AI791 (Computational Intelligence for Optimization Topic 1: Optimization Theory

AP Engelbrecht

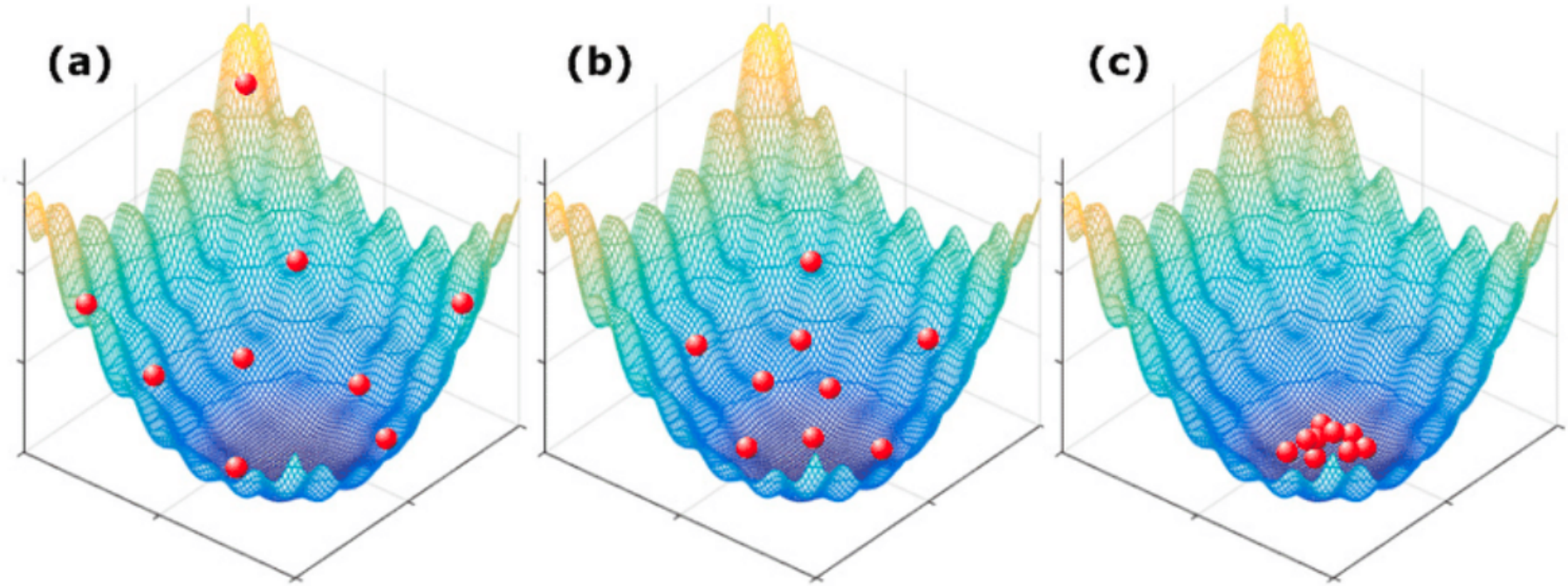
Department of Industrial Enigneering, and
Division of Computer Science
Stellenbosch University
South Africa
engel@sun.ac.za
https://engel.pages.cs.sun.ac.za/

Presentation Outline I

Reading Material

The content of this topic is from Appendix A of the prescribed book, *Computational Intelligence: An Introduction*, AP Engelbrecht, Second Edition, Wiley, 2007

The goal: Find the best values for decision variables such that some objective function is minimized or maximized



To be discussed:

- Optimization Problems and Algorithm Classes
- Formal Definitions of Optimization Problem Classes

Some real-world optimization problems:

- Many scheduling problems
- Data clustering
- DNA sequence alignment
- Training of a neural network
- Optimization of digital radiography
- Optimization of manure distribution to crop farms
- Aircraft wing optimization
- Portfolio optimization
- Wind farm layout optimization

And many others....

Each optimization problem consists of the following basic ingredients:

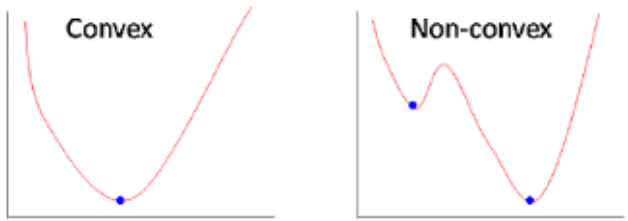
- An **objective function**, f, which represents the quantity to be optimized
- A **vector of unknowns or variables**, $\mathbf{x}$, with each decision variable affecting the value of the objective function
- A **set of constraints**, which restricts the values that can be assigned to the decision variables

The task is to find a minimum or maximum of the function,

$$f : \mathbb{R}^{n_x} \rightarrow \mathbb{R}$$

Optimization problems are characterized based on a number of characteristics:

- The **number of decision variables** that influence the objective function
- The **type of decision variables**
- The **degree of nonlinearity of the objective function**
- The **constraints used**
- The **number of optima**
- The **number of objective functions**
- Whether the **objective function changes** over time.
- Dependence among variables
- Is the function continuous or discrete?
- Is the function convex or not?



Based on the problem characteristics, the following classes of optimization problems are defined:

- **unconstrained**, for problems where there are no constraints on the values assigned to decision variables
- **boundary (box) constrained**, for problems with boundary constraints on the decision variables
- **constrained**, for problems with functional constraints on the values of the decsion variables
- **multi-/many-objective** for problems with more than one objective to optimize
- **multi-modal**, where multiple solutions have to be found
- **dynamic**, where the search landscape changes over time
- combinations of the above

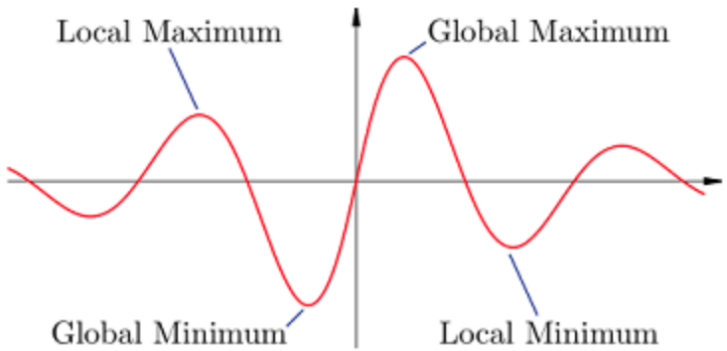
Classes of optimization algorithms:

- Based on the amount of information used to guide the search:
  - Local search algorithms
  - Global search algorithms
- Based on the way in which candidate solutions are updated:
  - Deterministic
  - Stochastic
- Based on the number of search points:
  - Only one search point
  - “Population”-based

The solution $\mathbf{x}^* \in \mathcal{F}$, is a global minimum of the objective function, f, if

$$f(\mathbf{x}^*) \leq f(\mathbf{x}),\ \forall \mathbf{x} \in \mathcal{F}$$

where $\mathcal{F} \subseteq \mathcal{S}$.



$\mathcal{F}$ is the feasible space
$\mathcal{S}$ is the search space

Strong local minimum: The solution, $\mathbf{x}^*_{\mathcal{N}} \in \mathcal{N} \subseteq \mathcal{F}$, is a strong local minimum of f, if

$$f(\mathbf{x}^*_{\mathcal{N}}) < f(\mathbf{x}),\ \forall \mathbf{x} \in \mathcal{N}$$

where $\mathcal{N} \subseteq \mathcal{F}$ is a set of feasible points in the neighborhood of $\mathbf{x}^*_{\mathcal{N}}$.

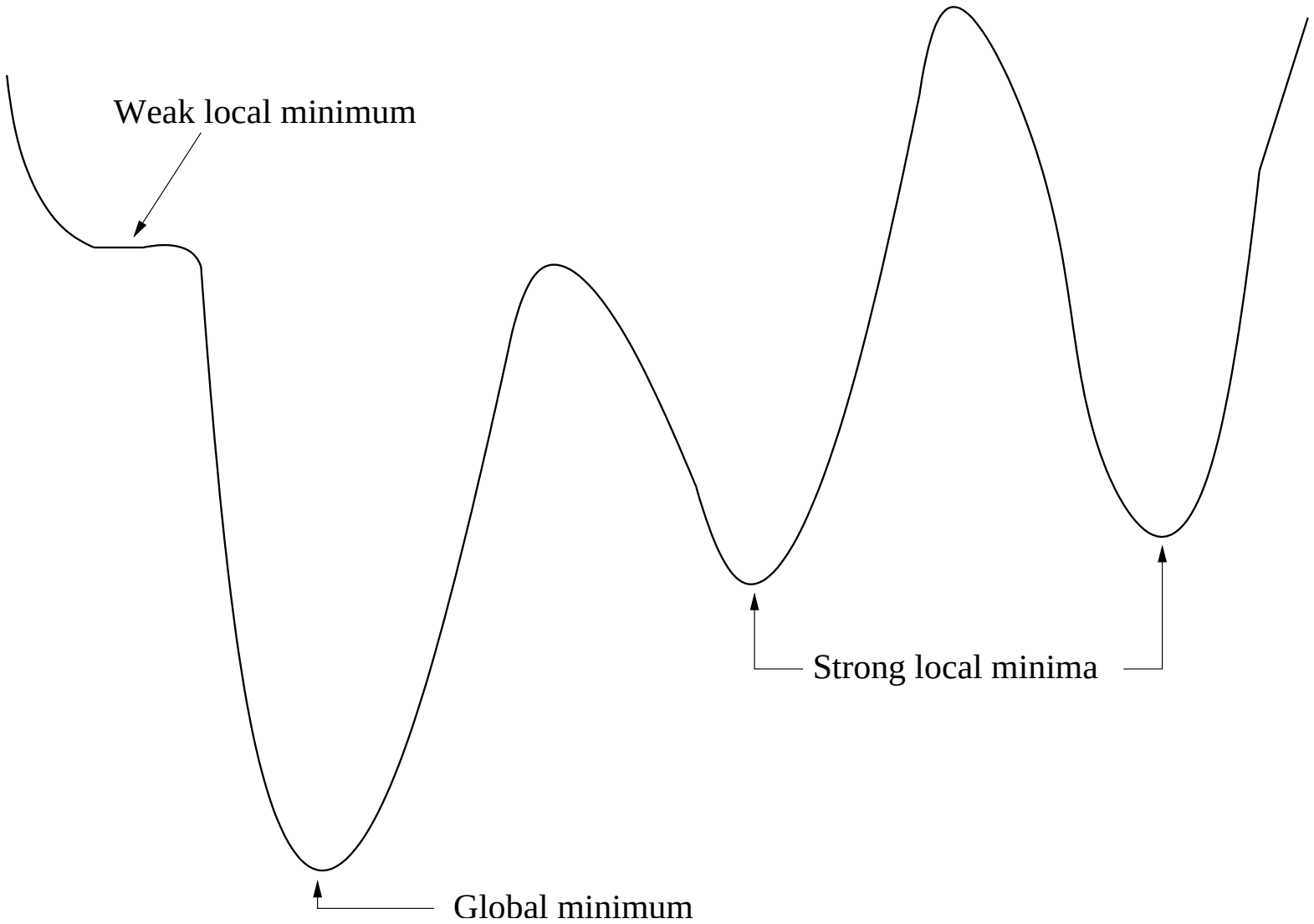
Weak local minimum: The solution, $\mathbf{x}^*_{\mathcal{N}} \in \mathcal{N} \subseteq \mathcal{F}$, is a weak local minimum of f, if

$$f(\mathbf{x}^*_{\mathcal{N}}) \leq f(\mathbf{x}),\ \forall \mathbf{x} \in \mathcal{N}$$

where $\mathcal{N} \subseteq \mathcal{F}$ is a set of feasible points in the neighborhood of $\mathbf{x}^*_{\mathcal{N}}$.

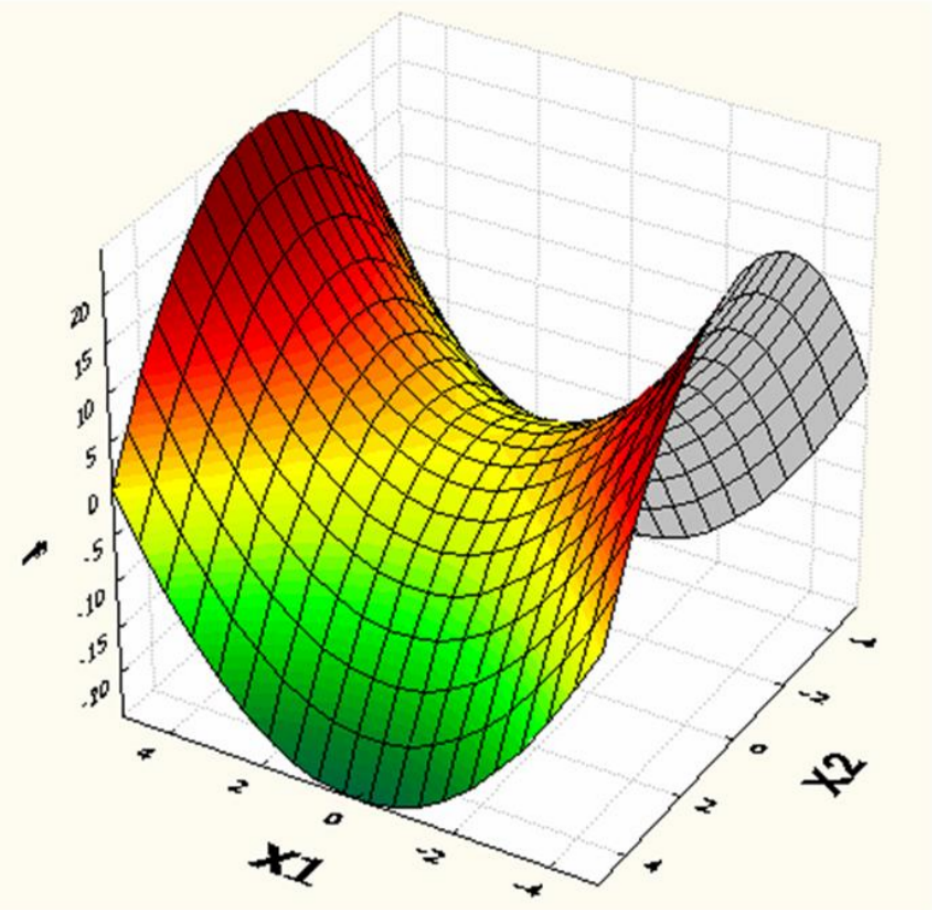
Formal Definitions

Minima Illustrated



Optimization Theory

Saddle Points



- saddle point or minimax point is a point on the surface of the graph of a function where the slopes (derivatives) in orthogonal directions are all zero
- in two dimensions, the surface curves up in one direction, and curves down in a different direction
- a value of a function which is a maximum wrt one variable and a minimum wrt the other

Boundary (box) constrained optimization problem:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}), \quad \mathbf{x} = (x_1, x_2, \ldots, x_{n_x}) \\ \text{subject to} \quad & x_j \in \text{dom}(x_j) \end{aligned}$$

where $\mathbf{x} \in \mathcal{F} = \mathcal{S}$, and $\text{dom}(x_j)$ is the domain of variable x_j.
For

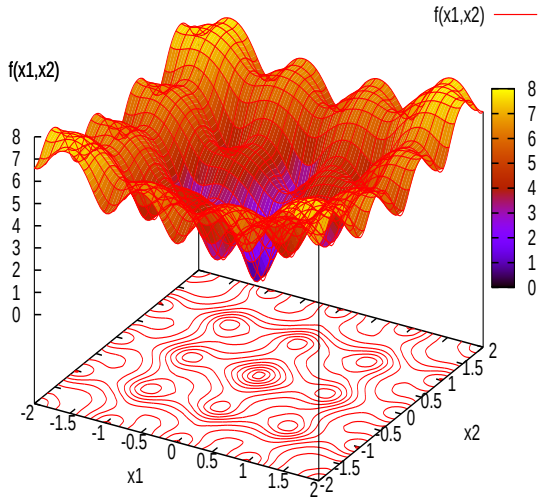
- a continuous-valued problem, the domain of each variable is $\mathbb{R}$, i.e. $x_j \in \mathbb{R}$
- an integer-valued problem, $x_j \in \mathbb{Z}$
- $\text{dom}(x_i)$ for a general discrete-valued problem is a finite set of values
- a binary-valued problem, $\mathbf{x} \in \mathbb{B}^{n_x}$, i.e. $x_j \in \{0, 1\}$

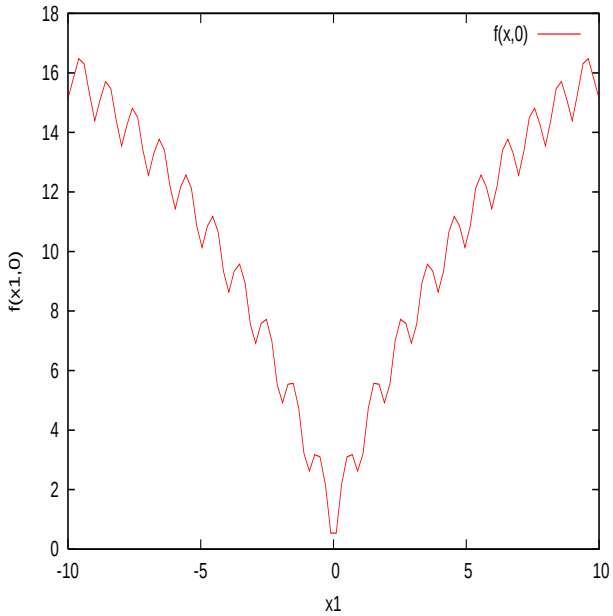
For unconstrained problems, $\text{dom}(x_j) = (-\infty, \infty), \forall j = 1, \ldots, n_x$

The Ackley Problem:

$$f(\mathbf{x}) = -20e^{-0.2\sqrt{\frac{1}{n_x}\sum_{j=1}^{n_x} x_j^2}} - e^{\frac{1}{n_x}\sum_{j=1}^{n_x}\cos(2\pi x_j)} + 20 + e$$

with $x_j \in [-30, 30]$ and $f^*(\mathbf{x}) = 0.0$

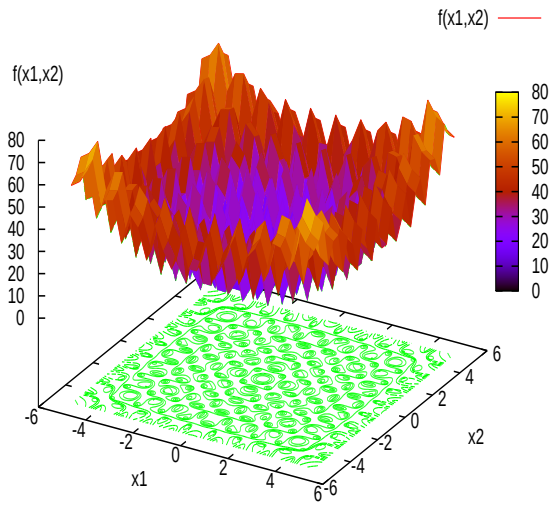


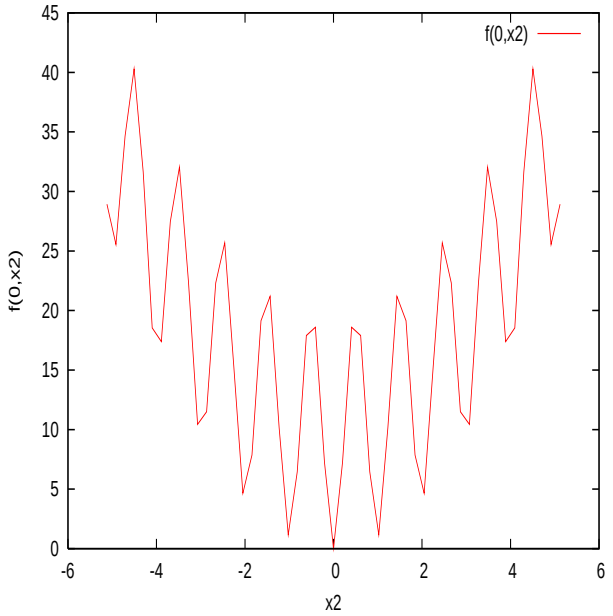


The Rastrigin Problem:

$$f(\mathbf{x}) = \sum_{j=1}^{n_x} (x_j^2 - 10\cos(2\pi x_j) + 10)$$

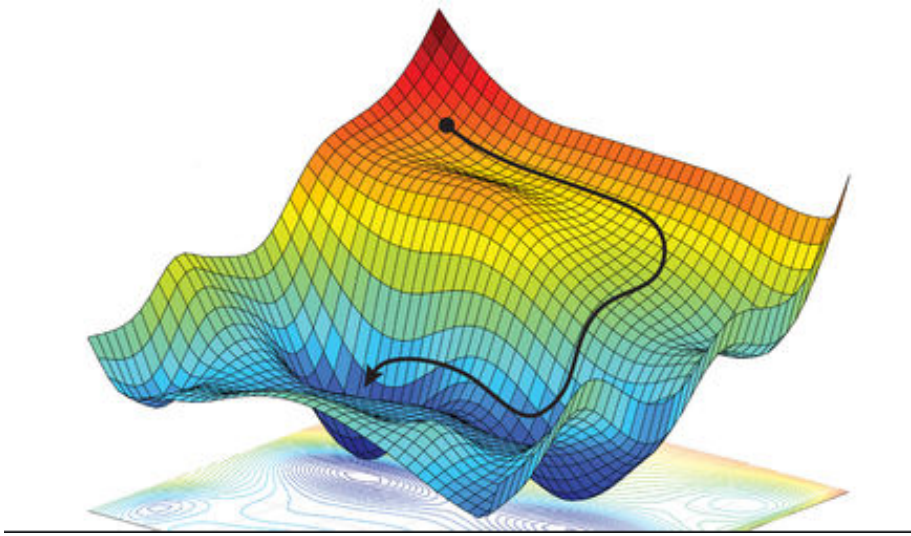
with $x_j \in [-5.12, 5.12]$ and $f^*(\mathbf{x}) = 0.0$



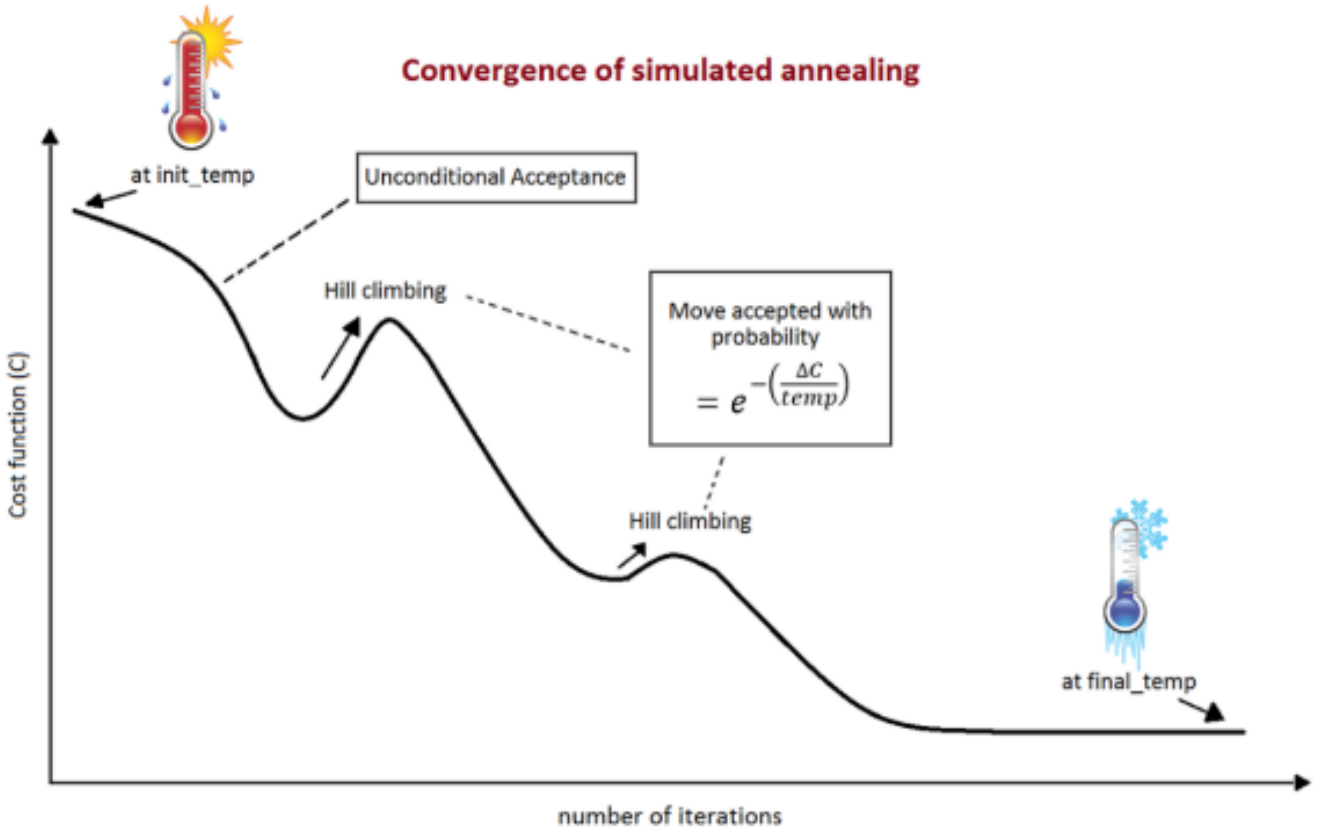


General Local Search Algorithm

Find starting point $\mathbf{x}(0) \in \mathcal{S}$;
$t = 0$;
repeat
  Evaluate $f(\mathbf{x}(t))$;
  Calculate a search direction, $\mathbf{q}(t)$;
  Calculate step length $\eta(t)$;
  Set $\mathbf{x}(t+1)$ to $\mathbf{x}(t) + \eta(t)\mathbf{q}(t)$;
  $t = t + 1$;
until *stopping condition is true*;
Return $\mathbf{x}(t)$ as the solution;



Simulated Annealing



Simulated Annealing Algorithm

Create initial solution, $\mathbf{x}(0)$, and set initial temperature, $T(0)$, $t = 0$;
repeat
  Generate new solution, $\mathbf{x}$, and determine quality, $f(\mathbf{x})$;
  Calculate acceptance probability using

$$P = \begin{cases} 1 & \text{if } f(\mathbf{x}) < f(\mathbf{x}(t)) \\ e^{-\frac{f(\mathbf{x}) - f(\mathbf{x}(t-1))}{c_b T}} & \text{otherwise} \end{cases}$$

  if $U(0,1) \leq$ *acceptance probability* **then**
    $\mathbf{x}(t+1) = \mathbf{x}$;
  end
  Adapt the temperature;
  t=t+1;
until *stopping condition is true*;
Return $\mathbf{x}(t)$ as the solution;

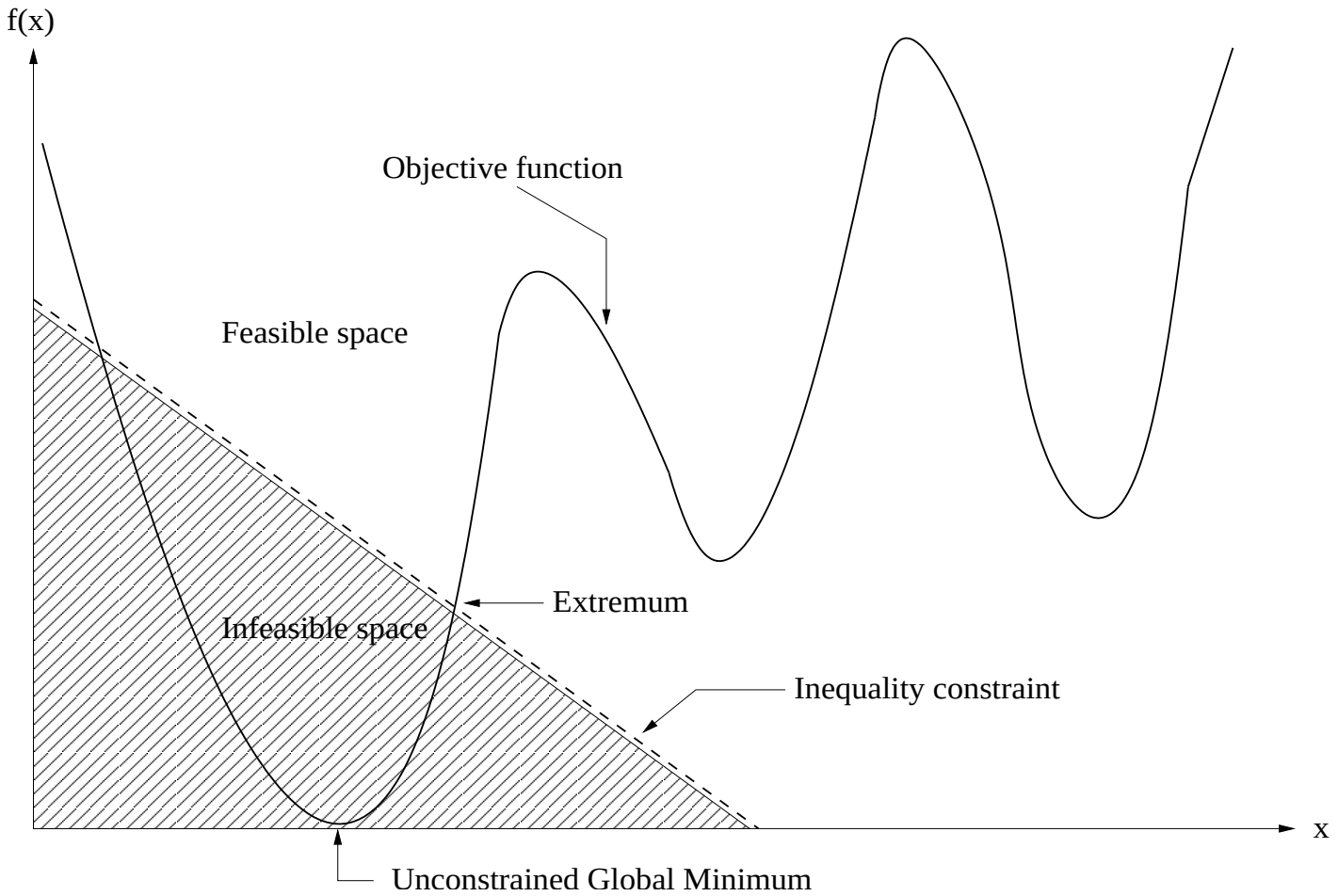
Constrained optimization problem:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}), \quad \mathbf{x} = (x_1, \ldots, x_{n_x}) \\ \text{subject to} \quad & g_m(\mathbf{x}) \leq 0, \ m = 1, \ldots, n_g \\ & h_m(\mathbf{x}) = 0, \ m = n_g + 1, \ldots, n_g + n_h \\ & x_j \in \text{dom}(x_j) \end{aligned}$$

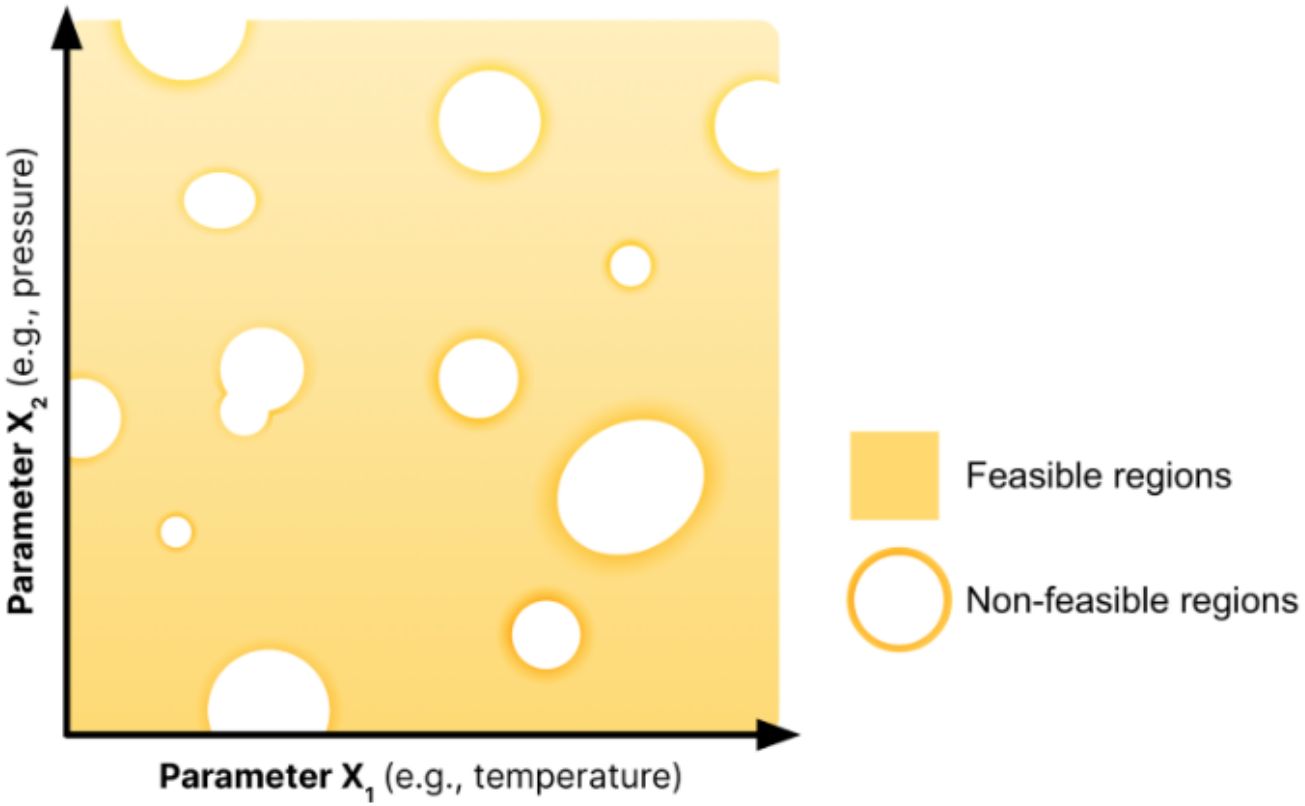
where n_g and n_h are the number of inequality and equality constraints respectively

Optimization Problem Classes

Constrained Optimization Problems (cont)



Effect of constraints: Feasible pockets are formed



The following issues need to be considered:

- How should two feasible solutions be compared?
- How should two infeasible solutions be compared?
  - should the infeasible solution with the best objective function value be preferred?
  - should the solution with the least number of constraint violations or lowest degree of violation be preferred?
  - should a balance be found between best objective function value and degree of violation, i.e. a multi-objective problem?
- How should a feasible solution and an infeasible solution be compared?
  - Should it be assumed that any feasible solution is better than any unfeasible solution?

Deb has proposed the following rules when two solutions are compared to decide on the best:

1. Any feasible solution is preferred to an unfeasible solution
2. If two solutions are feasible, the better solution based on fitness (objective function value) is preferred
3. If two solutions are infeasible, the solution with less violation is preferred

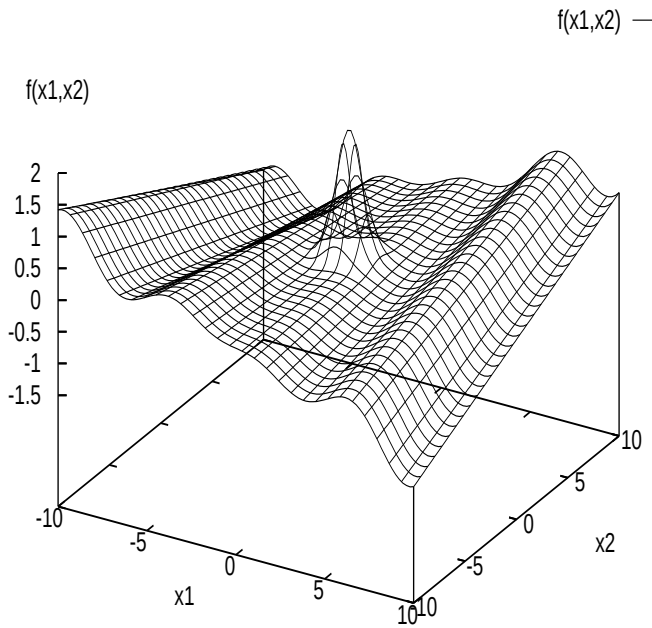
Minimization using the Penalty method:

$$\text{minimize} \quad F(\mathbf{x}, t) = f(\mathbf{x}, t) + \lambda p(\mathbf{x}, t)$$

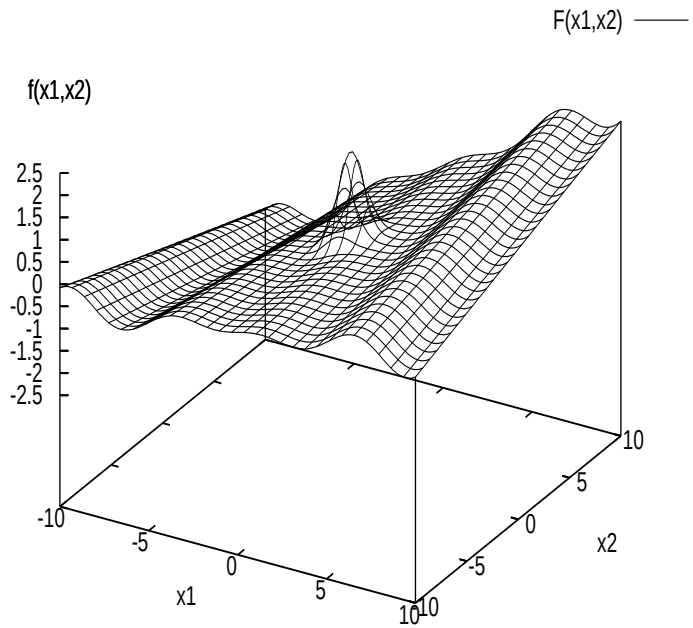
where λ is the penalty coefficient and $p(\mathbf{x}, t)$ is the (possibly) time-dependent penalty function

What are the problems with penalty methods?

$$f(x_1, x_2) = \frac{x_1 \cos(x_1)}{20} + 2e^{-x_1^2-(x_2-1)^2} + 0.01 x_1 x_2$$



(e) Original function



(f) With penalty $p(x_1, x_2) = 3x_1$ and $\lambda = 0.05$

$$p(\mathbf{x}_i, t) = \sum_{m=1}^{n_g+n_h} \lambda_m(t) p_m(\mathbf{x}_i)$$

where

$$p_m(\mathbf{x}_i) = \begin{cases} \max\{0, g_m(\mathbf{x}_i)^\alpha\} & \text{if } m \in [1, \ldots, n_g] \text{ (inequality)} \\ |h_m(\mathbf{x}_i)|^\alpha & \text{if } m \in [n_g + 1, \ldots, n_g + n_h] \text{ (equality)} \end{cases}$$

with α a positive constant, representing the power of the penalty

$\lambda_m(t)$ represents a time-varying degree to which violation of the m-th constraint contributes to the overall penalty

Convert the constrained (primal) problem to an unconstrained problem by defining the Lagrangian for the constrained problem:

$$L(\mathbf{x}, \lambda_g, \lambda_h) = f(\mathbf{x}) + \sum_{m=1}^{n_g} \lambda_{gm} g_m(\mathbf{x}) + \sum_{m=n_g+1}^{n_g+n_h} \lambda_{hm} h_m(\mathbf{x})$$

Then maximize the Lagrangian (dual problem):

$$\begin{array}{ll} \text{maximize}_{\lambda_g, \lambda_h} & L(\mathbf{x}, \lambda_g, \lambda_h) \\ \text{subject to} & \lambda_{gm} \geq 0, \quad m = 1, \ldots, n_g + n_h \end{array}$$

The vector $\mathbf{x}^*$ that solves the primal problem, as well as the Lagrange multiplier vectors, λ_g^* and λ_h^*, can be found by solving the min-max problem,

$$\min_{\mathbf{x}} \max_{\lambda_g, \lambda_h} L(\mathbf{x}, \lambda_g, \lambda_h)$$

Constrained problem 1: Minimize the function

$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

subject to the nonlinear constraints,

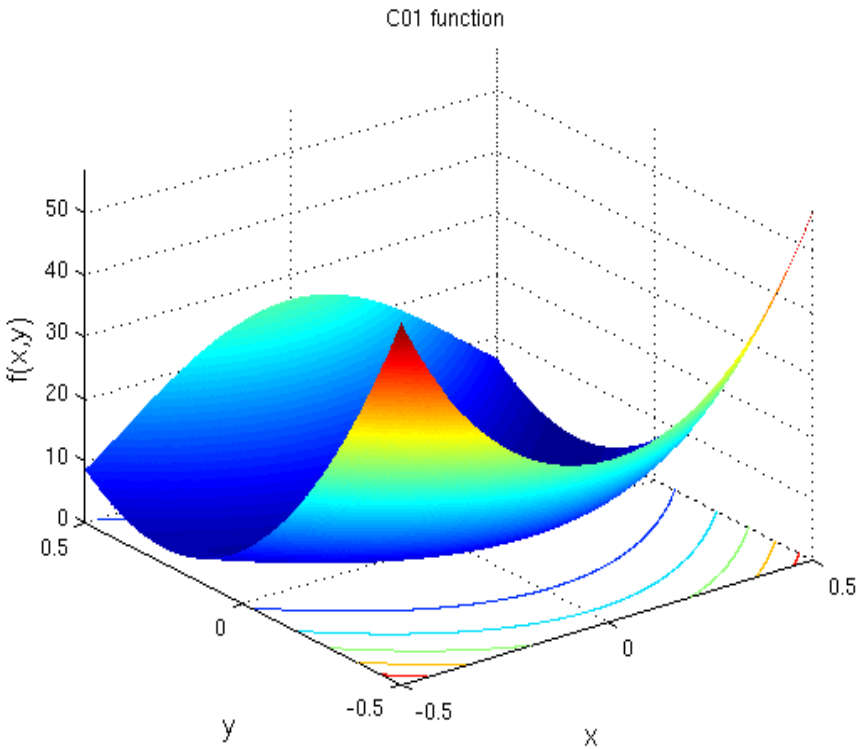
$$x_1 + x_2^2 \geq 0$$
$$x_1^2 + x_2 \geq 0$$

with $x_1 \in [-0.5, 0.5]$ and $x_2 \leq 1.0$.

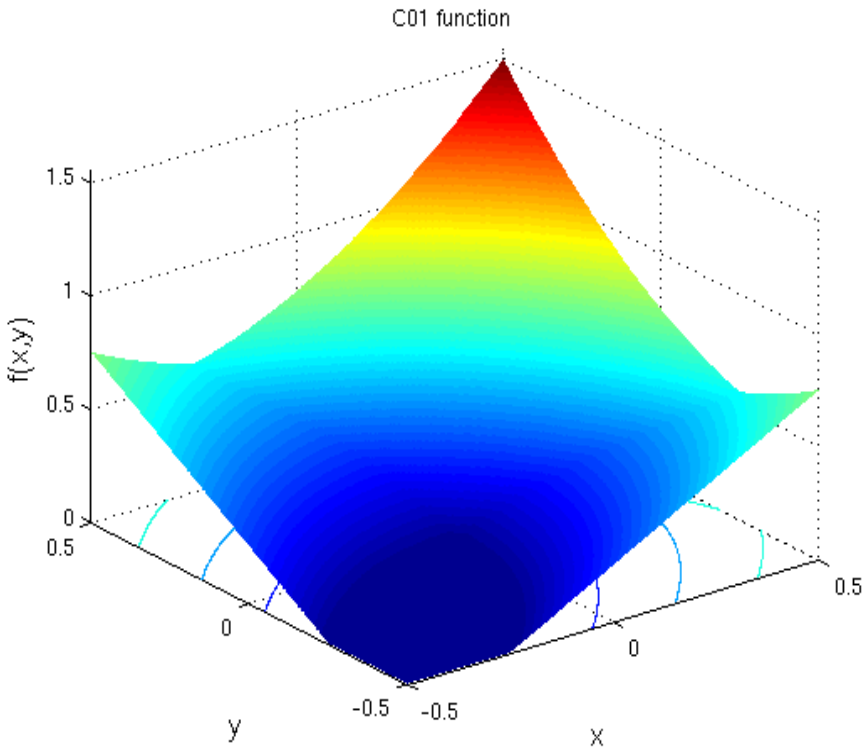
The global optimum is $\mathbf{x}^* = (0.5, 0.25)$, with $f(\mathbf{x}^*) = 0.25$

Optimization Problem Classes

Constrained Optimization Problem: Problem Examples (cont)



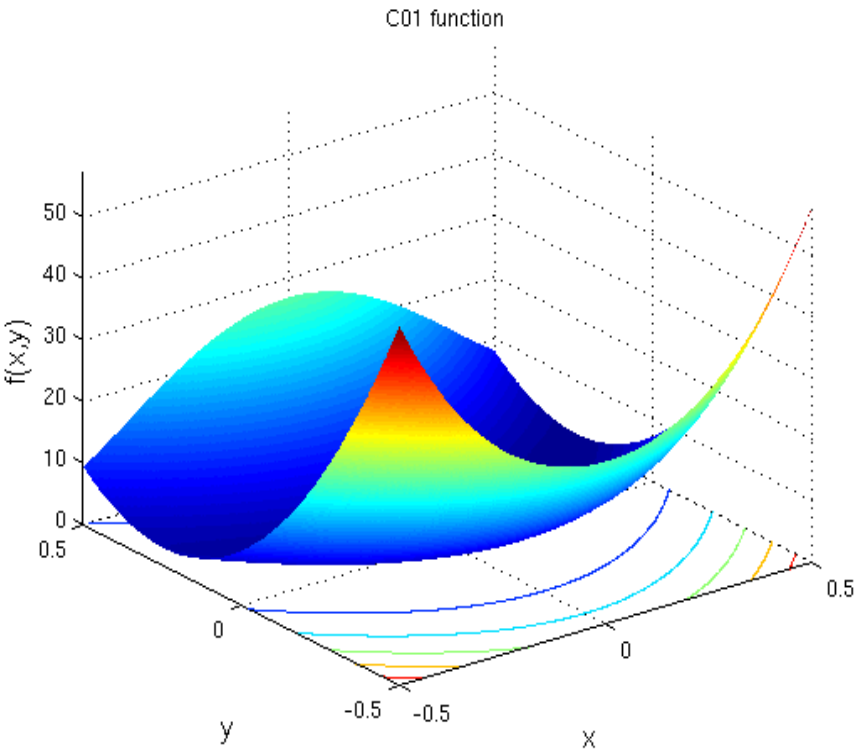
Function Landscape



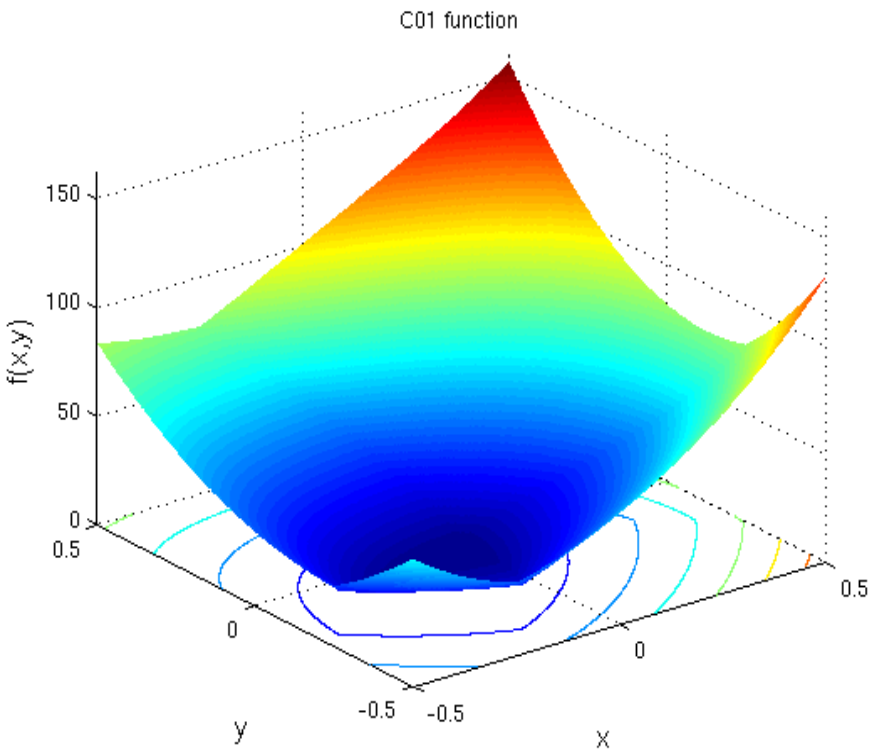
Violation Space

Optimization Problem Classes

Constrained Optimization Problems: Problem Examples (cont)



With $\lambda = 1$



With $\lambda = 100$

Constrained problem 2: Maximize the function

$$f(\mathbf{x}) = (\sqrt{n_x})^{n_x} \prod_{j=1}^{n_x} x_j$$

subject to the equality constraint,

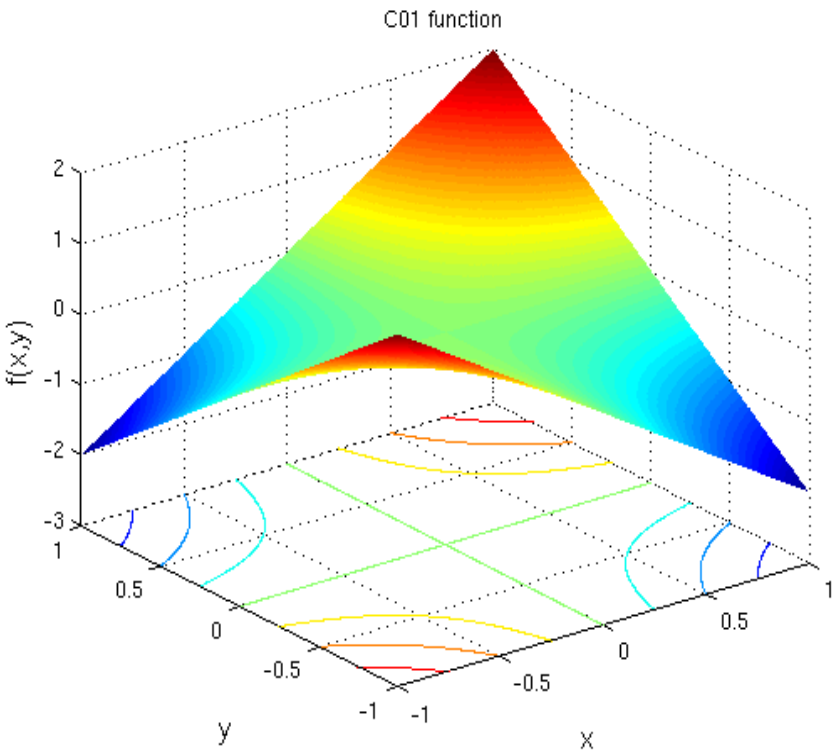
$$\sum_{j=1}^{n_x} x_j^2 = 1$$

with $x_j \in [0, 1]$.

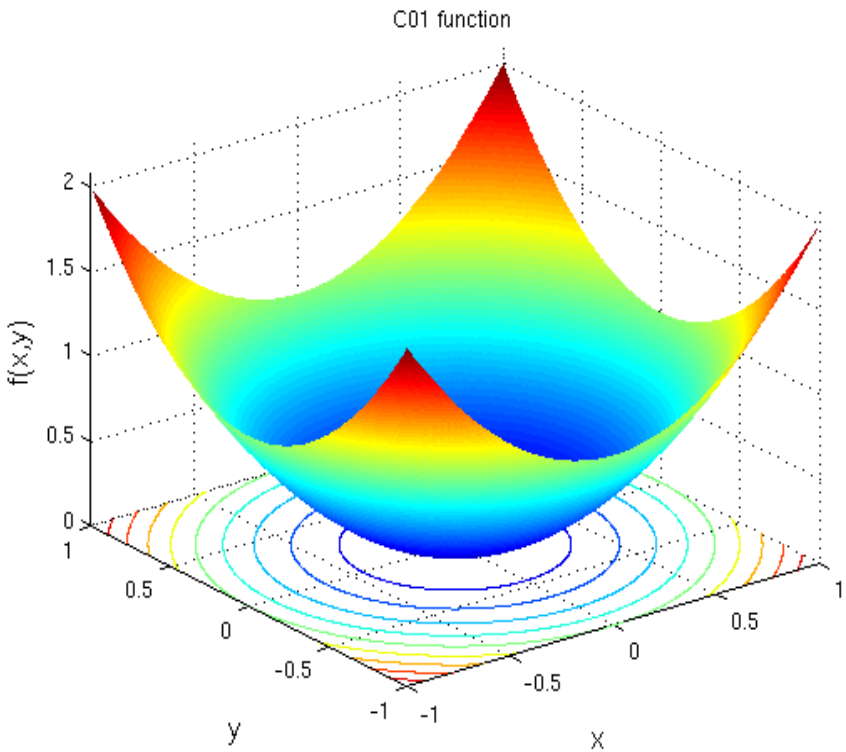
The solution is $\mathbf{x}^* = (\frac{1}{\sqrt{n_x}}, \ldots, \frac{1}{\sqrt{n_x}})$, with $f(\mathbf{x}^*) = 1$.

Optimization Problem Classes

Constrained Optimization Problems: Problem Examples (cont)



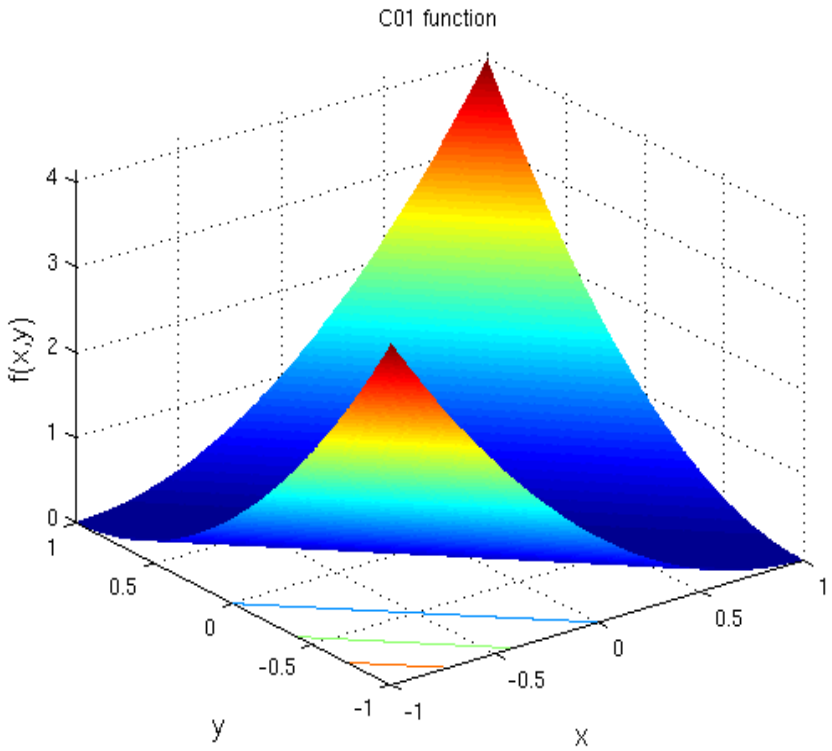
Function Landscape



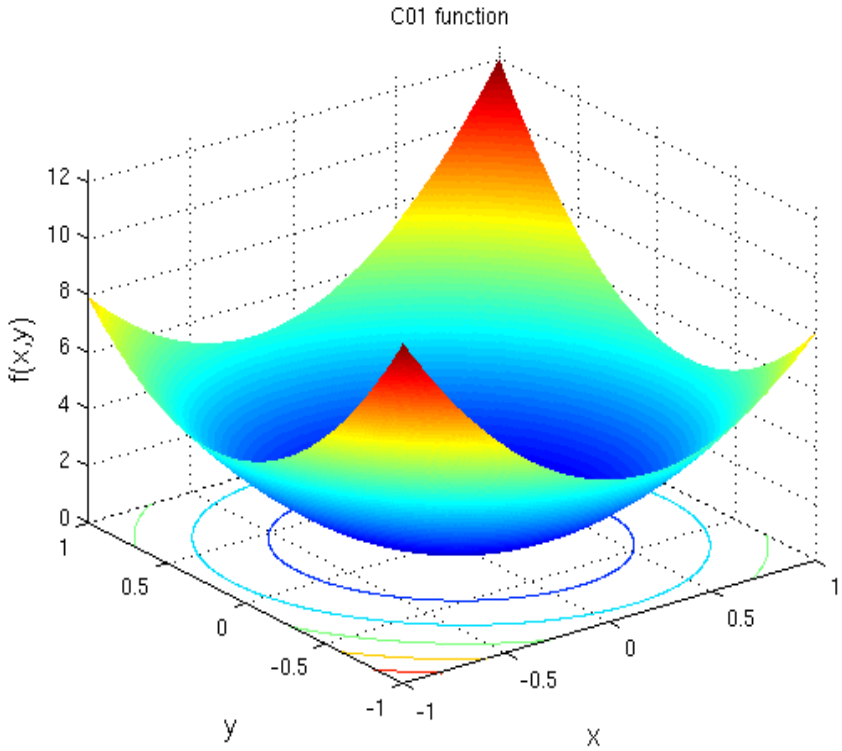
Violation Space

Optimization Problem Classes

Constrained Optimization Problems: Problem Examples (cont)

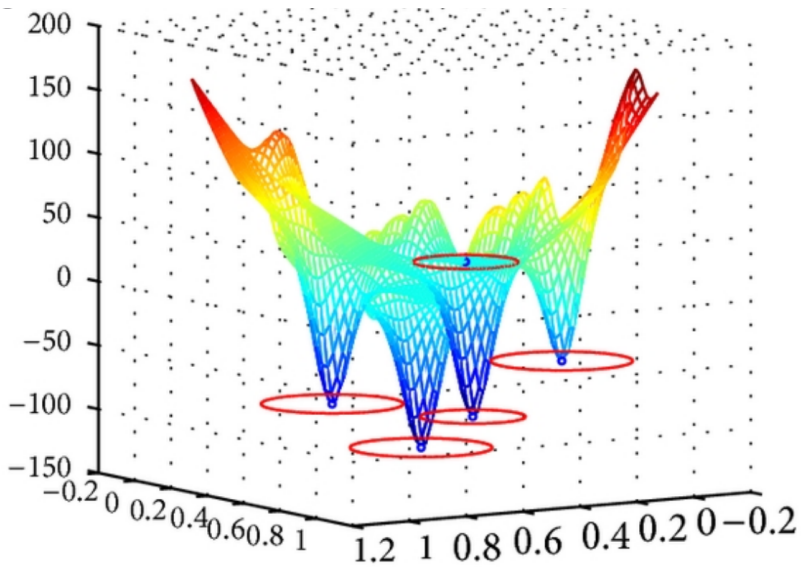


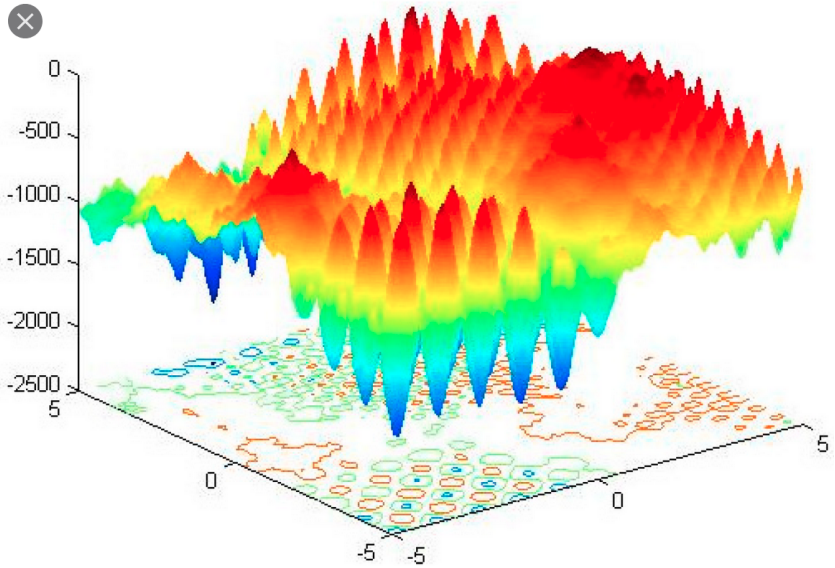
With $\lambda = 1$



With $\lambda = 5$

Multi-solution problem: Find a set of solutions, $\mathcal{X} = \{\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_{n_\mathcal{X}}^*\}$, such that each $\mathbf{x}^* \in \mathcal{X}$ is a minimum of the optimization problem





Also referred to as **multi-modal optimization**

Dynamic optimization problem:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}, \varpi_x(t)), \quad \mathbf{x} = (x_1, \ldots, x_{n_x}), \varpi_x(t) = (\varpi_1(t), \ldots, \varpi_{n_\varpi}) \\ \text{subject to} \quad & g_m(\mathbf{x}, \varpi_g(t)) \leq 0, \ m = 1, \ldots, n_g \\ & h_m(\mathbf{x}, \varpi_h(t)) = 0, \ m = n_g + 1, \ldots, n_g + n_h \\ & x_j \in \text{dom}(x_j) \end{aligned}$$

$\varpi(t)$ is a vector of time-dependent objective function parameters; $\varpi_g(t)$ and $\varpi_h(t)$ are parameters which changes the constraint functions

The objective is to find and to track

$$\mathbf{x}^*(t) = \min_{\mathbf{x}} f(\mathbf{x}, \varpi(t))$$

where $\mathbf{x}^*(t)$ is a global minimum found at time step t

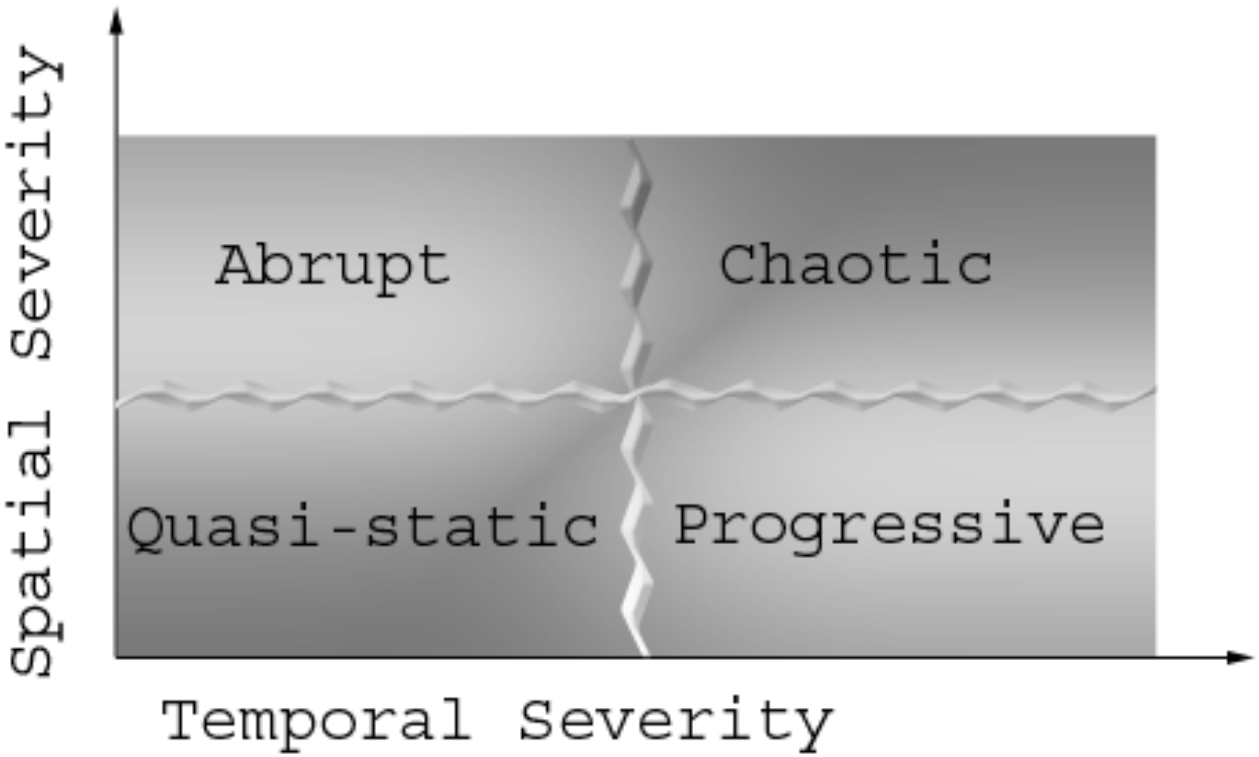
Based on effect on optima:

- **Type I environments**, where the location of the optimum in problem space is subject to change. The change in the optimum, $\mathbf{x}^*(t)$ is quantified by the severity parameter, ζ, which measures the jump in location of the optimum.
- **Type II environments**, where the location of the optimum remains the same, but the value, $f(\mathbf{x}^*(t))$, of the optimum changes.
- **Type III environments**, where both the location of the optimum and its value changes.

Optimization Problem Classes

Dynamic Environment Types (cont)

Based on change severity and change frequency



Optimization Problem Classes

Dynamic Environment Types (cont)

Based on change trajectory:

- Cyclic
- Linear
- Random

Combinations of above classifications result in 27 different classes of dynamic optimization problems

Number of decision variables can also change over time

The Moving Peaks Benchmark generator can be used to generate instances for each of these problem classes

Consider the following dynamic optimization problem:

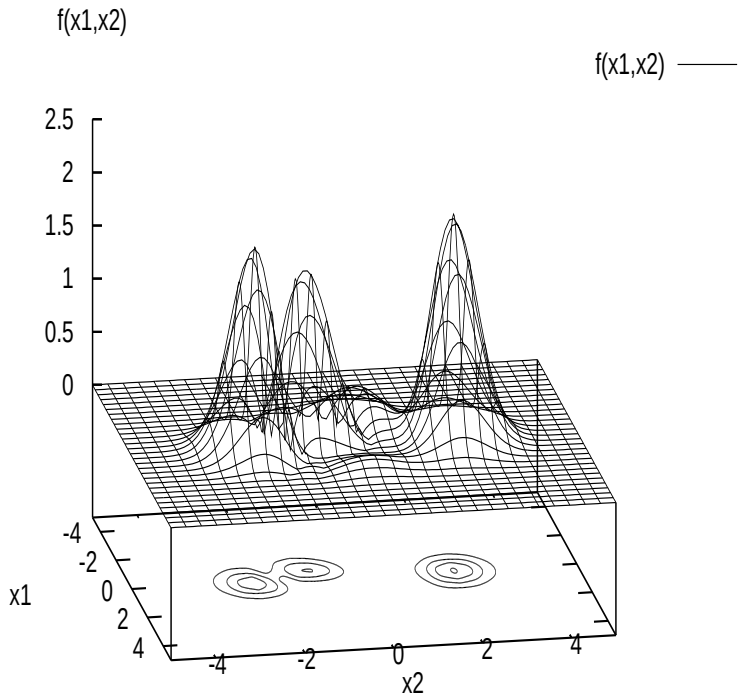
$$f(\mathbf{x}, \varpi) = |f_1(\mathbf{x}, \varpi) + f_2(\mathbf{x}, \varpi) + f_3(\mathbf{x}, \varpi)|$$

with

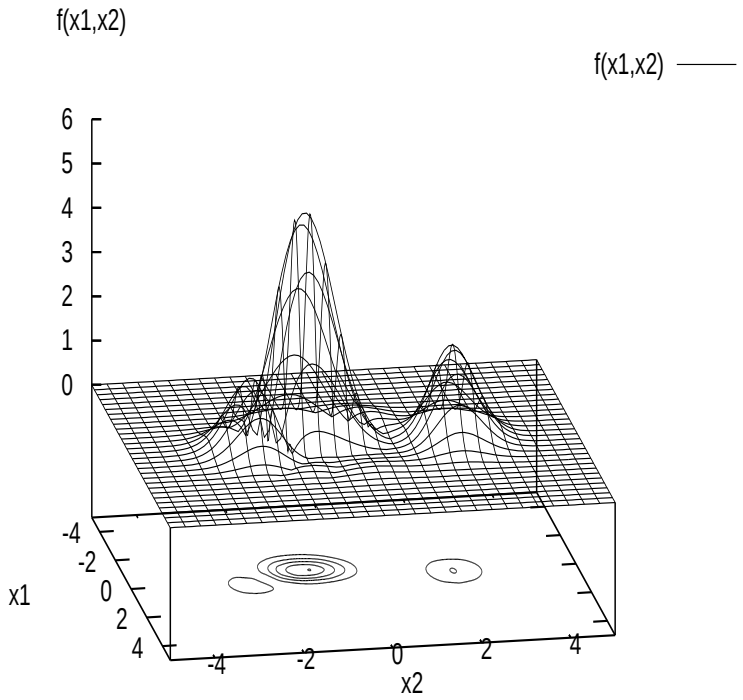
$$\begin{aligned} f_1(\mathbf{x}, \varpi) &= \varpi_1 (1 - x_1)^2 e^{(-x_1^2 - (x_2+1)^2)} \\ f_2(\mathbf{x}, \varpi) &= -0.1 \left(\frac{x_1}{5} - \varpi_2 x_1^3 - x_2^5 \right)^2 e^{(-x_1^2 - x_2^2)} \\ f_3(\mathbf{x}, \varpi) &= 0.5 e^{(-(x_1+1)^2 - x_2^2)} \end{aligned}$$

Optimization Problem Classes

Dynamic Optimization Problem Example (cont)



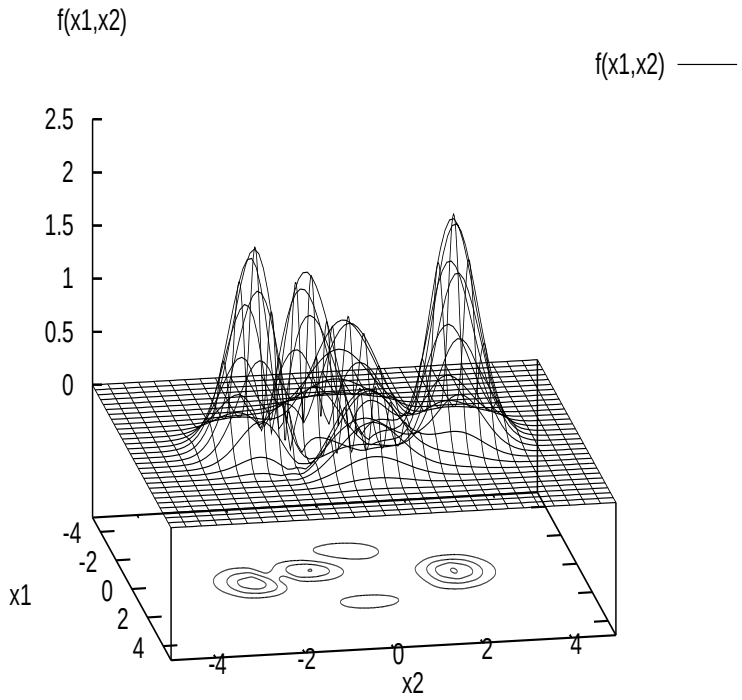
(a) $\varpi_1 = \varpi_2 = 0$



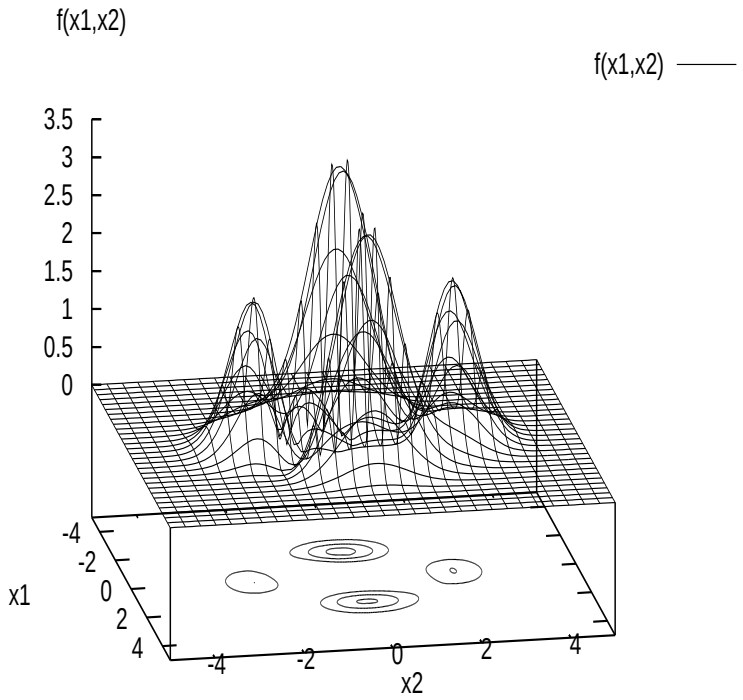
(b) $\varpi_1 = 3, \varpi_2 = 0$

Optimization Problem Classes

Dynamic Optimization Problem Example (cont)



(c) $\varpi_1 = 0, \varpi_2 = 50$



(d) $\varpi_1 = 3, \varpi_2 = 50$

Multi-objective optimization problem:

$$\begin{aligned} \text{minimize} \quad & \mathbf{f}(\mathbf{x}) \\ \text{subject to} \quad & g_m(\mathbf{x}) \leq 0, \quad m = 1, \ldots, n_g \\ & h_m(\mathbf{x}) = 0, \quad m = n_g + 1, \ldots, n_g + n_h \\ & \mathbf{x} \in [\mathbf{x}_{min}, \mathbf{x}_{max}]^{n_x} \end{aligned}$$

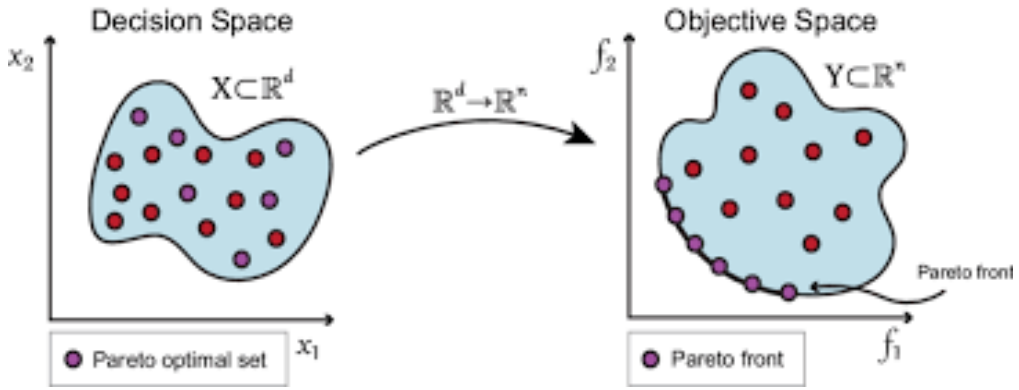
where $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_{n_k}(\mathbf{x})) \in \mathcal{O} \subseteq \mathbb{R}^{n_k}$

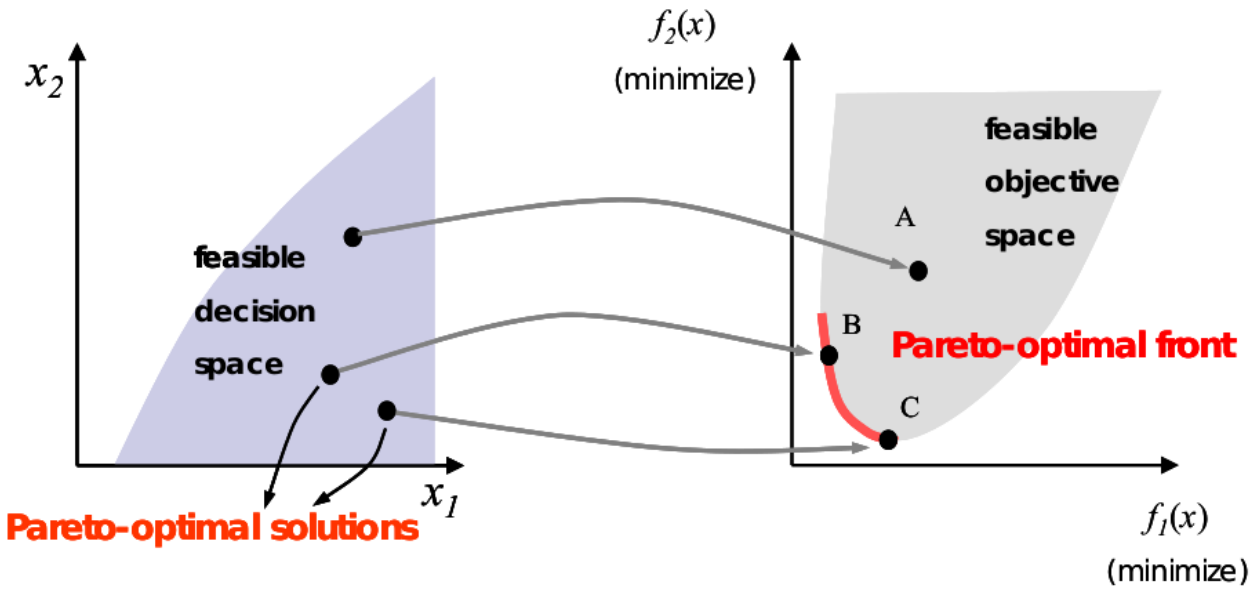
$\mathcal{O}$ is referred to as the *objective space*

The search space, $\mathcal{S}$, is also referred to as the *decision space*

Optimization Problem Classes

Multi-Objective Optimization Problems Illustrated





Problem examples:

- Imagine buying a car. You want to achieve the following goals:
  - minimise the cost
  - maximise the comfort
- These goals are in conflict with one another
  $\Rightarrow$ maximising the comfort increases the cost and vice versa
- Single solution does not exist $\Rightarrow$ Example of a MOP



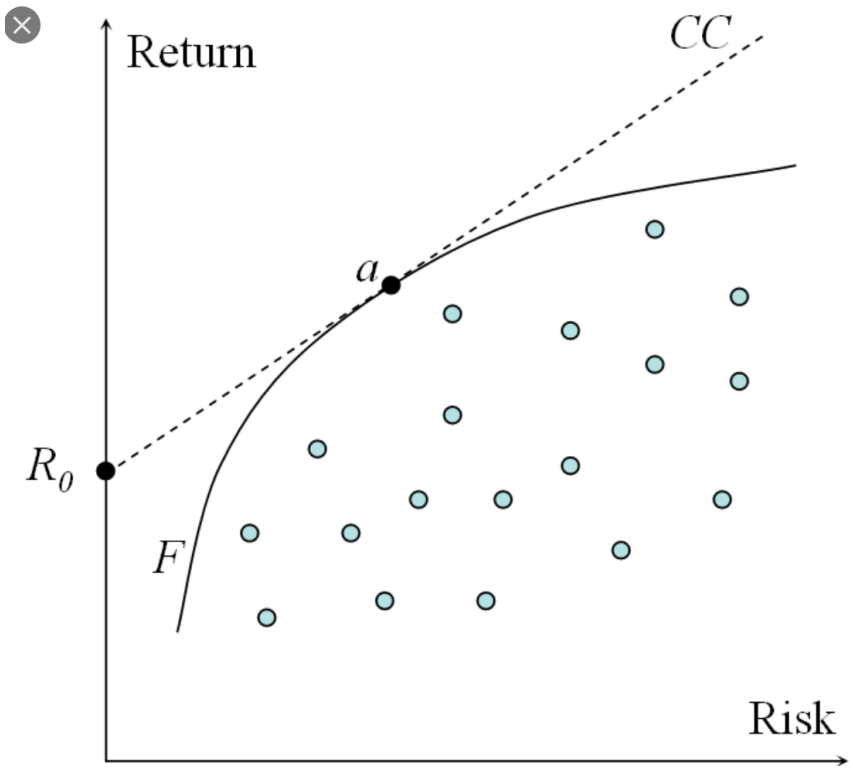


From computational finance: Portfolio optimization

Objectives:

- Minimize risk
- Maximize profit

Under the constraint of investment amount and minima and maxima per financial asset invested in



The problem is the presence of conflicting objectives

Need to achieve a balance between these objectives

A balance is achieved when a solution can not improve any objective without degrading one or more of the other objectives

There is usually not just one solution

Solutions are referred to as *non-dominated solutions*

Set of solutions is referred to as the Pareto-optimal set, and the corresponding objective vectors are referred to as the Pareto front

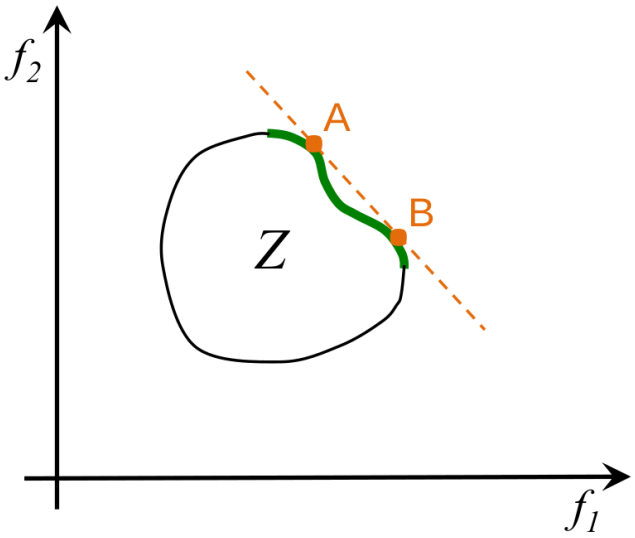
Weighted Aggregation Definition:

$$
\begin{array}{ll}
\text{minimize} & \sum_{k=1}^{n_k} \omega_k f_k(\mathbf{x}) \\
\text{subject to} & g_m(\mathbf{x}) \leq 0, \quad m = 1, \ldots, n_g \\
& h_m(\mathbf{x}) = 0, \quad m = n_g + 1, \ldots, n_g + n_h \\
& \mathbf{x} \in [\mathbf{x}_{min}, \mathbf{x}_{max}]^{n_x} \\
& \omega_k \geq 0, k = 1, \ldots, n_k
\end{array}
$$

It is also usually assumed that $\sum_{k=1}^{n_k} \omega_k = 1$

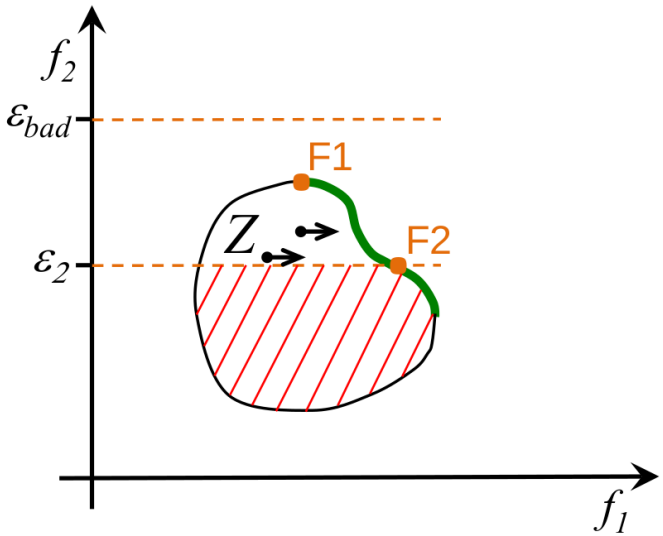
Aggregation methods have the following problems:

- The algorithm has to be applied repeatedly to find different solutions if a single-solution algorithm is used
- It is difficult to get the best weight values, ω_k, since these are problem-dependent
- Normalize objective functions
- Aggregation methods can only be applied to generate members of the Pareto-optimal set when the Pareto front is convex by varying the values of ω_k
- For non-convex Pareto fronts, it is possible that parts of the front can not be obtained by the weighted sum



One objective is selected to be optimized, and all others are handled as inequality constraints:

$$\begin{aligned} \text{minimize} \quad & f_k(\mathbf{x}), \quad \mathbf{x} = (x_1, \ldots, x_{n_x}), \quad k \in \{1, \ldots, n_k\} \\ \text{subject to} \quad & f_l(\mathbf{x}) \geq \epsilon_l, \quad l = 1, \ldots, n_k, \; l \neq k \\ & x_j \in \text{dom}(x_j) \end{aligned}$$



- Choose the most important objective to optimize
- Carefully choose values for ϵ_l
- Constrains the objective space, dividing in feasible and infeasible objective space

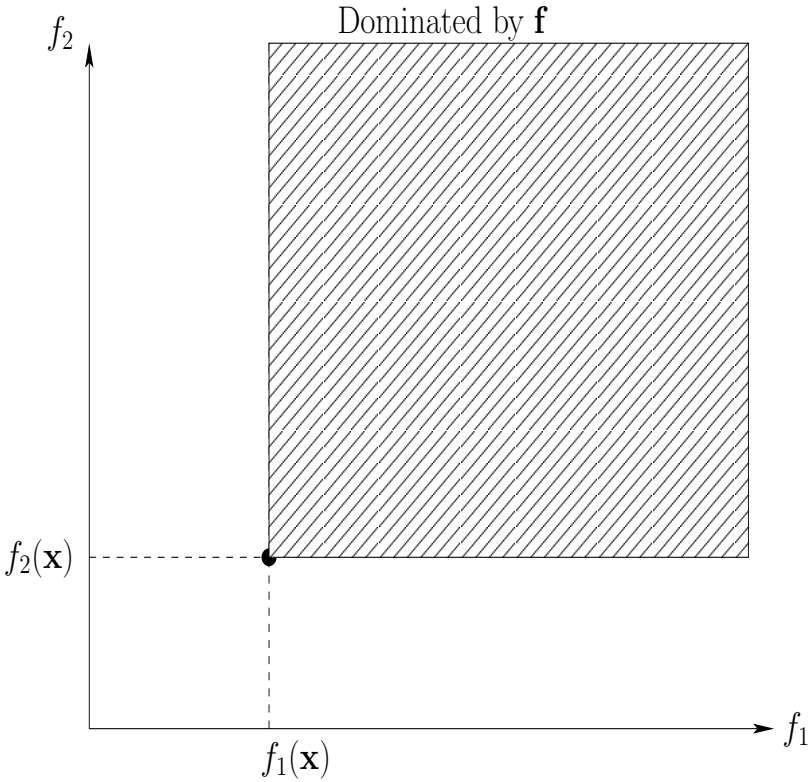
Domination: *A decision vector, $\mathbf{x}_1$ dominates a decision vector, $\mathbf{x}_2$ (denoted by $\mathbf{x}_1 \prec \mathbf{x}_2$), if and only if*

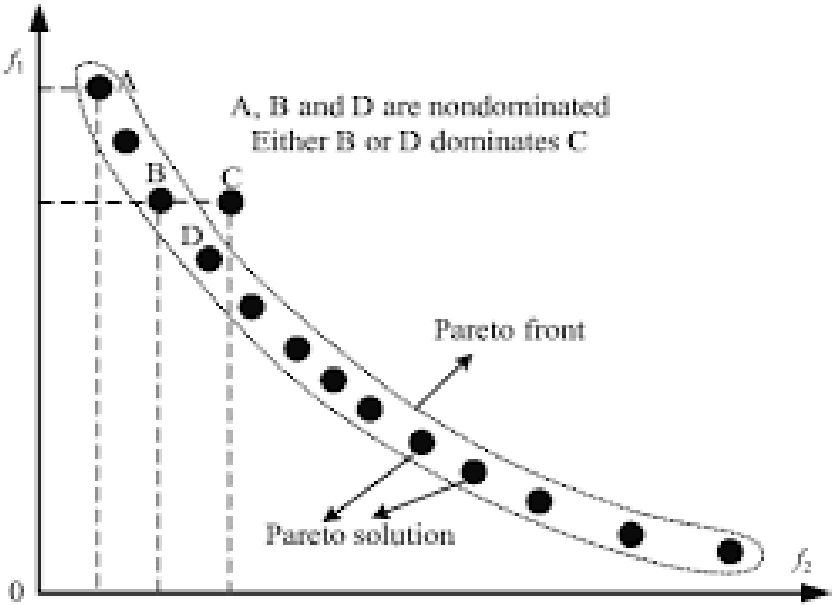
- $\mathbf{x}_1$ *is not worse than* $\mathbf{x}_2$ *in all objectives, i.e.* $f_k(\mathbf{x}_1) \leq f_k(\mathbf{x}_2), \forall k = 1, \ldots, n_k$, *and*
- $\mathbf{x}_1$ *is strictly better than* $\mathbf{x}_2$ *in at least one objective, i.e.* $\exists k = 1, \ldots, n_k : f_k(\mathbf{x}_1) < f_k(\mathbf{x}_2)$.

So, solution $\mathbf{x}_1$ is better than solution $\mathbf{x}_2$ if $\mathbf{x}_1 \prec \mathbf{x}_2$ (i.e. $\mathbf{x}_1$ dominates $\mathbf{x}_2$), which happens when $\mathbf{f}_1 \prec \mathbf{f}_2$

Optimization Problem Classes

Multi-Objective Optimization Problems: Pareto-Optimality (cont)





Pareto-optimal: *A decision vector, $\mathbf{x}^* \in \mathcal{F}$ is Pareto-optimal if there does not exist a decision vector, $\mathbf{x} \neq \mathbf{x}^* \in \mathcal{F}$ that dominates it. That is, $\nexists k : f_k(\mathbf{x}) < f_k(\mathbf{x}^*)$. An objective vector, $\mathbf{f}^*(\mathbf{x})$, is Pareto-optimal if $\mathbf{x}$ is Pareto-optimal.*

Pareto-optimal set: *The set of all Pareto-optimal decision vectors form the Pareto-optimal set, $\mathcal{P}^*$. That is,*

$$\mathcal{P}^* = \{\mathbf{x}^* \in \mathcal{F} | \ \nexists \mathbf{x} \in \mathcal{F} : \mathbf{x} \prec \mathbf{x}^*\}$$

Pareto-optimal front: *Given the objective vector, $\mathbf{f}(\mathbf{x})$, and the Pareto-optimal solution set, $\mathcal{P}^*$, then the Pareto-optimal front, $\mathcal{PF}^* \subseteq \mathcal{O}$, is defined as*

$$\mathcal{PF}^* = \{\mathbf{f} = (f_1(\mathbf{x}^*), f_2(\mathbf{x}^*), \ldots, f_k(\mathbf{x}^*)) | \mathbf{x}^* \in \mathcal{P}\}$$

With a convex, non-uniform Pareto front:

$$\begin{aligned} \mathbf{f}(\mathbf{x}) &= (f_1(\mathbf{x}), f_2(\mathbf{x})) \\ f_1(\mathbf{x}) &= x_1 \\ f_2(\mathbf{x}) &= g(\mathbf{x})(1 - \sqrt{f_1(\mathbf{x})/g(\mathbf{x})}) \end{aligned} \tag{1}$$

where

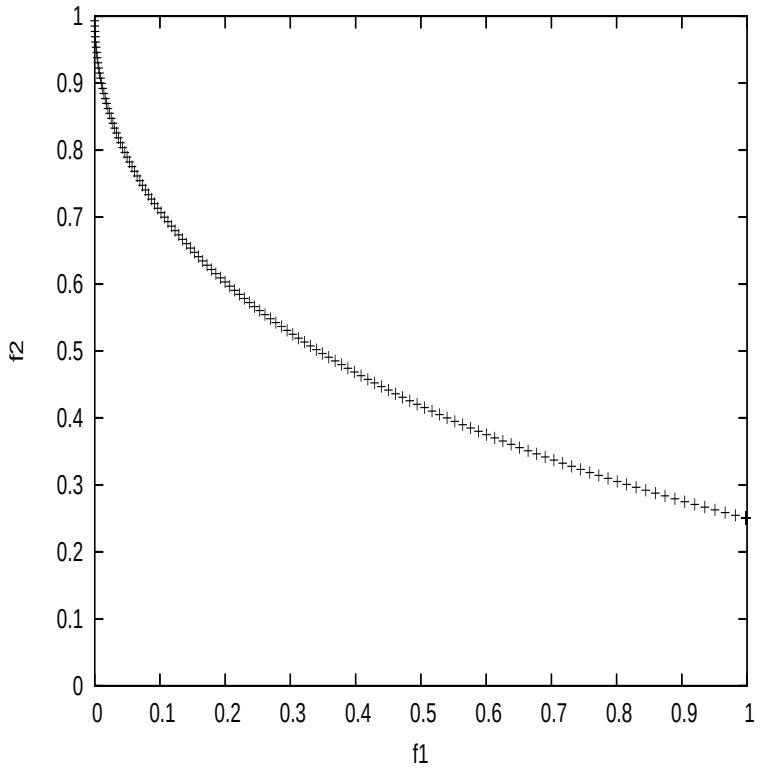
$$g(\mathbf{x}) = 1 + \frac{9}{n_x - 1} \sum_{j=2}^{n_x} x_j$$

With partially concave and partially convex Pareto front using the above equation, but with

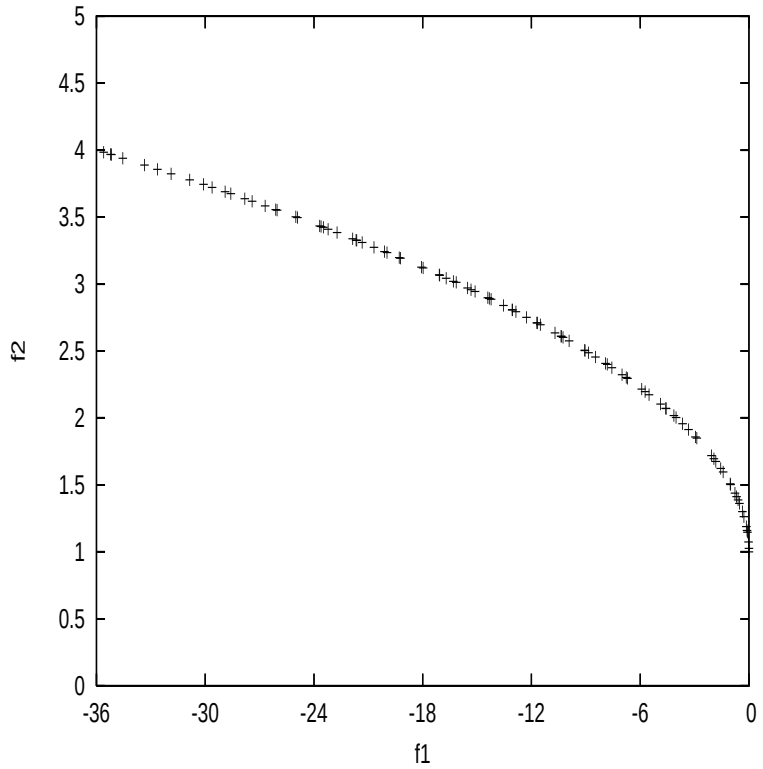
$$f_2(\mathbf{x}) = g(\mathbf{x})(1 - \sqrt[4]{f_1(\mathbf{x})/g(\mathbf{x})} - (f_1(\mathbf{x})/g(\mathbf{x}))^4)$$

Optimization Problem Classes

Multi-Objective Optimization Problem Examples(cont)



(a) Convex POF



(b) Concave POF

Optimization Problem Classes

Many-Objective Optimization Problem

Multi-objective optimization refers to problems where there are two or three sub-objectives, i.e. $2 \leq n_k \leq 3$

Many-objective optimization refers to problems where there are more than three sub-objectives, i.e. $n_k \geq 4$

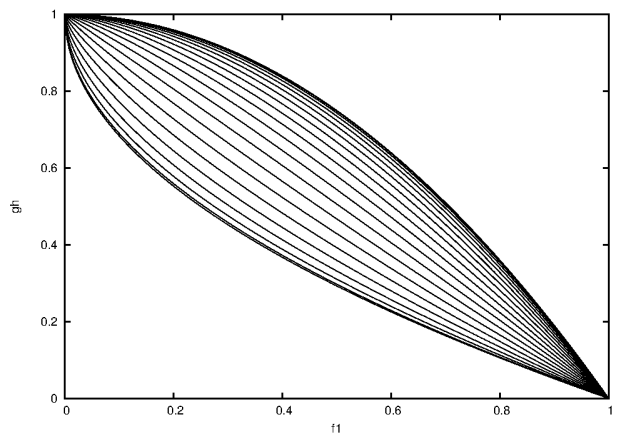
Problems with many-objective optimization:

- The more objectives a problem has, the more solutions to the problem will be non-dominated compared to other solutions that have been found
- Many multi-objective optimization algorithms fail to solve many-objective optimization problems, due to the difficulty of guiding the algorithm's search when many solutions are non-dominated with regards to each other
- Any operator used to preserve diversity becomes computationally expensive to calculate for many objectives

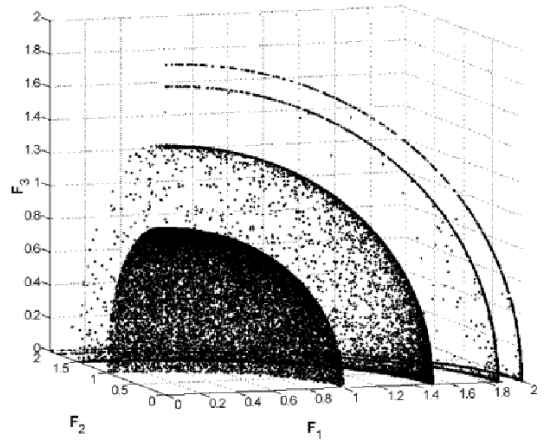
Optimization Problem Classes

Dynamic Multi-Objective Optimization Problems

- For the portfolio optimization problem, you have invested in Steinhoff...
- Risk increased significantly, and profits dove significantly
- Search space changed due to changes in invested assets
- Need to find a new optimal basket of assets
- Pareto front changes over time



(c) POF of dMOP2



(d) POF of FDA5

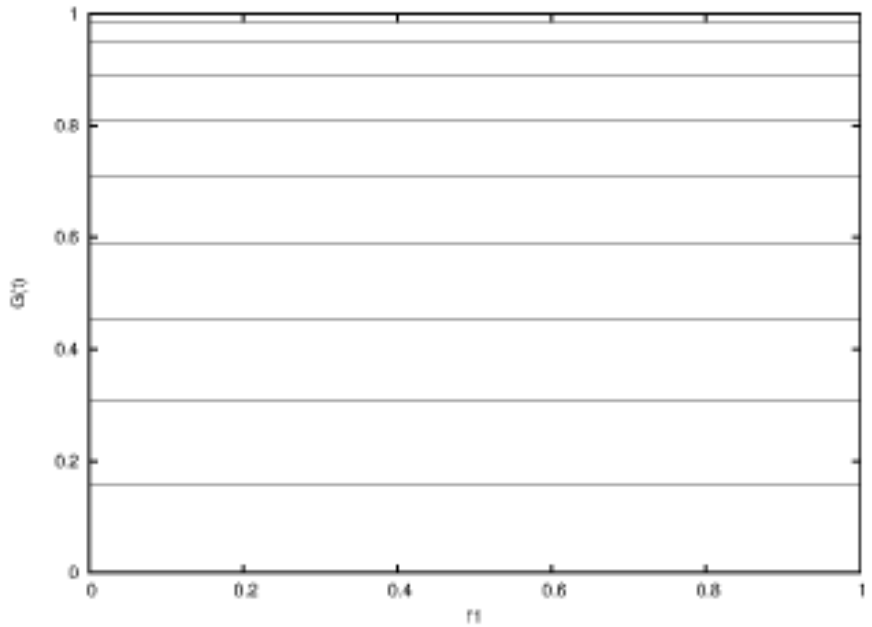
Four categories of dynamic environments for DMOOPs:

	POS	
POF	No Change	Change
No Change	Type IV	Type I
Change	Type III	Type II

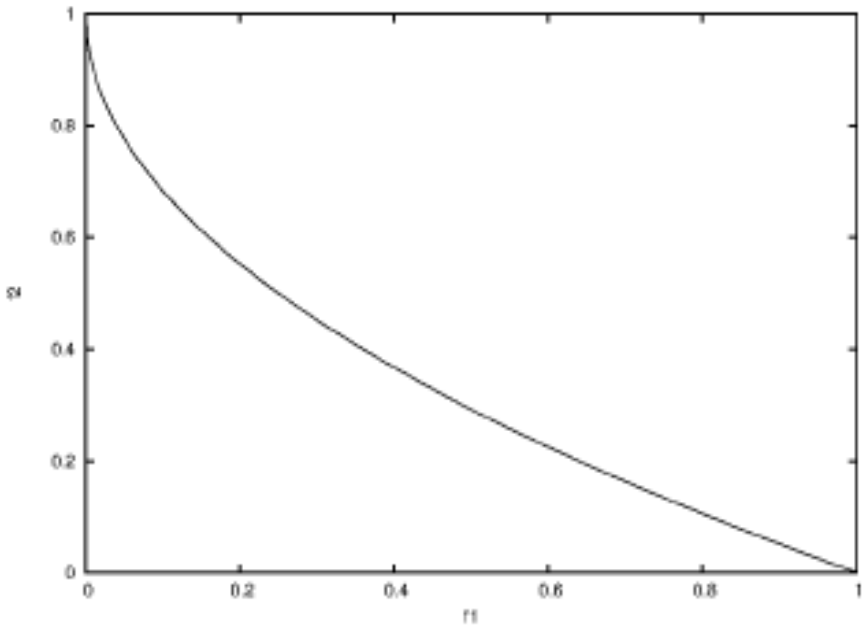
Other considerations:

- Frequency of change
- Severity of change

Has a convex POF



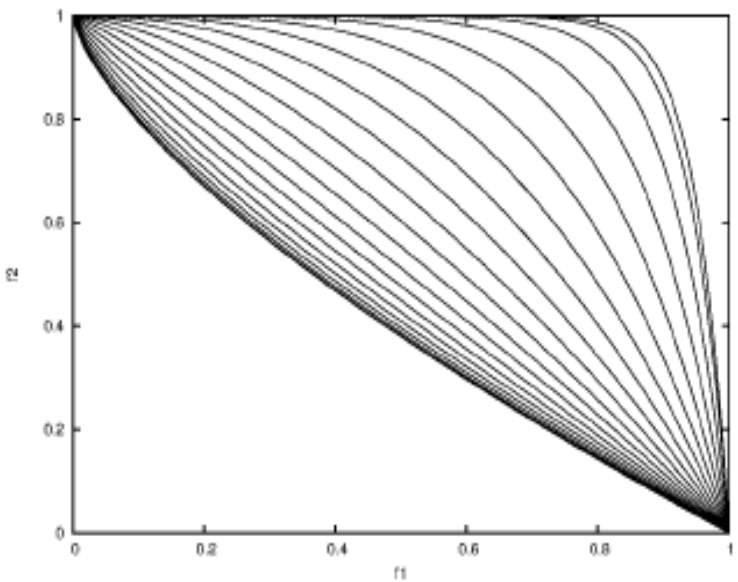
(a) POS



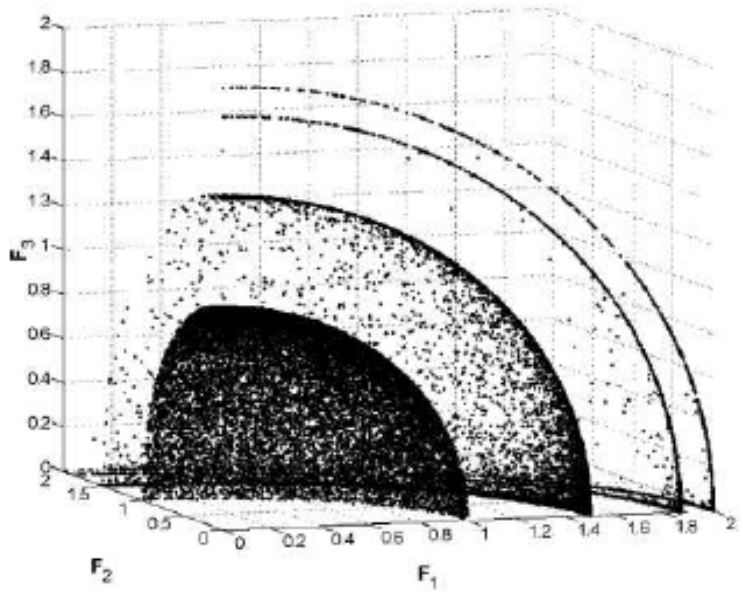
(b) POF

POF changes from convex to concave

Non-convex POF
Spread of solutions change over time



(e) FDA2



(f) FDA5

Optimization Problem Classes

Dynamic Multi-Objective Optimization Problems:Type III

Discontinuous POF

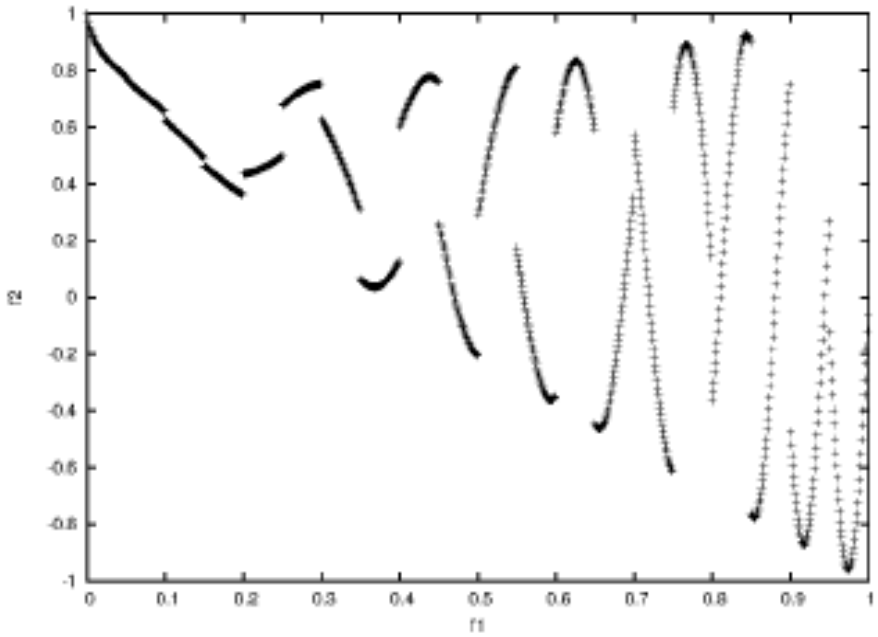
$$POF = 1 - \sqrt{f_1} - f_1 \sin(10\pi t f_1)$$

Discontinuous POF

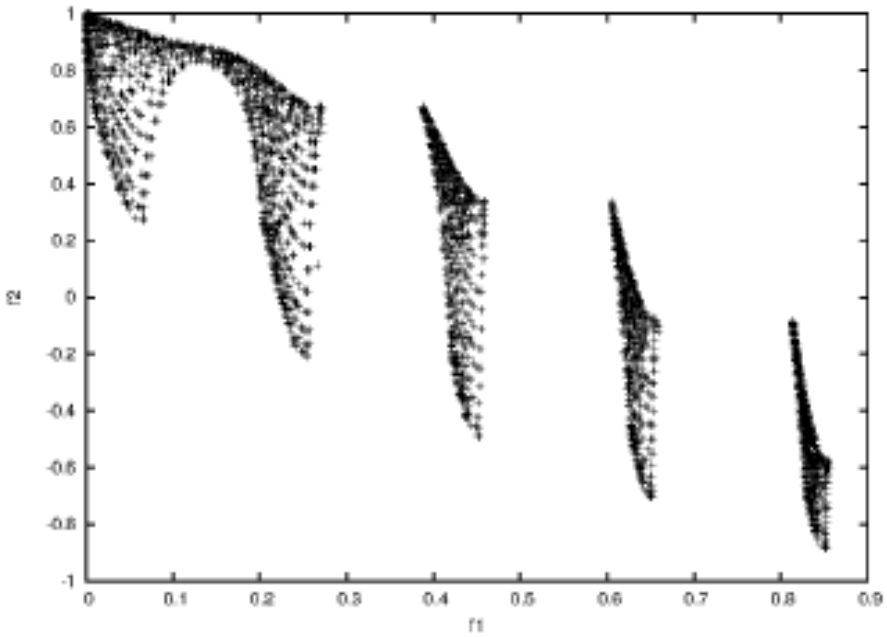
$$POF = 1 - \sqrt{f_1}^{H(t)} - f_1^{H(t)} \sin(10\pi t f_1)$$

$$H(t) = 0.75 \sin(0.5\pi t) + 1.25$$

$$t = \frac{1}{n_t} \lfloor \frac{\tau}{\tau_t} \rfloor$$



(a) POF of HE1



(b) POF of HE2

Other problem classes as combinations of the above:

- Tracking multiple optima in dynamic environments
- Dynamically constrained static or dynamic optimization problems
- Statically or dynamically constrained static multi-objective or dynamic multi-objective optimization problems

Then, there are also large-scale optimization problems