

Introductory Course: Using LS-OPT[®] on the TRACC Cluster

3 - Numerical Optimization

By: Vadim Sokolov, PhD

Some of the slides are adopted from lecture notes from the MIT course 16.888 / ESD.77 “*Multidisciplinary System Design Optimization*”

Some of the figures were adapted from *Numerical Optimization*, Jorge Nocedal and Stephen J. Wright, Springer, 1999

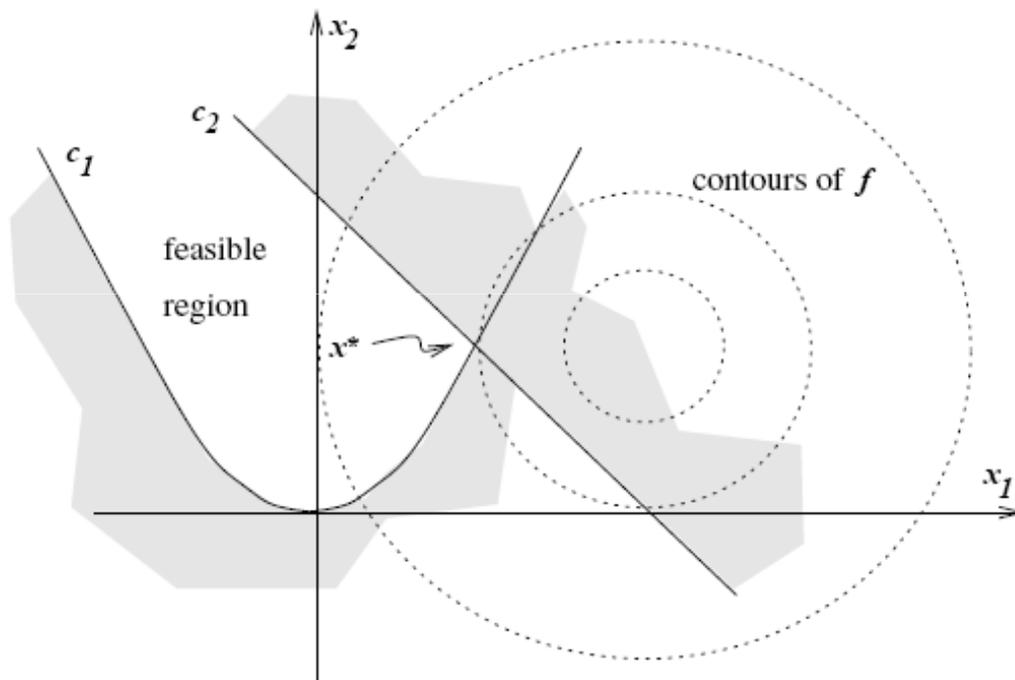
Different Algorithms for Different Optimization Problems

- Constraint vs. Unconstraint
- Type of constraints (equality vs. inequality)
- Global solution vs. Local Solution
- Stochastic vs. Deterministic
- Size of the problem (number of design variables)
- Type of design variables (real vs. integer, continuous vs. discrete)
- Linear vs. Nonlinear
- Discontinuous feasibility space



Constrained and Unconstrained Optimization

$$\min_{x \in \mathbb{R}^n} f(x) \text{ subject to } \begin{cases} c_i(x) = 0, i \in E \\ c_i(x) \geq 0, i \in I \end{cases}$$



$$\min_{x \in \mathbb{R}^2} (x_1 - 2)^2 + (x_2 - 1)^2 \text{ subject to } \begin{cases} x_1^2 - x_2 \leq 0 \\ x_1 + x_2 \leq 2 \end{cases}$$



Numerical Methods

Unconstrained	Constrained
Line Search	Simplex Method (linear programming)
Trust Region	Interior-Point Methods (linear program.)
Conjugate Gradient	SLP
Newton Methods	SQP
Quasi-Newton	Penalty Method
LFOP	Barrier Method
	Augmented Lagrangian Methods
	LFOPC



Linear vs. Nonlinear

The objective function is a linear function of the design variables if each design variable appears only to the first power with constant coefficients multiplying it.

$f(x) = x_1 + 2x_2 - 3.5x_3$ is linear in $x = [x_1, x_2, x_3]^T$

$f(x) = x_1x_2 + 2x_2 - 3.5x_3$ is nonlinear in x

$f(x) = \cos(x_1) + 2x_2 - 3.5x_3$ is nonlinear in x



Linear vs. Nonlinear

A constraint is a linear function of the design variables if each design variable appears only to the first power with constant coefficients multiplying it.

$$\begin{array}{ll} x_1 - 2x_2 + 5x_3 < 20 & \text{is linear in } x \\ x_1 - 2x_2 + 5x_3^2 < 20 & \text{is linear in } x \\ x_1 - 2\sin(x_2) + 5x_3 < 20 & \text{is linear in } x \end{array}$$

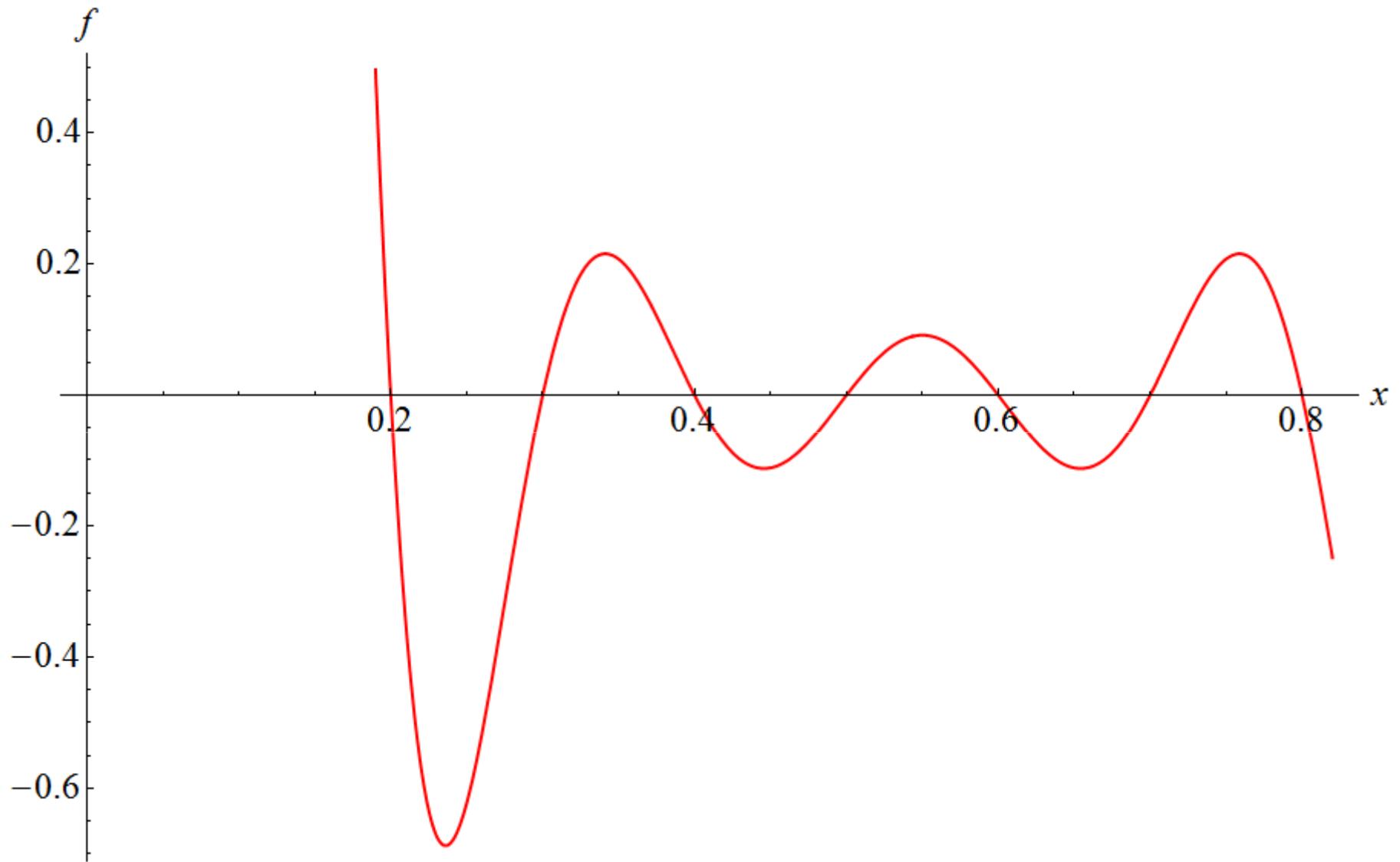


Global and Local Optimization

- The fast optimization algorithms seek only a local solution (objective function is smaller than all other feasible points in its vicinity)
- The local solution is not always the best of all such minima, the global solution
- Usually it is hard to say whether global solution exists or not. Even harder to find the global solution.
- There is certain class of problems for which all local solutions are also global solutions - convex programming
- Linear programming problems are convex
- Many global optimization algorithms proceed by solving a sequence of local optimization problems



Global minimizer is hard to find



Iterative Optimization Procedures

Most optimization algorithms are of an iterative nature

Starting with initial guess x_0

$$x_i = x_{i-1} + \alpha_i p_i$$

i – iteration number

p – search direction vector

α_i - search distance

The optimization algorithm determines the direction p and the distance α_i .

Gradient based algorithms use the gradient to calculate the direction.



Gradient

Given function $f(x)$, where x is a vector, $x = [x_1, x_2, \dots, x_n]$

The gradient at point y is a vector

$$\nabla f(y) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(y) \\ \frac{\partial f}{\partial x_2}(y) \\ \vdots \\ \frac{\partial f}{\partial x_n}(y) \end{pmatrix}$$



Hessian Matrix

Given function $f(x)$, where x is a vector, $x = [x_1, x_2, \dots, x_n]$

The second derivate of $f(x)$ at point y is a matrix of size $n \times n$

$$H(y) = \nabla^2 f(y) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & & & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

Each element of the matrix is evaluated at point y



Calculate Hessian and Gradient

$$f(x) = 2x_2 + x_1 x_2 + 3x_3^2 + x_2^2 x_3$$



Taylor Series

When the variable is a scalar

$$f(x + h) = f(x) + f'(x)h + \frac{1}{2} f''(x)h^2 + \dots$$

When variable is a vector

$$f(x + h) = f(x) + \underbrace{\nabla f(x)^T}_{1 \times n} \underbrace{h}_{n \times 1} + \frac{1}{2} \underbrace{h^T}_{1 \times n} \underbrace{H(x)}_{n \times n} \underbrace{h}_{n \times 1} + \dots$$



Finite difference approximation

$$x = (x_1, \dots, x_n)$$

δ = some small number (i.e. 10^{-6})

Approximate derivative with respect to x_i

consider $\delta x = (x_1, \dots, x_i + \delta, \dots, x_n)$

$$\frac{\partial f}{\partial x_i} \approx \frac{f(\delta x) - f(x)}{\delta}$$

Approximation for second derivative

consider $\delta x^- = (x_1, \dots, x_i - \delta, \dots, x_n)$

$$\frac{\partial^2 f}{\partial x_i^2} \approx \frac{f(\delta x) - 2f(x) + f(\delta x^-)}{\delta^2}$$

What is a Solution?

- A point x^* is a **global** minimizer if

$$f(x^*) \leq f(x) \text{ for all } x$$

- A point x^* is a **local** minimizer if

$$f(x^*) \leq f(x) \text{ for all } x \text{ in some neighborhood of } x$$

- A point x^* is a **strict local** minimizer if

$$f(x^*) < f(x) \text{ for all } x \text{ in some neighborhood of } x$$



Recognizing a Local Minima (Taylor's theorem)

- If f is continuously differentiable then

$$f(x + p) = f(x) + \nabla f(x + tp)^T p, \text{ for some } t \in (0, 1)$$

- If f is twice continuously differentiable then

$$f(x + p) = f(x) + \nabla f(x)^T p + \frac{1}{2} p^T \nabla^2 f(x + tp)^T p, \text{ for some } t \in (0, 1)$$

- If x^* is a local minimizer then $\nabla f(x^*)=0$. If $\nabla f(x^*)=0$ then x^* called a **stationary point**
- If x^* is a local minimizer then $\nabla f(x^*)=0$ and $\nabla^2 f(x^*)>0$ is positive definite.

Sufficient conditions

$\nabla^2 f(x)$ is called Hessian and sometimes denoted by $H(x)$



Positive Definite Matrices

- Positive definiteness: $y^T H y > 0$ for all non-zero y
- Two ways to identify positive definiteness

1) Consider eigenvalues of H : $H v_i = \lambda_i v_i$

If H is non-singular symmetric then eigenvectors form an orthogonal basis, i.e any vector y can be represented as a linear combination of eigenvectors

$$y = \sum_i a_i v_i$$

Then

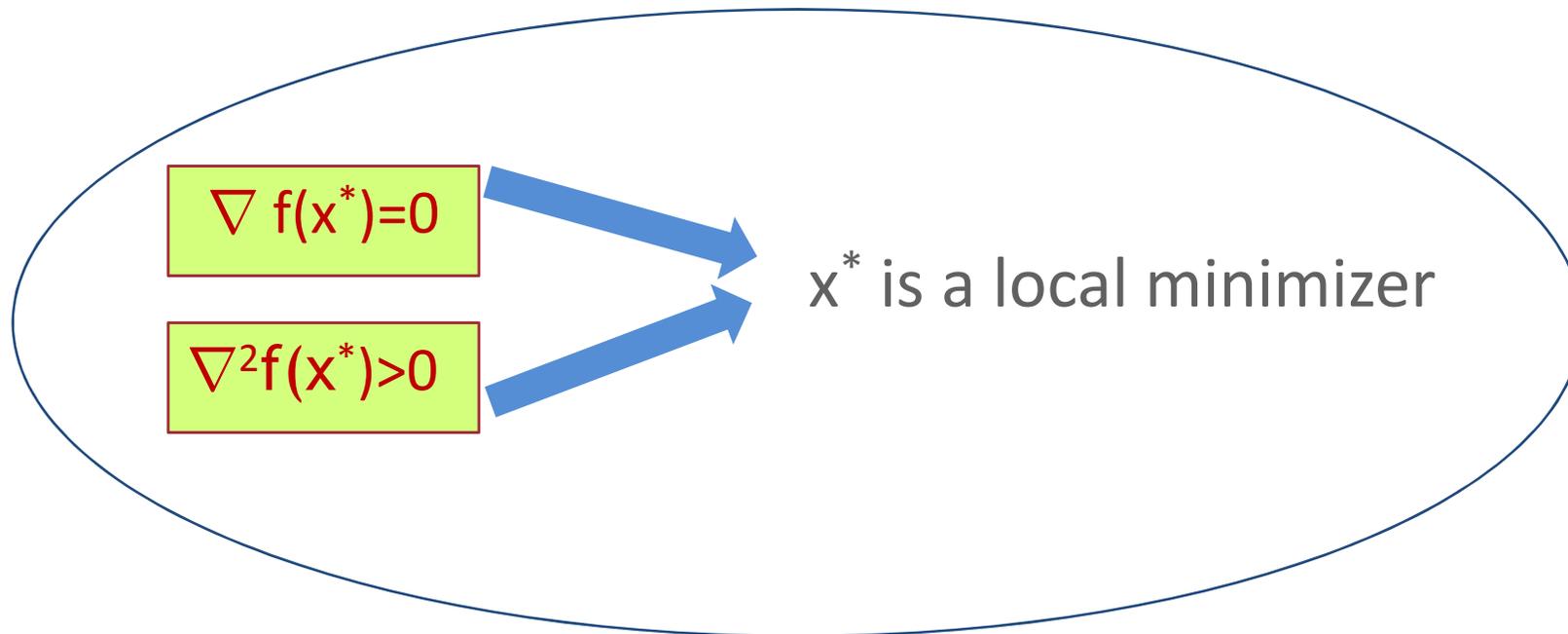
$$y^T H y = \sum_i a_i v_i^T H \sum_j a_j v_j = \sum_{i,j} a_i v_i^T \lambda_j a_j v_j = \sum_i a_i^2 \lambda_i$$

Therefore all of the eigenvalues of H are positive

2) Apply Cholesky decomposition algorithm to H and in case it doesn't break down matrix H is SPD.

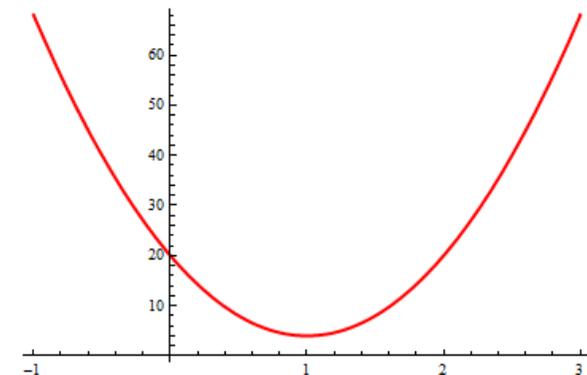


Sufficient Conditions for Unconstrained Problem



1. Gradient vanishes
2. Hessian is positive definite

Note when $\nabla^2 f(x)>0$ for all values of x , then local minima is also a **global** minima!



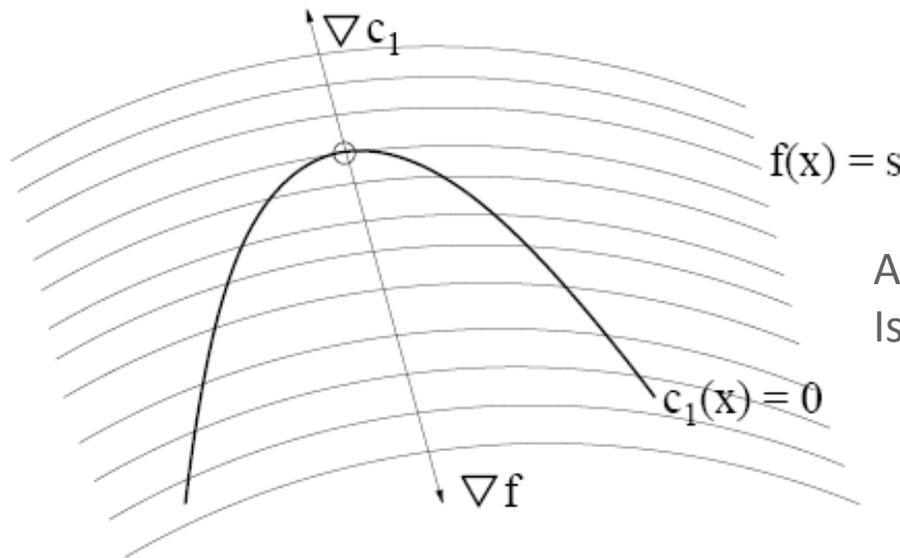
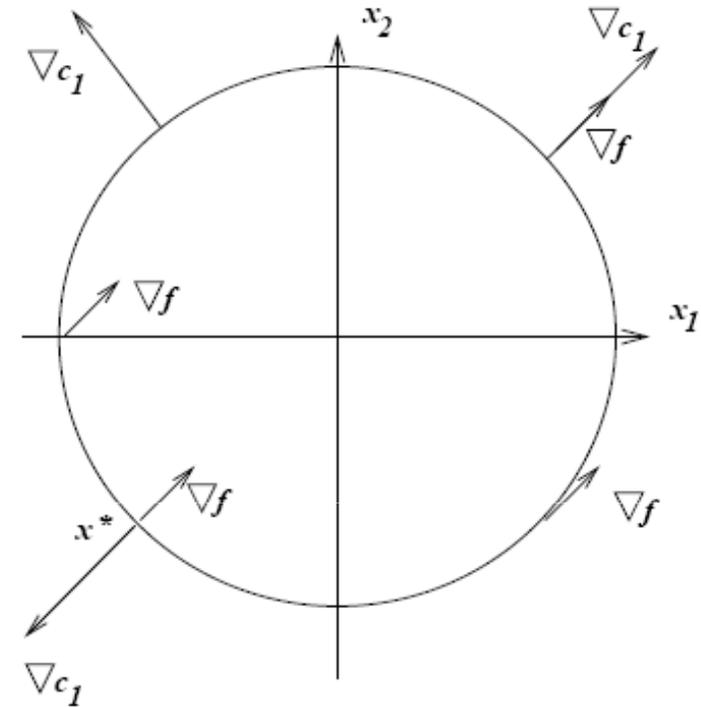
Existence and uniqueness for equality constrained problems

$$\min x_1 + x_2$$

$$\text{s.t. } x_1^2 + x_2^2 - 2 = 0$$

feasibility set = circle of radius $\sqrt{2}$

solution is $x = (-1, -1)$



At the solution the constraint normal $\nabla c_1(x^*)$ is parallel to $\nabla f(x^*)$. That is for some scalar α

$$\nabla f(x^*) = \lambda_1^* \nabla c_1(x^*)$$

Lagrangian Function

$$\mathcal{L}(x, \lambda_1) = f(x) - \lambda_1 c_1(x)$$

$$\nabla_x \mathcal{L}(x, \lambda_1) = \nabla f(x) - \lambda_1 \nabla c_1(x)$$

$$\nabla_x \mathcal{L}(x^*, \lambda_1^*) = 0$$

Necessary but not sufficient



Existence and Uniqueness for Inequality Constrained Problems

$$\begin{aligned} & \min x_1 + x_2 \\ \text{s.t. } & x_1^2 + x_2^2 - 2 \leq 0 \end{aligned}$$

Sign of the Lagrange multiplier is curtail!

$$\nabla_x \mathcal{L}(x^*, \lambda_1^*) = 0, \text{ for some } \lambda_1^* \geq 0$$

We also require that

$$\lambda_2^* c_1(x^*) = 0$$



First Order Optimality. Karush-Kuhn-Tucker (KKT) Conditions

Suppose that x^* is a local solution, then for some vector λ^*

$$\begin{aligned}\nabla_x \mathcal{L}(x^*, \lambda_1^*) &= 0 \\ c_i(x^*) &= 0, \quad \text{for all } i \in E \\ c_i(x^*) &\geq 0, \quad \text{for all } i \in I \\ \lambda_i^* &\geq 0, \quad \text{for all } i \in I \\ \lambda_i^* c_i(x^*) &\geq 0, \quad \text{for all } i \in I \cup E\end{aligned}$$



KKT: Interpretation

1. Constraints are satisfied
2. If constraint is not precisely satisfied then the corresponding Lagrange multiplier is zero
3. The gradient of the Lagrangian vanishes at the local solution

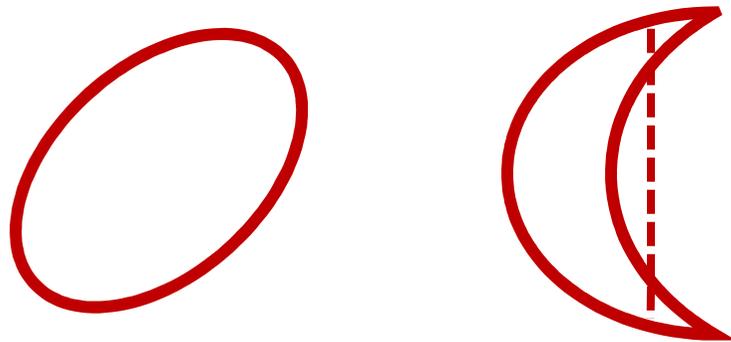


Convex Sets

Consider a set, and imagine drawing a line connecting any two points in the set.

If every point along that line is inside the set, then the set is **convex**.

If **any point along that line is outside the set**, then the set is **non-convex**.

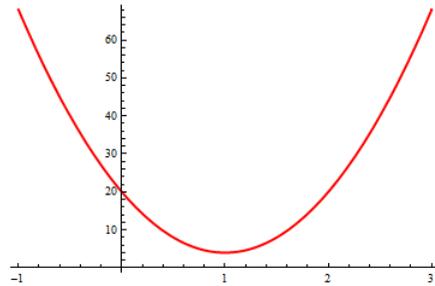


The line connecting points x_1 and x_2 is given by

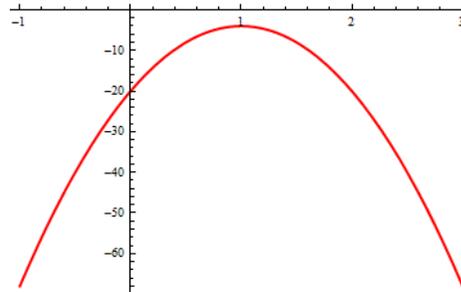
$$w = \alpha x_1 + (1 - \alpha)x_2, \quad 0 \leq \alpha \leq 1$$



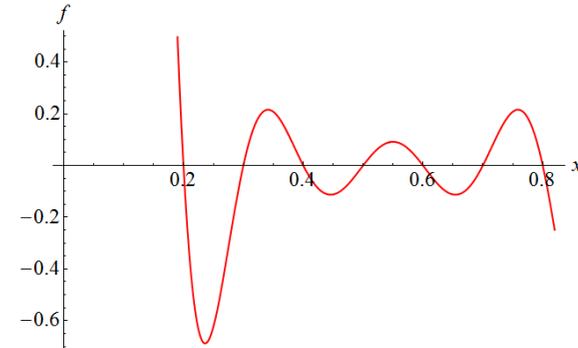
Convex Functions



convex



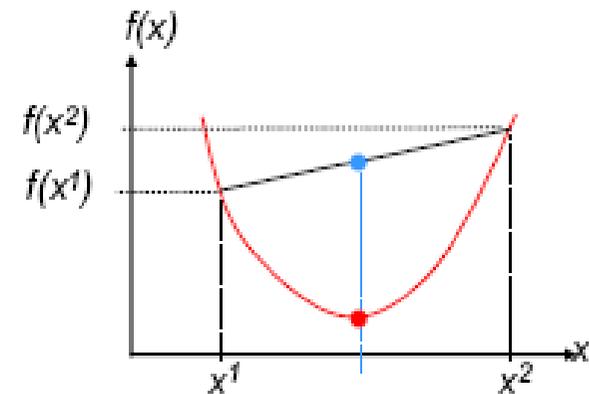
concave



neither

A function $f(x)$ is convex if

$$f(\alpha x_1 + (1 - \alpha)x_2) \leq \alpha f(x_1) + (1 - \alpha)f(x_2)$$



Convex Spaces

Pick any two points in the feasible region. If all points on the line connecting these points lie in the feasible region, then the constraint surfaces are convex.

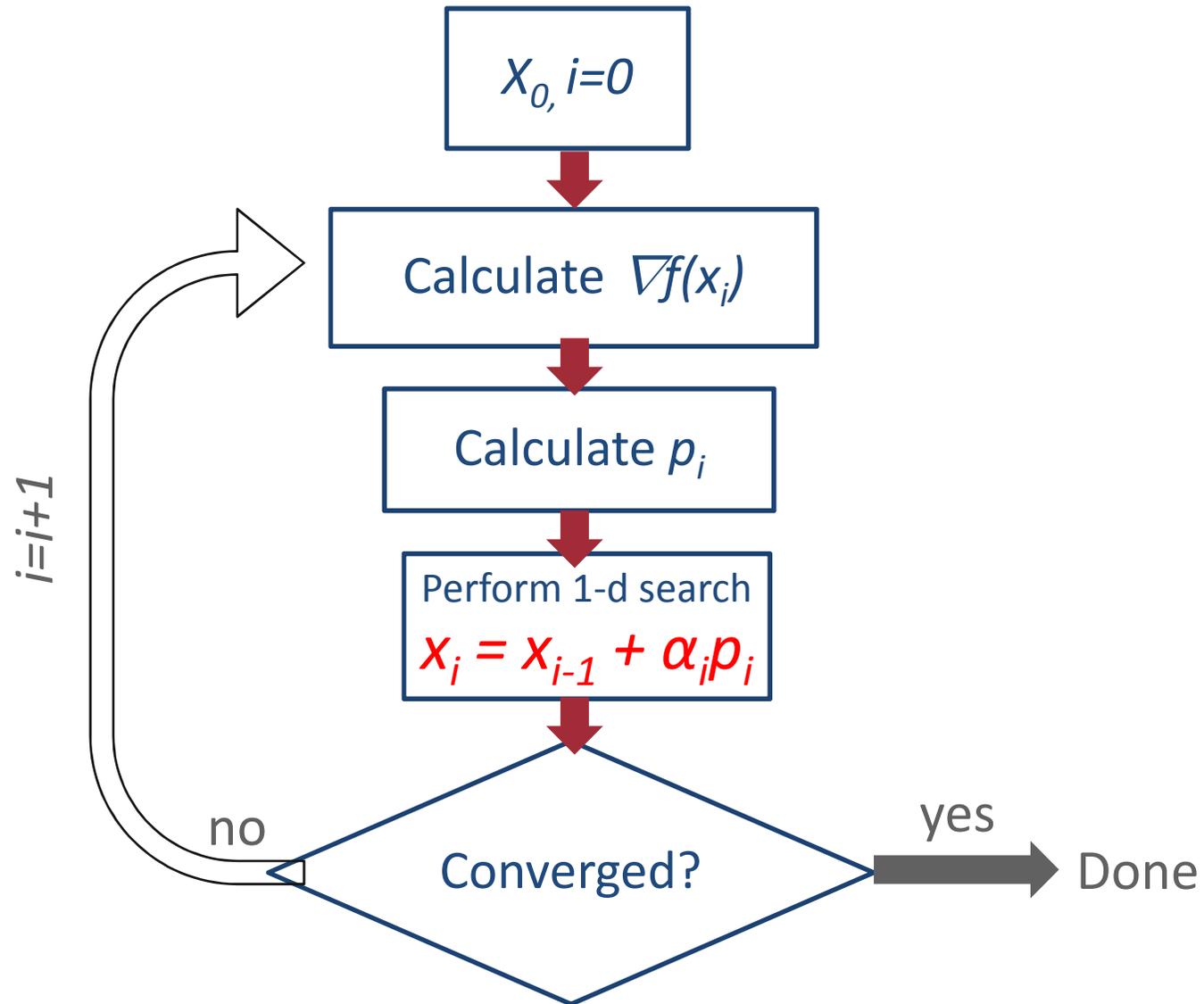
If the objective function is convex, then it has only one optimum (the global one) and the Hessian matrix is positive definite for **all possible designs**.

If the objective function and all constraint surfaces are convex, then the design space is convex, and the Kuhn-Tucker conditions are sufficient to guarantee that **x^* is a global optimum**.

In general, for engineering problems, the design space is not convex ...



Optimization Process (Line Search) 1/2



Optimization Process (Line Search) 2/2

p_i – search direction

usually the search direction has the form

$$p_i = -B_i^{-1} f_i$$



Unconstrained Problems: Solution Methods

- First-Order Methods
 - use gradient information to calculate p
 - steepest descent method
 - conjugate gradient method
 - quasi-Newton methods
- Second-Order Methods
 - use gradients and Hessian to calculate p
 - Newton method
- Often, a constrained problem can be cast as an unconstrained problems and these techniques used.



One-Dimensional Search (Choosing α)

- Polynomial interpolation
 - pick several values for α
 - fit polynomials to $f(\alpha)$
 - efficient, but need to be careful with implementation
- Golden section search
 - easy to implement, but inefficient
- The one-dimensional search is one of the more challenging aspects of implementing a gradient-based optimization algorithm



Steepest Descent

$$p_i = - \nabla f(x_{i-1})$$

Algorithm:

choose x_0 , $x = x_0$

Repeat until converges:

$$p_i = - \nabla f(x)$$

choose α to minimize $f(x + \alpha p)$

$$x = x + \alpha p_i$$

- *Uses only information for the previous step*
- *Slow convergence*



Conjugate Gradient

$$p_1 = -\nabla f(x_0)$$

$$p_i = -\nabla f(x_{i-1}) + \beta_i p_{i-1}$$

$$\beta_i = \frac{|\nabla J(x_{i-1})|^2}{|\nabla J(x_{i-1})|^2}$$

- Search directions are conjugate ($p_i^T H p_K = 0$, sometimes called H-orthogonal)
- Makes use of information from previous iterations



Newton's Method (1/2)

Taylor Series:

$$f(x + h) = f(x) + \nabla f(x)^T h + \frac{1}{2} h^T H(x) h + \dots$$

Approximate $\nabla f(x+h) \approx \nabla f(x) + H(x)h$

At optimum $\nabla f(x^*) = 0$

$$\Rightarrow \nabla f(x) + H(x)h = 0$$

$$h = -H(x)^{-1} \nabla f(x)$$



Newton's Method (2/2)

$$p_i = -H(x_{i-1})^{-1} \nabla f(x_{i-1})$$

- if $f(x)$ is quadratic, method gives exact solution in one iteration
- if $f(x)$ not quadratic, perform Taylor series about new point and repeat until converged
- a very efficient technique if started near the solution
- H is not usually available analytically, and finite difference is too expensive ($n \times n$ matrix)
- H can be singular if f is linear in a design variable



Quasi Newton

$$p_i = -A_i \nabla f(x_{i-1})$$

- Also known as variable metric methods
- Objective and gradient information is used to create an approximation to the inverse of the Hessian
- A approaches H^{-1} during optimization of quadratic functions
- Convergence is similar to second-order methods (strictly 1st order)
- Initially: $A=I$, so p_1 is steepest descent direction

then: $A_{i+1} = A_i + D_i$

where D is a symmetric update matrix

$$D_i = \text{fn}(x_i - x_{i-1}, \nabla f(x_i) - \nabla f(x_{i-1}), A_i)$$

- Various methods to determine D
e.g. Davidon-Fletcher-Powell (DFP)
Broydon-Fletcher-Goldfarb-Shanno (BFGS)



Constrained Problems: Solution Methods

- Sequential Linear Programming
- Penalty and Barrier Methods
- Sequential Quadratic Programming
- Mixed Integer Programming

$$\min_{x \in R^n} f(x) \text{ subject to } \begin{cases} c_i(x) = 0, i \in E \\ c_i(x) \geq 0, i \in I \end{cases}$$



Constrained Optimization: Vocabulary

- Feasible design: a design that satisfies all constraints
- Infeasible design: a design that violates one or more constraints
- Optimum design: the choice of design variables that minimizes the objective function while satisfying all constraints

In general, constrained optimization algorithms try to cast the problem as an unconstrained optimization and then use one of the techniques we looked at.



Linear Programming (1/2)

Most engineering problems of interest are nonlinear

- Can often simplify nonlinear problem by linearization
- LP is often the basis for developing more complicated NLP algorithms

Standard LP problem:

$$\begin{aligned} \min_x f(x) &= \sum_{i=1}^n c_i x_i \\ \text{s.t.} \quad \sum_{i=1}^n a_{ij} x_i &= b_j, \quad j = 1, \dots, m \\ x_i &\geq 0, \quad i = 1, \dots, n \end{aligned}$$



$$\begin{aligned} \min_x f(x) &= c^T x \\ \text{s.t.} \quad Ax &= b \\ x_i &\geq 0, \quad i = 1, \dots, n \end{aligned}$$



Linear Programming (2/2)

To convert inequality constraints to equality constraints, use additional design variables:

$$\sum_{i=1}^n a_{ij}x_i \leq b_j \longrightarrow \sum_{i=1}^n a_{ij}x_i + x_{n+1} = b_j$$

Where x_{n+1} is non negative

x_{n+1} is called a **slack variable**



Sequential Linear Programming (1/3)

Consider a general nonlinear problem linearized via first order Taylor series:

$$\begin{aligned} \min_x \quad & f(x) \approx f(x_0) + \nabla f(x_0)\delta x \\ \text{s.t} \quad & c_i(x) \approx c_i(x_0) + \nabla c_i(x_0)\delta x \geq 0, \quad i \in E \\ & c_i(x) \approx c_i(x_0) + \nabla c_i(x_0)\delta x = 0, \quad i \in I \end{aligned}$$

where $\delta x = x - x_0$

This is an LP problem with the design variables contained in δx . The functions and gradients evaluated at x_0 are constant coefficients.



Sequential Linear Programming (2/3)

1. Initial guess x_0
2. Linearize about x_0 using first-order Taylor series
3. Solve resulting LP to find δx
4. Update: $x_1 = x_0 + \delta x$
5. Linearize about x_1 and repeat:

$$x_i = x_{i-1} + \delta x$$

where δx is the solution of an LP (model linearized about x_{i-1}).



Sequential Linear Programming (3/3)

- Linearization approximation is only valid close to x_0
- Need to restrict size of update δx
- Not considered to be a good method



Sequential Quadratic Programming (1/4)

- Create a quadratic approximation to the Lagrangian
- Create linear approximations to the constraints
- Solve the quadratic problem to find the search direction, p
- Perform the 1-D search
- Update the approximation to the Lagrangian



Sequential Quadratic Programming (2/4)

Create a sub problem with quadratic objective function and linear constraints

$$\begin{aligned} \min_p \quad & Q(p_i) = f(x_i) + \nabla f(x_i)^T p_i + \frac{1}{2} p^T W_i p \\ \text{s.t.} \quad & \nabla c_j(x_i)^T p_i + c_j(x_i) \geq 0, \quad j \in I \\ & \nabla c_j(x_i)^T p_i + c_j(x_i) = 0, \quad j \in E \end{aligned}$$

$W=I$ for $i=0$ and then W is an approximation of the Hessian



Sequential Quadratic Programming (3/4)

- The constraints of the subproblem can be incompatible, even if the original problem has a well-posed solution
- For example, two linearized constraints could be linearly dependent
- This is a common occurrence in practice
- Likelihood of incompatible constraints reduced by allowing flexibility in RHS

$$\nabla c_j(x_i)^T p_i + \gamma_i c_i(x_i) = 0, \quad j \in E$$

- Typically $\gamma=0.9$ if constraint is violated and $\gamma=1$ otherwise
- Doesn't affect convergence, since specific form of constraint is only crucial when x_i is close to x^*



Sequential Quadratic Programming (4/4)

- Widely used in engineering applications
- Considered to be the best gradient-based algorithm
- Strong theoretical basis



Subproblems

- Many optimization algorithms get to the optimum by generating and solving a sequence of unconstrained subproblems.
- One fundamental approach to constrained optimization is to replace the original problem by a penalty function that consists of
 - the original objective of the constrained optimization problem, *plus*
 - one additional term for each constraint, which is positive when the current point x violates that constraint and zero otherwise.
- Typically there are two tasks at each iteration:
 - Calculate search direction
 - Calculate the step length
- Sometimes, the initial formulation of a subproblem may be defective *i.e. the subproblem has no solution or the solution is unbounded*
- A valid subproblem is one for which a solution exists and is well defined
- The option to abandon should be available if the subproblem appears defective
- It is possible that a defective subproblem is an accurate reflection of the original problem having no valid solution



Penalty and Barrier Methods

General Approach:

1. minimize objective as unconstrained function
2. provide penalty to limit constraint violations
3. magnitude of penalty varies throughout optimization
4. called sequential unconstrained minimization techniques (SUMT)
5. create pseudo-objective:

$$\Phi(x, r_p) = f(x) + r_p P(x)$$

$f(x)$ = original objective function

$P(x)$ = imposed penalty function

r_p = scalar multiplier to determine penalty magnitude

p = unconstrained minimization number



Mixed Integer Programming

- Rounding
 - unlikely to achieve optimum discrete solution
 - rounded solution may be infeasible
- Branch and Bound
 - create tree with several optimization branches
 - continuous solution is a lower bound on the mixed solution
 - usually solve a large number of optimization problems
 - expensive



Leapfrog Optimizer for Constrained minimization (LFOPC)

- First order method (uses gradient $\nabla f(x)$)
- No explicit line search performed compared to SQP
- Is robust and handles steep valleys and discontinuous functions and gradients
- The algorithm seeks low local minimum and can be used as a basic component in a methodology for global optimization
- The method is not as efficient as classical methods on smooth and near-quadratic functions



LFOP: Basic algorithms for unconstrained problem (1/2)

- Assume a particle of unit mass in n-dimensional conservative force field
- Potential energy at x given by $f(x)$
- Force at x is $-\nabla f(x)$
- Equation for motion of the particle (Second Newton's Law)

$$a = \ddot{x} = -\nabla f(x)$$

- The motion of the particle over time period $[0,t]$:

$$\frac{1}{2}\|\dot{x}(t)\|^2 - \frac{1}{2}\|\dot{x}(0)\|^2 = f(x(0)) - f(x(t))$$

or

$$T(t) - T(0) = f(0) - f(t)$$

$\frac{1}{2}\|\dot{x}(t)\|^2$ - kinetic energy. Conservation of energy:

$$\frac{1}{2}\|\dot{x}(t)\|^2 + f(x(t)) = \text{constant}$$

As kinetic energy increases, $f(x(t))$ decreases



LFOP: Basic algorithms for unconstrained problem (2/2)

- Compute the dynamic trajectory by solving the initial-value problem (IVP)

$$\ddot{x}(t) = -\nabla f(x(t))$$

$$\dot{x}(0) = 0, \quad x(0) = x_0$$

- Monitor $\dot{x}(t) = v(t)$. As long as $\|v(t)\|$ increases, $f(x)$ decreases
- When $\|v(t)\|$ decreases apply some interfering strategy to extract energy and therefore increase the likelihood of descent
- In practice the leap-frog scheme is used to integrate the IVP numerically with time step Δt

$$x_{i+1} = x_i + v_i \Delta t$$

$$v_{i+1} = v_i + a_{i+1} \Delta t$$

where $a_i = -\nabla f(x_i)$, $v_0 = \frac{1}{2} a_0 \Delta t$

- A typical interfering strategy is if $\|v_{i+1}\| > \|v_i\|$ continue, else:

$$v_i = \frac{v_{i+1} + v_i}{4}, \quad x_i = \frac{x_{i+1} + x_i}{2}$$

compute new v_{i+1} and continue



LFOPC: Modification for Constrained Problems

LFOP is applied to a penalty function

$$P(x, \mu) = f(x) + \mu \sum_{i \in E} c_i^2(x) + \sum_{i \in I} \beta_i c_i^2(x)$$

$$\text{where } b_i = \begin{cases} 0, & c_i(x) \leq 0 \\ \mu, & c_i(x) > 0 \end{cases}$$

μ is called **penalty parameter**

Phase 0: given x_0 , $\mu = \mu_0 = (10^2)$ apply FLOP to $f(x, \mu_0)$ to get $x^*(\mu_0)$

Phase 1: $x_0 = x^*(\mu_0)$, $\mu = \mu_1 = (10^4)$ apply FLOP to $f(x, \mu_1)$ to get $x^*(\mu_1)$ and identify active constraints

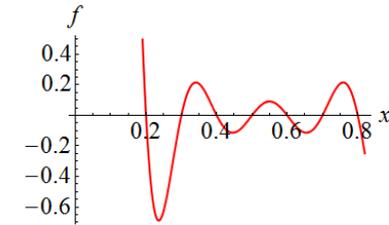
Phase 2: with $x_0 = x^*(\mu_1)$, $i \in A$, if $c_i(x^*(\mu_1)) > 0$

$$P_A(x, \mu_1) = f(x) + \mu \sum_{i \in E} c_i^2(x) + \sum_{i \in A} \beta_i c_i^2(x)$$

Notes: Default parameters for FLOPC are listed in section 20.6 Setting parameters in the LFOPC algorithm of the manual. Scaling is used in the numerical implementation of the algorithm



Heuristic Optimization Techniques



Main Motivation for Heuristic Techniques:

1. To deal with local optima and not get trapped in them
2. To allow optimization for systems, where the design variables are not only continuous, but discrete, integer or even Boolean

$$x = (1,2,3,4), x = ('a','b','c') x = (1,0,1,0,0,1)$$

These techniques do not guarantee that global optimum can be found. Generally Karush-Kuhn-Tucker conditions do not apply.



Algorithms

- Genetic Algorithms (Holland – 1975)
 - Inspired by genetics and natural selection
- Simulated Annealing (Kirkpatrick – 1983)
 - Inspired by molecular dynamics – energy minimization
- Particle Swarm Optimization (Eberhart and Kennedy - 1995)
 - Inspired by the social behavior of swarms of insects or flocks of birds

These techniques all use a combination of randomness and heuristic “rules” to guide the search for global maxima or minima



Basics

- Natural Selection is a very successful organizing principle for optimizing individuals and populations of individuals
- If we can mimic natural selection, then we will be able to optimize more successfully
- A possible design of a system – as represented by its design vector x - can be considered as an individual who is fighting for survival within a larger population.
- Only the fittest survive – Fitness is assessed via objective function f .



Natural Selection

Charles Darwin (1809-1882)

Extremely controversial and influential book (1859) *On the origin of species by means of natural selection, or the preservation of favored races in the struggle for life*

Observations:

- Species are continually developing
- Homo sapiens comes from ape-like stock
- Variations between species are enormous
- Huge potential for production of offspring, but only a small percentage survives to adulthood

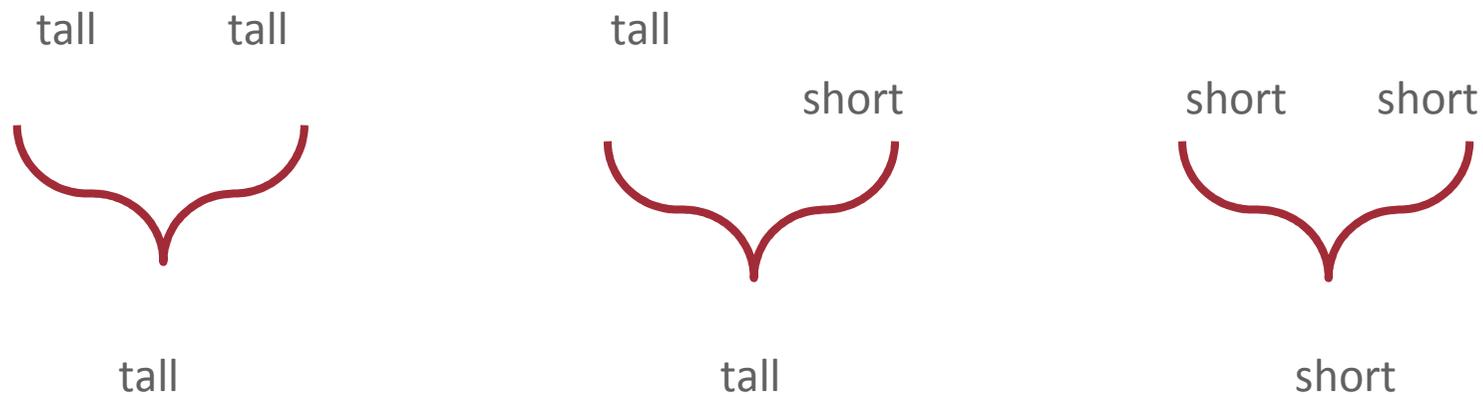
Evolution = natural selection of inheritable



Inheritance of Characteristics

Gregor Mendel (1822-1884)

Investigated the inheritance of characteristics (“traits”) Conducted extensive experiments with pea plants Examined hybrids from different strains of plant

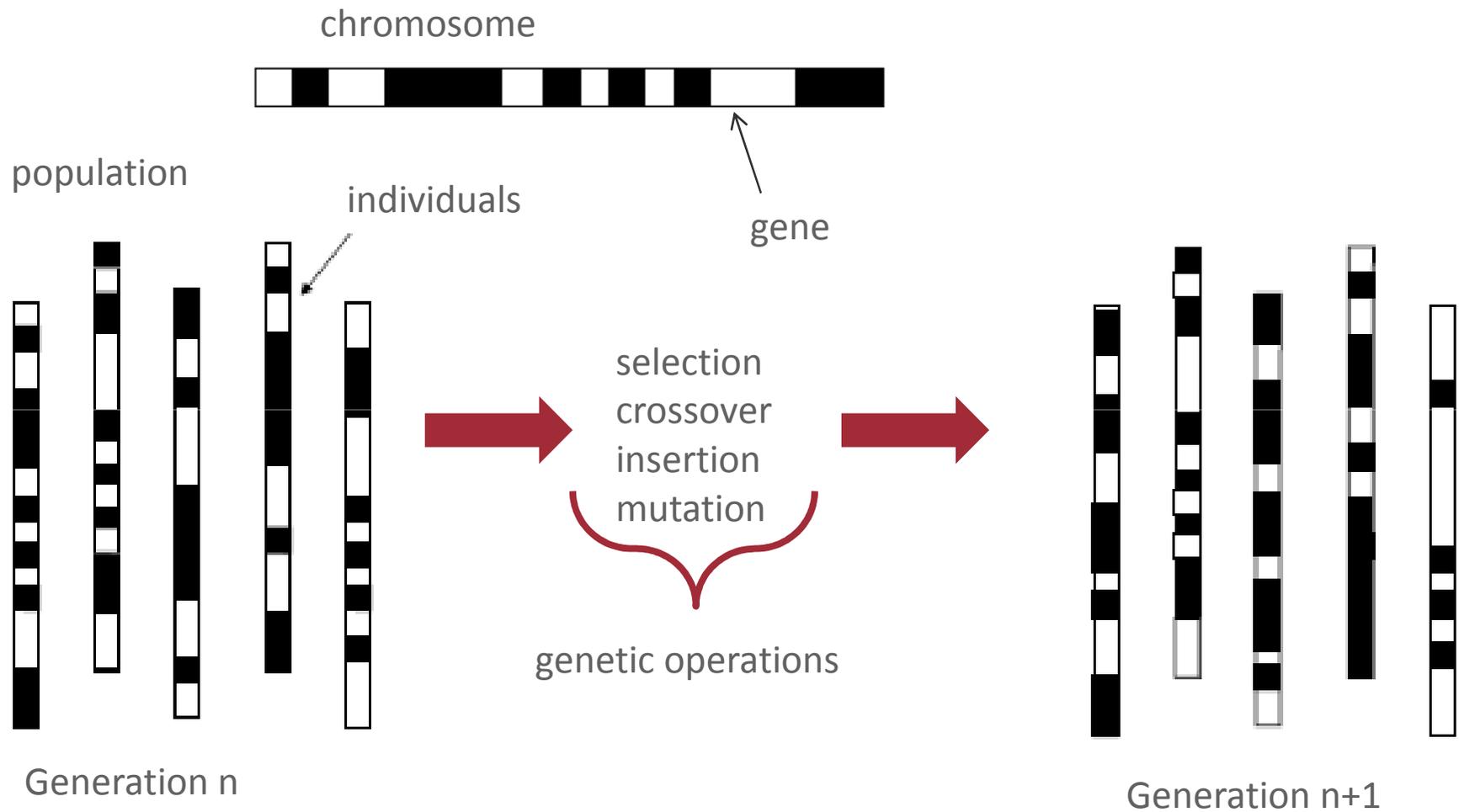


Tall – dominant gene

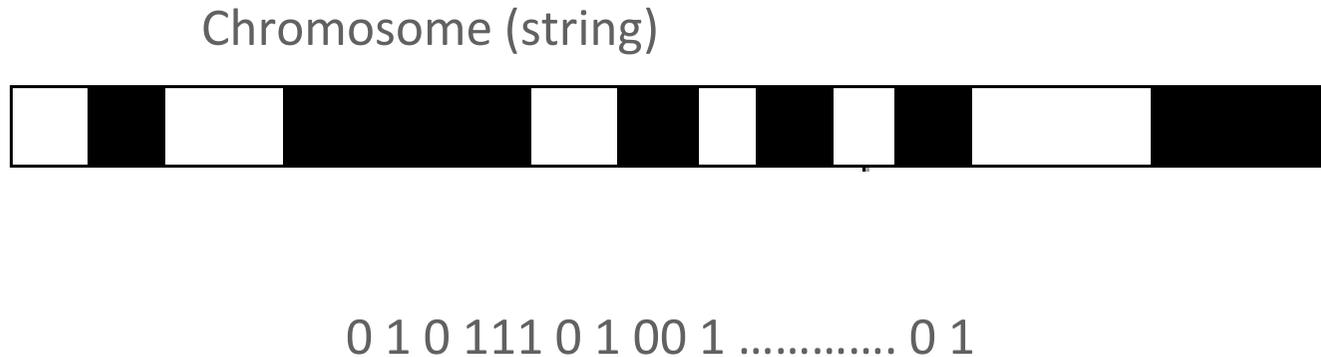
Short – recessive gene



GA Terminology



Chromosomes



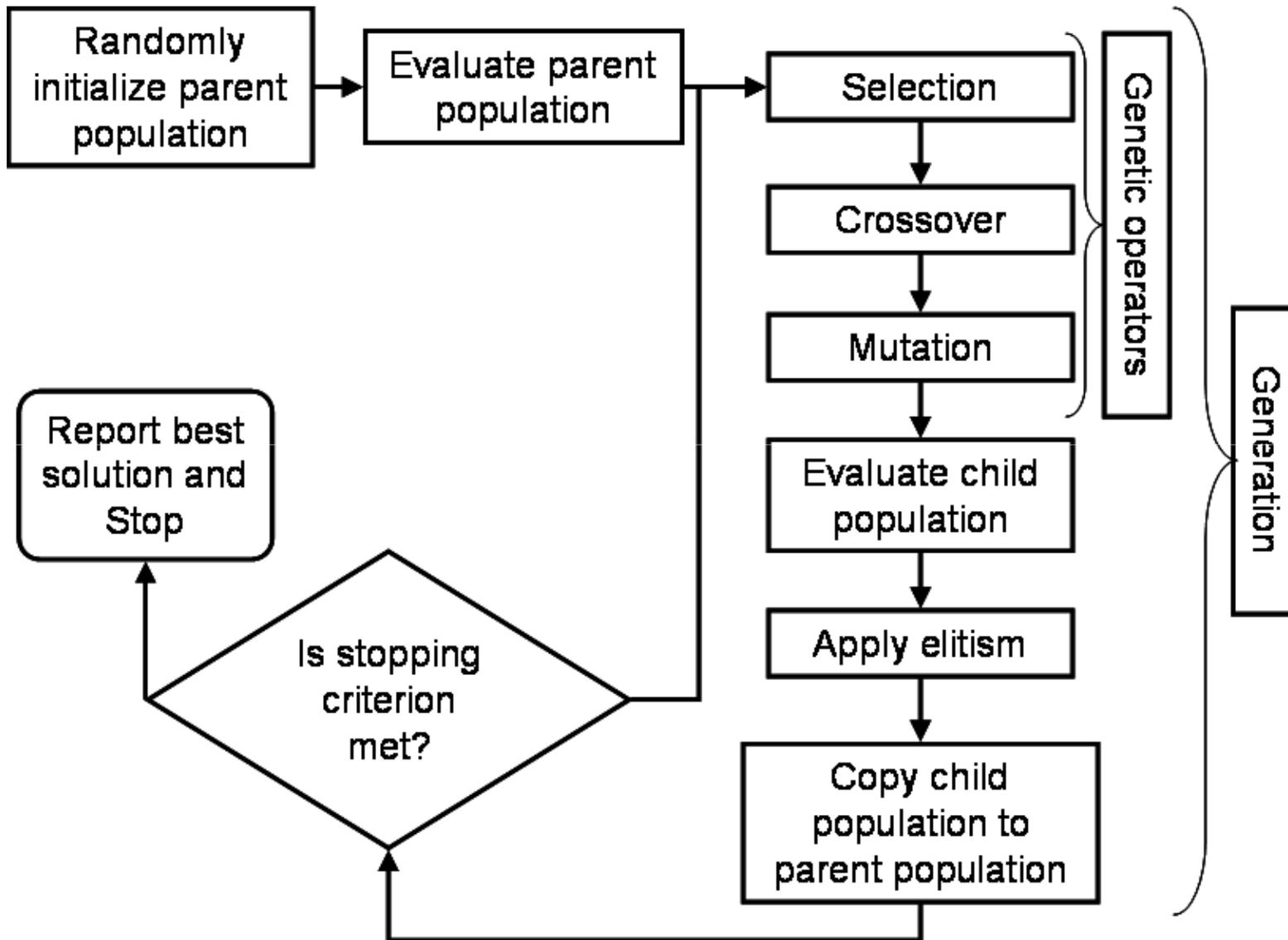
Each chromosome represents a solution, often using strings of 0's and 1's.
Each bit typically corresponds to a gene. This is called binary encoding.

The values for a given gene are the alleles.

A chromosome in isolation is meaningless need decoding of the chromosome into phenotypic values



GA Over Several Generation



Selection (1/4)

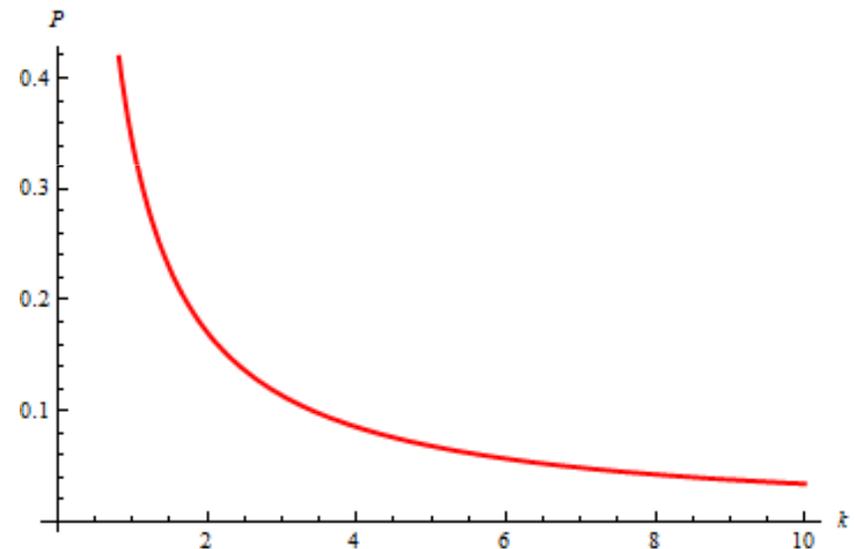
- Goal is to select parents for crossover
- Should create a bias towards more fitness
- Must preserve diversity in the population

(1) Selection according to RANKING

Example: Let $D = \sum_{j \in P} (1/j)$

select the k th most fit member of a Population to be a parent with

probability $P_k = (1/k)D^{-1}$



Better ranking has a higher probability of being chosen



Selection (2/4)

- Proportional to FITNESS Value Scheme
- Let $F = \sum_{j \in P} \text{Fitness}(j)$

Select kth most fit member of a population to be a parent with probability

$$P_k = \text{Fitness}(k)F^{-1}$$

Probability of being selected for crossover is directly proportional to raw fitness score.

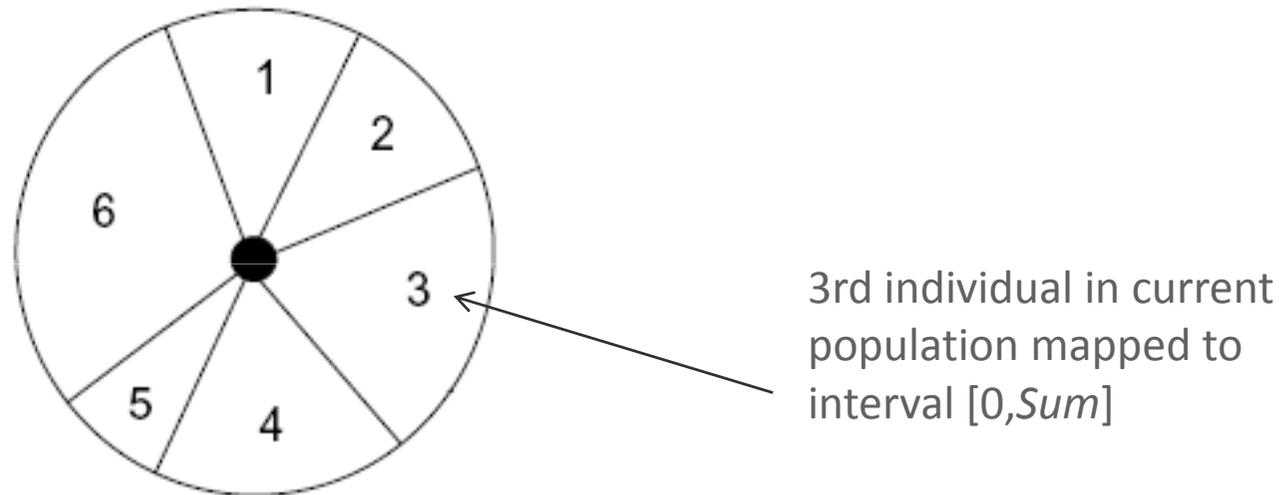
This scheme tends to favor the fittest individuals in a population more than the ranking-scheme, faster convergence, but can also be a disadvantage.



Selection (3/4) - Roulette Wheel Selection

Probabilistically select individuals based on some measure of their performance.

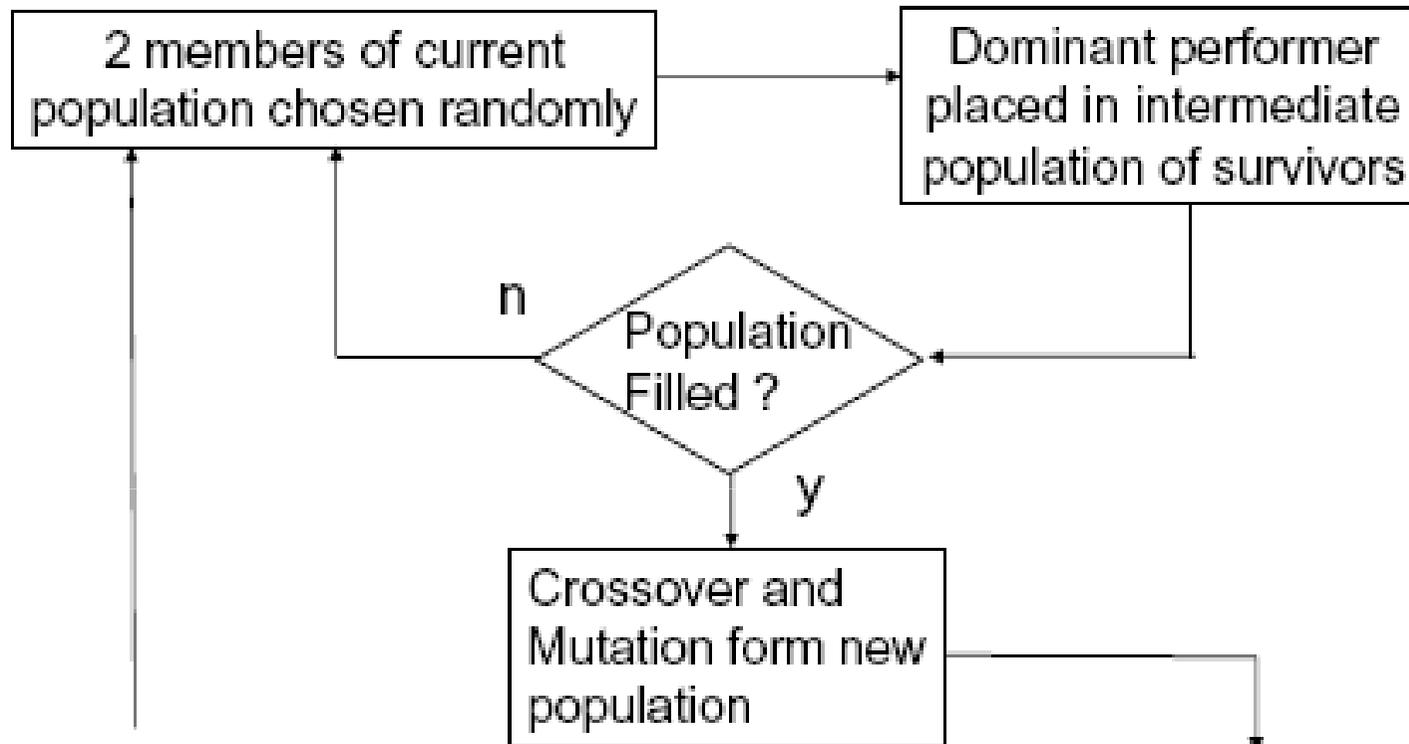
Sum: Sum of individual's selection probabilities



Selection: generate random number in $[0,Sum]$ Repeat process until desired # of individuals selected Basically: stochastic sampling with replacement (SSR)



Selection (4/4) - Tournament Selection

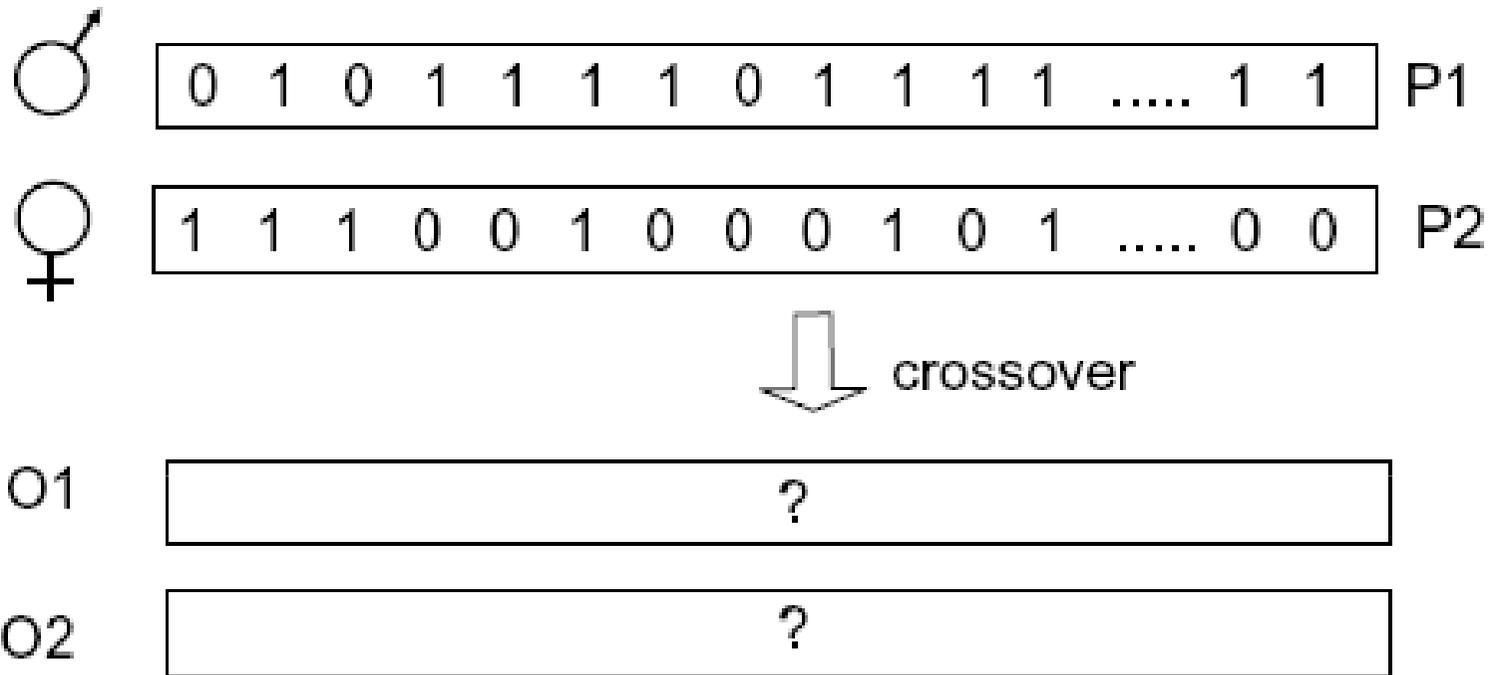


Old Population	Fitness
101010110111	8
100100001100	4
001000111110	6

Survivors	Fitness
101010110111	8
001000111110	6
101010110111	8



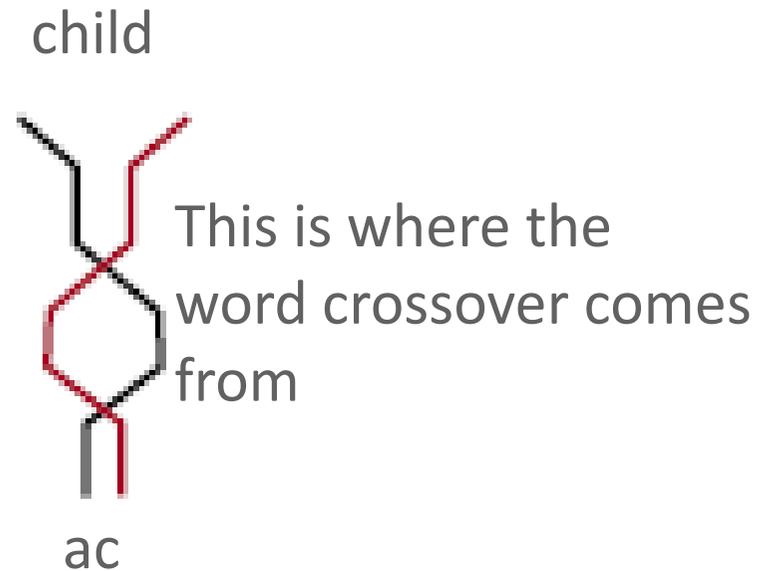
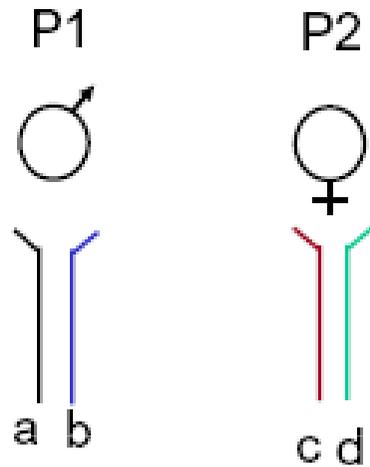
Crossover



Question: How can we operate on parents P1 and P2 to create offspring O1 and O2 (same length, only 1's and 0's)?



Crossover in Biology



Crossover produces either of these results for each chromosome

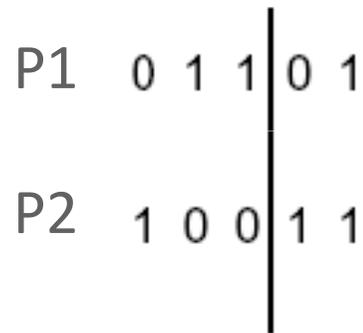
ac OR ad OR bc OR bd



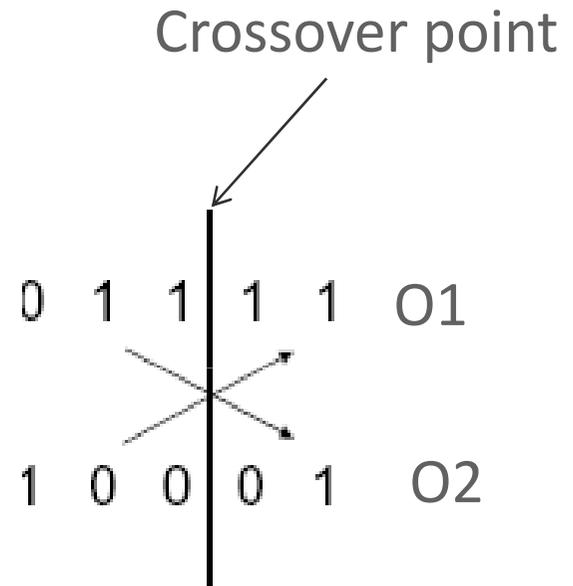
Crossover Operator (1/3)

Crossover (mating) is taking 2 solutions, and creating 1 or 2 more

Classical: single point crossover



The parents

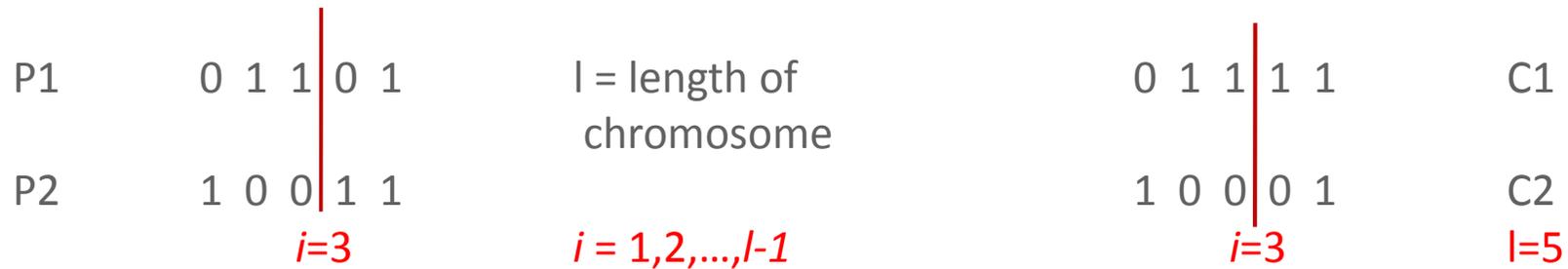


The children
("offspring")



Crossover Operator (2/3)

More on 1-point crossover



A crossover bit “ i ” is chosen (deliberately or randomly), splitting the chromosomes in half.

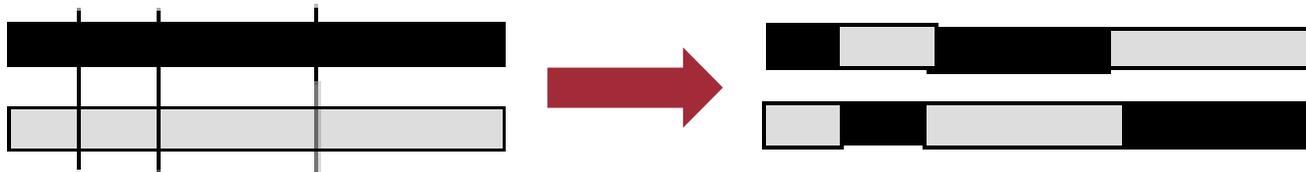
Child C1 is the 1st half of P1 and the 2nd half of P2

Child C2 is the 1st half of P2 and the 2nd half of P1



Crossover Operator (3/3)

2-point crossover or a multi-point crossover



The essential aspect is to create at least one child (solution/design) from two (or more) parent (solutions/designs)

- there are many solutions to do this
- do not necessarily have to do crossover, and do crossover with a probability P_x after pairs are chosen

Some crossover operations:

- single point, versus multiple point crossover
- path relinking
- permutation operators (list operators), incl. Random keys approach



Stopping Criteria

- No improvement in the last few generations
- Fixed number of generations or function evaluations (A user defined number of generations is used as the stopping criterion in LS-OPT)
- The best individual (among all searched individuals) is reported as the optimal solution
- Hope is that the reported best individual would represent the global optimal solution.



Elitism

- The best chromosome (or a few best chromosomes) is copied to the population in the next generation.
- The rest are chosen in classical way.
- Elitism can very rapidly increase performance of GA, because it prevents losing the best found solution to date.
- A variation is to eliminate an equal number of the worst solutions, i.e. for each "best chromosome" carried over a "worst chromosome" is deleted.



Example

Fit function: $f(x) = x_1 + 2x_2 + 3x_3$

Population:



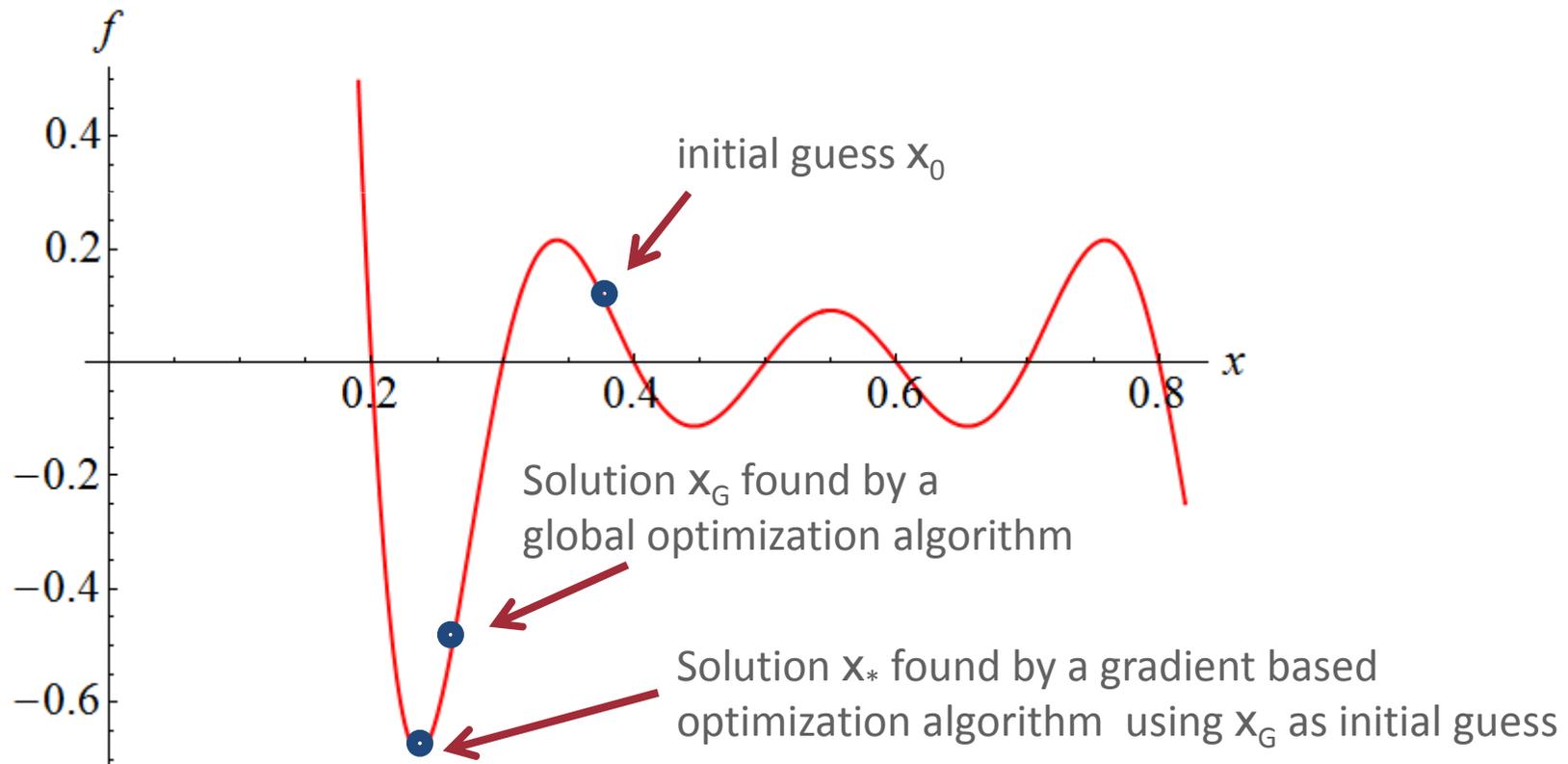
GA vs traditional methods

- GAs search a population of points in parallel, not only a single point
- Gas use probabilistic transition rules, not deterministic ones
- Gas work on an encoding of the parameter set rather than the parameter set itself
- Gas do not require derivative information or other auxiliary knowledge - only the objective function and corresponding fitness levels influence search



Hybrid approach

- Combines the advantage of global convergence properties of algorithms like GA and ASA and the property of fast convergence of a gradient based methods like FLOPC
- First find a solution which is close enough to the global minimizer and refine the solution using locally convergent FLOPC method



Stochastic and Deterministic Optimization

- Stochastic Optimization arises when a model depends on unknown quantities
- Frequently unknown quantities can be predicted or estimated with some degree of confidence
- To formulate an optimization problem the uncertainties must be quantified (scenario identification, probability distribution)
- Quantification of uncertainties are used by stochastic optimization algorithms to optimize the expected performance of the model
- In deterministic optimization problems, the model is fully specified



References

- Lecture noted from the MIT course 16.888 / ESD.77
“Multidisciplinary System Design Optimization”
- ***Numerical Optimization***, Jorge Nocedal and Stephen J. Wright, Springer, 1999
- LS-OPT[®] User’s Manual, 2009

