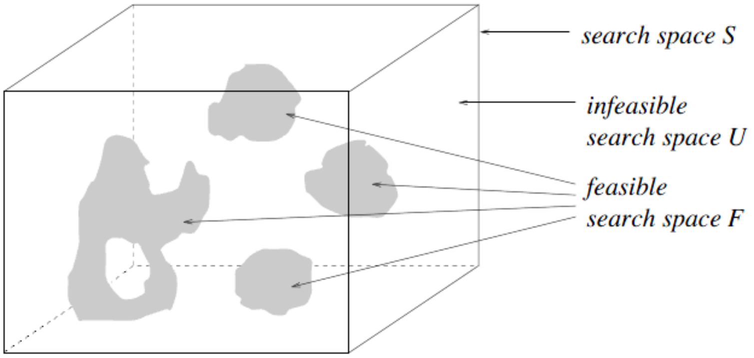
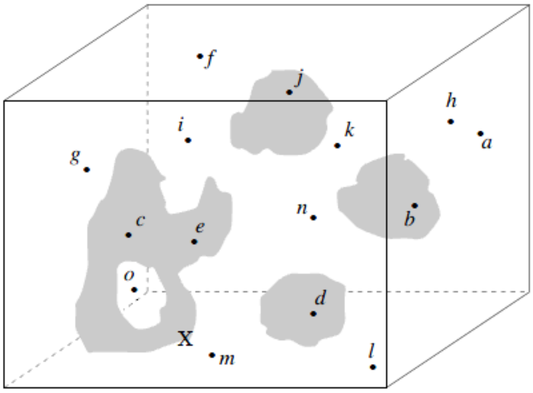
Computational Intelligence

Unit # 13

Constrained Optimization



Constrained Optimization (Cont'd)



Issues

- The presence of feasible and unfeasible individuals in the population influences other parts of the evolutionary algorithm; for example, should the elitist selection method consider a possibility of preserving the best feasible individual, or just the best individual overall?
- The problem of how to evaluate individuals in the population is also far from trivial.

Constraints Handling

- Several papers have been written on the handling of constraints in Evolutionary Algorithms but the “Do Not Kill Unfeasible Individuals” by Zbigniew Michalewicz is an excellent read.
- The 7 approaches (A-G) discussed in the next slides are taken from Michalewicz paper.

Question # A
Comparison of Feasible Solutions

- How should two feasible individuals be compared, e.g., ‘c’ and ‘j’ from Figure (on slide 4)? In other words, how should the evaluation function be designed?

Question A (Cont'd)

- For numerical optimization problems, the evaluation function f for feasible solutions is typically available/given.
- However, for some problems the selection of evaluation function might be far from trivial.
- For example, in building an evolutionary system to control a mobile robot there is a need to evaluate robot's paths. It is unclear, which particular path for a robot should have better evaluation, when we take into account, for example, their total distance, clearance from obstacles, and smoothness.
- For such problems there is a need for some heuristic measures to be incorporated into the evaluation function.

Question # B
Comparison of Unfeasible Solutions

- How should two unfeasible individuals be compared, e.g., 'a' and 'n' in Figure on Slide 4?
- This is a quite hard problem. We can avoid it altogether by rejecting unfeasible solutions (Question # C) or putting a penalty on unfeasible solutions (Question # F).

Question # B (Cont'd)

- Is it possible for an unfeasible solution to have a better fitness than a feasible solution?
- The issue of establishing a relationship between evaluation functions for feasible and unfeasible individuals is one of the most challenging problems to resolve while applying an evolutionary algorithm to a particular problem.

Question # C
Elimination of Unfeasible Solutions

- Should we consider unfeasible individuals harmful and eliminate them from the population?
- This "death penalty" heuristic is a popular option in many evolutionary techniques (e.g., evolution strategies).
- The method of eliminating unfeasible solutions from a population may work reasonably well when the feasible search space is convex and it constitutes a reasonable part of the whole search space.

Question # C (Cont'd)

- Otherwise such an approach has serious limitations.
- For example, for many search problems where the initial population consists of unfeasible individuals only, it might be essential to improve them (as opposed to rejecting them).
- Moreover, quite often the system can reach the optimum solution easier if it is possible to "cross" an unfeasible region.

Question # D
Repairing Unfeasible Solutions

- Should we 'repair' unfeasible solutions by moving them into the closest point of the feasible space (e.g., the repaired version of 'm' might be the optimum 'X' (Figure on Slide 4)
- Repair algorithms enjoy a particular popularity in the evolutionary computation community: for many combinatorial optimization problems (e.g., traveling salesman problem, knapsack problem, etc.) it is relatively easy to 'repair' an unfeasible individual.

Question # E
Replacement of Unfeasible Solutions

- If we repair unfeasible individuals, should we replace an unfeasible individual by its repaired version in the population or rather should we use a repair procedure for evaluation purpose only?

Question # F
Penalization of Unfeasible Solutions

- Since our aim is to find a feasible optimum solution, should we choose to penalize unfeasible individuals?
- This is the most common approach in the genetic algorithms community.
- The major question is, how should such a penalty function Q(p) be designed? The intuition is simple: the penalty should be kept as low as possible, just above the limit below which infeasible solutions are optimal.

Question # F (Cont'd)

- Penalties which are functions of the distance from feasibility are better performers than those which are merely functions of the number of violated constraints.

Sajjad Haider Spring 2012 15

Question # G Initialization

- Should we start with initial population of feasible individuals and maintain the feasibility of offspring by using specialized operators?

Sajjad Haider Spring 2012 16

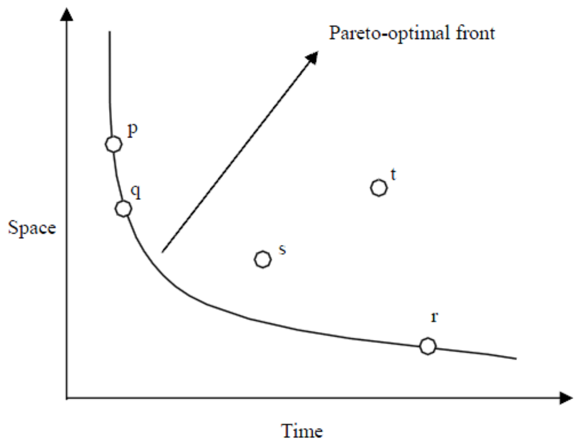
Multi-Criteria Optimization

- In a multi-criterion optimization problem, there is more than one objective function, each of which may have a different individual optimal solution.
- If there is a sufficient difference in the optimal solutions corresponding to different objectives then we say that the objective functions are conflicting to each other.
- Multi-criterion optimization with such conflicting objective functions give rise to a set of optimal solutions, instead of one optimal solution.

Pareto Optimal Solutions

- The reason for the optimality of many solutions is that no one can be considered to be better than any other with respect to all other objective functions.
- These optimal solutions have a special name called Pareto-optimal solutions following the name of an economist Vilfredo Pareto.

Pareto-Optimal Front



Dominant and Non-dominant Solutions

Let us consider a problem having m objectives (say $f_i, i = 1,2,3,\ldots\ldots, m$ and $m > 1$). Any two solutions $u^{(1)}$ and $u^{(2)}$ (having 't' decision variables each) can have one of two possibilities-one dominates the other or none dominates the other. A solution $u^{(1)}$ is said to *dominate* the other solution $u^{(2)}$, if the following conditions are true:

1. The solution $u^{(1)}$ is no worse (say the operator $\prec$ denotes worse and $\succ$ denotes better) than $u^{(2)}$ in all objectives, or $f_i(u^{(1)}) \geq f_i(u^{(2)}), \forall i = 1,2,3,\ldots, m$.
2. The solution $u^{(1)}$ is strictly better than $u^{(2)}$ in at least one objective, or $f_i(u^{(1)}) \succ f_i(u^{(2)})$ for at least one, $i \in \{1,2,3,\ldots, m\}$.

Two Important Goals

- There are primarily two goals that a multi-criterion optimization algorithm must try to achieve:
  - Guide the search towards the global Pareto-optimal region, and
  - Maintain population diversity in the Pareto-optimal front.

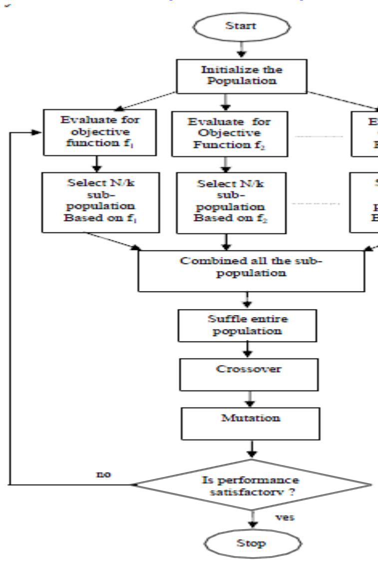
Weighted Sum Approach

- Generate the initial population randomly.
- Compute the value of the m objectives for each individual in the population.
  - Place dominant solution separately (external list)
  - The remaining solutions are stored in 'current'.
- Apply crossover/mutation to solutions in 'current'.
- Merge the external list with the new solutions and repeat the process.

Vector Evaluation Genetic Algorithm (VEGA)

- Generate the initial population randomly.
- Compute the value of the m objectives for each individual in the population.
  - Select k best solutions based on each objective.
  - Combine the k-best solutions (for each objective).
- Apply crossover/mutation to solutions in 'the combined list.
- Repeat the process.

Vector Evaluation Genetic Algorithm (VEGA)



Strength Pareto Approach

- Initialize the population of size n.
- Determine all the dominant solutions from current population and store them in a separate population called EXTERNAL.
- Assign fitness to individuals according to how many solutions it dominates.
- After assigning the fitness to all the individuals, select n individuals for next generation.
- Apply crossover and mutation to get the new population of size n.
- Update EXTERNAL population.