CIS
RESEARCH GROUP

Heuristics & Metaheuristics for Optimization & Learning

Mario Pavone

mpavone@dmi.unict.it

http://www.dmi.unict.it/mpavone/

AA. 2024/2025 - LM 18

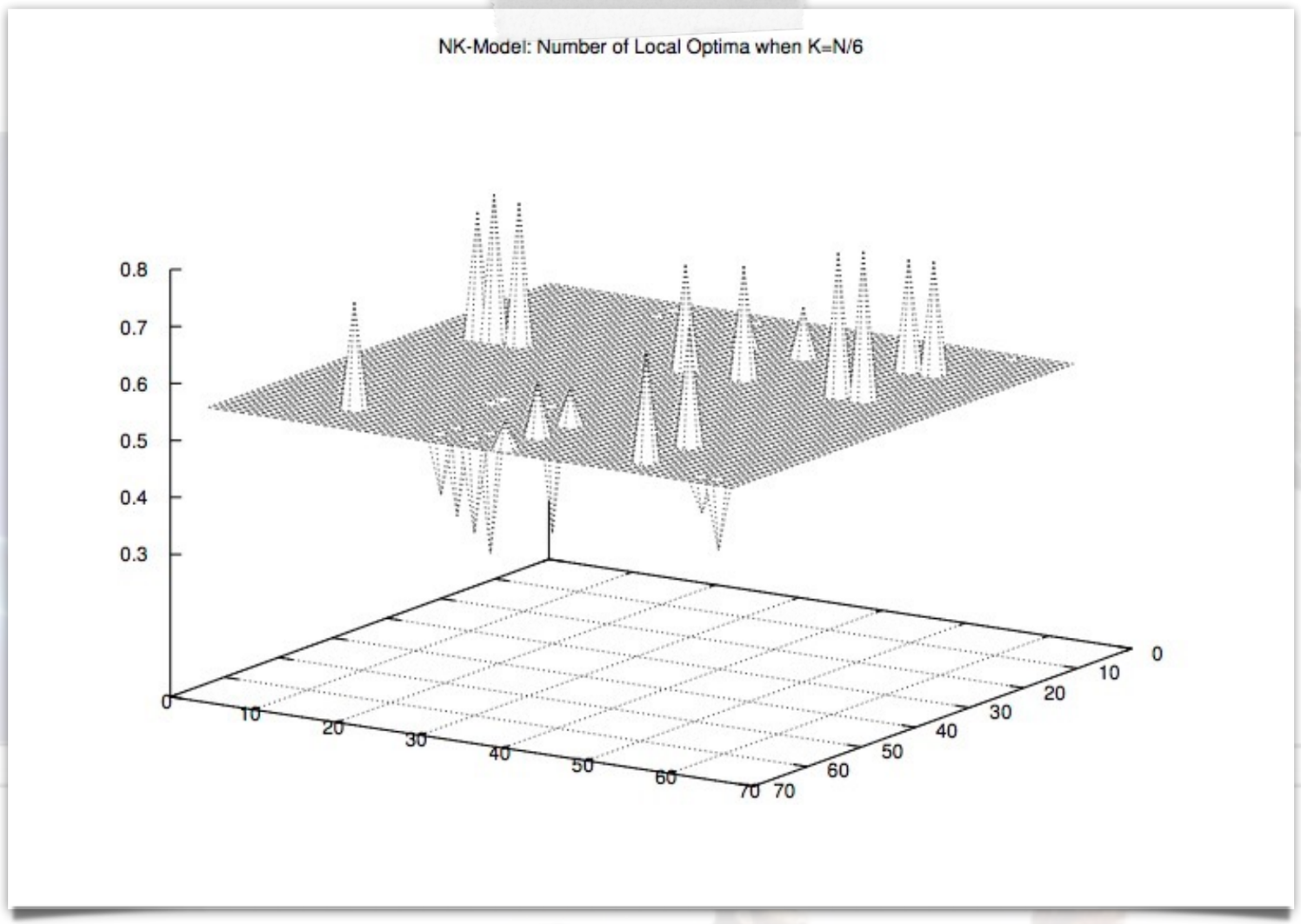
Success Key: how the search space is visited

Search Space

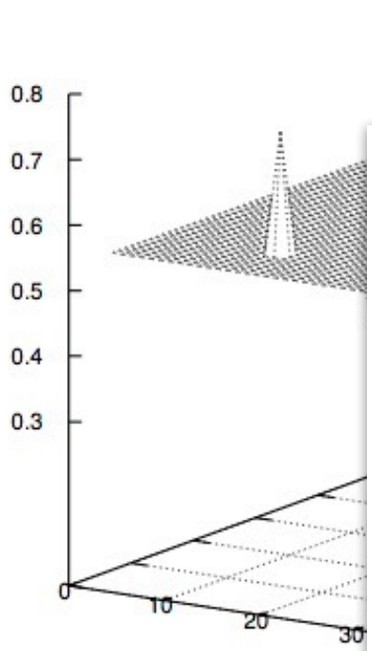
Let we consider F = (A ∨ ¬B) ∧ (¬ A ∨ ¬B)

A	B	$f_{sat}(F,s)$
T	T	1
T	F	2
F	T	1
F	F	2

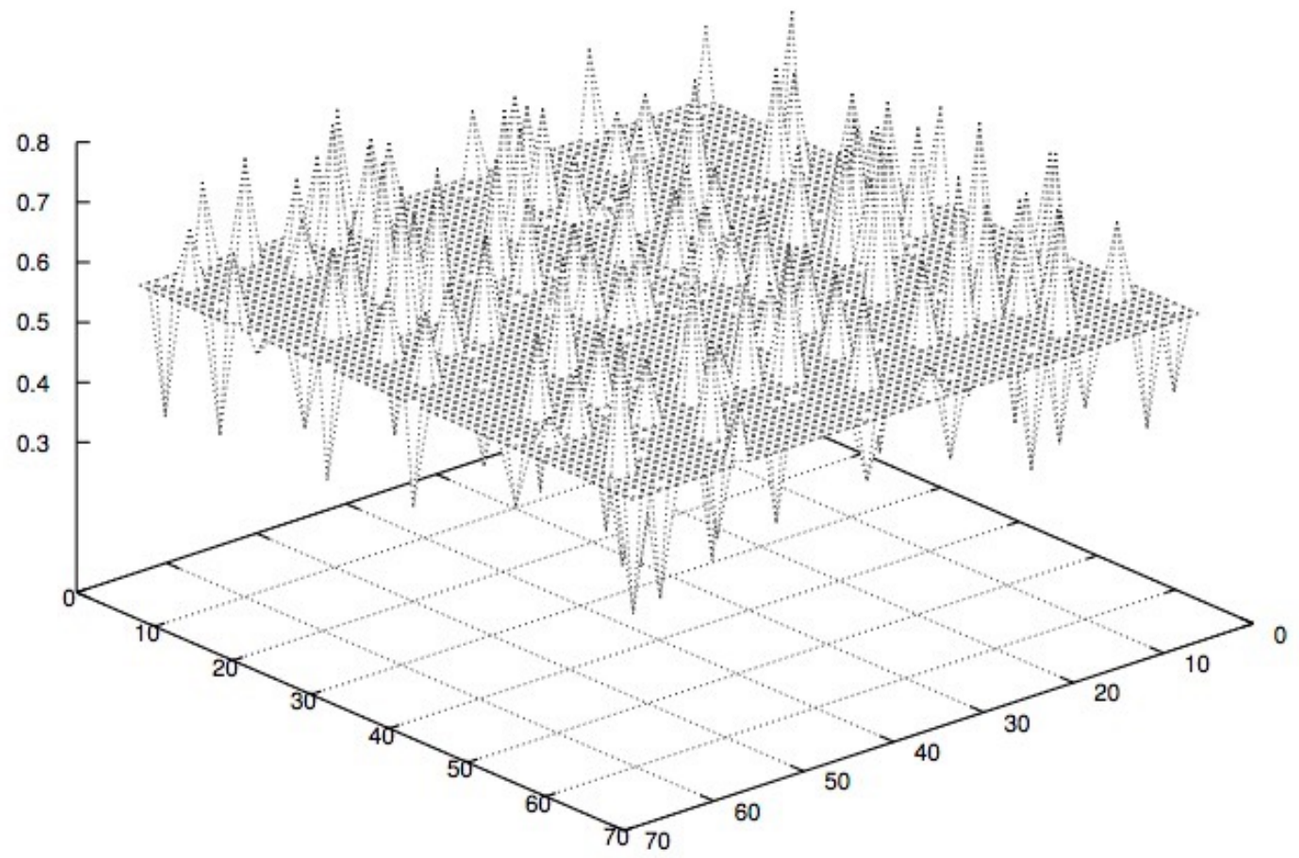
Energy level	No. of Conformations
0	36.098.079
-1	31.656.934
-2	12.473.446
-3	2.934.974
-4	517.984
-5	77.080
-6	10.364
-7	1194
-8	96
-9	**4**
Total	**83.779.155**


NK-Model: Number of Local Optima when K=N/6
0.8
0.7
0.6
0.5
0.4
0.3
0
10
20
30
40
50
60
70
0
10
20
30
40
50
60
70

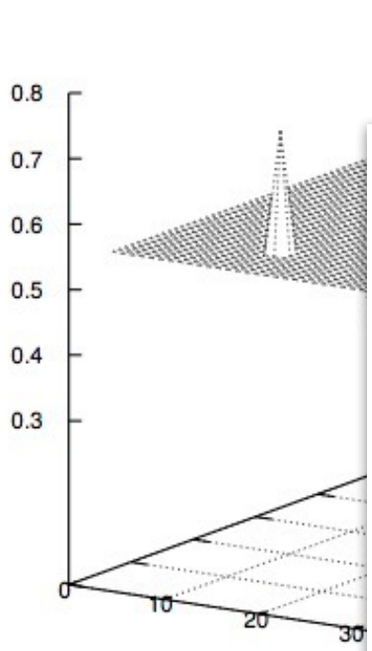
NK-Model: Number of Local Optima when K=N/6



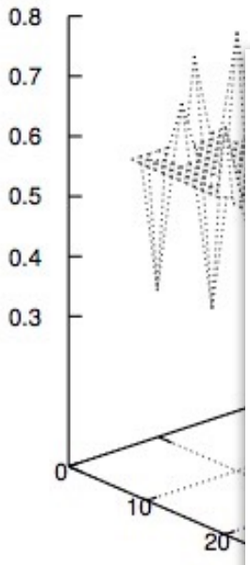
NK-Model: Number of Local Optima when K=N/2



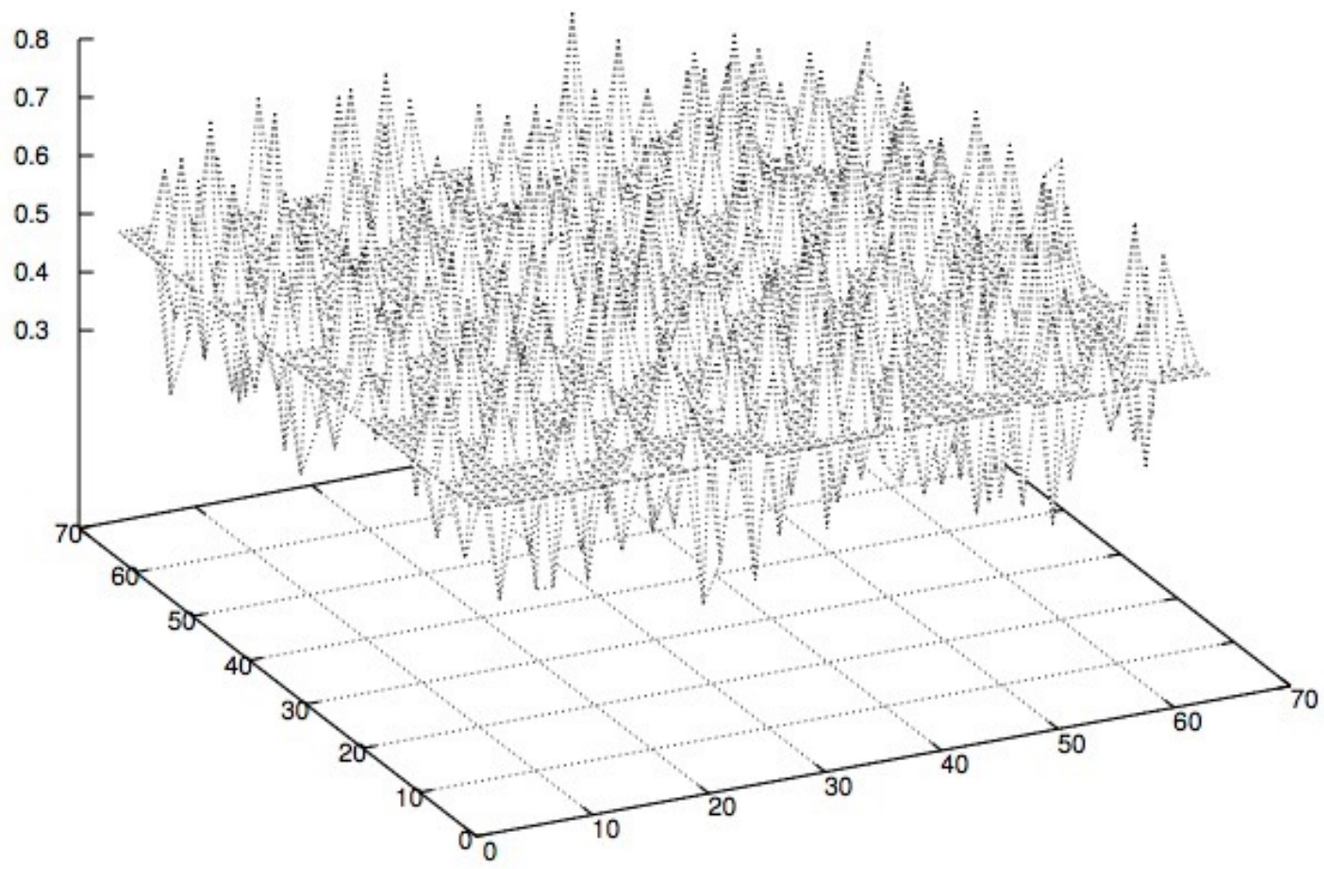
NK-Model: Number of Local Optima when K=N/6



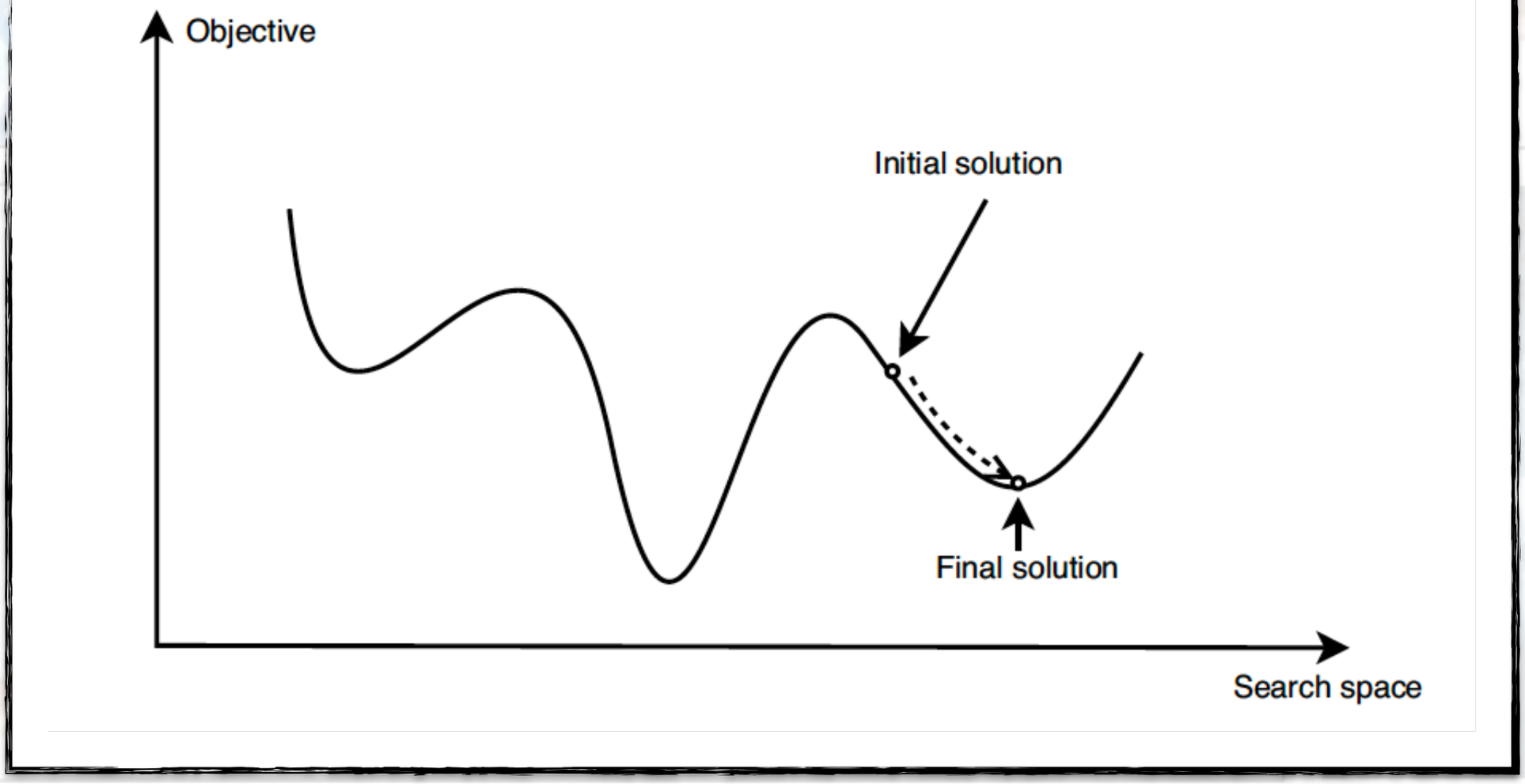
NK-Model: Number of Local Optima when K=N/2



NK-Model: Number of Local Optima when K=N-1



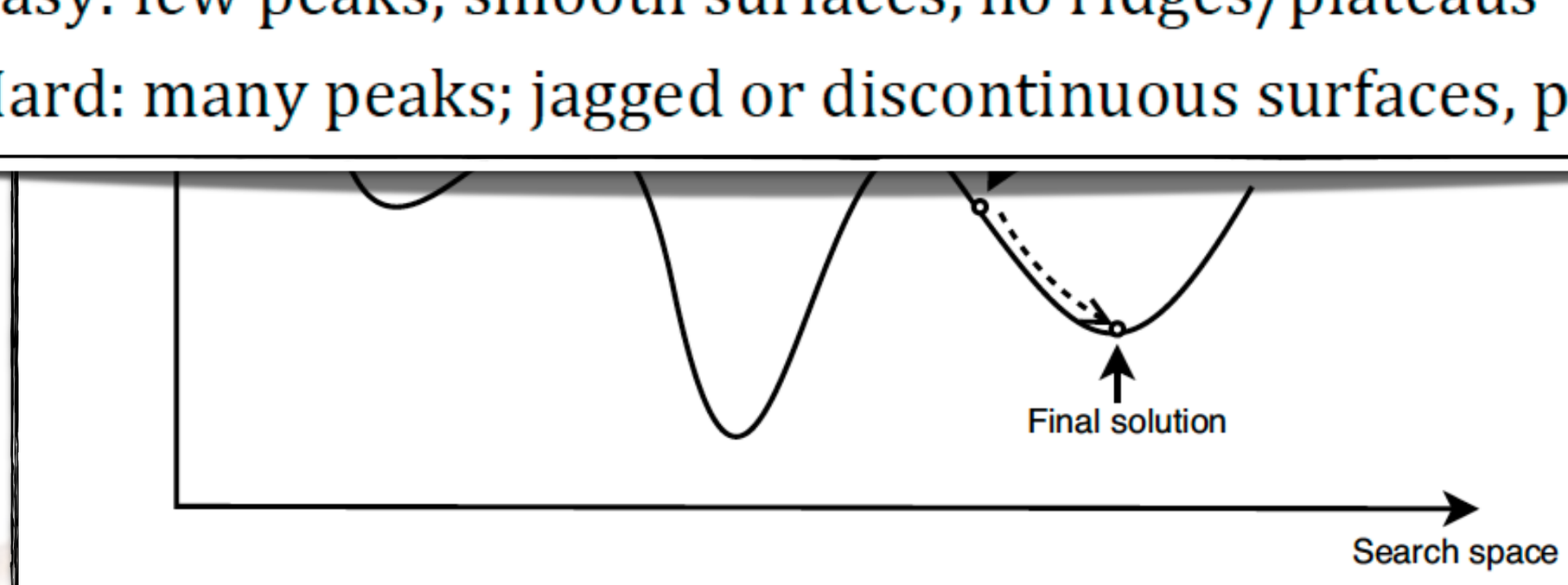
Local Optima vs Global Optima



Local Optima vs Global Optima

Easy Problems, and Hard Problems:

- Easy: few peaks; smooth surfaces, no ridges/plateaus
- Hard: many peaks; jagged or discontinuous surfaces, plateaus



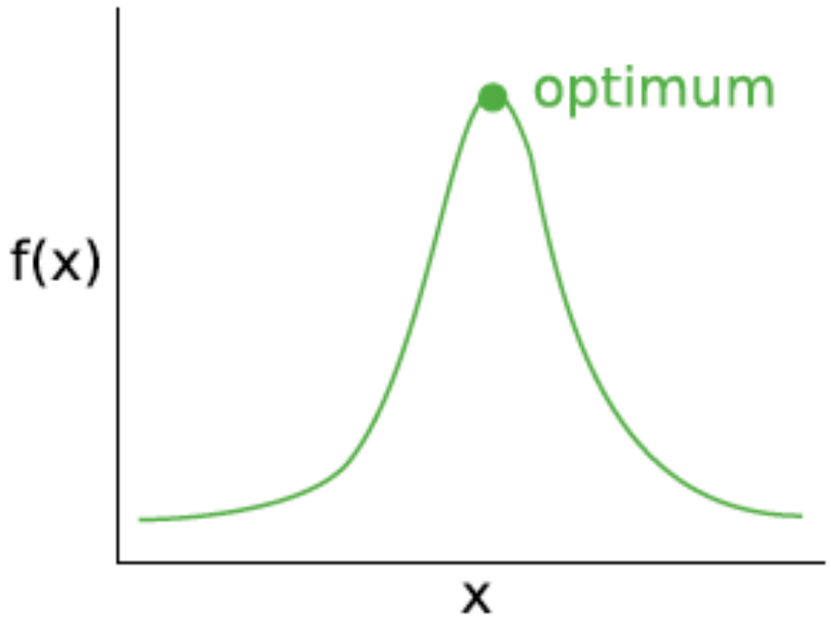
Local Optima vs Global Optima

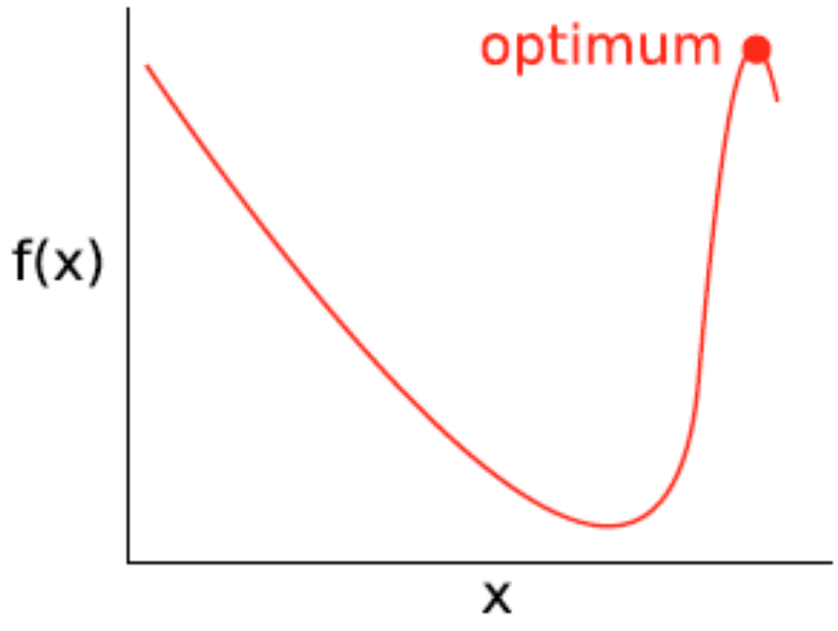
Simple functions

- Being closer to a global optimum leads to higher fitness.
- Climbing directly to the optimum is rather easy.

Difficult functions

- Being closer to a global optimum may lead to lower fitness.
- Climbing directly to the optimum is not that easy.





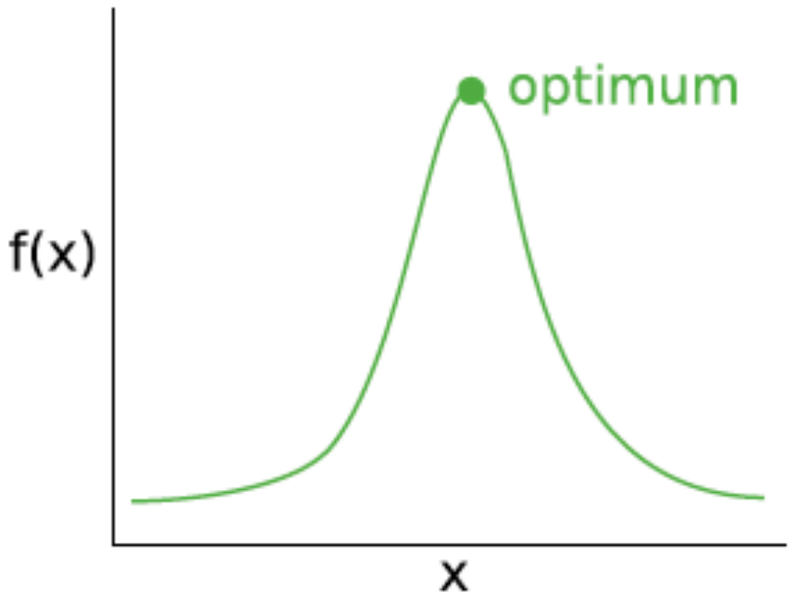
Local Optima vs Global Optima

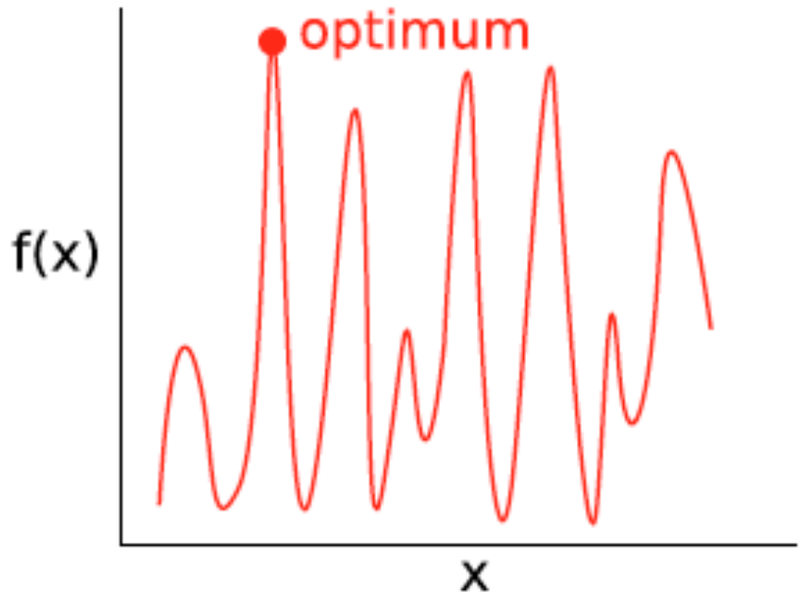
Simple functions

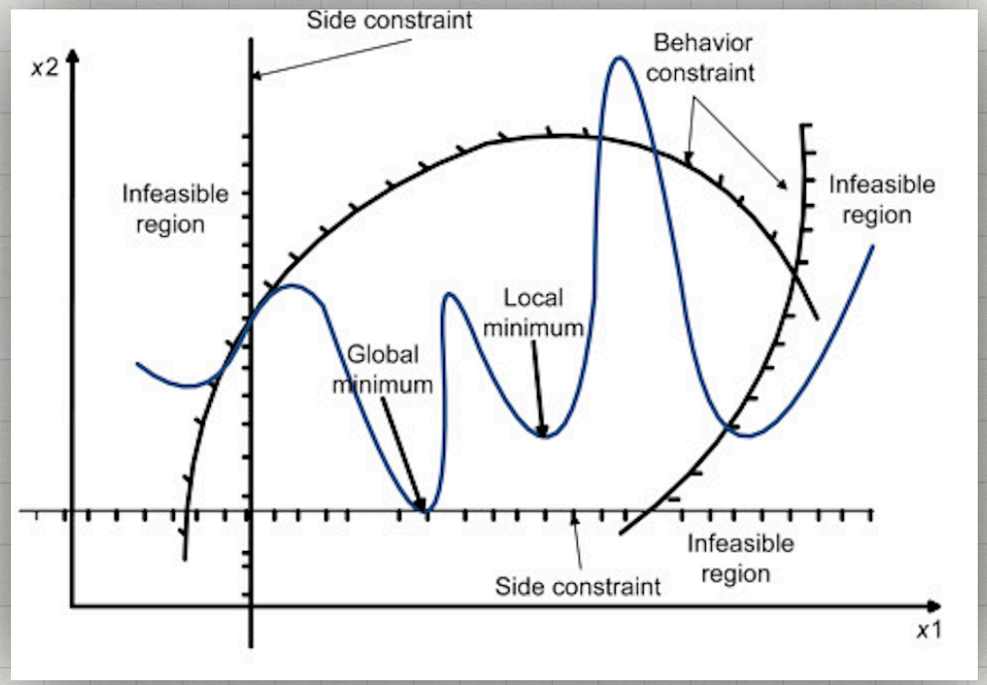
- Solutions close to each other have similar fitness values.
- Fitness differences increase slowly with distance.

Difficult functions

- Solutions close to each other have different fitness values.
- Fitness differences increase fast with distance.

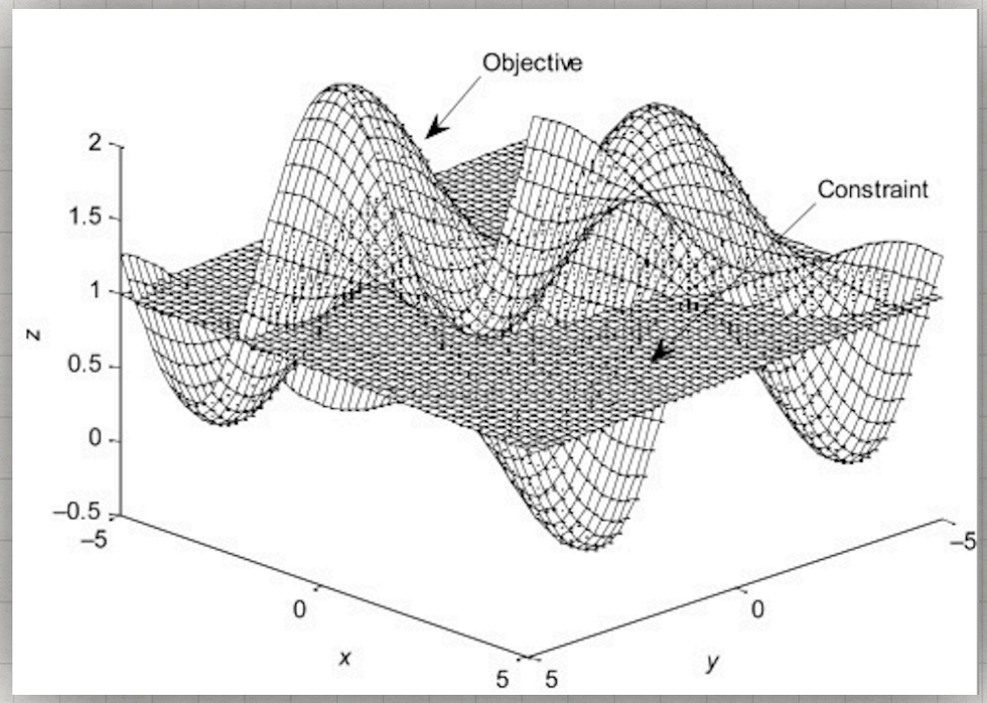




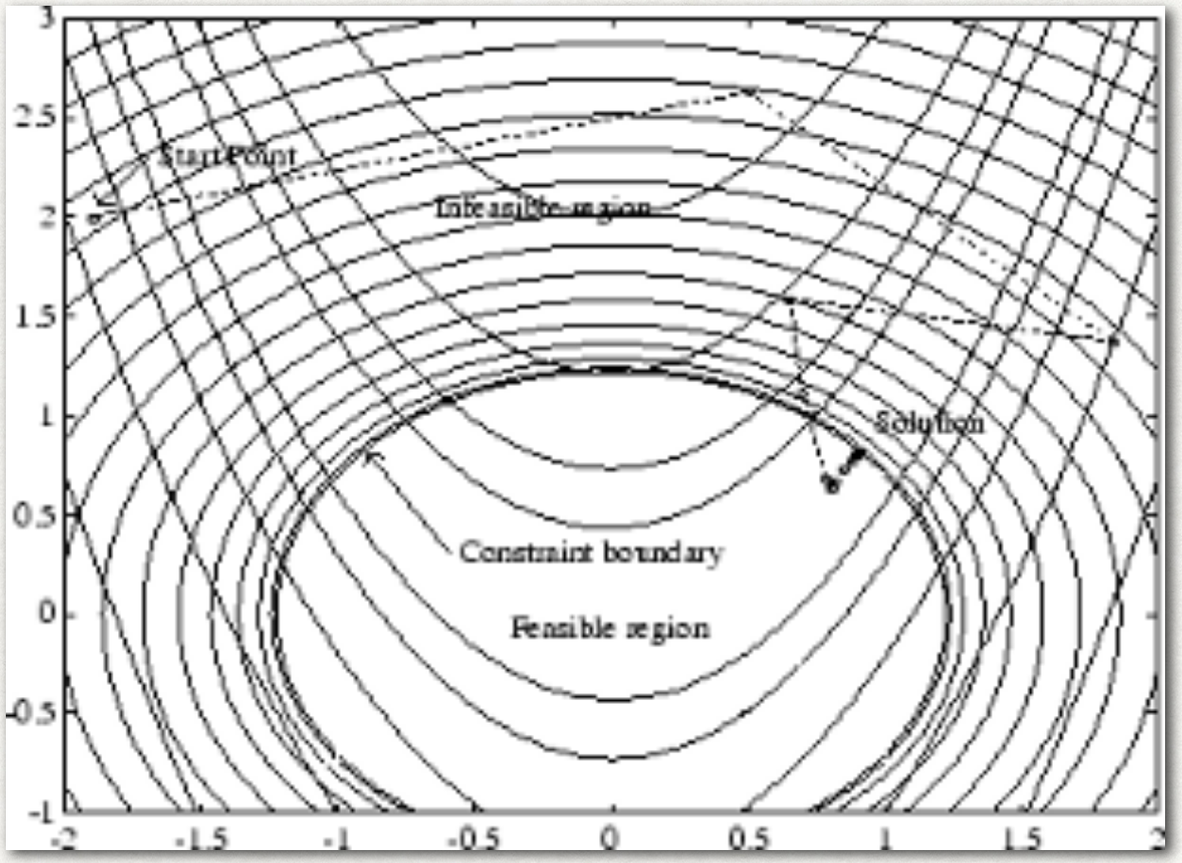


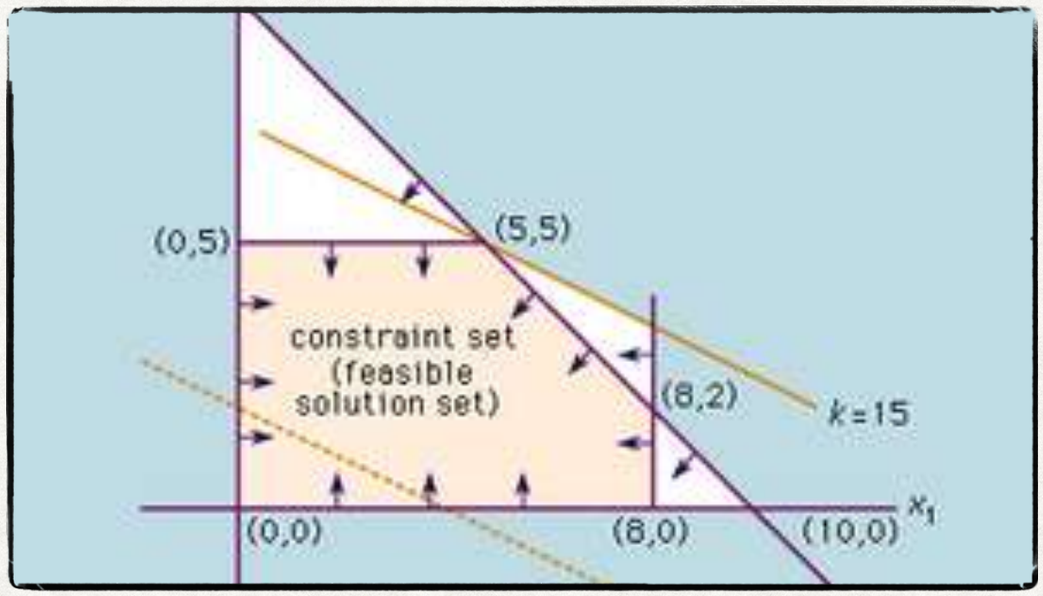
Constraint Optimization

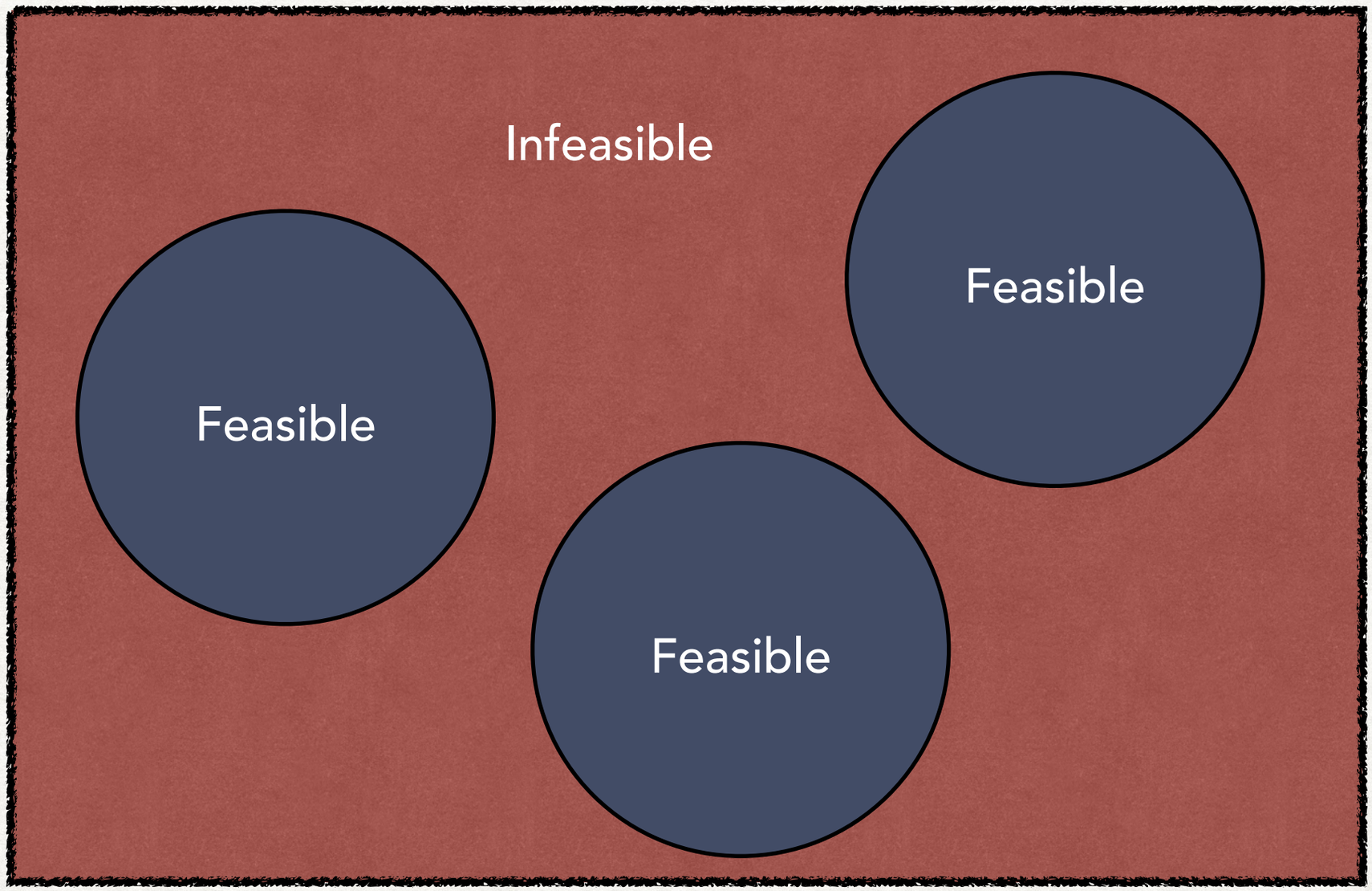
Problems



Search Space Vs. Solutions Space







S=Search Space

WHAT IS CONSTRAINED OPTIMIZATION

The general problem we considered here can be described as:

$$\min_{\mathbf{x}}\{f(\mathbf{x})\}$$

subject to

$$g_i(\mathbf{x}) \leq 0, \quad i = 1, 2, \ldots, m$$

$$h_j(\mathbf{x}) = 0, \quad j = 1, 2, \ldots, p$$

where $\mathbf{x}$ is the n dimensional vector, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$; $f(\mathbf{x})$ is the objective function; $g_i(\mathbf{x})$ is the inequality constraint; and $h_j(\mathbf{x})$ is the equality constraint.

Denote the whole search space as $\mathcal{S}$ and the feasible space as $\mathcal{F}$, $\mathcal{F} \subset \mathcal{S}$. It is important to note that the global in $\mathcal{F}$ might not be the same as that in $\mathcal{S}$.

CONSTRAINT HANDLING TECHNIQUES

The penalty function approach converts a constrained problem into an unconstrained one by introducing a penalty function into the objective function.

The repair approach maps (repairs) an infeasible solution into a feasible one.

The purist approach rejects all infeasible solutions in search.

The separatist approach considers the objective function and constraints separately.

The hybrid approach mixes two or more different constraint handling techniques.

PENALTY FUNCTION APPROACH

NewObjectiveFunction = OriginalObjectiveFunction +
PenaltyCoefficient * DegreeOfConstraintViolation

The general form of the exterior penalty function method:

$$\psi(\mathbf{x}) = f(\mathbf{x}) + \left(\sum_{i=1}^{m} r_i G_i(\mathbf{x}) + \sum_{j=1}^{p} c_j H_j(\mathbf{x}) \right),$$

where $\psi(\mathbf{x})$ is the new objective function to be minimised, $f(\mathbf{x})$ is the original objective function, r_i and c_j are penalty factors (coefficients), and

$$G_i(\mathbf{x}) = \left(\max(0, g_i(\mathbf{x}))\right)^{\beta}, \quad H_j(\mathbf{x}) = \max\left(0, |h_j(\mathbf{x})|^{\gamma}\right),$$

β and γ are usually chosen as 1 or 2.

PENALTY FUNCTION APPROACH

Static Penalties The penalty function is pre-defined and fixed during evolution.

Dynamic Penalties The penalty function changes according to a pre-defined sequence, which often depends on the generation number.

Adaptive and Self-Adaptive Penalties The penalty function changes adaptively. There is no fixed sequence to follow.

STATIC PENALTY FUNCTIONS

$$\psi(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^{m} r_i \left(G_i(\mathbf{x})\right)^2$$

where r_i's are pre-defined and fixed.

- Equality constraints can be converted into inequality ones:

  $$h_j(\mathbf{x}) \Longrightarrow |h_j(\mathbf{x})| - \varepsilon \leq 0$$

  where $\varepsilon > 0$ is a small number.
- Simple and easy to implement.
- Requires rich domain knowledge to set r_i's.
- r_i's can be divided into a number of different levels. When to use which is determined by a set of heuristic rules.

DYNAMIC PENALTY FUNCTIONS

General principle: The large the generation number, the larger the penalty coefficient. *Why?*

Joines and Houck's method:

$$\psi(\mathbf{x}) = f(\mathbf{x}) + (Ct)^{\alpha} \left(\sum_{i=1}^{m} |g_i(\mathbf{x})|^{\beta} + \sum_{j=1}^{p} |h_j(\mathbf{x})|^{\beta} \right),$$

where C, α and β are constants defined by the user, e.g., $C = 0.5, \alpha = 1, 2, \beta = 2$, t is the generation number.

- Many different forms of dynamic penalties are possible, e.g., $\frac{t}{T}$. The best one can be difficult to find.
- Different coefficients can be used for inequality and equality constraints.

DYNAMIC PENALTY FUNCTIONS

$$\psi(\mathbf{x}) = f(\mathbf{x}) + r(t)\sum_{i=1}^{m} G_i^2(\mathbf{x}) + c(t)\sum_{j=1}^{p} h_j^2(\mathbf{x}),$$

where $r(t)$ and $c(t)$ are two penalty coefficients.

Polynomials a_i and b_i are user-defined parameters.

$$r(t) = a_0 + a_1 t + a_2 t^2 + \cdots, \quad c(t) = b_0 + b_1 t + b_2 t^2 + \cdots.$$

Exponentials a and b user-defined parameters.

$$r(t) = e^{at}, \quad c(t) = e^{bt}.$$

Hybrid

$$r(t) = e^{a_0 + a_1 t + a_2 t^2 + \cdots}, \quad c(t) = e^{b_0 + b_1 t + b_2 t^2 + \cdots}.$$

ADAPTIVE PENALTIES

Bean and Hadj-Alouane's method:

$$\psi(\mathbf{x}) = f(\mathbf{x}) + \lambda(t) \left(\sum_{i=1}^{m} G_i^2(\mathbf{x}) + \sum_{j=1}^{p} |h_j(\mathbf{x})| \right),$$

where λ is updated at generation t as follows:

$$\lambda(t) = \begin{cases} \frac{1}{\beta_1}\lambda(t), & \text{if case 1,} \\ \beta_2\lambda(t), & \text{if case 2,} \\ \lambda(t), & \text{otherwise,} \end{cases}$$

where case 1 (or 2) indicates that the best individual in the last k generations was always feasible (or infeasible), $\beta_2 > \beta_1 > 1$.

FITNESS FUNCTION & SELECTION

- Let $\Phi(\mathbf{x}) = f(\mathbf{x}) + rG(\mathbf{x})$, where $G(\mathbf{x}) = \sum_{i=1}^{m} G_i(\mathbf{x})$ and $G_i(\mathbf{x}) = \max\{0, g_i(\mathbf{x})\}$.
- Given two individuals $\mathbf{x}_1$ and $\mathbf{x}_2$. Their fitness values will now be determined by $\Phi(\mathbf{x})$.
- Because fitness values are used primarily in selection,

  Changing fitness $\iff$ changing selection probabilities.

When r Plays an Important Role

$$\Phi(\mathbf{x}_1) < \Phi(\mathbf{x}_2)$$

means

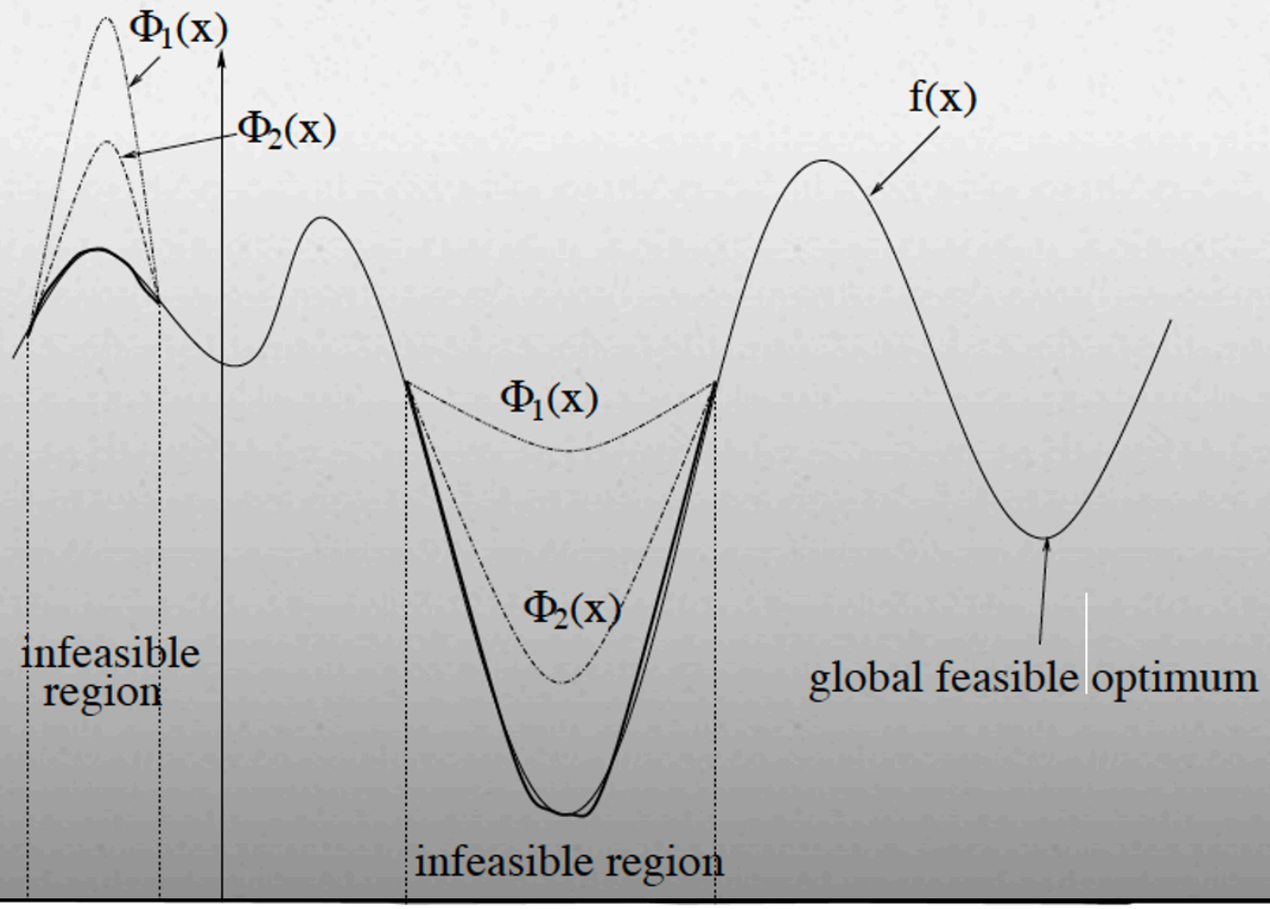
$$f(\mathbf{x}_1) + rG(\mathbf{x}_1) < f(\mathbf{x}_2) + rG(\mathbf{x}_2).$$

1. $f(\mathbf{x}_1) < f(\mathbf{x}_2)$ and $G(\mathbf{x}_1) < G(\mathbf{x}_2)$: r has no impact on the comparison.
2. $f(\mathbf{x}_1) < f(\mathbf{x}_2)$ and $G(\mathbf{x}_1) > G(\mathbf{x}_2)$: Increasing r will eventually change the comparison.
3. $f(\mathbf{x}_1) > f(\mathbf{x}_2)$ and $G(\mathbf{x}_1) < G(\mathbf{x}_2)$: Decreasing r will eventually change the comparison.

In essence, different r's lead to different rankings of individuals in the population.

NewObjectiveFunction = OriginalObjectiveFunction +
PenaltyCoefficient * DegreeOfConstraintViolation

Different penalty functions lead to different new objective functions.

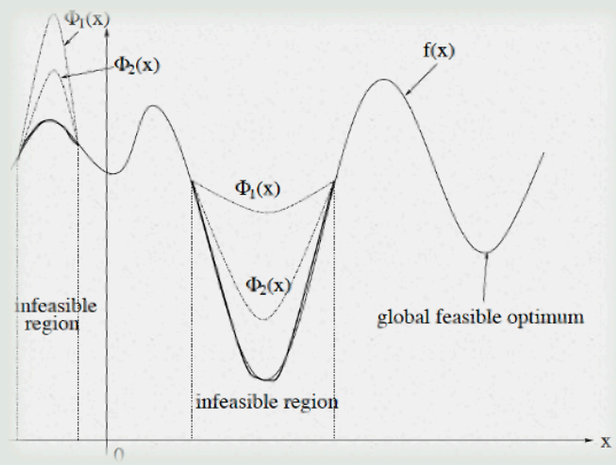


Penalties & Fitness Landscape Transformation

1. $\Phi_1(\mathbf{x})$ on the pevious page is well transformed because the global optimum of $\Phi_1(\mathbf{x})$ is the feasible optimum we want.
2. $\Phi_2(\mathbf{x})$ is poorly transformed because its global optimum is infeasible. This is a typical case of *under-penalisation*.

Does it mean the penalty function should be really large?

Well Transformed Objective Functions

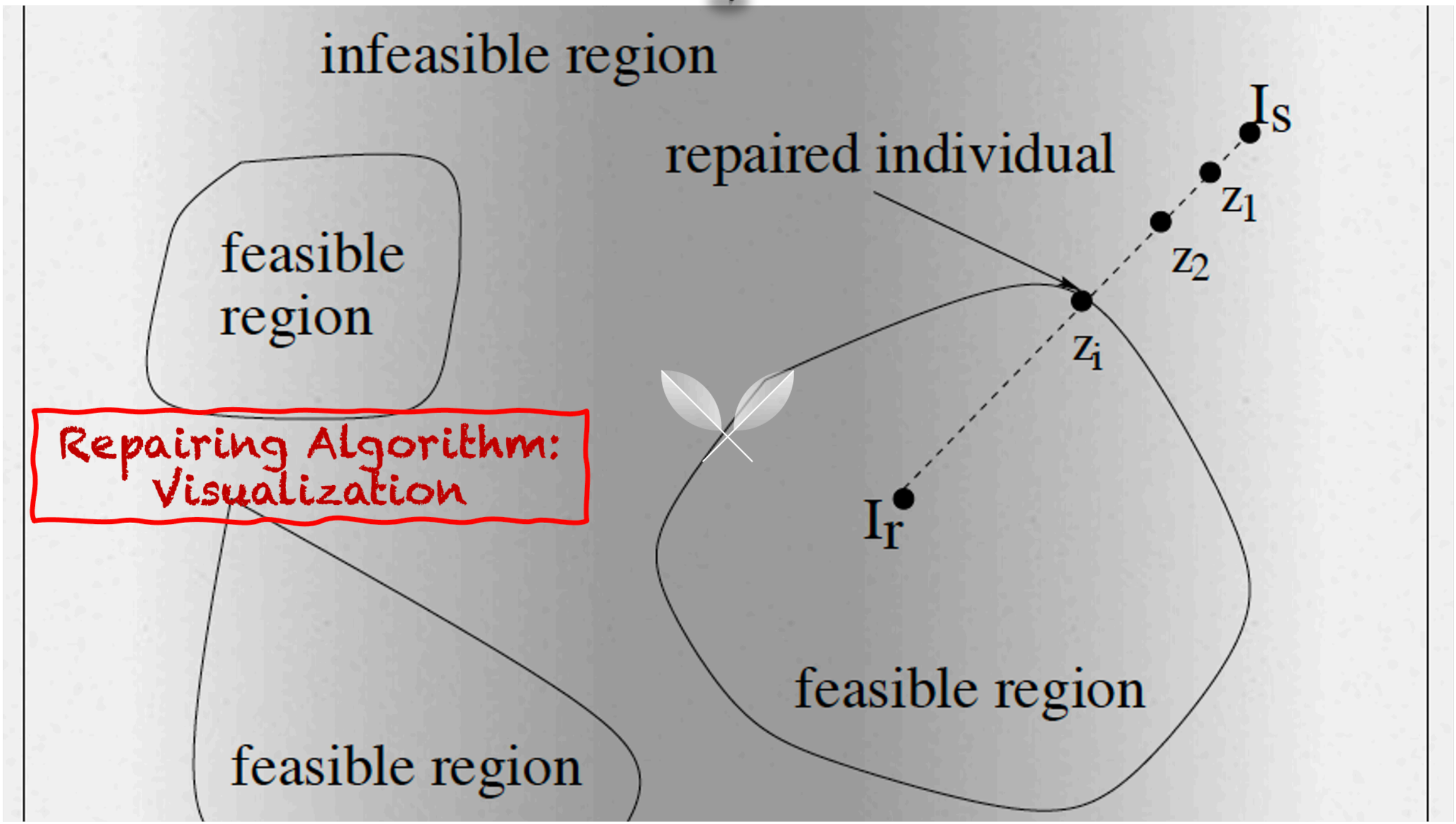


Repairing Algorithm

1. Select a reference individual I_r.
2. Create a sequence of candidate individuals z_i between I_s and I_r:

   $$z_i = a_i I_s + (1 - a_i) I_r,$$

   where $0 < a_i < 1$ can be generated at random or deterministically. The process of creating z_i stops when z_i is feasible (i.e. ,when the first feasible z_i is found).
3. Let this z_i be the repaired individual of I_s. Its objective function value will be used as I_s's fitness value.
4. Replace I_s by z_i with probability P_r. (Even if I_s is not replaced by z_i, its fitness is still that of z_i's.)
5. If z_i is better than I_r, replace it.


infeasible region
feasible region
repaired individual
I_s
z_1
z_2
z_i
I_r
feasible region
feasible region
Repairing Algorithm: Visualization

Repairing Algorithm: Implementation Issues

How to find initial reference individuals? 1. Preliminary exploration

2. Human knowledge

How to select I_r? 1. Uniformly at random

2. According to the fitness of I_r

3. According to the distance between I_r and I_s

How to determine a_i? 1. Uniformly at random between 0 and 1

2. Using a fixed sequence, e.g., $\frac{1}{2}, \frac{1}{4}, \ldots$

How to choose P_r? A small number, usually < 0.5.