CO3091 - Computational Intelligence and Software Engineering

Lecture 06

Image from: http://images.fineartamerica.com/images-medium-large/restriction-ashish-kumar-rudra.jpg

Constraints Handling

Leandro L. Minku

Announcements

Spare handouts in my office!

Overview

- What is a constraint?
- How to deal with constraints?

What is a Constraint?

A **condition** that **must** be satisfied by a solution.

What is a Constraint?

- Optimisation problem:
  - Maximise / minimise $f(x)$, where x is the design variable
  - Subject to:
    - $g_i(x) \leq 0$, where $i=1,\dots,n$
      [inequality constraints]
    - $h_j(x) = 0$, where $j=1,\dots,p$
      [equality constraints]

We will refer to both inequality and equality constraints by $\phi_i(x)$, $i = 1,\dots,m$, where $m = n + p$ is the total number of constraints.

A solution that satisfies all constraints is called **feasible**.

A solution that does not satisfy one or more constraints is called **infeasible**.

Example

- Consider the following problem:
  - You need to load a lorry with products. The **maximum** total weight of products that the lorry can stand is W.
  - You have N products that can be loaded, and each product i has a weight w_i, and a profit p_i.
  - You would like to decide which products to load so as to **maximise** the total profit of loaded products.

Example

- Problem formulation:
  - Design variable: $v \in \{0,1\}^N$, where 0 represents not loaded and 1 represents loaded.
  - Objective function:

$$f(v) = \sum_{i=1}^{N} v_i p_i \quad \text{(to be maximised).}$$

Profit of product i

  - Constraint:

$$\sum_{i=1}^{N} v_i w_i \leq W$$

Example

- Problem formulation:
  - Design variable: $v \in \{0,1\}^N$, where 0 represents not loaded and 1 represents loaded.
  - Objective function:

$$f(v) = \sum_{i=1}^{N} v_i \, p_i \quad \text{(to be maximised).}$$

0 if product i is not loaded
1 otherwise

  - Constraint:

$$\sum_{i=1}^{N} v_i \, w_i \leq W$$

Example

- Problem formulation:
  - Design variable: $v \in \{0,1\}^N$, where 0 represents not loaded and 1 represents loaded.
  - Objective function:

$$f(v) = \sum_{i=1}^{N} v_i \, p_i$$

(to be maximised).

Total profit of loaded products

  - Constraint:

$$\sum_{i=1}^{N} v_i w_i \leq W$$

Weight of product i

Example

- Problem formulation:
  - Design variable: $v \in \{0,1\}^N$, where 0 represents not loaded and 1 represents loaded.
  - Objective function:

$$f(v) = \sum_{i=1}^{N} v_i \; p_i \qquad \text{(to be maximised).}$$

  - Constraint:

$$\sum_{i=1}^{N} v_i \, w_i \leq W \qquad \text{equivalent to: } \left(\sum_{i=1}^{N} v_i \, w_i\right) - W \leq 0$$

Total weight of loaded products

Image from: https://s.yimg.com/pw/images/spaceout.gif

Constraints in Real World Problems

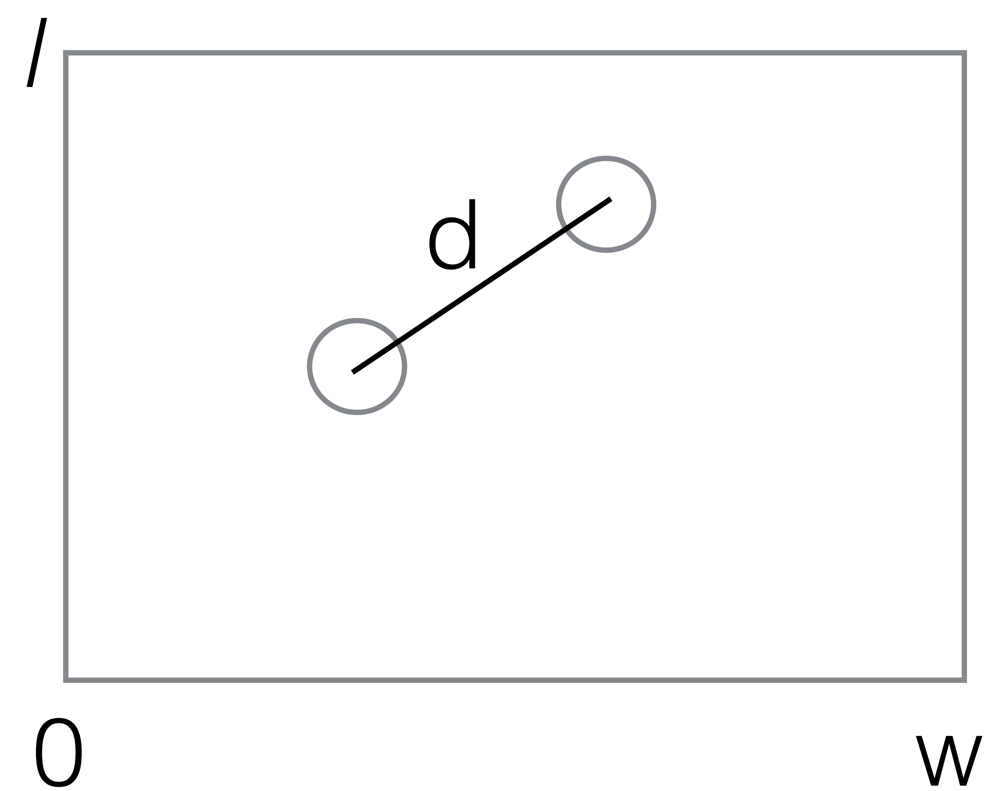
Many real world problems have constraints.

[Video posted by the Optimisation and Logistics group from the University of Adelaide: http://cs.adelaide.edu.au/~optlog/research/energy.php]

Optimising Wind Farms

- Design variable: $[(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)]$ location of N turbines, $x_i \in R$, $y_i \in R$.
- Objective: maximise total power output.
- Constraints: for all $i,j \in \{1,\ldots,N\}$ where $i \neq j$:
  - $x_i \geq 0$ $-x_i \leq 0$
  - $x_i \leq w$ equivalent to: $x_i - w \leq 0$
  - $y_i \geq 0$ $-y_i \leq 0$
  - $y_i \leq l$ $y_i - l \leq 0$
  - $\text{distance}((x_i,y_i),(x_j,y_j)) \geq C'$

    $\text{distance}((x_i,y_i),(x_j,y_j)) - C' \geq 0$ $-\text{distance}((x_i,y_i),(x_j,y_j)) + C' \leq 0$



Implicit Constraints

- Sometimes problem formulations account for constraints implicitly.
  - E.g.:

    Design variable: $x \in Z$
    Objective function: $f(x) = x^2$ (to be maximised)
    Constraints: $x \geq -15$ and $x \leq 15$

    or

    Design variable: $x \in \{-15,-14,\ldots,0,1,\ldots,15\}$
    Objective function: $f(x) = x^2$ (to be maximised)
    Constraints: none?

Implicit Constraints

- Our optimisation algorithms must be designed so that they can handle constraints, no matter if they were implicitly or explicitly considered in the problem formulation.

EA's Pseudocode

Evolutionary Algorithm

1. Initialise population with random individuals
2. Evaluate each individual (determine their fitness)
3. Repeat (until a termination condition is satisfied)

   3.1 Select parents

   3.2 Recombine parents with probability Pc

   3.3 Mutate resulting offspring with probability Pm

   3.4 Evaluate offspring

   3.5 Select survivors for the next generation

Constraints are not mentioned!

How to Deal with Constraints?

- Penalty functions.
- Maintaining only feasible solutions based on special representation and genetic operators.
- Separation of objectives and constraints.
- Hybrid methods.
- Novel approaches.

How to Deal with Constraints?

- **Penalty functions.**
- **Maintaining only feasible solutions based on special representation and genetic operators.**
- Separation of objectives and constraints.
- Hybrid methods.
- Novel approaches.

How to Deal with Constraints?

- **Penalty functions.**
- Maintaining only feasible solutions based on special representation and genetic operators (recombination, mutation).
- Separation of objectives and constraints.
- Hybrid methods.
- Novel approaches.

Penalty Functions

- Most common approach.
- Create a fitness function that considers both the objective and the constraint.
- Problem formulation vs algorithm to solve the problem.
- Infeasible solutions have their fitness penalised.

$$\text{fitness}(x) = f(x) +- Q(x)$$

Actual objective → $f(x)$

Penalty → $Q(x)$

Penalty Functions

- Most common approach.
- Create a fitness function that considers both the objective and the constraint.
- Problem formulation vs algorithm to solve the problem.
- Infeasible solutions have their fitness penalised.

Actual objective

$$\text{fitness}(x) = \begin{cases} f(x) & \text{if } x \text{ is feasible} \\ +\!-Q(x) & \text{otherwise} \end{cases}$$

Penalty

Penalty Functions

- Death penalty
- Penalty based on the level of infeasibility

Image from: http://globe-views.com/dcim/dreams/death/death-02.jpg

Death Penalty

Death Penalty

Assigns worse possible fitness to infeasible solutions.

$$Q(x) = \begin{cases} 0 & \text{if solution is feasible} \\ c & \text{otherwise} \end{cases}$$

very large positive constant making infeasible solutions worse than any feasible ones.

Death Penalty

Not usually computed for infeasible solutions

very large positive constant for infeasible solutions

$$\text{fitness}(x) = f(x) +\text{-}\ Q(x)$$

$$\text{fitness}(x) = \begin{cases} f(x) & \text{if } x \text{ is feasible} \\ +\text{-}\ Q(x) & \text{otherwise} \end{cases}$$

For maximisation problems, should we add or subtract Q(x)?

Death Penalty

Not usually computed for infeasible solutions

very large positive constant for infeasible solutions

fitness(x) = f(x) +- Q(x)

$$
\text{fitness}(x) = \begin{cases} f(x) & \text{if } x \text{ is feasible} \\ +\text{-}Q(x) & \text{otherwise} \end{cases}
$$

For maximisation problems: use -Q(x)

For minimisation problems: use +Q(x)

Death Penalty

- Easy to apply.
- Efficient:
  - No need to calculate fitness of infeasible solutions.
  - No need to calculate level of infeasibility.
- Problem:
  - All infeasible solutions are considered equally bad.

Death penalty offers no guidance towards feasibility.

Using Level of Infeasibility

$$Q(x) = \begin{cases} 0 & \text{if } x \text{ is feasible} \\ c \cdot \phi(x)^2 & \text{otherwise} \end{cases}$$

Violated constraint (squared)

Very large positive constant

Example for Lorry Problem

$$Q(x) = \begin{cases} 0 & \text{if } x \text{ is feasible} \\ c \cdot \phi(x)^2 & \text{otherwise} \end{cases}$$

Violated constraint (squared)

Very large positive constant

Example with 1 constraint

$$\left(\sum_{i=1}^{N} v_i \, w_i\right) - W \leq 0 \qquad \phi(v) = \left(\sum_{i=1}^{N} v_i \, w_i\right) - W$$

Example for Lorry Problem

$$Q(x) = \begin{cases} 0 & \text{if } x \text{ is feasible} \\ c\ \phi(x)^2 & \text{otherwise} \end{cases}$$

Example with 1 constraint

The more overloaded the lorry is, the more penalty it gets.

$$\left(\sum_{i=1}^{N} v_i\, w_i\right) - W \leq 0 \qquad \phi(v) = \left(\sum_{i=1}^{N} v_i\, w_i\right) - W$$

Example for Lorry Problem

very large positive value for overloaded lorries

maximise fitness(x) = f(x) +- Q(x)

$$\text{maximise fitness}(x) = \begin{cases} f(x) & \text{if } x \text{ is feasible} \\ +- Q(x) & \text{otherwise} \end{cases}$$

Should we add or subtract Q(x)?

Subtract

Using Level of Infeasibility

If you have k violated constraints instead of 1:

$$Q(x) = \begin{cases} 0 & \text{if } x \text{ is feasible} \\ c \cdot [\phi_1(x)^2 + \phi_2(x)^2 + \ldots + \phi_k(x)^2] & \text{otherwise} \end{cases}$$

Using Level of Infeasibility

- Less computationally efficient than death penalty, but
- may be able to guide the search from infeasible to feasible solutions.

Penalty Functions

Easy or relatively easy to implement.

May not work well for problems where it is extremely difficult to find a single feasible solution.

How to Deal with Constraints?

- Penalty functions.
- **Maintaining only feasible solutions based on special representation and genetic operators.**
- Separation of objectives and constraints.
- Hybrid methods.
- Novel approaches.

Special Representation and Genetic Operators

- Restrictive representation and variation (mutation / recombination) operators
- Decoding operator
- Repair operator

Restrictive Representation and Variation Operators

- Representation and variation operators that ensure the constraints to be satisfied by simplifying the search space.
    - Representation is used to restrict the search space.
- Operators are used to preserve feasibility.

Restrictive Representation and Variation Operators

- Example:
  - If an integer variable must be between 0 and 15, set a binary representation to use 4 bits.
  - Use of permutations as the representation for traveling salesman problem, design operators that produce offspring that are also permutations.
- Care must be taken not to oversimplify the search space.
  - This could result in optimal solutions being eliminated from the search space.

Decoding Operators

- Decoding operation gives instructions on how to create a feasible phenotype from a possibly infeasible genotype.
- Does not change the genotype.
- Example:
  - Once the maximum weight of the lorry is reached, stop loading products.
- Depending on the design, it is more difficult to guide the search towards optimal solutions.
  - E.g., parts of the genotype may be bad, but the individual could still be evaluated as good.
  - E.g., parts of the genotype may be unused, and any mutation / recombination affecting those parts would be useless.
- Depending on the design, it may be computationally expensive.

Repair Operator

- Fix an infeasible genotype to make it feasible.
- Example:
  - If the maximum loaded weight is exceeded, delete less profitable products from the genotype until the maximum weight is satisfied.
- Depending on the problem, it may harm the evolutionary process:
  - It may destroy good building blocks of the parent solutions carried in the children.
  - Greediness might cause algorithms to get stuck in local optima.
  - E.g., one of the less profitable products may be a good product to have in the solution, depending on its weight.

Special Representation and Operators

Work well for problems where it is extremely difficult to find a single feasible solution.

May be very difficult to design.

Further Reading

Carlos Coello, A Survey of Constraint Handling Techniques used with Evolutionary Algorithms, Laboratorio Nacional de Informática Avanzada, 1999. Read sections 1-3, section 4 until the end of section 4.1, and section 4.8.

http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.43.9288

Reminder: lab session today at 3pm!