Multidisciplinary System Design Optimization

Genetic Algorithms (cont.)
Particle Swarm Optimization
Tabu Search
Optimization Algorithm Selection

Lecture 12

Olivier de Weck

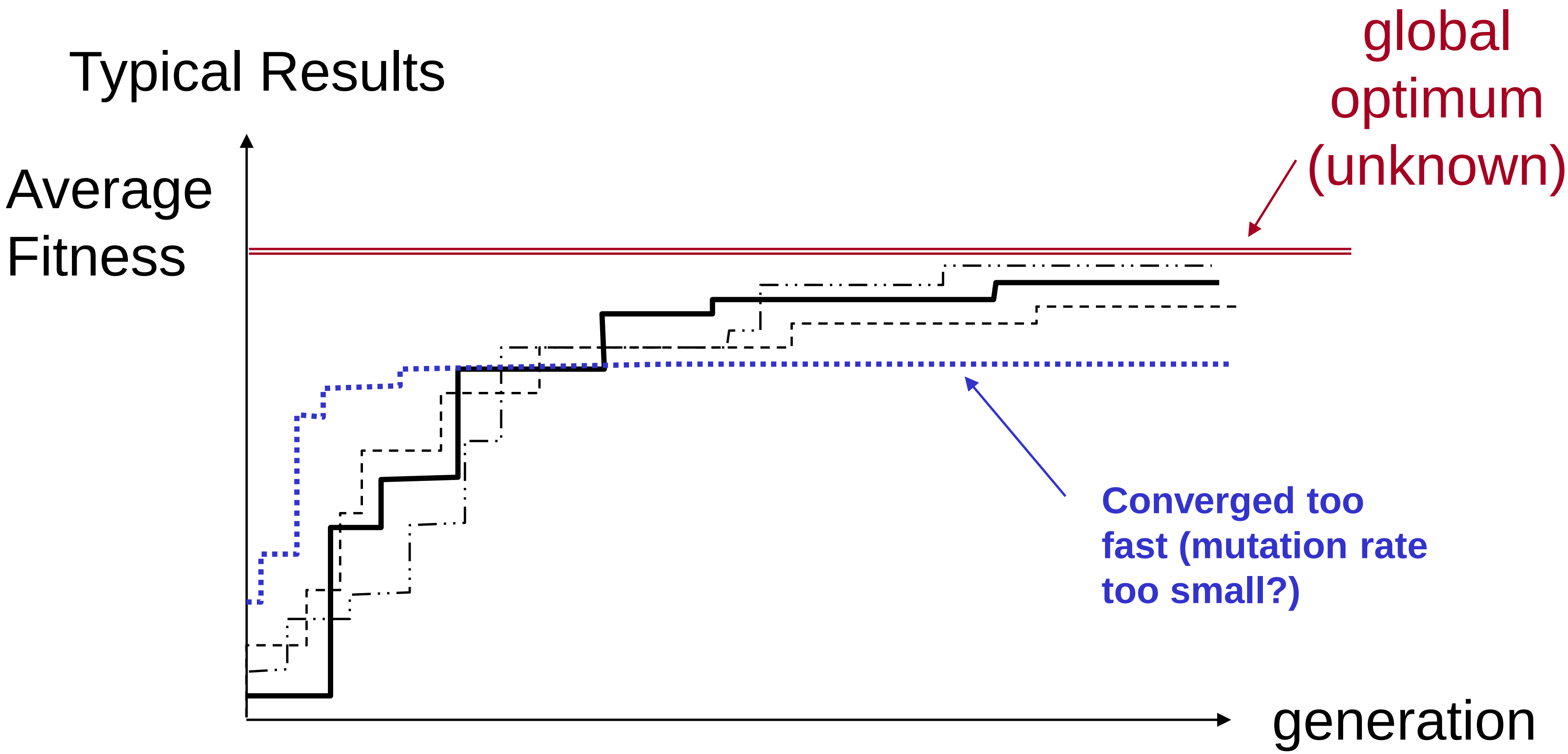
Today's Topics

- Genetic Algorithms (part 2)
- Particle Swarm Optimization
- Tabu Search
- Selection of Optimization Algorithms

Genetic Algorithms (Part 2)

GA Convergence

Typical Results



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

GAs versus traditional methods

Differ from traditional search/optimization 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 design variable set rather than the variable set itself
- GAs do not require derivative information or other auxiliary knowledge - only the objective function and corresponding fitness levels influence search

Parallel GA's

GA's are very ameniable to parallelization.

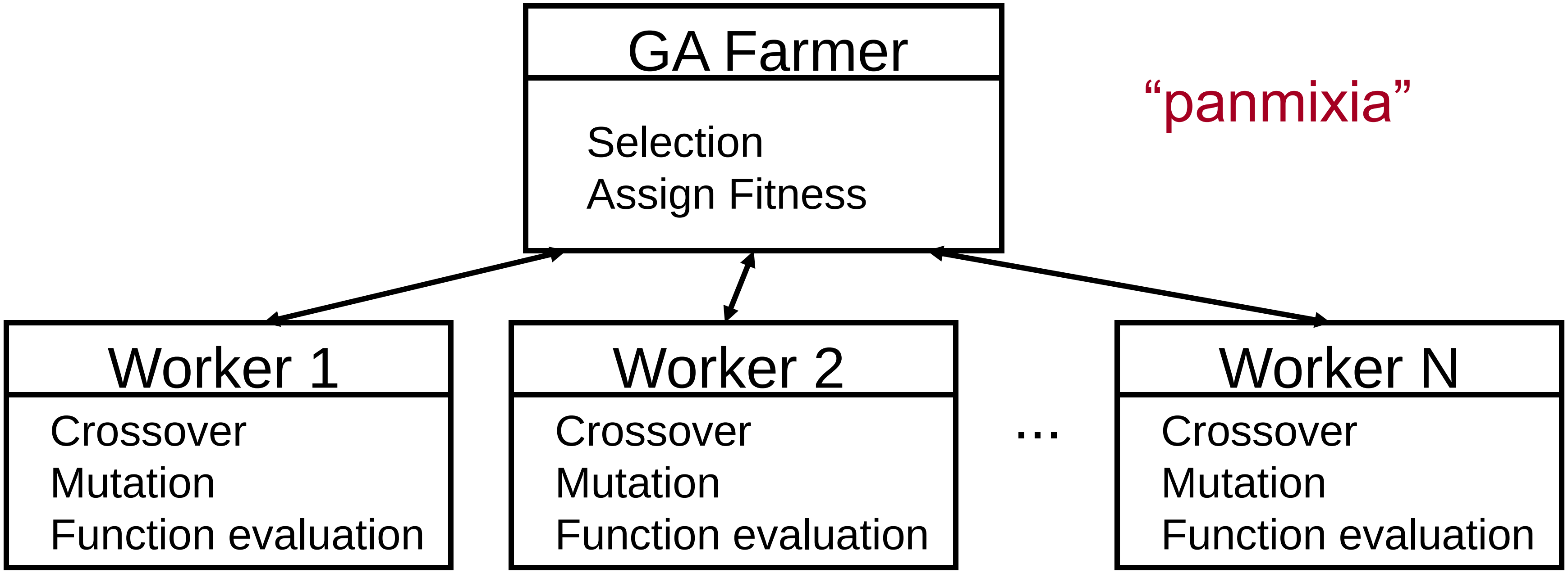
Motivations: - faster computation (parallel CPU's)
- attack larger problems
- introduce structure and geographic location

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

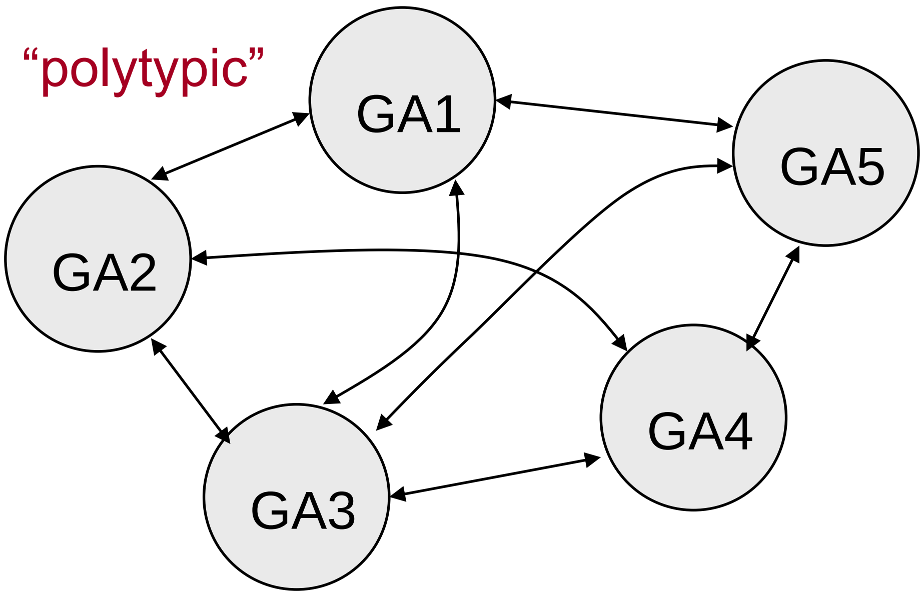
Global GA



"panmixia"

- 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




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

Does NOT operate globally on a single population

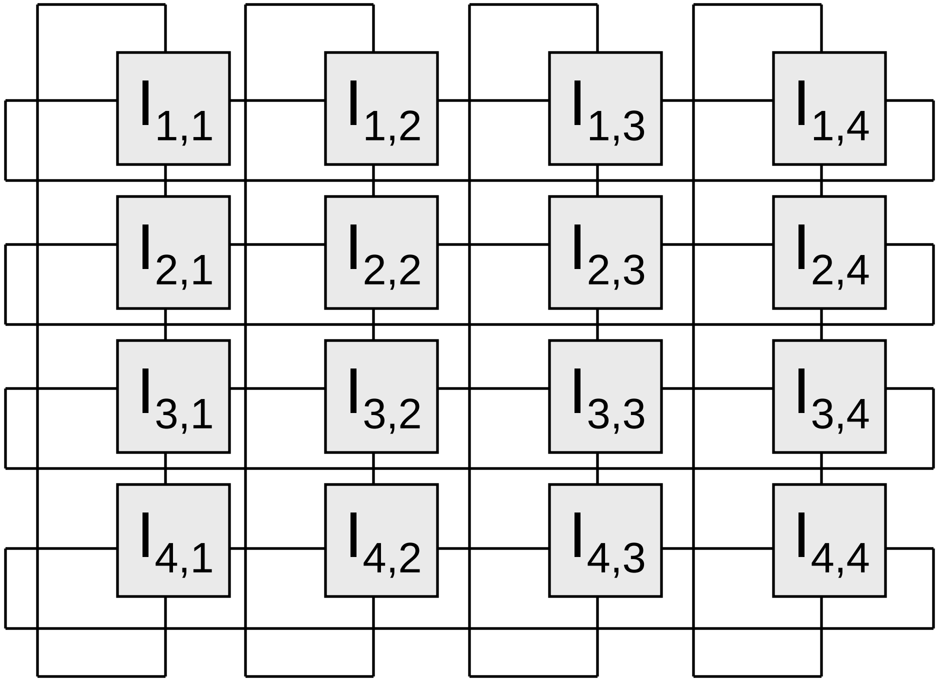
Each node represents a subgroup relatively isolated from each other

"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)

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

- 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

Particle Swarm Optimization

Introduced in 1995: Kennedy, J. and Eberhart, R., “Particle Swarm Optimization,” Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia 1995, pp. 1942-1945.

Lecture Notes prepared by and courtesy of Rania Hassan. Used with permission.

Particle Swarm Optimization

A pseudo-optimization method (heuristic) inspired by the collective intelligence of swarms of biological populations.

Flocks of Birds

Colonies of Insects

Weimerskirch, H. et al. "Energy saving in flight formation." Nature 413, (18 October 2001): 697 - 698.

A study of great white pelicans has found that birds flying in formation use up to a fifth less energy than those flying solo (Weimerskirch *et al*.).

PSO Conceptual Development

- How do large numbers of birds produce seamless, graceful flocking choreography, while often, but suddenly changing direction, scattering and regrouping?
  - "Decentralized" local processes.
  - Manipulation of inter-individual distances (keep pace and avoid collision).
- Are there any advantages to the swarming behavior for an individual in a swarm?
  - Can profit from the discoveries and previous experience of other swarm members in search for food, avoiding predators, adjusting to the environment, i.e. information sharing yields evolutionary advantage.
- Do humans exhibit social interaction similar to the swarming behavior in other species?
  - Absolutely, humans learn to imitate physical motion early on; as they grow older, they imitate their peers on a more abstract level by adjusting their beliefs and attitudes to conform with societal standards.

Basic PSO Algorithm

- The swarming behavior of the birds could be the reason for finding optimal food resources.
- A swarming model could be used (with minor modifications) to find optimal solutions for *N*-dimensional, non-convex, multi-modal, nonlinear functions.

Algorithm Description

- Particle Description: each particle has three features
  - Position $\mathbf{x}_k^i$ (this is the i^{th} particle at time k, notice vector notation)
  - Velocity $\mathbf{v}_k^i$ (similar to search direction, used to update the position)
  - Fitness or objective $f\left(\mathbf{x}_k^i\right)$ (determines which particle has the best value in the swarm and also determines the best position of each particle over time.

Basic PSO Algorithm

- Initial Swarm
  - No well established guidelines for swarm size, normally 10 to 60.
  - particles are randomly distributed across the design space.

$$\mathbf{x}_0^i = \mathbf{x}_{\min} + rand\left(\mathbf{x}_{\max} - \mathbf{x}_{\min}\right)$$

where $\mathbf{x}_{\min}$ and $\mathbf{x}_{\max}$ are vectors of lower and upper limit values respectively.

  - Evaluate the fitness of each particle and store:
    - particle best ever position (particle memory $\mathbf{p}^i$ here is same as $\mathbf{x}_0^i$)
    - Best position in current swarm (influence of swarm $\mathbf{p}_0^g$)
  - Initial velocity is randomly generated.

$$\mathbf{v}_0^i = \frac{\mathbf{x}_{\min} + rand\left(\mathbf{x}_{\max} - \mathbf{x}_{\min}\right)}{\Delta t} = \frac{\text{position}}{\text{time}}$$

16.888
ESD.77

Basic PSO Algorithm

- Velocity Update
  - provides search directions
  - Includes deterministic and probabilistic parameters.
  - Combines effect of current motion, particle own memory, and swarm influence.

New velocity

$$\mathbf{v}_{k+1}^{i} = w\,\underbrace{\mathbf{v}_{k}^{i}}_{\text{current motion}} + c_1\, rand\, \underbrace{\frac{\left(\mathbf{p}^{i} - \mathbf{x}_{k}^{i}\right)}{\Delta t}}_{\text{particle memory influence}} + c_2\, rand\, \underbrace{\frac{\left(\mathbf{p}_{k}^{g} - \mathbf{x}_{k}^{i}\right)}{\Delta t}}_{\text{swarm influence}}$$

inertia factor

0.4 to 1.4

self confidence

1.5 to 2

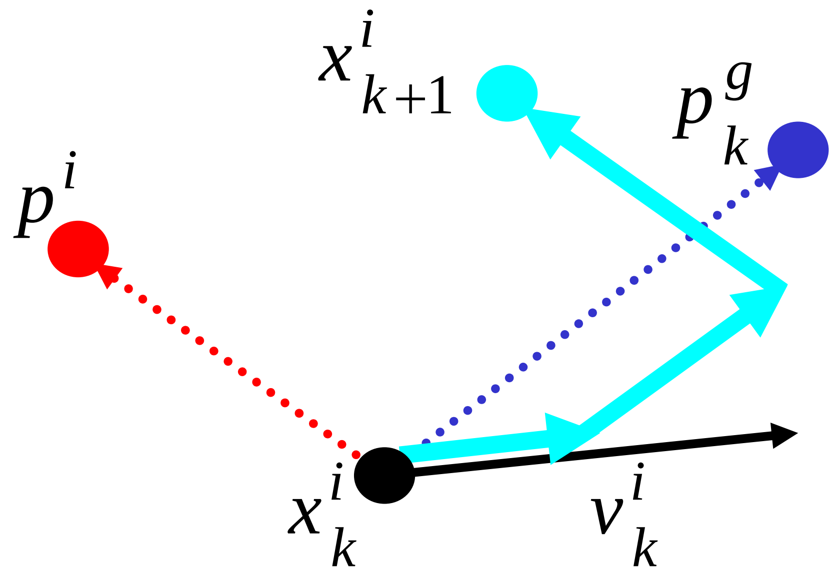
swarm confidence

2 to 2.5

Basic PSO Algorithm

- Position Update
  - Position is updated by velocity vector.

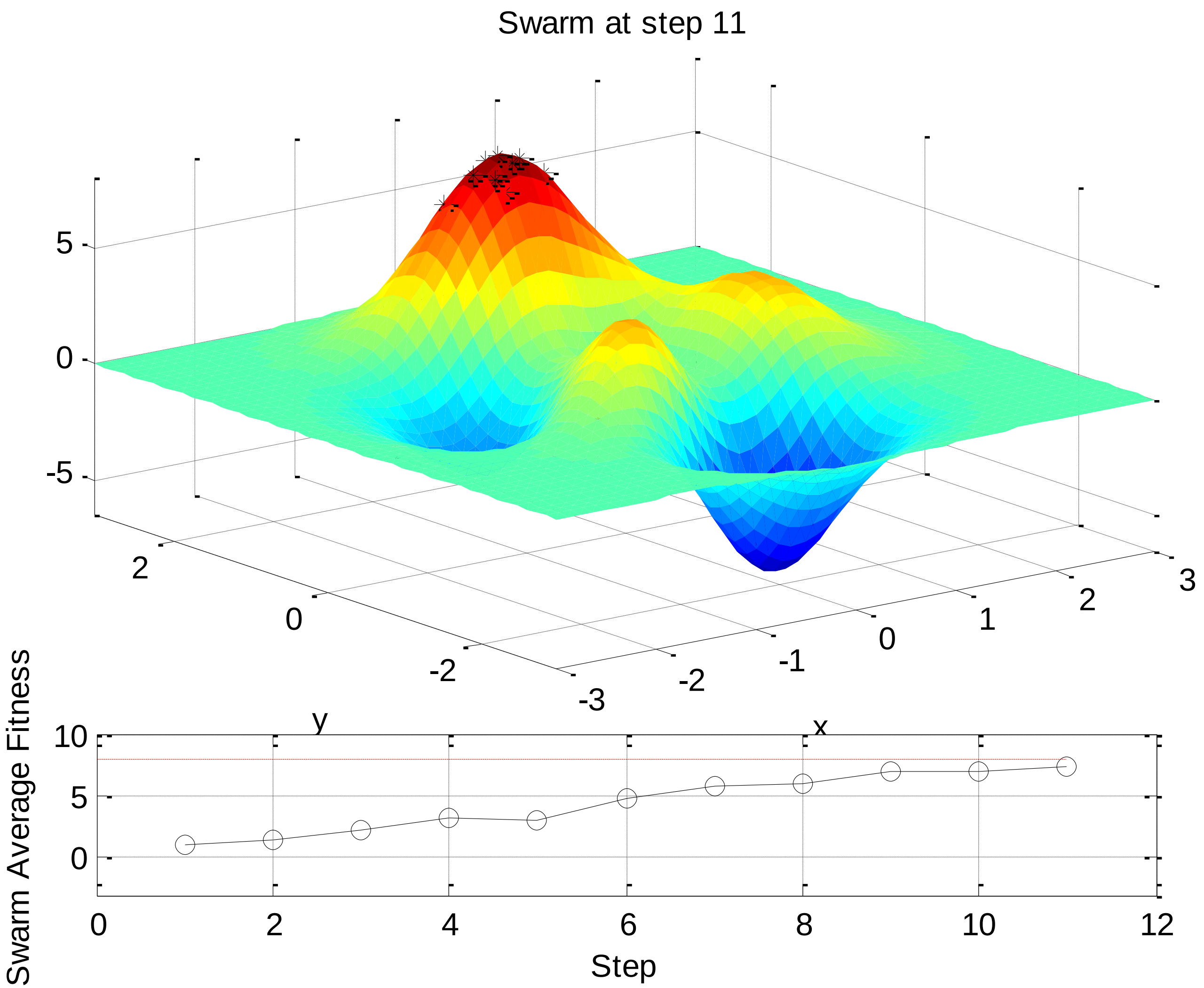
$$\mathbf{x}_{k+1}^{i} = \mathbf{x}_{k}^{i} + \mathbf{v}_{k+1}^{i} \Delta t$$



- Stopping Criteria
  - Maximum change in best fitness smaller than specified tolerance for a specified number of moves (*S*).

$$\left| f\left(\mathbf{p}_{k}^{g}\right) - f\left(\mathbf{p}_{k-q}^{g}\right) \right| \leq \varepsilon \quad q = 1,2,..S$$

PSO Peaks Demo



Constraint Handling

- Side Constraints
  - Velocity vectors can drive particles to “explosion”.
  - Upper and lower variable limits can be treated as regular constraints.
  - Particles with violated side constraints could be reset to the nearest limit.
- Functional Constraints
  - Exterior penalty methods (linear, step linear, or quadratic).

fitness function

$$f(\mathbf{x}) = \underbrace{\phi(\mathbf{x})}_{\text{objective function}} + \underbrace{\sum_{i=1}^{N_{con}} \overbrace{r_i}^{\text{penalty multipliers}} \left(\max\left[0, g_i\right]\right)^2}_{\text{penalty function}}$$

  - If a particle is infeasible, last search direction (velocity) was not feasible. Set current velocity to zero.

$$\mathbf{v}_{k+1}^{i} = c_1\, rand\, \frac{\left(\mathbf{p}^{i} - \mathbf{x}_k^{i}\right)}{\Delta t} + c_2\, rand\, \frac{\left(\mathbf{p}_k^{g} - \mathbf{x}_k^{i}\right)}{\Delta t}$$

Discretization

- System problems typically include continuous, integer, and discrete design variables.
- Basic PSO works with continuous variables.
- There are several methods that allows PSO to handle discrete variables.
- The literature reports that the simple method of rounding particle position coordinates to the nearest integers provide the best computational performance.

Constrained Benchmark Problems
Golinski Speed Reducer

- This problem represents the design of a simple gear box such as might be used in a light airplane between the engine and propeller to allow each to rotate at its most efficient speed.
- The objective is to minimize the speed reducer weight while satisfying a number of constraints (11) imposed by gear and shaft design practices.
- Seven design variables are available to the optimizer, and each has an upper and lower limit imposed.
- PSO parameters:
  - Swarm Size = 60
  - Inertia, w = 0.5 (static)
  - Self Confidence, $c_1 = 1.5$
  - Swarm Confidence, $c_2 = 1.5$
  - Stopping Tolerance, ε = 5 g

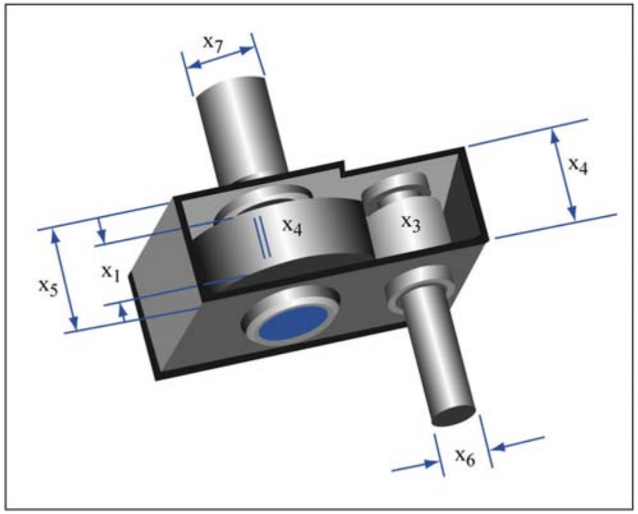


Image by MIT OpenCourseWare.

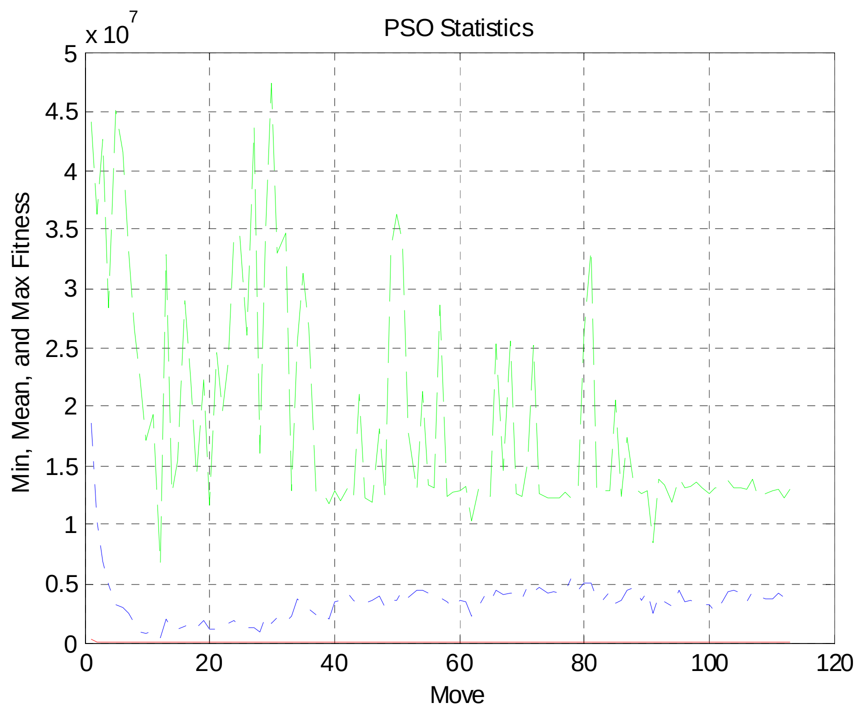
Constrained Benchmark Problems
Golinski Speed Reducer

- Known solution

$$\mathbf{x} = [3.50 \quad 0.7 \quad 17 \quad 7.3 \quad 7.30 \quad 3.35 \quad 5.29]$$
$$f(\mathbf{x}) = 2985 \text{ g}$$

- PSO solution

$$\mathbf{x} = [3.53 \quad 0.7 \quad 17 \quad {\color{red}8.1} \quad {\color{red}7.74} \quad 3.35 \quad 5.29]$$
$$f(\mathbf{x}) = {\color{red}3019 \text{ g}}$$



Final Comments on PSO

- This is a method “in the making” - many versions are likely to appear.

- Poor repeatability in terms of:
  - finding optimal solution
  - computational cost

- More robust constraint (side and functional) handling approaches are needed.

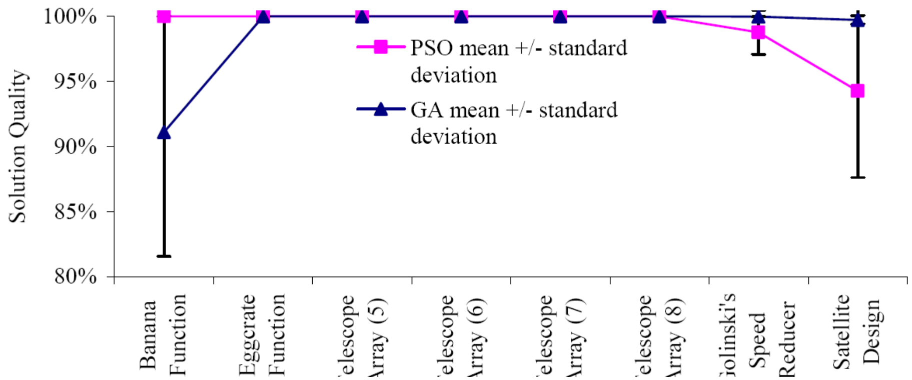
- Guidelines for selection of swarm size, inertia and confidence parameters are needed.

- We performed some research on the comparison of effectiveness and efficiency of PSO versus GA
  - Claim is that PSO is more computationally efficient than GA

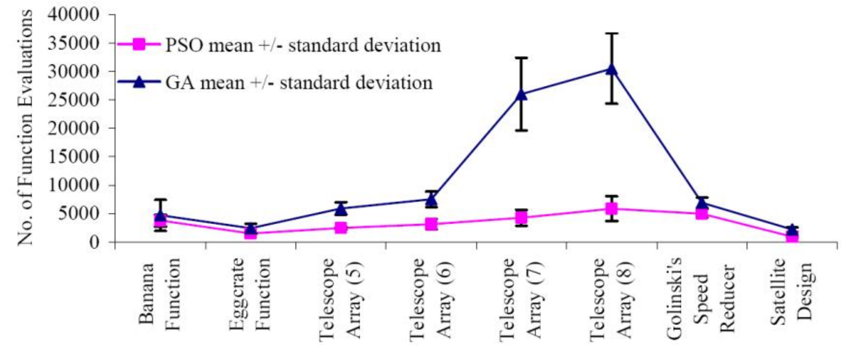
Comparison PSO versus GA

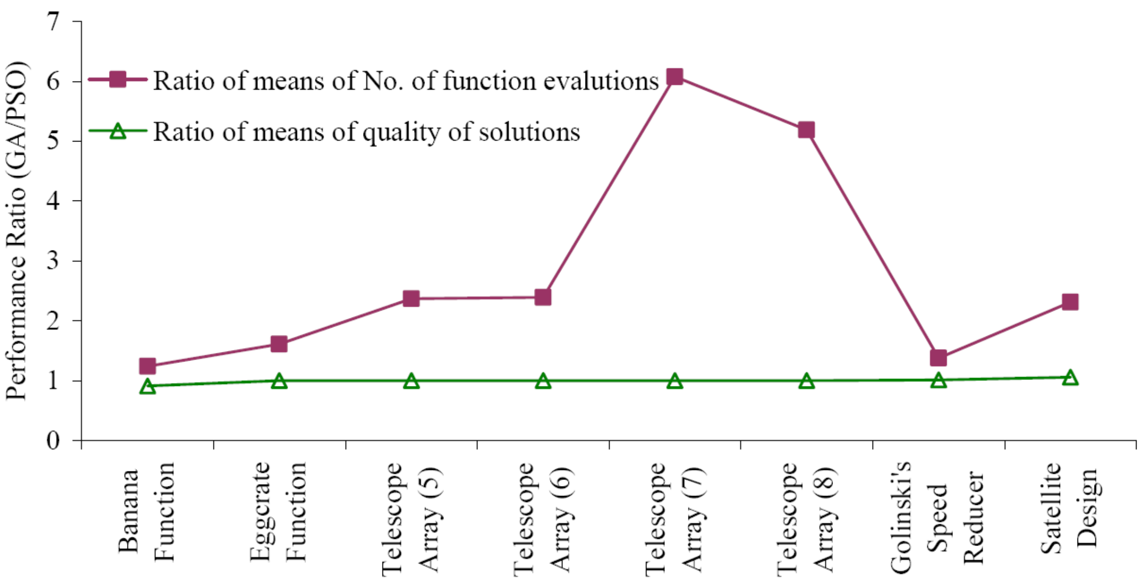
16.888
ESD.77

Effectiveness



Efficiency





- Implemented both for 8 test problems of increasing complexity
- PSO and GA deliver nearly equivalent solution quality
- PSO is generally more efficient requiring between 1-6 times fewer function evaluations
- PSO main advantage for unconstrained, non-linear problems with continuous d.v.

Hassan R., Cohanim B., de Weck O.L., Venter G., “A Comparison of Particle Swarm Optimization and the Genetic Algorithm”, AIAA-2005-1897, 1st AIAA Multidisciplinary Design Optimization Specialist Conference, Austin, Texas, April 18-21, 2005

PSO References

- Kennedy, J. and Eberhart, R., “Particle Swarm Optimization,” Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia 1995, pp. 1942-1948.
- Venter, G. and Sobieski, J., “Particle Swarm Optimization,” AIAA 2002-1235, 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Denver, CO., April 2002.
- Kennedy, J. and Eberhart, R., *Swarm Intelligence*, Academic Press, 1st ed., San Diego, CA, 2001.
- Hassan R., Cohanim B., de Weck O.L., Venter G., “A Comparison of Particle Swarm Optimization and the Genetic Algorithm”, AIAA-2005-1897, 1st AIAA Multidisciplinary Design Optimization Specialist Conference, Austin, Texas, April 18-21, 2005

Tabu Search

Tabu Search (TS)

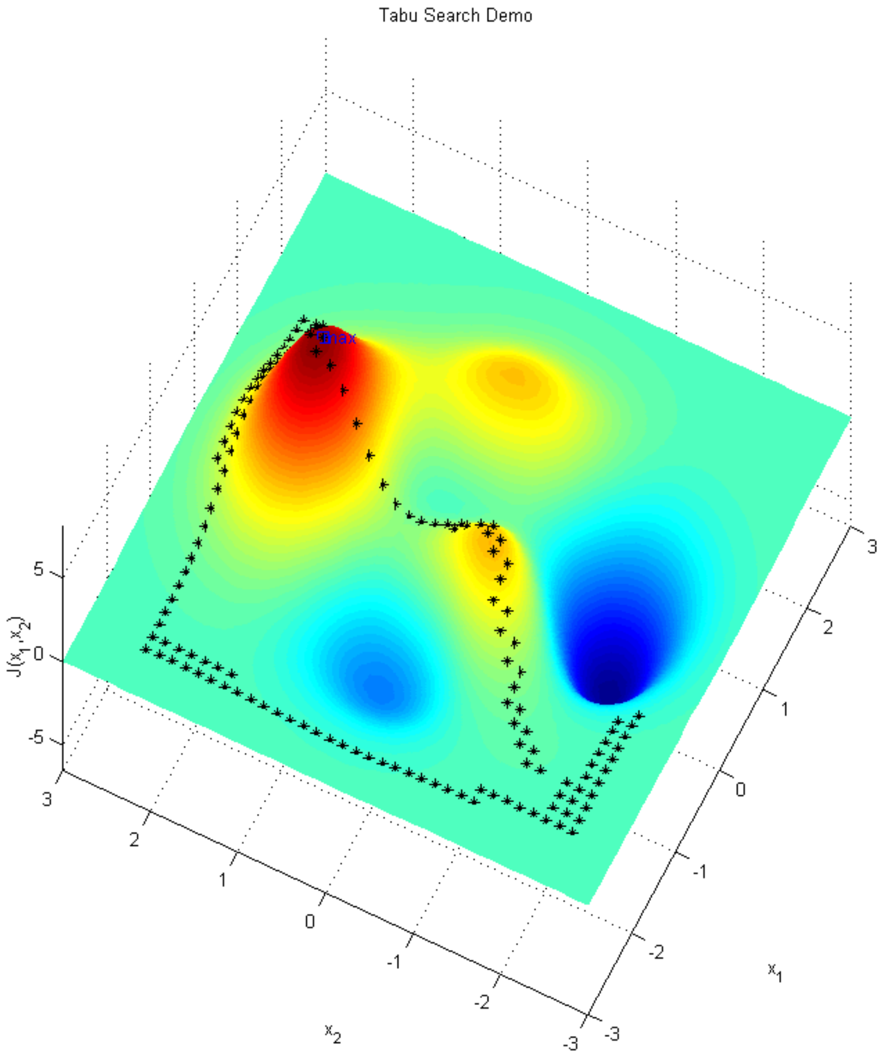
- Attributed to Glover (1990)
- Search by avoiding points in the design space that were previously visited ("tabu") – keep memory !
- Accept a new "poorer" solution if it avoids a solution that was already investigated – maximize new information
- Intent: Avoid local minima
- Record all previous moves 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 M. Laguna. *Tabu Search*. Kluwer, Norwell, MA Glover, F. and M. Laguna. (1997).

Tabu Search (pseudo code)


```
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
(1)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



Algorithm Selection

Selection of Algorithms

First characterize the design optimization problem:

1. Linearity and smoothness of objective function $J(\mathbf{x})$ and constraints $\mathbf{g}(x)$, $\mathbf{h}(x)$
2. Type of design variables **x** (real, integer,…)
3. Number of design variables *n*
4. Expense of evaluating $J(\mathbf{x})$, $\mathbf{g}(x)$, $\mathbf{h}(x)$
    1. [CPU time, Flops]
5. Expense of evaluating gradient of $J(\mathbf{x})$
6. Number of objectives, z

Nonlinearity

Crumpled Paper Analogy to Show Nonlinearity:

- **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 *Linear* Function which can be represented as**

$$y = c_0 + c_1 x_1 + c_2 x_2$$

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

$$y = c_0 + c_1x_1 + c_2x_2 + c_3x_1^2 + c_4x_2^2 + c_5x_1x_2$$

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

(Rough) Algorithm Selection Matrix

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

Golf Club Analogy

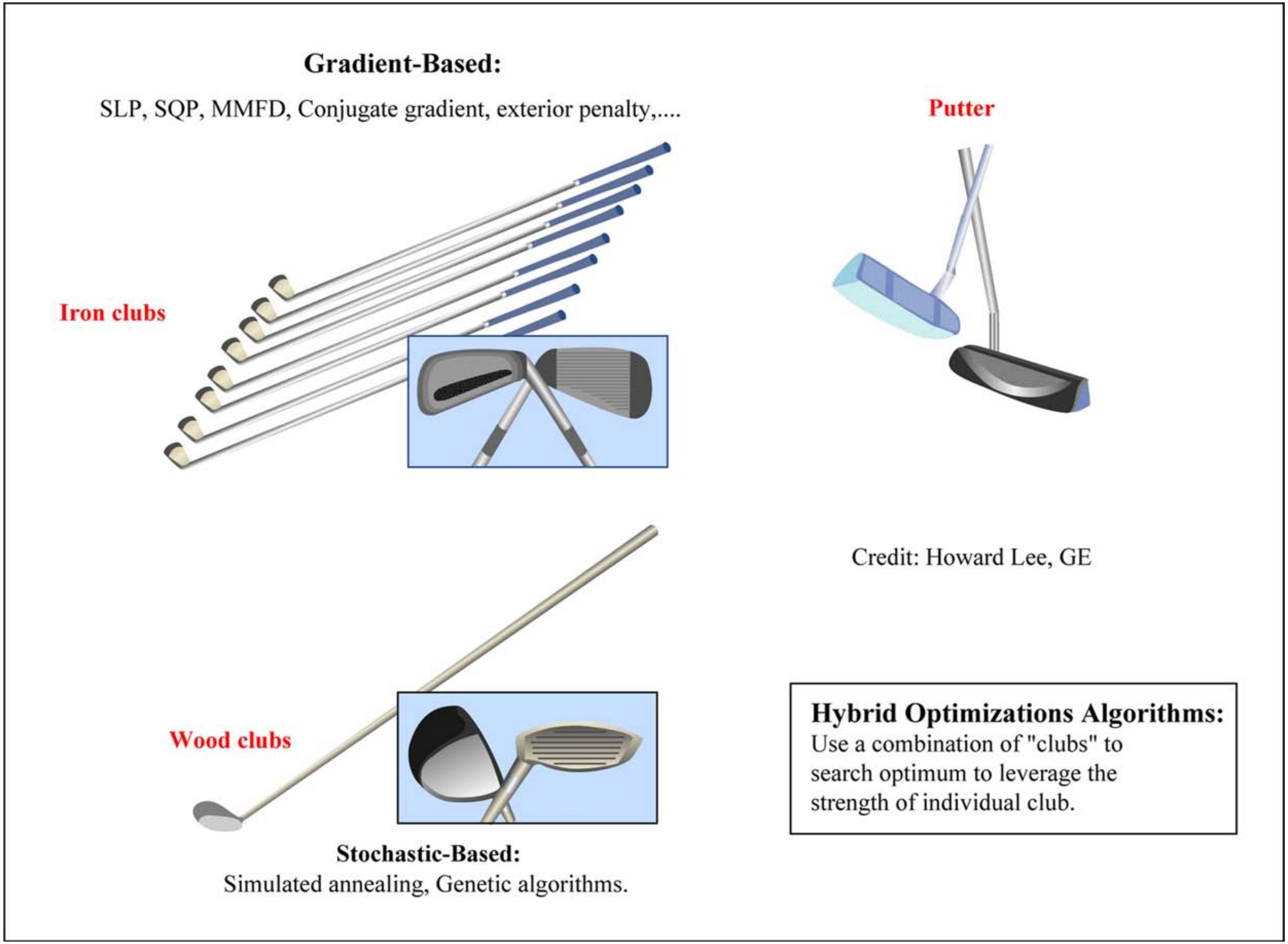
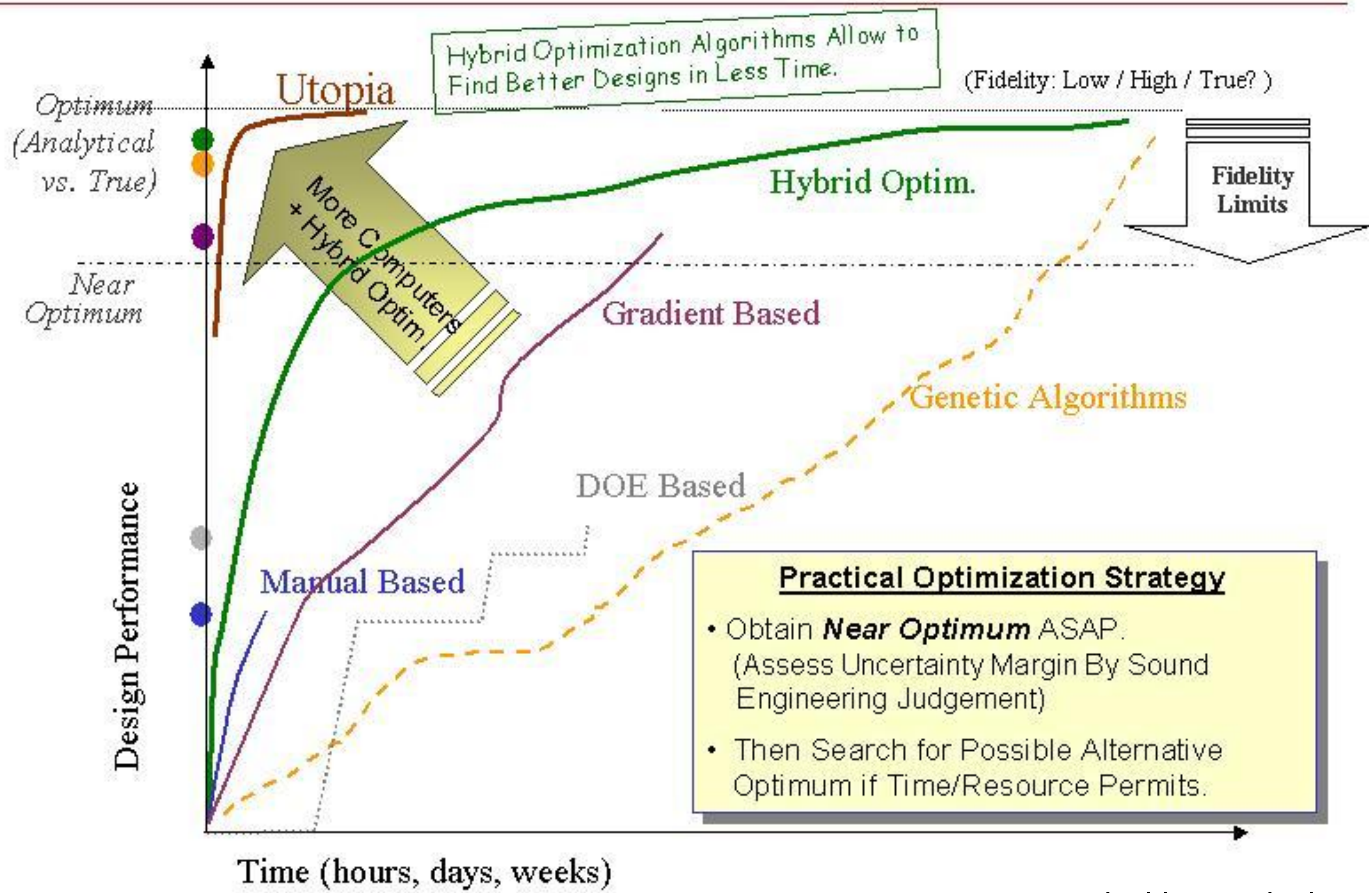


Image by MIT OpenCourseWare.

Practical Optimization Strategy



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

used with permission

Summary

- Gradient Search Techniques
  - Efficient, repeatable, use gradient information
  - Can test solution via KKT (Optimality) conditions
  - Well suited for nonlinear problems with continuous variables
  - Can easily get trapped at local optima

- (Meta-) 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
  - Two sub-approaches
    - Use the classical “pure” algorithms in sequence
    - Hybridize algorithms to include elements of memory, swarm behavior, mixing etc ….. Ongoing research

ESD.77 / 16.888 Multidisciplinary System Design Optimization
Spring 2010