Gray Box Optimization: Theory and Practice

Darrell Whitley, Colorado State University

THANKS TO:

Renato Tinós
University of Saõ Paulo

Francisco Chicano
University of Málaga

Gabriela Ochoa
University of Stirling

Andrew Sutton
Hasso-Plattner-Institut
Know your Landscape! And Go Downhill!
How do we Exploit Problem Structure?
Problem Decomposition

There are often many opportunities to exploit problem decomposition.

Decomposition requires the use of Gray Box Optimization.
Black Box Optimization

Strong Claim: Never use Black Box Optimization if it can be avoided.

Black Box Optimization does not allow us to ask the right questions, because we cannot see the answers.
Mk Landscapes: $k$-bounded Pseudo-Boolean Problems

$$f(x) = \sum_{i=1}^{m} f_i(x, \text{mask})$$
MAX-kSAT, NK Landscapes, Spin Glass

\[ f(x) = \sum_{i=1}^{m} f_i(x, \text{mask}) \]
Mk-Landscapes

For example: An NK Landscape: $n = 10$ and $k = 3$.
The subfunctions:

\[ f_0(x_0, x_1, x_6) \quad f_1(x_1, x_4, x_8) \quad f_2(x_2, x_3, x_5) \quad f_3(x_3, x_2, x_6) \]
\[ f_4(x_4, x_2, x_1) \quad f_5(x_5, x_7, x_4) \quad f_6(x_6, x_8, x_1) \quad f_7(x_7, x_3, x_5) \]
\[ f_8(x_8, x_7, x_3) \quad f_9(x_9, x_7, x_8) \]

But this could be a MAXSAT Function, or an arbitrary Spin Glass problem.
A General Result over Bit Representations

By Constructive Proof: Every problem with a bit representation and a closed form evaluation function can be expressed as a quadratic pseudo-Boolean Optimization problem. (Boros and Hammer 2002)

For example, depending on the nonlinearity:

\[
f(x_1, x_2, x_3, x_4, x_5, x_6) \quad \text{becomes} \quad f(x) = f_1(z_1, z_2, z_3) + f_2(z_1, x_1, x_2) \\
+ f_3(z_2, x_3, x_4) + f_4(z_3, x_5, x_6)
\]

The SAT community regularly transforms general SAT expressions into MAX-kSAT (which is also an Mk Landscape).
GRAY BOX OPTIMIZATION

Exploit the *general properties of every* Mk Landscape:

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

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

\[ f(x) = \sum_{i=1}^{k} E_k(W(f(x))) \]
P and NP

NP

P

Closed Problems
Random and Localized Mk Landscapes

Random Mk Landscapes

NP

Structured Mk Landscapes

Localized Mk Landscapes

P

Closed Problems
Does the theoretical complexity analysis of problems in the class P tell us anything about the complexity of problems in NP? Not Clear.
Do empirical results on Random Landscapes tell us anything about Structured Landscapes in the class NP? Not Clear.
Some problems can be solved by dynamic programming.

An Adjacent NK Landscape: $n = 6$ and $k = 3$. The subfunctions:

\[
\begin{align*}
    f_0(x_0, x_1, x_2) \\
    f_1(x_1, x_2, x_3) \\
    f_2(x_2, x_3, x_4) \\
    f_3(x_3, x_4, x_5) \\
    f_4(x_4, x_5, x_0) \\
    f_5(x_5, x_0, x_1)
\end{align*}
\]

These problems can be solved to optimality in $O(N2^{2k})$ time (Hammer 1965, Crama 1990). This is faster than the Wright algorithm (2000) which has $O(N2^{3k})$ complexity.
Localized Mk-Landscapes

**Definition:** A “Tree Decomposition” Mk Landscape has bounded tree width $w$.

**Under Gray Box Optimization:**

1. All Tree Decomposition Mk-Landscapes can be solved in $O(N \ 2^w)$.

2. Separable Mk-Landscapes (including ONEMAX and TRAP functions) are solved in 1 evaluation and $O(N \ 2^k)$ time.
Tree Decomposition Mk Landscapes

Figure: Example of Variable Interaction Graph for a Localized Mk Landscape. This corresponding function can be embedded into a Tree Decomposition.
Figure: Example of $M \times k$ look-up table of variables of a TD Mk Landscape. In the example $n = 23$, $M = 10$ and $k = 6$. Each row of the table can become a subfunction in an Mk Landscape, with variables V1 to V6. The table also corresponds to a Tree Decomposition of that same set of functions.
We can compute the Walsh coefficients in $O(n)$ time (assuming $m = O(n)$).

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

The average fitness of hyperplane order 1 $h$ is

$$Avg(h) = w_0 + w_{\alpha(h)}(-1^{\text{bitcount}(\beta(h))})$$
Definition: A pseudo-Boolean Optimization problem is *not deceptive* if:
\[ \forall H_1, H_2 : \text{Avg}(H_1) < \text{Avg}(H_2) \iff \text{global minima is contained in } H_1 \]

Recall that the average fitness of hyperplane order 1 \( h \) is
\[
\text{Avg}(h) = w_0 + w_{\alpha(h)}(-1^{\text{bitcount}(\beta(h))})
\]

**Theorem:** Every Mk Landscape that is not deceptive can be solved in 1 Evaluation in \( O(n) \) time by a Gray Box optimizer.
ONEMAX is solved in 1 Evaluation and $O(n)$ time.

LEADING ONES is solved in 1 Evaluation and $O(n)$ time.

TRAP FUNCTIONS are solved in 1 Evaluation and $O(n)$ time.

The "Clause Count" used in SAT solvers also calculates the Order 1 Hyperplane averages. Any nontrivial MAXSAT is deceptive.

All of these problems are decomposable in simpler, solvable problems.
A Random NK Landscape

For example: A Random NK Landscape: $n = 10$ and $k = 3$.

The subfunctions:

\[
\begin{align*}
  f_0(x_0, x_1, x_6) & \quad f_1(x_1, x_4, x_8) & \quad f_2(x_2, x_3, x_5) & \quad f_3(x_3, x_2, x_6) \\
  f_4(x_4, x_2, x_1) & \quad f_5(x_5, x_7, x_4) & \quad f_6(x_6, x_8, x_1) & \quad f_7(x_7, x_3, x_5) \\
  f_8(x_8, x_7, x_3) & \quad f_9(x_9, x_7, x_8) & & \\
\end{align*}
\]

But this could also be a MAXSAT Function, or an arbitrary Spin Glass problem.
The Variable Interaction Graph (VIG)

Variables $v_i$ and $v_j$ are connected there is a nonlinear interaction between $v_i$ and $v_j$. The number of edges is $O(M)$.

The VIG is representation invariant.
Partition Crossover: Decomposed Evaluation

The Variable Interaction Graph

The decomposed Recombination Graph

When recombining 0000000000 and 1100011101: delete the shared variables 2, 3, 4, 8.

Given $q$ partitions, Partition Crossover returns the best of $2^q$ solutions.
Decomposed Evaluation

THEOREM: the offspring is locally optimal in the largest hyperplane containing both parents.

Inherit all red bits together

Inherit all green bits together

1100011101 11 000 111 01
0000000000 00 000 000 00
A new evaluation function can be constructed:

\[ g(x) = c + g_1(x_5, x_7, x_9) + g_2(x_0, x_1, x_6) \]
The Variable Interaction Graph (VIG)
The PX Recombination Graph

The parents are: 000000000000000000 and 111100011101110110.
4,5,6,10,14,17 = 0
The hyperplane is ****000***0***0**0

Random Parents: Half of the bits are the same in expectation.
More bits are the same for parents that are local optima.
The Subspace Optimality Theorem for Partition Crossover: if the parents are locally optimal, the offspring must be a local optima in the largest hyperplane that contains both parents.

Example: if the parents 0000000000 and 1100011101 are locally optimal, all offspring are locally optimal in the hyperplane subspace ***000***0**.
**Partition Crossover and Local Optima**

**Corollary:** The only possible improving move for offspring generated from parents that are locally optimal must flip a bit that the parents shared in common.

The only improving moves are on shared bits: ***000***0**.
Very Simple Experiment

Construct a very simple genetic algorithm; add local search.

Population = 50. Tournament Selection. Mutation = $1/n$

Partition Crossover

Run for 100 generations (5000 evaluations)

Run the experiment 50 times.
## Percent of Offspring that are Local Optima

<table>
<thead>
<tr>
<th>$N$</th>
<th>$K$</th>
<th>Model</th>
<th>2-point Xover</th>
<th>Uniform Xover</th>
<th>PX</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>1</td>
<td>Adjacent</td>
<td>78.0 ±2.3</td>
<td>0.0 ±0.0</td>
<td>97.9 ±5.0</td>
</tr>
<tr>
<td>500</td>
<td>3</td>
<td>Adjacent</td>
<td>31.0 ±2.5</td>
<td>0.0 ±0.0</td>
<td>93.8 ±4.0</td>
</tr>
<tr>
<td>500</td>
<td>1</td>
<td>Random</td>
<td>0.0 ±0.0</td>
<td>0.0 ±0.0</td>
<td>98.3 ±4.9</td>
</tr>
<tr>
<td>500</td>
<td>3</td>
<td>Random</td>
<td>0.0 ±0.0</td>
<td>0.0 ±0.0</td>
<td>83.6 ±16.8</td>
</tr>
</tbody>
</table>
## Number of partition components discovered

<table>
<thead>
<tr>
<th>$N$</th>
<th>$K$</th>
<th>Model</th>
<th>Partition Mean</th>
<th>Crossover Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>1</td>
<td>Adjacent</td>
<td>7.66 ±0.47</td>
<td>55</td>
</tr>
<tr>
<td>500</td>
<td>3</td>
<td>Adjacent</td>
<td>7.52 ±0.16</td>
<td>41</td>
</tr>
<tr>
<td>500</td>
<td>1</td>
<td>Random</td>
<td>6.98 ±0.47</td>
<td>47</td>
</tr>
<tr>
<td>500</td>
<td>3</td>
<td>Random</td>
<td>2.46 ±0.05</td>
<td>13</td>
</tr>
</tbody>
</table>
Optimal Solutions: Adjacent NK Model

<table>
<thead>
<tr>
<th></th>
<th>2-point</th>
<th>Uniform</th>
<th>Partition Crossover</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>500</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>500</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Percentage over 50 runs where the global optimum was found in the experiments of the hybrid GA with the Adjacent NK Landscape.
Maximization: the Random NK Model

Mean evaluation over 50 runs for the hybrid GA with the random model. The evaluation functions is being maximized. The * indicates significantly better results.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$K$</th>
<th>2-point</th>
<th>Uniform</th>
<th>Paired PX</th>
<th>s.d.</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>1</td>
<td>0.7047</td>
<td>0.7049</td>
<td>0.7142*</td>
<td>±0.007</td>
</tr>
<tr>
<td>500</td>
<td>2</td>
<td>0.7306</td>
<td>0.7305</td>
<td>0.7402*</td>
<td>±0.006</td>
</tr>
<tr>
<td>500</td>
<td>3</td>
<td>0.7440</td>
<td>0.7442</td>
<td>0.7455*</td>
<td>±0.005</td>
</tr>
</tbody>
</table>
Using Crossover to Tunnel Between Optima
But a Hybrid Genetic Algorithm is NOT how we should solve these NK Landscape Problems.
Walsh Analysis

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

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

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

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

If $m$ is $O(n)$ then the number of Walsh coefficients is $O(n)$. 
A General Model for all bounded Pseudo-Boolean Problems

\[ f(x) = \sum_{i=1}^{m} f_i(x, \text{mask}) \]
When 1 bit flips what happens?

\[ f(x) = \sum_{i=1}^{m} f_i(x, \text{mask}_i) \]
Assume we flip bit $p$ to move from $x$ to $y_p \in N(x)$.

Construct a vector $Score$ such that

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

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

**NOTE:** Others have suggested a constant time result, but without proof. An average case complexity proof is required to obtain a general constant time complexity result (Whitley 2013, AAAI).
Lemma 1.
Let $y_p \in N(x)$ be the neighbor of string $x$ generated by flipping bit $p$.

$$f(y_p) = f(x) + \text{Score}(x, y_p)$$

$$\text{Score}(x, y_p) = f(y_p) - f(x)$$

On average only a constant number of Scores change after a bit flip.

Minimizing $\text{Score}(x, y_p)$ minimizes the neighborhood of $f(x)$. 
The locations of the updates are obvious

\[
\begin{align*}
Score(y_p, y_1) &= Score(x, y_1) \\
Score(y_p, y_2) &= Score(x, y_2) \\
Score(y_p, y_3) &= Score(x, y_3) - 2 \sum_{\forall b, (p \land 3) \subseteq b} w'_b(x) \\
Score(y_p, y_4) &= Score(x, y_4) \\
Score(y_p, y_5) &= Score(x, y_5) \\
Score(y_p, y_6) &= Score(x, y_6) \\
Score(y_p, y_7) &= Score(x, y_7) \\
Score(y_p, y_8) &= Score(x, y_8) - 2 \sum_{\forall b, (p \land 8) \subseteq b} w'_b(x) \\
Score(y_p, y_9) &= Score(x, y_9)
\end{align*}
\]
**Theorem**

Select a constant $\lambda$ such that $\lambda \geq ck$. Select a second constant $\beta$. If any variable appears in more than $\lambda$ subfunctions it will only be flipped $\beta$ times during any sequence of $n$ moves. Then the amortized cost per bit flip move associated with the updates to the Score vector is $\Theta(1)$ and is bounded by $(\beta + 1)\alpha k \lambda$ over any sequence of improving moves.
Constant Time Steepest Descent

Sketch of the PROOF:

Let $C$ be the set of variables that appear in more than $\lambda$ subfunctions. These variables can be flipped at most $\beta$ times. Collectively the associated runtime cost is bounded by:

$$\beta \sum_{j \in C} U_j < \beta \sum_{i=1}^{n} U_i = \beta n \bar{U} \leq \beta n \alpha k^2.$$  

where $U_i$ is the update cost after flipping bit $i$ and $\alpha$ is the cost to update one location in the Score vector. The total work is bounded by

$$\beta n \alpha k^2 + n \lambda \alpha k \leq n(\beta + 1) \alpha k \lambda,$$

since $\lambda \geq ck$. Thus, over any sequence of $n$ moves, the amortized number of updates to the Score vector is less than $(\beta + 1) \alpha k \lambda$ for 1 move.
What if we could look 4, 5, or 6 Steps Lookahead?

Assume we wish to look 3 moves ahead by flipping bits $i, j, k$.

Let $Score(3, x, y_{i,j,k})$ indicate we move from $x$ to $y_{i,j,k}$ by flipping the 3 bits $i, j, k$. In general, we will compute $Score(r, x, y_p)$ when flipping $r$ bits.

\[
\begin{align*}
    f(y_i) &= f(x) + Score(1, x, y_i) \\
    f(y_{i,j}) &= f(y_i) + Score(1, y_i, y_j) \\
    f(y_{i,j}) &= f(x) + Score(2, x, y_{i,j}) \\
    f(y_{i,j,k}) &= f(y_{i,j}) + Score(1, y_{i,j}, y_k) \\
    f(y_{i,j,k}) &= f(x) + Score(3, x, y_{i,j,k})
\end{align*}
\]
Why Doesn’t this exponentially EXPLODE???

\[ f(y_{i,j,k}) = ((f(x) + \text{Score}(1, x, y_i)) + \text{Score}(1, y_i, y_j)) + \text{Score}(1, y_i, y_j, y_k) \]

\[ \text{Score}(3, x, y_{i,j,k}) = \text{Score}(2, x, y_{i,j}) + \text{Score}(1, y_{i,j}, y_{i,j,k}) \]

If there is no Walsh Coefficient \( w_{i,j} \) then \( \text{Score}(1, y_i, y_{i,j}) = 0 \).

If there are no Walsh Coefficients “linking” \( i, j, k \) then \( \text{Score}(3, x, y_{i,j,k}) = 0 \).
The Variable Interaction Graph
What if we could look 3, 5, 7 Steps Lookahead? No Problem.

Deterministic Crossover + Deterministic Moves = Optimal Solutions on 1,000,000 variable Adjacent NK Landscapes.
In this figure, $N = 12,000$, $k=3$, and $q=4$. The radius is 1, 2, 3, 4, 5, 6. At $r=6$ the global optimum is found.
Deterministic Moves + Deterministic Crossover

The Variable Interaction Graph

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>4</td>
<td>12</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>9</td>
<td>11</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>7</td>
<td>11</td>
<td>9</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>9</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>9</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>9</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>9</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>9</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>9</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>11</td>
<td>9</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>COUNT: 49 (About 2*n)</td>
<td>POSSIBLE</td>
<td>165</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Variable Interaction Graph
Black Box Optimization does not allow us to ask the right questions, because we cannot see the answers.
We only need a tiny amount of information for Gray Box Optimization to yield dramatic improvements.
Latest Results

- Best TSP Solver in the World (2016). Yes we can!

- Best Solver in the World (2016) for Gene Marker Sequencing. Yes we can!

- Optimal solutions on 1 million variable problems. Yes we can!

- Find the best of $2^{1000}$ solutions in one recombination. Yes we can!

- Define a new class of search algorithms using deterministic recombination and deterministic $O(1)$ moves.

- Multi-Objective Deterministic Operators. Some results.
Partition Crossover for the TSP

We want to limit our attention to PX (GPX) recombination operators that have $O(n)$ complexity. Simple version of GPX execute in approximately $4n$ steps.
What if you could ... "Tunnel" between local optima.

Tunneling = jump from local optimum to local optimum
What if you could …

recombine P1 and P2

“Tunnel” between local optima.

Tunneling = jump from local optimum to local optimum
Partition Crossover

A Depiction of Multiple funnels

Objective Function Value

Best local minimum in this funnel

Local minima

Global minimum

Search Space
Partition Crossover
Partition Crossover in $O(N)$ time
Generalize Partition Crossover is always feasible if the partitions have 2 exits (same color in and out). If a partition has more than 2 exits, the “colors” must match.

This will automatically happen if all of the partitions have cut two.
GPX, Cuts on Nodes of Degree 4
GPX, Cuts Crossing 4 Edges
How Big is $q$? How Many Partitions?

<table>
<thead>
<tr>
<th>Instance</th>
<th>rand1500</th>
<th>u1817</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-opt</td>
<td>$q = 25 \pm 0.2$</td>
<td>$q = 26 \pm 0.7$</td>
</tr>
</tbody>
</table>

This illustrates the average number of *partition subgraphs* used by Partition Crossover recombining random local optima found by 3-opt for modest size problems. (The sample size is 50.)

With 25 components, $2^{25}$ represents millions of local optima; more than $2^{24}$ are true local optima.
Lin-Kernighan-Helsgaun-LKH

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

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

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

Iterated Partial Transcription (IPT)

It recombines every new local optima with the best-so-far solution.
A diagram depicting 10 runs of multi-trial LKH-2 run for 5 iterations per run. The circles represent local optima produced by LKH-2. GPX across runs crosses over solutions with the same letters. GPX across restarts crosses over solutions with the same numbers.
GPX on Clustered Problems

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

(a) (b) (c) (d)
Table: Results for SYMMETRIC TSPs. The number of times that GPX2+LKH improved the mean and best results of LKH is presented. For the best results, only the times that LKH did not reached known optimum are counted.

<table>
<thead>
<tr>
<th>Problem Type</th>
<th>all-to-all recombination</th>
<th>improved mean</th>
<th>improved best</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-,C-Instances</td>
<td>18 out of 18</td>
<td>15 out of 16</td>
<td></td>
</tr>
<tr>
<td>VLSI TSPs</td>
<td>5 out of 9</td>
<td>2 out of 5</td>
<td></td>
</tr>
<tr>
<td>National TSPs</td>
<td>8 out of 10</td>
<td>5 out of 8</td>
<td></td>
</tr>
<tr>
<td>Art TSP</td>
<td>1 out of 1</td>
<td>1 out of 1</td>
<td></td>
</tr>
</tbody>
</table>

REMINDER: IPT recombination is similar to GPX. GPX2 must find more partitions and/or opportunities to apply crossover.
Table: Results for ASYMMETRIC TSPs. The number of times that GPX2+LKH improved the mean and best results of LKH is presented. For the best results, only the times that LKH did not reached known optimum are counted.

<table>
<thead>
<tr>
<th>Problem Type</th>
<th>all-to-all recombination</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>improved mean</td>
<td>improved best</td>
<td></td>
</tr>
<tr>
<td>rtilt</td>
<td>12 out of 12</td>
<td>10 out of 12</td>
<td></td>
</tr>
<tr>
<td>stilt</td>
<td>12 out of 12</td>
<td>9 out of 12</td>
<td></td>
</tr>
<tr>
<td>crane</td>
<td>11 out of 12</td>
<td>8 out of 12</td>
<td></td>
</tr>
<tr>
<td>disk</td>
<td>4 out of 12</td>
<td>0 out of 10</td>
<td></td>
</tr>
<tr>
<td>coin</td>
<td>12 out of 12</td>
<td>10 out of 12</td>
<td></td>
</tr>
<tr>
<td>shop</td>
<td>5 out of 12</td>
<td>2 out of 11</td>
<td></td>
</tr>
<tr>
<td>super</td>
<td>0 out of 12</td>
<td>0 out of 9</td>
<td></td>
</tr>
</tbody>
</table>

REMINDER: IPT recombination is similar to GPX. GPX2 must find more partitions and/or opportunities to apply crossover.
Veerapen et al: Tunnelling Crossover Networks

For TSP (Tuesday, 2pm, Session 5)
Structured MAXSAT Problems

When converting general SAT expressions into MAXSAT, there is an intermediate form:

$$\phi = (x_{18} \leftrightarrow (x_{38} \lor x_{14}))$$

This intermediate form can be directly converted into CNF SAT.

$$\phi \equiv (\neg 38 \lor 14 \lor 18) \land (\neg 38 \lor \neg 14 \lor 18) \land (38 \lor 14 \lor \neg 18) \land (38 \lor \neg 14 \lor 18)$$

This can also be expressed as a much more compact Mk Landscape, which we will call “DNF functions”.

$$\phi \equiv f(38, 14, 18) = <1, 0, 0, 1, 0, 1, 0, 1>$$
Nonlinear Walsh terms per variable

<table>
<thead>
<tr>
<th></th>
<th>$k = 3$</th>
<th>$k = 4$</th>
<th>$k = 5$</th>
<th>$k = 6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random MAXSAT/NK</td>
<td>4n</td>
<td>11n</td>
<td>26n</td>
<td>57n</td>
</tr>
<tr>
<td>Adjacent NK Landscapes</td>
<td>3n</td>
<td>7n</td>
<td>15n</td>
<td>31n</td>
</tr>
<tr>
<td>Industrial SAT</td>
<td>1.7n</td>
<td>1.7n</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NP

Random Mk Landscapes

Structured Mk Landscapes

Localized Mk Landscapes

Closed Problems
A “natural block” of CNF clauses in the DIMACS format. This was drawn from a problem in the 2014 SAT Competition. These are in the same order in which they were generated:

\[-38 \ 14 \ 18 \ 0\]
\[-38 \ -14 \ -18 \ 0\]
\[38 \ 14 \ -18 \ 0\]
\[38 \ -14 \ 18 \ 0\]
\[-14 \ -18 \ 39 \ 0\]
\[14 \ 18 \ -39 \ 0\]
\[14 \ -39 \ 0\]
\[18 \ -39 \ 0\]

\[-38 \ 14 \ 18 \ 0 = (\neg 38 \lor 14 \lor 18)\]
Structured MAXSAT Problems

Why do “natural blocks” of CNF clauses occur?
When converting general SAT expressions into MAXSAT, there is an intermediate form:

\[ \phi = (x_{18} \leftrightarrow (x_{38} \lor x_{14})) \]

This intermediate form can be directly converted into CNF SAT.

\[ \phi = (\neg 38 \lor 14 \lor 18) \land (\neg 38 \lor \neg 14 \lor 18) \land (38 \lor 14 \lor \neg 18) \land (38 \lor \neg 14 \lor 18) \]

This can also be expressed as a much more compact MK Landscape, which we will call “DNF functions”.

\[ f(38, 14, 18) = < 1, 0, 0, 1, 0, 1, 0, 0 > \]

\[ \phi = (x_{18} \leftrightarrow (x_{38} \lor x_{14})) \equiv f(38, 14, 18) = < 1, 0, 0, 1, 0, 1, 0, 1 > \]
Structured MAXSAT Problems

And what about these 8 clauses?

\[-38 \ 14 \ 18 \ 0\]
\[-38 \ -14 \ -18 \ 0\]
\[38 \ 14 \ -18 \ 0\]
\[38 \ -14 \ 18 \ 0\]
\[-14 \ -18 \ 39 \ 0\]
\[14 \ 18 \ -39 \ 0\]
\[14 \ -39 \ 0\]
\[18 \ -39 \ 0\]

\[F(38,14,18,19) = <1,0,0,0,0,0,0,1,0,0,1,0,1,0,0,0,0>\]

The only optimal assignments are

\[0000 \ 0111 \ 1010 \ 1100\]
Structured MAXSAT problems

Consider a MAX semi-prime factoring problems with 83 variables and 369 clauses.

This was generated using the TOUGH SAT generator.

How much does it compress when converted into a compact Mk Landscape?
Structured MAXSAT problems

Consider a MAX-kSAT semi-prime factoring problem with 83 variables and 369 clauses.

This was generated using the TOUGH SAT generator.

How much does it compress when converted into a compact Mk Landscape?
A SAT factoring problem with 369 clauses

\[ F_1(2, 3, 4, 5) := F_1(7, 8, 9, 10) := 0111 \ 1111 \ 1111 \ 1111 \]

\[ F_2(42, 16, 20, 41, 43) := F_2(44, 17, 21, 43, 45) := \]
\[ F_2(50, 25, 29, 49, 51) := F_2(52, 26, 30, 51, 53) := \]
\[ F_2(54, 27, 31, 53, 55) := F_2(60, 46, 34, 59, 61) := \]
\[ F_2(62, 47, 35, 61, 63) := F_2(70, 42, 48, 69, 71) := \]
\[ F_2(72, 58, 50, 71, 73) := F_2(74, 60, 52, 73, 75) := \]
\[ F_2(76, 62, 54, 75, 77) := F_2(78, 64, 56, 77, 79) := \]
\[ F_2(80, 66, 57, 79, 81) := F_2(40, 15, 19, 39, 41) := \]
\[ 1000 \ 0001 \ 0001 \ 0100 \ 0010 \ 1000 \ 1000 \ 0001 \]

\[ F_3(46, 22, 45, 47) := F_3(48, 24, 28, 49) := F_3(56, 32, 55, 57) := \]
\[ F_3(58, 44, 33, 59) := F_3(66, 37, 65, 67) := F_3(68, 40, 23, 69) := \]
\[ F_3(82, 67, 81, 83) := F_3(64, 36, 63, 65) := F_3(38, 14, 18, 39) := \]
\[ 1000 \ 0001 \ 0010 \ 1000 \]

\[ F_4(18, 1, 7, 23, 8) := F_4(28, 1, 9, 33, 10) := \]
\[ F_4(14, 2, 6, 19, 7) := F_4(29, 2, 9, 24, 8) := \]
\[ F_4(15, 3, 6, 20, 7) := F_4(25, 3, 8, 30, 9) := \]
\[ F_4(16, 4, 6, 21, 7) := F_4(26, 4, 8, 31, 9) := \]
\[ F_4(17, 5, 6, 22, 7) := F_4(27, 5, 8, 32, 9) := \]
\[ 1100 \ 1100 \ 1001 \ 0000 \ 0000 \ 0000 \ 0000 \ 1001 \]

\[ F_5(34, 2, 10, 35, 3) := F_5(37, 5, 10, 36, 4) := \]
\[ 1101 \ 1000 \ 1001 \ 0000 \ 0100 \ 0000 \ 1101 \ 0000 \]

\[ V := \{1, 6, -38, -68, -70, 72, -74, -76, -78, 80, -82, -83\} \]
What’s (Obviously) Next?

- Put an End to the Domination of Black Box Optimization.
- Wait for Tonight and Try to Take over the World.
- Questions?