2. Two typical geometries of fitness landscapes

Fitness landscape analysis for understanding and designing local search heuristics

Sébastien Verel

LISIC - Université du Littoral Côte d'Opale, Calais, France
http://www-lisic.univ-littoral.fr/~verel/

The 51st CREST Open Workshop
Tutorial on Landscape Analysis
University College London

27th, February, 2017

Outline of this part

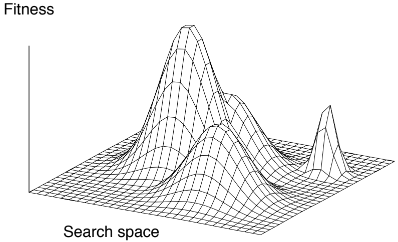
- Basis of fitness landscape :
  - introductory example *(Done)*
  - brief history and background of fitness landscape *(Done)*
  - fundamental definitions *(Done)*
- **Two typical geometries of fitness landscapes** :
  - multimodality
  - ruggedness
  - neutrality
  - neutral networks

Multimodal Fitness landscapes

Local optima s^*

no neighbor solution with strictly higher fitness value (maximization)

$$\forall s \in \mathcal{N}(s^*), \quad f(s) \leqslant f(s^*)$$



Typical example : bit strings

Search space : $X = \{0,1\}^N$

$$\mathcal{N}(x) = \{y \in X \mid d_{Hamming}(x,y) = 1\}$$

Example :
$x = 01101$ and $f_1(x) = f_2(x) = f_3(x) = 5$

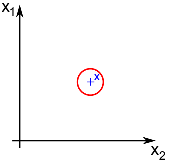
	11101	00101	01001	01111	01100
f_1	4	2	3	0	3
f_2	2	3	6	2	3
f_3	1	5	2	2	4

Question

Is x is a local maximum for f_1, f_2, and/or f_3 ?

Not so typical example : continuous optimization

Still an open question...



Search space : $X = [0,1]^d$

$$\mathcal{N}_\alpha(x) = \{y \in X \mid \|y - x\| \leqslant \alpha\}$$

with $\alpha > 0$

Classical definition of local optimum

x is local maximum iff

$$\exists \varepsilon > 0, \forall y \text{ such that } \|y - x\| \leqslant \varepsilon,\ f(y) \leqslant f(x)$$

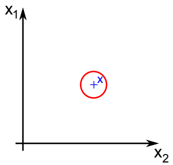
Questions

Local search definition with $\mathcal{N}_\alpha \Rightarrow$ classical definition ?

Classical definition $\Rightarrow$ local search definition with $\mathcal{N}_\alpha$?

Not so typical example : continuous optimization

Still an open question...



Search space : $X = [0,1]^d$

$$\mathcal{N}_\alpha(x) = \{y \in X \mid \|y - x\| \leqslant \alpha\}$$
with $\alpha > 0$

Classical definition of local optimum

x is local maximum iff

$$\exists \varepsilon > 0, \forall y \text{ such that } \|y - x\| \leqslant \varepsilon, \; f(y) \leqslant f(x)$$

Questions

Local search definition with $\mathcal{N}_\alpha \Rightarrow$ classical definition ?

Classical definition $\Rightarrow$ local search definition with $\mathcal{N}_\alpha$?

Still some works to do...

Sampling local optima

Basic estimator (Alyahya, K., & Rowe, J. E. 2016 [AR16])

Expected proportion of local optima :

Proportion of local optima in a sample of random solutions

- Complexity : $n \times |\mathcal{N}|$
- Pros :
  unbiased estimator
- Cons :
  poor estimation when expected proportion is lower than $1/n$

Sampling local optima by adaptive walks

Adaptive walk

$(x_1, x_2, \ldots, x_\ell)$ such that $x_{i+1} \in \mathcal{N}(x_i)$ and $f(x_i) < f(x_{i+1})$

Sampling local optima by adaptive walks

Adaptive walk

$(x_1, x_2, \ldots, x_\ell)$ such that $x_{i+1} \in \mathcal{N}(x_i)$ and $f(x_i) < f(x_{i+1})$

Hill-Climbing algorithm (first-improvement)

Choose initial solution $x \in X$
repeat
    choose $x' \in \{y \in \mathcal{N}(x) \mid f(y) > f(x)\}$
    if $f(x) < f(x')$ **then**
        $x \leftarrow x'$
    end if
until x is a Local Optimum

Sampling local optima by adaptive walks

Adaptive walk

$$(x_1, x_2, \ldots, x_\ell) \text{ such that } x_{i+1} \in \mathcal{N}(x_i) \text{ and } f(x_i) < f(x_{i+1})$$

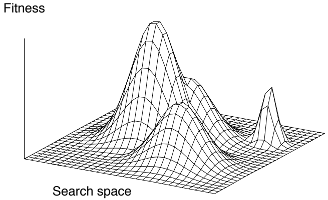
Hill-Climbing algorithm (first-improvement)

Choose initial solution $x \in X$
repeat
  choose $x' \in \{y \in \mathcal{N}(x) \mid f(y) > f(x)\}$
  if $f(x) < f(x')$ **then**
    $x \leftarrow x'$
  end if
until x is a Local Optimum

Basin of attraction of x^*

$$\{x \in X \mid HillClimbing(x) = x^*\}.$$

Multimodal Fitness landscapes and difficulty



The idea :

- if the size of attractive basin of global optimum is "small",
- then, the "time" to find the global optimum is "long"

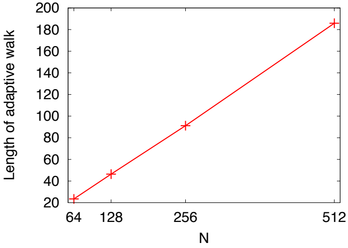
Optimisation difficulty :
Number and size of attractive basins (Garnier *et al.* [GK02])

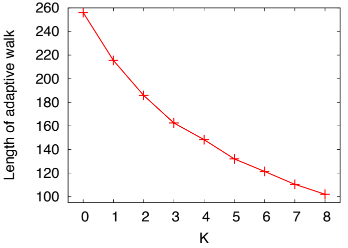
Feature to estimate basin size :

- **Length of adaptive walks**

complexity : sample size $\times \ell \times |\mathcal{N}|$

Multimodal Fitness landscapes and difficulty





ex. nk-landscapes with $n = 512$

The idea :

- if the size of attractive basin of global optimum is "small",
- then, the "time" to find the global optimum is "long"

Optimisation difficulty :
Number and size of attractive basins (Garnier *et al.* [GK02])

Feature to estimate basin size :

- **Length of adaptive walks**

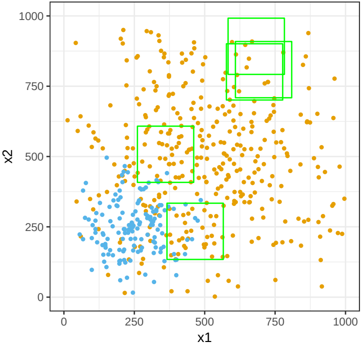
complexity : sample size $\times \ell \times |\mathcal{N}|$

Practice : the Squares Problem

a program design problem ?

Squares Problem (SP)

Find the position of 5 squares in order to maximize inside squares the number of brown points without blue points



Candidate solutions

$$X = ([0,1000] \times [0,1000])^5$$

	x_1	x_2
1	577	701
2	609	709
3	366	134
4	261	408
5	583	792

Fitness function

$f(x) =$ number of brown points $-$ number of blue points inside squares

Source code in R : ex01.R

Source code : `http://www-lisic.univ-littoral.fr/~verel/`

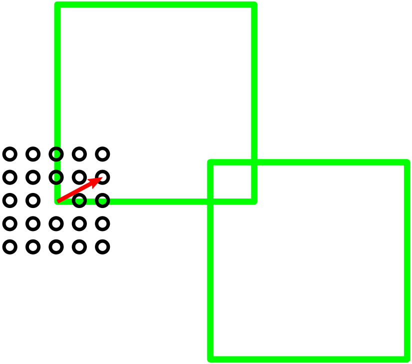
Different functions are already defined :

- `main` : example to execute the following functions
- `draw` and `draw_solution` :
  draw a problem and the squares of a solution
- `fitness_create` :
  create a fitness function from a data frame of points
- `pb1_create` and `pb2_create` :
  create two particular SP problems
- `init` :
  create a random solution with *n* squares
- `hc_ngh` :
  hill-climbing local search based on neighborhood

Neighborhood

Questions

- Execute line by line the `main` function
- Define the `neighborhood_create` which creates a neighborhood : a neighbor move one square



Adaptive walks to compare problem difficulty

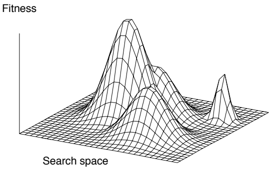
Pre-defined functions :

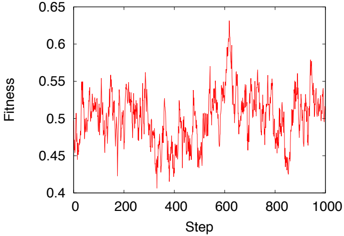
- `adaptive_length` :
  run the hill-climber and compute a data frame with the length of adaptive walks
- `main_adaptive_length_analysis` :
  Compute the adaptive length of two different SP problems

Questions

- Execute line by line the `main_adaptive_length_analysis` function to compute a sample of adaptive walk lengths.
- Compare the lengths of adaptive walks for the two SP problems.
- Which one is the more multi-modal ?

Random Walk to measure the ruggedness





Random walk :

- $(x_1, x_2, \ldots)$ where $x_{i+1} \in \mathcal{N}(x_i)$ and equiprobability on $\mathcal{N}(x_i)$

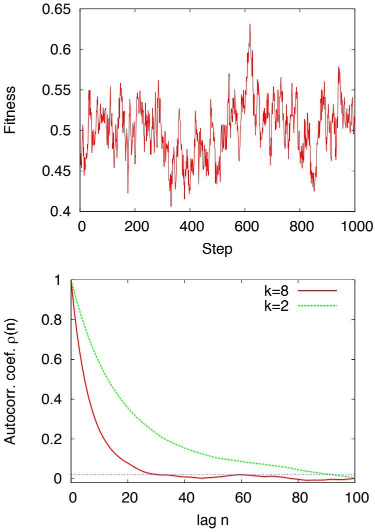
The idea :

- if the profile of fitness is irregular,
- then, the "information" between neighbors is low.

Feature :

- Study the fitness profile like a signal

Rugged/smooth fitness landscapes



Autocorrelation function of time series of fitnesses along a random walk (Weinberger 90 [Wei90]) :

$$\rho(n) = \frac{\mathrm{E}[(f(x_i) - \bar{f})(f(x_{i+n}) - \bar{f})]}{\mathrm{var}(f(x_i))}$$

Autocorrelation length $\tau = \frac{1}{\rho(1)}$

"How many random steps such that correlation becomes insignificant"

- small τ : rugged landscape
- long τ : smooth landscape

complexity : sample size $\approx 10^3$

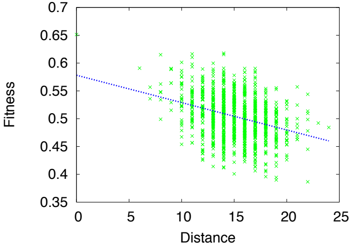
Results on rugged fitness landscapes (Stadler 96 [Sta96])

Ruggedness decreases with the size of thoses problems

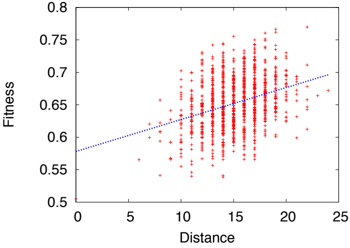
Problem	parameter	$\rho(1)$
symmetric TSP	n number of towns	$1 - \frac{4}{n}$
anti-symmetric TSP	n number of towns	$1 - \frac{4}{n-1}$
Graph Coloring Problem	n number of nodes α number of colors	$1 - \frac{2\alpha}{(\alpha-1)n}$
NK landscapes	N number of proteins K number of epistasis links	$1 - \frac{K+1}{N}$
random max-k-SAT	n number of variables k variables per clause	$1 - \frac{k}{n(1-2^{-k})}$

Fitness Distance Correlation (FDC) (Jones 95 [Jon95])

Correlation between fitness and distance to global optimum



easy



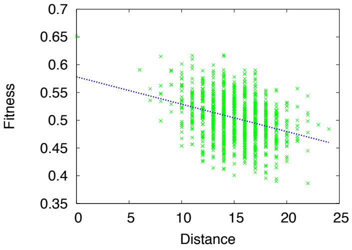
hard

Classification based on experimental studies :

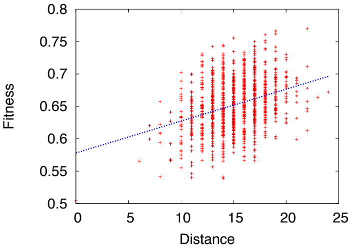
- $\rho < -0.15$, easy optimization
- $\rho > 0.15$, hard optimization
- $-0.15 < \rho < 0.15$, undecided zone

Fitness Distance Correlation (FDC) (Jones 95 [Jon95])

Correlation between fitness and distance to global optimum



easy



hard

- Important concept to understand
- Not useful in "practice" (difficult to estimate)

Practice : computation the autocorrelation function

Source code `exo02.R` :

- `mutation_create` :
  Create a mutation operator,
  modify each square according to rate p,
  a new random value from $[(x-r, y-r), (x+r, y+r)]$.
- `main` :
  Code to obtain autocorrelation function

Questions

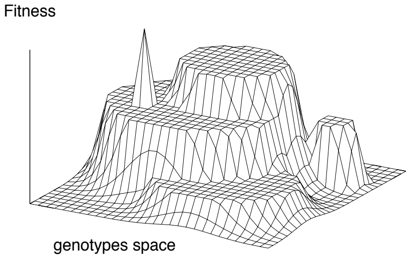
- Define the function `random_walk` to compute the fitness values during a random walk.
- Execute line by line the `main` function to compute a sample of fitness value collected during a random walk.
- Compare the first autocorrelation coefficient of the SP problems 1 and 2.

Neutral Fitness Landscapes

Neutral theory (Kimura ≈ 1960 [Kim83])

Theory of mutation and random drift

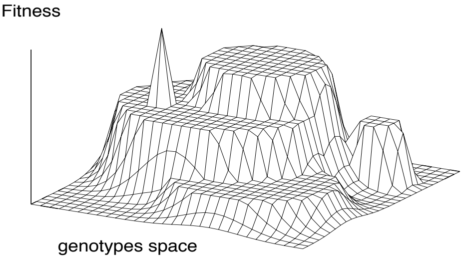
A considerable number of mutations have no effects on fitness values



- plateaus
- neutral degree
- neutral networks [Schuster 1994 [SFSH94], RNA folding]

Neutral Fitness Landscapes

- Redundant problem (symmetries, etc.) [GS87]
- Problem “not well” defined or dynamic environment [IT04]
- Unused variables, discrete values, etc.



Real-world problems :

- Robot controler
- Circuit design
- Genetic Programming
- Protein folding
- Learning problems
- Scheduling problems
- Graph problems...

Neutrality and difficulty

- In our knowledge, there is no definitive answer about the relation between neutrality and problem hardness
- Certainly, it is dependent on the "nature" of neutrality

Solving optimization problem and neutrality

3 ways to deals with neutrality :

- Decrease the neutrality : reduce the entropy barrier
- Increase the neutrality : reduce the fitness barrier
- Unchange the neutrality : use a specific algorithm

Sharp description of the geometry
of neutral fitness landscapes is needed

Neutrality and difficulty

We know for certain that :

- **No information** is better than **Bad information** :
  From a non-optimal solution, hard trap functions are more difficult than needle-in-a-haystack functions
- **Good information** is better than **No information** :
  Onemax problem is much easier than needle-in-a-haystack functions

Neutrality and difficulty

We know for certain that :

- **No information** is better than **Bad information** :
  From a non-optimal solution, hard trap functions are more difficult than needle-in-a-haystack functions
- **Good information** is better than **No information** :
  Onemax problem is much easier than needle-in-a-haystack functions

- When there is No information :
  you should have a good method to create it !

Objects of neutral fitness landscapes

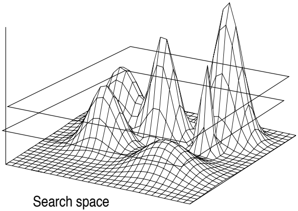
Description of multimodal fitness landscapes is based on :

- Local optima
- Basins of attraction

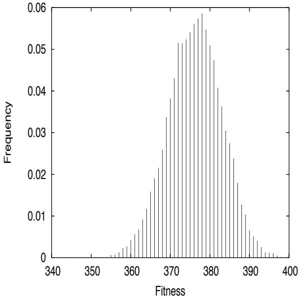
Description of neutral fitness landscapes is based on :

- **Neutral sets** :
  set of solutions with the same fitness
- **Neutral networks** :
  neutral sets with neighborhood relation

Neutral sets : Density Of States



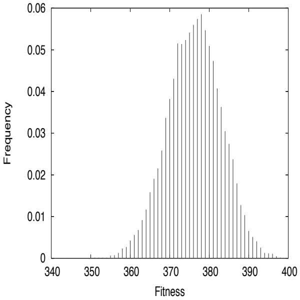
Set of solutions with fitness value



Density of states (D.O.S.)

- Introduce in physics (Rosé 1996 [REA96])
- Optimization (Belaidouni, Hao 00 [BH00])

Neutral sets : Density Of States



Informations given :

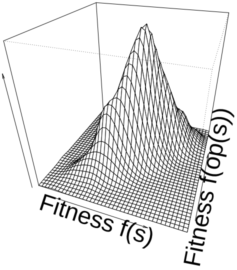
- Performance of random search
- Tail of the distribution is an indicator of difficulty :
  - the faster the decay, the harder the problem
- But do not care about the neighborhood relation

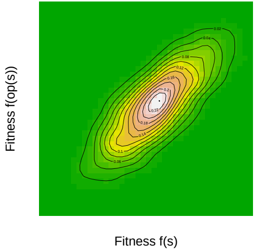
Features :

- **Average**, **sd**, **kurtosis**, etc.

complexity : sample size

Neutral sets : Fitness Cloud [Verel et al. 2003]





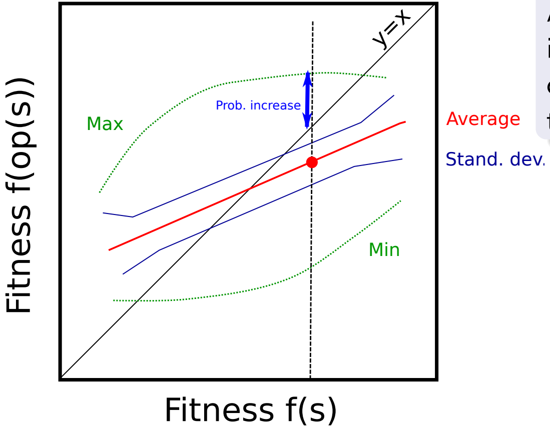
- $(X, \mathcal{F}, \Pr)$: probability space
- $op : X \to X$ stochastic operator of the local search

- $X(s) = f(s)$
- $Y(s) = f(op(s))$

Fitness Cloud of *op*

Conditional probability density function of Y given X

Fitness cloud : Measure of evolvability



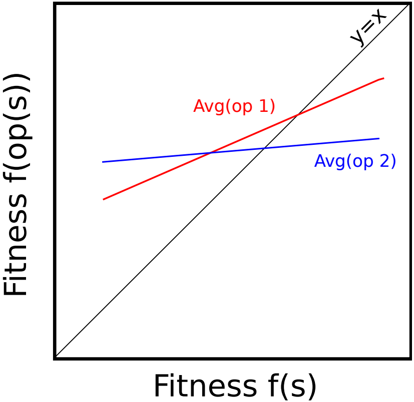
Evolvability

Ability to evolve : fitness in the neighborhood compared to the fitness of the solution

- Probability of finding better solutions
- Average fitness of better neighbor solutions
- Average and standard deviation of fitnesses

Fitness cloud : Comparison of difficulty

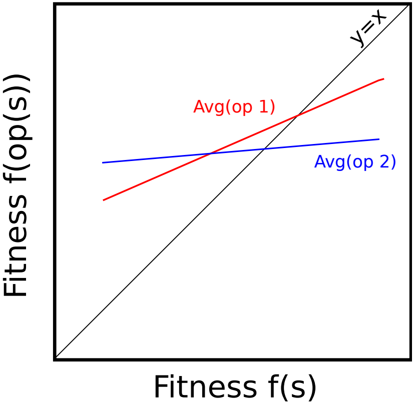
Average of evolvability



- Operator 1 ? ? Operator 2

Fitness cloud : Comparison of difficulty

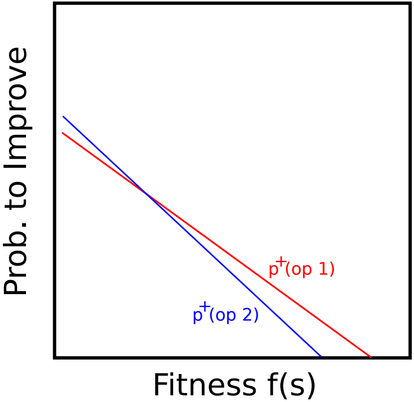
Average of evolvability



- Operator 1 > Operator 2
- Because Average 1 more correlated to fitness
- Linked to autocorrelation
- Average is often a line :
  - See works on Elementary Landscapes (Stadler, D. Wihtley, F. Chicano and others)
  - See the idea of Negative Slope Coefficient (NSC)

Fitness cloud : Comparison of difficulty

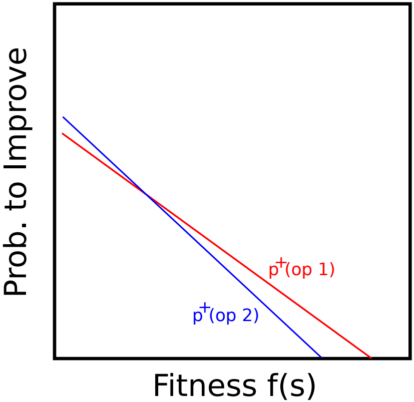
Probability to improve



- Operator 1 ? ? Operator 2

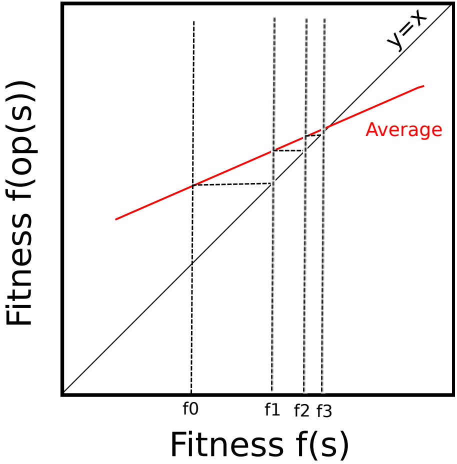
Fitness cloud : Comparison of difficulty

Probability to improve



- Operator 1 > Operator 2
- Prob. to improve of 1 is often higher than Prob. to improve of 2
- Probability to improve is often a line
- See also works on fitness-probability cloud (G. Lu, J. Li, X. Yao [LLY11])
- See theory of EA and fitness level technics.

Fitness cloud : estimation of convergence point

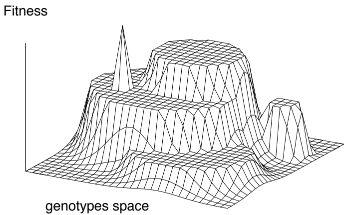


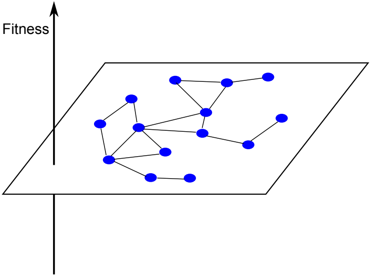
- Approximation (only approximation) of the fitness value after few steps of local operator
- Indication on the quality of the operator
- See fitness level technic

Outline

- Neutral sets (done) :
  set of solutions with the same fitness
  $\Rightarrow$ No structure
- Fitness cloud (done) :
  Bivariate density $(f(s), f(op(s)))$
  $\Rightarrow$ Neighborhood relation **between** neutral sets
- Neutral networks (to be done) :
  $\Rightarrow$ neutral sets with neighborhood relation : graph

Neutral networks (Schuster 1994 [SFSH94])





Basic definition of Neutral Network

Node = solution with same fitness value

Edge = neighborhood relation

Definitions

Test of neutrality

$$isNeutral : S \times S \rightarrow \{true, false\}$$

For example, $isNeutral(x_1, x_2)$ is *true* if :

- $f(x_1) = f(x_2)$.
- $|f(x_1) - f(x_2)| \leqslant 1/M$ with M is the search population size.
- $|f(x_1) - f(x_2)|$ is under the evaluation error.

Neutral neighborhood

of s is the set of neighbors which have the same fitness $f(s)$

$$\mathcal{N}_{neut}(s) = \{s' \in \mathcal{N}(s) \mid isNeutral(s, s')\}$$

Neutral degree of s

Number of neutral neighbors : $nDeg(s) = \sharp(\mathcal{N}_{neut}(s) - \{s\})$.

Definitions

Neutral walk

$W_{neut} = (x_0, x_1, \ldots, x_m)$

- for all $i \in [0, m-1]$, $x_{i+1} \in \mathcal{N}(x_i)$
- for all $(i, j) \in [0, m]^2$, $isNeutral(x_i, x_j)$ is *true*.

Neutral Network

graph $G = (N, E)$

- $N \subset X$: for all s and $s^{'}$ from N, there is a neutral walk belonging to N from s to $s^{'}$,
- $(x_1, x_2) \in E$ if they are neutral neighbors : $x_2 \in \mathcal{N}_{neut}(x_1)$

A fitness landscape is neutral
if there are many solutions with high neutral degree.

Practice : computation of the neutral rate

The neutral rate is the proportion of neutral neighbors.
It can be estimated by a random walk :

$$\frac{\sharp\{(x_t, x_{t+1}) \ : \ f(x_t) = f(x_{t+1}), t \in \{1, \ell - 1\}\}}{\ell - 1}$$

Source code `exo03.R` :

- `main` :
  Code to compute the neutral rates

Questions

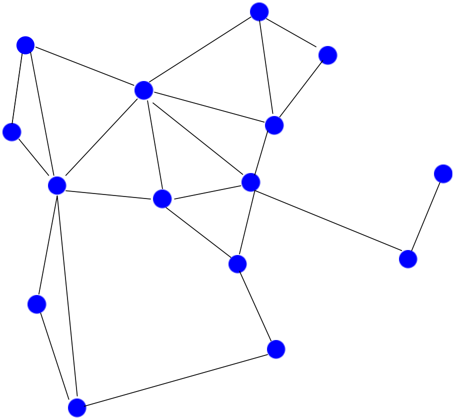
- Define the function `neutral_rate` to compute the neutral rate estimated with a random walk.
- Execute the `main` function to compute the neutral rate.
- Compare the neutrality of the SP problems 1 and 2.

Features inside neutral network

Classical graph metrics :

1. **Size of NN** :
   - number of nodes of NN,
2. **Neutral degree distribution** :
   - measure of the quantity of "neutrality"
3. **Autocorrelation of neutral degree** (Bastolla 03 [BPRV03]) during neutral random walk :
   - comparaison with random graph,
   - measure of the correlation structure of *NN*

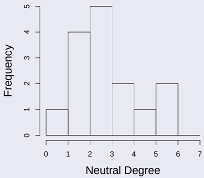
Features inside neutral network



1. **Size**

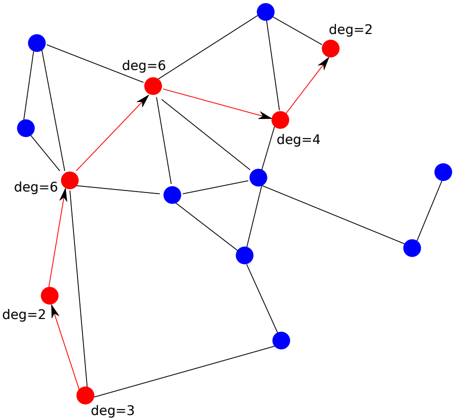
   avg,distribution,etc.

2. **Neutral degree distribution**



3. **Autocorrelation of neutral degree**
   - random walk on NN
   - autocorr. of degrees

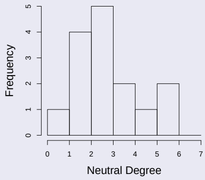
Features inside neutral network



1. **Size**

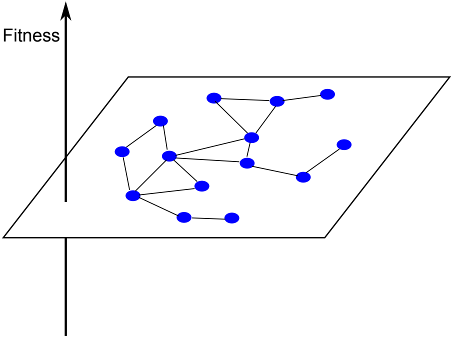
   avg,distribution,etc.

2. **Neutral degree distribution**



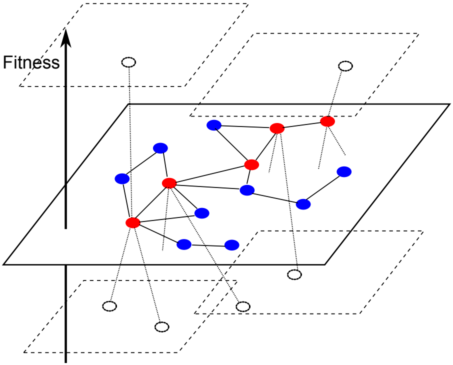
3. **Autocorrelation of neutral degree**
   - random walk on NN
   - autocorr. of degrees

Features between neutral networks



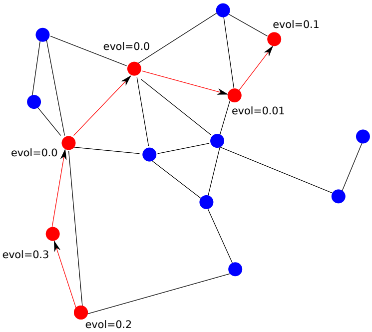
1. **Rate of innovation** (Huynen 96 [Huy96]) : the number of new accessible structures (fitness) per mutation
2. **Autocorrelation of evolvability** [VCC06] : autocorrelation of the sequence $(evol(x_0), evol(x_1), \ldots)$.

Features between neutral networks



1. **Rate of innovation** (Huynen 96 [Huy96]) : the number of new accessible structures (fitness) per mutation
2. **Autocorrelation of evolvability** [VCC06] : autocorrelation of the sequence $(evol(x_0), evol(x_1), \ldots)$.

Features between neutral networks



- Autocorrelation of evolvability :
  - Autocorrelation of $(evol(x_0), evol(x_1), \ldots)$.
  - Evolvability *evol* :
    - average fitness in the neigh.
    - prob. to improve, etc.
- Informations :
  - if high correlation $\Rightarrow$ "easy" (you can use this information)
  - if low correlation $\Rightarrow$ "difficult"

Summary of neutral fitness landscapes features

- Density of States :
  Size of neutral sets
- Fitness cloud and related statistics :
  Evolvability of solutions
- Neutral degrees distribution :
  "How neutral is the fitness landscape ?"
- Autocorrelation of neutral degrees :
  Network "structure"
- Autocorrelation of evolvability :
  Evolution of evolvability on NN

Practice : Performance *vs.* fitness landscape features

Explain the performance of ILS with fitness landscape features ?

- 20 random SP problems have been generated : `pb_xx.csv`
- The performance of Iterated Local Search have been computed in `perf_ils_xx.csv` (30 runs)
- Goal : regression of ILS performance with fitness landscape features

Practice : Performance *vs.* fitness landscape features

Source code `exo04.R` :

- `fitness_landscape_features` :
  Compute the basic fitness landscape features
- `random_walk_samplings` :
  Random walk sampling on each problem (save into file)
- `fitness_landscape_analysis` :
  Compute the features for each problems
- `ils_performance` :
  Add the performance of ILS into the data frame
- `main` :
  Execute the previous functions.

Practice : Performance *vs.* fitness landscape features

Questions

- What are the features computed by the function `fitness_landscape_features`?
- Execute the `random_walk_samplings` function to compute the random walks samplings.
- Compute the correlation plots between features and ILS performance (use ggpairs).
- Compute the linear regression of performance with fitness landscape features.

References I

Khulood Alyahya and Jonathan E Rowe.
Simple random sampling estimation of the number of local optima.
In *International Conference on Parallel Problem Solving from Nature*, pages 932–941. Springer, 2016.

Meriema Belaidouni and Jin-Kao Hao.
An analysis of the configuration space of the maximal constraint satisfaction problem.
In *PPSN VI : Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, pages 49–58, London, UK, 2000. Springer-Verlag.

U. Bastolla, M. Porto, H. E. Roman, and M. Vendruscolo.
Statiscal properties of neutral evolution.
Journal Molecular Evolution, 57(S) :103–119, August 2003.

References II

Josselin Garnier and Leila Kallel.
Efficiency of local search with multiple local optima.
SIAM Journal on Discrete Mathematics, 15(1) :122–141, 2002.

David E. Goldberg and Philip Segrest.
Finite markov chain analysis of genetic algorithms.
In *ICGA*, pages 1–8, 1987.

M. Huynen.
Exploring phenotype space through neutral evolution.
Journal Molecular Evolution, 43 :165–169, 1996.

References III

E. Izquierdo-Torres.
The role of nearly neutral mutations in the evolution of dynamical neural networks.
In J. Pollack and al, editors, *Ninth International Conference of the Simulation and Synthesis of Living Systems (Alife 9)*, pages 322–327. MIT Press, 2004.

T. Jones.
Evolutionary Algorithms, Fitness Landscapes and Search.
PhD thesis, University of New Mexico, Albuquerque, 1995.

M. Kimura.
The Neutral Theory of Molecular Evolution.
Cambridge University Press, Cambridge, UK, 1983.

References IV

Guanzhou Lu, Jinlong Li, and Xin Yao.
Fitness-probability cloud and a measure of problem hardness for evolutionary algorithms.
In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 108–117. Springer, 2011.

Helge Rosé, Werner Ebeling, and Torsten Asselmeyer.
The density of states - a measure of the difficulty of optimisation problems.
In *Parallel Problem Solving from Nature*, pages 208–217, 1996.

P. Schuster, W. Fontana, P. F. Stadler, and I. L. Hofacker.
From sequences to shapes and back : a case study in RNA secondary structures.
In *Proc. R. Soc. London B.*, volume 255, pages 279–284, 1994.

References V

Peter F. Stadler.
Landscapes and their correlation functions.
J. Math. Chem., 20 :1–45, 1996.

Sebastien Verel, Philippe Collard, and Manuel Clergue.
Measuring the evolvability landscape to study neutrality.
In M. Keijzer and et al., editors, *Poster at Genetic and Evolutionary Computation – GECCO-2006*, pages 613–614, Seatle, 8-12 July 2006. ACM Press.

E. D. Weinberger.
Correlated and uncorrelatated fitness landscapes and how to tell the difference.
In *Biological Cybernetics*, pages 63 :325–336, 1990.