Appendix A: Optimization Theory

Computational Intelligence: Second Edition

Contents

Basic Ingredients

Each optimization problem consists of the following basic ingredients:

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

Problem Classificaion

Optimization problems are classified based on a number of characteristics:

- The **number of variables** that influences the objective function
- The **type of variables**
- The **degree of nonlinearity of the objective function**
- The **constraints used**
- The **number of optima**
- The **number of optimization criteria**
- Whether the **objective function changes** over time.

Global Minimum

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

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

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

Local Minimum

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

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

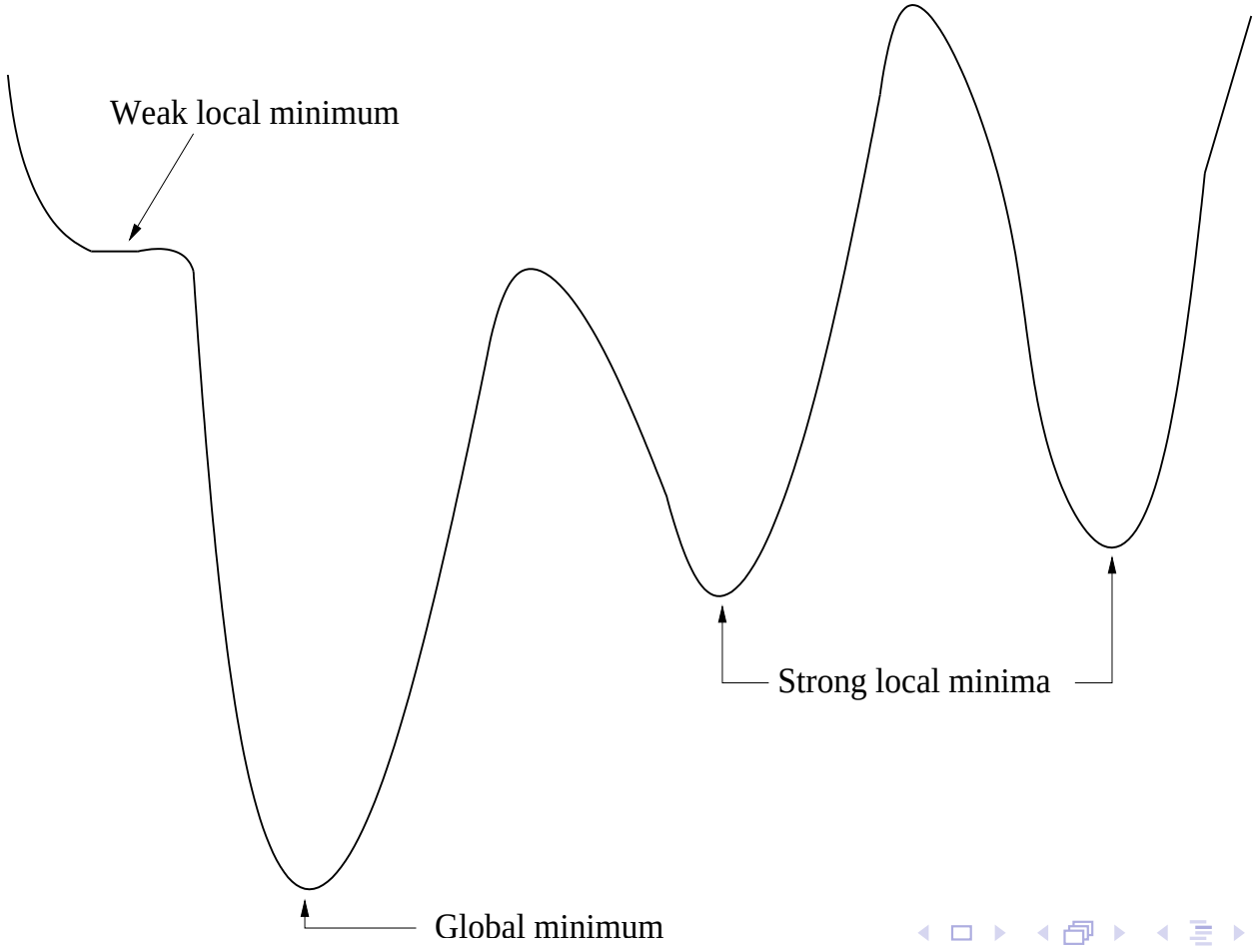
where $\mathcal{N} \subseteq \mathcal{F}$ is a set of feasible points in the neighborhood of $\mathbf{x}^*_{\mathcal{N}}$.

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 $\mathbf{x}^*_{\mathcal{N}}$.

Figure A.1: Types of Optima for Unconstrained Problems



Classes of Optimization Algorithms

Based on the type of solution:

- Local search algorithms
- Global search algorithms

Based on the way in which candidate solutions are updated:

- Deterministic
- Stochastic

Classes of Optimization Algorithms (cont)

Based on the problem characteristics:

- **unconstrained methods**, used to optimize unconstrained problems
- **constrained methods**, used to find solutions in constrained search spaces
- **multi-objective optimization methods** for problems with more than one objective to optimize
- **multi-solution (niching) methods** with the ability to locate more than one solution
- **dynamic methods** with the ability to locate and track changing optima

Definition

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

Some Optimization Algorithms

Algorithm 1: 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;

Some Optimization Algorithms (cont)

Beam search

Tabu search

Gradient descent

LeapFrog

Some Optimization Algorithms (cont)

Algorithm 2: 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

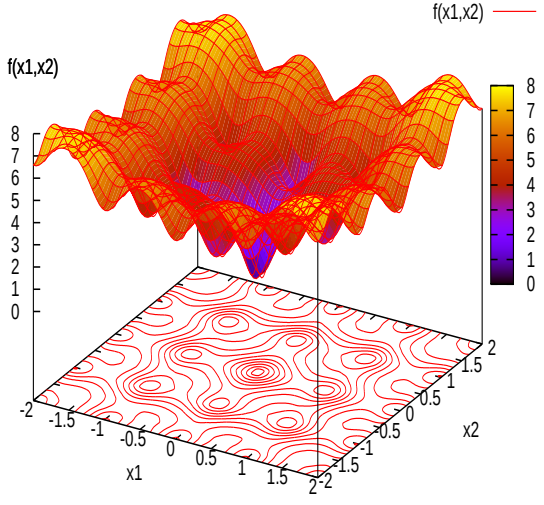
$$P = \begin{cases} 1 & \text{if } f(\mathbf{x}) < f(\mathbf{x}(t)) \\ e^{-\frac{f(\mathbf{x})-f(\mathbf{x}(t-1))}{c_b T}} & \text{otherwise} \end{cases}$$

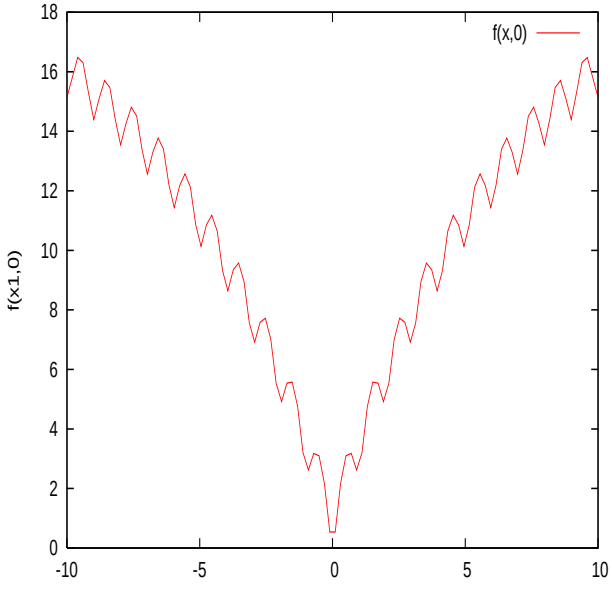
if $U(0,1) \leq$ *acceptance probability* **then**
$\mathbf{x}(t+1) = \mathbf{x}$;
end
t=t+1;
until *stopping condition is true* ;
Return $\mathbf{x}(t)$ as the solution;

Examples: Ackley

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

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

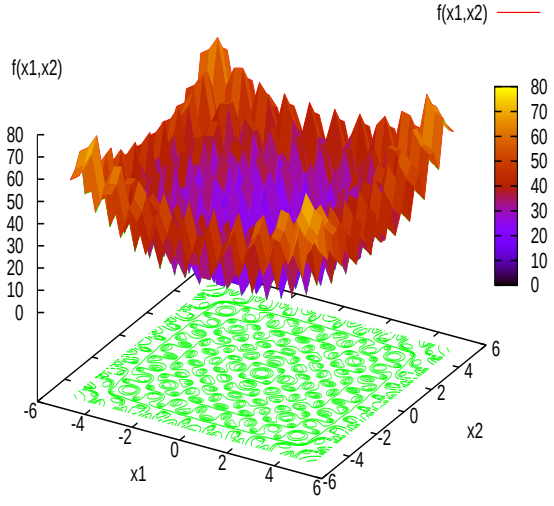


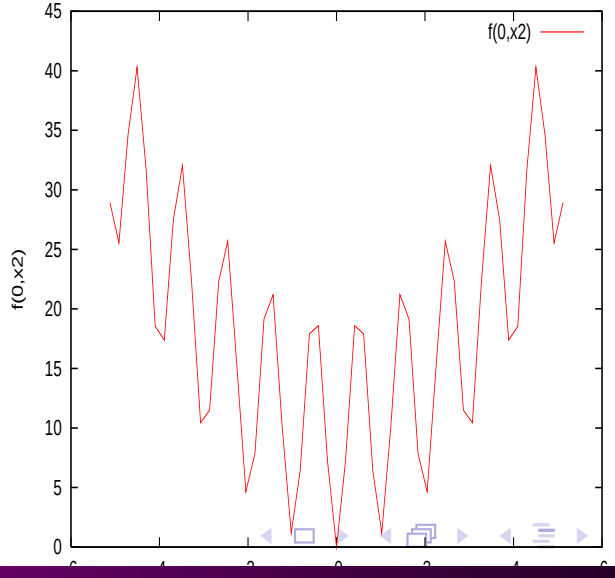


Examples: Rastrigin

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

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





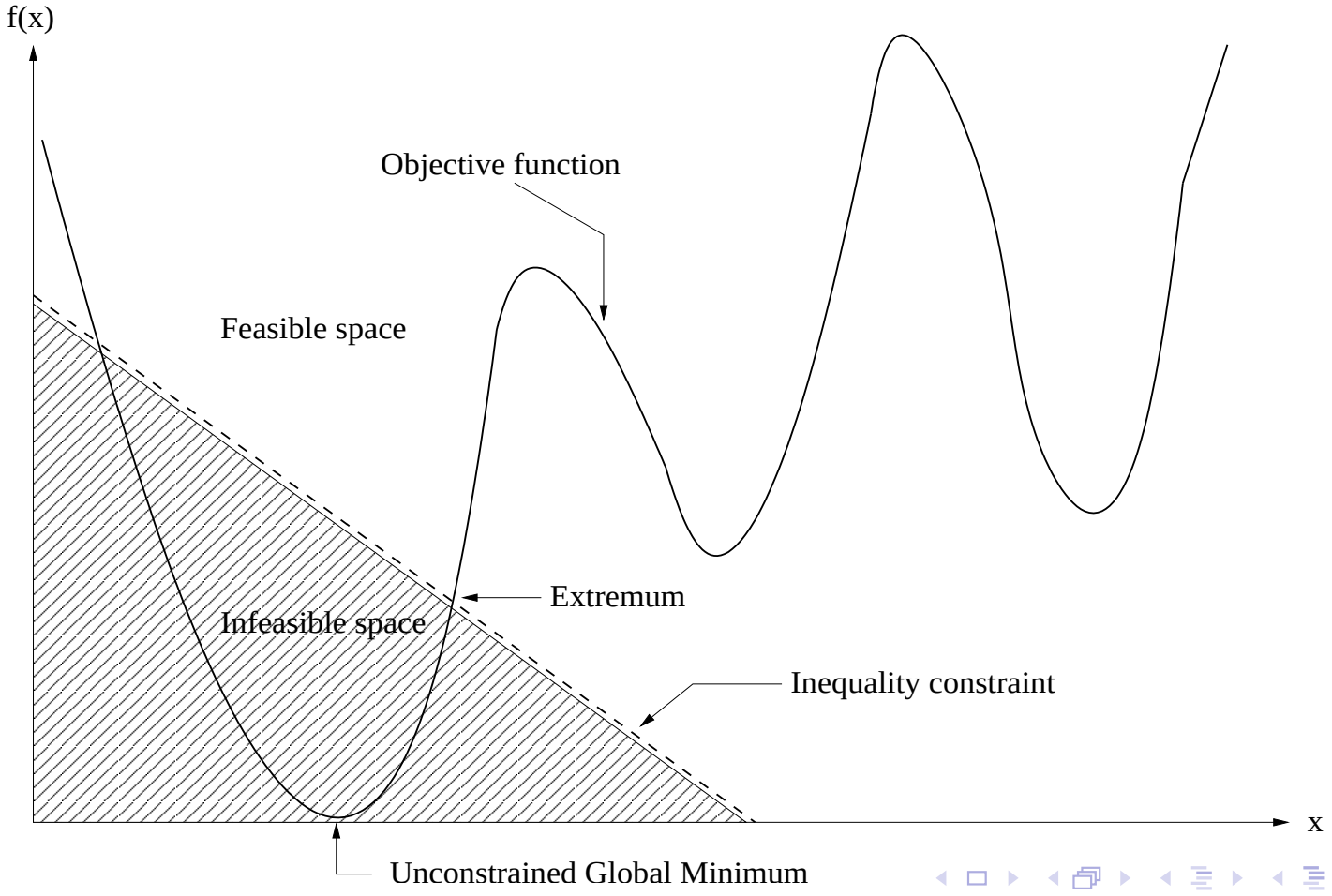
Definition

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 resp ectively

Figure A.3: Constrained Problem Illustration



Constraint Handling Methods

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?
- Should it be assumed that any feasible solution is better than any unfeasible solution? Alternatively, can objective function value and degree of violation be optimally balanced?

Penalty Methods

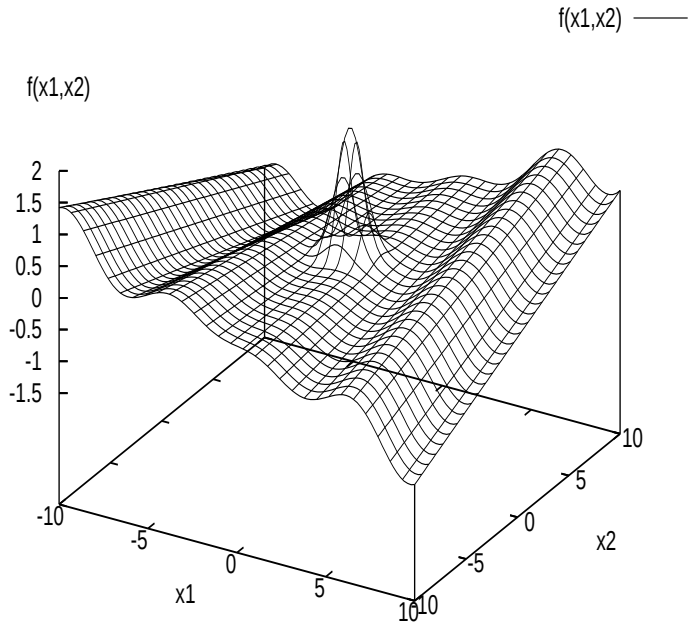
Minimization using the Penalty method:

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

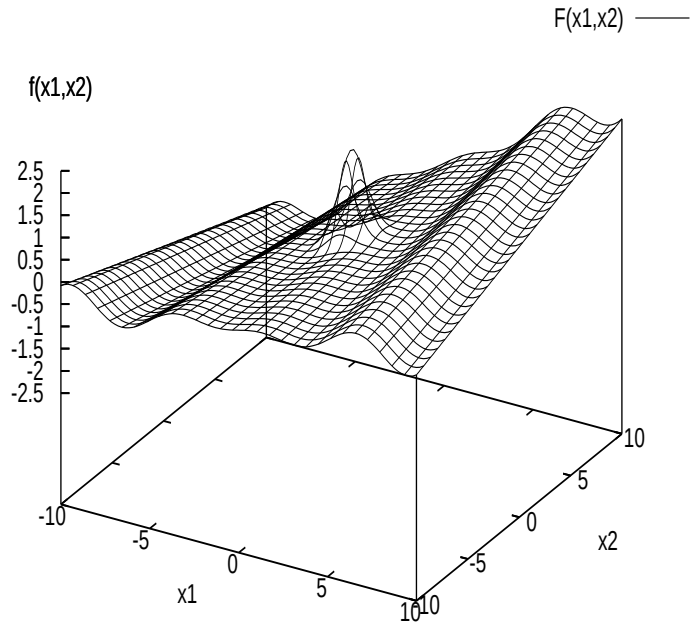
where λ is the penalty coefficient and $p(\mathbf{x}, t)$ is the (possibly) time-dependent penalty function

What are the problems with penalty methods?

Figure A.4: Effect of Penalty Functions



(e) Original function



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

Constructing the Penalty Function

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

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 to an Unconstrained Problem

Covert 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):

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

Convert to an Unconstrained Problem (cont)

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,

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

Definition

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 general optimization problem

Definition

Dynamic optimization problem:

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

where $\varpi(t)$ is a vector of time-dependent objective function control parameters. The objective is to find

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

where $\mathbf{x}^*(t)$ is the optimum found at time step t.

Dynamic Environment Types

- **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.

Example

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

with

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

Figure A.8

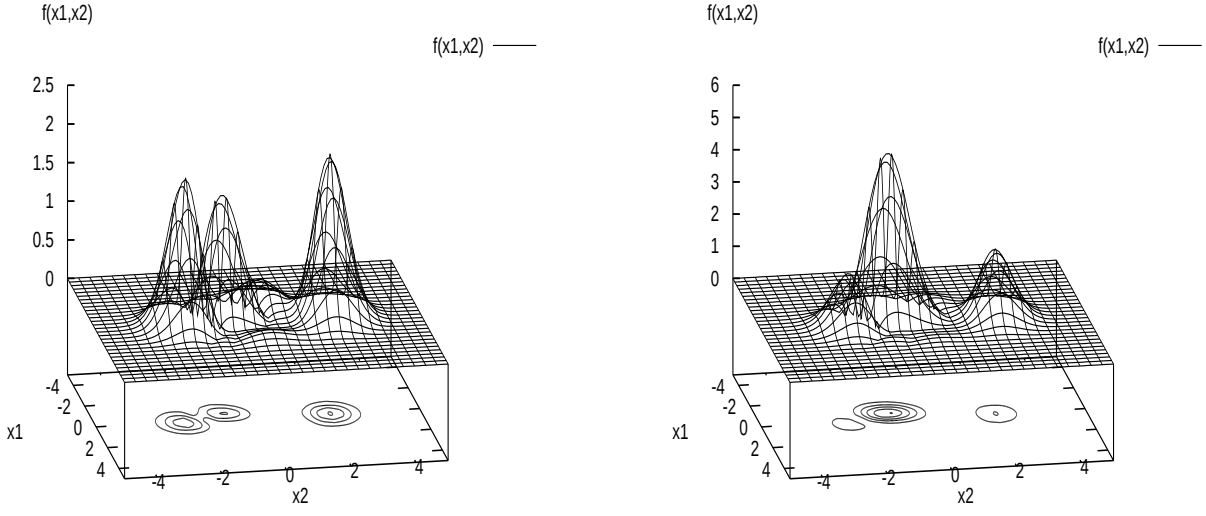
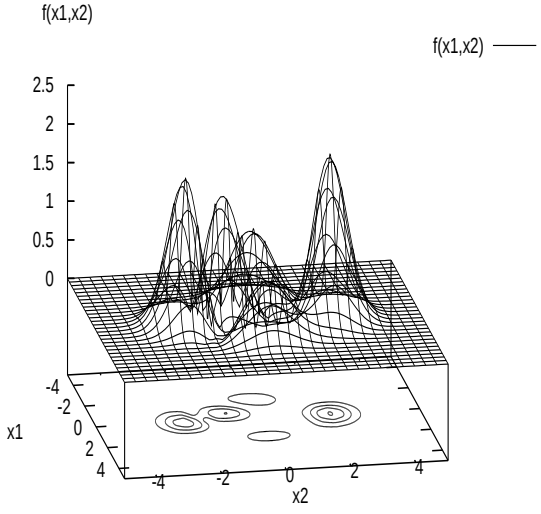
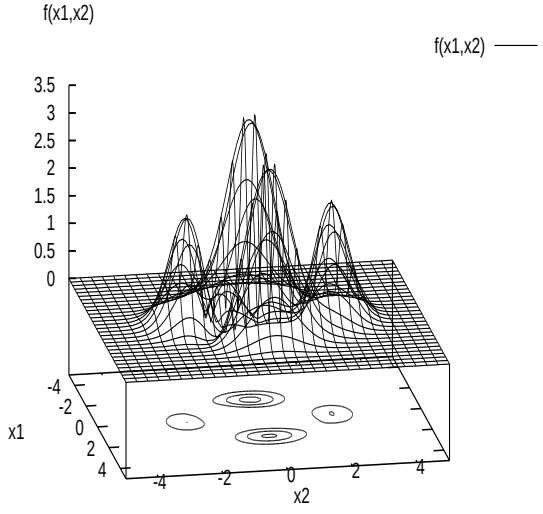

f(x1,x2)
f(x1,x2)
2.5
2
1.5
1
0.5
0
x1
-4
-2
0
2
4
x2
-4
-2
0
2
4
f(x1,x2)
f(x1,x2)
6
5
4
3
2
1
0
x1
-4
-2
0
2
4
x2
-4
-2
0
2
4

Figure A.8 (cont)





Definition

Multi-objective 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*

Meaning of an Optimum

The problem is the presence of conflicting objectives

Need to achieve a balance between these objectives

A balance is achived when a solution cannot improve any objective without degrading one or more of the other objectives

There is 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 Methods

Definition:

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

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

Weighted Aggregation Methods (cont)

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
- Aggregation methods can only be applied to generate members of the Pareto-optimal set when the Pareto front is concave, regardless of the values of ω_k

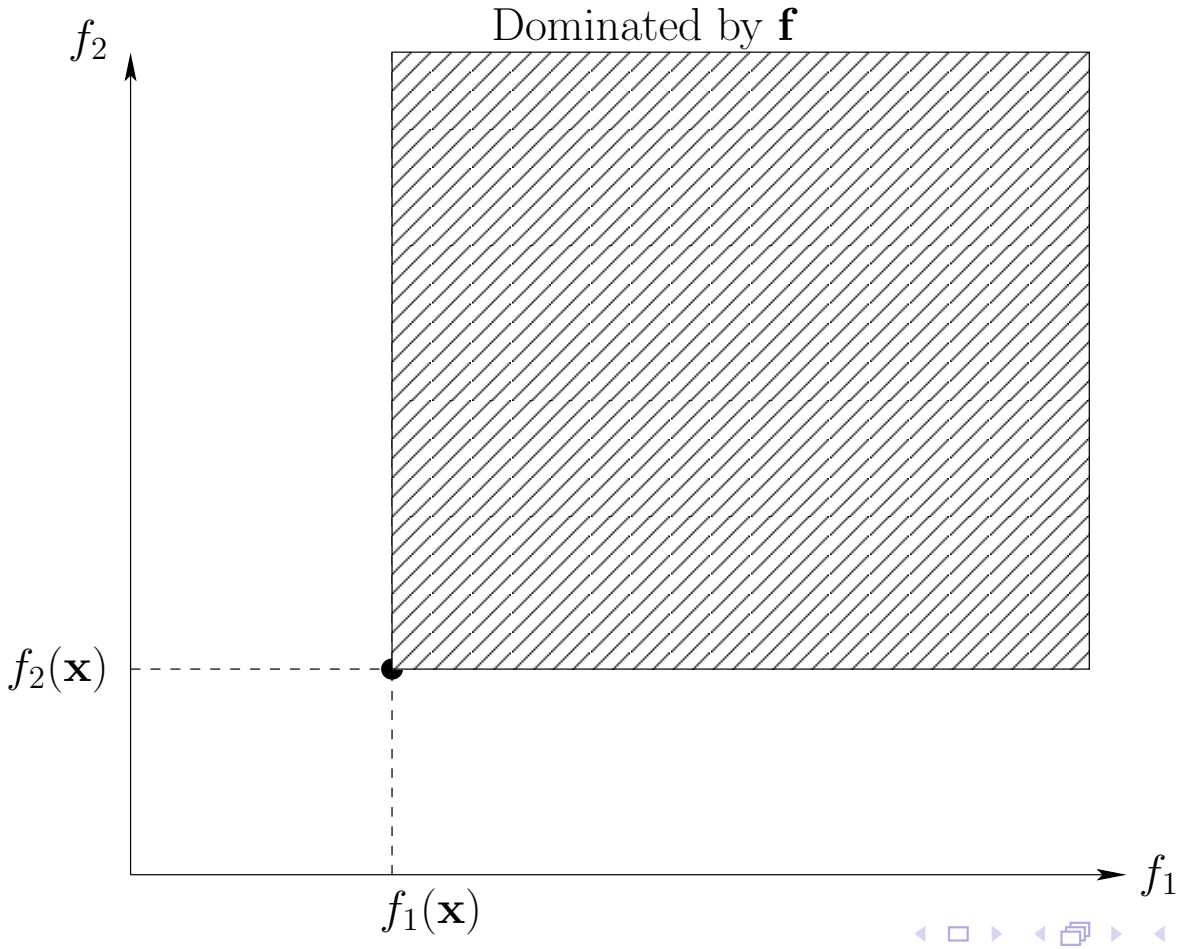
Pareto-Optimality

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 $\mathbf{f}_1 \prec \mathbf{f}_2$

Figure A.5: Illustration of Dominance



More Definitions

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*

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

Examples of MOPs

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

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

Figure A.6: Example Pareto-Optimal Fronts