Multi-Objective Optimization: Principles and Algorithms

AE 6310: Optimization for the Design of Engineered Systems

Spring 2017

Dr. Glenn Lightsey

Lecture Notes Developed By Dr. Brian German

Georgia Institute of Technology

Daniel Guggenheim School of Aerospace Engineering

Multi-Attribute Problems

- So far, we have considered "single-objective" optimization problems in this class. These have the form:

$$\min y = f(\boldsymbol{x})$$
$$\boldsymbol{h}(\boldsymbol{x}) = \mathbf{0}$$
$$\boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}$$

Multi-Attribute Problems

- In many real-life problems, we are concerned with multiple attributes of a problem, for example:
  - y_1 = Cost
  - y_2 = Performance
  - y_3 = Manufacturability
  - y_4 = Safety
  - y_5 = Robustness
  - …

Multi-Attribute Problems

- Multi-attribute problems may be optimized by defining a “utility function” or “value function” that aggregates the multiple attributes we care about into a single function.
- If we can obtain such a function, it can serve as the objective function in one of the “single-objective” optimization techniques previously introduced in this class. Thus we can obtain a single “best” solution.

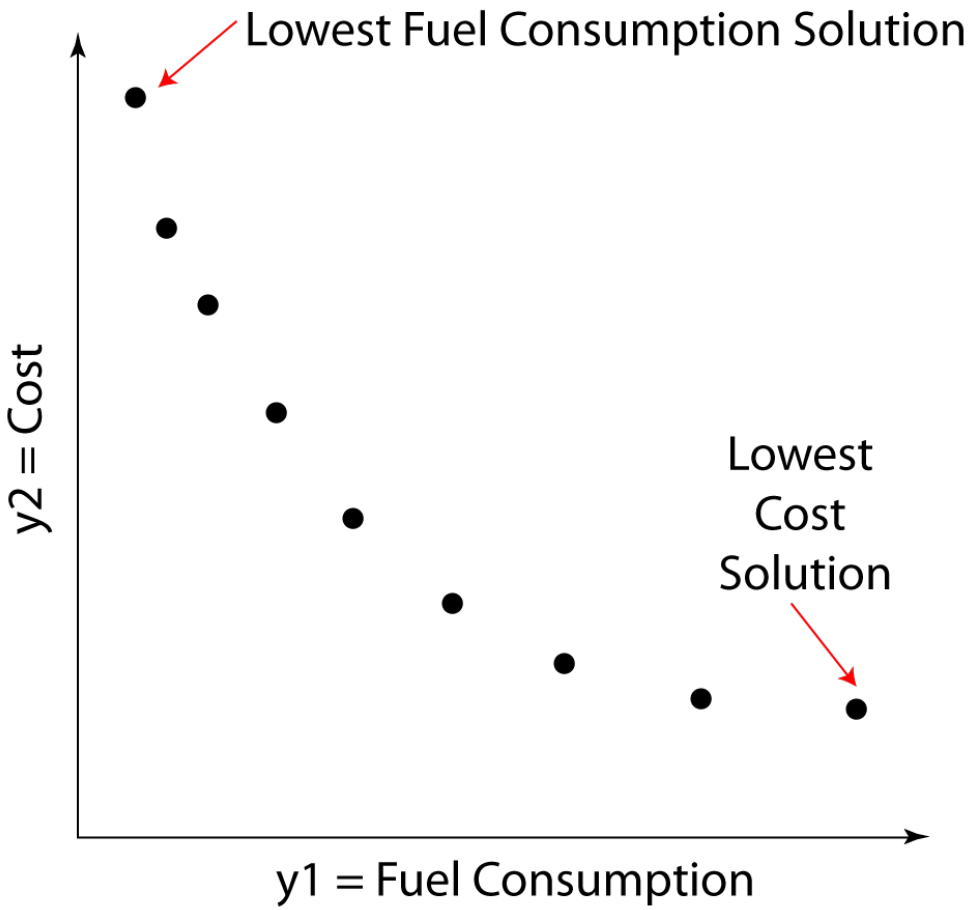
Multi-Attribute Problems

- E.g. $f = 0.5y_1 + 3/y_2 + 0.7y_3*y_4 + y_5$
- This approach may be called “a priori” multi-objective optimization.
- The difficulty of this approach is that *f* must be chosen very carefully
- For example, consider that each attribute likely has very different units. How do you combine “specific fuel consumption” with “mean time between failures” in a value function?

Multi-Attribute Problems

- This lecture is about "a posteriori" multi-objective optimization (which we will simply call "multi-objective optimization")
- In a posteriori multi-objective optimization, we do not combine the multiple attributes $y_1,\ldots,y_k$ into an aggregate objective function that has a single "best" or "optimal" solution
- Instead, the goal of a posteriori multi-objective optimization is to generate a set of solutions, each of which represents some relative optimality between the competing attributes.
- The "best" solution out of this set is then selected later as a separate exercise

Multi-Attribute Problems



Ordering Solutions

- In “single-objective” optimization, any two designs, x1 and x2 can be compared, by using the objective function, to determine which is preferred (or that both are equally preferable)

- The objective function therefore defines a “total order” on the feasible designs

Ordering Solutions

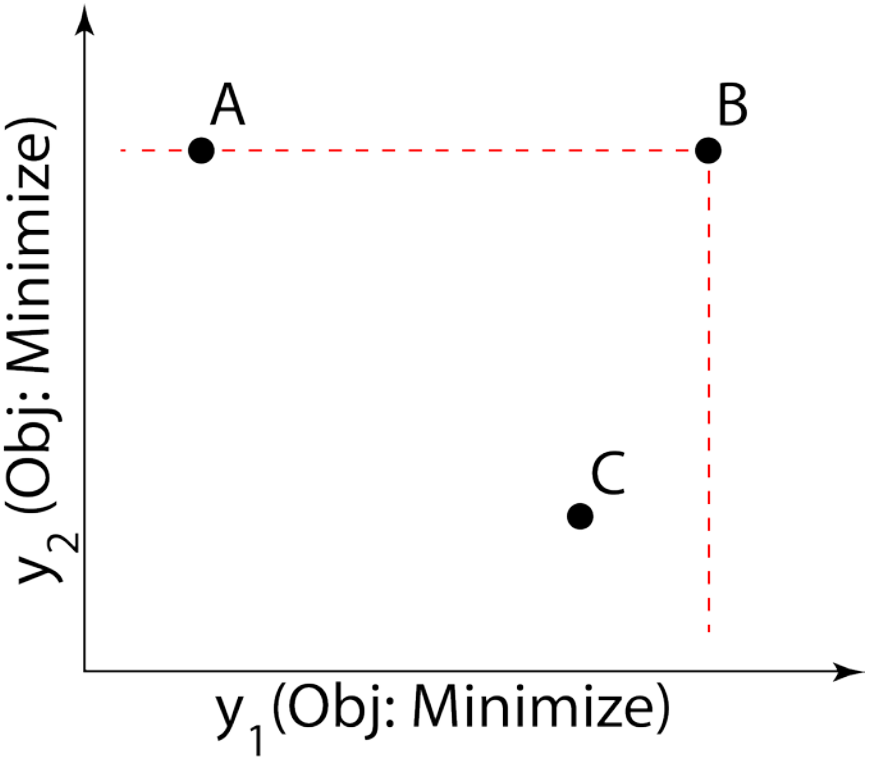
- Requirements for a total ordering
  - Antisymmetry: If $f(x_1) \leq f(x_2)$ and $f(x_2) \leq f(x_1)$ then $f(x_1) = f(x_2)$
  - Transitivity: If $f(x_1) \leq f(x_2)$ and $f(x_2) \leq f(x_3)$ then $f(x_1) \leq f(x_3)$
  - Totality: Either $f(x_1) \leq f(x_2)$ *or* $f(x_2) \leq f(x_1)$ (or both)
- The third property, totality, ensures that we can compare ANY two designs with the objective function to determine which design is better (or that both are equally good).

Ordering Solutions

- Multi-objective optimization is based on a partial ordering
- Partial orderings have the property that some elements (i.e. designs) may be compared, but others are incomparable.
- The goal of multi-objective optimization is to find a set of designs that are better than all designs to which they can be compared, but that are incomparable to each other.
- In principle, any partial ordering could be used for "multi-objective optimization." However, in practice, the Pareto Dominance partial ordering is always used.

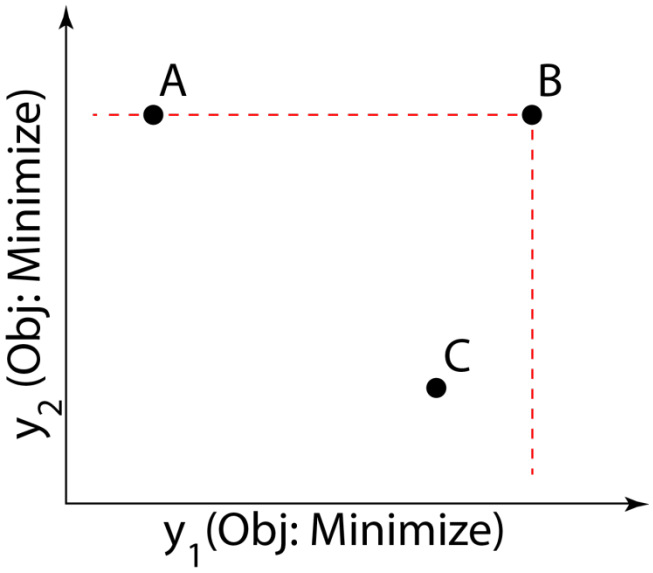
Pareto Dominance

- Which point(s) would you prefer?



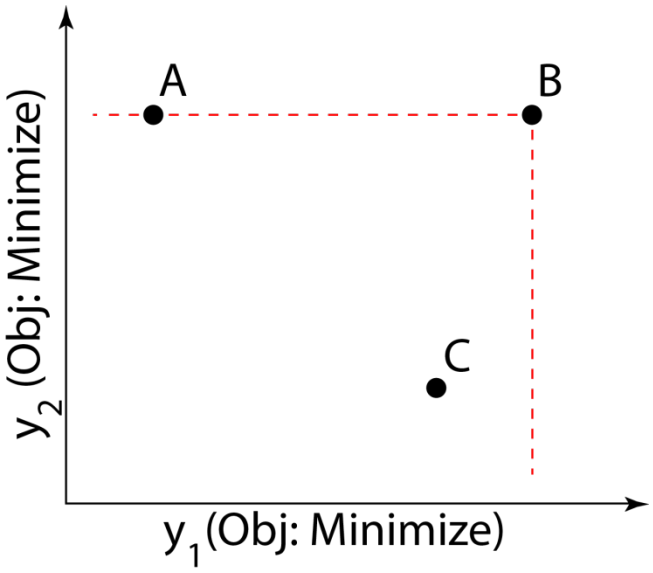
Pareto Dominance

- Compared to B:
  A has a better value of y_1
  C has a better value of y_1 & y_2
- If y_1 and y_2 are the only attributes we care about, there is no reason to choose B over either A or C
- A has a better value of y_1 than C, but C has a better value of y_2. Since we haven't specified any preference for *relative* optimality in y_1 vs y_2, A and C are incomparable.



Pareto Dominance

- We say that A "**weakly dominates**" B because it is better in some attributes and equal in others.
- We say that C "**strongly dominates**" B because it is better in all attributes.
- A and C are incomparable because A is better than C in some attributes but worse in others.
- Pareto Dominance is a *partial ordering* because some points are incomparable.

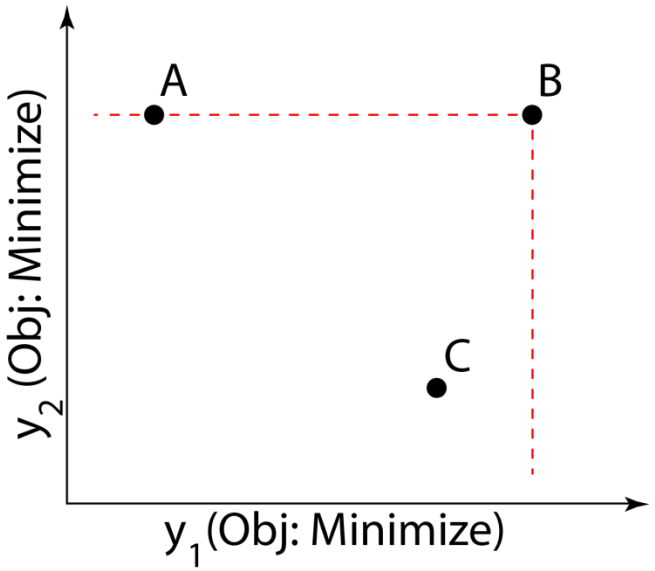


Definition of Pareto Dominance

- Given: a design problem with scalar-valued attributes y_1 ,… y_k
- Suppose that the objective is to minimize each attribute
- Let *A* and *B* denote two different vectors of attribute values (i.e. two different points in the objective space)
- Then A dominates B if and only if:
  - A is no worse than B in all attributes y_1,… y_k
  - A is better than B in at least one attribute (equivalently, A and B are not equal)
- Note that if A is better than B in some attributes but worse in others, neither A nor B dominates the other (incomparable).

Pareto Optimality

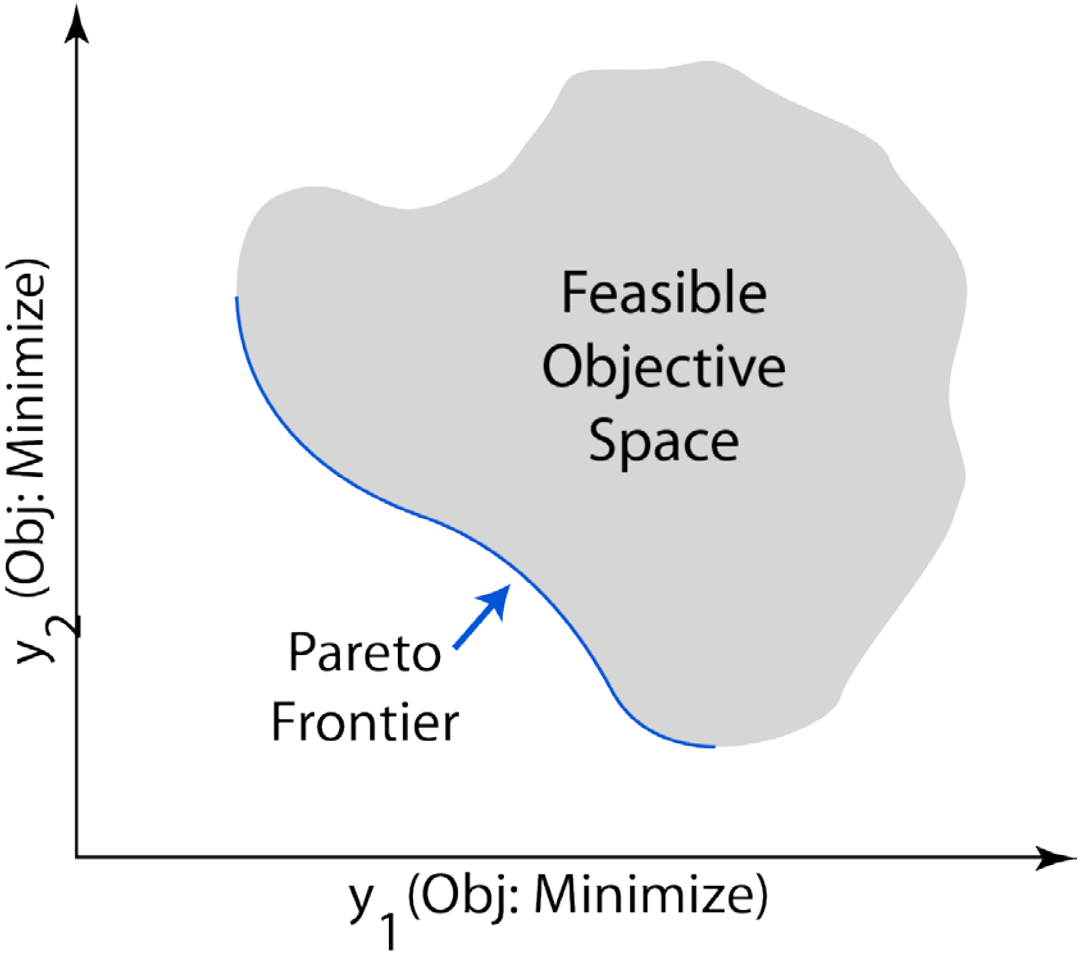
- A point in the feasible objective space that is not dominated by any other point in the feasible objective space is called "*non-dominated.*" The corresponding design is called "*Pareto optimal*" or "*Pareto efficient*"
- (Sometimes all three of these terms are used interchangeably)
- A point is non-dominated if there are no other points in its "**dominance cone**" (the region to the lower left of the dashed lines in the figure)



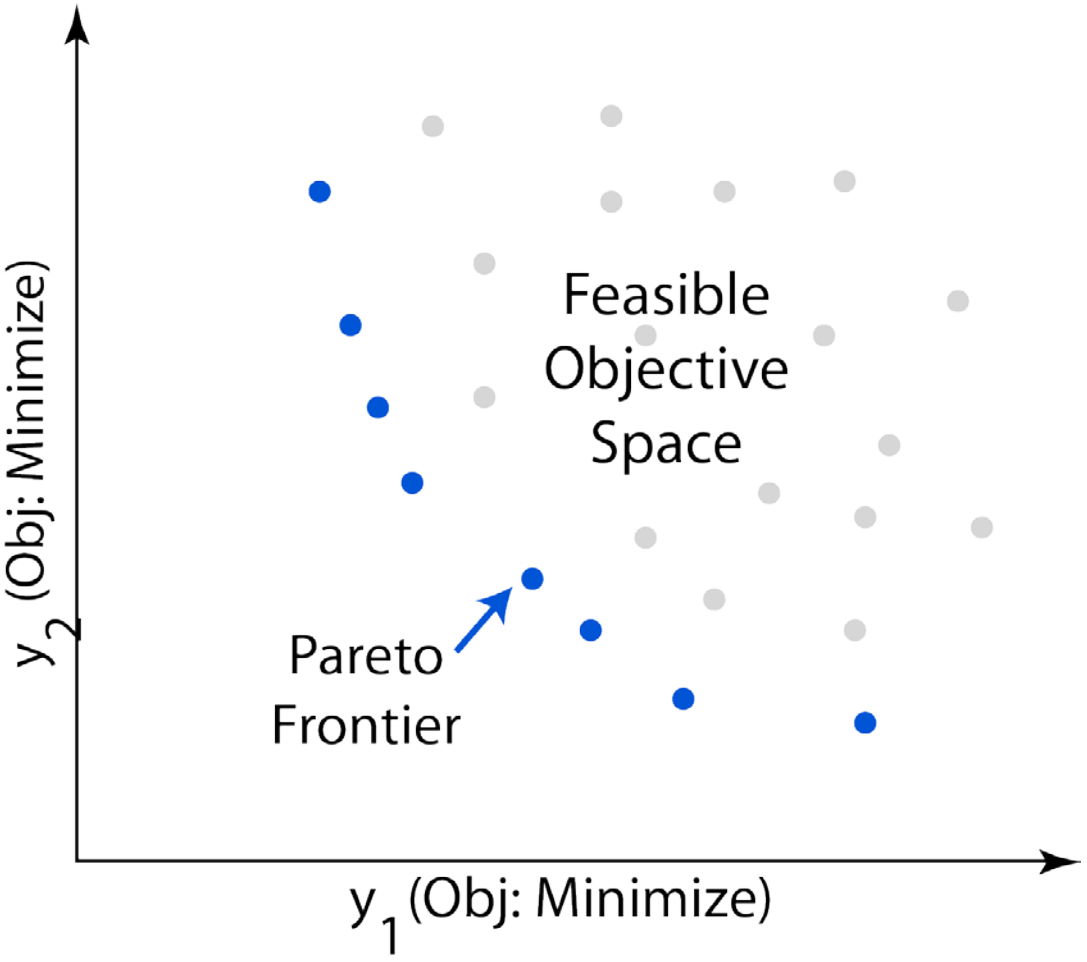
Pareto Frontiers

- The subset that consists of all non-dominated points in a given set is called the set's "**Pareto Frontier**."
- The Pareto Frontier is always a portion of the set's "boundary."
- Points on the Pareto Frontier have the property that no attribute can be improved by moving from point to point without worsening at least one other attribute, i.e. the Pareto frontier describes necessary **tradeoffs** in optimality

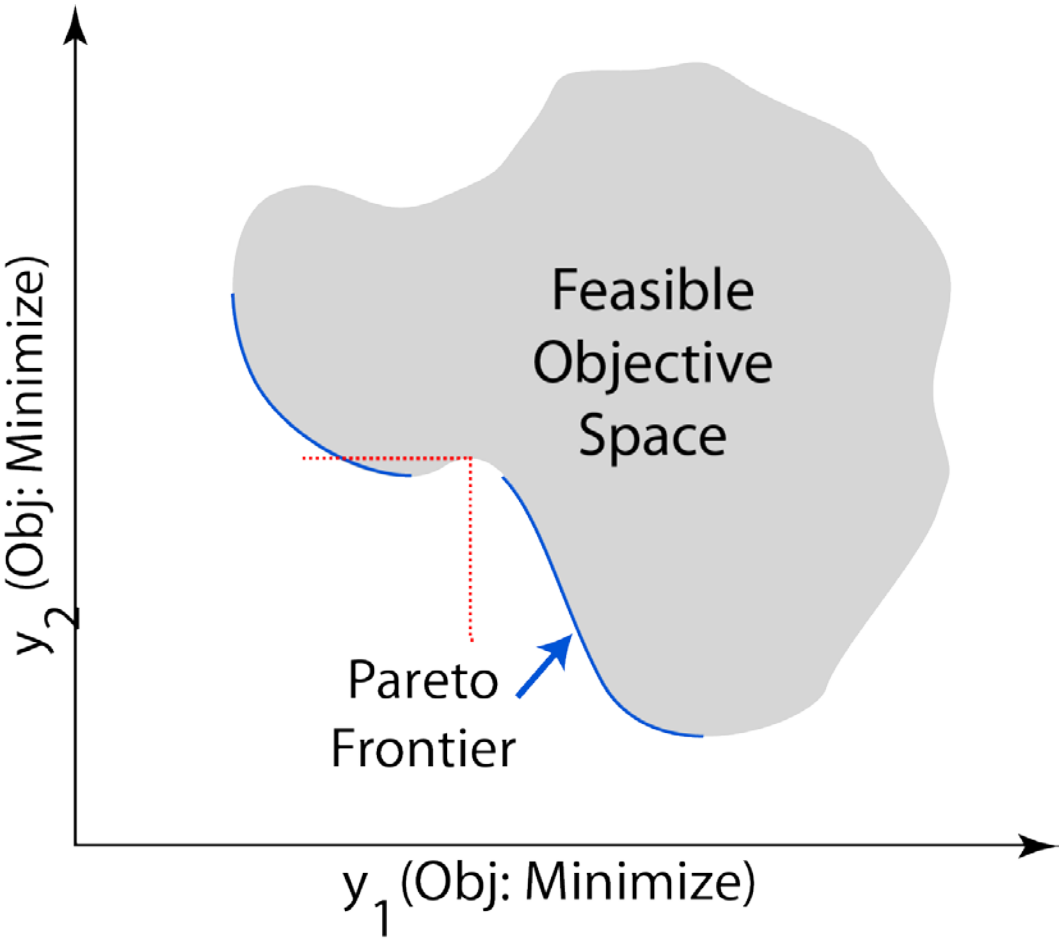
Pareto Frontier of a Continuous Set



Pareto Frontier of a Discrete Set



Pareto Frontier of a Concave Set



Theory Summary

- Multi-objective optimization is based on a *weaker* premise than single-objective optimization: partially ordering designs instead of totally ordering.
- Because of this, multi-objective optimization is unable to identify a unique "best" solution to a problem.
- Instead, a set of solutions is obtained, any of which might be considered the "best" for some defined preference for relative optimality among the competing attributes.

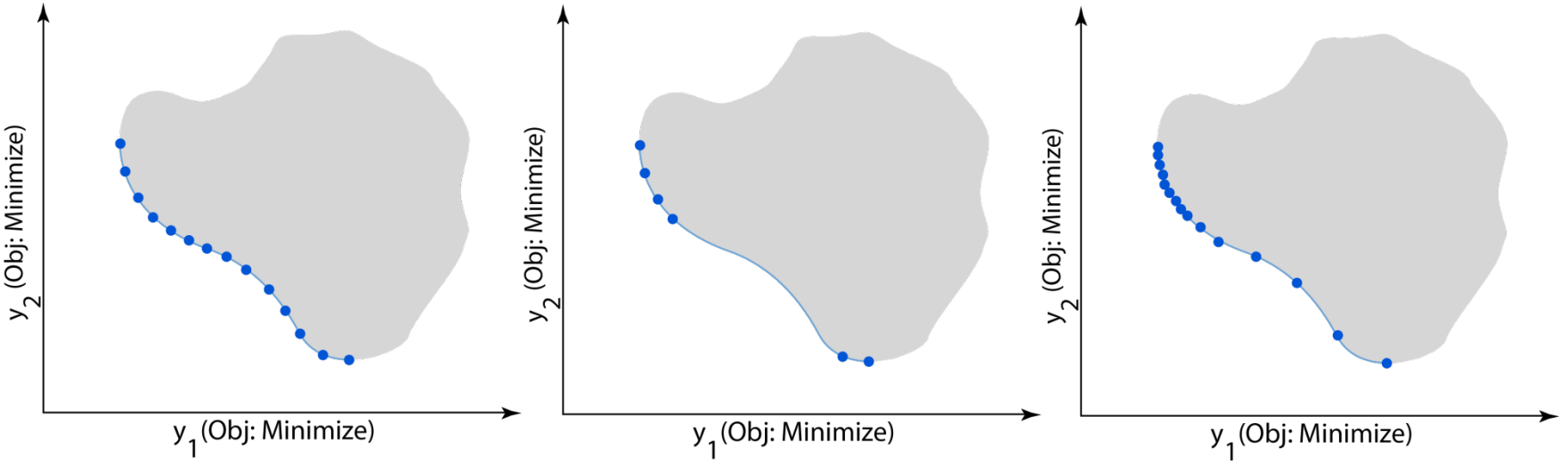
Theory Summary

- Multi-objective optimization is useful when a single objective function cannot be agreed upon *a priori*
- The set of non-dominated designs obtained from multi-objective optimization can be examined by the designer in order to:
  - Refine the optimization search space
  - Formulate a single-objective optimization problem
  - Select a best design
  - Understand tradeoffs in the objective space and their causes
  - …

Multi-Objective Optimization

- The goal of multi-objective optimization is to sample points from the Pareto frontier.
- For continuous optimization problems, there are an infinite number of non-dominated points. Our goal is to find a discrete sampling that is representative of the Pareto frontier.
- A "representative" sampling usually entails:
  - Points are sampled from all regions of the Pareto frontier
  - The sampled points are fairly evenly spaced along the Pareto frontier
  - Enough points are sampled to fulfill some purpose

Multi-Objective Optimization



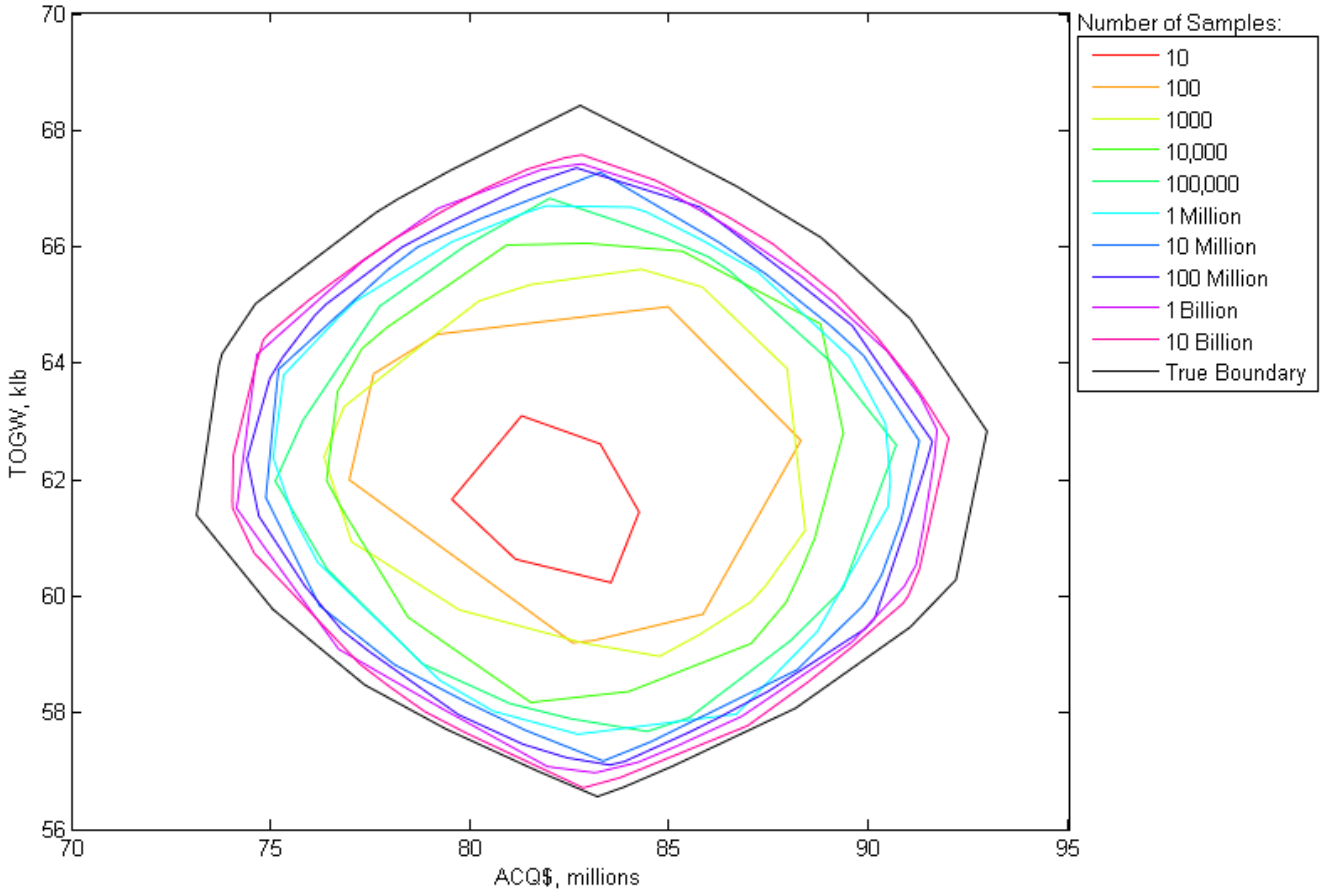
Good

Bad

Bad

Multi-Objective Optimization vs. Monte Carlo

- Extent of objective space explored by (uniform) random sampling of a problem with 2 attributes and 12 independent variables:



Multi-Objective Optimization Algorithms

- There are two general strategies used by multi-objective optimization algorithms:

1. Define a series of "single-objective" subproblems such that the solution of each subproblem gives one point on the Pareto frontier. Subproblems may be defined by:
   a) Changing the objective function
   b) Changing the constraints
2. Simultaneously optimize a population of points to iteratively approach the true Pareto frontier.

Pareto Filtering

- The various multi-objective optimization approaches *attempt* to sample points on the Pareto frontier, but generally there is no guarantee that they will succeed.
- As a final step to any multi-objective optimizer, a “Pareto filter” that removes any dominated points from the final sampling should be used. This filter just applies logical checks of the inequalities involved in the definition of dominance.

Aggregate Objective Function Approach

- One approach to multi-objective optimization is to define a "weighted" aggregate objective function of the form

$$f(w_1 \cdot g(y_1), \ldots, w_k \cdot g(y_k))$$

  which may be used as the objective function in a "single-objective" optimization algorithm
- Here w_i represents a "weight" on each attribute such that the weights sum to 1.
- By specifying different values for the weights, different subproblems are defined. Each subproblem is solved to sample a single point.
- Typically the weights are selected to form an evenly spaced grid in the "weight space" with the hopes that this will yield an even sampling of the Pareto frontier.

Aggregate Objective Function Approach

- The aggregate objective function is used to form a single-objective optimization subproblem.
- The subproblem also includes any constraints on the independent variables and/or attributes that the designer wishes to impose.

Aggregate Objective Function Approach

- A well-known pitfall of the AOF approach is that it can sample only from regions of the Pareto frontier whose curvature is greater or equal to the curvature of the AOF.
- See Messac et al. “Ability of objective functions to generate points on nonconvex Pareto frontiers” AIAA Journal 38, 2000
- AOFs are generally convex, so this problem only affects concave Pareto frontiers.
- Even with the required curvatures, AOFs often lead to irregular samplings of the frontier for an evenly spaced weight grid.

Weighted p-norm

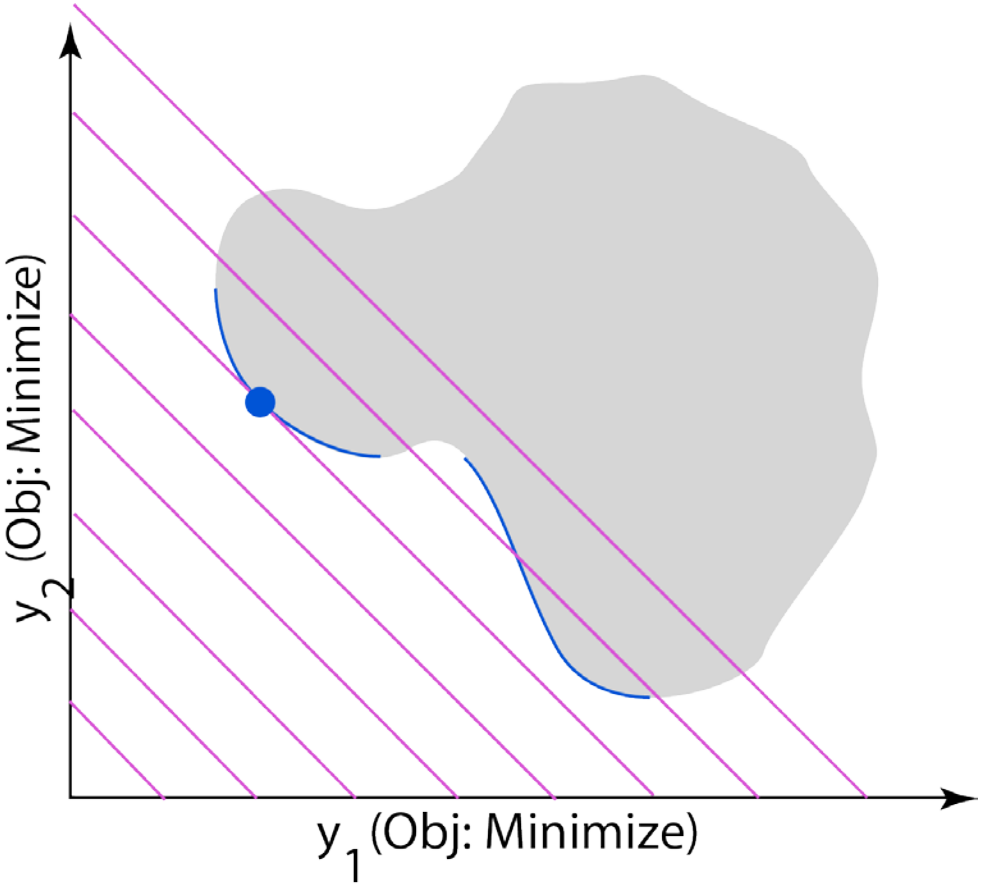
- A common aggregate objective function is the weighted p-norm:

$$f(\boldsymbol{x}) = \left(\sum_{i=1}^{k} w_i y_i^p \right)^{(1/p)}$$

- When p=1, this is simply a weighted sum of the attributes

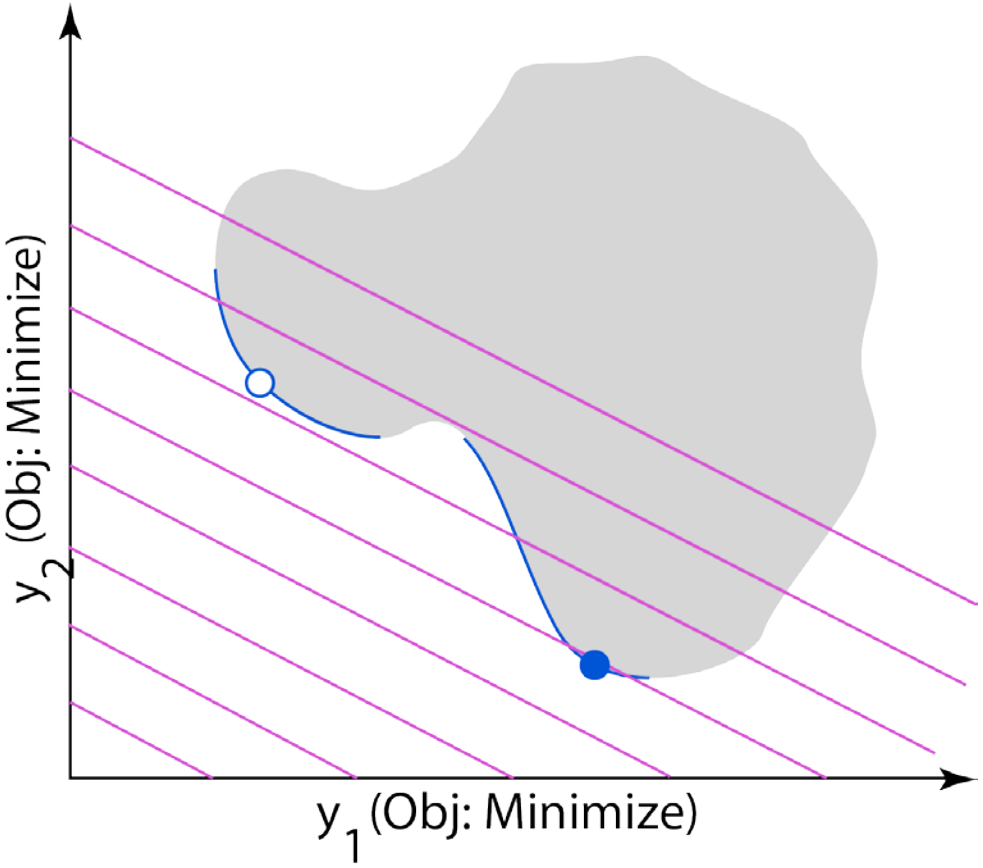
Weighted Sum Example

- $f(x) = 0.5y_1(x) + 0.5y_2(x)$



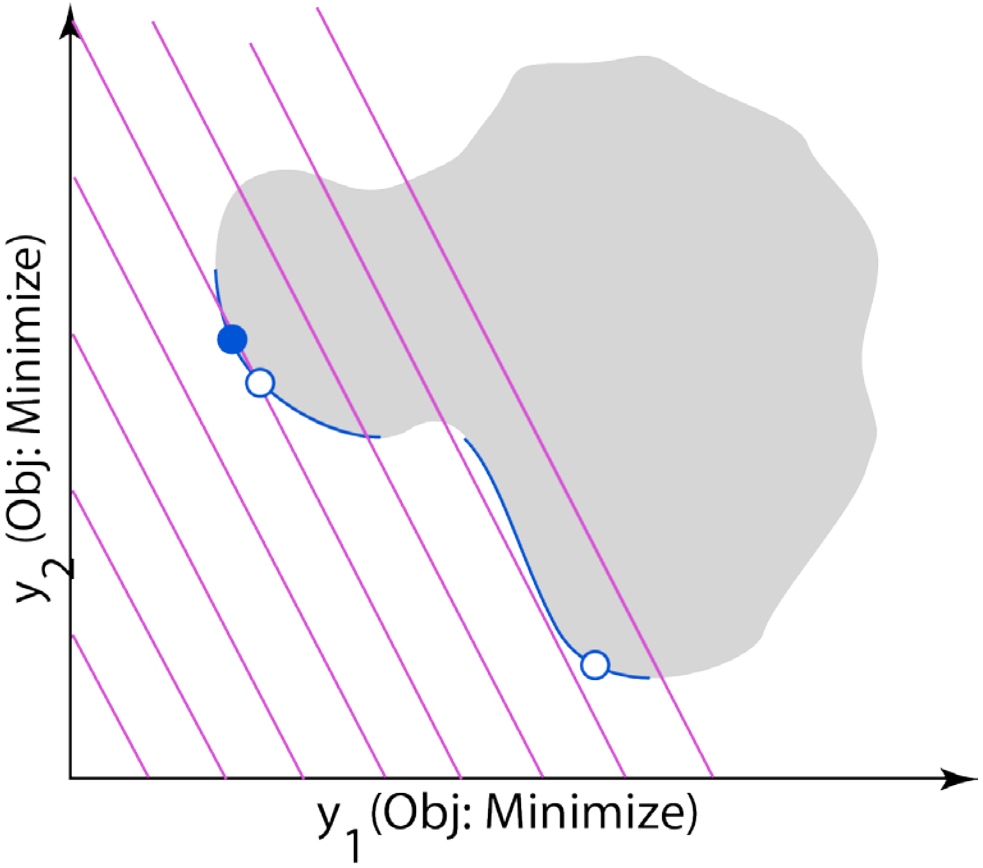
Weighted Sum Example

- $f(x) = 0.333y_1(x) + 0.667y_2(x)$

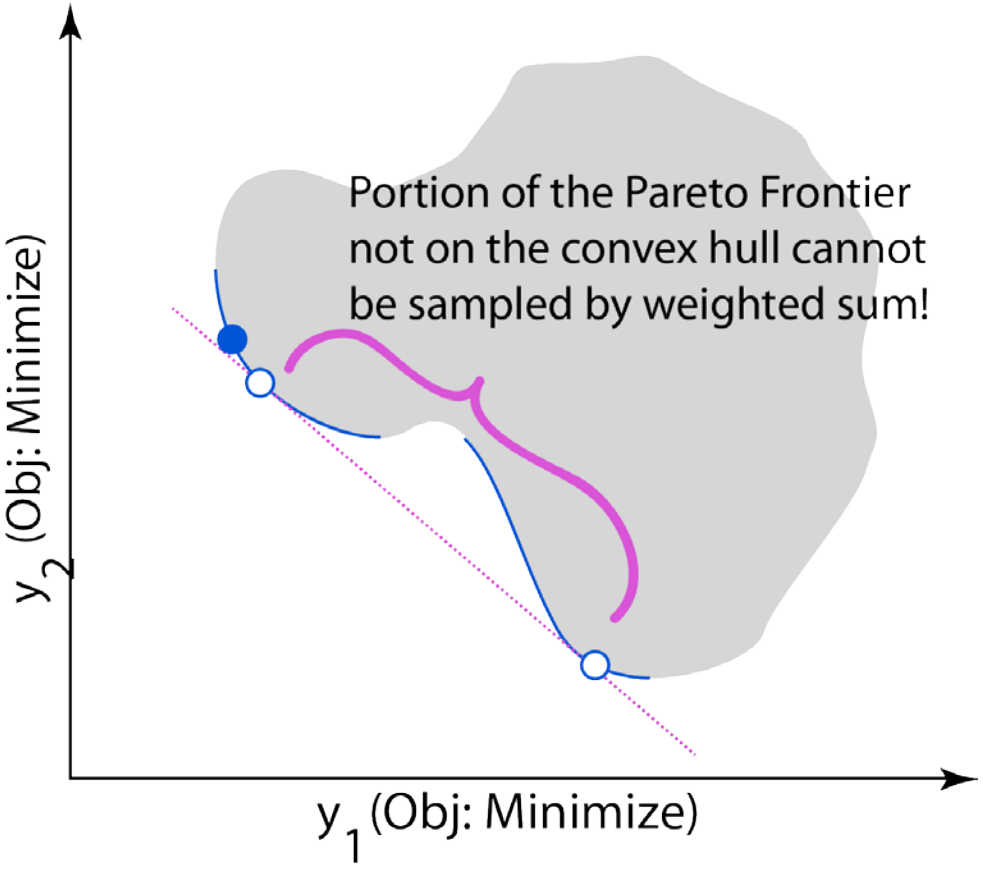


Weighted Sum Example

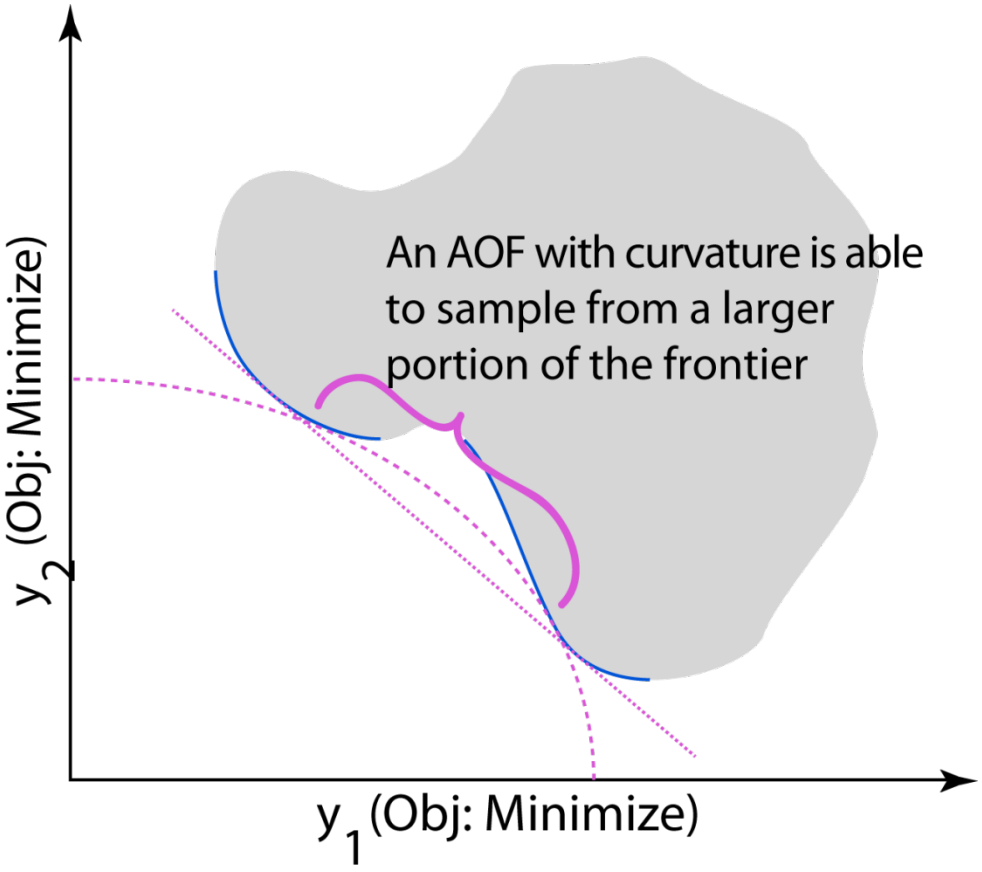
- $f(x) = 0.667y_1(x) + 0.333y_2(x)$



Weighted Sum Example



Weighted Sum Example

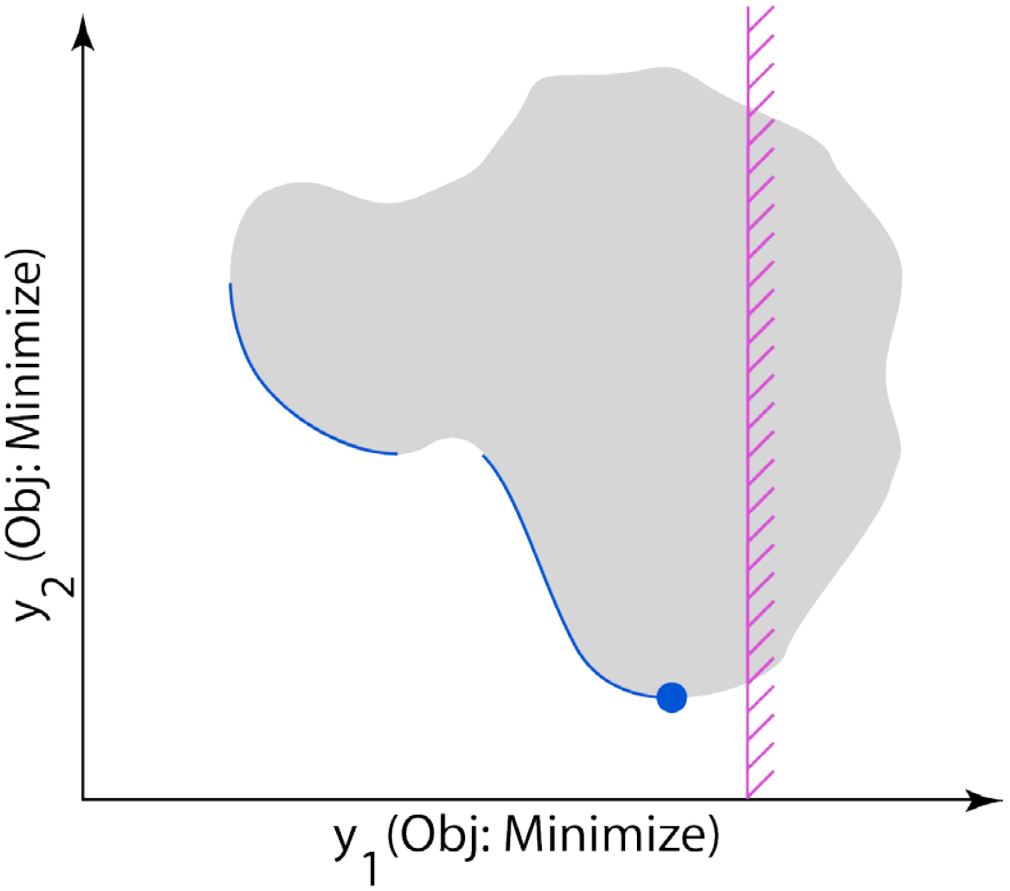


Epsilon-Constraint Method

- An alternative to the aggregate objective function is to simply use one of the attributes as the objective function, while constraining the remaining attributes to be at least as good as some value.
- By varying the values of these threshold constraints, different points on the Frontier are sampled.
- This is called the "**epsilon constraint method**".
- For unusually shaped objective spaces, many subproblems will have no solution.
- The similar "**method of proper equality constraints**" uses equality constraints instead of inequality constraints.

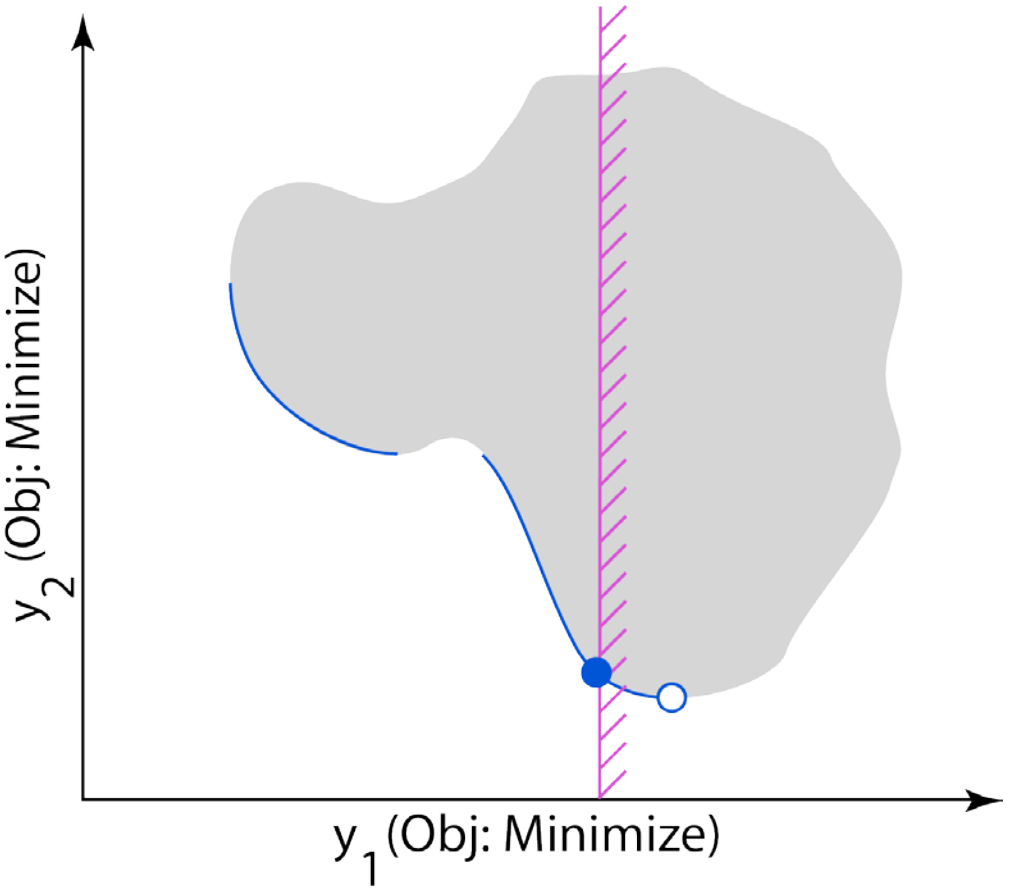
Epsilon-Constraint Method

- $min\ (y_2)\ s.\ t.\ y_1 < a$



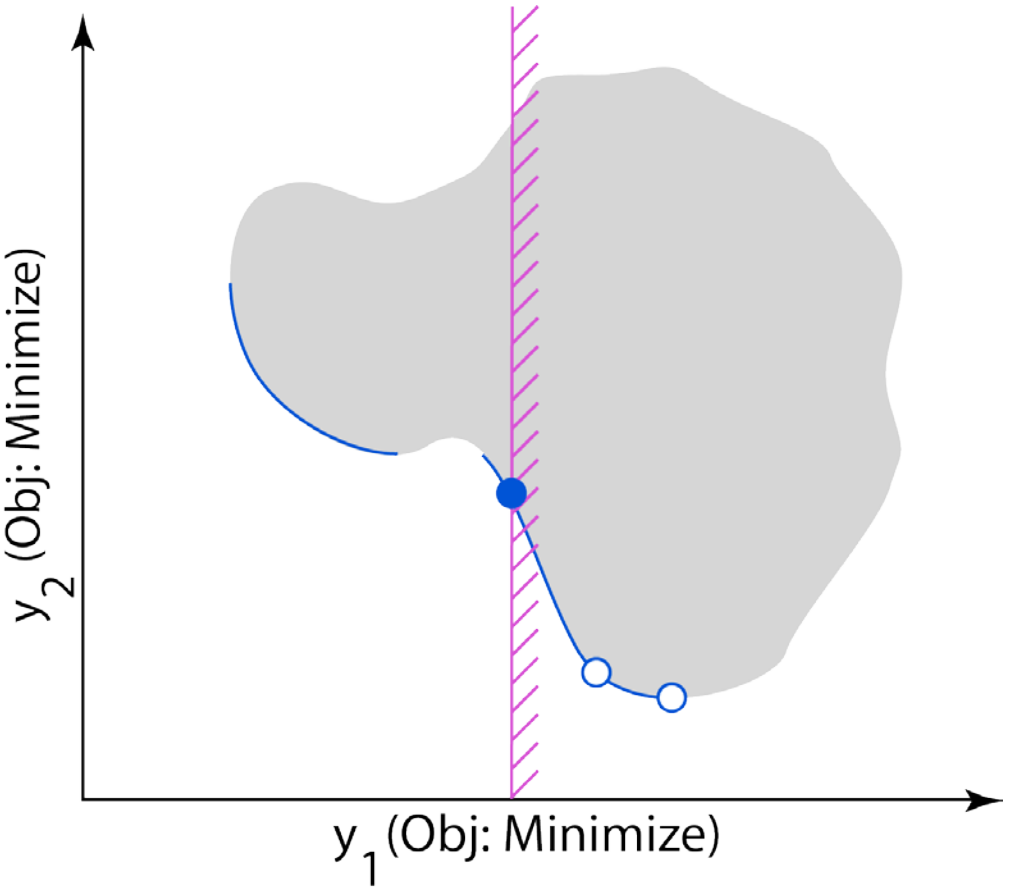
Epsilon-Constraint Method

- $min\ (y_2)\ s.\ t.\ y_1 < b$



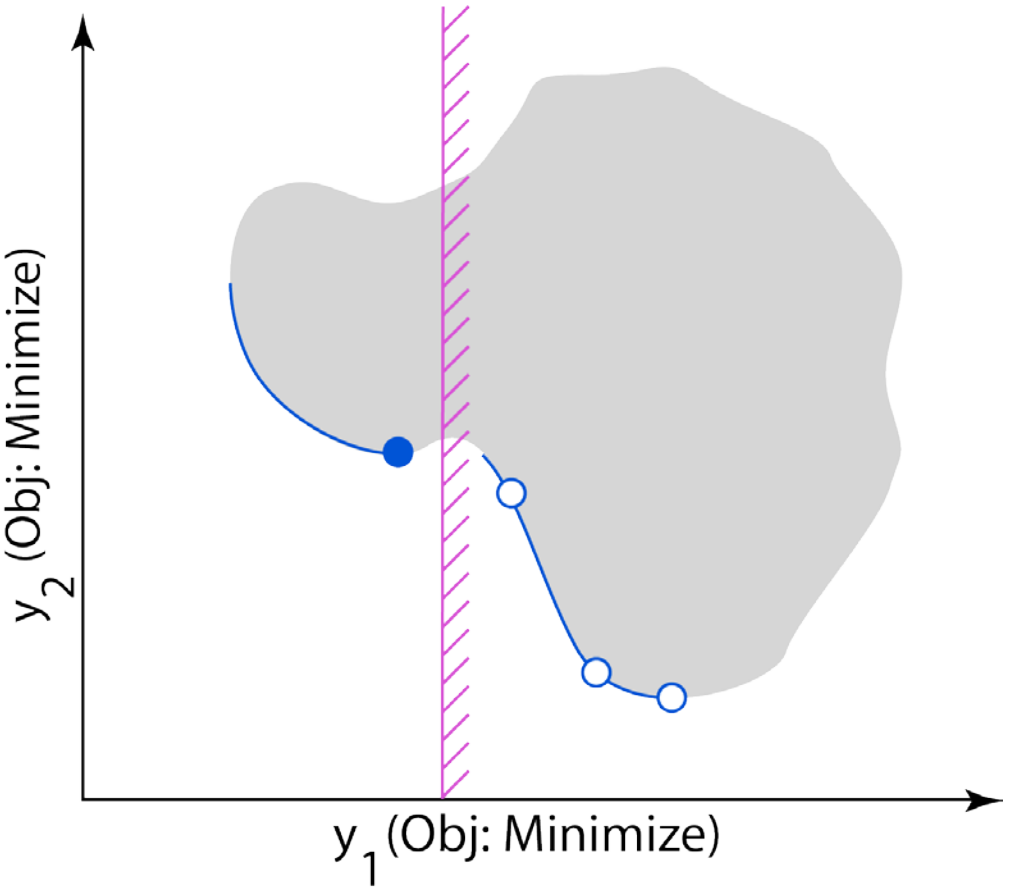
Epsilon-Constraint Method

- $min\ (y_2)\ s.\ t.\ y_1 < c$



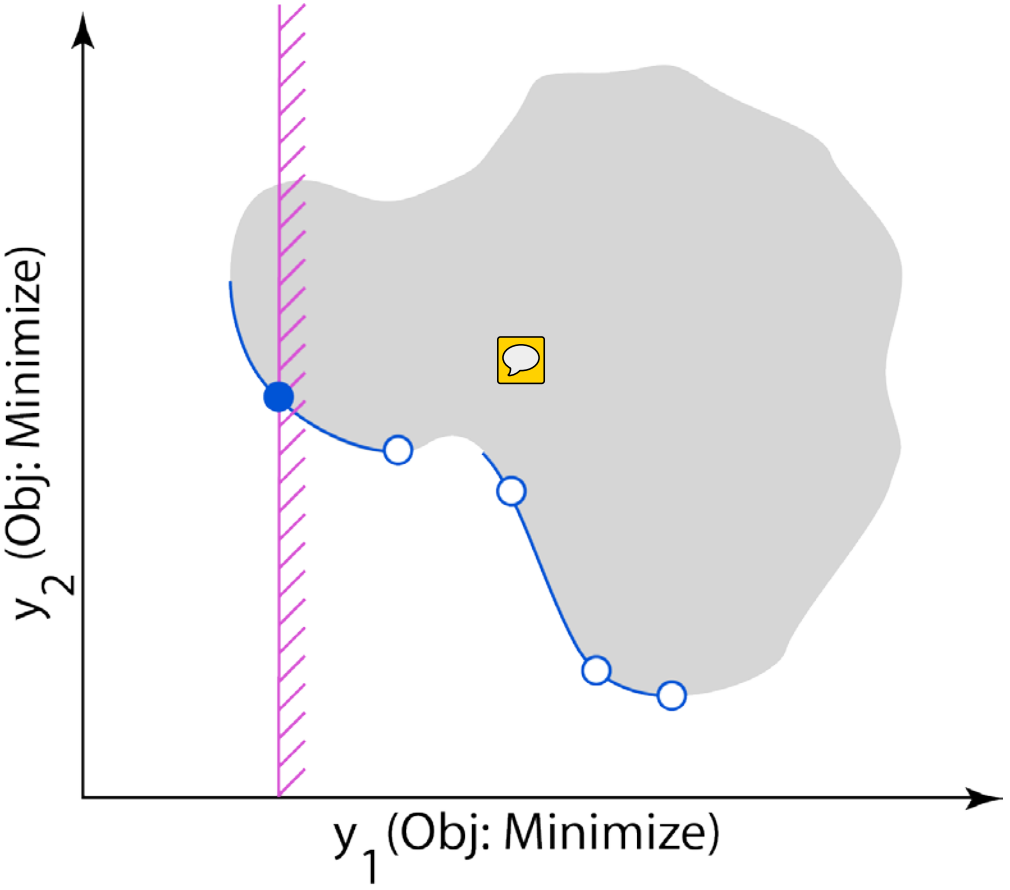
Epsilon-Constraint Method

- $min\ (y_2)\ s.\ t.\ y_1 < d$



Epsilon-Constraint Method

- $min\ (y_2)\ s.\ t.\ y_1 < e$



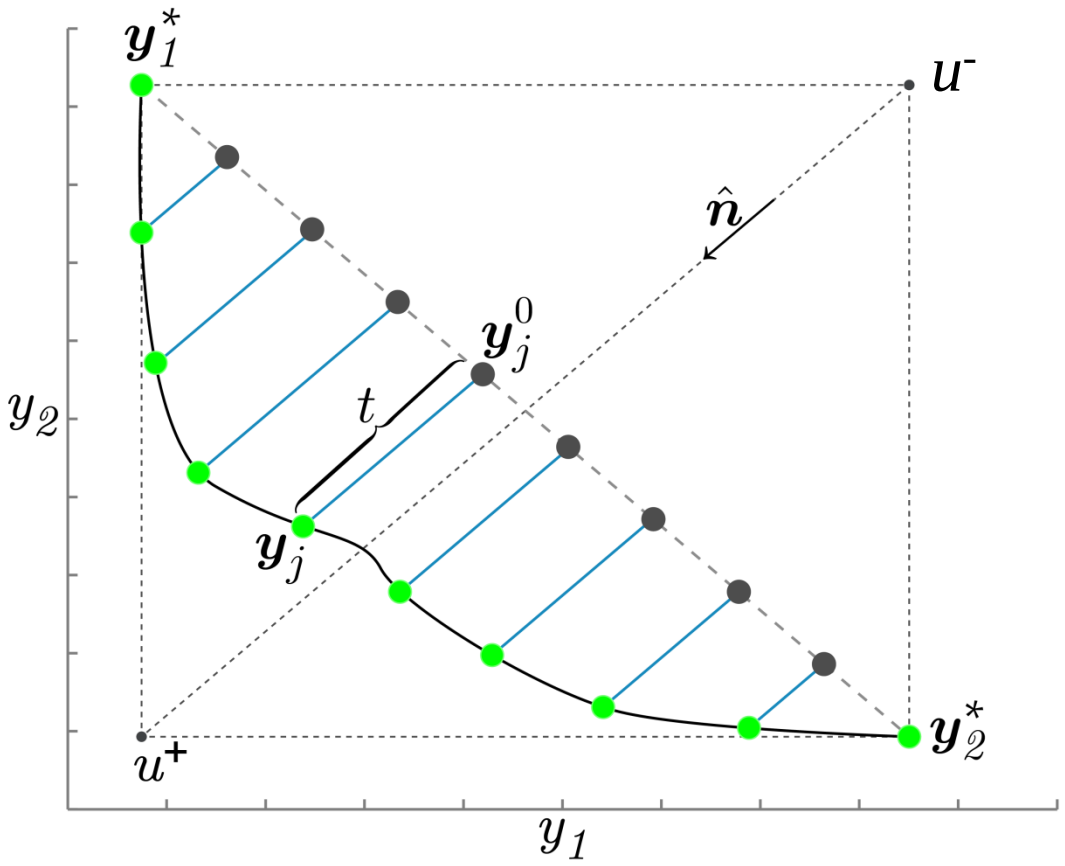
Normal Boundary Intersection

- Normal Boundary Intersection was invented by Das and Dennis (1998) to yield more evenly spaced samplings of the Pareto frontier.
- Similar to the epsilon-constraint method, NBI works by forming a series of subproblems, each of which has different constraints.
- The constraints are more smartly defined in order to minimize the number of failed subproblems.
- However, NBI has a characteristic inability to sample from regions near the edges of < 2-attribute Pareto frontiers

Normal Boundary Intersection

1. Using an appropriate single-objective optimizer, find the k points that optimize each attribute individually, y_i^*
2. Form an evenly spaced simplex-grid of "basepoints" in the objective space by taking weighted sums of the individual optima: $y_j^0 = w_1 y_1^* + w_2 y_2^* + \ldots + w_k y_k^*$
3. Compute the "utopia point" and "antiutopia point" by taking the best and worst attributes from the set $\{y_1^*, y_2^*, \ldots, y_k^*\}$
4. The search direction is the vector from the antiutopia point to the utopia point.
5. For each basepoint, maximize the distance along this direction, while still remaining in the feasible objective space.

Normal Boundary Intersection

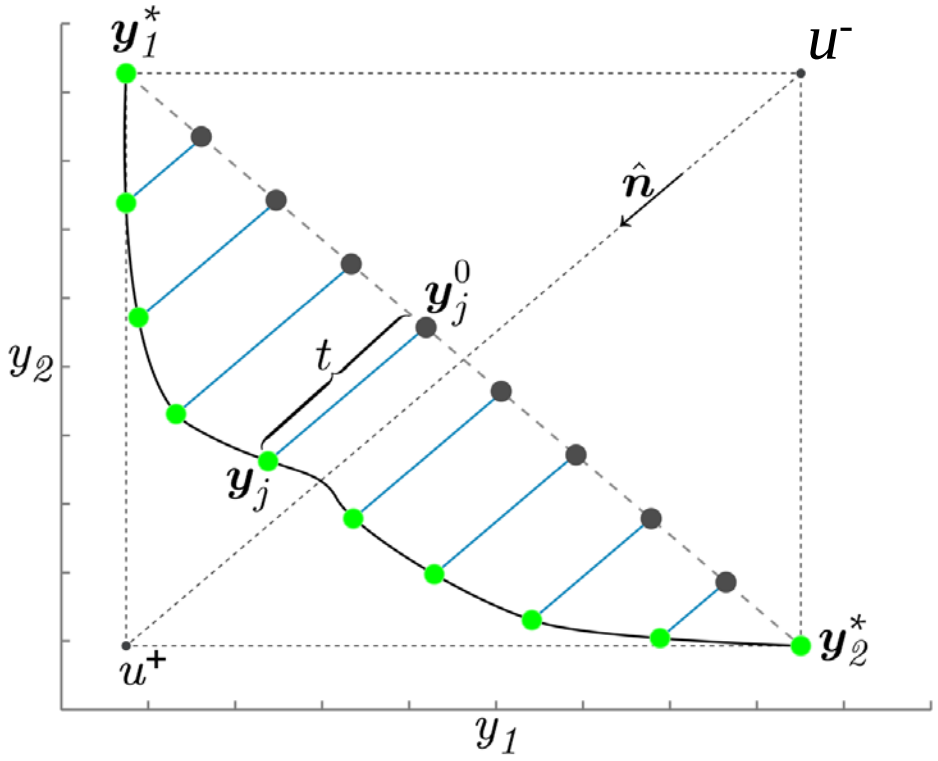


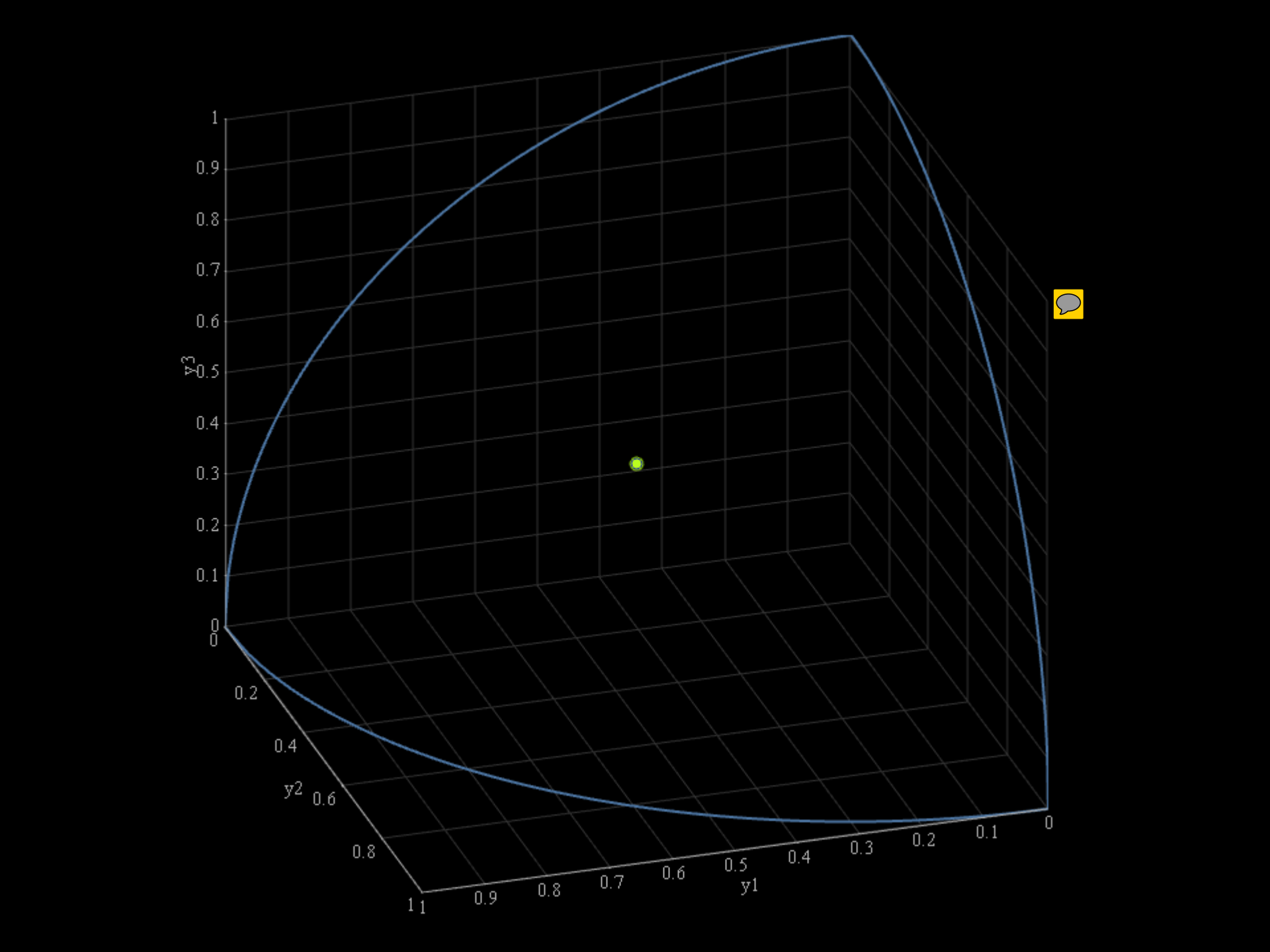
Normal Boundary Intersection

$$\max_{\boldsymbol{x},t} \; t$$

$$\text{s. t. } \boldsymbol{y}_j^0 + t\hat{\boldsymbol{n}} = f(\boldsymbol{x})$$

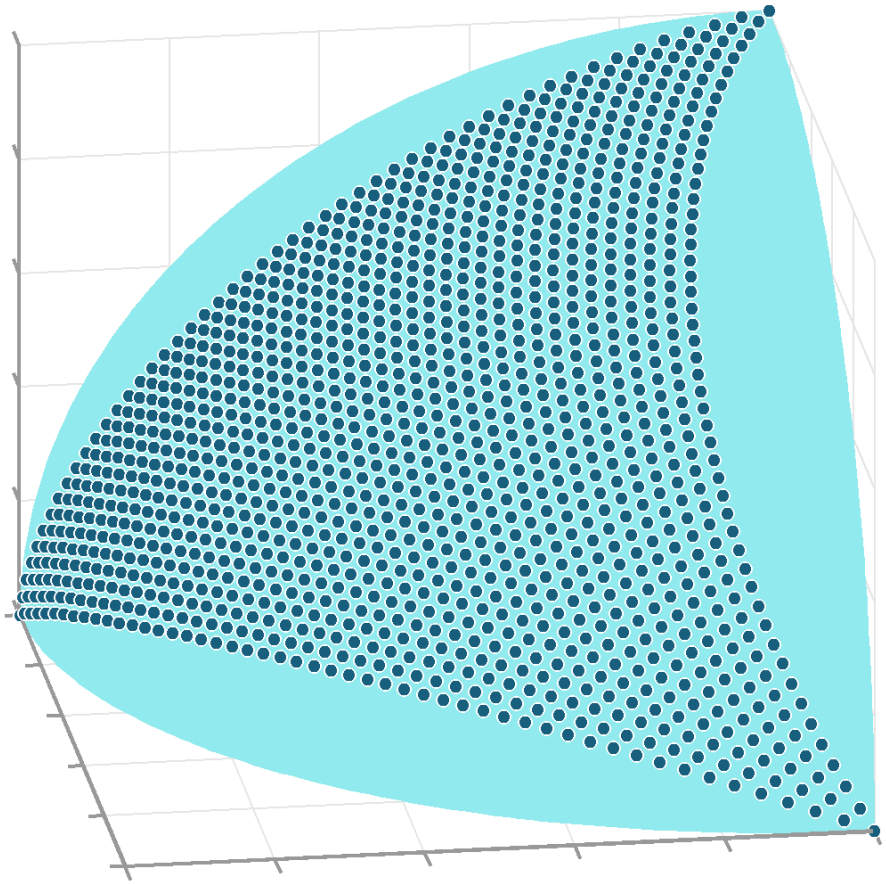
$$\boldsymbol{x} \in D$$



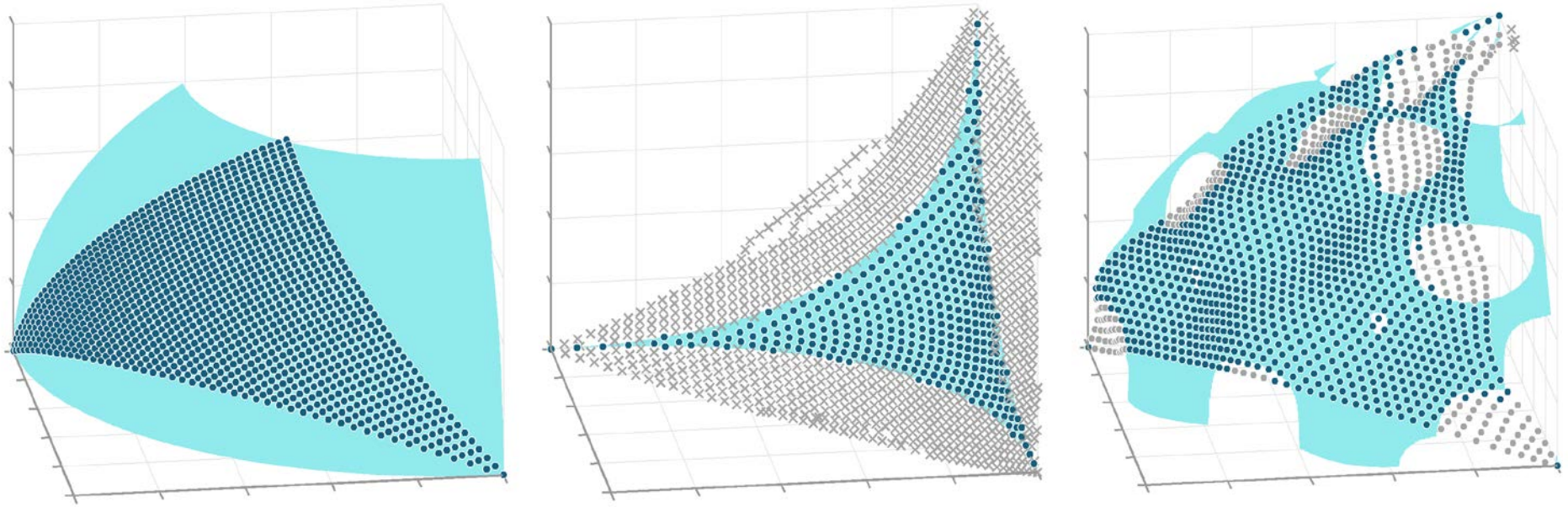

y3
y2
y1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

Normal Boundary Intersection

- NBI is unable to sample from regions near the boundary of most Pareto frontiers involving 3 or more attributes.



Normal Boundary Intersection



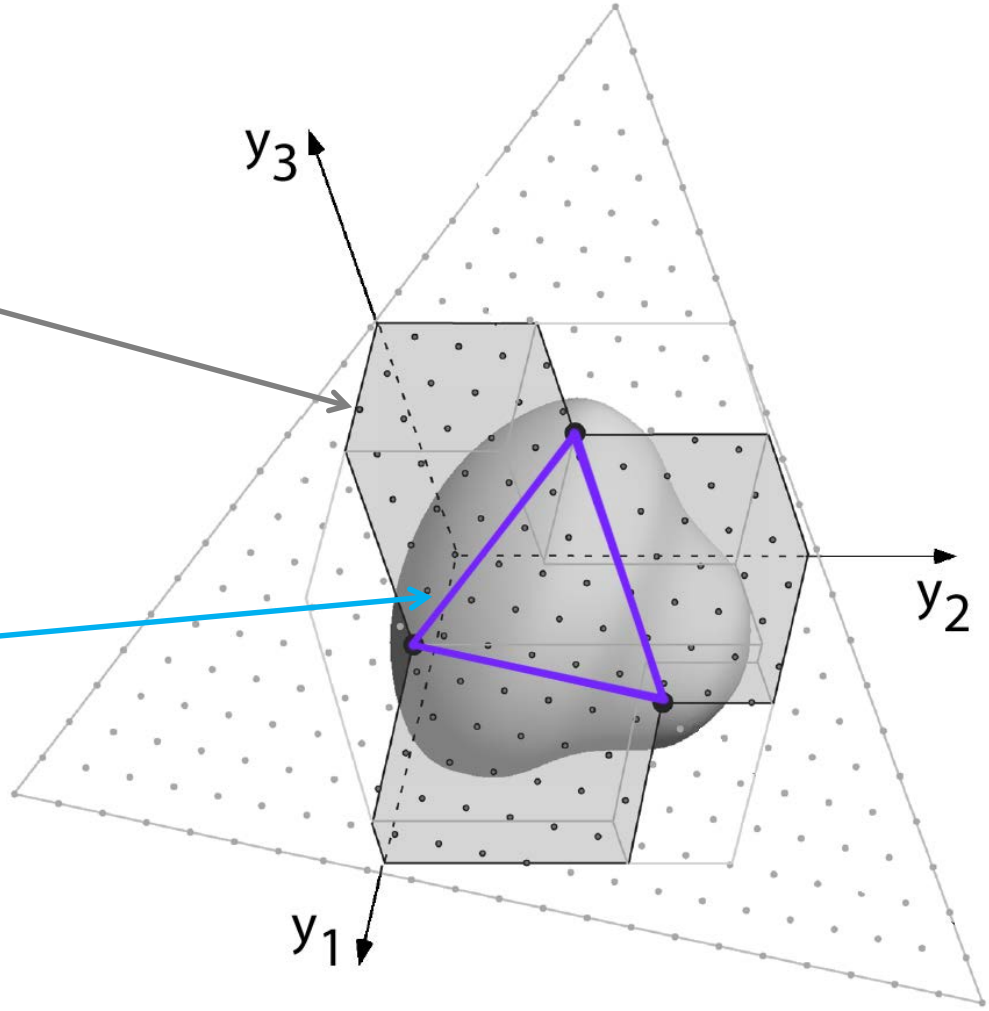
(Normalized) Normal Constraint Method

- Messac et al. invented the Normal Constraint Method, which is conceptually similar to NBI except that it…
  - Replaces NBI's equality constraint with several inequality constraints
  - Locates the search basepoints such that they are guaranteed to cover the entire Pareto frontier.
    … But the basepoints are still uniformly distributed and this leads to many subproblems failing.
- You will often see NNC mentioned in the same sentence as NBI

(Normalized) Normal Constraint Method

Region Searched by NNC

Region Searched by NBI



NSGA-II

- The methods just described all sample the Pareto frontier by solving single-objective subproblems.
- An alternative approach to multi-objective optimization is to use a multi-objective evolutionary algorithm (MOEA) to optimize a population of points that sample the Pareto frontier.
- The most popular MOEA is NSGA-II (Non-Dominated Sorting Genetic Algorithm II)
- NSGA-II was published by Deb et al. in 2002 and is used by the MATLAB function “gamultiobj”

NSGA-II

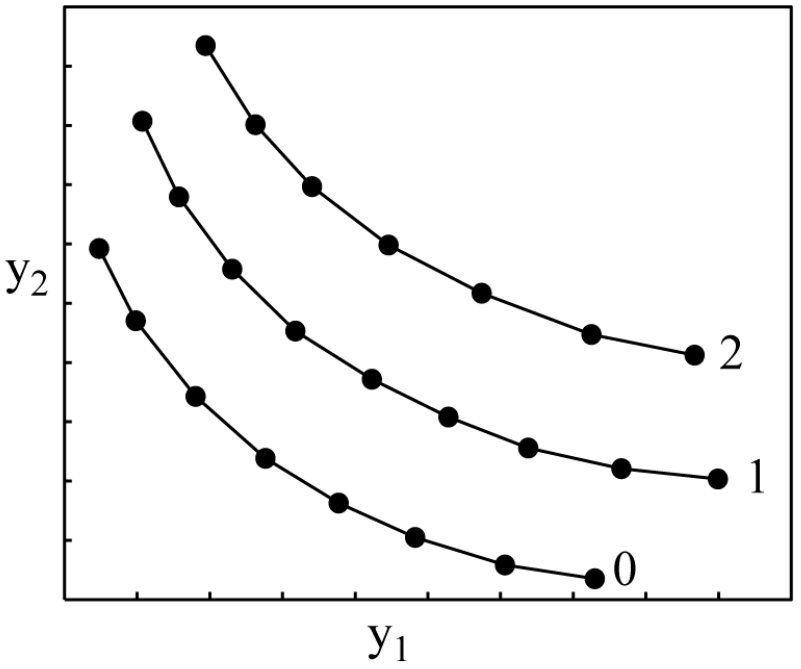
- Unlike the previously described algorithms, NSGA-II directly uses the concept of dominance to optimize the population.
- NSGA-II's "fitness function" is based on two quantities that are calculated for each population member at each iteration:
  - Non-Domination Level – A measure of relative dominance
  - Crowding Distance – A measure of relative isolation

Non-Domination Levels

- Non-domination levels generalize the idea of “dominated” vs. “non-dominated” to integer valued levels that indicate *how dominated* each population member is relative to the other population members.
- Non-domination level of 0 indicates that a population member is non-dominated.
- Higher non-domination levels indicate increasingly higher levels of dominance (i.e. further from the Pareto frontier)

Non-Domination Levels

1. Assign all non-dominated population members to level 0.
2. "Set aside" these points and consider only the remaining population members
3. Assign all non-dominated members of this remaining subset to level 1.
4. Set aside these points…
5. Repeat until all population members have been assigned to a level.



Crowding Distance

- Crowding distance is a measure of how isolated each point in the population is.
- By preferring more isolated points, i.e. those with the largest crowding distance, the final sampling becomes more evenly spaced.
- Crowding distance is only calculated among population members of the same non-domination level. A point's crowding distance is unaffected by points on higher and lower non-domination levels.

Crowding Distance

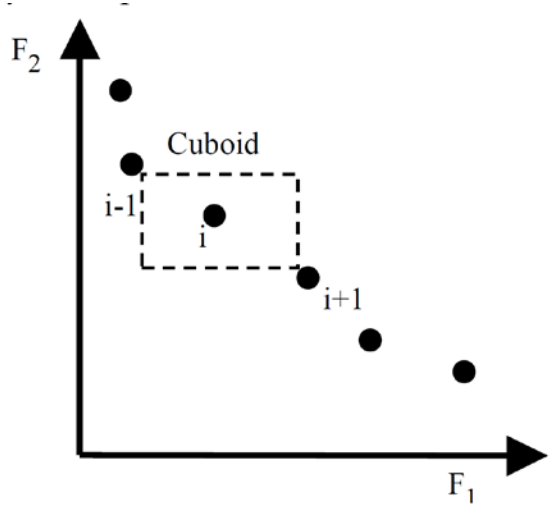


Figure 1. Crowding distance computation.

Crowding distance is calculated by first sorting the set of solutions in ascending objective function values. The crowding distance value of a particular solution is the average distance of its two neighboring solutions. The boundary solutions which have the lowest and highest objective function values are given an infinite crowding distance values so that they are always selected. This process is done for each objective function. The final crowding distance value of a solution is computed by adding the entire individual crowding distance values in each objective function.

Crowding Distance

The pseudocode of crowding distance computation is shown below.

1. Get the number of nondominated solutions in the external repository
   a. n = | S |
2. Initialize distance
   a. FOR i=0 TO MAX
   b. S[i].distance = 0
3. Compute the crowding distance of each solution
   a. For each objective m
   b. Sort using each objective value
      S = sort(S, m)
   c. For i=1 to (n-1)
   d. S[i].distance = S[i].distance + (S[i+1].m – S[i-1].m)
   e. Set the maximum distance to the boundary points so that they are always selected
      S[0].distance = S[n].distance = maximum distance

Crowding Distance

- Among the set, I, of points with non-domination level F_i

```
crowding-distance-assignment(I)
l = |I|                          Number of points in I
for each i, set I[i]_distance = 0
for each objective m
    I = sort(I, m)               Sort according to this attribute
    I[1]_distance = I[l]_distance = ∞
    for i = 2 to (l − 1)
        I[i]_distance = I[i]_distance + (I[i + 1].m − I[i − 1].m)/(f_m^max − f_m^min)
```

Points with min/max of each attribute have infinite crowding distance so they are always preferred

NSGA-II Algorithm

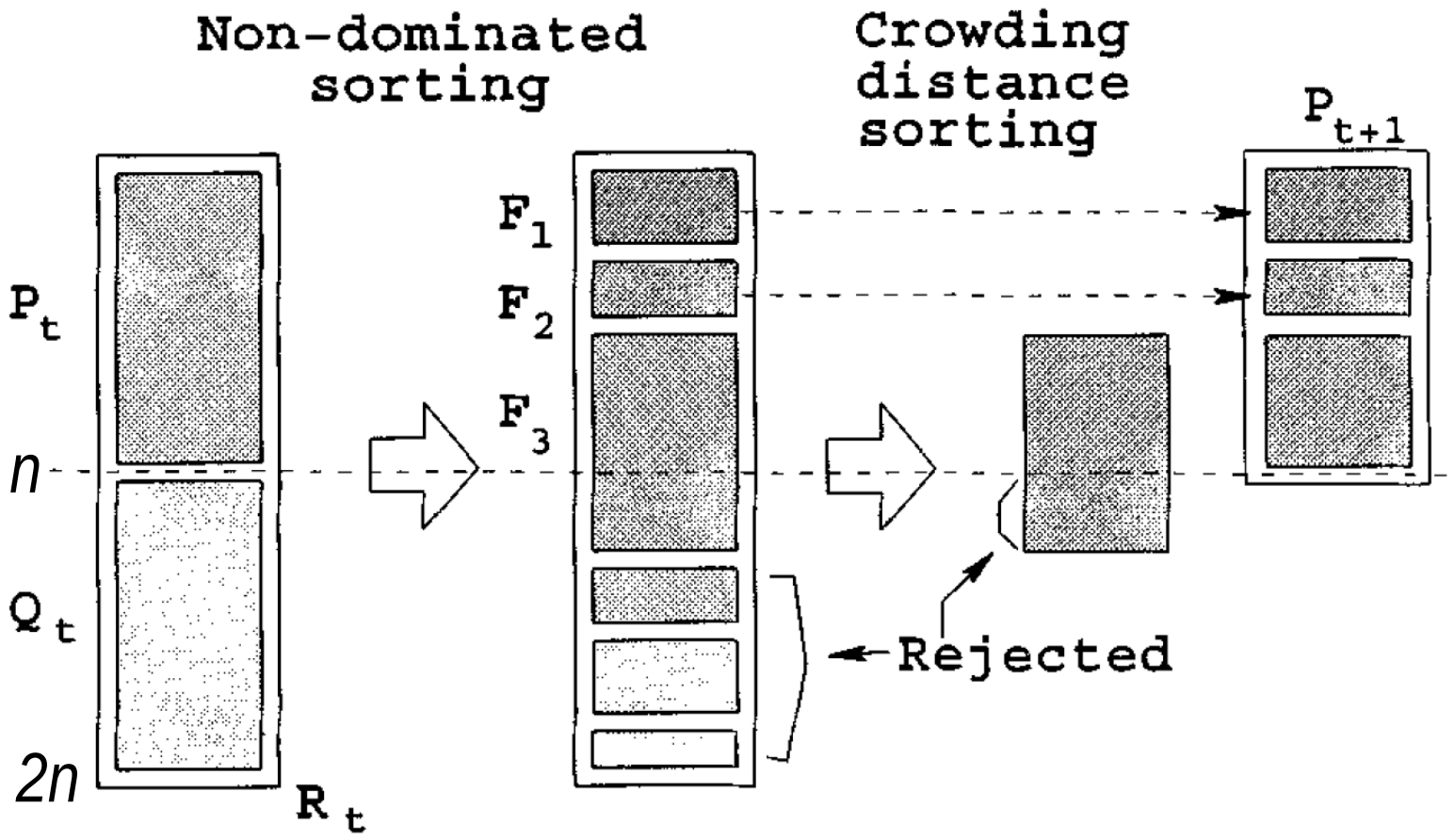
- Begin with *n* randomly generated population members
- For each generation, *t*:
  - Begin with the population, P(t), of size *n*
  - Perform crossover and mutation to generate a new population, Q(t)
  - Append Q(t) to P(t) to yield a 2*n* member population, R(t)
  - Calculate the attribute values $y_1,\ldots,y_k$ for all points in R(t)
  - Calculate non-domination levels of R(t)
  - Determine the maximum non-domination level of R(t) such that fewer than *n* points have a non-domination level strictly lower than this. (This limit may be level 0). Call this level F_{limit} and let *m* denote the number of points with level $<F_{limit}$.

NSGA-II Algorithm

- Copy the *m* points with level $<F_{limit}$ to the next iteration's population, P(t+1)
- Copy the *n-m* points in level F_{limit} with the largest crowding distance to P(t+1) (Note that by definition, level F_{limit} contains at least *n-m* points)
- P(t+1) now contains exactly *n* points, and the next iteration begins.

❖ Note that the crowding distance is only used for points on level F_{limit} and does not need to be calculated for points on other non-domination levels

NSGA-II Algorithm



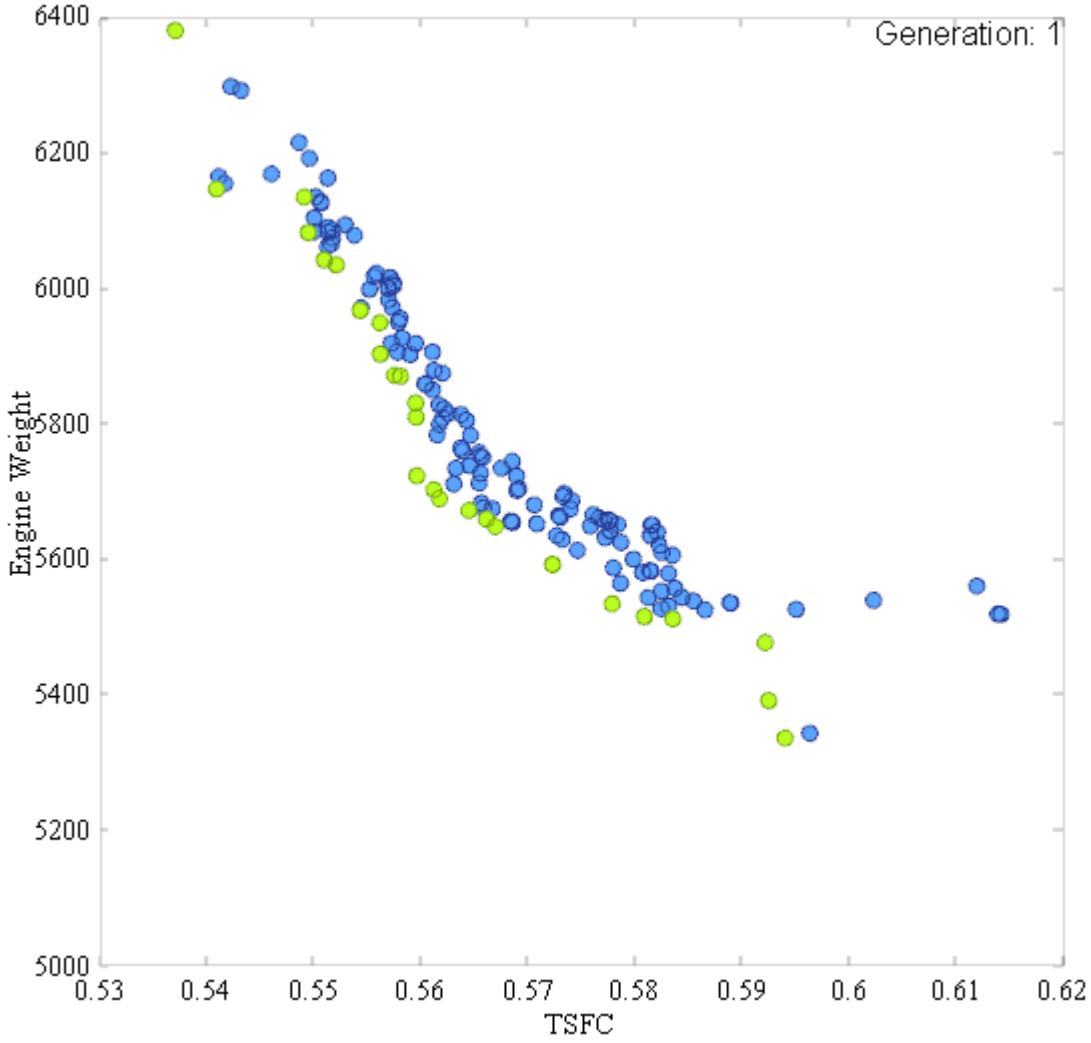
NSGA-II Algorithm

- Note that each iteration considers the “parent” population alongside the “children” formed by crossover/mutation.
- This is called an “elitist” approach because it ensures that the very best designs may be carried through every iteration.
- Because of this, there is no incentive to not crossover or mutate every population member.
- Consequently crossover and mutation fractions should be chosen to be quite high (e.g. 0.8), or the algorithm may be coded so that every population member is either mutated or used in a crossover operation.

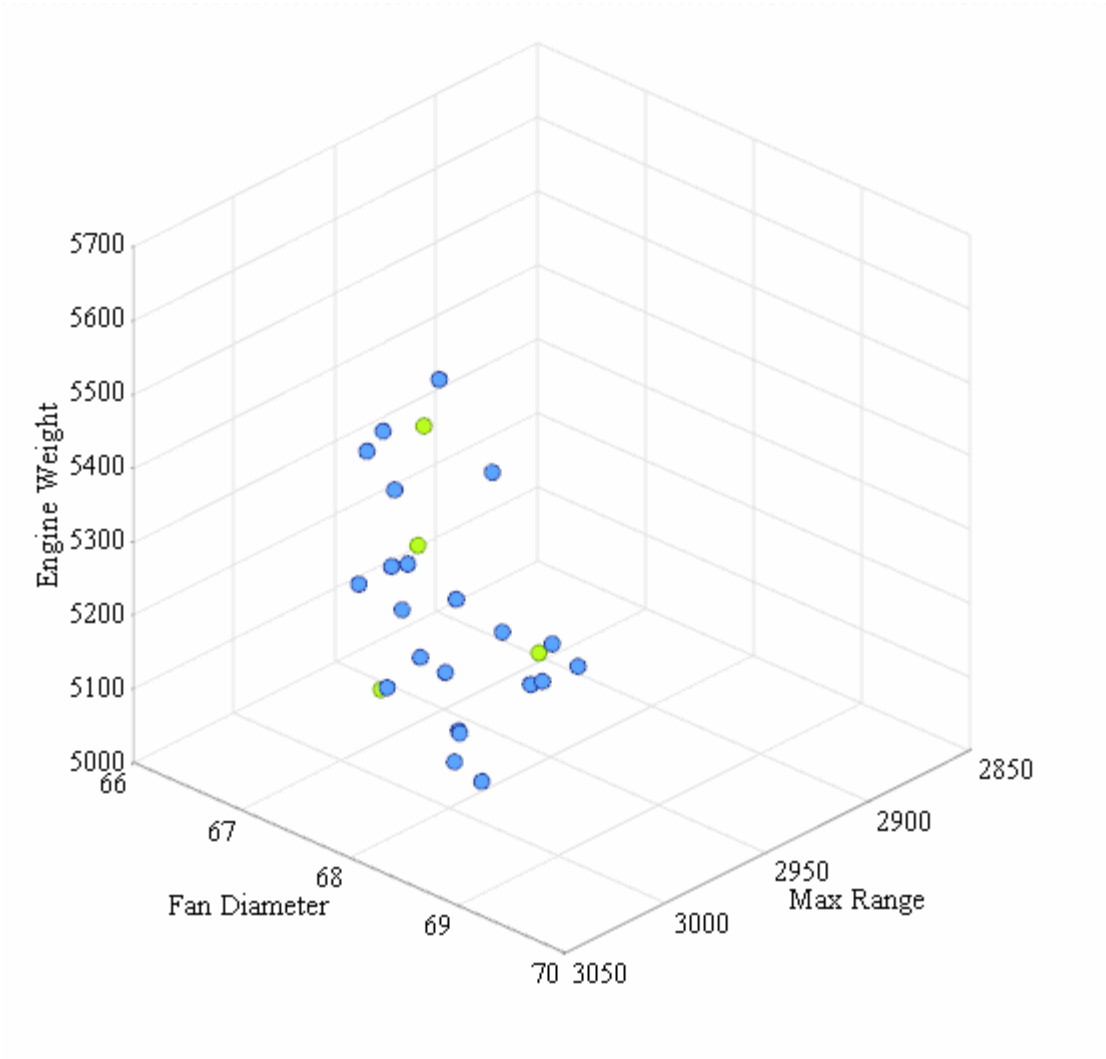
Constrained NSGA-II

- A simple method of enforcing constraints is that for constrained optimization, add (the largest non-domination level + 1) to the calculated non-domination level of each infeasible population member.
- E.g. if the largest non-domination level for the entire population is 6, an infeasible population member whose non-domination level is 3 would be assigned an effective non-domination level of 10.
- Note that this method does not indicate *how infeasible* the population member is. More complicated "penalty functions" could be used to better guide the population toward the feasible design space.

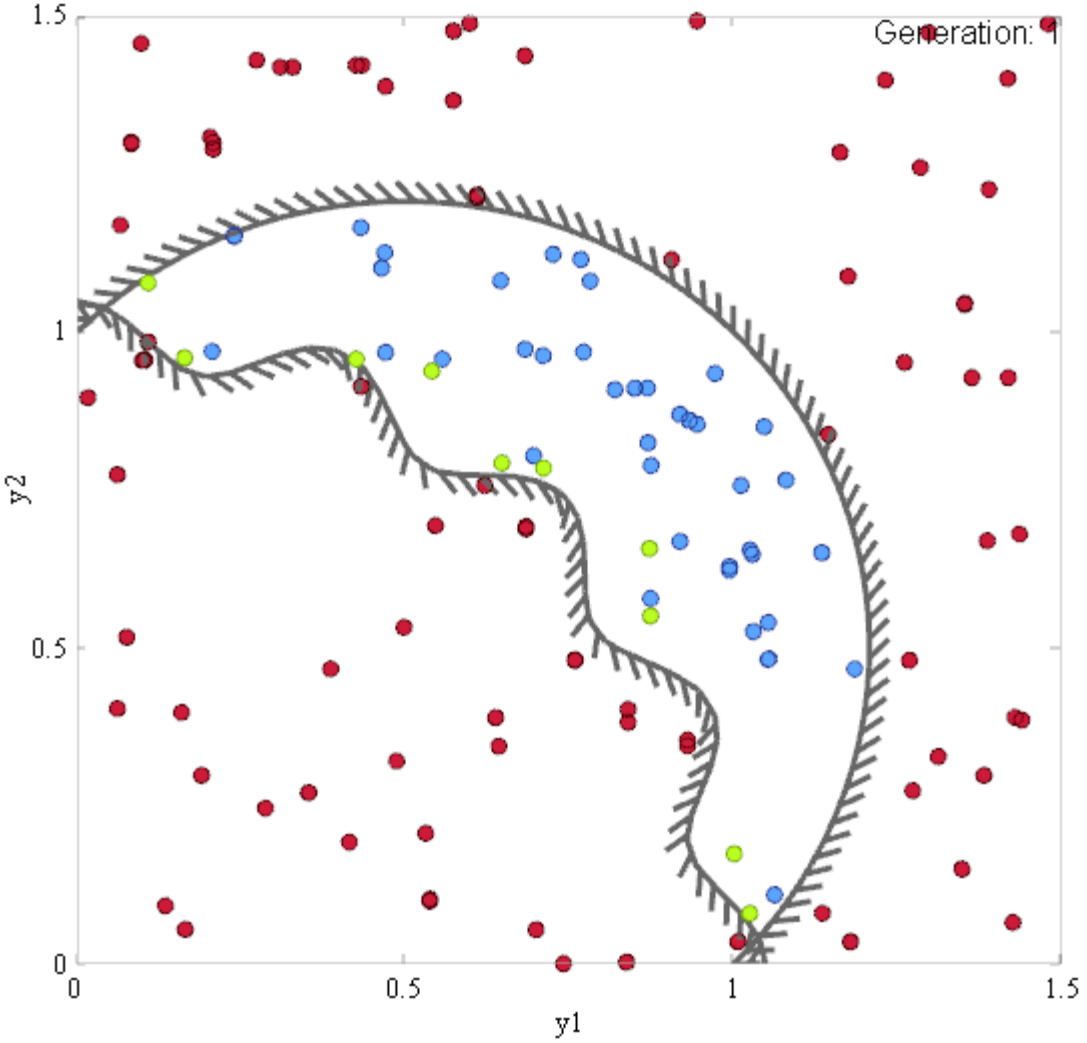
NSGA-II Example



NSGA-II Example



NSGA-II Example



References

- Multi-objective KKT Conditions (not covered in this lecture)
  - Kuhn, H. W. & Tucker, A. W.
    Nonlinear Programming
    Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, Berkeley, 1951
- Weighted Sum Method (not the original source of this method)
  - Das, I. & Dennis, J.
    A closer look at the drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems
    Structural Optimization, 1997, *14*, 63-69

References

- Epsilon Constraint Method
  - Haimes, Y. Y.; Lasdon, L. S. & Wismer, D. A.
    On a Bicriterion Formulation of the Problems of Integrated System Identification and System Optimization
    IEEE Transactions of Systems, Man, and Cybernetics, 1971, *1*, 296-297
- Normal Boundary Intersection
  - Das, I. & Dennis, J.
    Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems
    SIAM Journal of Optimization, 1998, *8*, 631-657
- Normalized Normal Constraint Method
  - Messac, A. & Mattson, C. A.
    Normal constraint method with guarantee of even representation of complete pareto frontier
    AIAA Journal, 2004, *42*, 2101 - 2111

References

NSGA-II

- Deb, K.; Pratap, A.; Agarwal, S. & Meyarivan, T.
  A fast and elitist multiobjective genetic algorithm: NSGA-II
  IEEE Transactions on Evolutionary Computation, 2002*, 6*, 182 - 197

Strength Pareto Evolutionary Algorithm

- Zitzler, E. & Thiele, L.
  An evolutionary algorithm for multiobjective optimization: The strength Pareto approach
  Zurich, Switzerland: Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), 1998