Evolutionary Computation and Tunneling at the Edge of Quantum Computing

Darrell Whitley, GECCO 2023

GECCO '23 Companion, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0120-7/23/07□$15.00
https://doi.org/10.1145/3583133.3595063

1

1

Evolutionary Computation and Tunneling at the Edge of Quantum Computing

Darrell Whitley

Computer Science
Colorado State University

July 2023

2

2

Thanks To:

AFOSR, NSF, TRW, Coors, Texas Instruments, Amoco, Landmark, CIAI CASI, hp, Sandia National Laboratories, ITT, Raytheon Technologies, AGT, Texaco, Northrop Grumman, noranda

3

3

Thanks to my PhD Students!

- N. Karunanithi
- Tim Starkweather
- Scott Gordon
- Keith Mathias
- Raja Das
- Soraya Rana Stevens

- Robert Heckendorn
- Cesar Guerra
- Laura Barbulescu
- Jean Paul Watson
- Monte Lunacek
- Rinku Dewri

- Andrew Sutton
- Doug Hains
- Elmadhi Omar
- Wenxiang Chen
- Swetha Varadarajan
- Sachini Weerawardhana

4

4

SPECIAL THANKS TO:

- Francisco Chicano, Renato Tinos, Swetha Varadarajan, Wenxiang Chen, Andrew Sutton, Gabriela Ochoa.

5

5

WARNING:
THIS TALK IS NOT DIRECTLY ABOUT QUANTUM COMPUTING
(WELL, ONLY A LITTLE)

BUT IT IS ABOUT TUNNELING AND CONSTANT TIME MOVES FOR LOCAL SEARCH

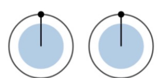
AND WHAT WE (SHOULD) KNOW ABOUT TRANSFORMS AND QUBO PROBLEMS (QUADRRATIC UNCONTRAINED BOOLEAN OPTIMIZATION)

6

6

A LITTLE QUANTUM COMPUTING

Superposed = $\begin{bmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{bmatrix}$ "Off" "On"



Superposition allows us probabilistically to be in two or more states at the same time.

But we must use quantum operators to move the probabilities toward the right answer.

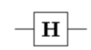
7

7

A LITTLE QUANTUM COMPUTING

(ONE MORE REASON FOR LEARNING ABOUT HADAMARD TRANSFORMS.)

Hadamard (H)



$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

Discrete (Square Wave) Fourier = Hadamard = Walsh Transform

8

8

The Hadamard Transform

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

9

MAX-3SAT

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

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

10

10

The Hadamard Transform: MAXSAT

$$\frac{1}{8}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}^T \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 0.875 \\ -0.125 \\ -0.125 \\ -0.125 \\ 0.125 \\ 0.125 \\ 0.125 \\ 0.125 \end{bmatrix}$$

The transform of a MAX-3SAT clause

11

The Hadamard Transform: MAXSAT

$$\frac{1}{8}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}^T \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 0.875 \\ -0.125 \\ -0.125 \\ -0.125 \\ 0.125 \\ 0.125 \\ 0.125 \\ 0.125 \end{bmatrix}$$

The transform of a MAX-3SAT clause

12

The Hadamard Transform: MAXSAT

$$\frac{1}{8}\begin{bmatrix}1\\1\\1\\1\\0\\1\\1\\1\end{bmatrix}^T\begin{bmatrix}1&1&1&1&1&1&1&1\\1&-1&1&-1&1&-1&1&-1\\1&1&-1&-1&1&1&-1&-1\\1&-1&-1&1&1&-1&-1&1\\1&1&1&1&-1&-1&-1&-1\\1&-1&1&-1&-1&1&-1&1\\1&1&-1&-1&-1&-1&1&1\\1&-1&-1&1&-1&1&1&-1\end{bmatrix}=\begin{bmatrix}0.875\\-0.125\\-0.125\\-0.125\\0.125\\0.125\\0.125\\0.125\end{bmatrix}$$

W W
W -W

13

The Hadamard Transform: MAXSAT

$$\frac{1}{8}\begin{bmatrix}1\\1\\1\\1\\0\\1\\1\\1\end{bmatrix}^T\begin{bmatrix}1&1&1&1&1&1&1&1\\1&-1&1&-1&1&-1&1&-1\\1&1&-1&-1&1&1&-1&-1\\1&-1&-1&1&1&-1&-1&1\\1&1&1&1&-1&-1&-1&-1\\1&-1&1&-1&-1&1&-1&1\\1&1&-1&-1&-1&-1&1&1\\1&-1&-1&1&-1&1&1&-1\end{bmatrix}=\begin{bmatrix}0.875\\-0.125\\-0.125\\-0.125\\0.125\\0.125\\0.125\\0.125\end{bmatrix}$$

W W W W
W -W W -W

W W -W -W
W -W -W W

14

A little QUANTUM COMPUTING

What Do Quantum Computers Have To Offer

QC changes the game:
They make use of physics with quantum behaviors

- Superposition
- Entanglement
- Wave Interference (constructive & destructive)

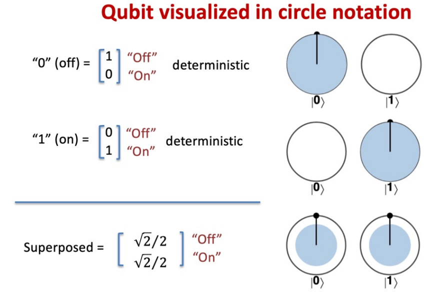
To create and manipulate QUBITS!!!

15

15

A little QUANTUM COMPUTING

Qubit visualized in circle notation

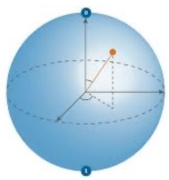


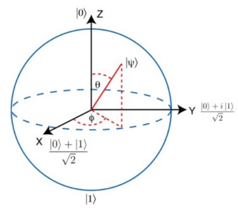
Qubits can be "On" or "Off" or "Part On, Part Off".

16

16

Qubit visualized as a BLOCH SPHERE





- Can be a deterministic 0 (top of sphere)
- Can be a deterministic 1 (bottom of sphere)
- Or anywhere in-between; a combination or "superposition" of 0 and 1
  - Example: 50% "0" and 50% "1" (point on equator of the sphere)
  - Example: 25% "0" and 75% "1" (red point on sphere)

17

17

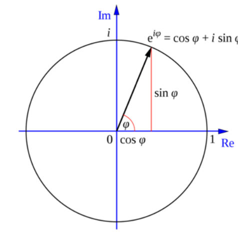
Euler's Formula

Euler's Formula

$$e^{i\phi} = \cos\phi + i\sin\phi$$

Euler's identity

$$e^{i\pi} + 1 = 0$$



18

18

- **Qubit State Vectors can contain *complex numbers* representing the quantum phase:**

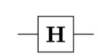
$$|q_0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{bmatrix} \qquad |q\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle$$

19

19

- **Where you see a "quantum gate", I see a "transformation matrix":**

q 0 $\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{-\sqrt{2}}{2} \end{bmatrix}$

Hadamard (H)  $\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

20

20

Quantum Operations

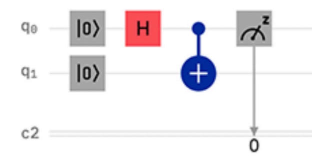
Operator	Gate(s)		Matrix
Pauli-X (X)	X	⊕	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)	Y		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)	Z		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)	H		$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)	S		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)	T		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)			$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)	Z		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP			$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)			$\begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0 \end{bmatrix}$

Hadamard (H) H $\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

21

21

Bell state + z-measurement on qubit 0



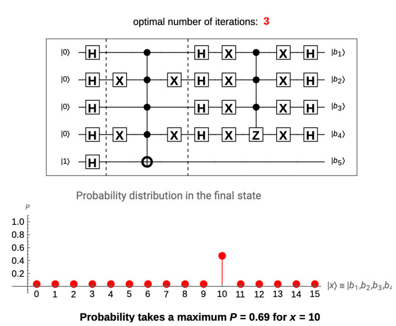
An entangled state of the two qubits can be created using a Hadamard H gate on the control qubit (q_0), followed by the CNOT toffoli gate.

22

22

Quantum Gates: Looks like quantum circuits
(written in something that looks a little like musical notation)

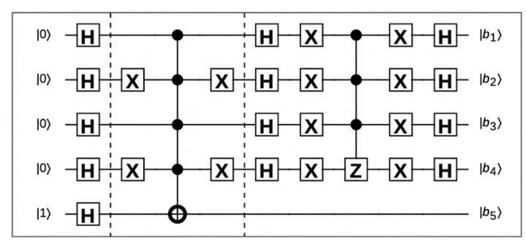
Quantum Gates: Currently works on a small number of qubits.



23

23

Pauli-X (X)	X	⊕	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)	Y		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)	Z		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)	H		$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

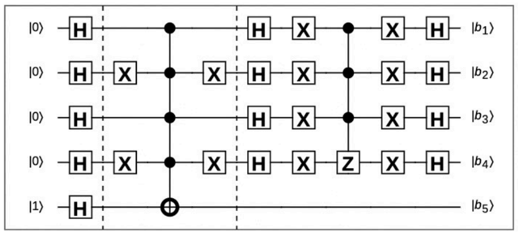


24

24

Often called "Grover's Algorithm"

Lov Grover



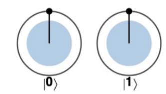
25

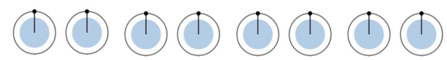
25

Quantum Annealing

Quantum annealing starts from a quantum superposition of all possible states with equal weights.

Superposed = $\begin{bmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{bmatrix}$ "Off" "On"

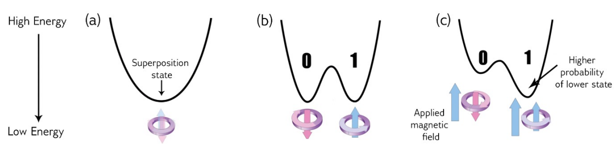




26

26

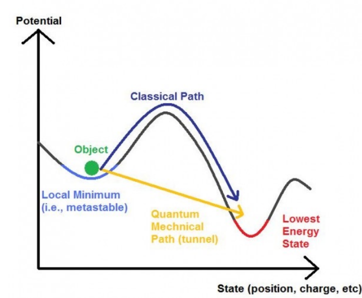
The system evolves following the time-dependent Schrödinger equation (a Wave Equation).



27

27

The amplitudes of all candidate states keep changing, realizing a quantum parallelism, which causes quantum tunneling between states.



28

28

FORGET BLACK BOX OPTIMIZATION !

(My 2 cents: it was always a silly idea for combinatorial problems.)

The name of the game is using information to bias the probability of reaching a global optimum.

You need insight to decide which bits can be conventional Boolean variables and which bits need to be quantum qubits.
Clever encodings (at least for now) minimize the number of qubits that are needed.

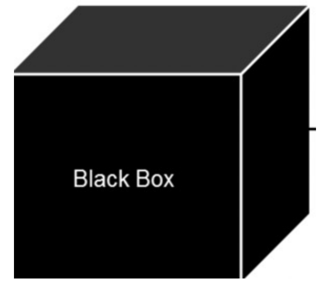
Many common quantum tools are ***impossible*** under black box assumptions.

How do you use entanglement in a Black Box scenario?

29

29

WHAT IS IN THE BOX?



30

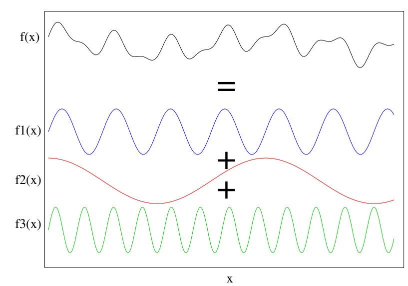
30

WHAT IS IN THE BOX?
WAVES OF COURSE!

31

31

WHAT IS IN THE BOX?
WAVES OF COURSE!



32

32

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)

33

33

The Wave Equation for Neighborhoods (Grover 1992)

Average value

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

34

34

The Wave Equation: A Simpler Explanation.

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

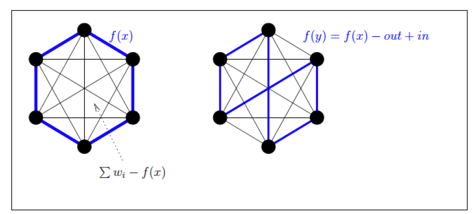
Starting from average...

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

35

35

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

36

36

FOR THE TRAVELING SALESMAN PROBLEM

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

37

37

K-BOUNDED PSEUDO-BOOLEAN FUNCTIONS: MAX-KSAT

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

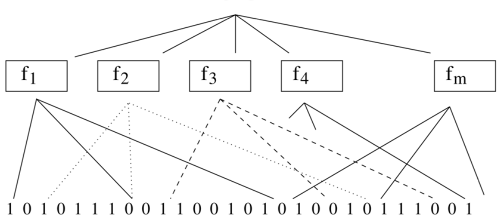
$$(\neg x_2 \vee x_1 \vee x_0) + (x_3 \vee \neg x_2 \vee x_1) + (x_3 \vee \neg x_1 \vee \neg x_0)$$

38

38

K-BOUNDED PSEUDO-BOOLEAN FUNCTIONS

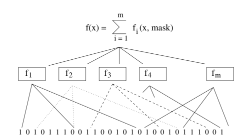
$$f(x) = \sum_{i=1}^{m} f_i(x, \text{mask})$$



39

39

K-BOUNDED PSEUDO-BOOLEAN FUNCTIONS



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

Which can be expressed as a Walsh Polynomial

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

Or can be expressed as a sum of k Elementary Landscapes

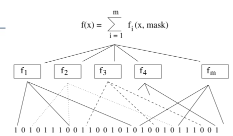
$$f(x) = \sum_{i=1}^{k} \varphi^{(i)}(W(f(x)))$$

40

40

Again, these are basically Wave Equations

The Eigenvectors of MAX-3SAT



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

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

$$\varphi^{(1)}(x) = f1_a(x) + f2_a(x) + f3_a(x) + f4_a(x)$$
$$\varphi^{(2)}(x) = f1_b(x) + f2_b(x) + f3_b(x) + f4_b(x)$$
$$\varphi^{(3)}(x) = f1_c(x) + f2_c(x) + f3_c(x) + f4_c(x)$$

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

41

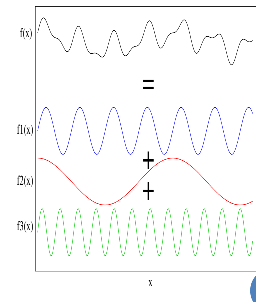
41

What is in the box? Waves of course!

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

$$\varphi^{(1)}(x) = f1_a(x) + f2_a(x) + f3_a(x) + f4_a(x)$$
$$\varphi^{(2)}(x) = f1_b(x) + f2_b(x) + f3_b(x) + f4_b(x)$$
$$\varphi^{(3)}(x) = f1_c(x) + f2_c(x) + f3_c(x) + f4_c(x)$$

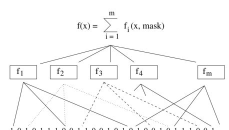
$$f(x) = a + \varphi^{(1)}(x) + \varphi^{(2)}(x) + \varphi^{(3)}(x)$$



42

42

All k-bounded Pseudo-Boolean Functions



And it is still a Wave Equation

Both $f(x)$ and $Avg(N(x))$ can be computed with Walsh Spans.

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

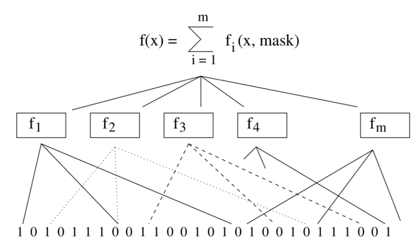
$$Avg(N(x)) = f(x) - 1/d \sum_{z=0}^{3} 2z\varphi^{(p)}(x)$$

$$Avg(N(x)) = \sum_{z=0}^{3} \varphi^{(z)}(x) - 2/N \sum_{z=0}^{3} z\varphi^{(z)}(x)$$

43

43

K-bounded Pseudo-Boolean Functions



RECENT CLAIMS (2022): We can use Quantum Computing to find which bit flips lead to an improving move in O(1) time.

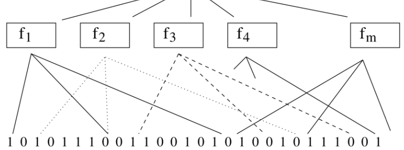
I heard this at a Quantum Workshop Keynote in July 2022 and at a Local Search Conference in September 2022.

44

44

K-BOUNDED PSEUDO-BOOLEAN FUNCTIONS

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



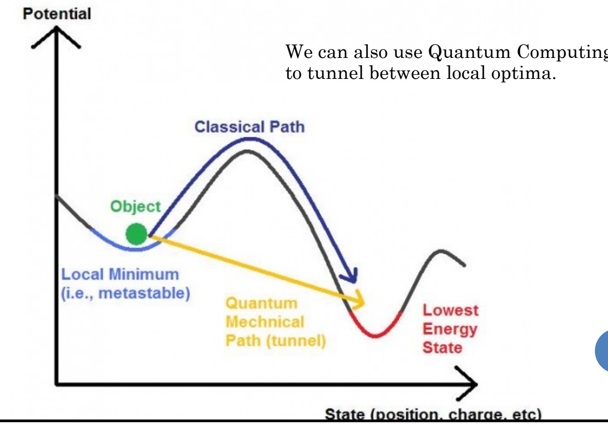
RECENT CLAIMS (2022): We can use Quantum Computing to find which bit flips lead to an improving move in O(1) time.

THIS IS A SOLVED PROBLEM: You don't need Quantum.

45

45

QUANTUM TUNNELING



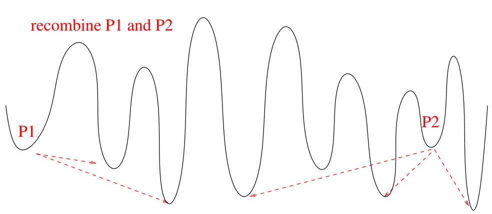
46

46

BUT, WE CAN ALREADY TUNNEL BETWEEN OPTIMA ON SOME NP-HARD PROBLEMS IN O(N) TIME:

MAX-3SAT, TRAVELING SALESMAN, QUBO

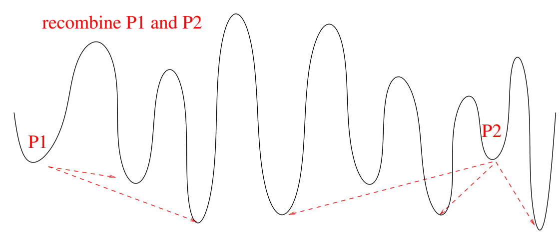
WITHOUT QUANTUM COMPUTING



47

47

FURTHERMORE, THE LOCAL OPTIMA ARE ARRANGED IN LATTICES

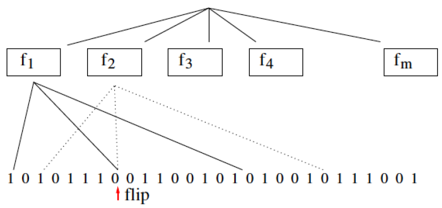


48

48

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.

49

49

Improving Moves in Hadamard Space

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

(Warning, the notation is compressed.)

First, let's take the Hadamard (Fourier, Walsh) Transform and convert our evaluation function into a polynomial.

If the number of subfunctions is O(n), the number of terms in the polynomial is also O(n).

50

50

The Hadamard Transform: MAXSAT

$$\frac{1}{8}\begin{bmatrix}1\\1\\1\\1\\0\\1\\1\\1\end{bmatrix}^T \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix}0.875\\-0.125\\-0.125\\-0.125\\0.125\\0.125\\0.125\\0.125\end{bmatrix}$$

The transform of a MAX-3SAT clause

51

The Hadamard Transform: MAXSAT

$$\frac{1}{8}\begin{bmatrix}1\\1\\1\\1\\0\\1\\1\\1\end{bmatrix}^T \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix}0.875\\-0.125\\-0.125\\-0.125\\0.125\\0.125\\0.125\\0.125\end{bmatrix}$$

MAX-3SAT:

If there are 1 million variables, and 4 million clauses, there are at most 17 million coefficients in the polynomial.

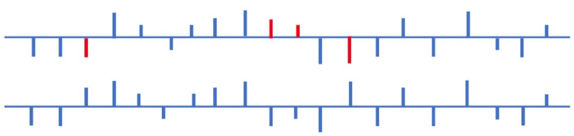
On real world problems it is more like 8 million coefficients.

52

What happens when you FLIP a bit?

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

(Warning, the notation is compressed.)



53

53

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

54

54

Constant Time Improving Moves

IMPROVING_MOVE_LIST: y_6, y_5

Flip 6, which interacts with 3 and 8, UPDATE.

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

IMPROVING_MOVE_LIST: y_8, y_5

55

55

What about looking 2 moves ahead?

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

56

56

What about looking 2 moves ahead?

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

57

57

What about looking 2 moves ahead?

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

If the number of subfunctions is $m=O(n)$
The number of pairs must be $O(n)$ and not $O(n^2)$

Thus, the number of possible improving moves is $O(n)$.

58

58

What about looking 2 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					

The number of pairs is also linear

59

59

What about looking 5 or more moves ahead?

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

60

60

What about looking 5 or more moves ahead?

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

61

61

What about looking 5 or more moves ahead?

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

There are always at most a linear number of nonlinear terms to consider.

62

62

Transforms

- SAT to MAXSAT
  - For decades, SAT problems have been converted into MAX-3SAT instances. Modern SAT solvers expect a MAXSAT form.
  - TRANSFORMS may also serve as REDUCTIONS used to prove NP-Completeness.

63

63

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

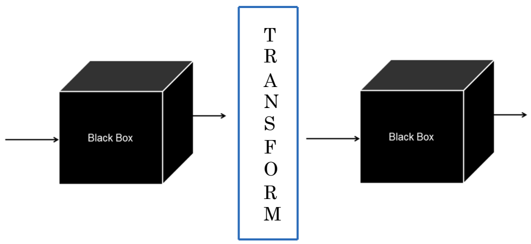
This assumes a polynomial evaluation function.

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

64

64

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



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

65

65

Transforms

For example, you could convert a Boolean function where

N = 10,000, k=10

Into an QUBO where

N= 20,000, k=2

A PROJECTION INTO A HIGHER DIMENSION
WITH LOWER NON-LINEARITY

66

66

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(01111111111111111) = 0

f(10011001100011111) = 1

f(11111000011000111 1) = 5

67

67

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 Fourier Polynomial is EXPONENTIAL in Size!

Every variable has a nonlinear interaction
with every other variable!

68

68

1519

TRANSFORMS: By 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 sufficiently large.

69

69

For Leading Ones n=4, p=5

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

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

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

70

70

For Leading Ones: QUBO (Quadratic)

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

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

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

71

71

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 Fourier Polynomial is LINEAR in size.

72

72

Leading Ones in General:

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

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

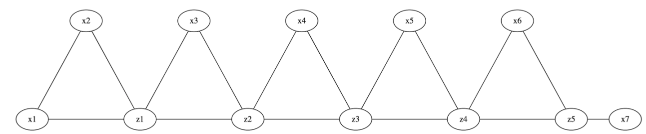
You can use **Local Search with Lookahead** to solve the problem in O(n) time.

The problem is also submodular.

73

73

Leading Ones: A "Variable Interaction Graph" (VIG)



RESULT:

Quadratic Leading Ones problem has a bounded tree width w < 4 and is solved by Dynamic Programming in O(n) time.

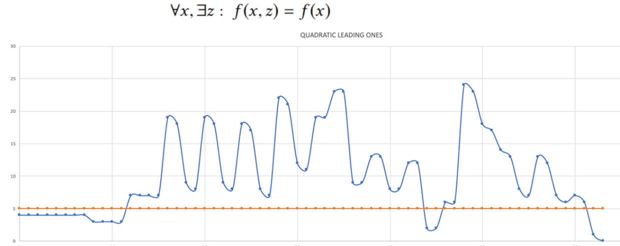
74

74

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:*

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



75

75

What about Tunneling?

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

We could consider any ***k-bounded Boolean***

The variables interactions are the same.

Note we have named the subfunctions: a to z.

76

76

What about Tunneling?

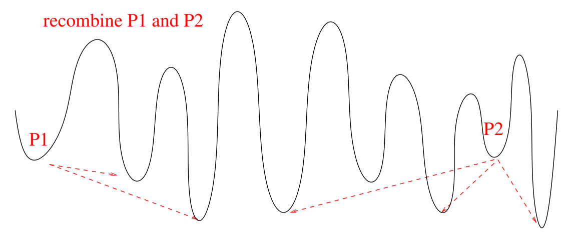
$f_a(1,0,6)$	$f_l(6,10,13)$	$f_q(11,16,17)$	$f_v(15,7,13)$
$f_b(2,1,6)$	$f_m(8,3,6)$	$f_r(12,10,17)$	$f_w(16,9,11)$
$f_c(1,2,4)$	$f_n(7,12,15)$	$f_s(13,12,15)$	$f_x(17,5,16)$
$f_d(4,1,14)$	$f_o(9,11,14)$	$f_t(14,4,16)$	$f_y(3,7,13)$
$f_e(5,4,2)$	$f_p(10,2,17)$	$f_u(9,14,16)$	$f_z(0,6,14)$

In this approach, we need two local optima before we can tunnel.

77

77

Tunneling is a form of "Recombination"

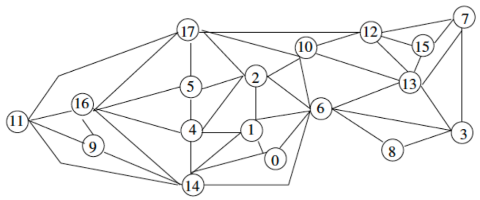


We will delete the bits that P1 and P2 share in common.
If we are lucky, this will "Shatter" the Variable Interaction Graph.

78

78

The Variable Interaction Graph

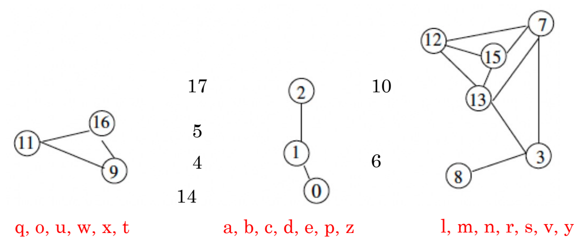


LOCAL OPTIMUM P1: 000000000000000000
LOCAL OPTIMUM P2: 111100011101110110

79

79

The Recombination Graph:
Parent 1: 000000000000000000
Parent 2: 111100011101110110

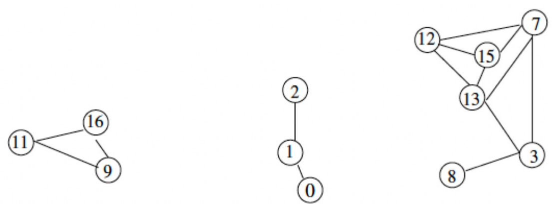


Delete vertices: 4, 5, 6, 10, 14, 17

80

80

The Recombination Graph: the Decomposed VIG.



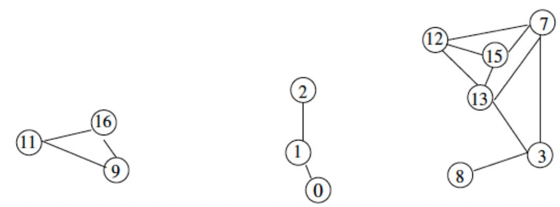
q, o, u, w, x, t   a, b, c, d, e, p, z   l, m, n, r, s, v, y

This decomposes the variables **and** the subfunctions.

81

81

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

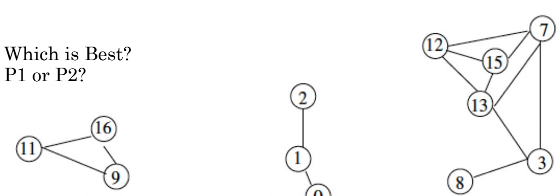
This decomposes the variables **and** the subfunctions.

82

82

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 decomposes the variables **and** the subfunctions.

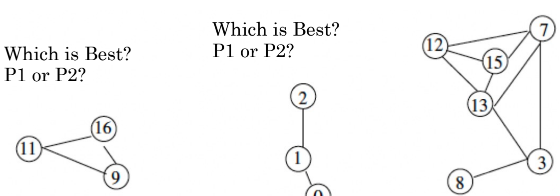
83

83

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

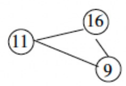
This decomposes the variables **and** the subfunctions.

84

84

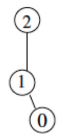
The Recombination Graph: Be GREEDY

Which is Best?
P1 or P2?



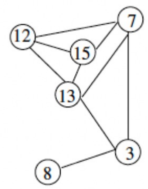
q, o, u, w, x, t

Which is Best?
P1 or P2?



a, b, c, d, e, p, z

Which is Best?
P1 or P2?



l, m, n, r, s, v, y

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

85

85

Partition Crossover and Local Optima.

The Subspace Optimality Theorem:

For all k-bounded pseudo-Boolean function, f:

If the parents are local optima then all of the offspring are local optima in the smallest hyperplace subspace that contains the two parents.

86

86

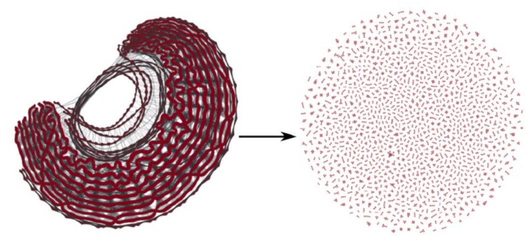
What does the VIG and Recombination Graph look like on Real World Problems?

87

87

The VIG

The Recombination Graph.



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)

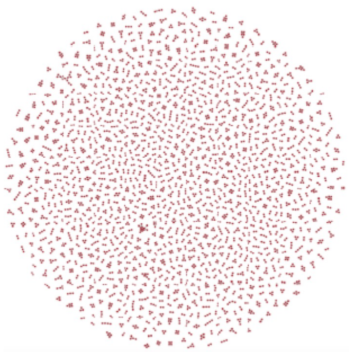
88

88

Decomposed Evaluation for MAXSAT

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

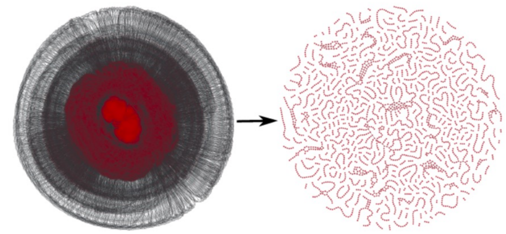
All offspring are Local Optima in this subspace.



89

89

More MAXSAT



LABS_n088_goal008

Finding low autocorrelation binary sequence: 371 components

PX returns the best of 2^{371} offsprings.

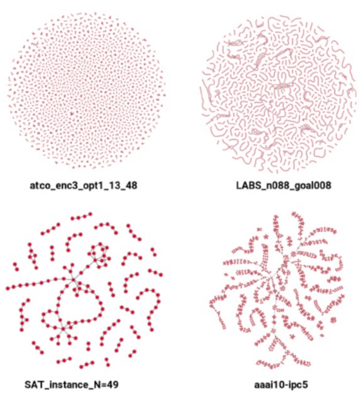
N= 182,015

(Thanks to Wenxiang Chen)

90

90

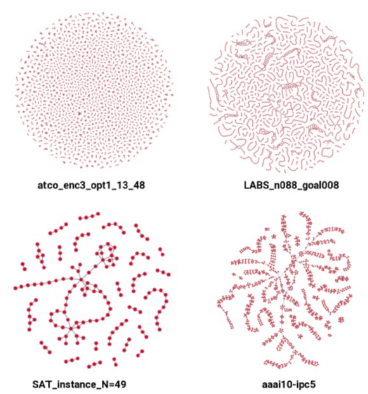
More MAXSAT



91

91

More MAXSAT



These subproblems have a tree decomposition with low width.

Thanks to Francisco Chicano.

These subproblems can be solved by Dynamic Programming!

92

92

Partition Crossover and Local Optima.

The Subspace Optimality Theorem:

For any k-bounded pseudo-Boolean function, *f:*

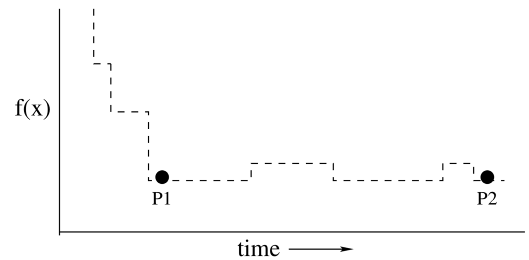
If the parents are local optima,
then all offspring are local optima
in the smallest hyperplane subspace
that contains the two parents.

TUNNELING BETWEEN OPTIMA in O(N) time.

93

93

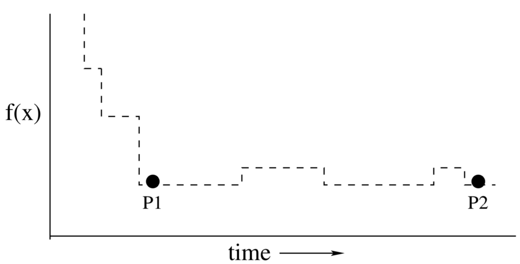
Max-3SAT and Plateaus



94

94

Max-3SAT and Plateaus



RUN LOCAL SEARCH FIRST, THEN APPLY CROSSOVER.
There is NO POPULATION.

95

95

Local Search Algorithms for MAXSAT

AdaptG^2WSAT: Best in the 2007 SAT Competition

NEW: AdaptG^2WSAT with Partition Crossover

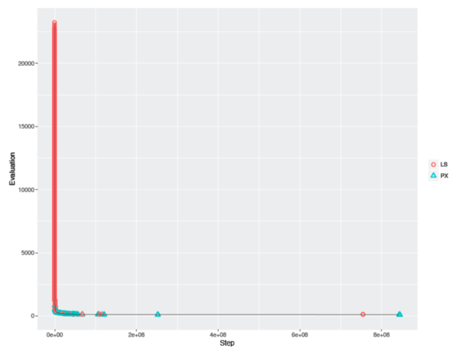
Sparrow: Best among all local search over in "crafted" and "Application" SAT Track in 2014 SAT Competition.

NEW: Sparrow with Partition Crossover

96

96

Early MAXSAT Results

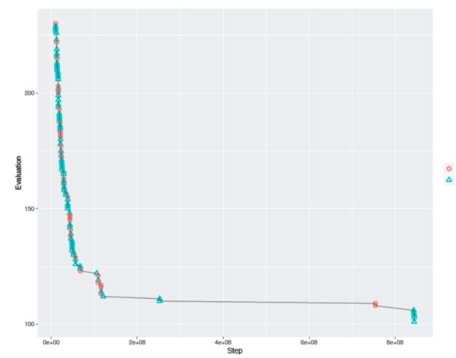


RED is intelligent local search. BLUE is Partition Crossover

97

97

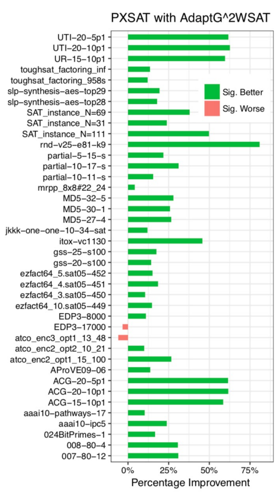
Early MAXSAT Results



RED is intelligent local search. BLUE is Partition Crossover

98

98



MAXSAT

RESULTS.

PARTITION CROSSOVER

HELPS ON

HARD PROBLEMS.

99

99

MAXSAT RESULTS

Theorem

When recombining parents P_1 and P_2:

$$\frac{f(P_1)}{2} + \frac{f(P_2)}{2} = \frac{1}{2^q}\sum_{i=1}^{2^q} f(C_i)$$

Corollary

Assume that $f(P1) = f(P2)$.
*If **any** offspring represents a disimproving move, there must also exist an offspring that yields an improving move.*

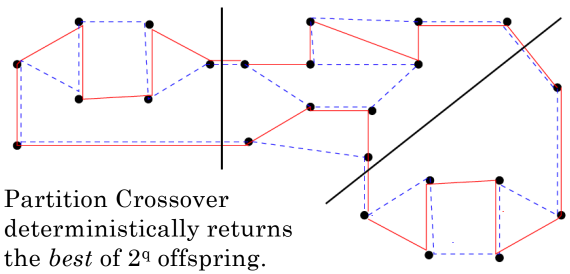
This makes Partition Crossover very different than local search for MAXSAT. For local search the discovery of a disapproving move says nothing about the existence of an improving move.

100

100

``Tunnels" and Traveling Saleman

Assume the Parents are Local Optima (***under ANY Operator***).

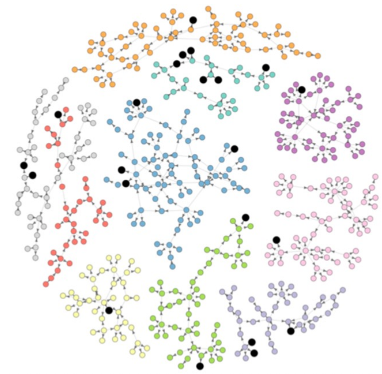


Partition Crossover deterministically returns the *best* of 2^q offspring.

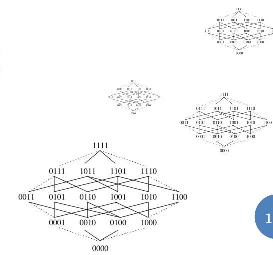
101

101

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



Each tunnel is one recombination, and each recombination is the *top of a lattice.*

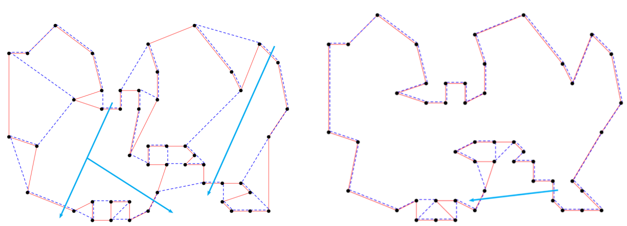


102

102

The quasi-local optima form a lattice in hyperspace:

Consider 4 "Parent tours" for a 50 city TSP: A, B, C, D



A cross D

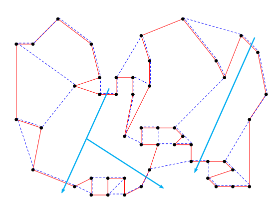
B cross C

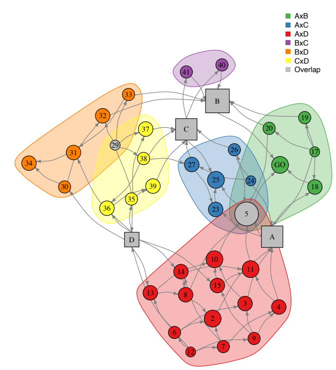
103

103

Local optima organize into lattices under recombination

There are 41 Local Optima

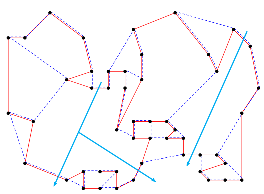


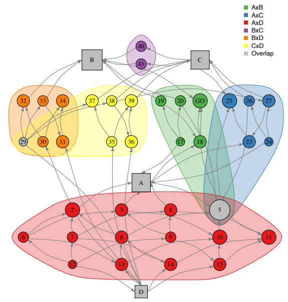


104

104

LOCAL OPTIMA ORGANIZE INTO LATTICES UNDER RECOMBINATION

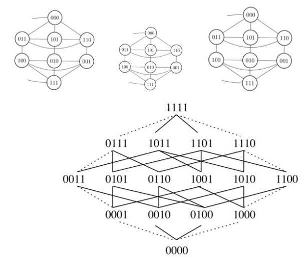


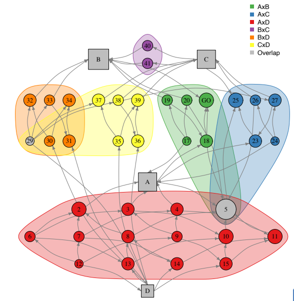


105

105

LOCAL OPTIMA ORGANIZE INTO LATTICES UNDER RECOMBINATION





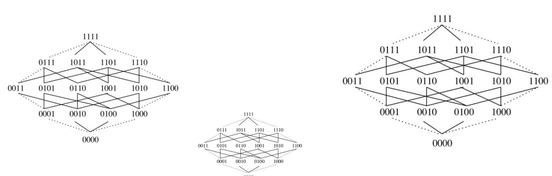
106

106

THEOREM: A LATTICE OF QUASI-LOCAL OPTIMA CAN BE EXPONENTIALLY LARGE:

PROOF BY CONSTRUCTION: Construct a traveling salesman problem (or MAXSAT instance) 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.

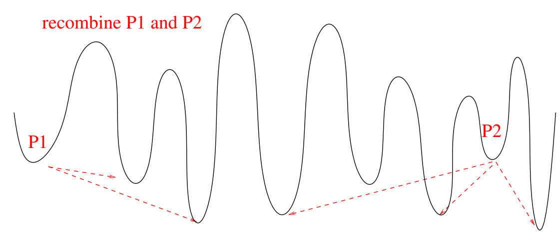
This results in a lattice of size $2^{N/c}$



107

107

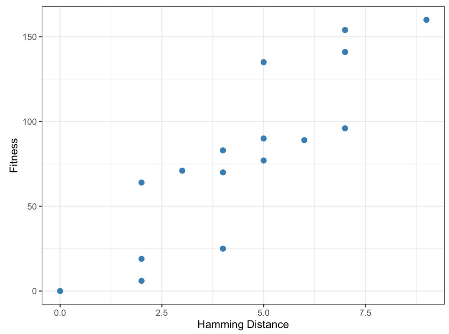
FURTHERMORE, THE LOCAL OPTIMA ARE ARRANGED IN LATTICES



108

108

THE
LOCAL OPTIMA ARE ARRANGED IN LATTICES



A k=3 Pseudo Boolean Function

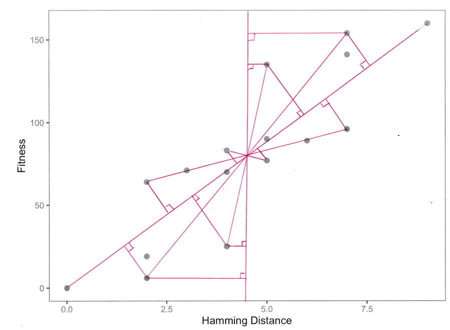
16 Local Opt

Partition Crossover

109

109

THE
LOCAL OPTIMA ARE ARRANGED IN LATTICES



A k=3 Pseudo Boolean Function

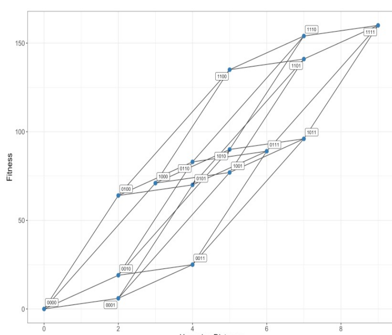
16 Local Opt

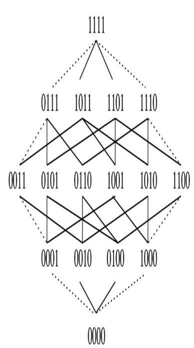
Partition Crossover

110

110

THE
LOCAL OPTIMA ARE ARRANGED IN LATTICES

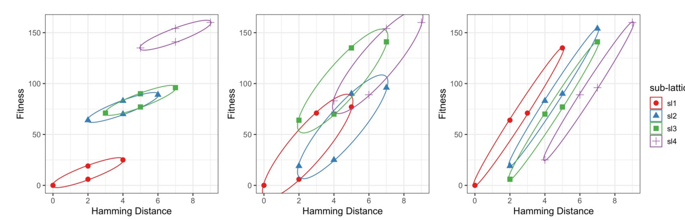




111

111

FURTHERMORE, THE
LOCAL OPTIMA ARE ARRANGED IN LATTICES



Local Optima under Partition Crosover.
New results use Linear Equations to exactly compute this distribution.

112

112

One Last Thought:

What if DNA is K-bounded?

E.g., the fitness landscape is an NK-Landscape

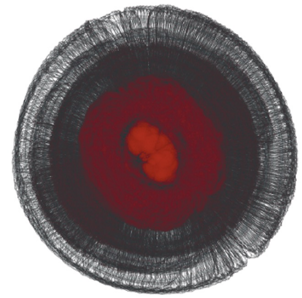
113

113

One Last Thought:

What if DNA is K-bounded?

What if "gene interaction" looks like this?



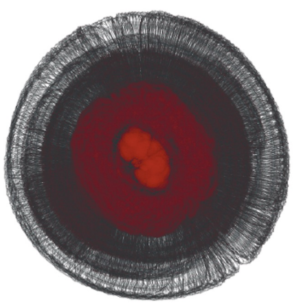
114

114

One Last Thought:

What if DNA is K-bounded?

99.9% of DNA is identical in all humans



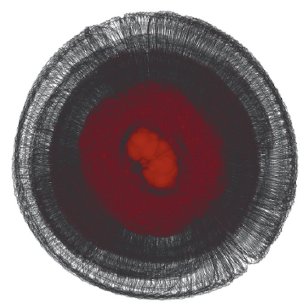
115

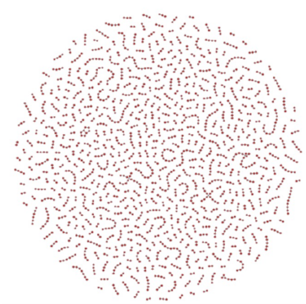
115

One Last Thought:

What if DNA is K-bounded?

99.9% of DNA is identical in all humans





116

116

Questions?

117

117