CONSTRAINT METHOD in objective space for a Min-Min Problem with Convex Feasible Space

**Actual Tradeoff = Pareto Front**

**Constraint Method:**
Minimize $F_2$
Subject to $F_1 \leq F_1^*$

Example of 3 values of $F_1^*$ required for 3 tradeoff points (requires 3 optimization solutions)

Constraint Method **will** work on non-convex tradeoff curves (next slide)

Handout 11-7-11
Constraint Method: Non-Convexity

Example

Here 4 points are picked. Explain what the value of the point from the Pareto Front will be for each of the $\varepsilon_k$ and why. Especially think about $\varepsilon_1$ and $\varepsilon_4$. 
Summary of Problems and Overcoming them with a GA Approach to MO

- The objective feasible space is **convex** if you can draw a line between any two points in the space and still remain in the space.

- If the objective feasible space is not convex, a weighted constrained optimization of the two functions can miss part of the Pareto front so repeated solutions with different weights will not give all the solutions.

- The Genetic Algorithm method (I will describe) will do a better job than a weighted nonlinear programming method even if the functions each have only one local minimum (i.e. they are not multi-modal functions).

- You probably don’t know in advance if you objective space is convex (unless you can do some mathematical analysis to show that it is).

- Hence genetic algorithms for multi objective problems are a preferred method for a wider range of functions than genetic algorithms for single objective functions.

- Multi Objective Optimization is a very important application for
Non Dominated Sorting Genetic Algorithm: NSGA II

• K. Deb and his students developed an elitist non-dominated sorting GA called NSGA-II in 2000.

• The algorithm is available as public software and is widely used.

• The algorithm has several important features
  - It divides the population into multiple fronts
  - It uses the front to determine fitness
  - It examines the distance between points close together on a front to determine fitness.
• One of the features is that it requires sorting the solutions into levels of dominance.
• This is done by first solving for the Pareto Front of non dominated solutions. (Level 1)
• Next the Pareto Front of solutions are removed from the population and the Pareto Front is calculated again (Level 2).
• Several more levels are also calculated.
Earlier we saw that 3 and 5 did not dominate each other.

Does 5 dominate 4 and 1 and 2? \textit{yes}\? 

Does 3 dominate 4 and 1 and 2? \textit{yes}\? 

Where is the Pareto Front?
Non Dominated Sorting Genetic Algorithm: NSGA II

- K. Deb and his students developed an elitist non-dominated sorting GA called NSGA-II in 2000.
- The algorithm is available as public software and is widely used.
- One of the features is that it requires sorting the solutions into levels of dominance.
- This is done by first solving for the Pareto Front of non-dominated solutions. (Level 1)
- Next the Pareto Front of solutions are removed from the population and the Pareto Front is calculated again (Level 2).
- Several more levels are also calculated, you can actually calculate the n-th front.
Figure 18 "Non dominated Sorting" into Levels

\( f_2 \) (minimize)

\( f_1 \) (maximize)

Level 3

Level 2

Level 1

(Pareto Front)
First step: Sort the population into “Fronts”

- Sort the population among solutions of “non-dominated fronts.
- The first “front” is the Pareto optimal solution, i.e. all the points on Front 1 are non-dominated by ALL points in the solution space.
- To generate the second front, remove all the points on the first front from the population. Now compute the Pareto Optimal solution for the remaining points. The points on the second Pareto curve are the “second” front.
- Continue this process to get the nth front. (Hence, remove all the points from fronts 1 to n-1 and find the Pareto front for the remaining points.)
Notation

• $P = \text{set of all fronts}$
• $P_i = i^{th} \text{ front}$

So

\[
P = \bigcup_{j=1}^{M} P_i
\]

Where $M = \text{total number of fronts}$.

Note that for all the members of $P_i$, there is no way to say that one member is better than another.
Example Problem

Figure 105 A two-objective search space and the corresponding Pareto-optimal front.
4 Pareto Fronts after Sorting

Figure 114 Ten solutions of Figure 105 are classified into different non-domination fronts.
NSGAII Approach

• The idea behind the NSGA II approach is for the sake of diversity, there is a probability of selecting as a parent and individual from any of the fronts.
• However, the probability of being selected is less if the individual is located on a “worse” front.
• NSGA II also tries to avoid parents that are too similar to each other so it has a method for computing “crowding”.
• The likelihood of selecting a point for a parent is reduced if it is crowded.
• Why does this approach help diversity and help the algorithm in the long run? What is the alternative? Think about this as we discuss the algorithm.
User defined **Algorithm Parameters in Multi-Objective NSGA-II**

- In the NSGA algorithm, there are the normal algorithm parameters for population size, crossover location probability and mutation rate.
- In addition, for multi-objective optimization we also define the following parameters:
  \[ \varepsilon \] , which is the difference in base fitness between the two adjacent fronts
  \[ \sigma \text{share} \] , the maximum Euclidean distance between two points on the same “Front” to consider them in the “crowding” calculations.

The meaning of these terms will become clearer when we go through the method on following slides.
Overview of Fitness Calculation for NSGA algorithm

NSGA Fitness Assignment

Step 1 Choose sharing parameter $\sigma_{\text{share}}$ and a small positive number $\epsilon$ and initialize $F_{\text{min}} = N + \epsilon$. Set front counter $j = 1$.

Step 2 Classify population $P$ according to non-domination: $(P_1, P_2, \ldots, P_\rho) = \text{Sort}(P, \preceq)$. (objective function is relevant)

Step 3 For each $q \in P_j$

Step 3a Assign fitness $F_j^{(q)} = F_{\text{min}} - \epsilon$. (this assumes max-max problem)

Step 3b Calculate niche count $nc_q$ among solutions of $P_j$ only.

Step 3c Calculate shared fitness $F_j^{(q)} = F_j^{(q)} / nc_q$.

Step 4 $F_{\text{min}} = \min(F_j^{(q)} : q \in P_j)$ and set $j = j + 1$.

Step 5 If $j \leq \rho$, go to Step 3. Otherwise, the process is complete.

This fitness assignment procedure can be embedded in a single-objective GA. The proportionate selection operator must be used. However, any crossover and mutation operator can be employed. In the following, we illustrate the above fitness assignment procedure on the Min-Ex problem.
Assessing Crowding on a Front

- “Niche Count” is a term used in the following.
- Niche in ecology refers to specialized resource for which only a few species compete. It is not important to understand the ecological concept.
- In the algorithm the “Niche Count” $nc_i$ has a very specific equation and you do need to understand why it is reasonable to use this equation.
- $nc_i$ is a measure of the degree of crowding due to all the points around a point $x_i$.
- $nc_i$ depends on $Sh(d_{ik})$, which is a measure of the crowding between points $x_i$ and $x_k$. 
4 Pareto Fronts after Sorting (Repeat Slide)

This graph is from a multi-objective problem.

\[ \text{crowded, so fitness } = \frac{F_{\text{min}}}{n_{cq}}, \quad q \in \{2, 3, 4\} \]

\[ \Rightarrow \text{maximize classify uncrowded points, e.g., improve diversity} \]

Figure 114: Ten solutions of Figure 105 are classified into different non-domination fronts.
Method to reduce fitness within a $P_i$ for Crowding

- Normalized Euclidian Distance between two points (indexed by $i$ and $j$) on the $m^{th}$ front:

\[
d_{ij} = \sqrt{\frac{1}{p_m} \sum_{k=1}^{p_m} \left( \frac{x_k^{(i)} - x_k^{(j)}}{x_{\text{MAX}} - x_{\text{MIN}}} \right)^2}
\]

(5.21)

\[
S_h(d_{jj}) = \begin{cases} 
1 - \left( \frac{d_{jk}}{\text{share}_{jk}} \right)^2 & \text{if } d_{jk} \leq \text{share}_{jk} \\
0 & \text{otherwise}
\end{cases}
\]

(4.59)
Niche Count $n_{c_i}$ to Indicate Crowding

- For the $i$th individual of a front $P_k$, the niche count is

$$n_{c_i} = \sum_{j=1}^{\left| P_k \right|} S_h (d_{i,j})$$

(4.60)

- Where $j$ ranges over all the individuals in $P_k$
NSGA--overview

• The NSGA is then like a regular GA in that there is crossover and can be mutation.

• The major unique feature about NSGA is how the fitness is calculated.
  
  - The population is divided into fronts
  
  - The kth front $P_k$ has a base fitness $F_k$ and that fitness is reduced for individuals $q$ on $P_k$ to $F_k / nc_q$ for individuals $q$ that are within a distance $\sigma_{\text{share}}$ of other individuals on the front $P_k$.

  - The base fitness on the next front $P_{k+1}$ has a base fitness $F_{k+1}$ that is the minimum fitness of an individual on front $P_k$ minus $\varepsilon$ (so the fitness on $k+1$ front is always worse than the fitness on $k$ front).
NSGA—overview continued

- The base fitness on the next front $P_{k+1}$ has a base fitness $F_{k+1}$ that is the minimum fitness of an individual on front $P_k$ minus $\epsilon$ (so the fitness on front $P_{k+1}$ is always less than the fitness for all members of front $P_k$).

- The algorithm starts computing the fitness in the first front, then moves to the second front, etc. until it has computed fitness for all the fronts.

- The purpose of retaining the members in the population for the higher level fronts is to maintain “diversity” in the population so that later crossovers might generate offspring that are even more fit and closer to the Pareto Front than the current members.
A new application: Going Through A Numerical Example

Note you have two types of variables: the x’s are decision variable values and the f’s are function values. It is easy to start confusing them when doing multi-objective analysis.

Table 18  Fitness assignment under an NSGA.

<table>
<thead>
<tr>
<th>Solution</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$f_1$</th>
<th>$f_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.31</td>
<td>0.89</td>
<td>0.31</td>
<td>6.10</td>
</tr>
<tr>
<td>2</td>
<td>0.43</td>
<td>1.92</td>
<td>0.43</td>
<td>6.79</td>
</tr>
<tr>
<td>3</td>
<td>0.22</td>
<td>0.56</td>
<td>0.22</td>
<td>7.09</td>
</tr>
<tr>
<td>4</td>
<td>0.59</td>
<td>3.63</td>
<td>0.59</td>
<td>7.85</td>
</tr>
<tr>
<td>5</td>
<td>0.66</td>
<td>1.41</td>
<td>0.66</td>
<td>3.65</td>
</tr>
<tr>
<td>6</td>
<td>0.83</td>
<td>2.51</td>
<td>0.83</td>
<td>4.23</td>
</tr>
</tbody>
</table>
Algorithm parameters

• Step 1

Set \[ \Delta_{\text{shar}} = 0.15 \quad \epsilon = 0.22 \]

\[ N = \text{population size} = 6 \]

Hence, from Step 1 (for front \( P_1 \))

\[ F_{\text{min}} = N + \epsilon = 6 + 0.22 = 6.22 \]

Step 2: Sort the population into fronts as shown in Figure 115.
A new application:
Going Through A Numerical Example

Figure 115  The population is classified into three non-dominated fronts.

In Step 2. The above population (given in Table 18) of 6 is classified into the 3 fronts shown.