Optimisation Theory

8. MM

Today's Lecture

- Evolutionary Methods
  - Genetic Algorithms
- Multi-Objective Optimisation Problems (MOOPs)
  - Pareto optimality
  - Dominated, non-dominated and utopia points
- Some Multi-Objective Optimisation "Methods"
  - Weighting methods
  - Lexicographic method
- Exam?
- Exercises

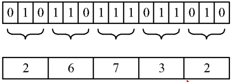
Evolutionary Methods

- Derives on the evolutionary theories by Darwin about "survival of the fittest"
  - A generation of designs are created
  - Among these designs parents are selected based on their fitness (the most fit are the most likely to be selected)
  - A child (or child generation) is generated based on the selected parents (crossover)
  - Mutations (and permutations) may happen during the child generation
  - The optimisation typically runs for a finite number of generations
- Four main categories (many similarities):
  - Genetic algorithms (GA)
  - Evolutionary Strategies (ES)
  - Evolutionary Programming (EP)
  - Genetic Programming (GP)

Evolutionary Methods - Representation

Typical representation
Genetic Algorithms (GA)

String like representation:

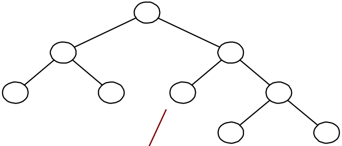


(bit representation)

(real value representation)

Typical representation
Evolutionary Strategies (ES)

Tree like representation:



Typical representation
Genetic Programming (GP)

Genetic Algorithms General Concepts

Population: A set of design points (designs) at a given iteration is called a population. The number of designs in a population is denoted by N_p.

Generation: A population for a given iteration of the genetic algorithm is called a generation (has population size N_p).

Parent: A given design in the current generation is called a parent if this is selected for reproduction.

Child: A design in the next generation (generated based on one or more parents).

See example

Genetic Algorithms General Concepts

Chromosone: Term used to represent a design point, i.e. a chromosone represents a set of design variables (encoded in a string).

Gene: Term used for a scalar value of the design vector, i.e. it represents the value of a particular design variable.

Design representation: A method to represent the design variables in a chromosone (string) - also called a *schema*. The most common approach is a binary encoding, but real value (Evolutionary Strategies) and integer representations are also used.

Fitness function: Defines the relative importance of the design (higher fitness value implies a better design). This is typically defined in terms of the objective function like e.g.:

$$F_i = (1 + \epsilon) f_{max} - f_i$$

Genetic Algorithms

Three fundamental steps:

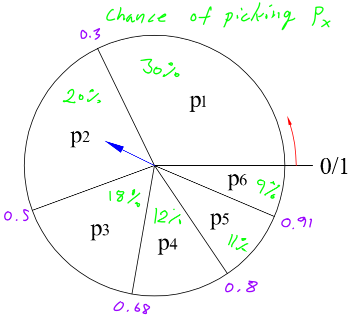
- Reproduction/Selection process
  - Dependent on their fitness parents are selected for reproduction
- Crossover
  - Generating a new child based on a set of parents
- Mutation
  - "Defects" in the crossover – one or more genes are altered randomly to safeguard the process of premature convergence
  - Permutation is a variation hereof, where two (or more) genes swap places.

Genetic Algorithms Reproduction/Selection Process

- Based on a given designs fitness value F_i, a probability of selection is calculated as:

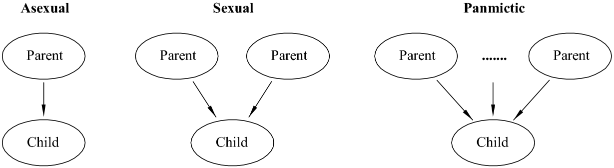
$$P_i = \frac{F_i}{Q} \qquad , \qquad Q = \sum_{j=1}^{N_p} F_i$$

- A random number is generated and a parent selected based on roulette wheel selection
- Typically the parents are sorted according to their ranking

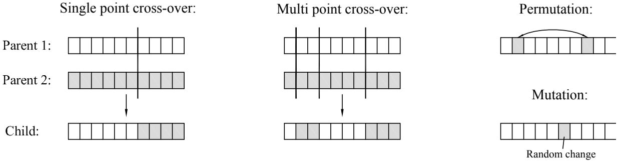


Genetic Algorithms Crossover and Mutations

- One or more random crossover points are generated and crossover performed based on selected parents
- Various genetic reformation operators:



- Crossover and mutation:



General Genetic Algorithm

Step 1: Define a schema to represent different design points. Randomly generate N_p genetic strings (members of the population) according to the schema, where N_p is the population size. Or use the seed to generate the initial population. For *constrained problems*, only the feasible strings are acepted when the penalty function approach is not used. Set iteration counter $K = 0$. Define a fitness function for the problem as e.g.

$$F_i = (1 + \epsilon)f_{max} - f_i$$

Step 2: Calculate the fitness values for all the designs in the population. Set $K = K + 1$, and the counter for the number of crossovers $I_c = 1$.

Step 3 (reproduction): Select designs from the current population according to the roulette wheel selection process for the mating pool (next generation) from which members for crossover and mutation are selected.

General Genetic Algorithm

Step 4 (crossover): Select two designs from the mating pool. Randomly choose two sites on the genetic strings and swap strings of 0's and 1's between the two chosen sites. Set $I_c = I_c + 1$.

Step 5 (mutation): Choose a fraction (P_m) of the members from the mating pool and switch a 0 to a 1 or vice versa at a randomly selected site on each chosen string. If, for the past I_g consecutive generations, the member with the lowest cost remains the same, the mutation fraction P_m is doubled. I_g: integer defined by user.

Step 6: If the member with the lowest cost remains the same for the past two consecutinve generations then increase I_{max} (integer that controls amount of crossover). If $I_c < I_{max}$, go to step 4. Otherwise continue.

Step 7 (stopping criterion): If after the mutation fraction P_m is double, the best value of the fitness is not updated for the past I_g consecutive generations, then stop. Alternative if K excees N_{gen} then stop (N_{gen} is the specified number of allowed generations). Otherwise, go to step 2.

Multi-Objective Optimisation Problems

- General problem:

$$\text{minimise } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x}))$$

subject to:

$$\mathbf{h}(\mathbf{x}) = 0$$
$$\mathbf{g}(\mathbf{x}) \leq 0$$

- Notice that the problem has several objective functions (a vector of object functions)
- "Our" problem: the methods we have considered have only covered single objective function. What to do?

Example

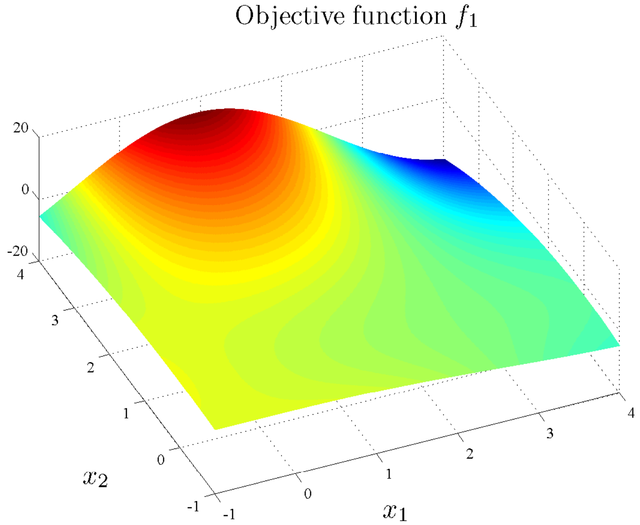
minimise

$$f_1(x_1, x_2) = x_2^2 \cos(x_1 - 1) - x_1$$
$$f_2(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2$$

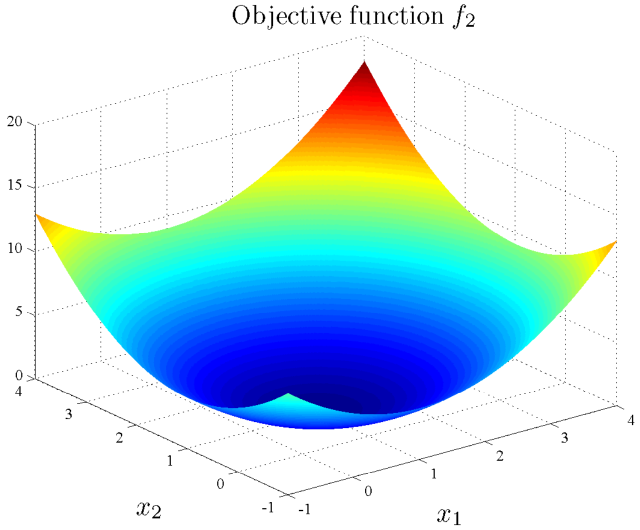
subject to:

$$g_1(x_1, x_2) = -x_1 - 1 \leq 0$$
$$g_2(x_1, x_2) = x_1 - 4 \leq 0$$
$$g_3(x_1, x_2) = -x_2 - 1 \leq 0$$
$$g_4(x_1, x_2) = -x_2 - -4 \leq 0$$

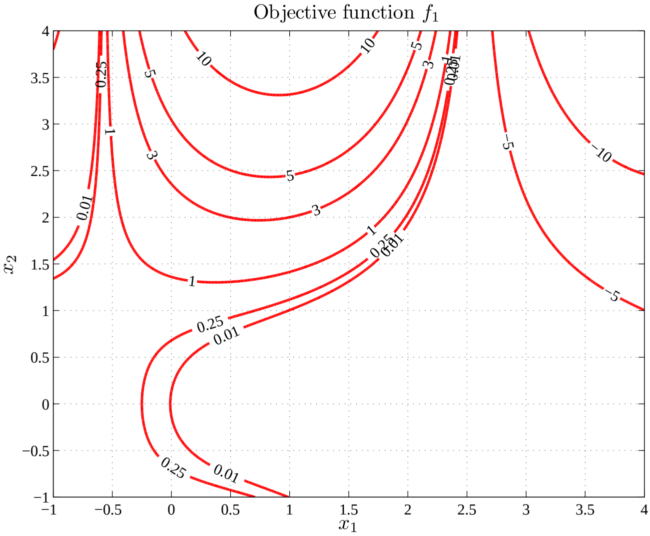
Surface plot of f_1



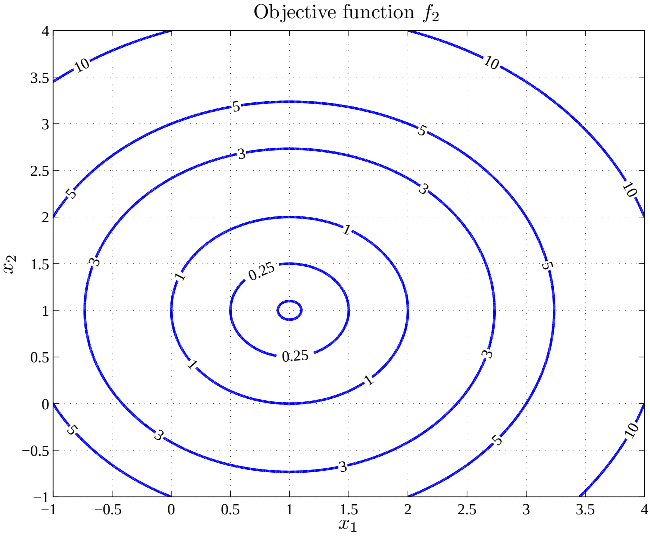
Surface plot of f_2



Contour plot of f_1



Contour plot of f_2



Multi-Objective Optimisation - Basic concepts

- The feasible *design space*:

$$S = \{\mathbf{x} \,|\mathbf{h}(\mathbf{x}) = \mathbf{0} \text{ and } \mathbf{g}(\mathbf{x}) \leq \mathbf{0}\}$$

- The feasible *criterion space*:

$$Z = \{\mathbf{f}(\mathbf{x}) \,|\mathbf{x} \text{ in the feasible set } (S)\}$$

- Attainability: implies that a point in the criterion space can be related to feasible point in the design space.
- Note:
  - One feasible point in the design space corresponds to one feasible point in the criterion space
  - One feasible point in the criterion space (one objective value) may correspond to many feasible points in the design space (also infeasible points!)

Basic Concepts - Pareto Optimality

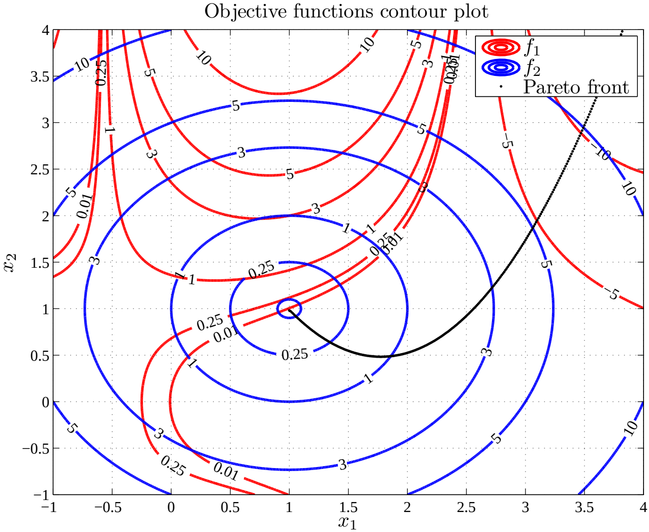
Pareto optimality:

A point $\mathbf{x}^*$ in the feasible design space S is Pareto optimal *if and only if* there does *not* exist another point, $\mathbf{x}$, in the set S, for which $\mathbf{f}(\mathbf{x}) \leq \mathbf{f}(\mathbf{x}^*)$ with at least one $f_i(\mathbf{x}) < f_i(\mathbf{x}^*)$.

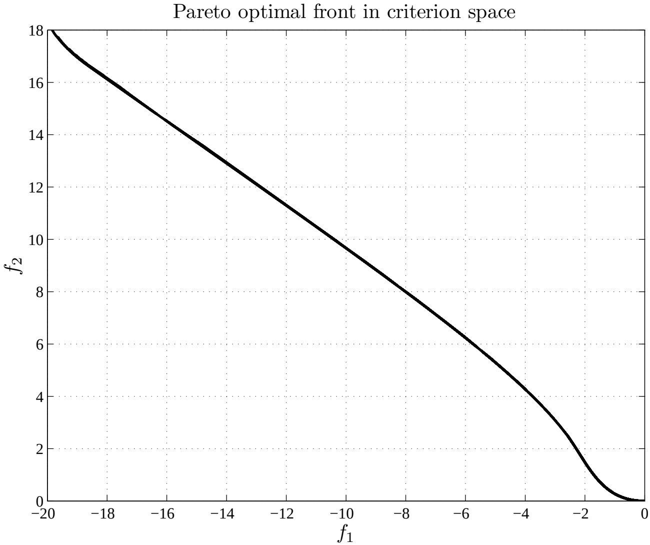
Weak pareto optimality:

A point $\mathbf{x}^*$ in the feasible design space S is weakly Pareto optimal *if and only if* there does *not* exist another point, $\mathbf{x}$, in the set S, for which $\mathbf{f}(\mathbf{x}) < \mathbf{f}(\mathbf{x}^*)$. That is, there is no other point that improves all the objective functions simultaneously; however, there may be points that improve some of the objectives while keeping the others unchanged.

Pareto Optimal Points

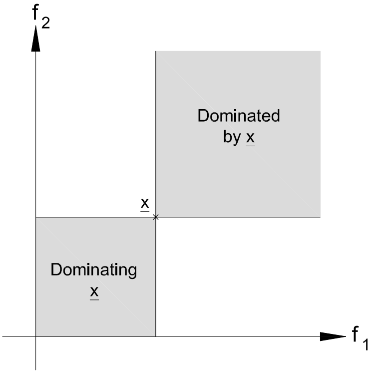


Pareto Front (Criterion Space)



Basic Concepts – Dominance

Domination:
A vector of objective functions $\mathbf{f}^* = \mathbf{f}(\mathbf{x}^*)$ in the feasible criterion space, Z, is nondominated *if and only if* there does *not* exist another vector $\mathbf{f}$ in the set Z such that $\mathbf{f} \leq \mathbf{f}^*$, with at least one $f_i < f_i^*$. Otherwise $\mathbf{f}^*$ is dominated.



Basic Concepts - Dominance

Dominance

A solution, $\mathbf{f}_1$, with a given set of design variables, $\mathbf{x}_1$ (i.e. $\mathbf{f}_1 = \mathbf{f}(\mathbf{x}_1)$), is said to dominate another solution, $\mathbf{f}_2$, with the design variables given by $\mathbf{x}_2$ if the following two conditions hold:

1. The solution $\mathbf{f}_1$ is partially less than $\mathbf{f}_2$, i.e. $f_{1,i} \leq f_{2,i}\ ,\ \forall\ i \in \{1, \ldots, k\}$.
2. The solution $\mathbf{f}_1$ is strictly better than $\mathbf{f}_2$ for at least one objective value, i.e. $f_{1,i} < f_{2,i}\ ,\ \exists\ i \in \{1, \ldots, k\}$.

The solution having the design variable set $\mathbf{x}_1$ is said to strongly dominate $\mathbf{x}_2$ if the solution corresponding to $\mathbf{x}_1$ is strictly better than the solution having the set of design variables $\mathbf{x}_2$ for all the k objective values.

Basic Concepts - Pareto Optimality

Pareto Optimality (Alternative def.)

A set of design variables, $x^* \in S$ is termed "Pareto optimal" if $f_i(\mathbf{x}^*)$ dominates any other feasible set of design variables. The corresponding objective vector is in this case also said to be "Pareto optimal" or non-dominated.

Basic Concepts - Utopia Point

Utopia Point:

A point $\mathbf{f}^\circ$ in the criterion space is called the utopia point if $f_i^\circ = \min\{f_i(\mathbf{x}) \,|\forall \mathbf{x} \in S\}$ for all $i = 1$ to k. This is also called the ideal point.

Compromise solution:

The point that is as close to the utopia point as possible. This is typically defined in terms of the Euclidian distance $D(\mathbf{x})$:

$$D(\mathbf{x}) = \|\mathbf{f}(\mathbf{x}) - \mathbf{f}^\circ\| = \left(\sum_{i=1}^{k} (f_i(\mathbf{x}) - f_i^\circ)^2\right)^{\frac{1}{2}}$$

Compromise solutions are Pareto optimal.

MOOP – Weighting Methods

Weighted Global Criterion Method (most common):

$$U = \left(\sum_{i=1}^{k} \left[w_i \left(f_i(\mathbf{x}) - f_i^{\circ} \right) \right]^p \right)^{\frac{1}{p}}$$

p is used to control the emphasis placed on minimising the function with the largest $f_i(\mathbf{x}) - f_i^{\circ}$.

- $p = 1$: Weighted sum method: $U = \sum_{i=1}^{k} w_i f_i(\mathbf{x})$
- $p = 2$ and $w = 1$: Compromise solution (U is a measure for the Euclidian distance from the utopia point
- $p = \infty$: Weighted min-max method (weighted Tchebycheff): $U = \max \left\{ w_i \left[f_i(\mathbf{x}) - f_i^{\circ} \right] \right\}$

MOOP – Lexicographic Method

Lexicographic method:

Design objectives (objective functions) are not assigned weights, but ordered relative to importance (determined by designer). Then the optimisation problem is solved once at a time for:

$$\begin{aligned} &\text{minimise} \quad f_i(\mathbf{x}) \\ \text{subject to:} & \\ &f_j(\mathbf{x}) \leq f_j(\mathbf{x}_j^*) \quad , \text{ for } j = 1 \text{ to } i-1 \\ &\qquad\qquad\qquad\quad , \text{ and } i = 1 \text{ to } k \end{aligned}$$

i: Functions position in the ordered sequence

$f_j(\mathbf{x}_j^*)$: the mininum for the j'th objective function in the j'th optimisation problem.

Selecting Methods

Method	Always yields Pareto optimal points?	All Pareto optimal points?	Involves weights?	Req. function continuity?	Use utopia point (or approx.)?
Genetic algorithms	Yes	Yes	No	No	No
Weighted sum	Yes	No	Yes	Determined by opt. algorithm (engine)	Used for function normalisation or method formulation
Weighted min-max	Yes[1]	Yes	Yes	—||—	—||—
Weighted global criterion	Yes	No	Yes	—||—	—||—
Lexicographic	Yes[2]	No	No	—||—	No
Bounded obj. func.	Yes[3]	No	No	—||—	No
Goal programming	No	No	No[4]	—||—	No

[1] Sometimes only weakly Pareto optimal

[2] If global optimisation engine is used or if solution point is unique

[3] Weakly Pareto optimal solution, unless solution point is unique

[4] Weights may be incorporated in objective function

Today's Exercises

- Exercise (Arora 2017 / 2012 / 2004): 18.9 / 17.9 /17.9
  - Use Matlab to solve the problem, use e.g. fmincon to solve the problem (notice constraints are linear)
- Exercise 18.1 / 17.1
  - Contours may be plotted in Matlab (using contour command)
  - Do not plot gradients nor Pareto fronts in Matlab!
- Exercise 18.2 / 17.2
- Exercise 18.4 / 17.4