Fitness landscapes and problem hardness in evolutionary computation

S. Periscic and T. Simon
University of Maribor, Slovenia, CRIIS (Center of Research in Informatics and Systems)
https://www.biboulibiki.com/taof
http://www.biboulibiki.com/tarif
July 7, 2007

Copyright is held by the author/owner(s).
ACM 978-1-59593-698-1/07/0007.

Fitness landscape ($S, V, F$):
- $S$: set of potential solutions,
- $V: S \to \mathbb{R}$: fitness function.
\[
V(S) \rightarrow S \\
\forall \in S
\]

\[
V(\in) = \{ \in \in S | \in(\in) \leq \in \}
\]

\[
\rho(s) = \left( \in(\in) - \bar{\in} \right) \left( \in(\in) + \bar{\in} \right)
\]

\[
\tau = \rho(\in) - \alpha(\alpha - \in) + \in
\]

Search space
Fitness

Results on rugged fitness landscapes [25]

Table:

<table>
<thead>
<tr>
<th>Problem</th>
<th>(\in)</th>
<th>(\tau)</th>
</tr>
</thead>
<tbody>
<tr>
<td>permutation TSP</td>
<td>(1 - \frac{1}{3})</td>
<td>(1 - \frac{1}{3})</td>
</tr>
<tr>
<td>permutation TSP</td>
<td>(1 - \frac{1}{3})</td>
<td>(1 - \frac{1}{3})</td>
</tr>
<tr>
<td>NK landscapes</td>
<td>(1 - \frac{1}{3})</td>
<td>(1 - \frac{1}{3})</td>
</tr>
</tbody>
</table>

Multimodal Fitness landscapes

Local optima:
- no neighbor solution with higher fitness value
  - adaptive walk \(\{x, y, \ldots\}\) where \(x, y \in F(\in)\) and \(F(x) < F(\in)\)

\[
\text{Optimization difficulty: number and size of attractive basins (Camill et al. [12])}
\]
Syntactic

• From a syntactic standpoint, the fitness landscape is a function that maps a solution to a fitness value, where the fitness value is a measure of how well the solution performs.

• Syntactic fitness landscapes are often used in the study of evolutionary algorithms, as they provide a way to evaluate the performance of different algorithms.

• In the context of the genetic algorithm, the fitness landscape is a function that maps a chromosome to a fitness value, where the fitness value is a measure of how well the chromosome performs.

• Syntactic fitness landscapes are often used in the study of evolutionary algorithms, as they provide a way to evaluate the performance of different algorithms.

• The fitness landscape is a function that maps a solution to a fitness value, where the fitness value is a measure of how well the solution performs.

• Syntactic fitness landscapes are often used in the study of evolutionary algorithms, as they provide a way to evaluate the performance of different algorithms.

• In the context of the genetic algorithm, the fitness landscape is a function that maps a chromosome to a fitness value, where the fitness value is a measure of how well the chromosome performs.

• Syntactic fitness landscapes are often used in the study of evolutionary algorithms, as they provide a way to evaluate the performance of different algorithms.

• The fitness landscape is a function that maps a solution to a fitness value, where the fitness value is a measure of how well the solution performs.
Three possibilities:
- Decrease the neutrality
- Use specific neutralistic
- Increase the neutrality with redundant genotype-phenotype mapping

Fitness in genotypes space
(F(0), ϕ(0)) = ∑(G, ϕ) ∈ ϕ(G) − ϕ(0) ∈ φ(G, ϕ)

Increasing neutrality (miniA: E. Rodriguez, PPSN'05 [20])

Problem “not well” defined or changing environment (Tomas in [20])
- Permutation problem (sawyers, ... Goldberg ET [18])
Fitness = \sum_{(x, y) \in \text{genotypes space}} \text{Fitness}(x, y)

\text{Reseau de Neutral Portes}:
\text{Problem}\times\text{Problem}\rightarrow\{0, 1\}
\text{Problem}\times\text{Problem} = |\text{Problem}\times\text{Problem} - \text{Neural Degree}|

v_{\text{Neural Degree}}(s) = |v - \text{Neural Degree}(s, s)|

The neural degree of a solution is the number of its neutral neighbors:
\text{Neural Degree}(s) = |v_{\text{Neural Degree}}(s) - |s||

Neutral Networks (Schwanz 2004) [23]

A neural network is neutral if there are many solutions with high neural degree.
Intra network Measures

Classical measures of graph (network) NH:

1. the size: number of nodes of NH
2. degree distribution: measure of the quantity of "neighbors"
3. Autocorrelation of neutral degrees during neutral wall (Bairem)
   - comparison with random graph
   - measure of the correlation structure of NH

Autocorrelation of neutral degrees

Distribution of neutral degrees ($N = 16, \alpha = 24$)

- $M = 10$
- $\alpha = 4$
- $\alpha = 8$
- $\alpha = 16$
- $\alpha = 24$

Experimental distribution (red), Poisson distribution (line).
Reseau de Neutralite

Performance

Marche neutre aleatoire

S0

S1

S2

S3

\( \ldots \)

\[ \rho(s) = 0 \]

\[ \rho(s) = 0.1 \]

\[ \rho(s) = 0.2 \]

\[ \rho(s) = 0.3 \]

\[ \rho(s) = 0.4 \]

\[ \rho(s) = 0.5 \]

\[ \rho(s) = 0.6 \]

\[ \rho(s) = 0.7 \]

\[ \rho(s) = 0.8 \]

\[ \rho(s) = 0.9 \]

\[ \rho(s) = 0.95 \]

\[ \rho(s) = 0.99 \]

Parametre q

\( M = 16 \)

\( M = 32 \)

\( M = 48 \)

Parametre M

\( K = 2 \)

\( K = 4 \)

\( K = 8 \)

\( K = 12 \)

\( K = 16 \)

Parametre p

\( p = 0.8 \)

\( p = 0.9 \)

\( p = 0.95 \)

\( p = 0.99 \)

Syntese

Mise en pages

- réseaux de distribution
- "How each connectivity in the landscape?"
- "Autocorrelation of network structure"
- "correlation of log-log log"
Neutrality in Majority Problem

Standard performance: error of evaluation due to random variation of sample 1/2. 

- If chosen independently, a random value of follows a normal distribution.

Definition

- Given a CA of size \(P \times P\) and rule \(r\), let \(c_P\) be the fraction of all initial configurations that reach the state \(\langle 0 \rangle\) after \(n\) time steps. 
- If \(c_P = 1\), the CA exhibits deterministic behavior. 
- If \(c_P < 1\), the CA exhibits stochastic behavior. 
- The standard deviation of the fraction of correct classifications over \(n = 10^4\) randomly chosen inputs is given by:

\[
\sigma = \sqrt{\frac{c_P(1-c_P)}{n}}
\]
Density Of States

**Sampling** vs **Random** vs **Metropolis Hastings**

- Metropolis Hastings sampling:
  - A significant number of solutions sampled with a fitness of 0.05
  - No solutions with a fitness value superior to 0.05

**Study of Neutral Networks**

- Study of two important large neutral networks:
  - **NN(1)**: fitness around 0.05
    - Automatic selection on only half of IC's.
    - 5 neutral nodes.
  - **NN(5)**: fitness around 0.765
    - 10 neutral nodes.

**Neutral Networks**:

- Same starting pulse on each NW
- Slightly increasing the Hamming distance from the starting solution.
- Stopped when there is no neutral step that increases distance.

**Statistical Measures of Neutral Networks**

- **Density Of State (ODS)** [H. Heikkinen et al.]
  - Diameter of NN:
    - Diameter of the graph

- **Neutral Degree** distribution:
  - Degree distribution of the vertices

- **Auto-correlation of neutral degree along a neutral random walk** [Heikkinen et al.]

- **Graph of fitness vs. neutral degree**
Definition of Olympic Landscape

Two symmetries that do not change performance:
- 0: symmetry in and right $AX^2$ symmetry.

Symmetry of $\text{Ogk}$ which minimizes the number of joint hits:
- $\text{GGL} = \text{GGL}$
- $\text{Dq} = \text{Dq}$
- $\text{Dwiv} = \text{Dwiv}$
- $\text{Dg} = \text{Dg}$
- $\text{APK} = \text{APK}$
- $\text{Cop} = \text{Cop}$
- $\text{Coe} = \text{Coe}$

Olympic Landscape, subspace of dimension 77:
Neutral Degree

- Two large neutral networks at fitness 0 and 0.05:
  - Neutral degree > 75.
- Over 80% of:
  - Average neutral degree is 87.6.

Main Results on Fitness Landscape

- Considerable number of Gs of performance for 0.5
- High neutrality of the landscape
- Neutral networks are not random graphs
- Fitness landscape of many problems has a high neutral degree
- Defined the high fitness landscapes:
  - Explaining similarities between the three best methods
  - Low solution with performance 0.5
  - Easy to find solution with 0.86 with a simple GA
  - Over-performance 0.5: neutrality of landscape is still high

Density Of States

Null Transition:
- Sampling: 500
- Uniform initial state: 50%
- Advantage of concentrating the search in the fitness landscape:

Neutral Degree: Sampling Method

- Solutions < 0.05 randomly chosen in landscape
- Solutions = 0.5: plus 0 runs of a GA during 100 generations

AG and:
- Based on GA defined by Mitchell
- Operators restricted to Olimpus subspace
- Solution: tournament selection allowing time account the neutrality
- Justified the bounds of Olimpus neutrality:
  - Discover a list of solutions between 0.00 and 0.0015
  - Over 50 runs, average performance 0.5545
  - (comparision: AIC of Olimpus = 0.8475)
Exploring phenotypic space through mutual evolution.
Difficulty of problems in GP

- The ability of GP to find *optimal* solutions.

By analogy with GA hardness theory:

- **Performance**: the proportion of executions for which the *global optimum* has been found in less than 500 generations over 100 runs.

Fitness Distance Correlation (*fdc*) [*T. Jones, 1995*]

Given a sample of *n* individuals, let’s suppose to know:

- the set $F = \{f_1, f_2, \ldots, f_n\}$ of the individual fitnesses
- the genotype of the global optimum (individual with the best fitness)
- a measure to express the genotypic distance between individuals

Let $D = \{d_1, d_2, \ldots, d_n\}$ be the *n* distances to the global optimum, then

*$fdc$ is the correlation between sets $F$ and $D$*

Main idea

- Notion of **distance**.
- Relationship between fitness and distance to the goal.

Fitness Landscapes and Problem Difficulty in Genetic Programming

Leonardo Vanneschi

Dipartimento di Informatica, Sistemistica e Comunicazione (DISCo)
University of Milano-Bicocca, Italy

*Audentes fortuna iuvat*
A language to code individuals

Inspired by B. Punch (The Royal Tree Problem)

\[ F = \{ A, B, C, D, \ldots \} \]
\[ T = \{ X \} \]

where:
A is a function of arity 1
B is a function of arity 2
C is a function of arity 3, ....
e, X is a terminal symbol

Examples

\[
\begin{align*}
A & \rightarrow X \\
B & \rightarrow X A \\
C & \rightarrow B X \\
D & \rightarrow B X A \\
E & \rightarrow B X A
\end{align*}
\]

1. Tree Distance (first attempt)

A distance based on depth and recursion has been defined. It has been called Depth Recursive Distance (drd).

\[ \text{drd}(T_1, T_2) : \]
- If the root of \( T_1 \) is different from the root of \( T_2 \), the distance is proportional to the difference of their depths.
- If the root of \( T_1 \) is equal to the root of \( T_2 \) and the depth of \( T_1 \) or the depth of \( T_2 \) are different from zero, then \( \text{drd} \) is applied recursively to the children of the root of the two trees, and all these results are summed.
- If the root of \( T_1 \) is equal to the root of \( T_2 \) and the depth of \( T_1 \) and the depth of \( T_2 \) are both equal to zero, then the result returned is zero.

\[ \text{fdc} \] as tool for problem hardness [T. Jones, 1995]

For GAs, problems can be classified in three classes:
- **Misleading** \((\text{fdc} \geq 0.15)\) in which fitness increases with distance.
- **Difficult** \((-0.15 < \text{fdc} < 0.15)\) in which there is no correlation between fitness and distance.
- **Straightforward** \((\text{fdc} \leq -0.15)\) in which fitness increases as the global optimum approaches.

To (experimentally) verify if the same property is also valid for GP:

First step: to choose a distance between genotypes (trees!)

Our approach

- To chose a distance between genotypes to calculate \( \text{fdc} \)
- To define some genetic operators consistent with this distance
- To test \( \text{fdc} \) on a set of "synthetic" functions:
  - Trap Functions (unimodal et multimodal) \((\text{Deb, Goldberg})\)
  - Royal Trees \((\text{B. Punch})\)
  - MAX Functions \((\text{Gathercole})\)
The \( k \) constant

- \( k \) is an arbitrary positive constant
- two trees with different roots but with the same depth must have a positive distance
- it must never become negative with the recursive calls

\[
\text{it must be at least as large as the maximum depth allowed for the trees}
\]

Genotypic Distance between Trees
A first attempt: Depth Recursive Distance (\( \text{drd} \))

Let \( T_1 \) and \( T_2 \) be two trees.

\[
drd(T_1, T_2, k) = \begin{cases} 
    k + | td(T_1) - td(T_2) | & \text{if } \text{root}(T_1) \neq \text{root}(T_2) \\
    0 & \text{if } td(T_1) = td(T_2) = 0 \\
    \sum_{i=1}^{n(T_1)} \frac{drd(s_i(T_1), s_i(T_2), k-1)}{n(T_1)} & \text{otherwise}
  \end{cases}
\]

Unimodal Trap Functions

\[
f : \text{distance} \rightarrow \text{fitness}
\]

\[
f(d) = \begin{cases} 
    1 - \frac{d}{B} & \text{if } d \leq B \\
    \frac{R(d - B)}{1 - B} & \text{otherwise}
  \end{cases}
\]

where:
- \( d \) is the distance between the current individual and the optimum
- \( B \) et \( R \) are constants \( \in [0, 1] \)

\[
B = 0.2, \ R = 0.8
\]

The difficulty of trap functions depends on \( B \) and \( R \)

Example of application of distance \( \text{drd} \)

Let:

\[
T_1 = \begin{array}{c}
A \\
X
\end{array}, \quad T_2 = \begin{array}{c}
B \\
X \\
X
\end{array}
\]

Since they have the same root:

\[
drd(T_1, T_2, k) = \frac{A}{2} + \frac{X}{2} = \frac{k}{2} + \frac{1 - 0}{2}
\]

\[
= k + \frac{1}{2}
\]

\[
= \frac{k}{2} + \frac{1}{2}
\]

The difficulty of trap functions depends on \( B \) and \( R \)
Royal Trees: definition

The language is the same, but we change the way of calculating fitness:

- The fitness of a tree is the score of its root
- The score of each node is the sum of the weighted scores of his sons
- If a son is a perfect tree of the appropriate level (i.e. for instance a perfect tree of root C under a node D) the scores of its subtrees, multiplied by a \textit{FullBonus}, is added to the score of the root.
- If the son has the correct root, but it isn’t a perfect tree, then the weight is \textit{PartialBonus}.
- If the root of the subtree is incorrect, then the weight is \textit{Penalty}.
- After having calculated the score of the root, if the tree whole is a perfect tree, the score is multiplied by a \textit{CompleteBonus}.

In our experiments we have used the following values:
\textit{FullBonus} = 2, \textit{PartialBonus} = 1, \textit{Penalty} = 0.0001, \textit{CompleteBonus} = 2

Royal Trees: an example

Let \( T = \)

We fix:
\[ \text{fitness}(X) = 1 \]

Then we have:
\[ \text{fitness}(\begin{array}{c} A \\ X \end{array}) = (1 \ast \textit{FullBonus}) \ast \textit{CompleteBonus} = (1 \ast 2) \ast 2 = 4 \]
\[ \text{fitness}(\begin{array}{c} B \\ X \\ X \end{array}) = (4 \ast \textit{FullBonus}) + (4 \ast \textit{FullBonus}) \ast \textit{CompleteBonus} = ((4 \ast \textit{Penalty}) + (4 \ast \textit{Penalty})) \ast 2 = 16 \ast 2 = 32 \]

Finally:
\[ \text{fitness}(T) = (32 \ast \textit{FullBonus}) + (1 \ast \textit{Penalty}) + (1 \ast \textit{Penalty}) \]
Results

<table>
<thead>
<tr>
<th>Racine</th>
<th>fdc</th>
<th>prediction</th>
<th>$p$</th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>-0.45</td>
<td>straightf.</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>-0.33</td>
<td>straightf.</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>-0.26</td>
<td>straightf.</td>
<td>0.77</td>
<td>0.81</td>
<td>0.81</td>
<td>0.81</td>
</tr>
<tr>
<td>E</td>
<td>-0.22</td>
<td>straightf.</td>
<td>0.42</td>
<td>0.62</td>
<td>0.74</td>
<td>0.79</td>
</tr>
<tr>
<td>F</td>
<td>0.035</td>
<td>unknown</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0.26</td>
<td>misleading</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$p$ (respectively $p_1$, $p_2$, $p_3$) = number of runs in which the optimum has been found in less than 200 (respectively 300, 400, 500) generations divided by the total number of runs (100)

Structural Distance (formal definition)

\[ \text{dist}(T_1, T_2) = d(R_1, R_2) + k \sum_{i=1}^{m} \text{dist}(\text{child}_i(R_1), \text{child}_i(R_2)) \]

where:
- $R_1$ is the root of $T_1$ and $R_2$ is the root of $T_2$.
- $d(R_1, R_2) = (|c(R_1) - c(R_2)|)^z$, where $c(X)$ is the weight of node $X$, and $z \in \mathbb{N}$
- $\text{child}_i(Y)$ is the $i$th of the $m$ possible sons of node $Y$ if $i \leq m$, or the empty tree otherwise.
- $k \in \mathbb{N}$

Structural Distance (Intuition)

[Ekàrt-Németh 2002]

- We assign a weight to each node
- We calculate the difference of the weights of nodes at corresponding positions
- The distance is the weighted sum of these differences

Next Step
1. To choose another distance between genotypes to calculate $fdc$
2. To define some genetic operators consistent with this distance

$fdc$ correctly predicts the difficulty of Trap Functions and Royal Trees, but:

- No relationship has been observed between distance measure ($drd$) and genetic operator (standard GP crossover [Koza, 1992])
  - "In order for $fdc$ to be more effective, the distance metric should be defined with regard to the actual neighborhood produced by the genetic operators, so to assure the conservation of the genetic material between neighbors" [Jones, 1995]
  - "...either Hamming distance is connected to the way GAs work, or this relation exists in a fortuitous way among the test set chosen by Jones" [Altenberg, 1997]
Parameters used in the experiments

- Population size = 200 individuals
- Ramped Half and a Half initialization (Koza)
- Tournament selection (tournament size = 10 individuals)
- Node with maximum arity = F
- If we use standard GP:
  - Crossover rate: 95%
  - Mutation rate: 0%
- If we use SMGP:
  - Crossover rate: 0%
  - Mutation rate: 95%
- Stop criterion: either the optimum has been found, or 500 generations have been executed.
- 100 independent runs.
- For the calculus of fdc: sampling of 40000 individuals without repetitions.

Operators of Structural Mutation

- Inflate Mutation
- Deflate Mutation

GP based only on these operators: Structural Mutation Genetic Programming (SMGP).

Results (trap functions 1)

Property (Distance/Operators Consistency)

Let:

- $F = \{A, B, C, \ldots\}$ $T = \{X\}$
- $\forall s \in \{F \cup T\} : c(s) = \text{arity}(s) + 1$
- $T_1$ et $T_2$ two trees composed by symbols $\in \{F \cup T\}$
- $k = 1, z = 1$

If $\text{dist} (T_1, T_2) = D$ then $T_2$ can be obtained from $T_1$ with a sequence of $D/2$ operations of structural mutation.
"W" Trap Functions (Multimodals)

- We chose a tree $T_0$ (origin);
- We set $R_0 = 1$;
- Between the trees with distance $= B_2$ to $T_0$ we chose one tree ($T_1$);
- $\forall T$ s.t. $\text{dist}(T, T_0) = B_2$ et $T \neq T_1$, fitness($T$) = rand($) \in [0, 1)$;

(only two global optima)

- To calculate fdc, $\forall T \in$ sampling, we consider $\min(\text{dist}(T, T_0), \text{dist}(T, T_1))$
Royal Trees Results

<table>
<thead>
<tr>
<th>root</th>
<th>fdc</th>
<th>fdc prediction</th>
<th>p (SMGP)</th>
<th>p (stGP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>-0.31</td>
<td>straightf.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>-0.25</td>
<td>straightf.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>-0.20</td>
<td>straightf.</td>
<td>0.76</td>
<td>0.70</td>
</tr>
<tr>
<td>E</td>
<td>0.059</td>
<td>unknown</td>
<td>0</td>
<td>0.12</td>
</tr>
<tr>
<td>F</td>
<td>0.44</td>
<td>misleading</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0.73</td>
<td>misleading</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Results ("W" Trap Functions 1)

To T1
dist (To, T1) = 0.5 (normalised)

Results ("W" Trap Functions 2)

To T1
dist (To, T1) = 0.1 (normalised)

"MAX" Functions

F = {+}, T = {1}
c(1) = 1, c(+)= 2

Inflate mutation
1
Deflate mutation
+(1,1)

F = {+}, T = {1, 2}
c(1) = 1, c(2) = 2, c(+) = 3

Inflate mutation
1
Deflate mutation
2

Inflate mutation
2
Deflate mutation
+(2,1)

Inflate mutation
1
Deflate mutation
2

Inflate mutation
2
Deflate mutation
+(1,2)

Results ("W" Trap Functions 2)
Counterexample
Ridge Functions (inspired by Quick et al. 1998)

If we call "F" the fitness of the optimal Royal Tree, then:

\[
\text{fitness}(T) := \\
\begin{align*}
\text{begin} & \\
\text{if } (T \text{ contains only } A \text{ and } X \text{ nodes}) & \\
\text{then} & \\
\text{return } (F \times \text{depth}(T)) & \\
\text{else} & \\
\text{return } (\text{fitnessRoyalTree}(T)) & \\
\text{endif} \\
\text{end}
\end{align*}
\]

"MAX" Functions Results

<table>
<thead>
<tr>
<th>max problem</th>
<th>fdc</th>
<th>fdc prediction</th>
<th>p (SMGP)</th>
<th>p (stGP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(+) {1}</td>
<td>- 0.87</td>
<td>straightf.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(+) {1,2}</td>
<td>- 0.86</td>
<td>straightf.</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Ridge Function (intuition)

- The optimal Royal Tree is a local optimum
- The global optimum is the tree composed only by the symbols A and X and with maximum depth
- We have defined a "path"
  \[
  \{X, \hat{X}, \hat{X}, \hat{X}, \ldots\}
  \]
- Each individual belonging to the path has a fitness greater or equal to the optimal Royal Tree.
- Each individual not belonging to the path has a fitness smaller or equal to the optimal Royal Tree.

fdc works for all the test functions used
Are we happy?
**Ridge Function** (intuition 2)

- **Local Optimum**
- **Global Optimum**

**Ridge Function Results**

- Maximum depth for the trees = 10
- \( fdc = 0.88 \)
- Over 100 runs, the optimum has been found 100 times !!! (performance = 1)

---

**fitdc drawbacks**

- Existence of counterexamples
  - Here: Ridged Royal Trees
    (inspired by the counterexample for GAs of [Quick et al., 1998])
- Not a predictive measure
  - Optima must be known “a priori”
    (this drawback makes fitdc “almost” unusable in practical cases)

A new measure is needed to quantify the difficulty of "real" problems.

The measure proposed in this thesis is based on the concept of fitness clouds.

**Fitness Clouds**

**fitness clouds** have been first introduced for binary landscapes in [Vérel, et al. 2003]

For each individual \( \gamma \) (in a sample or in the whole search space) a point is plotted:

- **abscissa** = fitness value of \( \gamma \)
- **ordinate** = fitness value of a “particular” neighbor (chosen randomly or by some particular technique).
  - here: neighbor = individual obtained by applying one step of mutation to \( \gamma \)

Fitness of Neighbors

- **Fitness Clouds**
  - For each individual \( \gamma \) a point is plotted:
    - **abscissa** = fitness value of \( \gamma \)
    - **ordinate** = fitness value of a “particular” neighbor (chosen randomly or by some particular technique).
  - here: neighbor = individual obtained by applying one step of mutation to \( \gamma \)
Utility of Fitness Clouds

They give a visual rendering of evolvability.

Evolvability

"The ability of an operator/representation scheme to produce offspring that are fitter than their parents"

Evolvability is related to (although not identical to) problem difficulty.

1. Sampling the search space

- Individuals with bad fitness are likely to be discarded by selection
- It is on good individuals that evolvability has to be calculated

thus, we have used an importance sampling (more weight to good points)

a sample of GP individuals \{y_1, y_2, \ldots, y_n\} can be calculated by the algorithm:

\begin{verbatim}
begin
    y is generated uniformly at random;
    for \( k = 2 \) to \( n \) do
        1. an individual \( \delta \) is generated uniformly at random;
        2. a random number \( s \) is generated from a uniform (0,1) distribution;
        3. if \( s < \alpha(y_i, y_\delta) \) then \( y_i = \delta \) else goto 1.
    endfor
end
\end{verbatim}

where:

\[ \alpha(x,y) = \min\{1, \frac{y}{x}\} \]

Metropolis-Hastings

2. Sampling the neighborhood

Offspring with bad fitness are probably discarded by selection

thus, the selection algorithm itself can be employed to choose which neighbors are to be used

In our experiments, we have used tournament selection of size 10, i.e.

for each neighbor that has to be chosen:

- 10 random neighbors are generated
- the best is chosen

Sampling the search space and the neighborhoods

Main idea:
Evolvability makes sense if it is calculated on "good" individuals ("bad" ones are probably discarded by selection).

Sampling the search space:
Importance sampling (Metropolis-Hastings technique)

Sampling the neighborhoods:
selection (tournament selection of size 10).
**Performance**

The fraction of executions for which a global optimum has been found before generation 500 over 100 runs.

**Parameters**

**Fitness clouds**
- 40000 points with Metropolis-Hastings
- **Standard subtree mutation** to generate neighborhoods
- For each sampled individual, 1 neighbor chosen by means of ** Tournament selection of size 10**

**GP runs**
- Generational GP
- Population size = 200 individuals
- Ramped half-and-half initialization
- ** Tournament selection of size 10**
- Standard subtree mutation rate = 95%
- No crossover

**Binomial-3 Problem [Daida et al. 2001]**

Goal: to find a program which matches the following function

\[ f(x) = 1 + 3x + 3x^2 + x^3 \]

- **Fitness cases** (values for \( x \)) = 50 equidistant points over the range [-1, 0)
- **Function set** = \{+, -, *, /\}
  - where / is the protected division (it returns 1 if the denominator is 0)
- **Terminal set** = \( \{x, R\} \)
  - where \( R \) is a set of Ephemeral Random Constants (ERCs) uniformly distributed over the range \([-a_R, a_R]\), where \( a_R \) is a positive integer constant

Daida et al. have shown that the difficulty of this problem increases as \( a_R \) increases.

**Measure of Problem Hardness Based on Fitness Clouds**

**Negative Slope Coefficient (nsc)**

- The scatterplot is partitioned into \( n \) bins
- For each bin, a point is calculated, such that its abscissa is the average of the abscissas and its ordinate is the average of the ordinates.
- All these points are joined by segments \( \{S_1, S_2, ..., S_{n-1}\} \)

**Negative Slope Coefficient Definition**

\[
\text{nsc} = \sum_{i=1}^{n-1} p_i
\]

where, \( \forall \ i \in [1, n-1] \)

\( p_i = \min \{0, \text{slope}(S)\} \)

**Hypothesis:**

- \( \text{nsc} = 0 \) \( \implies \) the problem is **easy**
- \( \text{nsc} < 0 \) \( \implies \) the problem is **difficult** and the magnitude of \( \text{nsc} \) quantifies the difficulty

**Idea:**

If \( \text{nsc} < 0 \) then there is at least one area of the fitness landscape where evolvability is bad.
Even Parity Problem [Koza, 1992]

- Primitive sets used: \( F = \{\text{NAND, NOR}\}, \ T = \{x_1, x_2, \ldots, x_k\}\) where \(k\) is the rank of the problem (even parity \(k\) problem)
- The difficulty of the problem can be changed by varying \(k\) (even parity 5 more difficult than even parity 3...)

<table>
<thead>
<tr>
<th>even parity 3</th>
<th>even parity 5</th>
<th>even parity 7</th>
<th>even parity 9</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>problem</th>
<th>(p)</th>
<th>(nsc)</th>
<th>(ffc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>even parity 3</td>
<td>0.98</td>
<td>0.57</td>
<td></td>
</tr>
<tr>
<td>even parity 5</td>
<td>0.01</td>
<td>-0.53</td>
<td>0.43</td>
</tr>
<tr>
<td>even parity 7</td>
<td>0</td>
<td>-0.42</td>
<td>0.30</td>
</tr>
<tr>
<td>even parity 9</td>
<td>0</td>
<td>-1.08</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Binomial-3 Problem (experimental results 1)

- \(nsc\) gets more negative as the problem gets more difficult
- \(nsc\) is zero for the “easy” instance of the problem (\(a_R = 1\))
- For comparison: \(ffc\) is the fitness-fitness correlation [Manderick et al., 1991] clearly it does not give any indication about problem difficulty

Even Parity 9 - Fitness Distribution

- The shape of the scatterplot does not depend on the choice of the Metropolis-Hastings sampling
- These results are coherent with the ones contained in [Langdon and Poli, Foundations of Genetic Programming, 2002]

Binomial-3 Problem (experimental results 2)

<table>
<thead>
<tr>
<th>(a_R)</th>
<th>(p)</th>
<th>(nsc)</th>
<th>(ffc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.89</td>
<td>0</td>
<td>0.68</td>
</tr>
<tr>
<td>10</td>
<td>0.42</td>
<td>-0.15</td>
<td>0.64</td>
</tr>
<tr>
<td>100</td>
<td>0.35</td>
<td>-0.98</td>
<td>0.65</td>
</tr>
<tr>
<td>1000</td>
<td>0.29</td>
<td>-2.03</td>
<td>0.65</td>
</tr>
</tbody>
</table>
Crossover Distance

Artificial Ant on the Santa Fe Trail [Koza, 1992]

- Difficulty tuning is obtained by changing the maximum tree depth
difficulty decreases as individuals' size increases [Langdon and Poli,
Foundations of Genetic Programming, 2002]

<table>
<thead>
<tr>
<th>max. tree depth</th>
<th>p</th>
<th>nsc</th>
<th>f/c</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.05</td>
<td>-3.21</td>
<td>-0.86</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>-9.84</td>
<td>-0.84</td>
</tr>
</tbody>
</table>

Summary of the results obtained with the nsc

- Good hardness indicator for:
  - Trap Functions
  - Royal Trees
  - Binomial-3 Problem [Daida et al., 2001]
  - Even Parity Problem [Koza, 1992]
  - Artificial Ant on the Santa Fe Trail [Koza, 1992]
- Many ways of calculating the nsc have been used:
  - Number of neighbors for each sampled individual
  - Number of mutations to generate neighbors
  - Different types of mutations to generate neighbors
  - Different techniques to partition the fitness clouds into bins
- nsc is predictive \( \Rightarrow \) it can be used on any problem
- nsc has not been normalized yet into a given range
  (classification of different problems by their difficulty)

Other Operator Based Distances

For instance: Structural Mutation is consistent with Structural Distance\(^{(1)}\)

Formal Proof in:
[L. Vanneschi et al., Fitness Distance Correlation in Structural Mutation Genetic Programming, EUROGP 2003]

\(^{(1)}\) A. Ekárt, S. Z. Németh, Maintaining the Diversity of Genetic Programs, EUROGP 2002

Subtree Crossover Distance is more difficult

- subtree crossover is binary,
- subtree crossover is "destructive",
- trees are dynamic structures, etc.
Why a Distance Based on Subtree Crossover?

- In Genetic Programming (GP), genetic operators define a neighborhood structure (topological space) over trees.
- To analyse various dynamics of GP (diversity, etc.), it is often useful to quantify the distance between two trees in this topological space.
- Subtree crossover is the most common genetic operator in GP → subtree crossover distance.

A difficult task:

- subtree crossover is binary,
- subtree crossover is "destructive",
- trees are dynamic structures, etc.

What do we mean by Distance Based on Subtree Crossover?

"Topological crossovers [...] are well defined once a notion of distance over the solution set is defined. Simply stated, the offspring they produce are between their parents" [A. Moraglio, R. Poli, Topological Crossover for the Permutation Representation, GSICE 2005]

In this paper, we take up a different point view:

the distance of a solution $T_1$ to a solution $T_2$ should not express "betweenness" of $T_1$ and $T_2$, but "how easy" (or "how difficult") it is to transform $T_1$ into $T_2$ (by means of a genetic operator, crossover in our case).

(If it is "easy" to transform $T_1$ into $T_2$, the distance between $T_1$ and $T_2$ must be "small". Otherwise it must be "large").
Leonardo Vanneschi - University of Milano-Bicocca, Italy

The "New" SCD Definition

```
func SCD(T1, T2, P){
    S = diff(T1, T2)
    res = 1
    ∀(sT1, sT2) ∈ S:
    ps1 = probSelecting(sT1, T1)
    ps2 = probCreating(sT2, P)
    res = res * (1 - ps1 * ps2)
    return res
}
```

The new operator \( \text{diff}(T_1, T_2) \) returns the set

\[
S = \{ (s_{T1}^1, s_{T2}^1), (s_{T1}^2, s_{T2}^2), \ldots, (s_{T1}^n, s_{T2}^n) \}
\]

such that:

if we replace \( s_{T1}^i \) with \( s_{T2}^i \), \( \forall i \in [1, n] \) in \( T_2 \), we obtain \( T_1 \).

the complexity is still "reasonable"!

Leonardo Vanneschi - University of Milano-Bicocca, Italy

Advantage of the "old" SCD definition

In [S. Gustafson, L. Vanneschi, Operator based distance for Genetic Programming: Subtree Crossover Distance, EUROGP 2005] we have shown that the complexity of SCD is "reasonable".

Theory meets Practice: Is The "New" SCD Definition Useful?

We have done some experiments:

• Fitness Distance Correlation (FDC)
• Fitness Sharing
• Diversity

Test Problems:

• Syntactic Trees
• Trap Functions

Drawback of the "old" SCD definition

If \( T_1 \) and \( T_2 \) differed in more than one subtree, we approximated the distance between them by setting it equal to one (very large).

In some cases, this approximation may be too coarse, thus compromising the accuracy of the measure (for instance when using the SCD for calculating the FDC).

Leonardo Vanneschi - University of Milano-Bicocca, Italy

\[
\text{func distance}(T_1, T_2, V, P)\
\begin{align*}
    & (s_{T1}, s_{T2}) = \frac{T_1}{T_2} \\
    & ps1 = \text{probSelecting}(s_{T1}, T_1) \\
    & ps2 = \text{probCreating}(s_{T2}, P) \\
    & \text{return } (1 - ps1 \times ps2)
\end{align*}
\]

where: \( T_1 / T_2 \) is the "difference" between \( T_1 \) and \( T_2 \).

This operator returns a pair of subtrees \( (s_{T1}, s_{T2}) \) such that \( s_{T2} \in T_2 \) must replace \( s_{T1} \in T_1 \) to make \( T_1 = T_2 \).
Fitness-Distance Correlation (Definition) (Jones, 1995)

Let \( \{i_1, i_2, \ldots, i_n\} \) be a set of \( n \) sampled individuals, 
\[ F = \{f_1, f_2, \ldots, f_n\} \]
their \( n \) fitness values and 
\[ D = \{d_1, d_2, \ldots, d_n\} \]
the \( n \) distances of each individual to a global optimum, we define
\[ \text{FDC} = \text{statistical correlation between sets } F \text{ and } D. \]

According to Jones:
- Positive FDC \( \rightarrow \) very hard problem
- FDC approximately equal to zero \( \rightarrow \) hard problem
- Negative FDC \( \rightarrow \) easy problem

We have studied FDC for GP on large samples of individuals. Here we study it \textit{dynamically} (how it changes \textit{during the evolution}) on the individuals in the population.

Experimental Results (FDC - Syntactic Trees)

Successful Runs

Optimal Tree =

\[ \text{FDC is negative and decreases during evolution} \]

(standard deviations on the paper)

Syntactic Trees (Gustafson, Vanneschi, 2005)

Trees are represented using the sets:
\[ F = \{N\}, \quad T = \{L\} \]

where
- \( N \) is a binary operator (\( N \) stands for “Non-terminal”)
- \( L \) stands for “Leaf”.

No “content” is associated with the nodes and fitness is equal to the \textit{edit distance} (ED) to a fixed global optimum.

Trap Functions (Deb, Goldberg, 1993)

\[ F = \{A, B, C, \ldots\}, \quad T = \{X\} \]
\[ \text{arity}(A) = 1, \quad \text{arity}(B) = 2, \quad \text{arity}(C) = 3, \ldots \]

\[ f : \text{distance} \rightarrow \text{fitness} \]
\[ b = 0.2, \quad r = 0.8 \]

\textit{The difficulty of trap functions depends on } \( b \text{ and } r \)

where:
- \( d \) is the distance between the current individual and the optimum
- \( b \text{ et } r \) are constants \( \in [0, 1] \)
Experimental Results (FDC - Royal Trees)

Unsuccessful Runs

Optimal Tree =

Unsuccessful Runs

Optimal Tree =

Experimental Results (FDC - Syntactic Trees)

Unsuccessful Runs

Optimal Tree =

Experimental Results (FDC - Royal Trees)

Successful Runs

Optimal Tree =

FDC Results - Discussion

For **successful runs**, FDC is negative → as individuals get **better** (in fitness) they also get **closer** (in terms of SCD) to the global optimum (**until it is reached by GP**).

For **unsuccessful runs**, FDC is positive or approximately equal to zero → as individuals get **better** (in fitness) they **don’t get closer** (in terms of SCD) to the global optimum (and thus GP does not reach it!).

FDC + SCD = **good indicator for GP dynamics**!

SCD = **a good distance to calculate the FDC**!
Fitness Sharing Results - Discussion

GP systems with fitness sharing using SCD "perform better" than standard GP and fitness sharing systems using ED (for the problems studied here!)

Because SCD is consistent with crossover (and thus appropriately models its behavior)?

Fitness Sharing (Goldberg, 1991)

\[ f_s(i) = \sum_{j \in P \setminus j \neq i} \frac{f(i)}{s(d(i,j))} \]

\[ s(x) = 1 - x \]

Test Problem: Syntactic Trees

Global Optima: randomly generated trees (with the grow method) with a maximum depth equal to 8

Diversity

Standard deviations of the distances of the individuals in the population from a pre-fixed tree (= "origin").

Test Problem: Syntactic Trees

Global optima = origin = randomly generated trees (with the grow initialisation method) with a maximum depth equal to 8

Experimental Results (Fitness Sharing)

Fitness sharing systems with SCD perform better than fitness sharing with ED and standard GP
Conclusions

SCD appropriate for:
- Measuring the FDC dynamically (during evolution)
- Fitness Sharing

Our hypothesis: SCD appropriately models subtree crossover

• SCD diversity behave differently than ED diversity (slightly increasing and larger than zero for successful runs, approximately zero for unsuccessful runs)

Can it be used to predict the behavior of GP runs?

Experimental Results (Diversity)

Successful Runs

Unsuccessful Runs

It grows!

"more or less" constant and "small"!

Diversity Results - Discussion

For the problems studied here, the population is, most of the time, composed by individuals which are far (in the sense of SCD) to the global optimum.

For successful runs, SCD diversity grows up → some individuals which can easily be transformed into the optimum are generated.

For unsuccessful runs, SCD remains very small → the population is always composed by individuals which are far to the optimum. This thing does not change during evolution!

SCD distance can be used to predict GP behavior?

Neutrality
What is "new" in this work?

- We study even parity fitness landscapes for standard tree-based GP (instead of Cartesian GP or linear GP).
- We don't artificially introduce neutrality into the landscapes (in order to study its effects on GP performance), but we study them "as they are".
- We introduce some new neutrality measures
- We introduce new genetic operators to define neighborhood between trees.
- We don't use a fixed set of functions to build solutions, but we compare the landscapes induced by two different sets of boolean operators: \{NAND\} and \{XOR, NOT\}
- We define a new sampling methodology

Goals of this study

1. Defining some indicators of the difficulty of GP even parity landscapes based on the concept of neutrality
   
   no "stand-alone" infallible hardness measures, but each one of them should give us the picture of one characteristic of the fitness landscape and all together they should give us a rather complete picture that should help us to draw some conclusions about problem hardness

2. Defining a new sampling strategy for GP in general (and particularly suitable for boolean landscapes).
   
   Sampling a fitness landscape is an important task independently from the hardness indicator we want to employ: if we don't sample the important parts of the landscape or if we don't give a sufficiently faithful picture of it, we will fail!

Our Approach

1. Defining some neutrality measures
2. Using them to characterize two "small" landscapes (which we are able to generate exhaustively)
   
   Partial goal: to test if the measures defined in point 1 are suitable indicators of fitness landscapes difficulty

3. Presenting a new sampling technique
4. Applying the same measures as in point 1 to two "large" landscapes, but "similar" to the ones studied at point 2 (which we study by means of the sampling technique presented in point 3).
   
   Partial goal: to test if the sampling technique presented in point 3 gives a faithful "picture" of the landscape (at least from the point of view of neutrality) and to have a confirmation of the fact that the measures defined in point 1 are suitable indicators of fitness landscapes difficulty

GP and Neutrality - State of The Art

- [Yu, Miller, 2001-2002]: artificially adding neutrality can help Cartesian GP to navigate some fitness landscapes.
- [Collins, 2005]: criticism towards the previous results
- Other contributions: [Reidys, Stadler, 2001], [Toussaint, Igel, 2002], [Geard, 2002], [Collard et al., 1999], ...: neutrality plays an important role in EAs.

GP Boolean Landscapes - State of The Art

- [Langdon, Poli, 2001]
- [Koza, 1992]
- [Collins, 2004]
- ...
2. **Strict inflate mutation**: transforms leaf into a tree of depth 1, rooted in a random operator and whose children are a random list of variables containing also the original leaf in a random position.

![Example](NAND Y X \[\rightarrow\] NAND X Y)

3. **Point Terminal Mutation**: replaces a leaf with another random leaf.

### Some Definitions

- **Fitness Landscape**: $L = (S, V, f)$, where
  - $S$ is the search space,
  - $V: S \rightarrow 2^S$ is a neighborhood function specifying for each $s \in S$ the set of its neighbors $V(s)$,
  - $f$ is the fitness function.

- **Neutral neighborhood**: $V_N(s) = \{s' \in V(s) | f(s') = f(s)\}$

- **Neural Network** [Schuster et al., 1994]: a graph connected component $(S, E)$, where: $E = \{(s, s') \in S^2 | s' \in V_N(s)\}$

### Genetic operators that induce the neighborhood

Inspired by the structural mutation operators defined in [Vanneschi et al., 2003]

1. **Strict deflate mutation**: transforms a subtree of depth 1 into a randomly selected leaf chosen among its children.

![Example](NAND Y X \[\rightarrow\] NAND X Y)
"Small" Landscapes

- Even Parity 2 Problem
- Maximum Tree Depth = 3
- Sets of Operators: {NAND} and {XOR, NOT}

Some statistics:

<table>
<thead>
<tr>
<th></th>
<th>$L^{{\text{XOR, NOT}}}_{(2,3)}$</th>
<th>$L^{{\text{NAND}}}_{(2,3)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of individuals</td>
<td>5552</td>
<td>1446</td>
</tr>
<tr>
<td>No. of optimal solutions</td>
<td>660</td>
<td>8</td>
</tr>
<tr>
<td>No. of neutral networks</td>
<td>1389</td>
<td>31</td>
</tr>
<tr>
<td>Average network size</td>
<td>3.99</td>
<td>46.64</td>
</tr>
</tbody>
</table>

Graphical Representation of the two landscapes

{NAND}  
{XOR, NOT}

Average Neutrality Ratio

Neutrality ratio of an individual $s$ = the number of neutral neighbors of $s$ divided by the total number of neighbors of $s$ ($= \frac{\#V_n(s)}{V(s)}$)

Average neutrality ratio of a neutral network = the average of the neutrality ratios of all the individuals belonging to that network.

- {NAND} “more neutral” for bad fitness values than for good ones
- {XOR, NOT} “more neutral” than {NAND}

Parity Landscapes Properties

- Supposing that all fitness values have been normalized into the range $[0,1]$, if an expression does not contain at least an occurrence of each variable, then its fitness is $= 0.5$ [Langdon, Poli, 2002]

  The wide majority of individuals have fitness $= 0.5$ !!!!

- {XOR, NOT}
  - All possible fitness values are only 0, 0.5 and 1 (in our work 0 = good, 1 = bad).
  - There is only one neutral network at fitness 0.5 (central network)
  - All other nets are composed by one individual (peripheral networks)
  - All peripheral networks are connected with the central one by just one mutation
Average $\Delta$-fitness

Let $N$ be a neutral network, then the average $\Delta$-fitness of $N$ is:

$$\Delta \bar{f}(N) = \frac{1}{|N|} \sum_{s \in N} \frac{\sum_{v \in V(s)} (f(v) - f(s))}{|V(s)|}$$

For (XOR, NOT) mutations can produce larger fitness improvements.

### Sampling Methodology

- **Modified Metropolis Sampling**
  
  Like Metropolis, but gives a reward to individuals with different fitness values (and thus also to the ones with a different fitness than 0.5).

- **Vertical Expansion**
  
  Takes as input the output of the Modified Metropolis sampling and enriches it with some non-neutral neighbors of its individuals.

- **Horizontal Expansion**
  
  Takes as input the output of the Vertical Expansion phase and enriches it with as many neutral neighbors of its individuals as possible (builds neutral networks).
The acceptance probability

\[ \alpha(f(P), f(T)) = (1 - p_m) \log_{10} \left( \frac{9 \cdot |f(P) - f(T)|}{\max(f(P), 1 - f(P))} + 1 \right) + p_m \]

- If \(|f(P) - f(T)| = 0\) then small probability of accepting \(T\) (\(= p_m\))
- The larger the value of \(|f(P) - f(T)|\) the higher the probability of accepting \(T\)
  - In particular, if \(T\) has the most different possible fitness value from \(P\), the \(|f(P) - f(T)| = \max(f(P), 1-f(P))\). In that case, the logarithmic term becomes \(\log_{10}(9+1) = \log_{10}(10) = 1\) and thus \(\alpha(f(P), f(T)) = 1\).

We have chosen a logarithmic function because it increases very quickly for small differences and thus it rewards solutions \(T\) with a slightly different fitness from \(P\).

Vertical Expansion

For each individual \(i \in S\), \(L\) different neighbors of \(i\) are generated by means of \(L\) different mutations.

Each one of these neighbors is accepted or rejected according to the same probability as in the Modified Metropolis algorithm.

Remark: since \(p_m\) is "small", there is a "small" probability of having neutral neighbors in \(S\) at the end of the vertical expansion

Modified Metropolis Sampling

Generate a random individual \(P\); 
\(S = \{P\}\); 
repeat

generate a random individual \(T\); 
\[ \alpha(f(P), f(T)) = (1 - p_m) \log_{10} \left( \frac{9 \cdot |f(P) - f(T)|}{\max(f(P), 1 - f(P))} + 1 \right) + p_m \]
if \(\alpha(f(P), f(T)) > \text{rand}(0,1)\) then \(P = T\); \(S = S \cup \{P\}\) endif

until \(#S = N\) (prefixed constant)
Horizontal Expansion

\[
\begin{align*}
&\text{iter} = 1; \\
&\text{while (at least one incomplete neutral network exists in } S \text{ and } (S) < S_{\text{max}}) \text{ do} \\
&\quad N \leftarrow \text{set of incomplete networks in } S \text{ of size less than } T_{\text{max}}; \\
&\quad \text{forall } N \in N \text{ do} \\
&\quad \quad \text{forall } i \in N \text{ do} \\
&\quad \quad \quad \text{forall } j \in V(i) \text{ do} \\
&\quad \quad \quad \quad \text{if } (\text{rand}(0,1) < \beta(f(i), f(j), \text{iter})) \text{ and } (|S| < S_{\text{max}}) \text{ then} \\
&\quad \quad \quad \quad \quad S \leftarrow S \cup \{j\}; \\
&\quad \quad \text{endif} \\
&\quad \text{endforall} \\
&\quad \text{endforall} \\
&\quad \text{endforall} \\
&\quad \text{endwhile} \\
&\text{return}(S);
\end{align*}
\]

where:
\[
\beta(f(i), f(j), \text{iter}) = \begin{cases} 
1 & \text{if } f(i) = f(j), \\
\frac{1}{k - \text{iter}} & \text{otherwise}
\end{cases}
\]

For each individual in the sample
We consider all its neighbors
And we accept them with probability $\beta$

"Large" Fitness Landscapes

- Even Parity 4 problem
- Maximum Tree Depth = 8
- Sets of Operators: \{NAND\} and \{XOR, NOT\}

Some statistics:

<table>
<thead>
<tr>
<th></th>
<th>{XOR, NOT}</th>
<th>{NAND}</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_m$ for Modified Metropolis</td>
<td>0.005</td>
<td>0.011</td>
</tr>
<tr>
<td>$p_m$ for vertical expansion</td>
<td>0.00065</td>
<td>0.046</td>
</tr>
<tr>
<td>$k$ for horizontal expansion</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Minimal size of an incomplete network</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Sample size of Modified Metropolis</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>$L$ of vertical expansion</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Size of generated sample</td>
<td>14589</td>
<td>30238</td>
</tr>
<tr>
<td>No. of networks contained into the sample</td>
<td>898</td>
<td>492</td>
</tr>
</tbody>
</table>

Average Neutrality Ratio

\{NAND\} “more neutral” for bad fitness values than for good ones
\{XOR, NOT\} “more neutral” than \{NAND\}
Discussion

• Networks with bad fitness values are "more neutral" than networks with good fitness values for \{NAND\}. It is not the case for \{XOR, NOT\}.
• It is unlikely to improve fitness mutating individuals belonging to neutral networks with good fitness for \{NAND\}. It is not the case for \{XOR, NOT\}.
• Mutations induce largest fitness modifications for \{XOR, NOT\} than for \{NAND\}.

Conclusions

Average Neutrality Ratio, Average \(\Delta\)-fitness NI Solutions Ratio and NW solutions ratio (which are all neutrality networks measures) are good hardness indicators for these two particular landscapes.

These results hold both for "small" landscapes that we have studied exhaustively and for "large" landscapes that we have studied by means of our (new!) sampling technique → it seems a "good" sampling technique!
Thank you

vanneschi@disco.unimib.it

http://personal.disco.unimib.it/Vanneschi