Exact Computation of the Expectation Curves of the Bit-Flip Mutation using Landscapes Theory

NEO Networking and Emerging Optimization

LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD DE MÁLAGA

Francisco Chicano and Enrique Alba

Motivation and Outline

- **Research Question: what is the expected fitness of a solution after bit-flip mutation?**
- **We answer this question with the help of landscape theory**

1 **Review of Bit-Flip Mutation**

2 **Background on landscape theory**

3 **Main result: expected fitness**

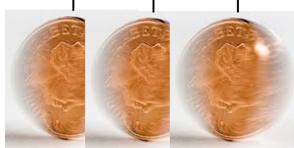
4 **Applications**

5 **Conclusions and future work**

Bit-flip Mutation Operator

- **Given a probability *p* of inverting one single bit…**

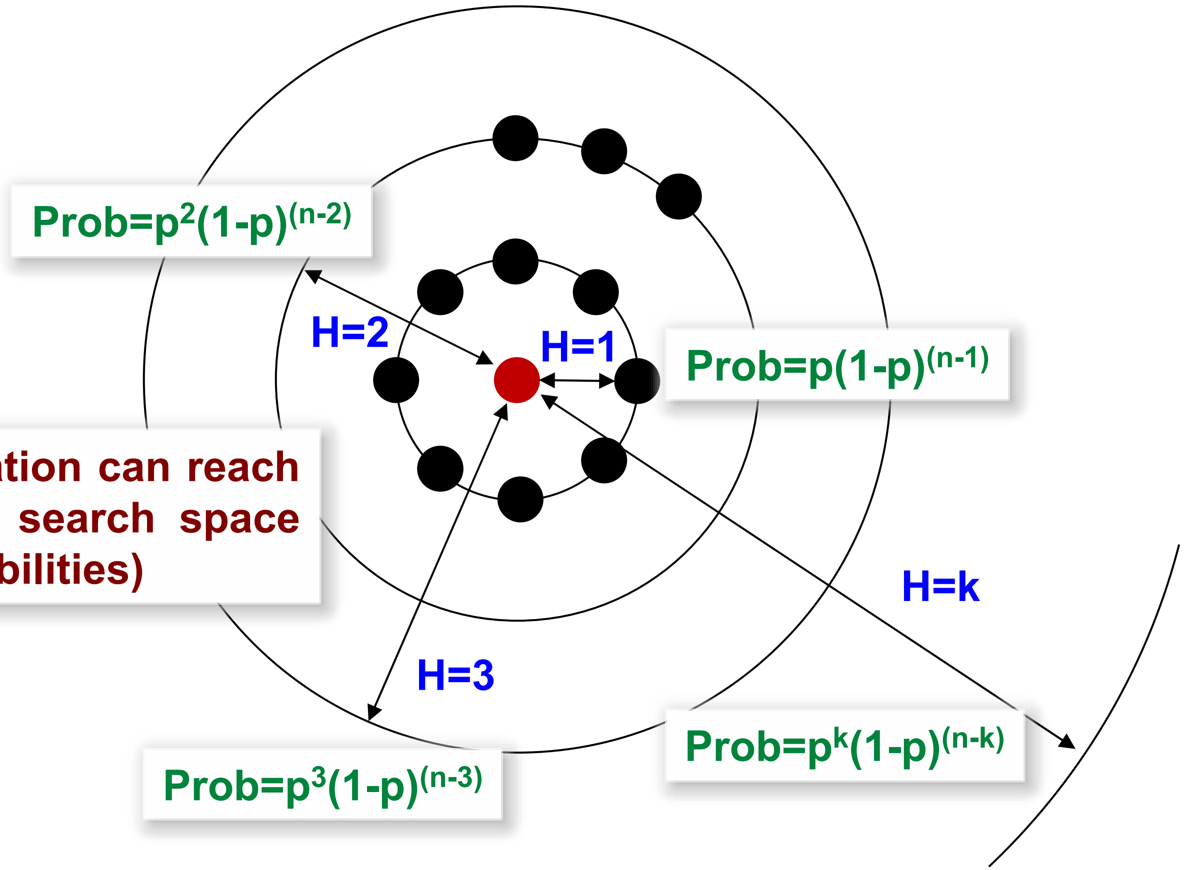
0 1 1 0 1 0 1 1 0 0



- **Special cases**
  - **p=0: no change** — 0 1 0 0 1 0 1 1 1 0 → 0 1 0 0 1 0 1 1 1 0
  - **p=1/2: random sol.** — 0 1 0 0 1 0 1 1 1 0 → 1 0 0 1 1 0 1 0 1 0
  - **p=1/n: average 1 flip** — 0 1 0 0 1 0 1 1 1 0 → 0 1 0 1 1 0 1 1 1 0
  - **p=1: complement** — 0 1 0 0 1 0 1 1 1 0 → 1 0 1 1 0 1 0 0 0 1

Bit-flip Mutation Operator

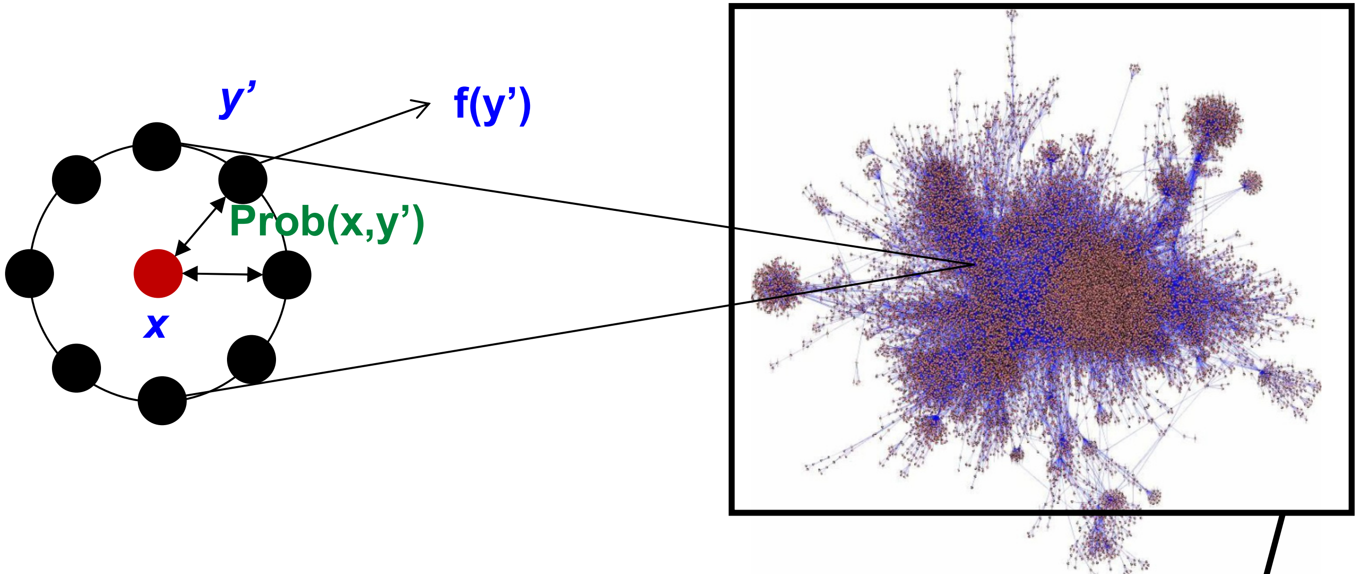
- **A different point of view:** ***p*** **is the probability of flipping a bit, H is the Hamming distance**



If $0 < p < 1$ the mutation can reach any solution in the search space (with different probabilities)

Bit-flip Mutation Operator

- The fitness expectation after a mutation, $E[f]_x$, can be computed as:



Alternatives

- Exact inefficient computation
- Approx. efficient computation
- Exact efficient computation

Sum over the search space!!!

Set of solutions

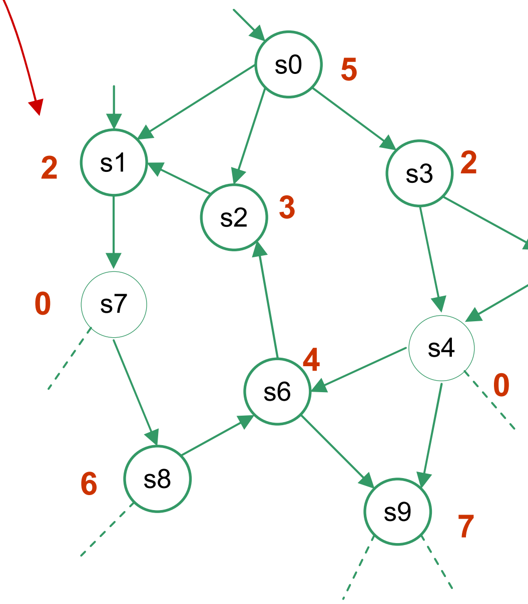
$$E[f]_x = Prob(x,y) \cdot f(y) + Prob(x,y') \cdot f(y') + \ldots = \sum_{z \in \boxed{X}} Prob(x,z) \cdot f(z)$$

- We compute $E[f]_x$ avoiding the sum using landscape theory

Landscape Definition

- A **landscape** is a triple (X,N,f) where
  - X is the solution space
  - N is the neighbourhood operator
  - f is the objective function

The pair (X,N) is called **configuration space**



- The **neighbourhood operator** is a function

  $N: X \rightarrow P(X)$

- Solution y is **neighbour of** x if $y \in N(x)$
- **Regular** and **symmetric neighbourhoods**
  - $d=|N(x)| \quad \forall\, x \in X$
  - $y \in N(x) \Leftrightarrow x \in N(y)$
- **Objective function**

  $f: X \rightarrow R$ (or N, Z, Q)

Elementary Landscapes: Formal Definition

- **An elementary function is an eigenvector of the graph Laplacian (plus constant)**

Adjacency matrix

$$A_{xy} = \begin{cases} 1 & \text{if } y \in N(x) \\ 0 & \text{otherwise} \end{cases}$$

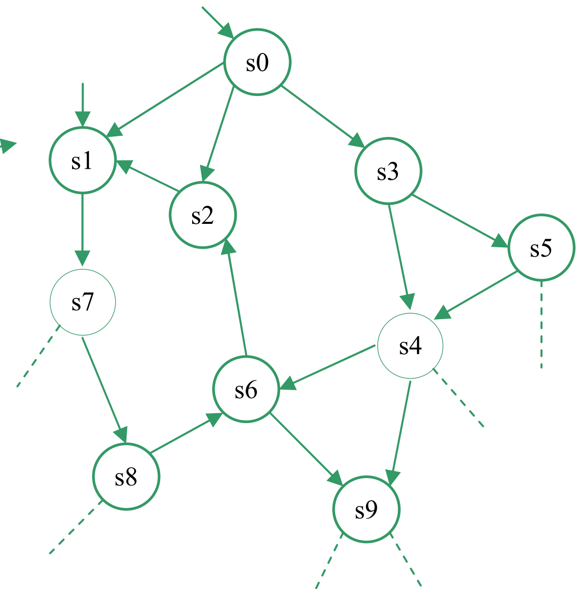
Degree matrix

$$D_{xy} = \begin{cases} |N(x)| & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$$

- **Graph Laplacian:**

$$\Delta = A - D$$

Depends on the configuration space



- **Elementary function: eigenvector of Δ (plus constant)**

$$(-\Delta) \times \left(\vec{f} - b\right) = \lambda \cdot \left(\vec{f} - b\right)$$

Eigenvalue

Elementary Landscapes: Characterizations

- An **elementary landscape** is a landscape for which

$$\underset{y \in N(x)}{\text{avg}}\{f(y)\} = \alpha f(x) + \beta \quad \forall x \in X$$

Depend on the problem/instance

where

$$\underset{y \in N(x)}{\text{avg}}\{f(y)\} \stackrel{\text{def}}{=} \frac{1}{d} \sum_{y \in N(x)} f(y)$$

Linear relationship

$$\bar{f} = \frac{1}{|X|} \sum_{y \in X} f(y)$$

- **Grover's wave equation**

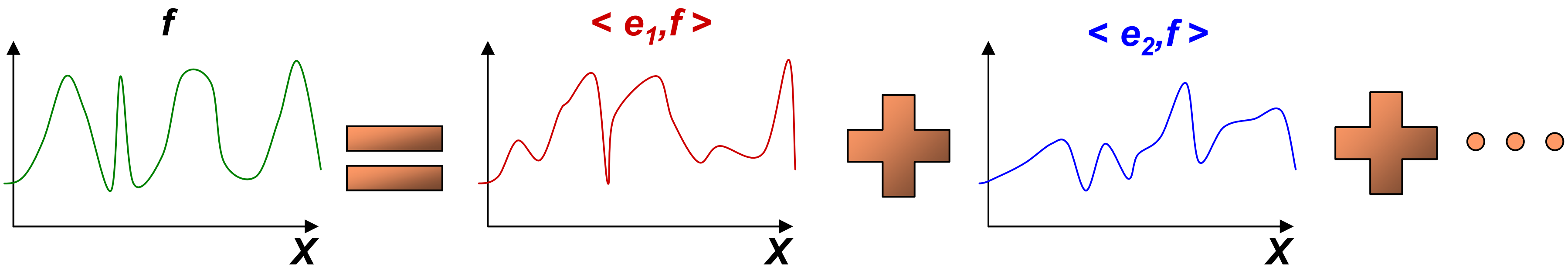
$$\underset{y \in N(x)}{\text{avg}}\{f(y)\} = f(x) + \frac{\lambda}{d}\left(\bar{f} - f(x)\right)$$

Eigenvalue

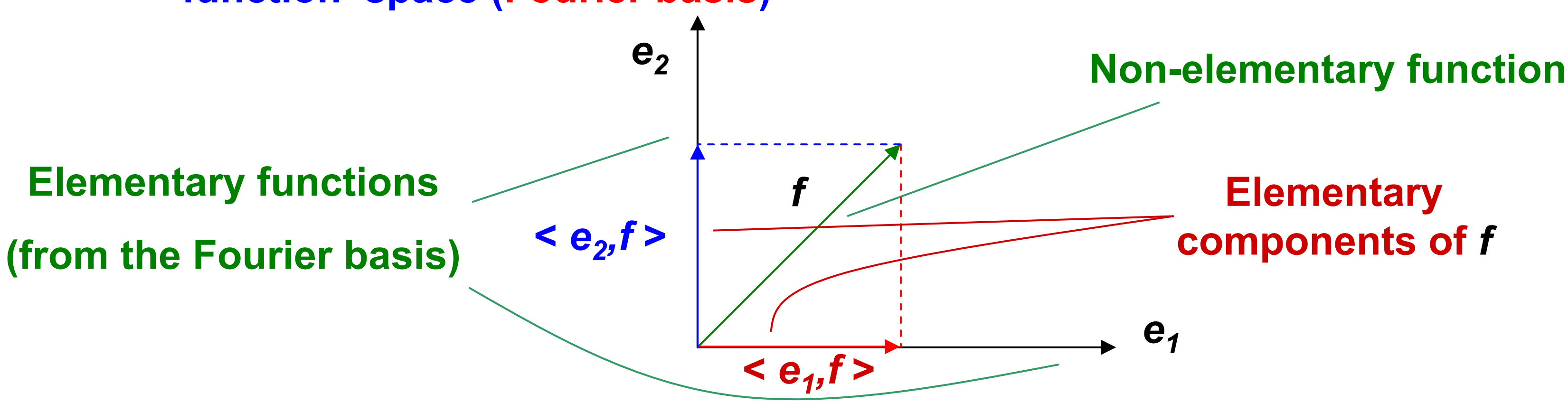
$$\alpha = 1 - \frac{\lambda}{d} \qquad \beta = \frac{\lambda}{d}\bar{f}$$

Landscape Decomposition

- **What if the landscape is not elementary?**
- **Any landscape can be written as the sum of elementary landscapes**



- **There exists a set of eigenfunctions of Δ that form a basis of the function space (Fourier basis)**



Examples

Elementary Landscapes

Problem	Neighbourhood	d	k
Symmetric TSP	2-opt	$n(n-3)/2$	$n-1$
	swap two cities	$n(n-1)/2$	$2(n-1)$
Graph α-Coloring	recolor 1 vertex	$(\alpha-1)n$	2α
Max Cut	one-change	n	4
Weight Partition	one-change	n	4

Sum of elementary Landscapes

Problem	Neighbourhood	d	Components
General TSP	inversions	$n(n-1)/2$	2
	swap two cities	$n(n-1)/2$	2
Subset Sum Problem	one-change	n	2
MAX k-SAT	one-change	n	k
QAP	swap two elements	$n(n-1)/2$	3

Binary Search Space

- **The set of solutions *X* is the set of binary strings with length *n***

0	1	0	0	1	0	1	1	1	0

- **Neighborhood used in the proof of our main result: one-change neighborhood:**
  - **Two solutions *x* and *y* are neighbors iff *Hamming(x,y)=1***

0	1	0	0	1	0	1	1	1	0

1	1	0	0	1	0	1	1	1	0
0	0	0	0	1	0	1	1	1	0
0	1	1	0	1	0	1	1	1	0
0	1	0	1	1	0	1	1	1	0
0	1	0	0	0	0	1	1	1	0

0	1	0	0	1	1	1	1	1	0
0	1	0	0	1	0	0	1	1	0
0	1	0	0	1	0	1	0	1	0
0	1	0	0	1	0	1	1	0	0
0	1	0	0	1	0	1	1	1	1

Landscape Decomposition: Walsh Functions

- **If *X* is the set of binary strings of length *n* and *N* is the one-change neighbourhood then a Fourier basis is**

$$\{\psi_w\} \quad w \in \{0,1\}^n$$

where

Bitwise AND

$$\psi_w(x) = (-1)^{bc(x \wedge w)} = \prod_{i=1}^{n} (-1)^{x_i w_i} = (-1)^{\sum_{i=1}^{n} x_i w_i}$$

Bit count function (number of ones)

- **These functions are known as Walsh Functions**
- **The function with subindex *w* is elementary with *k*=2 *j*, and *j*=*bc* (*w*) is called order of the Walsh function**

Expectation for Elementary Landscapes

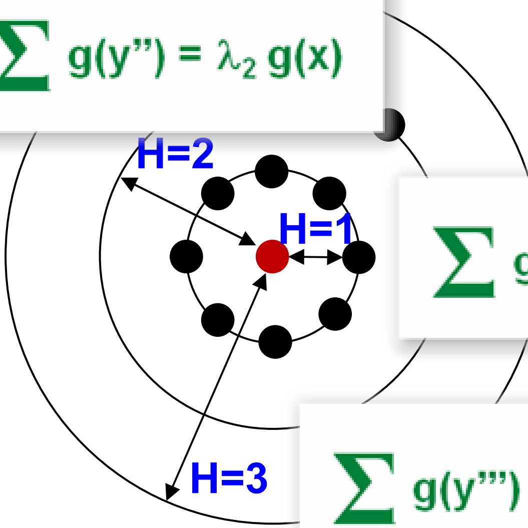
- **If f is elementary, the expected value after the mutation with probability p is:**

$$\mathbb{E}[f]_x = \bar{f} + (1-2p)^{j}(f(x) - \bar{f})$$

Order of the function

- **Sketch of the proof: if g is elementary and $\bar{g}$ = 0**

$$\sum g(y'') = \lambda_2\, g(x)$$



$$\sum g(y') = \lambda_1\, g(x)$$

$$\sum g(y''') = \lambda_3\, g(x)$$

$$E[g]_x = \sum_{z \in X} Prob(x,z) \cdot g(z)$$

$$= \underbrace{\sum_k p^k (1-p)^{(n-k)} \lambda_k\, g(x)}_{(1-2p)^j}$$

Analysis

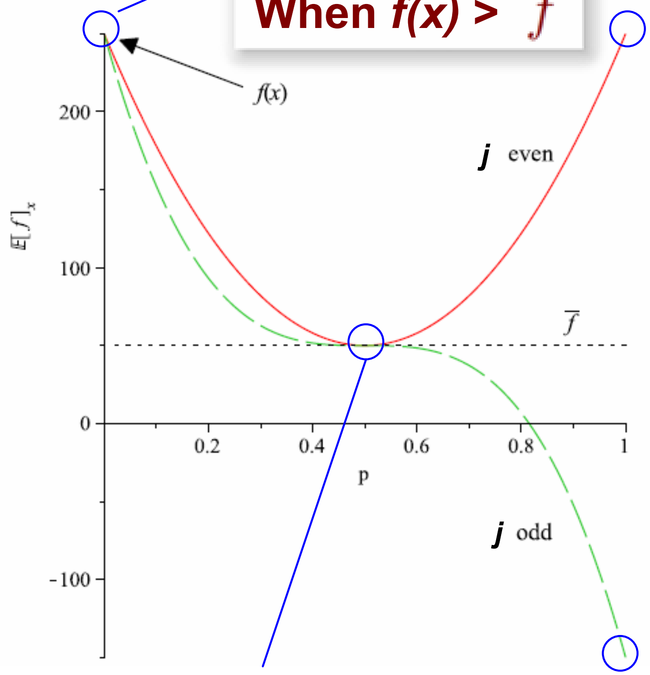
- **Analysis of the expected mutation**

$$\mathbb{E}[f]_x = \bar{f} + (1-2p)^j(f(x) - \bar{f})$$

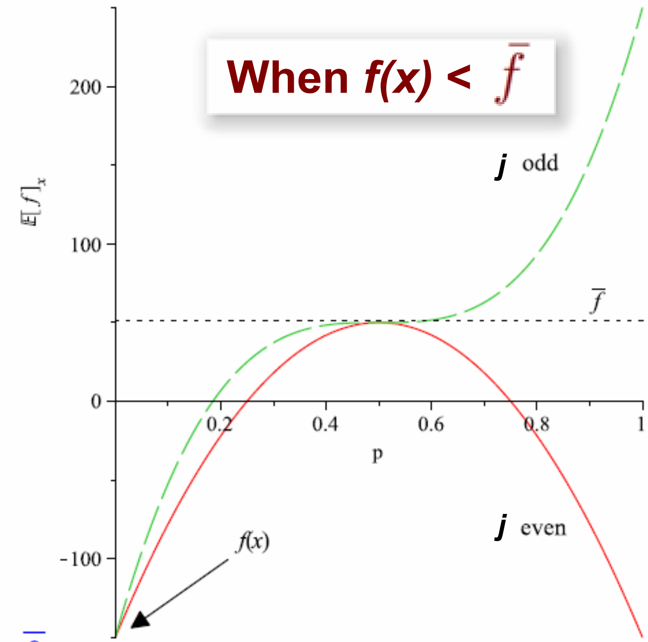
p=0 $\rightarrow$ fitness does not change

When $f(x) > \bar{f}$

p=1 $\rightarrow$ the same fitness



When $f(x) < \bar{f}$



p=1/2 $\rightarrow$ start from scratch

p=1 $\rightarrow$ flip around $\bar{f}$

Expectation for Arbitrary Functions

- **In the general case *f* is the sum of at most *n* elementary landscapes**

$$f = \sum_{j=1}^{n} \Omega_{2j}$$

- **And, due to the linearity of the expectation, the expected fitness after the mutation is:**

$$\mathbb{E}[f]_x = \overline{f} + \sum_{j=1}^{n} (1-2p)^j \left(\Omega_{2j}(x) - \overline{\Omega_{2j}}\right)$$

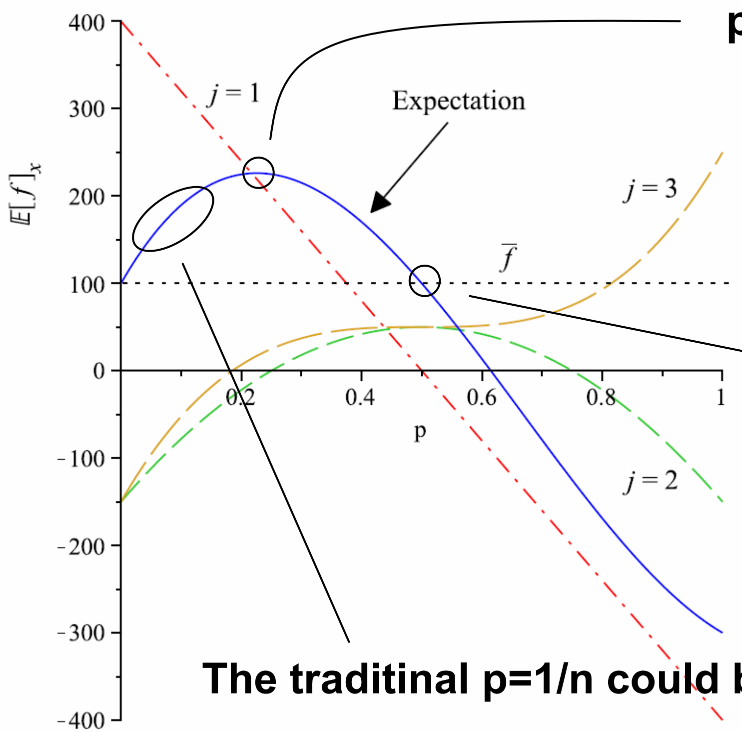
Average value of Ω_{2j} over the whole search space

Analysis

- **Analysis of the expected mutation**
- **Example (j=1,2,3):**

$$\mathbb{E}[f]_x = \overline{f} + \sum_{j=1}^{n} (1-2p)^j \left(\Omega_{2j}(x) - \overline{\Omega_{2j}}\right)$$

$$f = \Omega_2 + \Omega_4 + \Omega_6$$



p≈0.23 → maximum expectation

p=1/2 → start from scratch

The traditinal p=1/n could be around here

Example

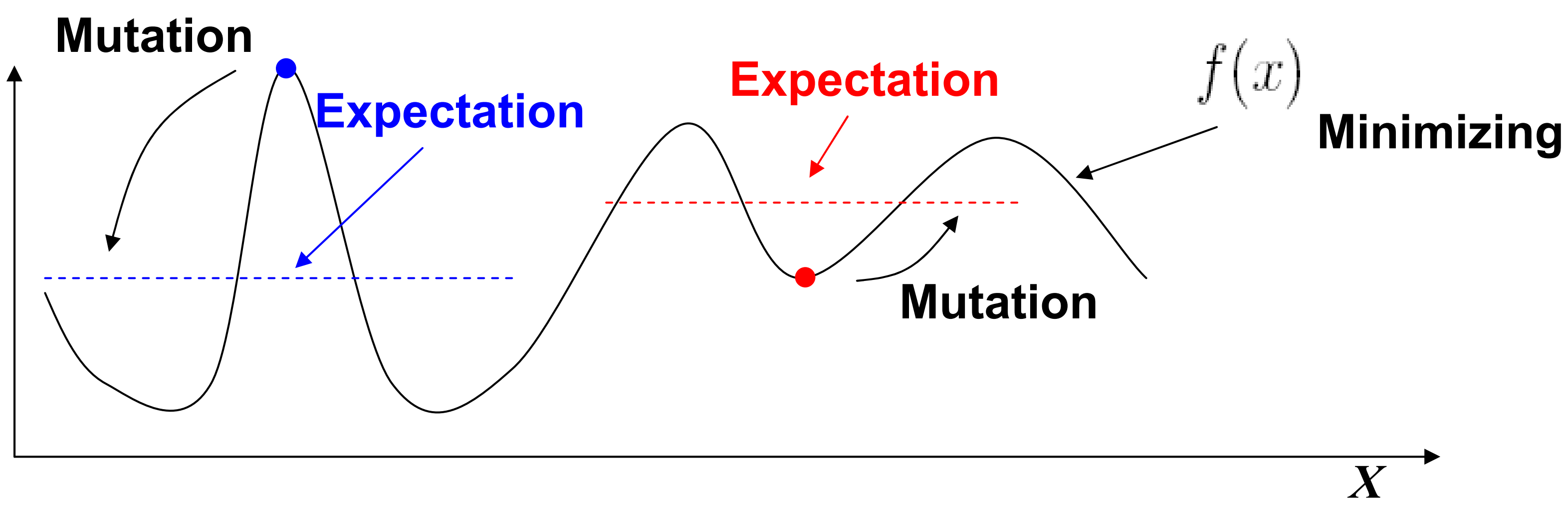
- **One example: ONEMAX**

$$\mathbb{E}[onemax]_x = np + (1-2p)\sum_{i=1}^{n} x_i$$

- **For p=1/2: E[onemax]$_x$ = n/2**
- **For p=0: E[onemax]$_x$ = ones(x)**
- **For p=1: E[onemax]$_x$ = n – ones(x)**
- **For p=1/n: E[onemax]$_x$ = 1 + (1-2/n) ones(x)**

Selection Operator

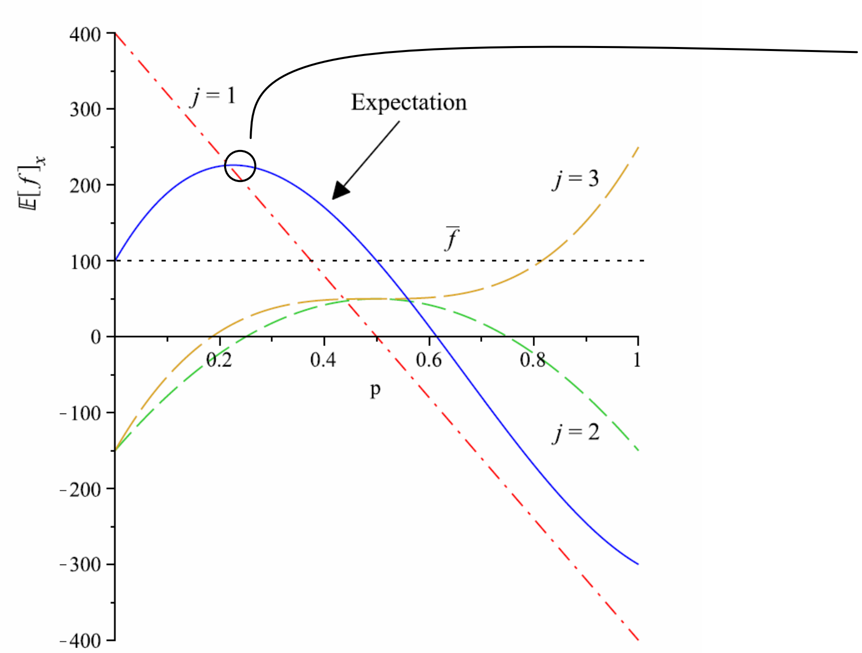
- **Selection operator**



- **We can design a selection operator selecting the individuals according to the expected fitness value after the mutation**

Mutation Operator

- **Mutation operator**
- **Given one individual *x*, we can compute the expectation against *p***



1. **Take the probability *p* for which the expectation is maximum**
2. **Use this probability to mutate the individual**

- **If this operator is used the expected improvement is maximum in one step**

(see the paper by Sutton, Whitley and Howe in the GA track!)

Other Applications

- **The expected fitness after mutation can be applied in:**
  - **Genetic Algorithms: traditional bip-flip mutation operator**
  - **Simulated Annnealing: neighborhood**
  - **Iterated Local Search: perturbation procedure**
  - **Variable Neighborhood Search: perturbation operator**
  - **Runtime analysis: for one iteration of the previous algorithms**

Conclusions & Future Work

Conclusions

- We give a very simple and efficient formula for the expected fitness after bit-flip mutation
- In elementary landscapes the dependence of the expectation with *p* is a simple monomial centred in ½
- The result is generalized to any arbitrary function, provided that we have the elementary landscape decomposition.
- Using the expected fitness, new operators can be designed (two examples are outlined)

Future Work

- Expressions not only for the expected value, but for higher-order moments (variance, skewness, kurtosis, etc.)
- Expressions for probability of improvement after a bit-flip mutation
- Design new operators and search methods based on the results of this work
- Connection with runtime analysis

Thanks for your attention !!!