Particle Swarm Optimization (PSO) Algorithm and Its Application in Engineering Design Optimization

By

Sushanta Kumar Mandal
Research Scholar
School of Information Technology
Indian Institute of Technology
Kharagpur
September 9, 2005

Rev. 2.01 on December 7,2012
by M. Miccio

Outline

- Introduction to Optimization
- Optimization Procedure
- Different Optimization Algorithms
- Different Global Optimization Algorithms
- Particle Swarm Optimization (PSO) Algorithm
- Application of PSO in Design Optimization Problems

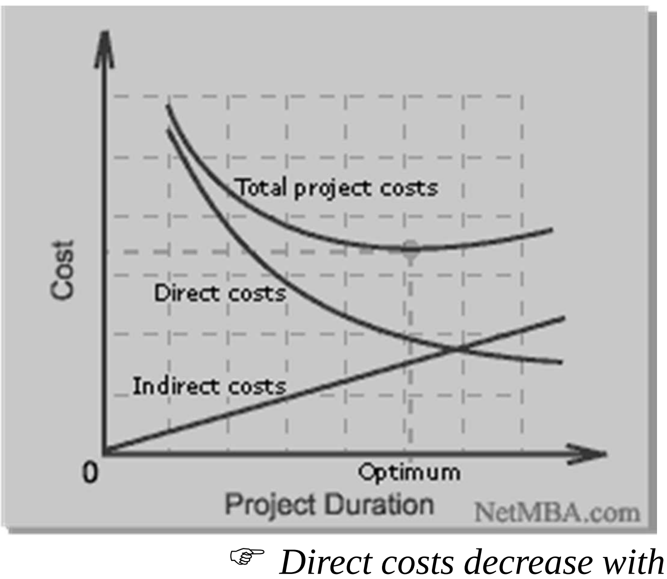
Optimization

As ageless

as time

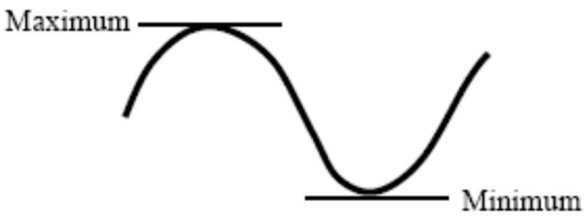
A first example

Total project costs:
optimum conditions at the intersection of two mechanisms



- *Direct costs decrease with Project duration*
- *Indirect costs increase with Project duration*
- *Total costs pass through a minimum*

A second example: Calculus



Maximum and minimum of a continuous smooth function is reached at a point where its derivative vanishes.

Optimization is Everywhere

- The more we know about something, the more we see where optimization can be applied.
- Some personal decision making:

  - Finding fastest route to home or class

  - Optimal allocation of time for study or home work

  - Optimal budgeting

Components of Optimization Problems

✶ Objective Function: **A real-valued function of one or more variables that has to be minimized or maximized in optimization theory.**

– For example, in a manufacturing process, we might want to *maximize the profit* or *minimize the cost.*

– In fitting experimental data to a user-defined model, we might *minimize the total deviation* of observed data from predictions based on the model.

– In designing an inductor, we might want to *maximize the Quality Factor* and *minimize the area*.

Components of Optimization Problems

✶ Design Variables: **A set of one or more unknowns or variables which affect the value of the objective function.**

– In the manufacturing problem, the variables might include the *amounts of different resources used* or the *time spent on each activity*.

– In fitting-the-data problem, the unknowns are the *parameters* that define the model.

– In the inductor design problem, the variables used define the *layout geometry* of the panel.

Components of Optimization Problems

✶ Constraints: **A set of *mathematical relationships* that allow the unknowns to take on certain values, but exclude others.**

– For the manufacturing problem, it does not make sense to spend a negative amount of time on any activity, so we constrain all the "time" variables to be non-negative.

– In the inductor design problem, we would probably want to limit the *upper and lower value* of layout parameters and to target an inductance value within the tolerance level.

Components of Optimization Problems

✶ Feasible Solution: **A combination of values of the design variables is called a feasible solution, or a feasible point in the case of algebraic problems, when:**

– **it is a solution of the optimization problem**

– **it simultaneously satisfies all of the constraints, if constraints have been established**

✶ Feasible Region: **The set in the space of the design variables of all feasible points, which satisfy all the constraints of the problem at the same time.**

Goal of Optimization

Find values of the variables that minimize or maximize the objective function, while satisfying the constraints.

Are all these ingredients necessary?

✶ Almost all optimization problems have **objective function.**

ⓘ ***No objective function**:*

In some cases (for example, design of integrated circuit layouts), the goal is to find a set of variables that satisfies the constraints of the model. The user does not particularly want to optimize anything so there is no reason to define an objective function. This type of problems is usually called a *feasibility problem.*

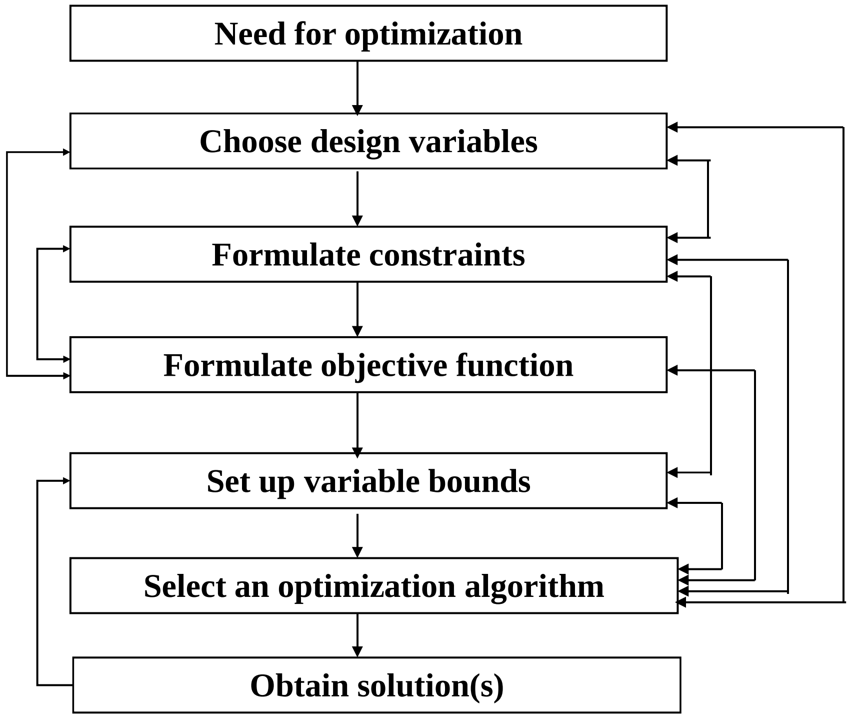
Are all these ingredients necessary?

- **Variables** are essential. If there are no variables, we cannot define the objective function and the problem constraints.
- **Constraints** are not essential. In fact, the field of unconstrained optimization is a large and important one for which a lot of algorithms and software are available. It's been argued that almost all problems *really do* have constraints.

What We Need for Optimization

- Models: Modeling is the process of identifying objective function, variables and constraints. The goal of models is “insight”, not the numbers. A good mathematical model of the optimization problem is needed.
- Algorithms: Typically, an interesting model is too complicated to be able to solve in with paper and pencil. An effective and reliable numerical algorithm is needed to solve the problem. There is no universal optimization algorithm. Algorithm should have ***robustness*** (good performance for a wide class of problems), ***efficiency*** (not too much computer time) and ***accuracy*** (can identify the error)

Flowchart of Optimal Design Procedure



Mathematical Formulation of Optimization Problems

minimize the objective function

$$\min f(x), x = \left(x_1, x_2, \ldots\ldots, x_n\right)$$

subject to constraints

$$c_i(x) \geq 0$$

$$c_j\left(x\right) = 0$$

Example

$$\min\left[\left(x_1 - 2\right)^2 + \left(x_2 - 1\right)^2\right]$$

$$\text{subject}: x_1^{\;2} - x_2^{\;2} \leq 0$$

$$x_1 + x_2 \leq 2$$

Type of Constraints

✶ Inequality constraints: Ex.: $x_1^2 - x_2^2 \le 0$

✶ Equality constraints: Ex.: $x_1 = 2$

Variable Bounds

✶ Maximum and minimum bounds may apply on each design variable.

✶ Without variable bounds the constraints completely surround the feasible region.

✶ Variable bounds are used to confine the search algorithm within these bounds.

✶ Ex: $x_i^{(L)} \le x_i \le x_i^{(U)}$

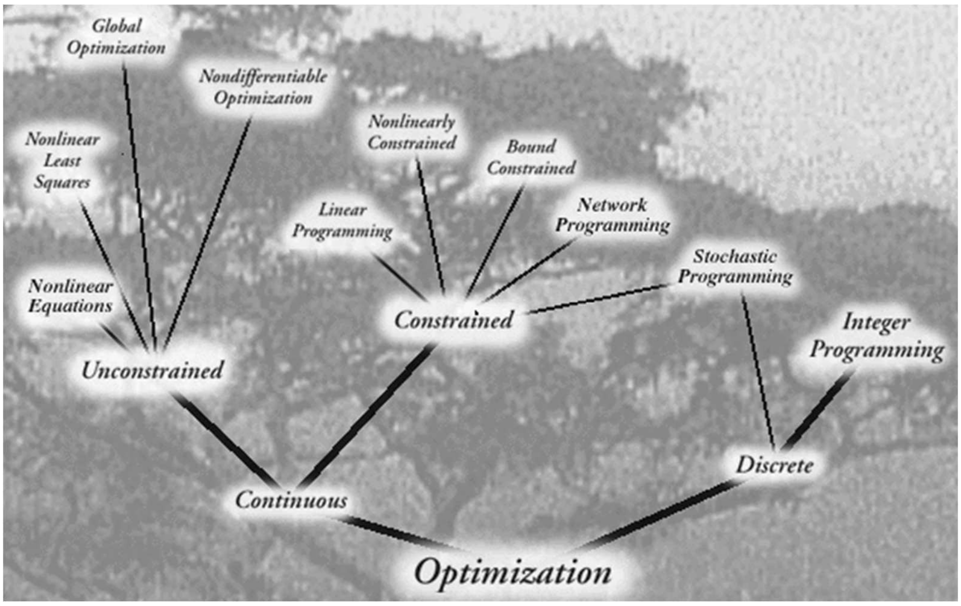
Classification of Optimization Problems

- ✶ Single variable vs Multi-variable
- ✶ Constrained vs Unconstrained
- ✶ Single objective vs Multi-objective
- ✶ Linear vs Non-linear
- ✶ Continuous variables vs Discrete variables

Classification of Optimization Methods

Method	Variables	Objective	Constraints	Comments
Linear Programming	real	linear	linear	
Networks	real	linear	linear	
shortest route (Dijkstra's algorithm)	real	linear	linear	can cast as network
minimum spanning tree	real	linear	linear	can cast as network
max flow and min cut (Ford/Fulkerson algorithm)	real	linear	linear	can cast as network
PERT and resource levelling				various related optimization problems: resource levelling, crashing
branch and bound	discrete			Flexible framework. Good bounding function is vital.
binary integer programming (Balas algorithm)	binary	linear	linear	special B&B
mixed-integer linear programming (Dakin's algorithm)	real and integer/binary	linear	linear	special B&B
simulated annealing				
genetic algorithms	discrete (normally)			
dynamic programming	discrete			Knapsack problems best solved this way
gradient methods	real	nonlinear	none	
Hooke and Jeeves pattern search	real	nonlinear	none	
method of Lagrange	real	nonlinear	nonlinear equalities	
GRG methods	real	nonlinear	nonlinear	linearly approximates nonlinear constraints
quadratic programming	real	quadratic	linear	
sequential linear programming	real	nonlinear	linear	
sequential quadratic programming	real	nonlinear		

Classifications of Optimization Methods



Local and Global Optimizers

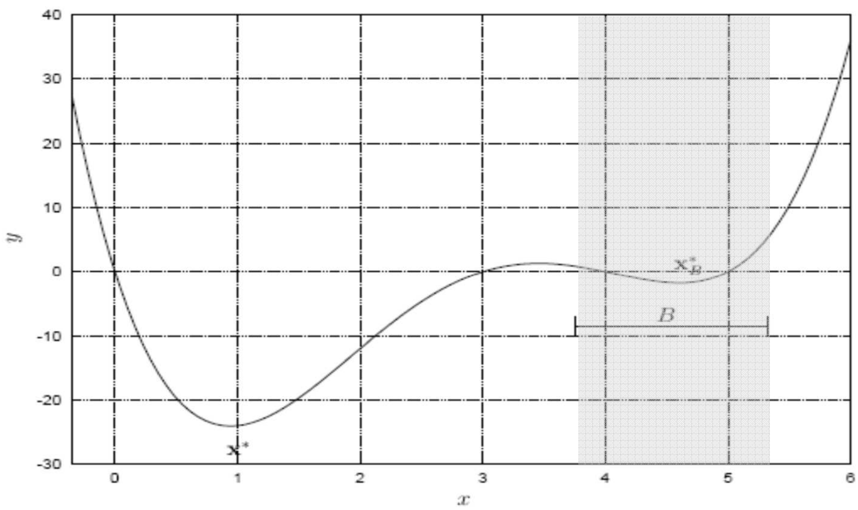


Figure 2.1: The function $f(x) = x^4 - 12x^3 + 47x^2 - 60x$, indicating the global minimiser $\mathbf{x}^*$, as well as a local minimiser $\mathbf{x}^*_B$.

Local and Global Optimizers

- A local minimizer, x_B^*, of the region B, is defined so that: $f(x_B^*) \le f(x), \forall x \in B$.
- Ex: Gradient based search methods, Newton-Rapson algorithms, Steepest Descent, Conjugate-Gradient algorithms, Levenberg-Marquardt algorithm etc.
- Shortcomings:

1) One requires an initial guess to start with.
2) Convergence to an optimal solution depends on the chosen initial guess.
3) Most algorithms tend to get stuck to a sub-optimal solution.
4) An algorithm efficient in solving one optimization problem may not be efficient in solving another one.
5) These are useful over a relatively narrow range.

Local and Global Optimizers

- The global optimizer, x^* , is defined so that $f(x^*) \le f(x), \forall x \in S$ where S is the search space.
- Ex.: Simulated Annealing algorithm, Genetic Algorithm, Ant Colony, Geometric Programming, Particle Swarm Optimization.