Constraint Handling

Fernando Lobo

University of Algarve

Outline

- Introduction
- Penalty methods
- Approach based on tournament selection
- Decoders
- Repair algorithms
- Constraint-preserving operators

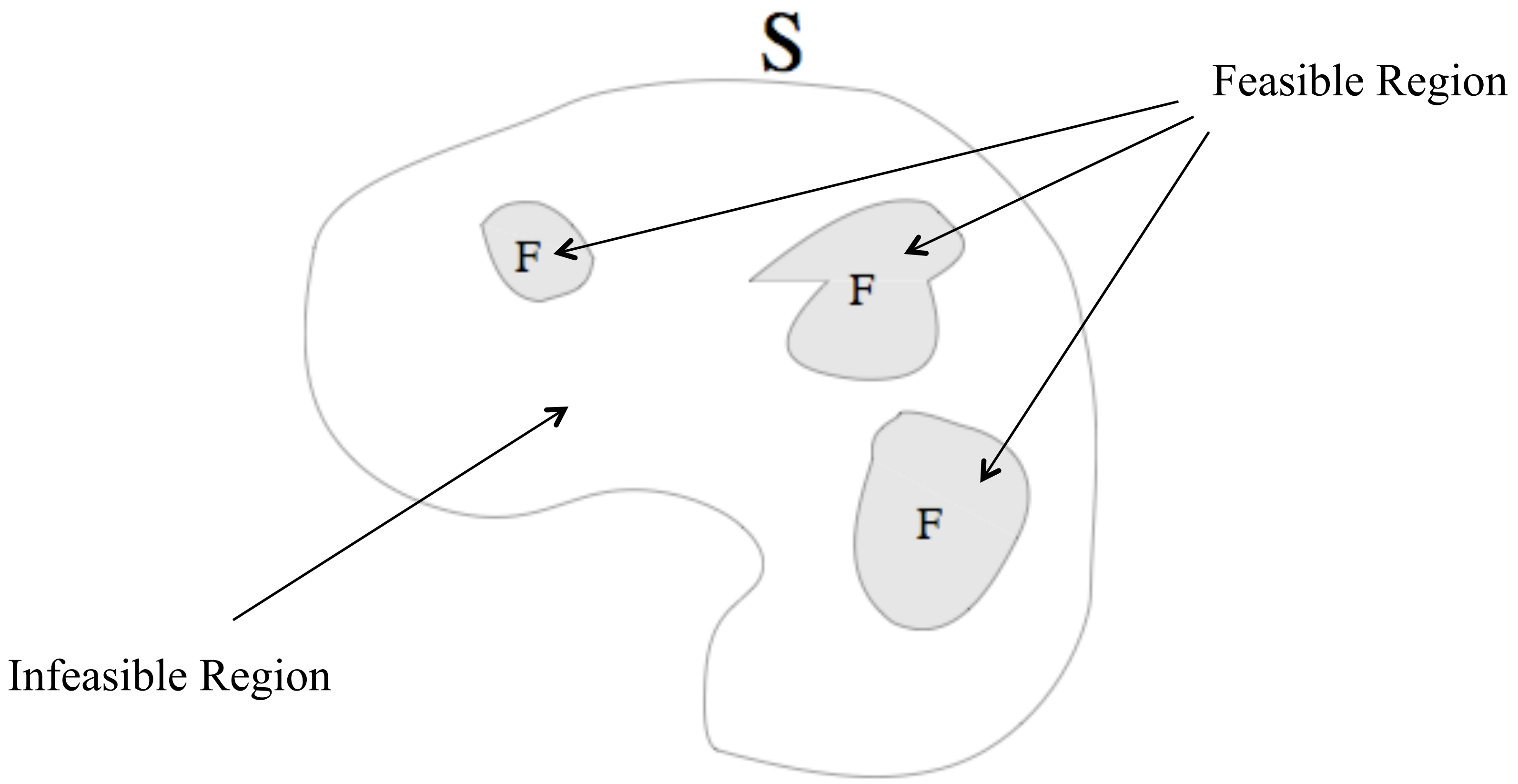
Introduction

- So far we have only looked at unconstrained optimization problems.
- But most real world problems have constraints that must be satisfied.
- In this lecture we will look at techniques that can be used to address constraints within an EA.

Introduction

- Naïve approach: If a solution is infeasible, discard it.
- Problem:
  - In many cases, the feasible region of the search space is very small.
  - We need to use the information from infeasible solutions to guide the search towards feasible regions.
- NOTE: The feasible region often consists of several disconnected, non-convex, regions.

Search Space



Note: Image adapted from Carlos Coello’s tutorial presented at GECCO 2010

General definition of a constrained optimization problem

$$\begin{aligned}
&\text{Minimize} && f(\vec{x}) \\
&\text{Subject to} && g_j(\vec{x}) \geqslant 0, \quad j = 1, \ldots, J, \\
& && h_k(\vec{x}) = 0, \quad k = 1, \ldots, K, \\
& && x_i^l \leqslant x_i \leqslant x_i^u, \quad i = 1, \ldots, n.
\end{aligned}$$

- Objective function defined over n variables, x_1, x_2, …, x_n
- Each x_i has a value within a given range.
- J inequality constraints.
- K equality constraints.

Penalty methods

- Idea:
  - Penalize infeasible solutions by adding a penalty term to the objective function value (for minimization problems).

$$\underbrace{F(\vec{x})}_{\text{Fitness value}} = \underbrace{f(\vec{x})}_{\text{Objective function value}} + \underbrace{\sum_{j=1}^{J} R_j \langle g_j(\vec{x}) \rangle^2}_{\text{Penalty term}}$$

Penalty methods (cont.)

$$F(\vec{x}) = f(\vec{x}) + \sum_{j=1}^{J} R_j \langle g_j(\vec{x}) \rangle^2$$

- R_j is the penalty parameter of the jth inequality constraint.
- Purpose of it is to make the constraint violation of the same order of magnitude as the objective function value.
- $\langle \, \rangle$ is the absolute value of the operand if operand is negative, and zero otherwise.

Equality constraints

- Equality constraints are usually transformed into inequality constraints as follows:

$$g_{k+J}(\vec{x}) \equiv \delta - |h_k(\vec{x})| \geqslant 0$$

- where δ is a very small number.

- Number of inequality constraints becomes $m = J + K$

- With m constraints we need m R_j' s (j=1…m) penalty parameters.

- By normalizing the constraints, only one parameter R is needed.

- Normalization makes all constraint violations of the same order of magnitude.

Problems with the penalty approach

- Problems with the approach:
  - Parameter R (or R_j' s) difficult to select.
  - User has to try different values and see what works best.

Different penalty method strategies

- Static
  - Penalty term is constant through time.

- Dynamic
  - Penalty term depends on the phase of evolution (generation number).
  - The idea is to increase the penalty term through time because at the end of the search we don’t want to have infeasible solutions in the population.

Different penalty method strategies (cont.)

- Adaptive
  - Penalty term changes according to the feasibility or infeasibility of the best solution during the most recent *k* generations.
    - Penalty <u>increases</u> if in the previous *k* generations, the best solution was always infeasible.
    - Penalty <u>decreases</u> if in the previous *k* generations, the best solution was always feasible.
    - Penalty <u>stays the same</u> otherwise.

Kalyanmoy Deb’s proposal based on tournament selection

- An elegant solution that does not require penalty parameters.

- Idea based on binary tournament selection. 3 cases:
    1. If the two solutions are feasible, choose the one with the best objective function value.
    2. If one is feasible and the other infeasible, choose the feasible one.
    3. If both are infeasible, choose the one that violates less the constraints.

Decoders

- The use of decoders is another approach to handle constraints.
- Decoders interpret the chromosome of an individual in such a way that a feasible solution is always constructed.

Decoders (cont.)

- Example for the 0-1 knapsack problem
  - Given a set of items X_i (i=1..n), each with weight W_i and profit P_i, find a subset of items such that the total profit is maximized, and such that the total weight does not exceed a maximum capacity C.

- Traditional encoding for this problem uses bitstrings:
  - $X_i=1$ means item i is included in the knapsack.
  - $X_i=0$ means item i is not included in the knapsack.

Decoders (cont.)

- Decoder approach:
  - A sequence of items for the knapsack is interpreted as "take an item if possible" (i.e., take the item if its inclusion does not violate the capacity constraint).
  - The sequence is usually sorted in decreasing order of profit-per-weight ratio, P_i / W_i.
  - Example: 110011
    - Take the 1st item if it fits in the knapsack.
    - Take the 2nd item if it fits in the knapsack.
    - Take the 5th item …
    - Take the 6th item …

Repair algorithms

- This is another approach for handling constraints.
- A repair algorithm maps an infeasible solution into a feasible one.
- Two approaches:
  - The repair is made for evaluation purposes only.
  - The repaired individual replaces the original one in the population. (Choice can be made probabilistically)

Repair algorithms (cont.)

- Example with the 0-1 knapsack problem:
  - If solution is infeasible, keep removing items from the knapsack until the capacity constraint is not violated.

Constraint-preserving operators

- The idea is to design genetic operators so that there is the guarantee that only feasible solutions are produced.

- We have seen examples of this earlier in the course with the permutation operators for ordering problems.