16.888
ESD.77

# Multidisciplinary System Design Optimization

## Genetic Algorithms (II)
## Tabu Search

**10 March 2004**

Lecture 11

Olivier de Weck

# Today's Topics

- More on Fitness Function Assignment
- Mutation
- Constraint implementation in GAs
- Multiobjective optimization with GAs
- Tabu Search
- Selection of Optimization Algorithms

# Fitness Function Mapping (I)



- Objective Function measures how individuals perform in the problem domain
- Raw measure of fitness usually only used as intermediate stage in determining relative performance of individuals in a GA

Transform objective function value into a measure of relative fitness:

*f* :  objective function

*g* :  transformation

*F*: relative Fitness (>= 0)

$$F(x) = g(f(x))$$

# Fitness Function Mapping (II)

$f \mapsto F$ Mapping always necessary for minimization (smaller objective value = higher fitness)

Often fitness function value corresponds to the number of offspring which an individual will likely produce.

E.g. Proportional fitness assignment

$$F(x_i) = \frac{f(x_i)}{\sum_{i=1}^{N_{ind}} f(x_i)}$$

Fitness of i-th individual = individuals raw performance relative to the whole population

$N_{ind}$ Population size

$x_i$ Phenotypic value of “i”

# Fitness Function Mapping (III)



How to account for negative objective function values ?

Linear transformation with offset: $F(x) = af(x) + b$

Scale factor: *a*>0 for maximizing, *a*<0 for minimizing
Offset: *b* ensures non-negative fitness values

Power law scaling: $F(x) = f(x)^k$

*k*: exponent (power) can be changed during execution

Tuning Knob: “SP” - selective pressure = degree of bias towards towards fittest

$$F(x_i) = 2 - SP + 2(SP - 1)\frac{x_i - 1}{N_{ind} - 1}$$

$x_i$ = position of i-th individual in ordered population

# Mutation(I)

16.888
ESD.77

(no) too little mutation leads to an impoverished genetic pool with increasing number of generations

dilemma

Too much mutation decreases convergence rate and undermines fitness-based selection bias

What is mutation? … a genetic operator

- Modifies chromosomes to restore diversity
- Permit random changes in a member of a population

Examples:

- with probability 1/20 randomly flip a single bit of a solution from 0 to 1 or 1 to 0
- probability of mutation often called “mutation rate”, expressing the probability $P_m$ that a bit is changed

# Example with Mutation

16.888
ESD.77

Improved population fitness with 1% mutation rate

| Original gen. | 5th gen. | 10th gen. |
|---|---|---|
| 1 0 0 1 1 | 1 1 0 1 1 | 1 1 1 1 1 |
| 0 1 0 0 0 | 1 0 1 1 1 | 1 1 1 1 1 |
| 0 0 0 0 1 | 1 1 1 1 1 | 1 1 0 1 1 |
| … | … | … |
| 0 0 0 0 0 | 0 1 1 1 0 | 1 1 1 1 1 |
| 1 1 0 1 1 | 1 1 1 1 1 | 1 1 1 1 1 |
| Avg. Fitness 2.6 | Avg. Fitness 4.8 | Avg. Fitness 4.9 |

16.888
ESD.77

# Example without Mutation

Stagnant population with 0% mutation rate

Original gen.

No “1”

1 0 0 1 1
0 1 0 0 0
0 0 0 0 1
…
0 0 0 0 0
1 1 0 1 1

Avg. Fitness
2.6

5th gen.

1 1 0 1 1
1 0 0 1 1
1 1 0 1 1
…
0 1 0 1 0
1 1 0 1 1

Avg. Fitness
3.2

10th gen.

1 1 0 1 1
1 1 0 1 1
1 1 0 1 1
…
1 1 0 1 1
1 1 0 1 1

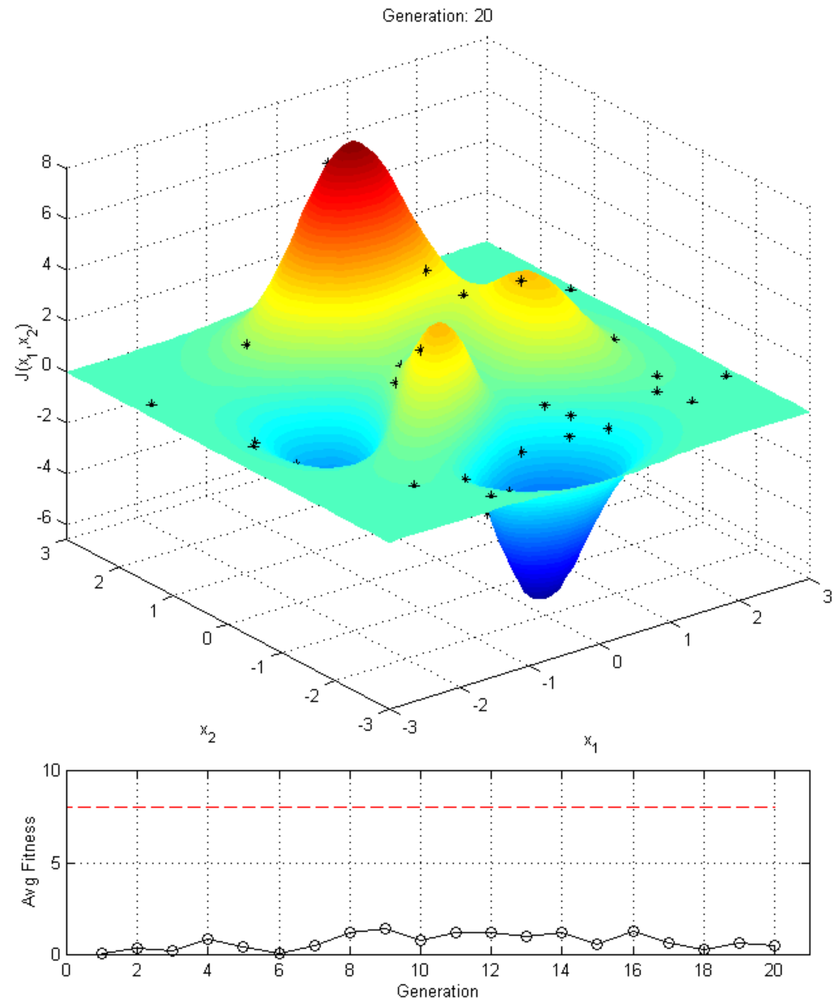
Can Never Achieve 11111

Avg. Fitness
4.0

# Mutation(II)

Example:

Before mutation: 010111000
After mutation: 010101000



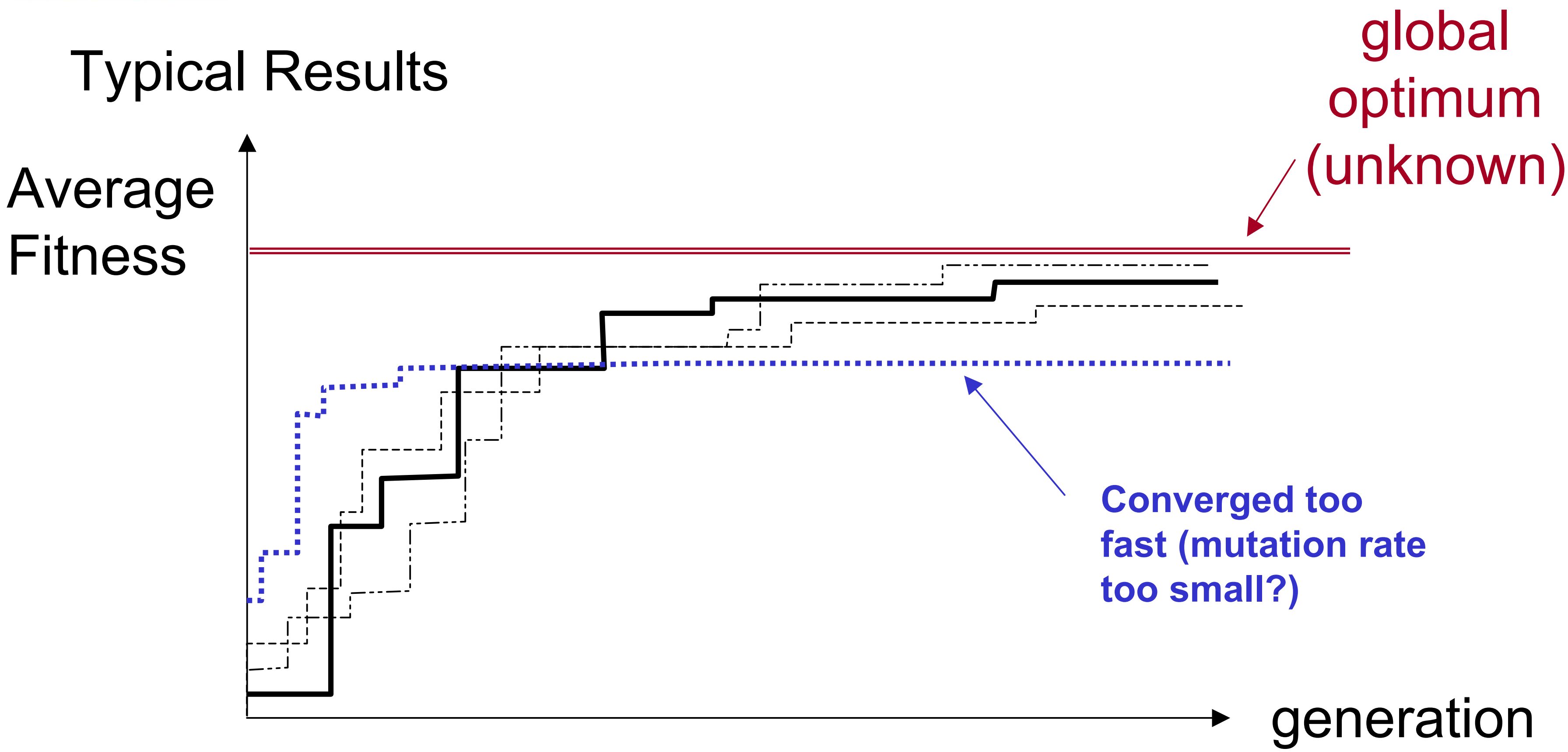

- Mutation rate can be variable (usually gradually decreasing with increasing number of generations)

- Mutation rate is an important "tuning knob" for a GA

# GA Convergence



Typical Results




<u>Average</u> performance of individuals in a population is expected to increase, as good individuals are preserved and bred and less fit individuals die out.

# Constraints in GA

Essentially three options:

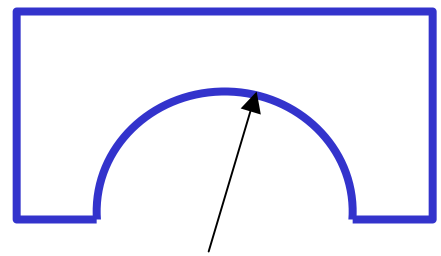
- Implement implicitely in coding/decoding scheme
- Penalize objective function for constraint violation
- Selection operator: only select valid solutions for mating

# Encoding/Decoding Scheme

16.888 ESD.77

Xi - design variable

$$\underbrace{\overbrace{0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1}^{x_1}}_{\text{Radius (genotype)}}\ \ 0\ \ 1\ \ 0\ \ldots$$



Radius R=2.57 [m]

Ensure that only "maximum" and "minimum" variable values are captured by coding scheme

E.g.
0 0 0 0 0 0 0 = 1.0 m (min)
1 1 1 1 1 1 1 = 3.0 m (max)

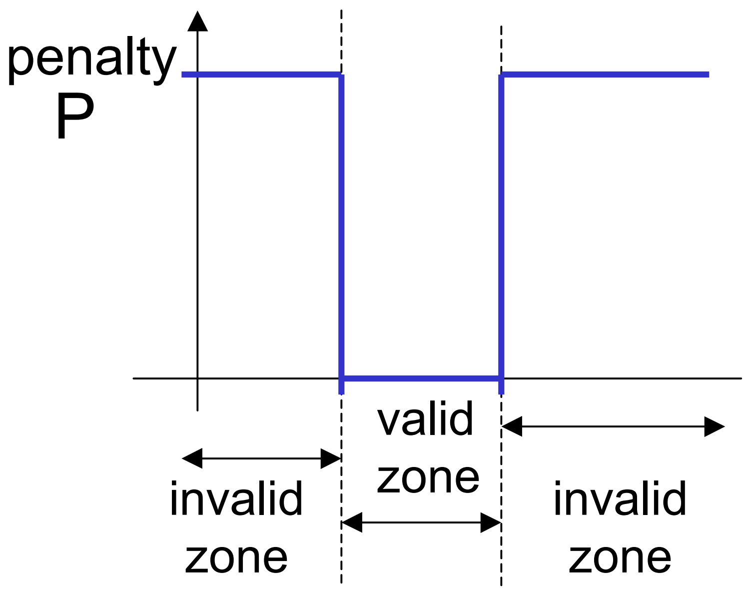
**Works well for implementing bounds $x_{i,LB} < x_i < x_{i,UB}$ but not general constraints such as $g(x)<0$, $h(x)=0$**

# Penalty Approach (I)

16.888
ESD.77

Usually some calculation is necessary to verify if a constraint is met or not, e.g. stresses, power output...

Solution: Penalize the fitness of solutions that violate constraints



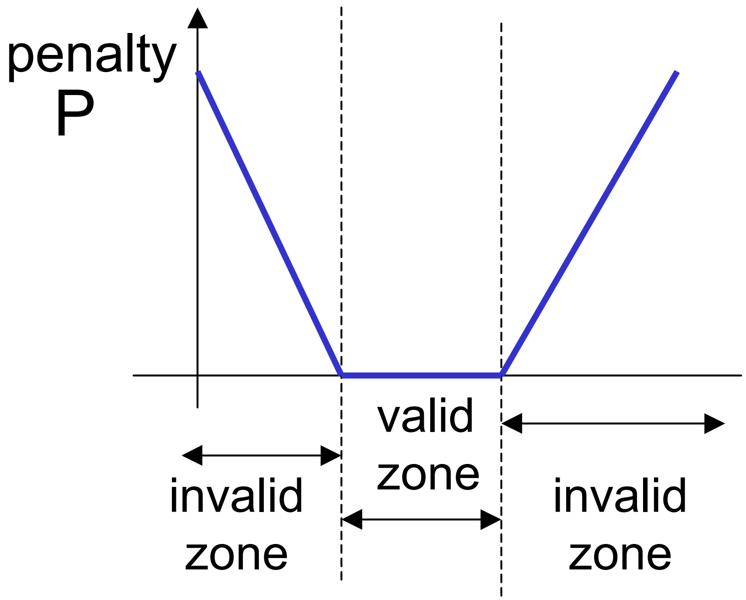

$$S(x_i) = F(x_i) - P(x_i)$$

***Fixed Penalty for Constraint Violation***

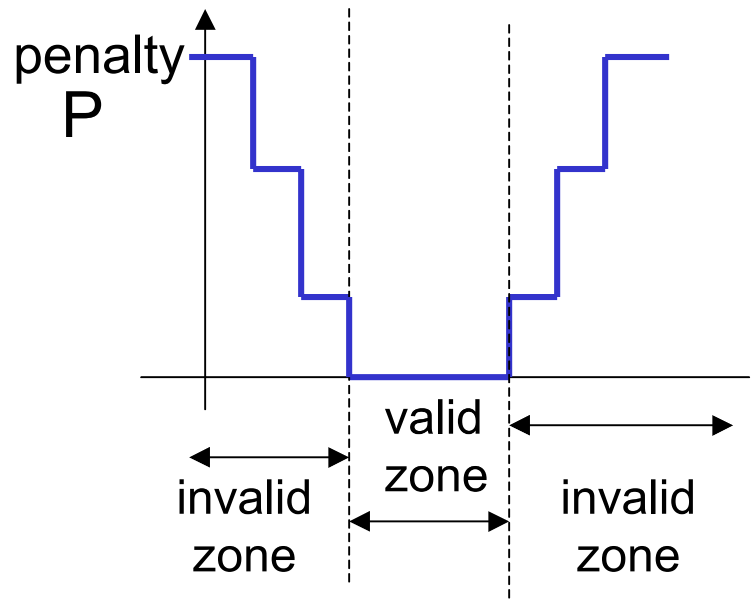
# Penalty Approach (II)



Fixed penalty provides no ranking of the degree of constraint violation - introduce variable penalty




*Linear variation*




*Stepped penalty*

Other schemes: polynomial, exponential (close to SA)

Important when constraints are hard to meet

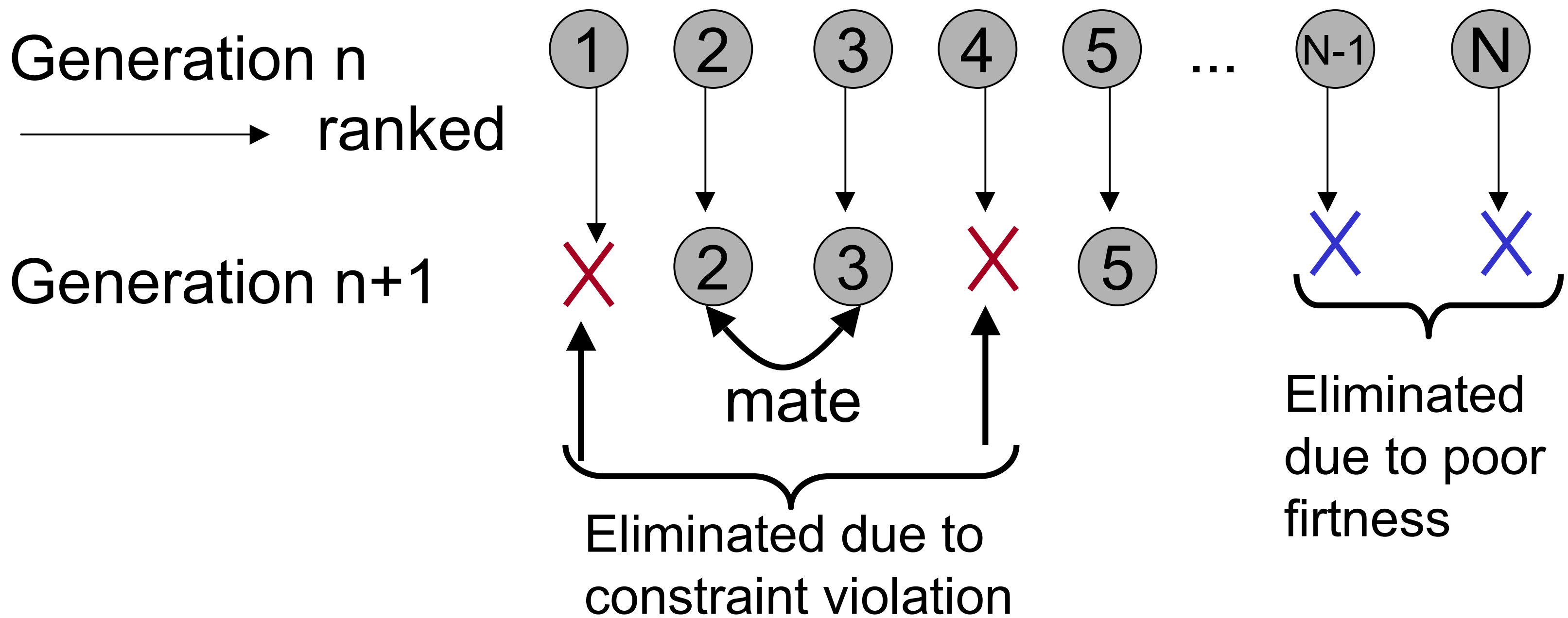
# Problems with penalty



- What is the right “balance” between objective function and penalty constraints ?
- Usually requires some amount of trial-and-error, tuning
- Usually amount of penalty varies during optimization
- Intially: small penalty = large search space
- Late: large penalty = focus on good feasible solutions
- But also opportunity: Allows for relative weigthing of constraints (crash worthiness vs. fuel economy)

# Selection Operator

Setting the Fitness of any of any invalid solution to zero ensures that only valid solutions are considered (selection)




Caution: Can eliminate valuable solutions from gene pool

# Operators: General Remarks



- 1 point crossover is one of many alternatives
- Goal of crossover: Take two parent solutions and create two children solutions
- Mutations: Flipping bits is one of many options
- Can take any neighborhood operator as in Simulated Annealing or Tabu Search
- Instead of doing random population initialization - start with a “fit” initial population
- Seed initial population with individuals known to be in the vicinity of the global optimum

# Parallel GA's

GA's are very ameniable to parallelization.

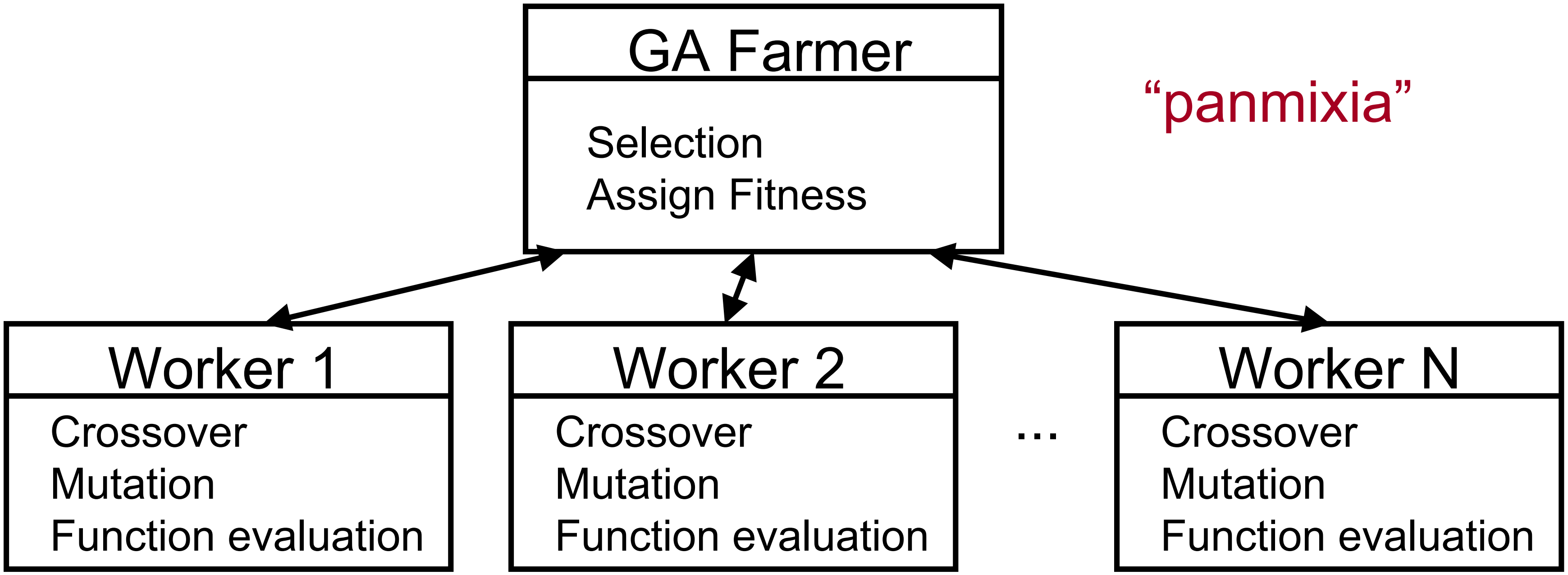
Motivations:
- faster computation (parallel CPU's)
- attack larger problems
- introduce structure and geographic location

There are three classes of parallel GA's:

- Global GA's
- Migration GA's
- Diffusion GA's

Main differences lie in :
- population structure
- method of selecting individuals for reproduction

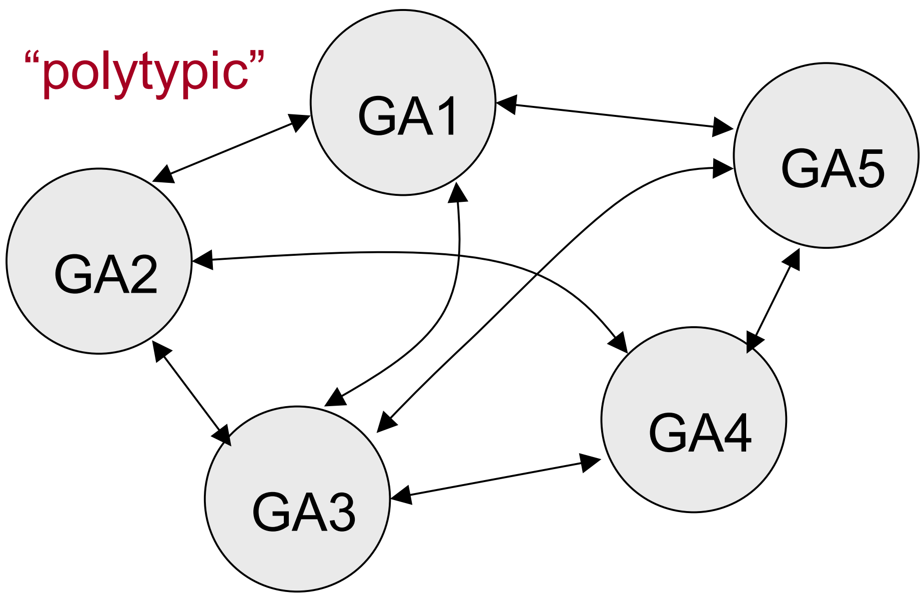
# Global GA




- GA Farmer node initializes and holds entire population
- Interesting when objective function evaluation expensive
- Typically implemented as a master-slave algorithm
- Balance serial-parallel tasks to minimize bottlenecks
- Issue of synchronous/asynchronous operation

# Migration GA

16.888
ESD.77




```
-- Each node (Gai)
WHILE not finished
  SEQ
      … Selection
      … Reproduction
      … Evaluation
      PAR
            … send emigrants
            … receive immigrants
```

Does NOT operate globally on a single population

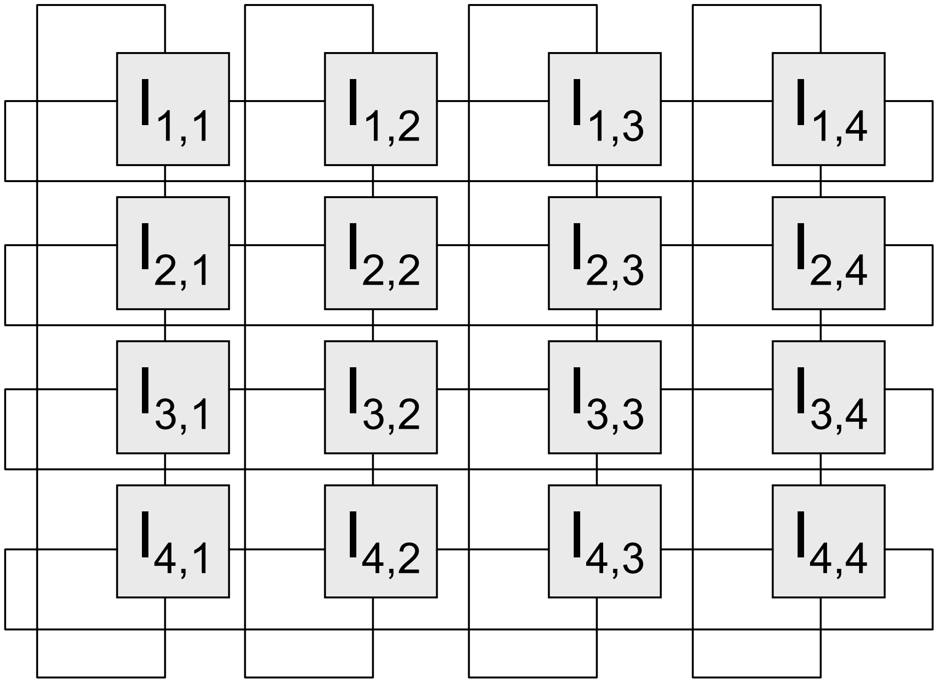
Each node represents a subgroup relatively isolated from each other

*"breeding groups"= demes*

More closely mimics biological metaphor

First introduced by Grosso in 1985

# Diffusion GA's




*Toroidal-Mesh parallel processing network*

```
-- Each Node (Ii,j)
WHILE not finished
SEQ
… Evaluate
PAR
   … send self to neighbors
   … receive neighbors
… select mate
… reproduce
```

Neighborhood, cellular or fine-grained GA

- Population is a single continuous structure, but
- Each individual is assigned a geographic location
- Breeding only allowed within a small local neighborhood
- Example: I(2,2) only breeds with I(1,2), I(2,1),I(2,3),I(3,2)

# Multiobjective GA's

- Many engineering design problems have multiple objectives (often competing)
- Example: Maximize range, minimize fuel usage, maximize cruise speed, maximize passenger volume ...

GA's are ameniable to multi-objective problems

Typically GA's are used similar to traditional Optimizers and multiple objectives are scalarized:

$J_j$ - j-th objective value
$w_j$ - weight of j-th objective
$n_j$ - j-th objective normalization

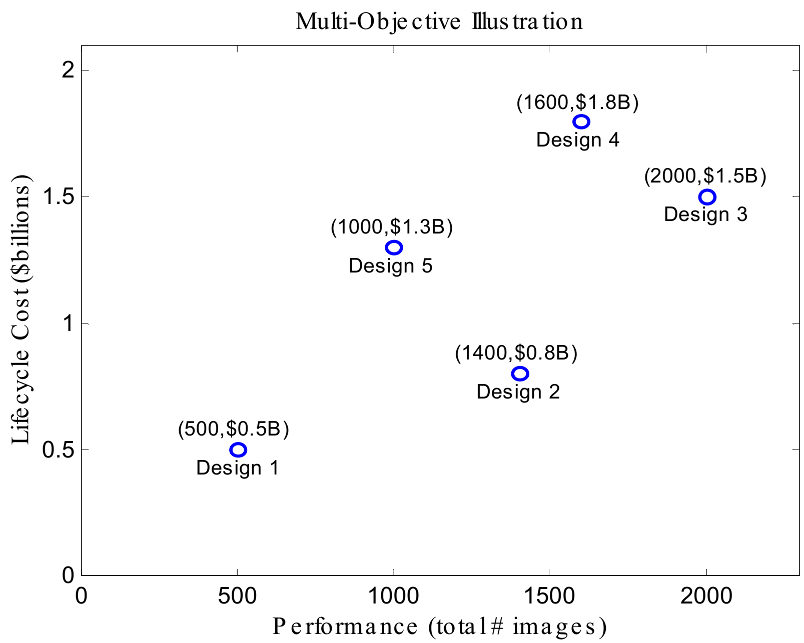
$$J_i = \sum_{j=1}^{n_z} w_j \frac{J_j}{n_j}$$

GA's can naturally deal with multiple objectives

# Pareto Optimality - Ranking



Simple: Pareto Ranking Schemes
Complicated: Mating Restrictions

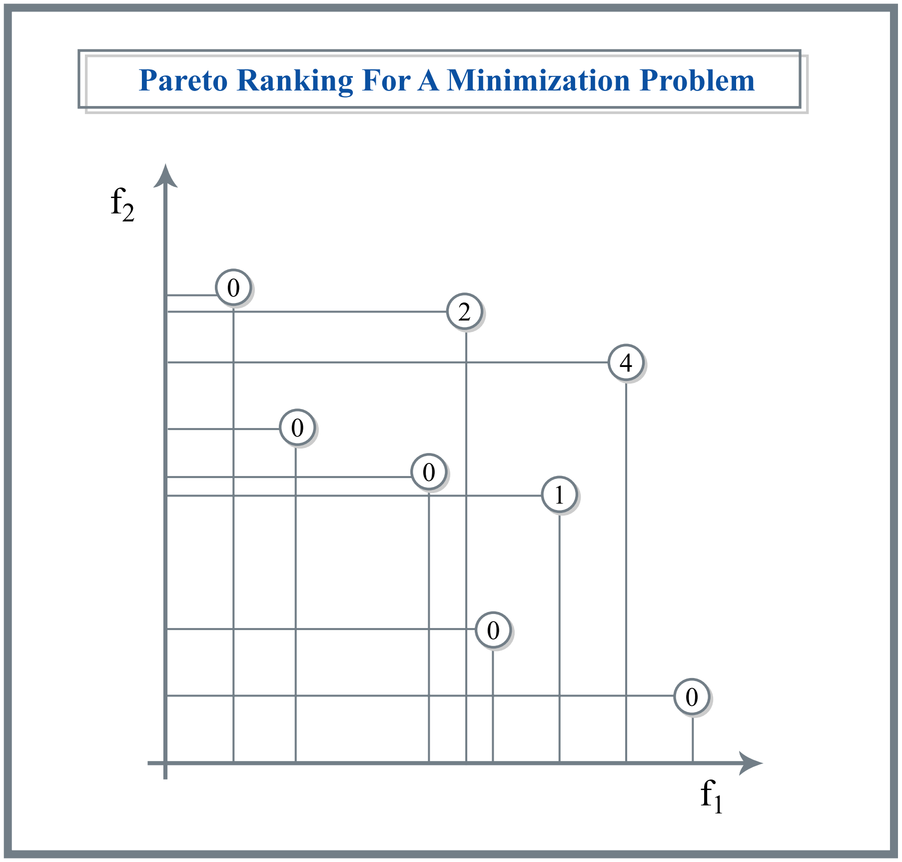



Pareto optimal: Best in a tradeoff sense.

An improvement in one objective can only be achieved at the expense of at least one other objective.

⇨ Which designs are pareto optimal ? (2 min)

# Pareto Fitness - Ranking




- Pareto ranking scheme
- Allows ranking of population without assigning preferences or weights to individual objectives
- Successive ranking and removal scheme
- Deciding on fitness of dominated solutions is more difficult.

# Multiobjective GA



Goldberg, David E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, January 1, 1989. ISBN: 0201157675.

# Example

Minimization

Objective 1

$$f_1\left(x_1,\ldots,x_n\right)=1-\exp\left(-\sum_{i=1}^{n}\left(x_i-\frac{1}{\sqrt{n}}\right)^2\right)$$

Objective 2

$$f_1\left(x_1,\ldots,x_n\right)=1-\exp\left(-\sum_{i=1}^{n}\left(x_i+\frac{1}{\sqrt{n}}\right)^2\right)$$

Images removed due to copyright considerations.

Need to think about mating restrictions in multiobj-GA

# Good News about GA's

- GA work well on mixed discrete/continuous problems
- GA's require little information about problem
- No gradients required
- Simple to understand and set up and implement
- Can operate on various representations
- GA's are very robust
- GA's are stochastic, that is, they exploit randomness
- GA's can be easily parallelized

# Bad News about GA's

16.888
ESD.77

- GA implementation is still an art and requires some experience
- Convergence behavior very dependent on some tuning parameters: mutation rate, crossover, population size
- Designing fitness function can be tricky
- Cumbersome to take into account constraints
- GA's can be computationally expensive
- No clear termination criteria
- No knowledge of true global optimum

# Frequent Applications of GA's



- Scheduling and Planning, Assy Sequencing
- Packing (2D and 3D)
- Travel, Path Planning, Trajectory Optimization
- Parameter Selection for Curve-Fitting
- Catalog Search
- Structural Topology Optimization
- Multidisciplinary Design Optimization (MDO)

# MATLAB GA Toolbox

- Can implement GA's directly in MATLAB
- Not officially part of Optimization Toolbox
- But have user-contributed toolbox in 16.888/GA Toolbox

The
main genetic algorithm M-file is **genetic.m**;

```
GENETIC tries to maximize a function using a simple genetic algorithm.
         X=GENETIC('FUN',X0,OPTIONS,VLB,VUB) uses a simple (haploid)
        genetic algorithm to find a maximum of the fitness function
        FUN (usually an M-file: FUN.M).
```

Demo using a simple function

# GA in iSIGHT

- GA is available in iSIGHT
- Algorithm tuning parameters can be set
- Demonstration using the Fence example.
- see Friday lab session

Compare behavior of gradient search technique versus genetic algorithms (A3)

# Tabu Search (TS)



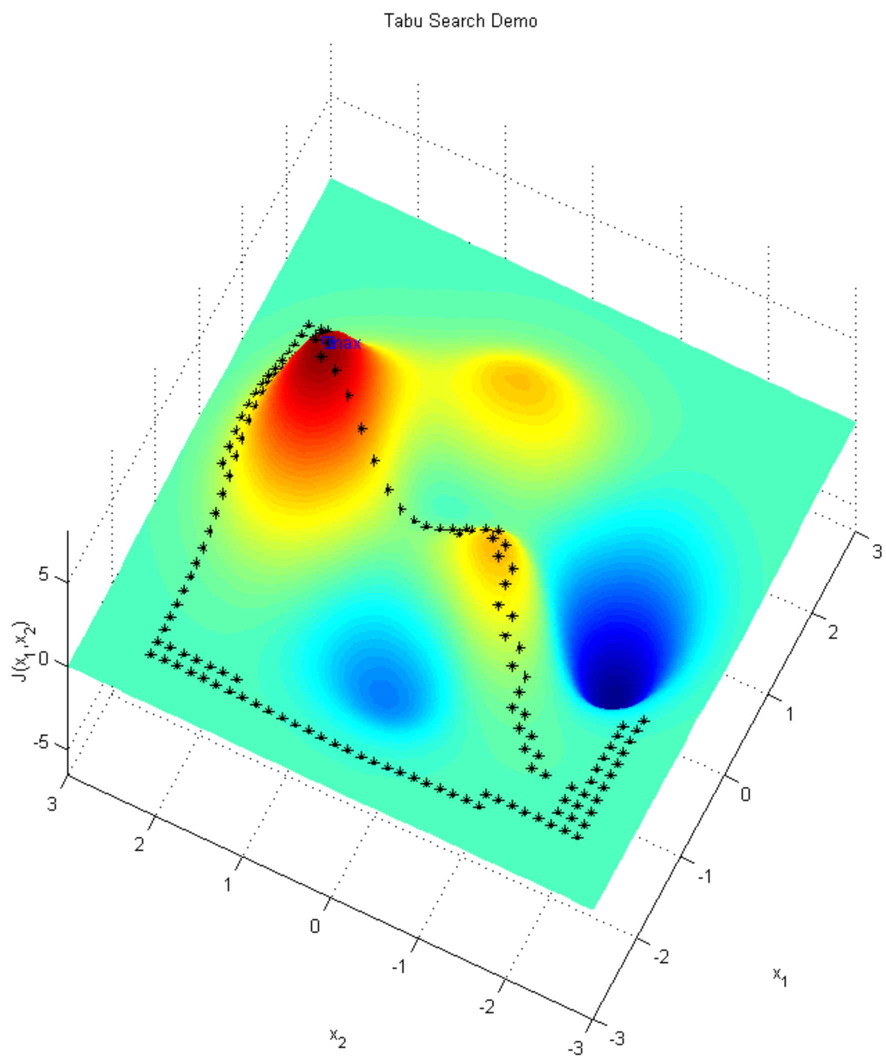
- Attributed to Glover (1986)
- Search by avoiding points in the design space that were previously visited ("tabu")
- Accept a new "poorer" solution if it avoids a solution that was already investigated
- Intent: Avoid local minima
- Record all previous moves in a "running list" = memory
- Record recent, now forbidden moves in a "tabu" list
- First "diversification" then "intensification"
- Applied to combinatorial optimization problems
- Glover F., and Laguna M., Tabu Search, in *Modern Heuristic Techniques for Combinatorial Problems*, C.R. Reeves, editor, John Wiley & Sons, Inc, 1993
- www.tabusearch.net

# Tabu Search (minimization)



```
  Given a feasible solution x* with objective
  function value J*, let x := x* with J(x) = J*.
  Iteration:
while stopping criterion is not fulfilled do
  begin
•  select best admissible move that transforms x
   into x' with objective function value J(x')
   and add its attributes to the running list
(2) perform tabu list management: compute moves
   (or attributes) to be set tabu, i.e., update
   the tabu list
(3) perform exchanges: x := x', J(x) = J(x'); if
   J(x) < J* then J* := J(x), x* := x
endif
endwhile
Result: x* is the best of all determined
   solutions, with objective function value J*.
```

# Tabu Search Demo



# Selection of Algorithms

- Linearity and smoothness of $J(\boldsymbol{x})$ and/or of the constraints $\mathbf{g}(x)$, $\mathbf{h}(x)$
- Type of design variables **x** (real, integer,…)
- Number of design variables *n*
- Expense of evaluating $J(\boldsymbol{x})$ – [CPU, Flops]
- Expense of evaluating gradient of $J(\boldsymbol{x})$
- Number of objectives, z

# Nonlinearity



**<u>Crumpled Paper Analogy to Show Nonlinearity:</u>**

- **Use a sheet of paper to represent the response surface of**

$$J = f(x_1, x_2)$$

- **If the paper is completely "flat", with or without slope, then y is a *<u>Linear</u>* Function which can be represented as**

$$y \quad c_0 + c_1\, x_1 + c_2\, x_2$$

- **If the paper is twisted slightly with some curvature, then it becomes a nonlinear function. Low nonlinearity like this may be approximated by a *<u>Quadratic</u>* function like**

$$y = c_0 + c_1 x_1 + c_2 x_2 + c_3 x_1^2 + c_4 x_2^2 + c_5 x_1 x_2$$

- **Crumple the paper and slightly flatten it, then it becomes a "*<u>very nonlinear</u>*" function.** Observe the irregular terrain and determine whether it is possible to approximate the irregular terrain by a simple quadratic function.

# Algorithm Selection Matrix



| | Linear<br>*J* and *g* and *h* | Nonlinear<br>J or g or h |
|---|---|---|
| Continuous, real<br>**x** (all) | Simplex<br>Barrier Methods | SQP (constrained)<br>Newton (unconstrained) |
| Discrete<br>**x** (at least one) | MILP<br>(Branch-and-Bound) | GA<br>SA, Tabu Search<br>PSO |

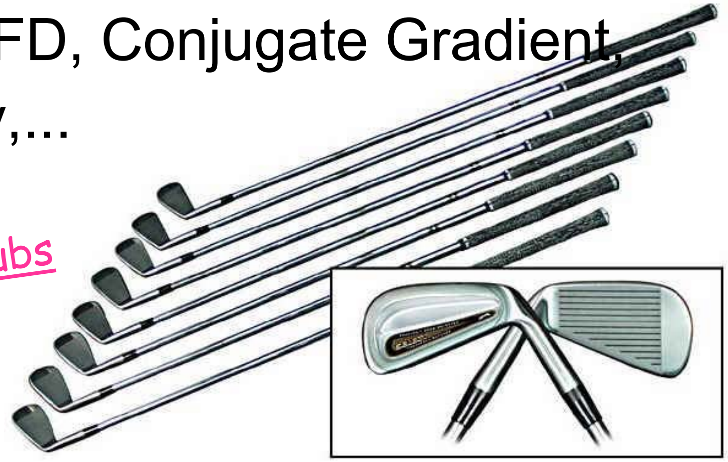
Courtesy of Howard Lee, General Electric. Used with permission.

## Golf Clubs Analogy

**Gradient-Based:**

SLP, SQP, MMFD, Conjugate Gradient, Exterior Penalty,...

Iron Clubs



**Heuristics-Based:**

Rules-Guided Search

Putter

**Stochastic-Based:**

Simulated Annealing, Genetic Algorithms.

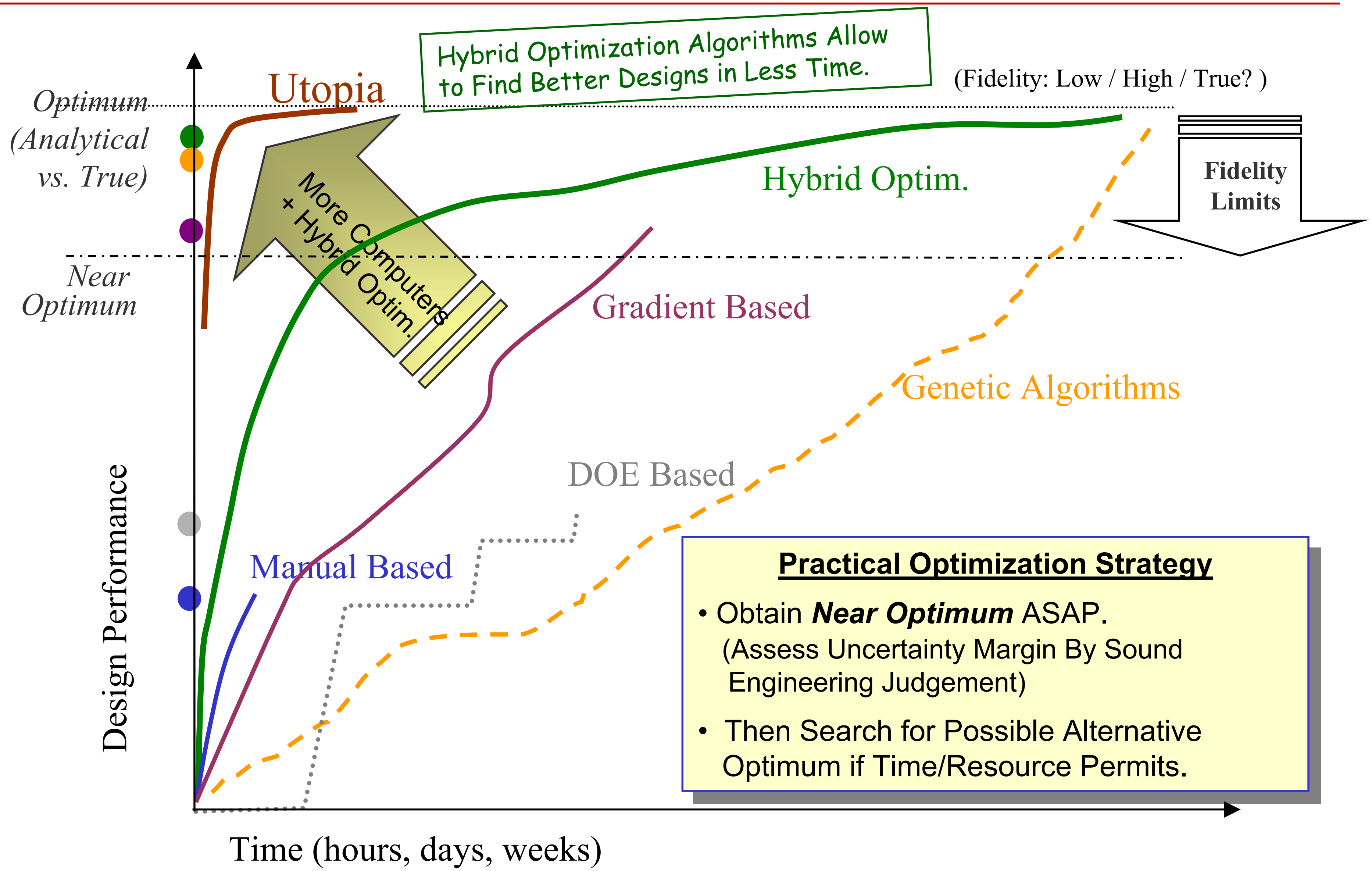
Wood Clubs

**Hybrid Optimization Algorithms:**

Use a Combination of "Clubs" to Search Optimum to Leverage the Strength of Individual Club.

Courtesy of Howard Lee, General Electric. Used with permission.

# Practical Optimization Strategy

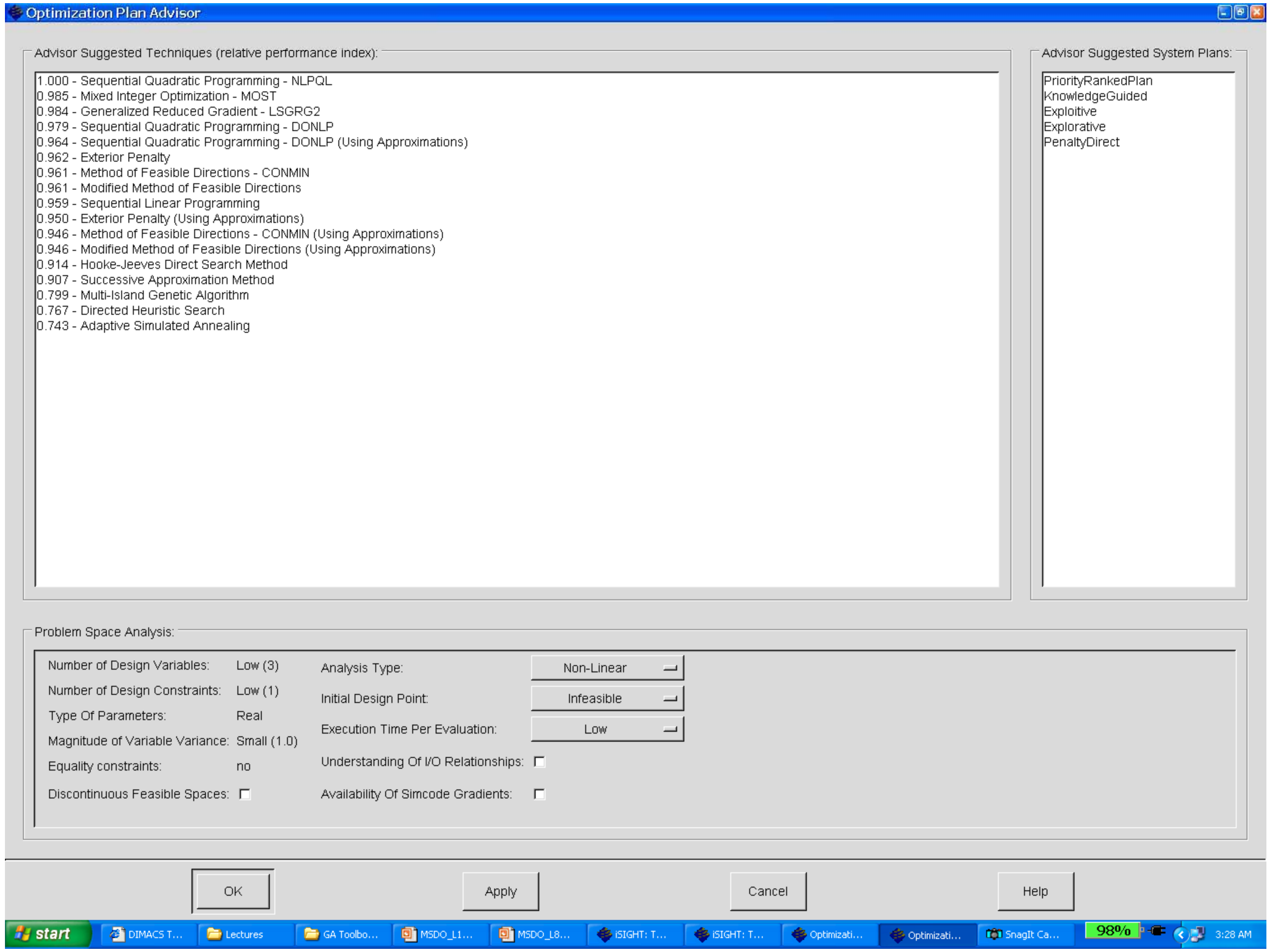



**Practical Optimization Strategy**

- Obtain ***Near Optimum*** ASAP.
  (Assess Uncertainty Margin By Sound Engineering Judgement)
- Then Search for Possible Alternative Optimum if Time/Resource Permits.

# iSIGHT Optimization Plan Advisor

Ranking of algorithms according to their suitability to the Problem at hand



# Summary

- Gradient Search Techniques
  - Efficient, repeatable, use gradient information
  - Well suited for nonlinear, continuous variables
  - Can easily get trapped at local optima
- Heuristic Techniques
  - Used for combinatorial and discrete variable problems
  - Use both a rule set and randomness
  - don’t use gradient information, search broadly
  - Avoid local optima, but are expensive
- Hybrid Approaches
  - Use effective combinations of search algorithms