

Transforms

- SAT to MAXSAT
  - For decades, SAT problems have been converted into MAX-kSAT instances. Modern SAT solvers expect a MAXSAT form.
  - TRANSFORMS may also serve as REDUCTIONS used to prove NP-Completeness.

104

Transforms

- Transforms exist for all Pseudo-Boolean Functions

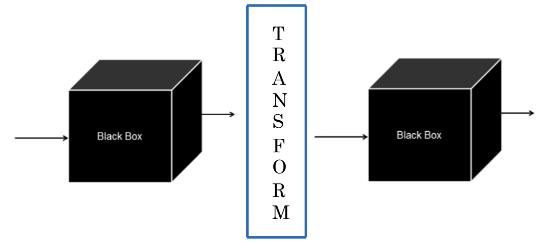
"All pseudo-Boolean optimization problems can be *reduced* to the quadratic case." Boros and Hammer (2002):186

This assumes a polynomial evaluation function.

The transformed function is polynomial in size relative to the original function.

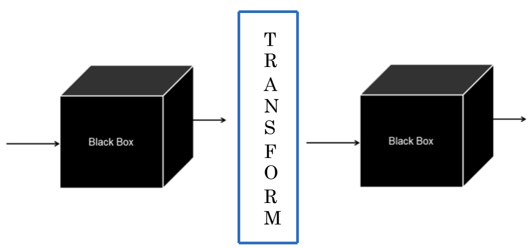
105

Transforms can be Quasi-Black Box (if you Insist).



106

Transforms can be Quasi-Black Box (But Not Really).



The quadratic function is recovered by sampling in $O(n^2)$ time.

107

The Multilinear form

$w_0 = f(000)$
$w_1 = f(001) - w_0$
$w_2 = f(010) - w_0$
$w_3 = f(011) - w_0 - w_1 - w_2$
$w_4 = f(100) - w_0$
$w_5 = f(101) - w_0 - w_4 - w_1$
$w_6 = f(110) - w_0 - w_4 - w_2$

$f(x) = w_0 + w_1x_1 + w_2x_2 + w_4x_4 + w_3x_1x_2 + w_5x_1x_4 + w_6x_2x_4$

108

Transforms

- Assume our function is written in Multilinear Form (helpful, but not necessary)

$$f(x_1, \ldots, x_n) = \sum_{S \subseteq V} c_S \prod_{j \in S} x_j$$

Where
x_i is a Boolean Variable, c_S is a weight (constant)
S is a single variable
or a subset of variables with nonlinear interactions
And j indexes a variable in S.

109

Leading Ones as a Multilinear Form

- The n=4 case.

$$f(x_1, x_2, x_3, x_4) = x_1 + x_1x_2 + x_1x_2x_3 + x_1x_2x_3x_4$$

- A general example

f(0111111111111111111) = 0

f(1001100110001111111) = 1

f(1111100001100011111) = 5

110

Leading Ones as a Multilinear Form

- The n=4 case.

$$f(x_1, x_2, x_3, x_4) = x_1 + x_1x_2 + x_1x_2x_3 + x_1x_2x_3x_4$$

- The Walsh Polynomial is EXPONENTIAL in Size!

111

Leading Ones as a Multilinear Form

- The n=4 case.

$$f(x_1, x_2, x_3, x_4) = x_1 + x_1x_2 + x_1x_2x_3 + x_1x_2x_3x_4$$

- In general for maximization.

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j$$

- In general for minimization.

$$f(x_1, \ldots, x_n) = n + \sum_{i=1}^{n} -1 \prod_{j=1}^{i} x_j$$

112

The Basic Substitution:
1 variable replaces 2 variables

$$x_1x_2 = z \text{ iff } x_1x_2 - 2x_1z - 2x_2z + 3z = 0$$

$$x_1x_2 \neq z \text{ iff } x_1x_2 - 2x_1z - 2x_2z + 3z > 0$$

Substitution with Penalty Constraints

$$P(x_1x_2 - 2x_1z - 2x_2z + 3z)$$

P is the sum of |Cs|

113

For Leading Ones n=4, p=5

$$f(x_1, x_2, x_3, x_4) = 4 - x_1 - x_1x_2 - x_1x_2x_3 - x_1x_2x_3x_4$$

$$f(x_1, x_2, x_3, x_4, z_1) = 4 - x_1 - x_1x_2 - z_1x_3 - z_1x_3x_4 + 5x_1x_2 - 10x_1z_1 - 10x_2z_1 + 15z_1$$

$$f(x_1, x_2, x_3, x_4, z_1, z_2) = 4 - x_1 - x_1x_2 - z_1x_3 - z_2x_4 + 5x_1x_2 - 10x_1z_1 - 10x_2z_1 + 15z_1 + 5z_1x_3 - 10z_1z_2 - 10x_3z_2 + 15z_2$$

114

Leading Ones in General:

$$f(x, z) = n - x_1 - z_{n-2}x_n + \sum_{i=1}^{n-2} f_i(z_i, z_{i-1}, x_{i+1})$$

$$f_i(z_i, z_{i-1}, x_{i+1}) = -z_{i-1}x_{i+1} + P(z_{i-1}x_{i+1} - 2z_iz_{i-1} - 2z_ix_{i+1} + 3z_i)$$

The Walsh Polynomial is LINEAR in size.

115

Leading Ones in General:

$$f(x, z) = n - x_1 - z_{n-2}x_n + \sum_{i=1}^{n-2} f_i(z_i, z_{i-1}, x_{i+1})$$

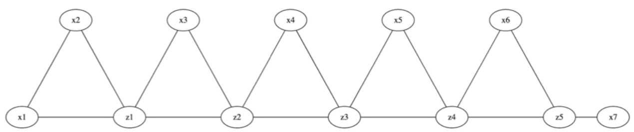
$$f_i(z_i, z_{i-1}, x_{i+1}) = -z_{i-1}x_{i+1} + P(z_{i-1}x_{i+1} - 2z_iz_{i-1} - 2z_ix_{i+1} + 3z_i)$$

One can optimize each subfunction z_i=1, z_{i-1}=1, x_{i+1} =1 to obtain the global optimum in O(n) time.

You can use **Local Search with Lookahead** to solve the Quadratized Leading Ones in O(n) time.

116

Leading Ones: The New Variable Interaction Graph



RESULT:

Quadratic Leading Ones has a bounded tree width and is solved by **Dynamic Programming** in O(n) time.

117

Leading Ones: The New Variable Interaction Graph

The same **Variable Interaction Graph** holds for all **Hidden Permutations** versions of Leading Ones.

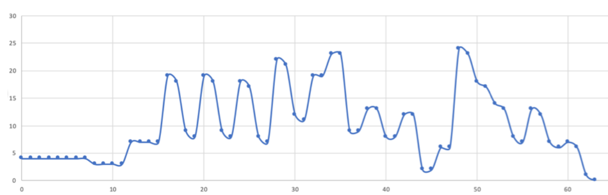
x(i) is just replaced with x(π(i))

The same **Variable Interaction Graph** holds for all **Hidden Permutations** versions of Leading Ones.

x(i) is just replaced with x(π(i))

118

For Leading Ones N=4



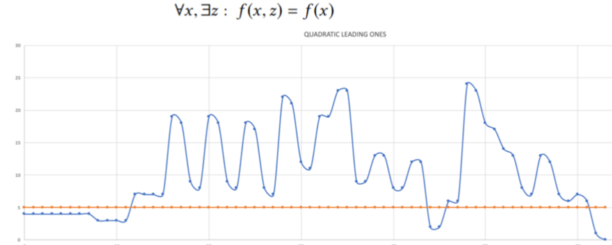
Every penalized point in f(x,z) is greater that every point in f(x).

119

General Result

THEOREM 1:. *Let $f(x)$ be a multilinear pseudo-Boolean function, and let $f(x,z)$ be a k-bounded pseudo-Boolean function produced by replacing variables in $f(x)$ with auxiliary variables in vector z. Index the variables in z and let z^j denote the first j variables in z. If z_{j+1} only replaces variables in x and z^j then:*

$$\forall x, \exists z : \ f(x,z) = f(x)$$



120

A More Recent Transform:

x1 + x1x2 + x1x2x3 + x1x2x3x4 + x1x2x3x4x5

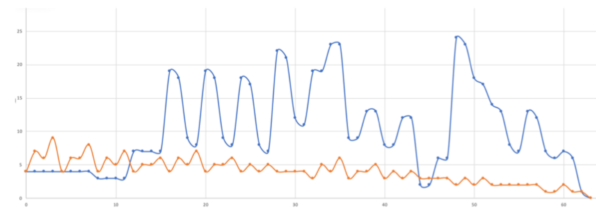
Transforms to:

x1 + x1x2
x1y1 + x2y1 + x3y1 -2y1
x1y2 + x2y2 + x3y2 + x4y2 -3y2
x1y3 + x2y3 + x3y3 + x4y3 + x5y3 -4y3

This transform can reveal submodular functions:
Leading Ones and Hidden Permutations are **submodular**

121

A More Recent Transform:

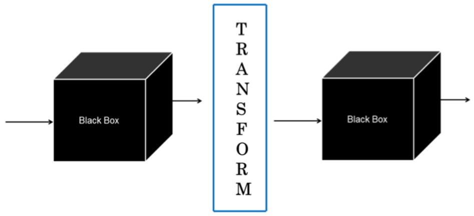


This transform can reveal submodular functions:
Leading Ones and Hidden Permutations are submodular

122

Transforms For Needle-In-A-Stack

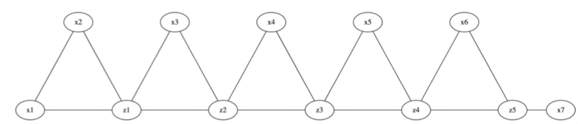
Needle(x): return 100 if x = ???????????????????????????????????????
else return 0



123

Transforms For Needle-In-A-Stack

Needle(x): return 100 if x = 1100010001001110101010011010110
else return 0



Using the same Transform, it has the same Variable Interaction Graph as LeadingOnes: it is now trivial to solve.

124

THERE ARE NO QUADRATIC NEEDLE-IN-A-STACK PROBLEMS!

Needle(x): return 100 if x = 110001000100111010101001101011 0
else return 0

We can NEVER hide a needle in a quadratic form, because we can construct the multilinear form of the quadratic in O(n^2) time.

125

QUADRATIC PSEUDO BOOLEAN FUNCTIONS IN QUANTUM COMPUTING

Nike Dattani, ArXiv, September 2019

"Quadratization in Discrete Optimization and Quantum Mechanics"

Lists 40 new transforms, many published in the last 5 years.

Dattani et al. hold the record for quantum factoring of Semi Prime Numbers used in Encryption.

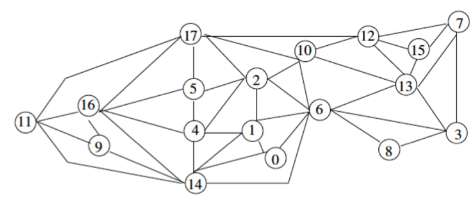
126

WHAT ABOUT LOOKING 2 OR 3 OR 10 MOVES AHEAD?

With Thanks to Francisco Chicano

127

The Variable Interaction Graph



If two variables are not connected in the VIG, there can be no improving move.

128

What about looking 2 or more moves ahead?

$$f(x) = \psi_x w_1 + \psi_x w_2 + \psi_x w_3 + \psi_x w_4 + \psi_x w_5 + \psi_x w_6 + \psi_x w_7 + \psi_x w_8 + \psi_x w_{1,2} + \psi_x w_{2,3} + \psi_x w_{3,4} + \psi_x w_{1,4} + \psi_x w_{3,5} + \psi_x w_{5,6} + \psi_x w_{6,7} + \psi_x w_{5,7} + \psi_x w_{7,8} + \psi_x w_{8,4} + \psi_x w_{5,6,7} + \psi_x w_{4,7,8}$$

Assume you have taken all single bit flip improving moves.

What happens when you flip bits 5 and 8 at the same time?

129

What about looking 2 or more moves ahead?

$$f(x) = \psi_x w_1 + \psi_x w_2 + \psi_x w_3 + \psi_x w_4 + \psi_x w_5 + \psi_x w_6 + \psi_x w_7 + \psi_x w_8 + \psi_x w_{1,2} + \psi_x w_{2,3} + \psi_x w_{3,4} + \psi_x w_{1,4} + \psi_x w_{3,5} + \psi_x w_{5,6} + \psi_x w_{6,7} + \psi_x w_{5,7} + \psi_x w_{7,8} + \psi_x w_{8,4} + \psi_x w_{5,6,7} + \psi_x w_{4,7,8}$$

Assume you have taken all single bit flip improving moves.

What happens when you flip bits 5 and 8 at the same time?

NOTHING. There are no nonlinear coefficients involving 5 and 8.

130

What about looking 2 or more moves ahead?

$f_a(1,0,6)$	$f_l(6,10,13)$	$f_q(11,16,17)$	$f_v(15,7,13)$
$f_b(2,1,6)$	$f_m(8,3,6)$	$f_r(12,10,17)$	$f_w(16,9,11)$
$f_c(1,2,4)$	$f_n(7,12,15)$	$f_s(13,12,15)$	$f_x(17,5,16)$
$f_d(4,1,14)$	$f_o(9,11,14)$	$f_t(14,4,16)$	$f_y(3,7,13)$
$f_e(5,4,2)$	$f_p(10,2,17)$	$f_u(9,14,16)$	$f_z(0,6,14)$

131

What about looking 2 or more moves ahead?

$f_a(1,0,6)$	$f_l(6,10,13)$	$f_q(11,16,17)$	$f_v(15,7,13)$
$f_b(2,1,6)$	$f_m(8,3,6)$	$f_r(12,10,17)$	$f_w(16,9,11)$
$f_c(1,2,4)$	$f_n(7,12,15)$	$f_s(13,12,15)$	$f_x(17,5,16)$
$f_d(4,1,14)$	$f_o(9,11,14)$	$f_t(14,4,16)$	$f_y(3,7,13)$
$f_e(5,4,2)$	$f_p(10,2,17)$	$f_u(9,14,16)$	$f_z(0,6,14)$

0,6	0,14	1,0	1,2	1,4	1,6	1,14	2,4
2,5	2,6	2,10	2,17	3,6	3,7	3,8	3,13
4,5	4,14	4,16	5,16	5,17	6,8	6,10	6,13
6,14	7,12	7,13	7,15	9,11	9,14	9,16	10,12
10,13	10,17	11,14	11,16	11,17	12,13	12,15	12,17
13,15	14,16	16,17					

132

What about looking 2 or more moves ahead?

If the number of subfunctions is linear,
the number of pairs is linear.

0,6	0,14	1,0	1,2	1,4	1,6	1,14	2,4
2,5	2,6	2,10	2,17	3,6	3,7	3,8	3,13
4,5	4,14	4,16	5,16	5,17	6,8	6,10	6,13
6,14	7,12	7,13	7,15	9,11	9,14	9,16	10,12
10,13	10,17	11,14	11,16	11,17	12,13	12,15	12,17
13,15	14,16	16,17					

133

For K-bounded Pseudo Boolean Functions:

Random Mutation is Obsolete

We can compute the location of improving moves
in constant time, multiple steps ahead.

For problems like the TSP and Graph Coloring
(and many others)

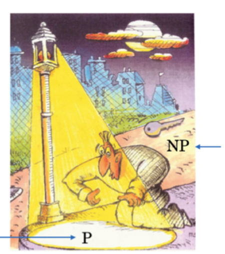
Random Mutation is useless.

134

The Trouble with "Runtime"

1. If P ≠ NP, you cannot achieve a polynomial time bound unless the function is in the class P.

"Searching under the Streetlight"



Unrealistic Assumptions
Lead to Meaningless Results,
No matter how complex the Math.

135

Good Theory must also be Good Science

"Algorithm A1 runs in $O(n^2)$ time on Problem P2."

By itself, this is meaningless counting.

It is not a *theory* of anything.

136

Good Theory must also be Good Science

"Algorithm A1 runs in $O(n^2)$ time on Problem P2."

By itself, this is meaningless counting.

It is not a *theory* of anything.

Remember No Free Lunch? (Sharpened and Focused)

We would never accept this in empirical research,
so why is is OK in theoretical work?

137

GIGA: Gene Invariant Genetic Algorithm

Crossover versus Mutation: Fueling the Debate: TGA versus **GIGA**.
JC **Culberson** - ICGA, 1993 - webdocs.cs.ualberta.ca
The goal of this paper is to entice researchers in genetic algorithms to consider a paradigm shift in the design and analysis of GAs. This shift is analogous to a shift from a focus on evolution as a mechanism that provides incredibly ne-tuned adaptations to particular ...
Cited by 30 Related articles All 2 versions

Genetic invariance: A new paradigm for genetic algorithm design
J **Culberson** - 1992 - era.library.ualberta.ca
This paper presents some experimental results and analyses of the gene invariant genetic algorithm (GIGA). Although a subclass of the class of genetic algorithms, this algorithm and its variations represent a unique approach with many interesting results. The primary ...
Cited by 44 Related articles All 6 versions

138

GIGA: Gene Invariant Genetic Algorithm

Joe Culberson, 1992, 2993, 1995

All individuals have an equal chance to reproduce.

There is NO selection in the traditional sense.

Two parents produce 2 offspring, so that no "genes" are lost.

Mating is based on matching individuals with similar fitness.

NO BITS (Alleles) ARE EVER LOST.

It is impossible to lose diversity.

139

GIGA: Gene Invariant Genetic Algorithm

000000000111111111
111111111000000000

My Own Simple Twist:

Sample a string,
Sample its complement.

GIGA: Does not uses Mutation.

140

GIGA: Gene Invariant Genetic Algorithm

Generation 1	Generation 2
111111111000000000	111111111001111111
000000000111111111	000000000110000000
111100000111110000	111100011000001111
000011111000001111	000011100111110000
010111010100110100	010111010100110011
101000101011001011	101000101011001100

141

GIGA: Gene Invariant Genetic Algorithm

111111111001111111

111100011000001111

010111010100110011

000011100111110000

101000101011001100

000000000110000000

SELECTION

Better Solutions move UP

Poorer Solutions down DOWN

142

GIGA: Gene Invariant Genetic Algorithm

111111111001111111

111100011000001111

010111010100110011

000011100111110000

101000101011001100

000000000110000000

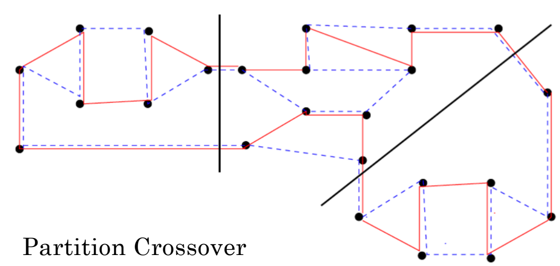
SELECTION

Recombine Strings with Similar Fitness.

So recombine adjacent neighbors.

143

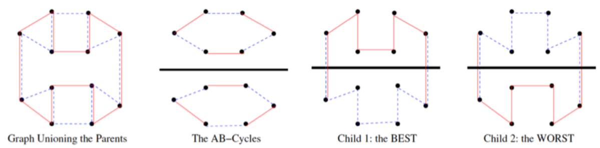
BACK TO THE TSP



Partition Crossover deterministically returns the *best* of 2^q offspring.

144

Create Two Offspring:

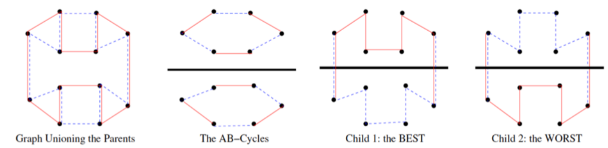


$$eval(P1) + eval(P2) = eval(C_{best}) + eval(C_{worst})$$

If you keep the best and worst child,
the population average never changes.

145

Create Two Offspring: Keep BOTH

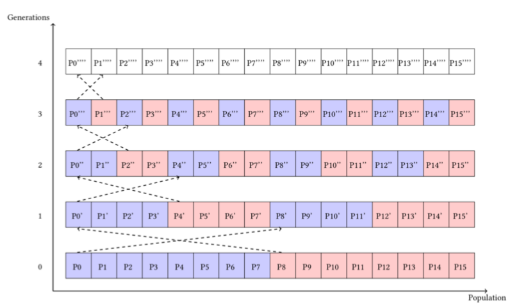


$$eval(P1) + eval(P2) = eval(C_{best}) + eval(C_{worst})$$

If you keep the best and worst child,
the population average never changes.

146

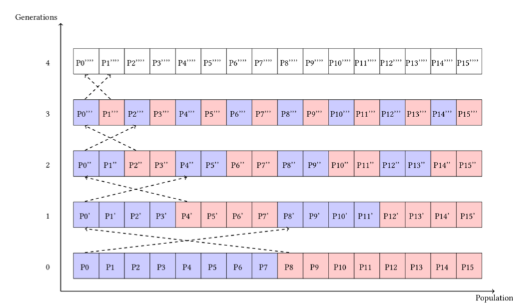
The Mixing GA
(An Updated Form of GIGA.)



Thanks to Swetha Varadarajan!

147

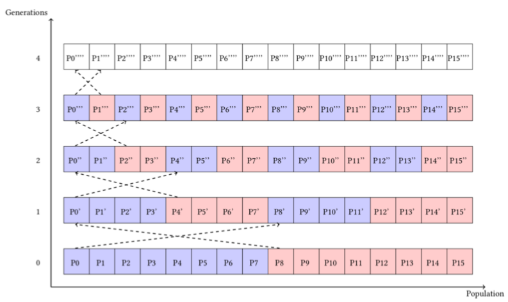
THERE IS NO SELECTION!



Thanks to Swetha Varadarajan!

148

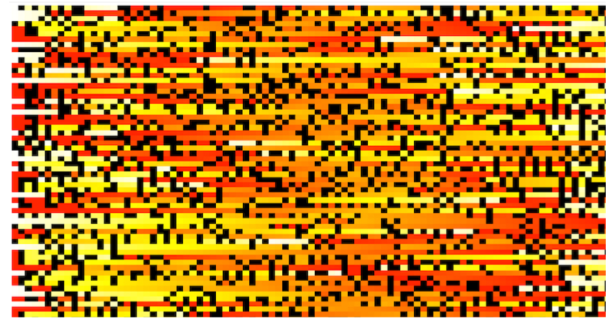
THERE IS NO SELECTION!
EXCEPT MATING SELECTION.



Thanks to Swetha Varadarajan!

149

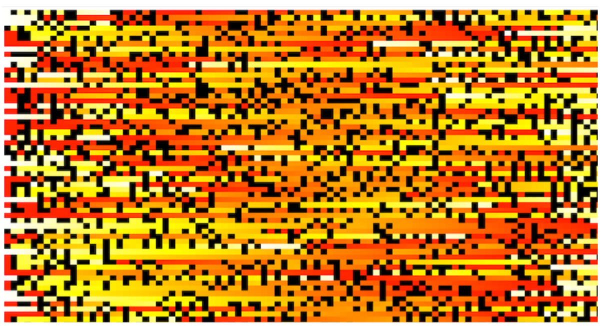
First Population: ATT532



EACH ROW IS A LOCALLY OPTIMAL TOUR.
EACH COLOR IS A EDGE.

150

First Population: ATT532

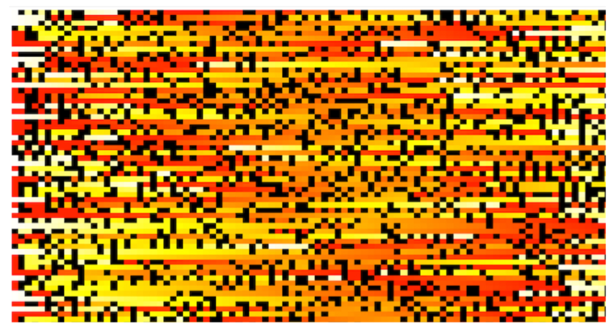


EACH ROW IS A TOUR.
EACH COLOR IS A EDGE.

Thanks to Swetha Vardarajan
and Gabriela Ochoa

151

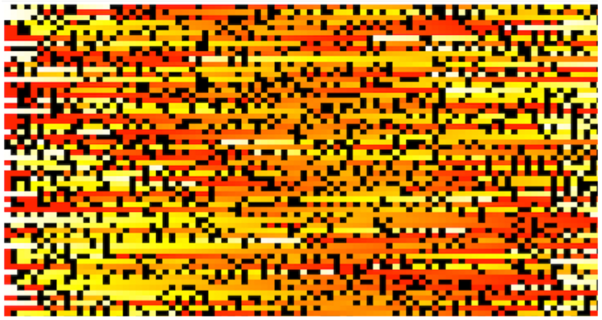
First Population: ATT532



BLACK edges are NOT part of the global optimum.

152

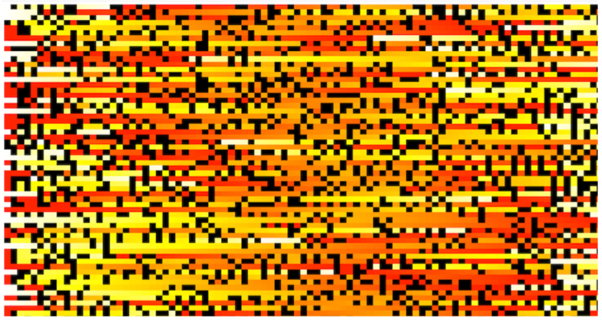
First Population: ATT532



COLOR edges are edges also found in the global optimum.

153

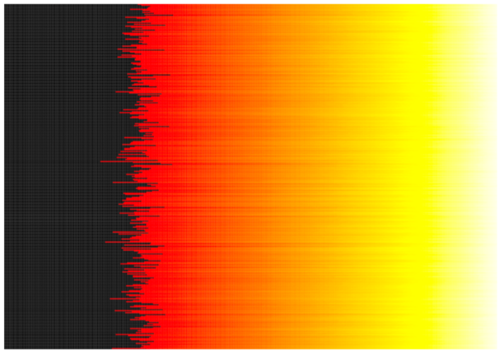
First Population: ATT532



Local Optima are 70% composed of globally optimal edges!!

154

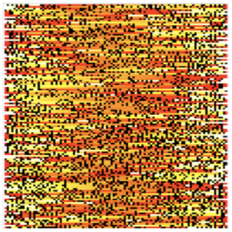
First Population: Same Data, Sorted.

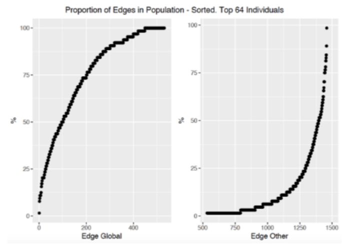


Local Optima are 70% composed of globally optimal edges!!

155

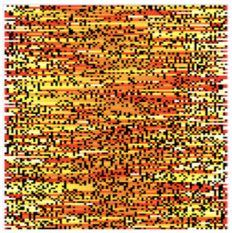
First Population: ATT532

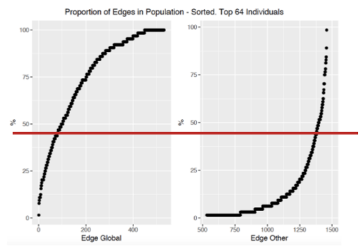




156

First Population: ATT532

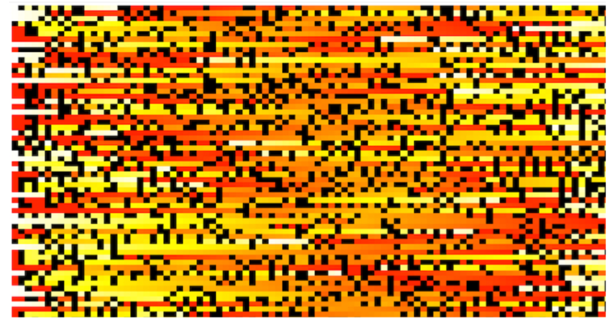




If we select the 532 most frequent edges, about 85% are global.

157

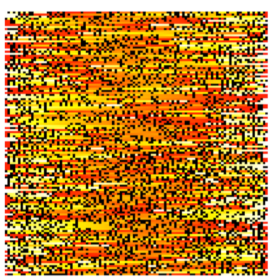
First Population: ATT532

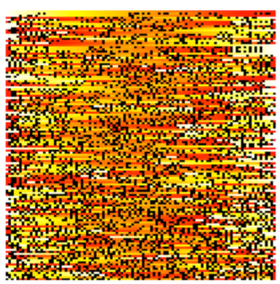


CONVERGENCE:

158

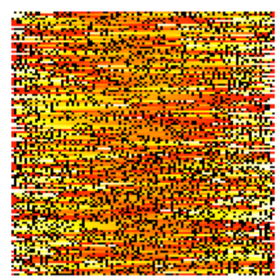
First Population: Last Population:

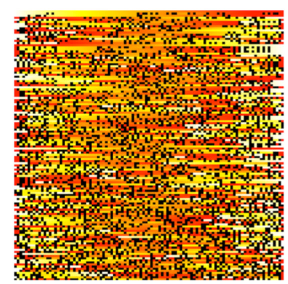




159

First Population: Last Population:





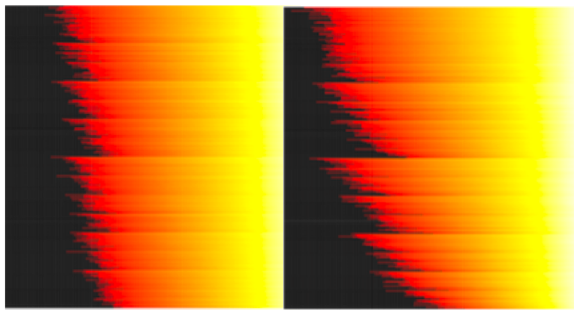
Good Tours Go UP

Bad Tours Go DOWN

In this case, the edges in the population NEVER CHANGED.

160

First Generation: Last Generation:



Good Tours Go UP

Bad Tours Go DOWN

In this case, the edges in the population NEVER CHANGED.

EAX Genetic Algorithm for the TSP

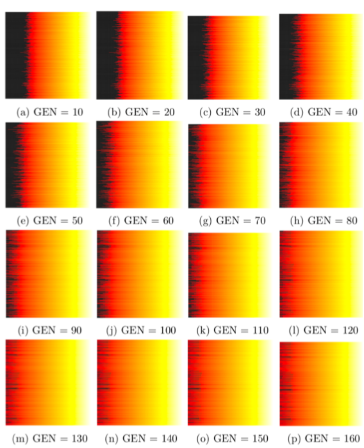
Nagata and Kobayashi (2012)
A Powerful Genetic Algorithm Using Edge Assembly Crossover for the Traveling Salesman Problem
INFORMS JOC

Probably the best off the shelf Inexact Solver for the TSP.

Highlights the power of crossover.

The only selection is Brood Selection.

EAX: Edge Assembly Crossover.



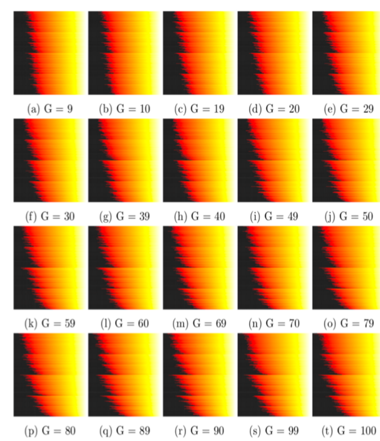
Convergence of EAX.

It converges to the Global Optimum
IF
It does not prematurely converge first.

EAX *usually* drives out non-global edges.

THANKS TO Swetha Varadarajan

A Hybrid of EAX, Partition crossover and the Mixing GA (a form of GIGA).

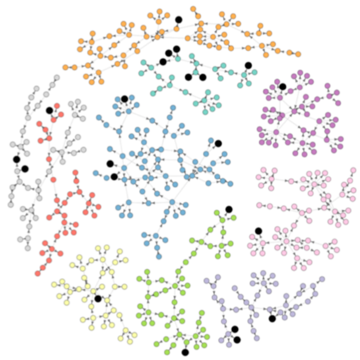


The algorithm can now converge while losing minima diversity.

THANKS TO Swetha Varadarajan

Tunneling Between Local Optima

Local Optima are "Linked" by Partition Crossover



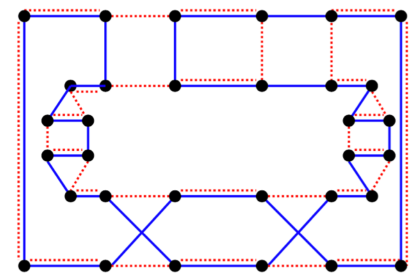
These were found using Chained-LK.

But it could have been Lin Kernighan Helsgaun (LKH).

Thanks to G. Ochoa and N. Veerapen.

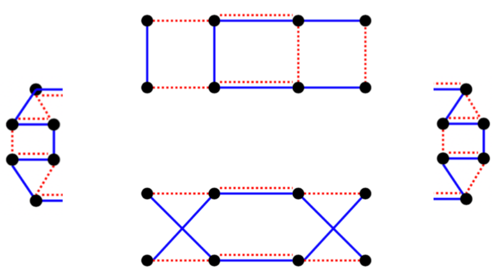
165

Partition Crossover and TSP



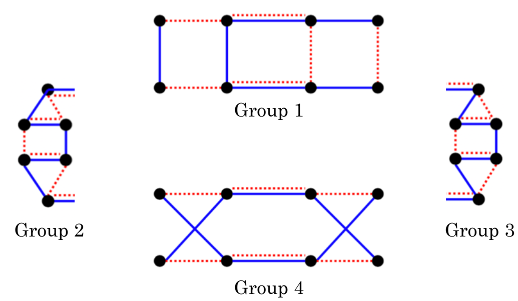
166

Partition Crossover and TSP



167

Partition Crossover and TSP



168

The quasi-local optima form a lattice in hyperspace:

Assume you have these connected groups of variables during recombination.

Group 1: v1, v2, v4, v5, v7, v9

Group 2: v11, v13, v14, v15, v17, v18

Group 3: v20, v21, v23, v26, v27, v28

Group 4: v32, v33, v34, v35, v36, v39

169

The quasi-local optima form a lattice in hyperspace:

Assume you have these connected groups of variables during recombination.

Group 1: v1, v2, v4, v5, v7, v9 Parent 1 or Parent 2?

Group 2: v11, v13, v14, v15, v17, v18 Parent 1 or Parent 2?

Group 3: v20, v21, v23, v26, v27, v28 Parent 1 or Parent 2?

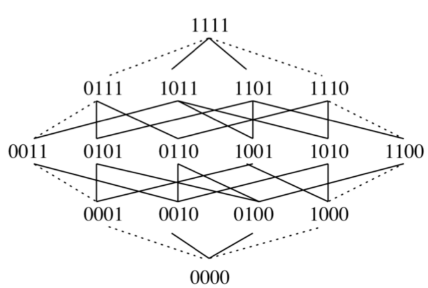
Group 4: v32, v33, v34, v35, v36, v39 Parent 1 or Parent 2?

Partition Crossover returns the best of 2^4 = 16 solutions.

170

The quasi-local optima form a lattice in hyperspace:

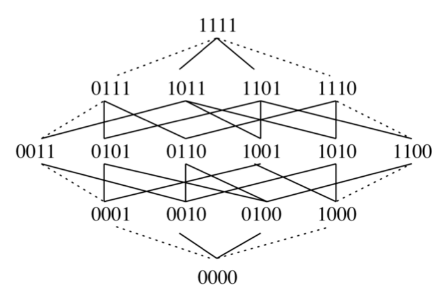
Group 1. Group 2. Group 3. Group 4.



171

The quasi-local optima form a lattice in hyperspace:

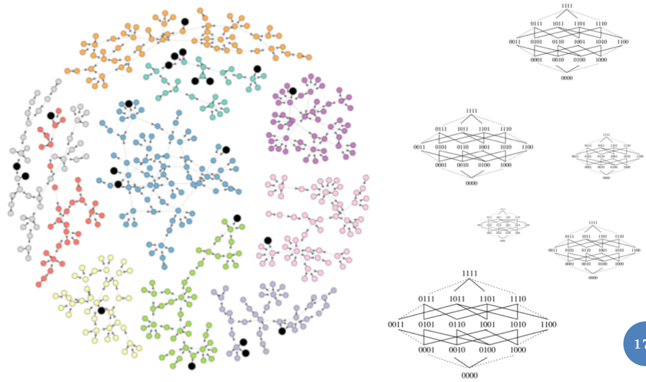
Group 1. Group 2. Group 3. Group 4.



ALL of these solutions are LOCAL OPTIMA In the Hyperplane Subspace.

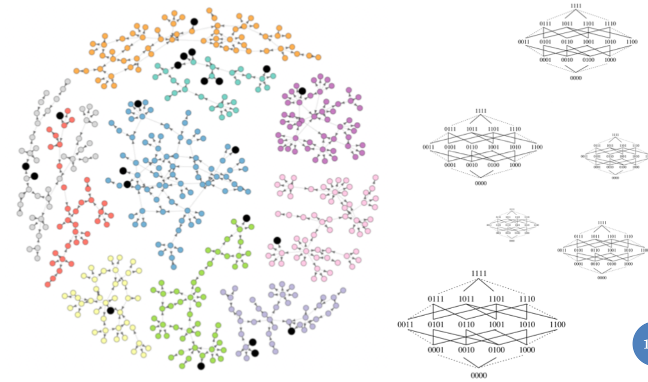
172

These tunnels are just the **TOPS** of lattices of quasi local optima.



173

The **BLACK** dots are Global Optima

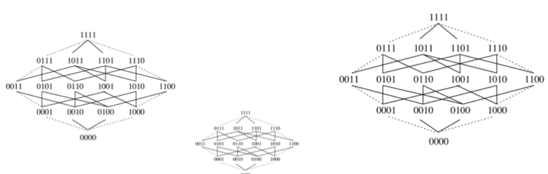


174

THEOREM: A lattice of quasi-local optima can be exponentially Large:

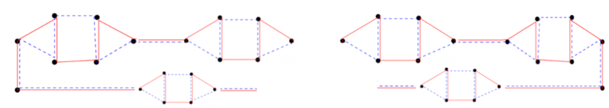
PROOF BY CONSTRUCTION: Construct a traveling salesman problem over N vertices such that it has two local optima, and these two local optima decompose into N/c recombining components for some constant c.

This results in a lattice of size $2^{N/c}$



175

THEOREM: A lattice of quasi-local optima can be exponentially Large:



The construction builds a chain of recombining components.

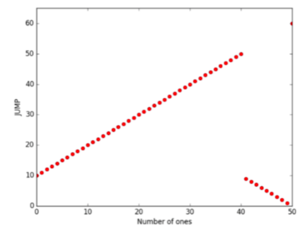
176

An ENSEMBLE OF SOLVERS

Instance	Sequential				Ensemble					
	LKH		EAX		LKH+EAX		MGA+EAX		Full Ensemble	
	S.R	Time	S.R	Time	S.R	Time	S.R	Time	S.R	Time
d2103	0/30	236	30/30	18	30/30	19	30/30	13	30/30	13
u2319	30/30	5	3/30	96	30/30	4	7/30	86	30/30	4
pr2392	30/30	4	30/30	37	30/30	3	30/30	23	30/30	3
pcb3038	30/30	47	30/30	110	30/30	46	30/30	57	30/30	46
fnl4461	30/30	92	30/30	273	30/30	90	30/30	210	30/30	91
rl5915	0/30	337	30/30	154	30/30	152	30/30	95	30/30	95
rl5934	30/30	80	11/30	161	30/30	79	30/30	94	30/30	79
pla7397	30/30	283	0/30	NA	30/30	283	5/30	241	30/30	241
rl11849	0/30	2600	30/30	853	30/30	849	30/30	556	30/30	556
usa13509	0/30	3844	17/30	2078	17/30	2223	17/30	1549	23/30	1549
brd14051	0/30	6367	30/30	4093	30/30	3693	30/30	1673	30/30	1673
d15112	30/30	5477	30/30	5936	30/30	4822	30/30	2646	30/30	2646
d18512	30/30	7705	30/30	5383	30/30	4716	30/30	3204	30/30	3204
pla33810	30/30	8334	30/30	5484	30/30	5695	30/30	3034	30/30	3034
pla85900	30/30	12129	30/30	31318	30/30	12099	30/30	19969	30/30	12099
AVERAGE	20	3169	24.06	3999	29.13	2318	25.93	2230	29.53	1688

177

THE JUMP FUNCTION.

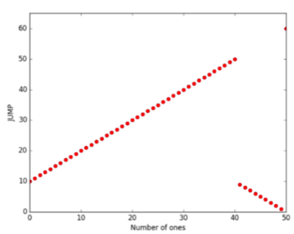


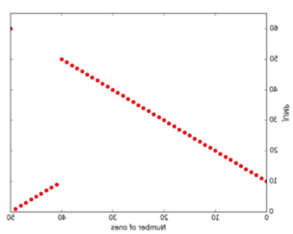
This is very nice example problem that can illustrate the power of crossover.

T. Jansen and I. Wegener (2002)
The Analysis of Evolutionary Algorithms: a proof that crossover really can help. *Algorithmic.* Springer.

178

THE JUMP FUNCTION.

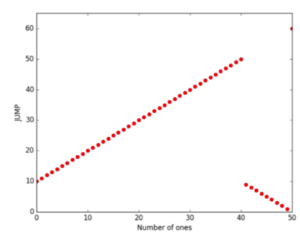


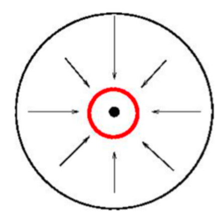


It is a symmetric Unitation function.
It is basically ONEMAX with a MOAT.

179

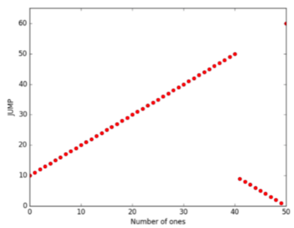
THE JUMP FUNCTION.





180

The JUMP Function.



Unintelligent
Random Local Search

(N choose k) 2^k

N is the number of bits

k is the size of the gap.

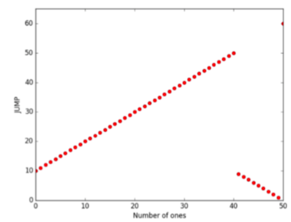
N = 128, k = lg(N) = 7

(N choose k) = 94,525,795,200 (94 Billion)

(N choose k) 2^k = 12,099,301,785,600 (12 Trillion)

181

The JUMP Function.



Doer's
"FAST GENETIC ALGORITHM"
(GECCO 2018).

It isn't a genetic algorithm,
It is really just the (1+1)ES.

We downloaded
the publicly available code.

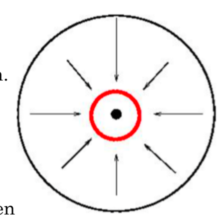
It ran for one week on a
N=1000, k = 10 problem
and didn't converge.

182

The JUMP Function.

How to really solve the JUMP Function.

Apply Systematic Local Search First.



The evaluation function can be rewritten
to evaluate in constant time.

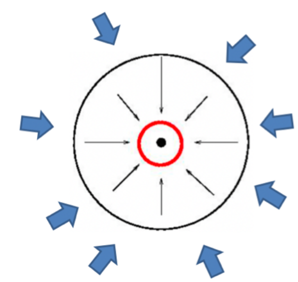
JUMP-FAST-EVAL(old-string, bit-to-change, old-eval) → Eval of new-string.

JUMP-FAST-EVAL executes in constant time.

183

The JUMP Function.

Apply Systematic Local Search First.

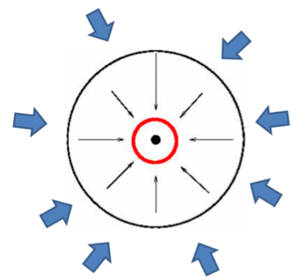


Population Size: 15

Local Search will converge to points
on the edge of the "Moat" in O(N) time.

184

The JUMP Function.



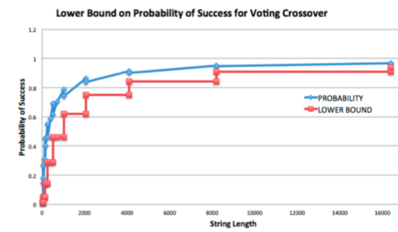
DETERMINISTIC VOTING CROSSOVER

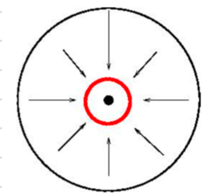
AKA: *Occurrence Based Scanning Crossover* (Eiben et al., PPSN 1994)

The child inherits the bit that most frequently appears in the parents.

Parent 1: 1111111111111111111111111111000000
Parent 2: 1111111110000001111111111111111111
Parent 3: 0001111111111111111111000111111111
Child: 1111111111111111111111111111111111

The JUMP Function.

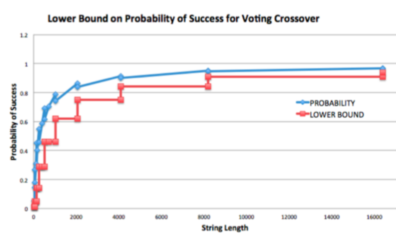


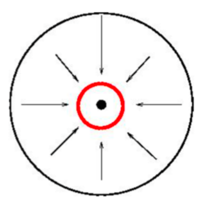


We proved that
Systematic Local Search converges
to the edge of the Moat in O(N) time.
Then crossover JUMPS the Moat in O(N) time.

Total runtime: O(N).

The JUMP Function.





Our simple Local Search + Crossover GA
solves N = 1,000,000 (one million) and k = 20 (log(N))
in a fraction of a second.

Now THAT is a "Fast Genetic Algorithm."

THEORY and PRACTICE.

Virtually NO THEORY in our
community asks what happens if you
apply local search first,
then apply crossover.

Most THEORY assumes
naïve unintelligent local search.

Many "theory benchmarks" are so trivial
they are easily solved,
even by unintelligent local search.

Questions?

189