UNIVERSITY of FLORIDA
The Foundation for The Gator Nation

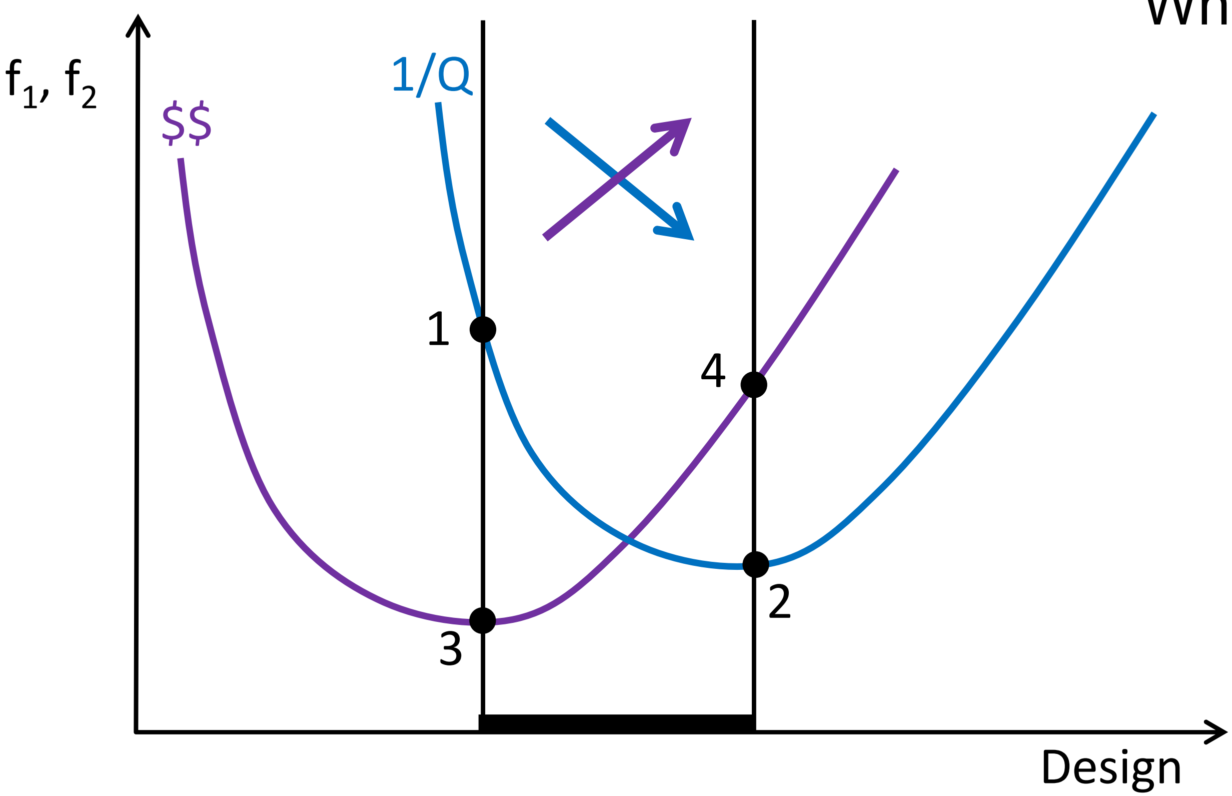
Multi-Objective Optimization

How to optimize multiple objectives?

Build Pareto-Front

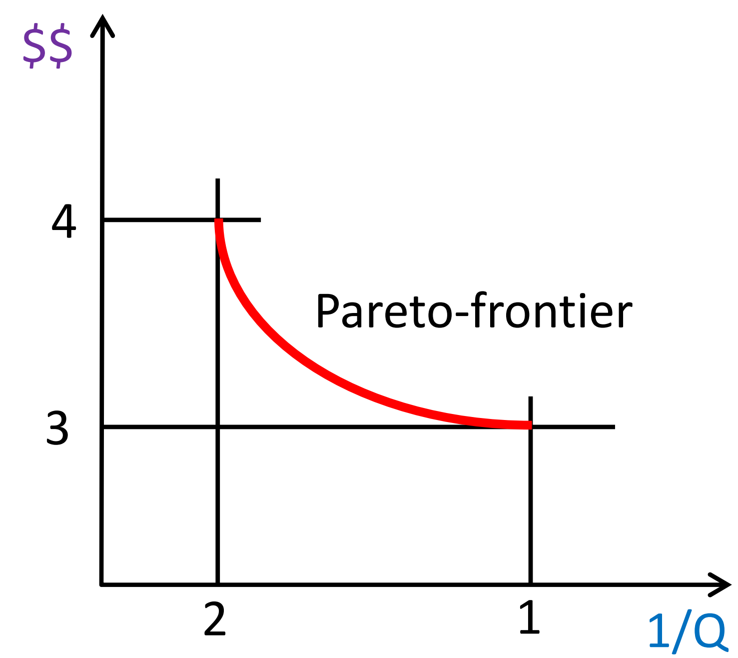
INTRODUCTION

- We often have more than one objective
- When a change in design variable can affect opposite way of different objectives



Which one is optimum?

What's the definition of optimum?



- This means that design points are no longer arranged in strict hierarchy
- There are points that are clearly poorer than others because all objectives are worse
- In optimization jargon we call these points dominated
- Points that are not dominated are called non-dominated or Pareto optimal

MULTI-OBJECTIVE DESIGN STRATEGY

- Vector of objective functions

$$\mathbf{F} = \{ f_1(\mathbf{b}) \quad f_2(\mathbf{b}) \quad \ldots \quad f_m(\mathbf{b}) \}$$

- In multi-objective optimization, objective functions are competing each other
  - We cannot improve one objective without deteriorating others
- We can combine them using weights, but often don't know them

$$f(\mathbf{b}) = w_1 f_1(\mathbf{b}) + w_2 f_2(\mathbf{b}) + \ldots + w_m f_m(\mathbf{b})$$

- Need to explore all possible combinations of competing objectives

MULTI-OBJECTIVE OPTIMIZATION

- Problem formulation

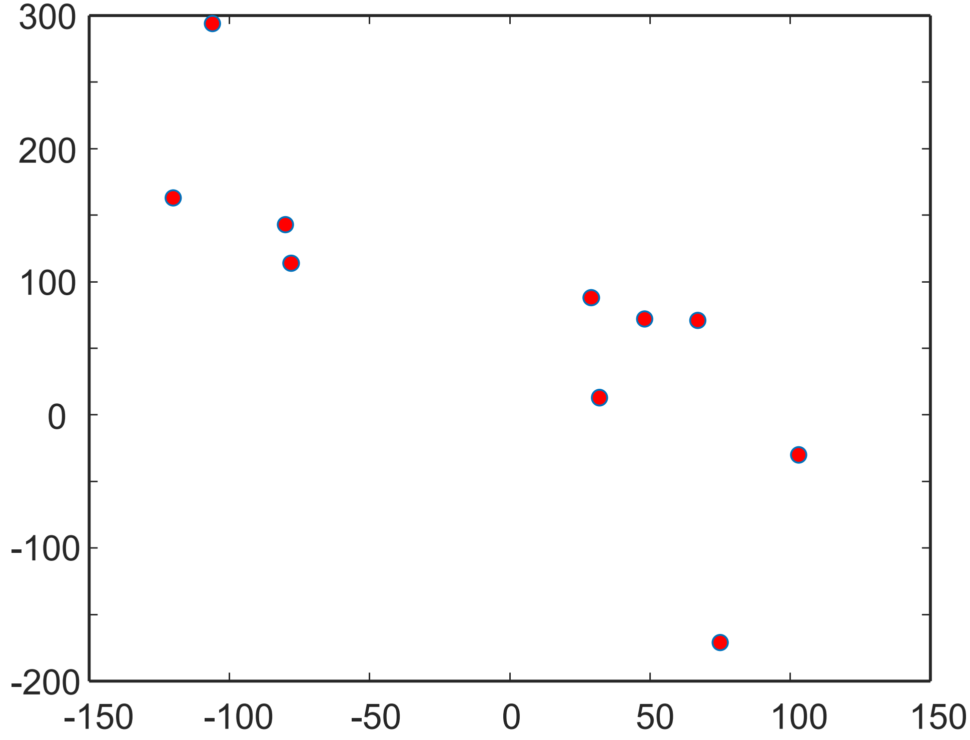
$$
\begin{aligned}
\text{Minimize} \quad & \mathbf{F}(\mathbf{b}) \\
\text{subject to } & h_i(\mathbf{b}) = 0 \\
& g_j(\mathbf{b}) \le 0 \\
& \mathbf{b}^L \le \mathbf{b} \le \mathbf{b}^U
\end{aligned}
$$

- If any objectives are competing, there is no unique solution
- Non-inferior solution: an improvement in one objective requires a degradation of another
- The set of non-inferior solutions is called Pareto front

EXAMPLE: DOMINATION

- We have two objectives that are to be minimized. The following are the pairs of objectives at 10 design points. Identify the dominated ones.

(67,71), (48,72), (29,88), (-106, 294), (32,13)
(-120,163), (103,-30), (-78,114), (-80,143), (75,-171)



EXAMPLE: DOMINATION

- Work choice: Minimize time and maximize fun so that you make $100. Will need between 33.3 to 100 items. Time can vary from 66.7 minutes to 300. Fun can vary between 33.3 and 300.

Item (task)	Pay ($)	Time (min)	Fun index
1	1	3	3
2	2	5	1
3	1	2	2
4	3	2	1

Task 2 is dominated by Task 4!

MULTI-OBJECTIVE FORMULATION

- Problem formulation

$$\text{Minimize} \quad time = 3b_1 + 2b_3 + 2b_4$$

$$\text{Maximize} \quad fun = 3b_1 + 2b_3 + b_4$$

$$\text{subject to} \quad pay = b_1 + b_3 + 3b_4 = 100$$

$$b_i \geq 0$$

- Pay constraint in standard normalized form

$$1 - (b_1 + b_3 + 3b_4)/100 = 0$$

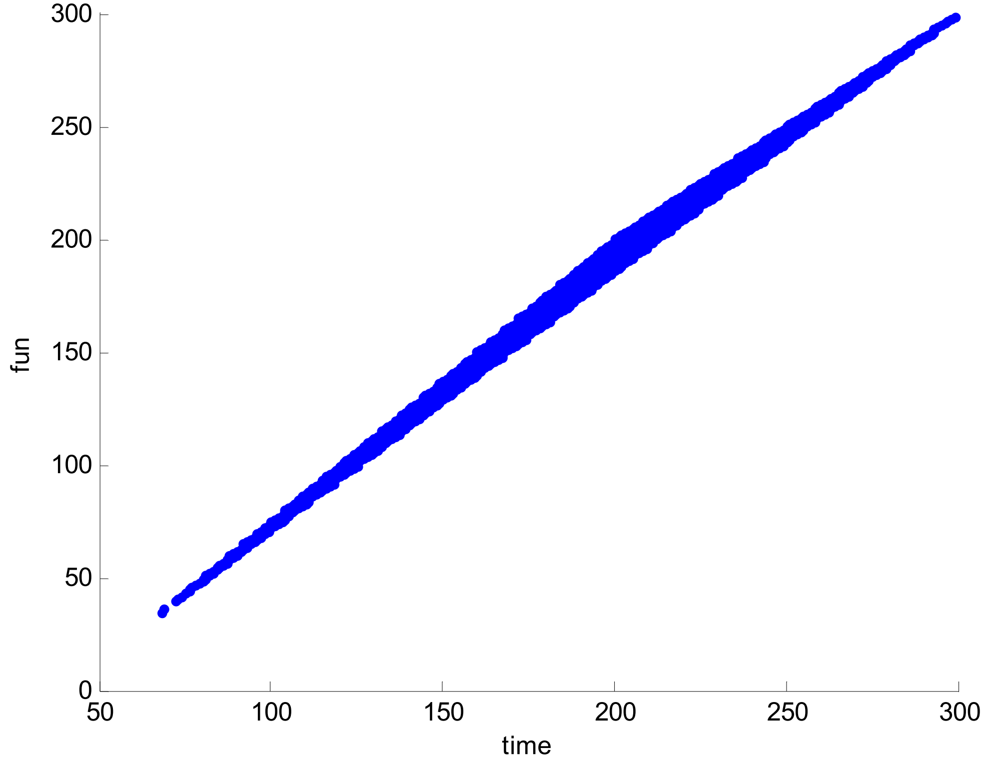
- Common sense constraints

$$b_1 \leq 100,\ b_3 \leq 100, b_4 \leq 33.3$$

Item	Pay	Time	Fun
1	1	3	3
~~2~~	~~2~~	~~5~~	~~1~~
3	1	2	2
4	3	2	1

SOLUTION BY ENUMERATION

- In the range of three variables calculate time and fun for all combinations that produce exactly $100.
- Pareto front is upper boundary and it is almost a straight line.
- Matlab results



EXERCISE: PAY-FUN PROBLEM

- Formulate the problem of maximizing fun and pay for five hours (300 minutes) including only non-dominated variables.
- Obtain the Pareto front analytically or by enumeration and writes its equation (fun as a function of pay).

MORE EFFICIENT SOLUTION METHODS

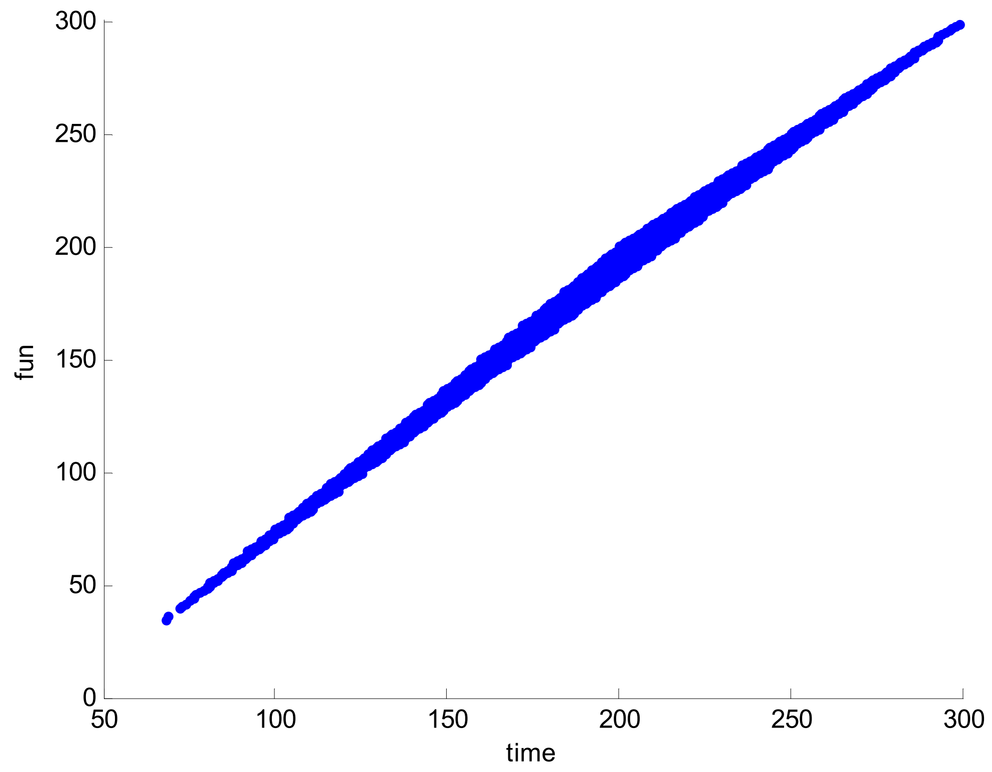
- Methods that try to avoid generating the Pareto front
  - Generate "utopia point"
  - Define optimum based on some measure of distance from utopia point
- Generating entire Pareto front
  - Weighted sum of objectives with variable coefficients
  - Optimize one objective for a range of constraints on the others
  - Niching methods with population based algorithms

EXAMPLE: UTOPIA POINT

- The utopia point is (66.7,300).
- One approach is to use it to form a compromise objective

$$\text{Minimize} \quad \frac{time}{66.7} + \frac{300}{fun}$$

- It gives
  time=168 minutes,
  fun=160
  (b_1=0, b_3=76,b_4=8)



SERIES OF CONSTRAINTS

$$\begin{aligned}
\text{Minimize} \quad & time = 3b_1 + 2b_3 + 2b_4 \\
\text{subject to} \quad & pay = b_1 + b_3 + 3b_4 \geq 100 \\
& fun = 3b_1 + 2b_3 + b_4 \geq fun_k \quad k = 1, 2, \ldots \\
& b_i \geq 0
\end{aligned}$$

The next slides provides a Matlab segment for solving this optimization problem using the function fmincon, but without the requirement of integer variables.

How will you change the formulation so that a single Matlab run will give us results for any required earning level R?

MATLAB SEGMENT

```
b0 = [10  10 10];
for fun_idx = 30:5:300
   A = [-1 -1 -3; -3 -2 -1]; c = [-100;-fun_idx];
   lb = zeros(3,1);
   options = optimset('Display','off');
   [b,fval,exitflag,output,lambda] =
fmincon('myfun’,b0,A,c,[],[],lb,[],[],options);
   pareto_sol(fun_idx,:) = b;
   pareto_fun(fun_idx,1) = fval;
   pareto_fun(fun_idx,2) = 3*b(1)  + 2*b(2) + b(3);
end

function f = myfun(b)
f = 3*b(1) + 2*b(2) + 2*b(3);
```

EXERCISE: PARETO-FRONT

- Generate the Pareto front for the pay and fun maximization using a series of constraints, and also find a compromise point on it using the utopia point.
- What is responsible for the slope discontinuity in the Pareto front on Slide 10?

EXAMPLE

- Minimize time & maximize fun while you make at least $100
- Will need between 33.3 to 100 items
- Fun can vary between 33.3. Try first for 150

Item	Pay ($)	Time (min)	Fun index
1	1	3	3
2	2	4	2
3	1	2	2
4	3	2	1

- Multi-objective optimization formulation

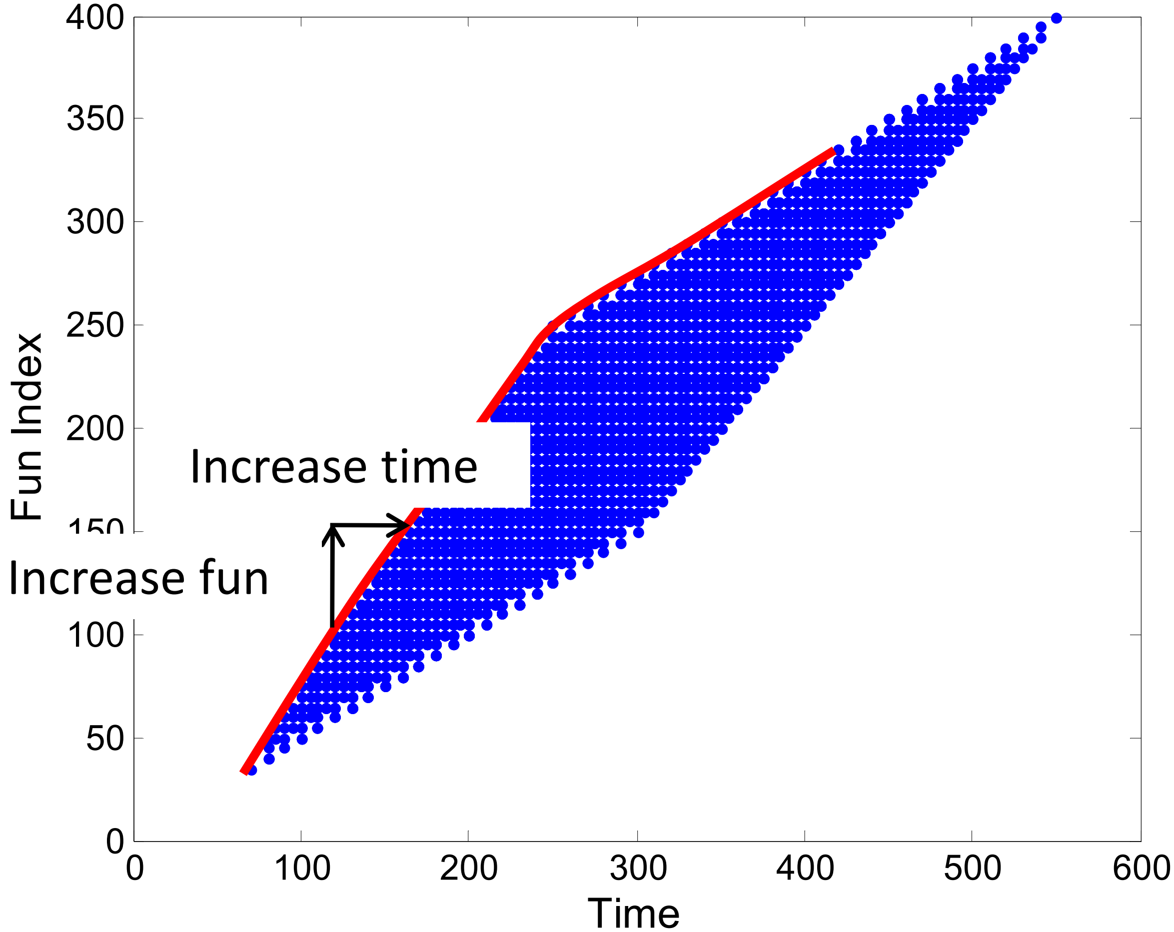
$$\text{Minimize} \quad time = 3b_1 + 4b_2 + 2b_3 + 2b_4$$

$$\text{Maximize} \quad fun = 3b_1 + 2b_2 + 2b_3 + b_4$$

$$\text{subject to } pay = b_1 + 2b_2 + b_3 + 3b_4 \geq 100$$

EXAMPLE

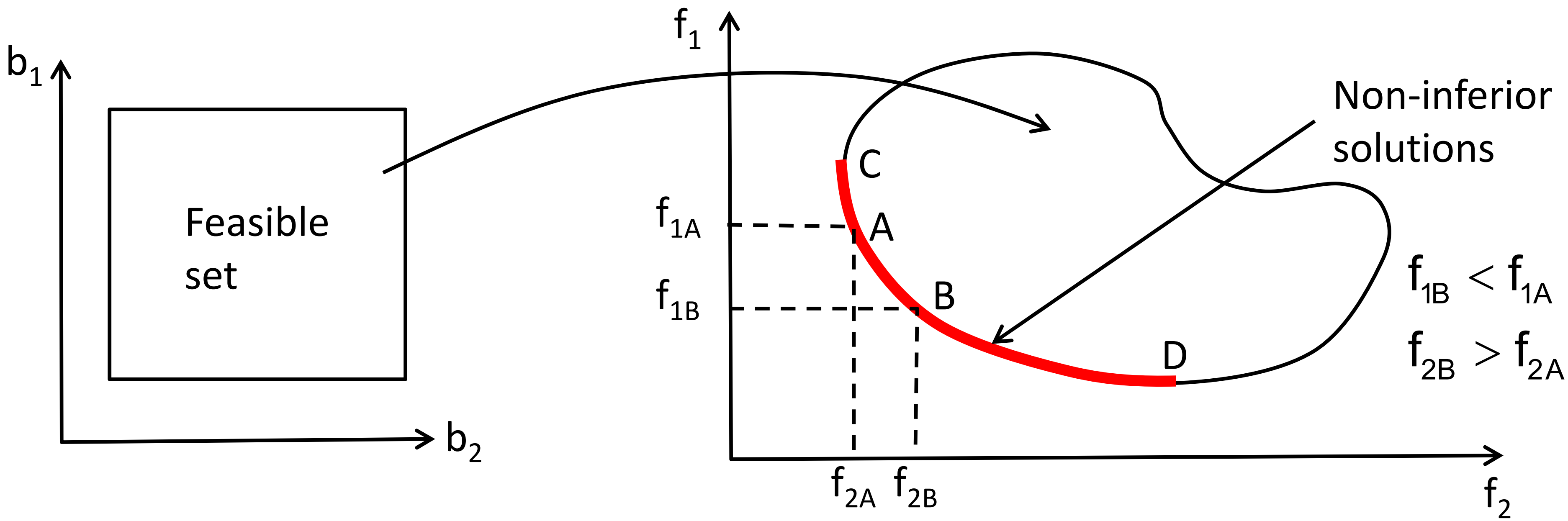
```
f1=zeros(1,100000);
f2=zeros(1,100000);
k=1;
for i1=0:5:50
 for i2=0:5:50
  for i3=0:5:50
   for i4=0:5:50
    pay=i1+2*i2+i3+3*i4;
    if pay >= 100
     f1(k)=3*i1+4*i2+2*i3+2*i4
     f2(k)=3*i1+2*i2+2*i3+i4;
     k=k+1;
    end
   end
  end
 end
end
k=k-1
plot(f1(1:k),f2(1:k),'b.');
```



- Point $\mathbf{b}^* \in \Omega$ is a non-inferior solution if for some neighborhood of $\mathbf{b}^*$ there does not exist a $\Delta\mathbf{b}$ such that $(\mathbf{b}^* + \Delta\mathbf{b}) \in \Omega$

$$f_i(\mathbf{b}^* + \Delta\mathbf{b}) \geq f_i(\mathbf{b}^*) \qquad i = 1, \ldots, m$$

$$f_j(\mathbf{b}^* + \Delta\mathbf{b}) < f_j(\mathbf{b}^*) \qquad \text{for some } j$$



- Multi-objective optimization: generation and selection of non-inferior solution points

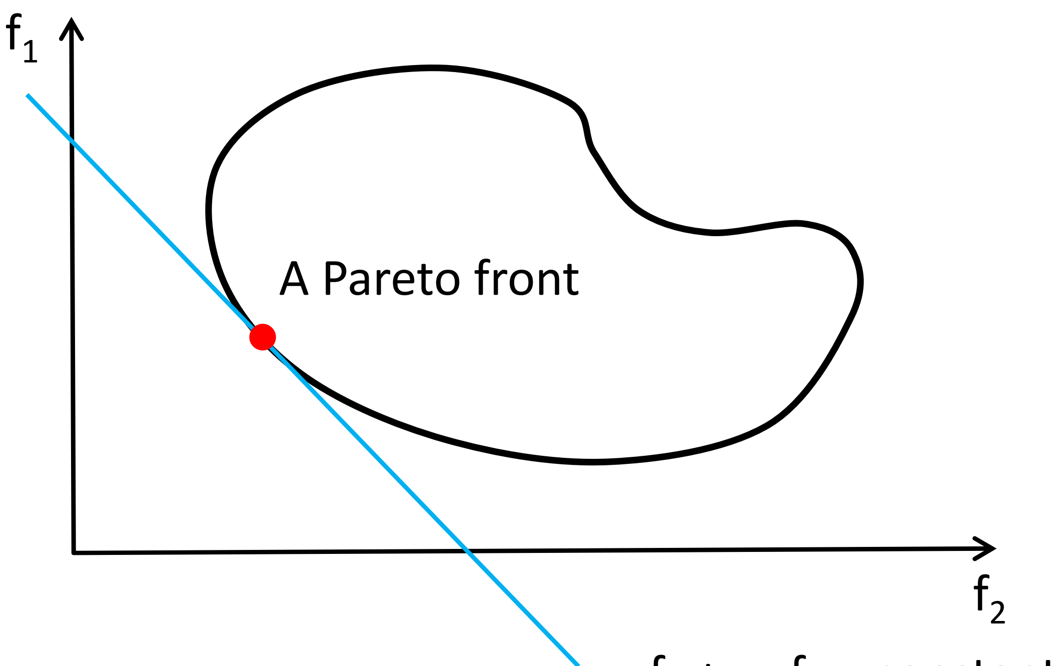
SOLUTION METHODS

- Methods that try to avoid generating the Pareto front
  - Generate "utopia point"
  - Define optimum based on some measure of distance from utopia point
- Generating entire Pareto front
  - Weighted sum of objectives with variable coefficients
  - Optimize on objective for a range of constraints on the others
  - Niching methods with population based algorithms

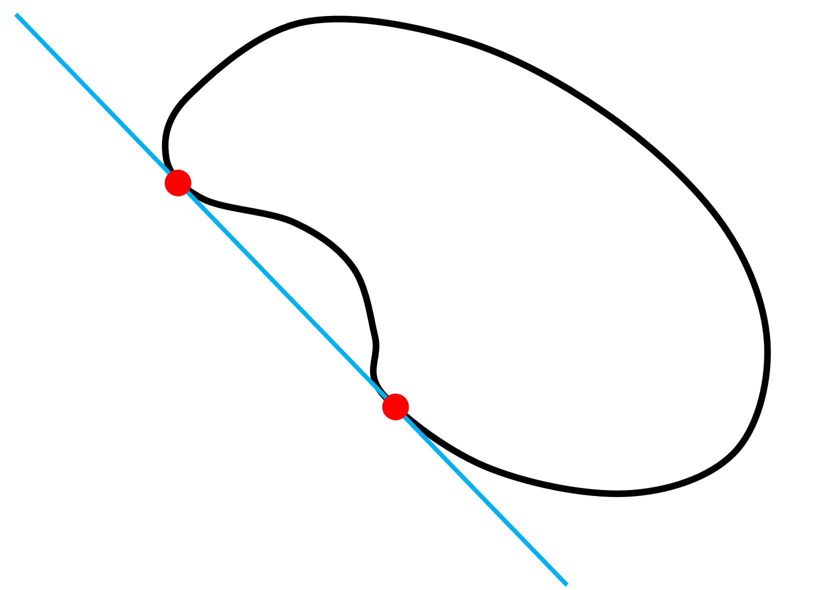
- Convert **F**(**b**) to a scalar objective function using weights

$$\text{Minimize} \quad f(\mathbf{b}) = \sum_{i=1}^{m} w_i f_i(\mathbf{b})$$

- By changing w_i, different Pareto optimum points can be found



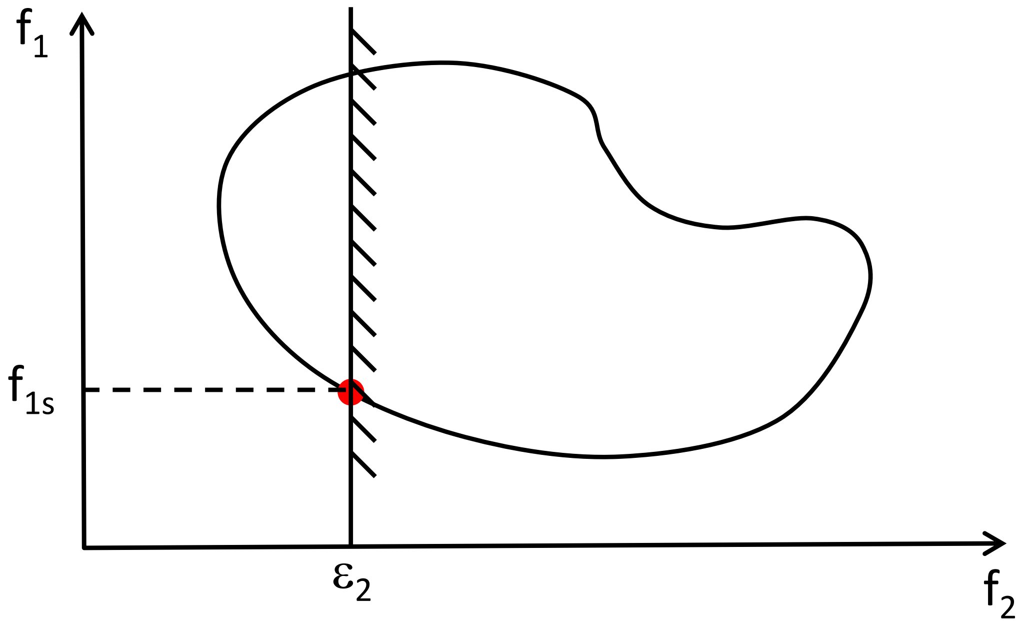
$w_1 f_1 + w_2 f_2 = \text{constant}$



Problem with non-convex front

EPSILON-CONSTRAINT METHOD

- Optimize a primary objective $f_p(\mathbf{b})$, while other objectives are considered as constraints



$$\text{Minimize} \quad f_p(\mathbf{b})$$

$$\text{subject to } f_i(\mathbf{b}) \le \varepsilon_i, \quad i \ne p$$

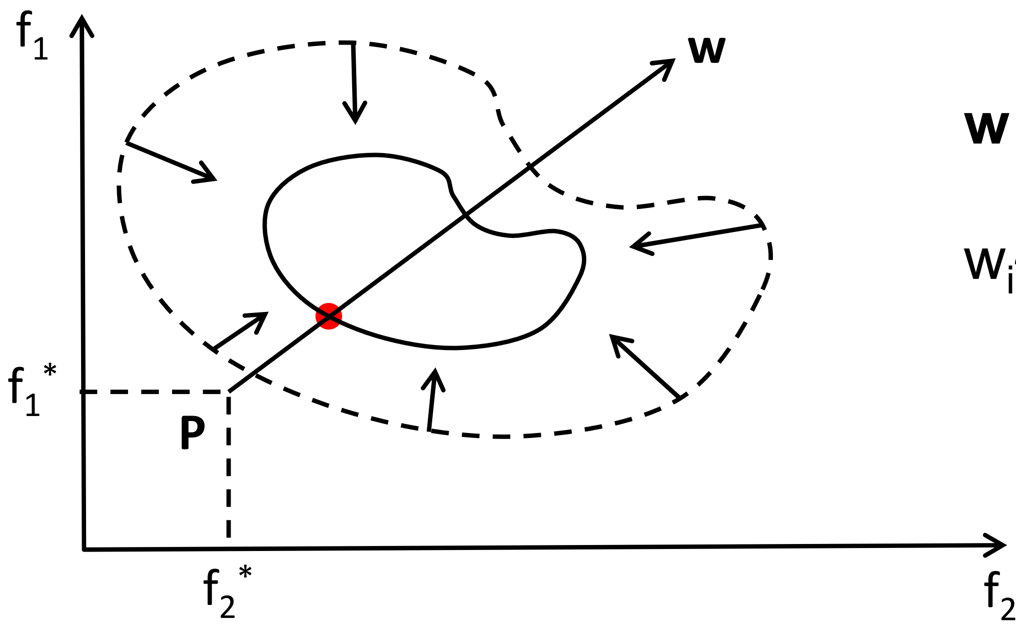
- Can find non-convex front
- If ε_2 is too small, there is no feasible point

- Try to achieve the design goal *f** using weighted relaxation

$$\text{Minimize} \quad \gamma$$

$$\text{subject to } f_i(\mathbf{b}) - w_i\gamma \le f_i^*$$

- Geometrically, starting from *P* = *f**, find a feasible point in the direction of w



$\mathbf{w} = [w_1, w_2, \ldots, w_m]$: weight

$w_i\gamma$: slackness

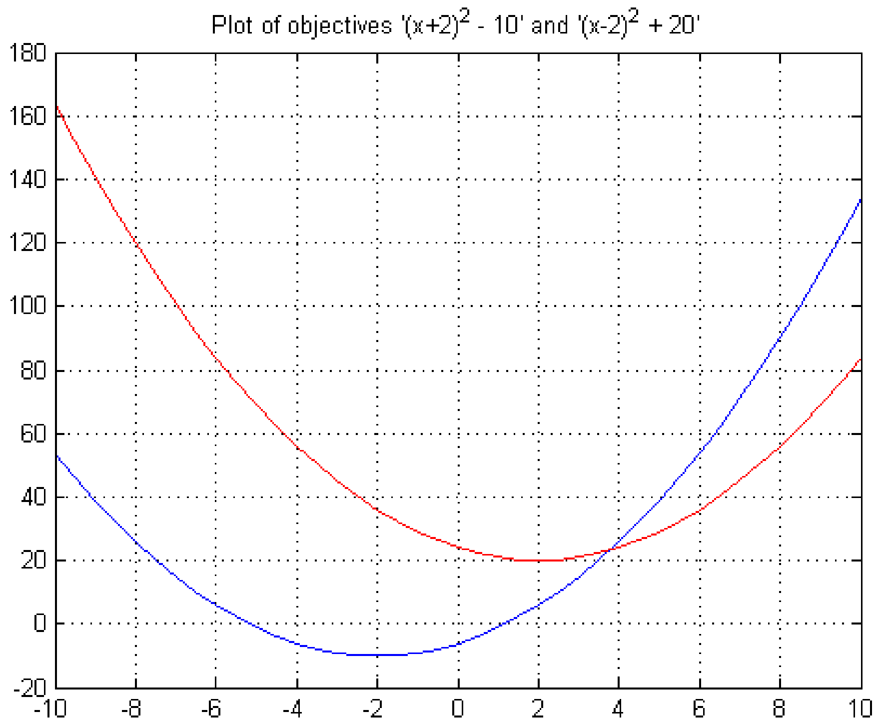
- Multiobjective optimization using the genetic algorithm

$$\text{Minimize} \quad \mathbf{F}(b) = [f_1(b),\ f_2(b)]$$

$$\text{where} \quad f_1(b) = (b+2)^2 - 10$$

$$f_2(b) = (b-2)^2 + 20$$

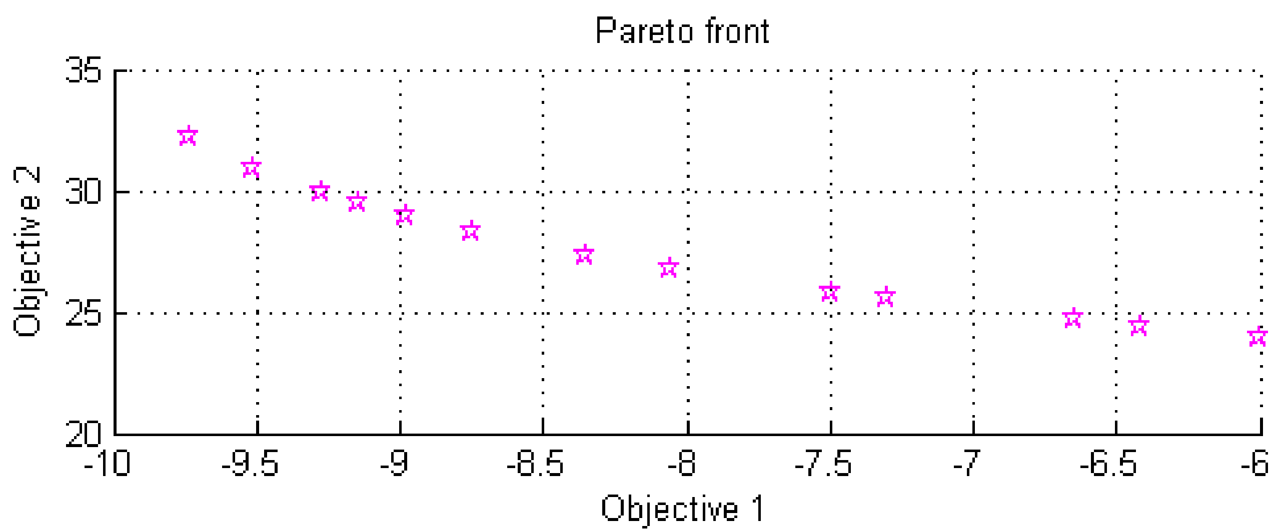
$$-1.5 \le b$$



MATLAB EXAMPLE

```
function y = simple_multiobjective(b)
y(1) = (b+2)^2 - 10;
y(2) = (b-2)^2 + 20;

Fitness = @simple_multiobjective;
NV = 1; lb = -1.5; ub = 0;
options = gaoptimset('PlotFcns',{@gaplotpareto,@gaplotscorediversity});
gamultiobj(Fitness, NV,[],[],[],[],lb, ub, options);
```



UF UNIVERSITY of FLORIDA
The Foundation for The Gator Nation

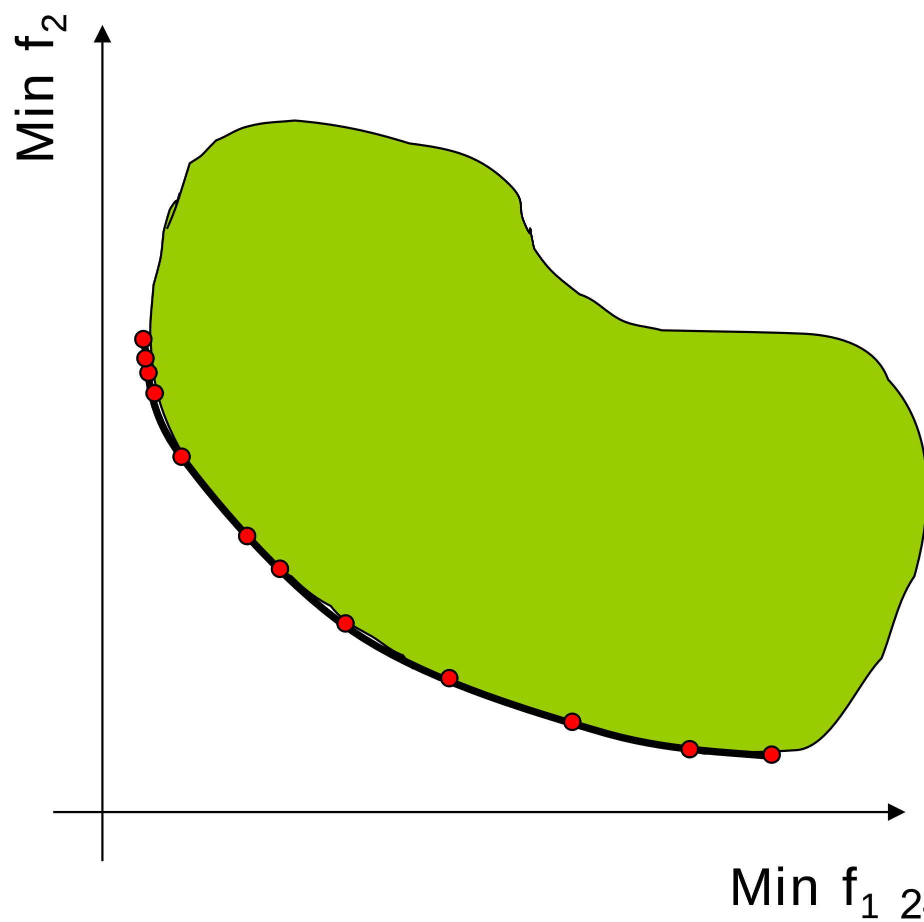
Elitist Non-dominated Sorting Genetic Algorithm: NSGA-II

Multi-objective optimization problem

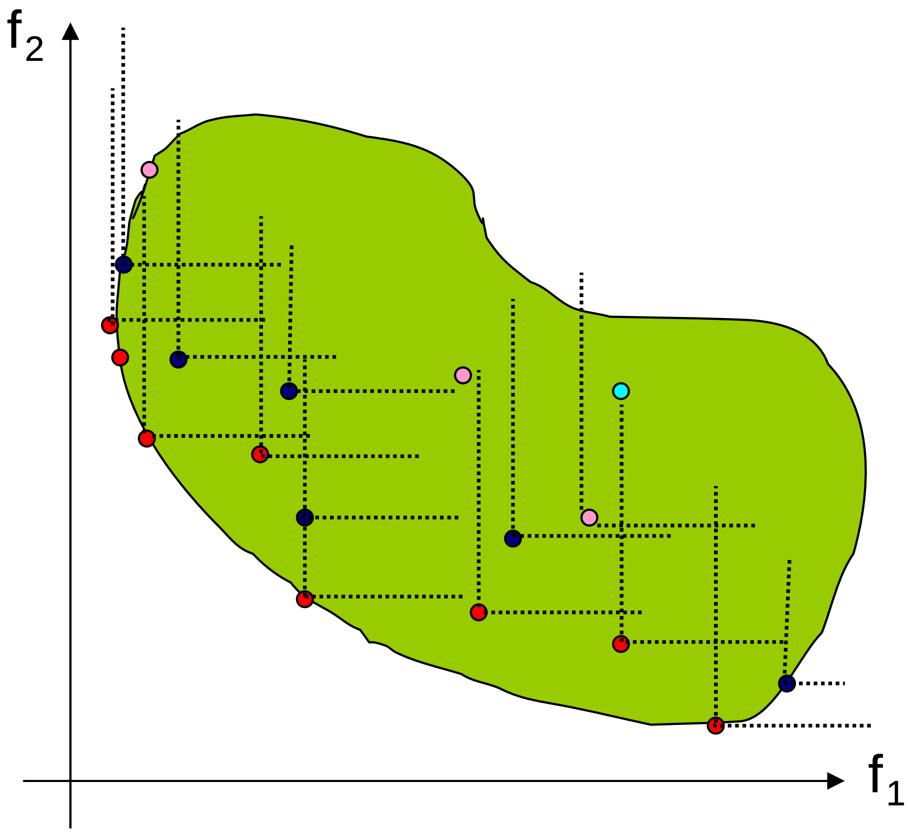
- Problems with more than one objective – typically conflicting objectives
  - Cars: Luxury vs. price
- Mathematical formulation
- Minimize $\mathbf{F}(\mathbf{b})$
  - where $\mathbf{F}(\mathbf{b}) = \{f_i, i = 1, \cdots, M\}$
    $\mathbf{b} = \{b_i, j = 1, \cdots, N\}$
- Subject to
  - $\mathbf{g}(\mathbf{b}) \leq 0, \mathbf{g} = \{g_k, k = 1, \cdots P\}$
  - $\mathbf{h}(\mathbf{b}) = 0, \mathbf{h} = \{h_l, l = 1, \cdots Q\}$

PARETO OPTIMAL FRONT

- Many optimal solution
- Usual approaches: weighted sum strategy, multiple-constraints modeling
- Alternative: Multi-objective GA
- Algorithm requirements:
  - Convergence
  - Spread

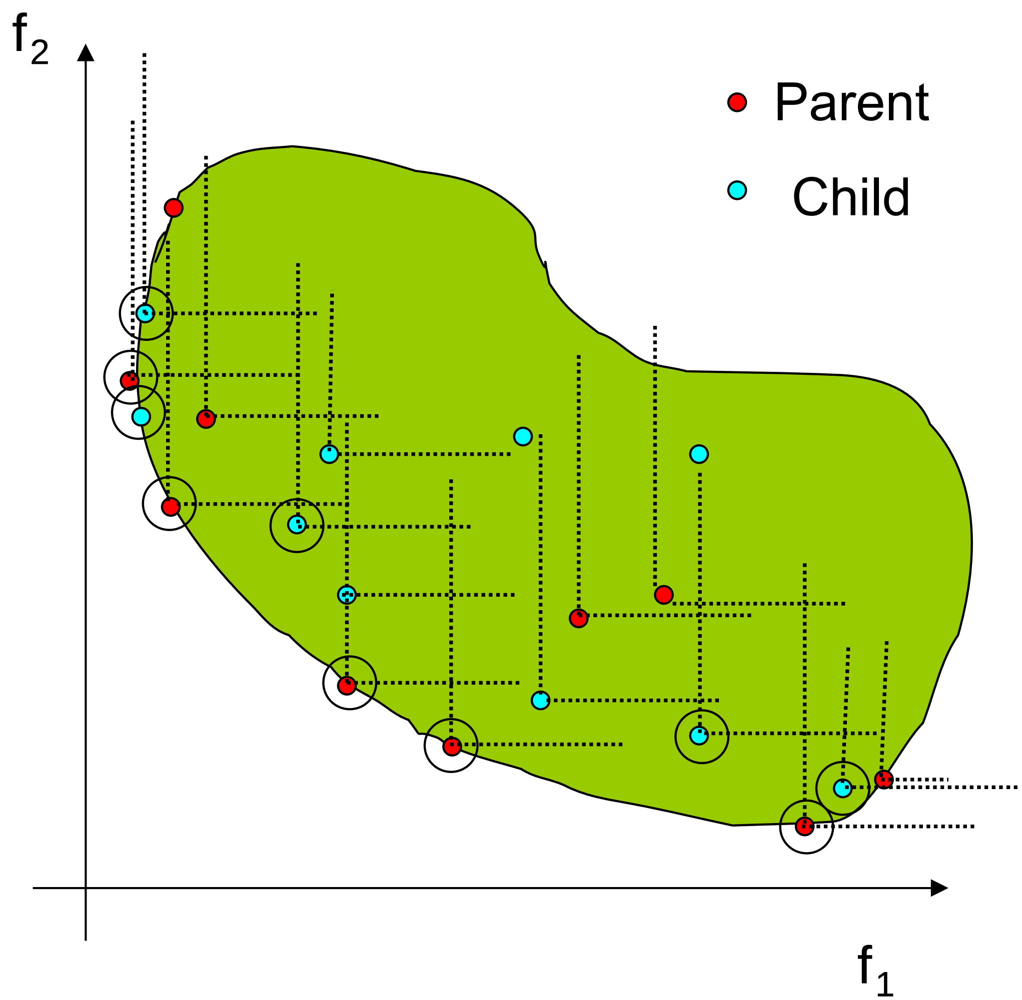


- Children and parents are combined.
- Non-dominated points belong to first rank.
- The non-dominated solutions from the remainder are in second rank, and so on.



ELITISM

- Elitism: Keep the best individuals from the parent and child population

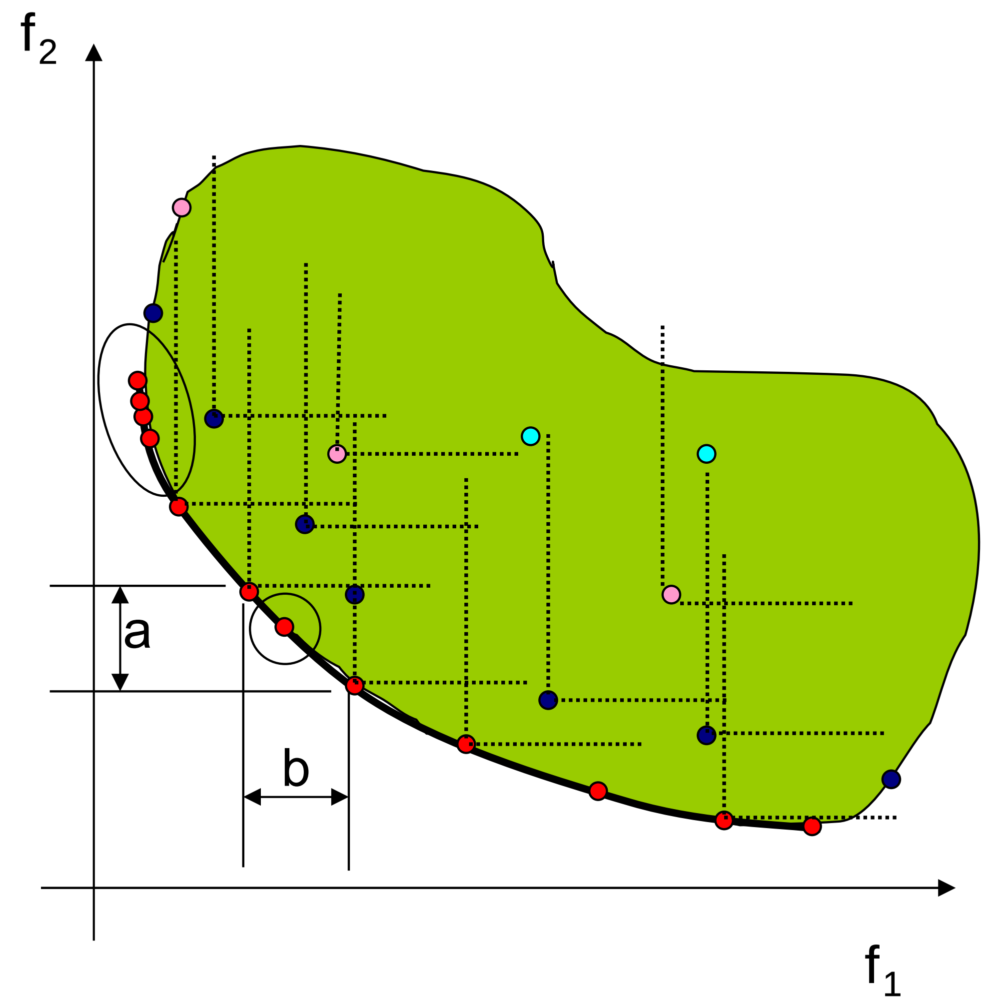


NICHING FOR LAST RANK

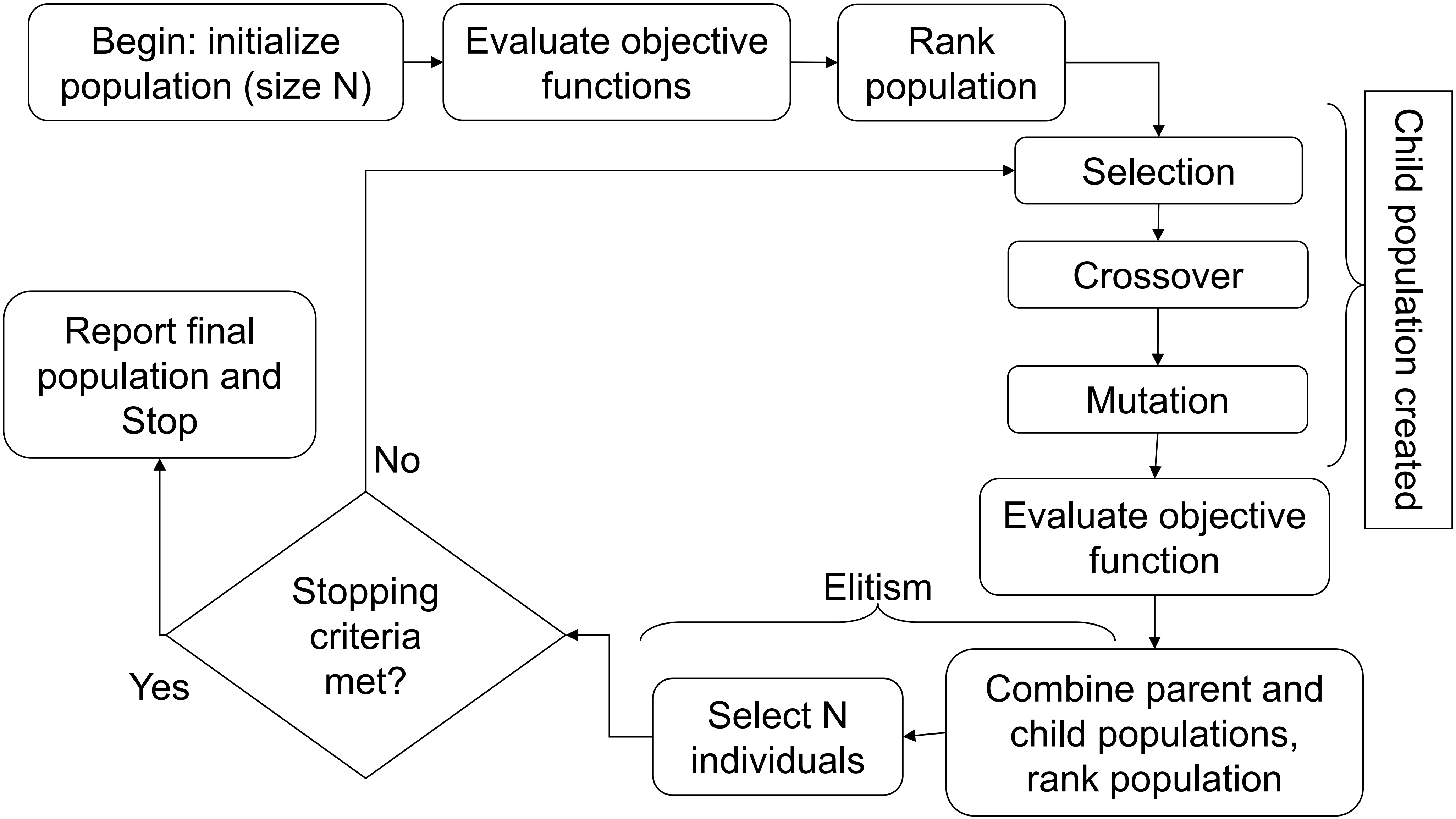
- Niching is an operator that gives preference to solutions that are not crowded
- Crowding distance

  $c = a + b$

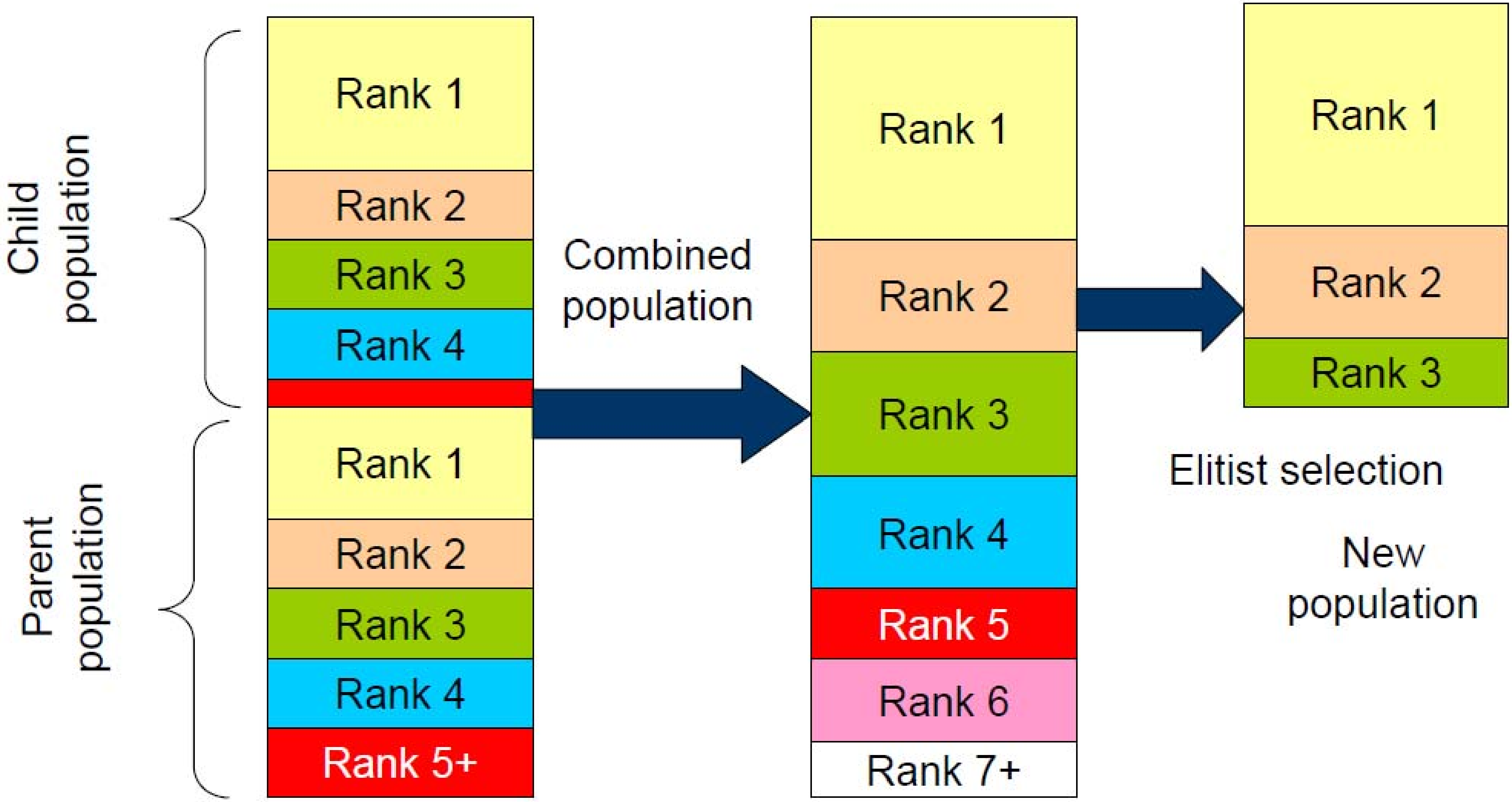
- End points have infinite crowding distance
- Solutions from last rank are selected based on niching



Flowchart of NSGA-II



Elitism process



Problems NSGA-II

- Sort all the individuals in slide 263 into ranks, and denote the rank on the figure in the slide next to the individual.
- Describe how the 10 individuals were selected, and check if any individuals were selected based on crowding distance.

Example: Bicycle frame design

- Objectives
  - Minimize area
  - Minimize max. deflection
- Constraints
  - Components should be a valid geometry
  - Max. stress ≤ Allowable stress
    $\sigma_{max} \leq \sigma_{allowable}$
  - Max. deflection ≤ Allowable deflection
    $\delta_{max} \leq \delta_{allowable}$

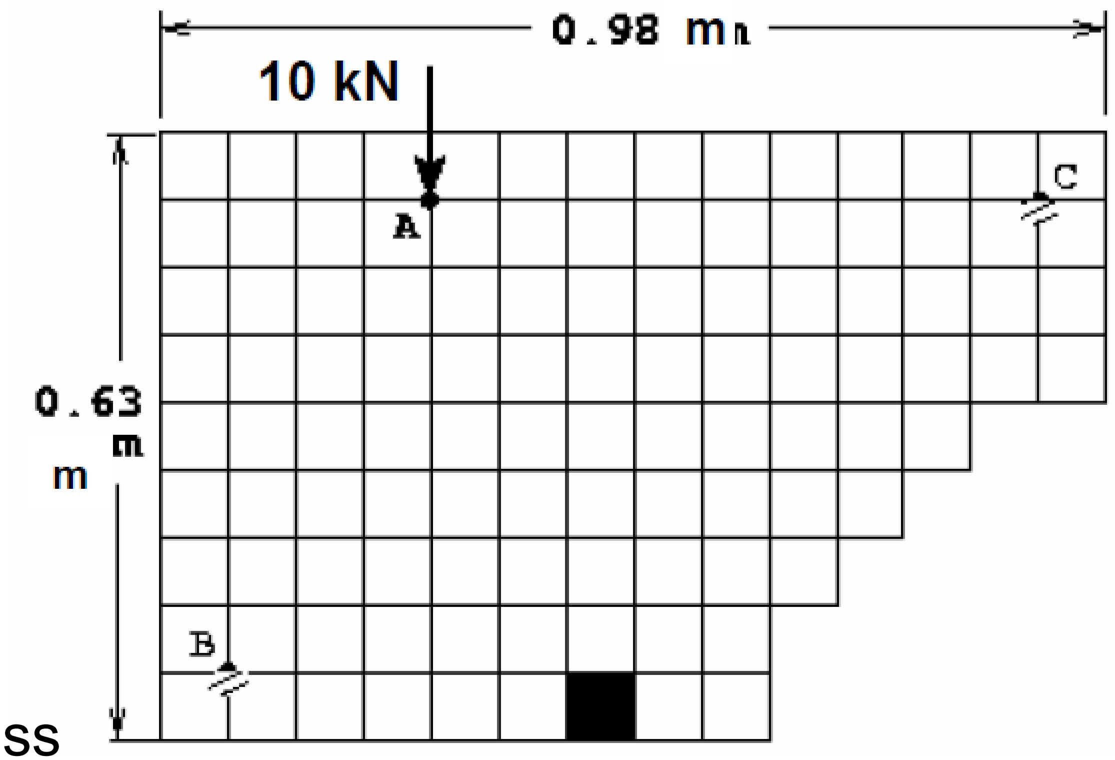


Plate thickness = 20 mm

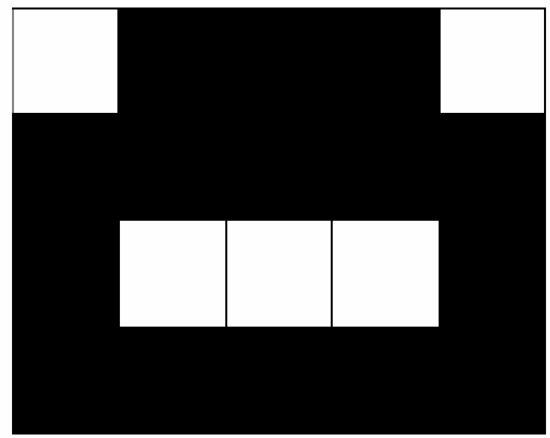
Problem modeling

- Shapes are represented by binary strings, where ‘0’ represents void region and ‘1’ represents material region
- Example : A typical binary string is

01110 11111 10001 11111

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

Left to right representation



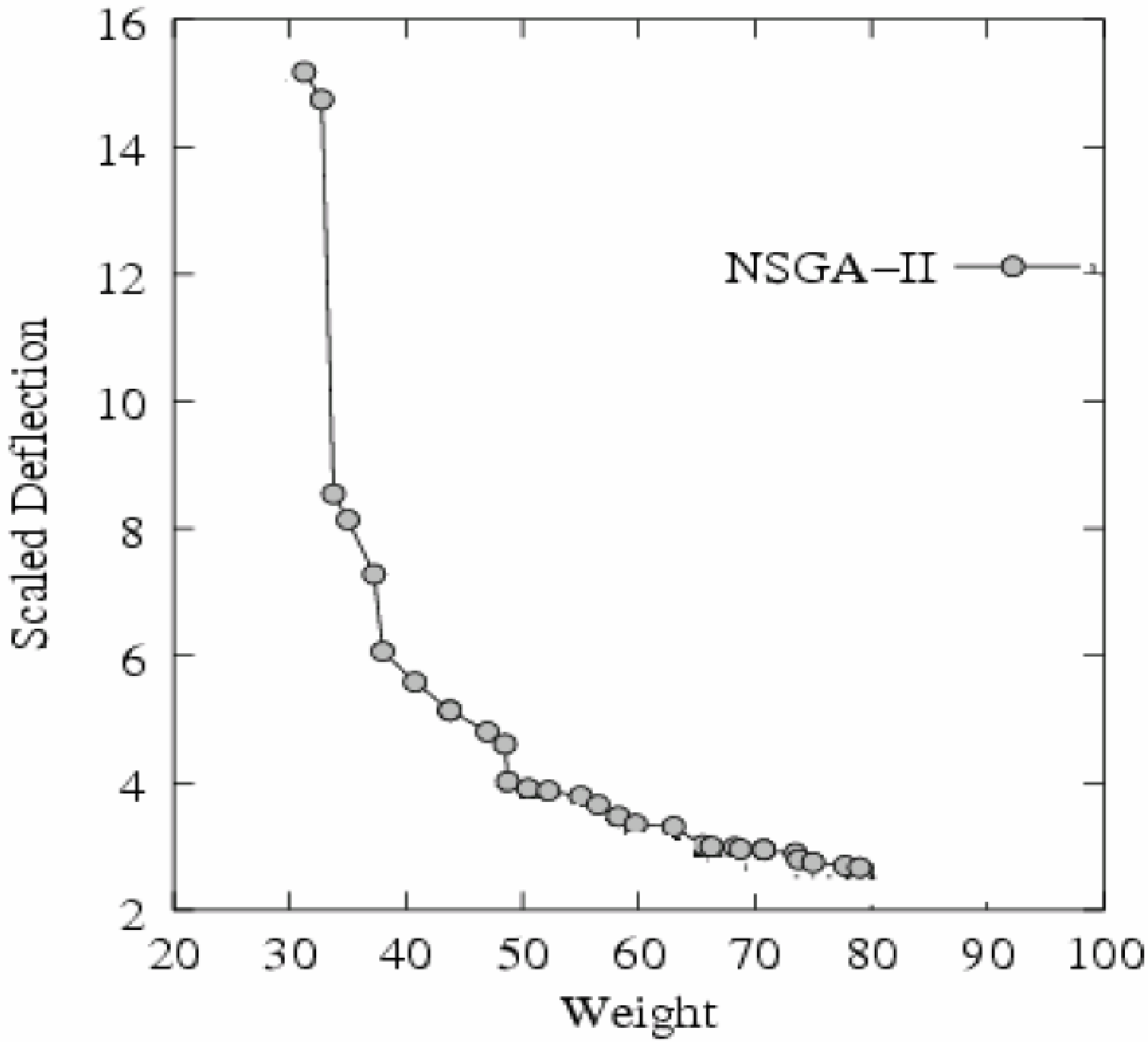
Shape corresponding to binary string

Material properties and GA parameters

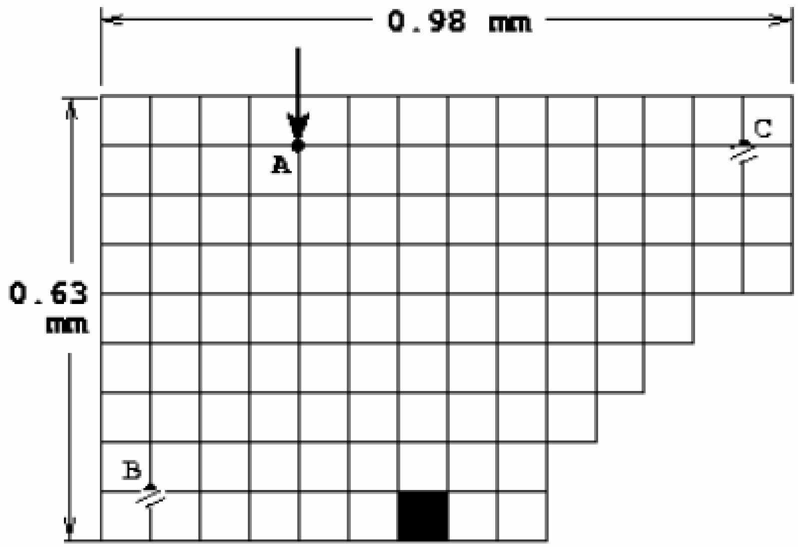
- Material Properties
  - Yield Stress ($\sigma_{allowable}$) 140 MPa
  - Max Deflection ($\delta_{allowable}$) 5 mm
  - Young's Modulus (E) 80GPa
  - Poisson's Ratio (ν) 0.25
- GA Parameters
  - Binary String Size (L) 14x9
  - Population Size 30
  - Crossover Probability 0.95
  - Mutation Probability 1/L
  - # of Generations 150

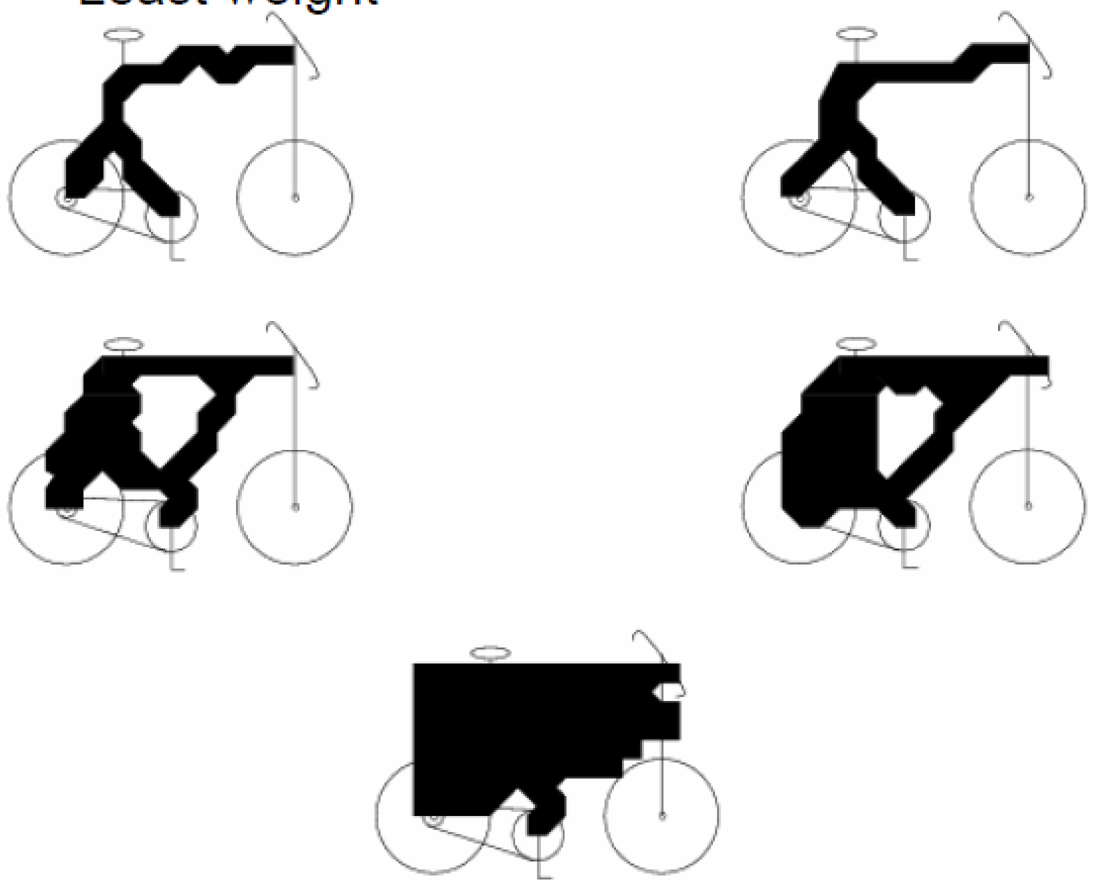
Pareto optimal front

- Small increase in weight leads to a large drop in deflection
- Similarly small change in deflection allows significant reduction of the weight



Optimal shapes





Different conceptual designs can be found!

Some more engineering applications

- Structural designs of mechanical components
- Design of turbo-machinery components
- Bioinformatics – protein unfolding
- VLSI circuit designs
- Packaging
- …

Other related topic of interest

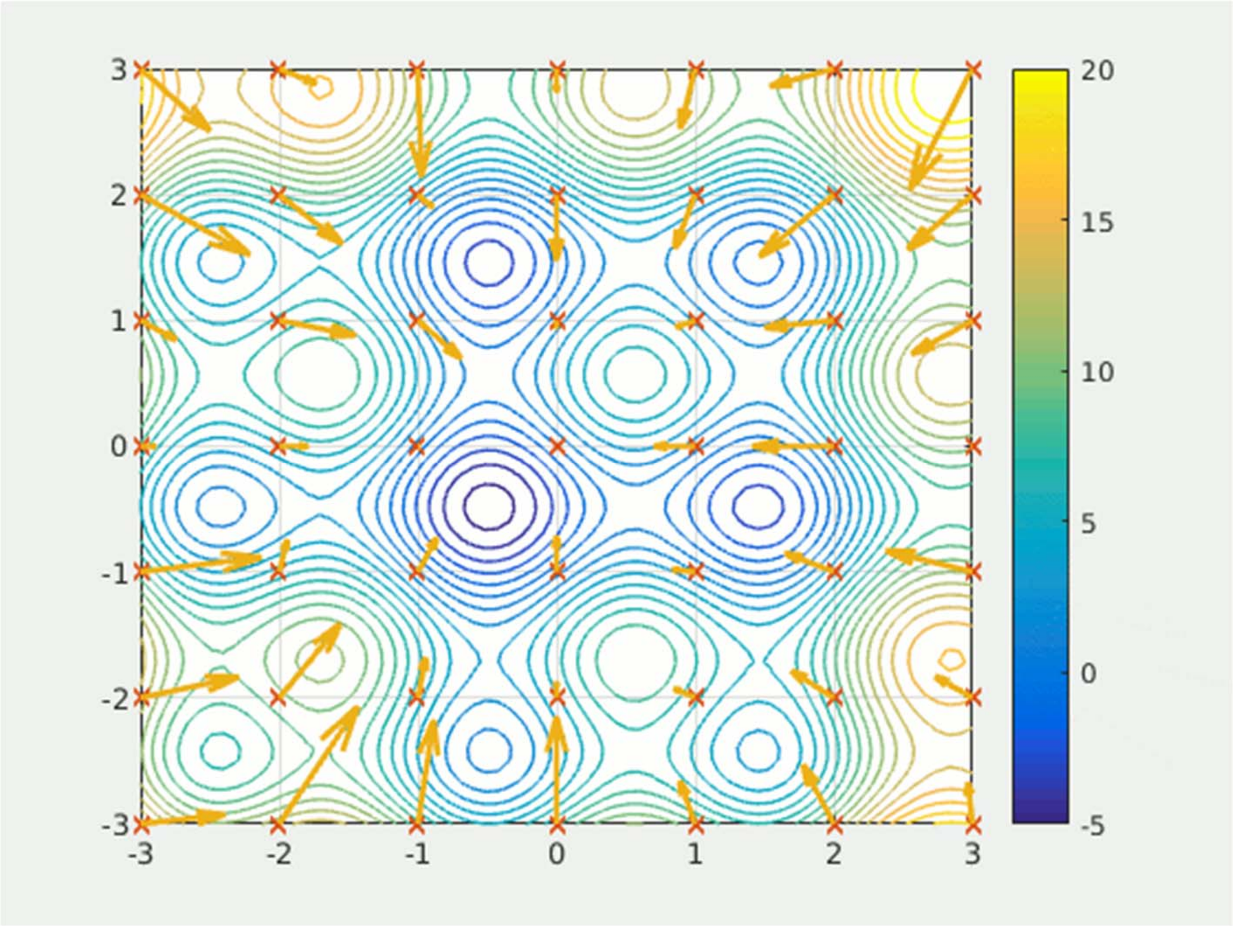
- Real-coded genetic algorithms
- Other multi-objective evolutionary algorithms
  - Pareto archived evolutionary strategies (PAES)
  - Strength Pareto evolutionary algorithm (SPEA)
  - ϵ–multi-objective evolutionary algorithm (ϵ-MOEA)
- Hybrid GAs
- Particle swarm algorithms
- Ant colony optimization

Constrained Particle Swarm Optimization

Is animal behavior optimized?

Learn from the social behavior of a bird flock or fish

- Mimicking social behavior of insects or birds

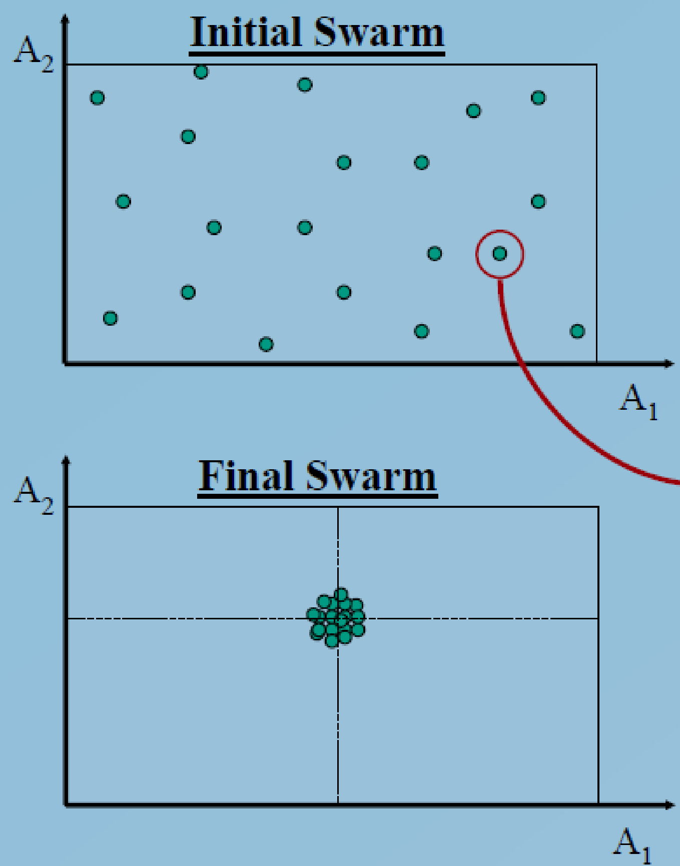


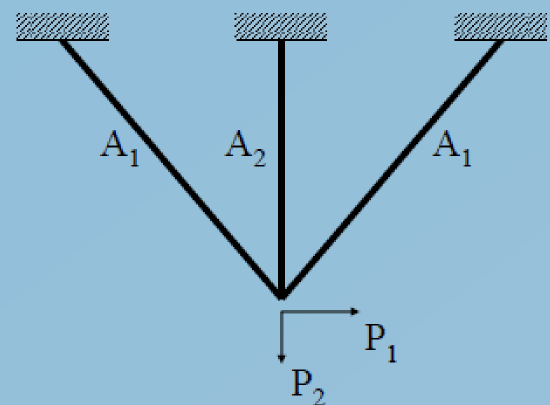
CONSTRAINED PARTICLE SWARM OPTIMIZATION

- Based on Venter, G. and Haftka, R.T., (2010), Structural and Multidisciplinary Optimization, Vol. 40(1-6), 65-76.
- Lecture will cover particle swarm optimization, a good global search algorithm for continuous problems.
- Constraints are treated using a bi-objective approach that minimizes both the objective function and a measure of the violation of the constraints.

PARTICLE SWARM OPTIMIZATION

- Based on a simplified social model: swarm adapts to underlying environment by returning to promising regions previously found
- Robust algorithm
- Global optimizer
- Easy to implement
- Parameter tuning
- High computational cost
- Unconstrained algorithm





- **Particle P_i properties:**
  - **Position:** (A_{1j}, A_{2j})
  - **Velocity Vector**
  - **Objective:** $W(A_{1j}, A_{2j})$
  - **Constraints:** $S(A_{1j}, A_{2j}, P_1, P_2)$

MOVING TOWARDS PERSONAL BEST AND GROUP BEST

Position Update:

$$\boldsymbol{x}^i_{k+1} = \boldsymbol{x}^i_k + \boldsymbol{v}^i_{k+1}\Delta t$$

$\mathbf{x}^i_k$ = position of *ith* particle in *kth* step

$\mathbf{p}^i$ = best past position of *ith* particle

$\mathbf{p}^g_k$ = best current position of swarm

Velocity Vector:

$$\boldsymbol{v}^i_{k+1} = \underbrace{\boxed{w\boldsymbol{v}^i_k}}_{\text{Inertia}} + \underbrace{\boxed{c_1 r_1 \frac{\boldsymbol{p}^i - \boldsymbol{x}^i_k}{\Delta t}}}_{\text{Trust in self}} + \underbrace{\boxed{c_2 r_2 \frac{\boldsymbol{p}^g_k - \boldsymbol{x}^i_k}{\Delta t}}}_{\text{Trust in swarm}}$$

YouTube visualization www.youtube.com/watch?v=_Y4A8Q2j0wA

EXERCISE: PARTICLE SWAM OPTIMIZATION

- Minimize the Rosenbrock Banana function using PSO starting with 30 particles distributed randomly in the region -$-2 \leq b_1 \leq 2, -1 \leq b_2 \geq 3$. Use $w = 0.5, c_1 = 1.75, c_2 = 2.25$.

WHAT IS BEST WHEN CONSTRAINTS ARE PRESENT?

- Constrained optimization problem

$$\text{Minimize} \quad f(\mathbf{b})$$
$$\text{subject to } g_j(\mathbf{x}) \leq 0 \quad j = 1, \ldots, K$$

- Penalty function approach

$$\bar{f}(\mathbf{b}) = f(\mathbf{b}) + \omega \sum_{j=1}^{K} \max(0, g_j(\mathbf{b}))$$

- Not easy to pick good ω
- Creates canyons in design space that can trap swarm

BI-OBJECTIVE OR FILTER APPROACH

- Replace

$$\bar{f}(\mathbf{b}) = f(\mathbf{b}) + \omega \sum_{j=1}^{K} \max(0, g_j(\mathbf{b}))$$

- By

$$\text{Minimize} \quad f(\mathbf{b})$$

$$\text{Minimize} \quad h(\mathbf{b}) = \sum_{j=1}^{K} \max(0, g_j(\mathbf{b}))$$

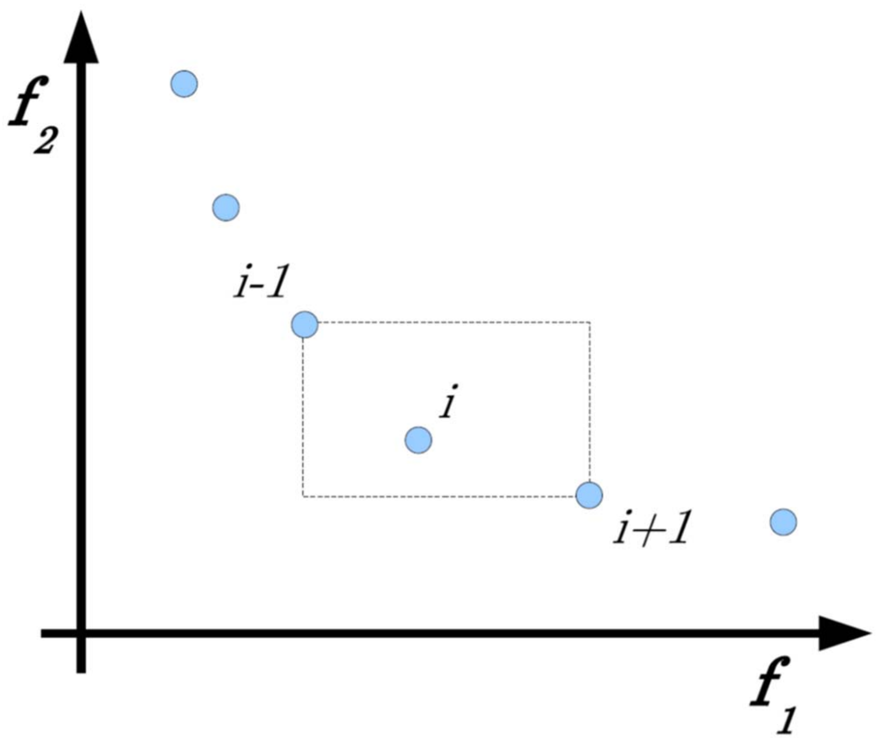
- Fletcher R., Leyffer, S. (2002) Nonlinear programming without a penalty function, Math Program 91(2);239-269
- Advantage is in allowing less constrained search in design space.

MULTI-OBJECTIVE PSO

- The bi-objective formulation requires a multi-objective PSO (MOPSO)
- Each iteration has multiple equally good “leaders”, non-dominated solutions
- Based on general algorithm by Reyes-Sierra and Coello Coello (2005)
- Create archive of non-dominated solutions
- Select leader for each particle based on a binary tournament
- Makes use of the crowding distance to maintain and select leaders

CROWDING DISTANCE

- First sort non-dominated points by objective function values.
- Crowding distance of point *i* is the distance between two nearest neighbors (*i*-1, to *i*+1)

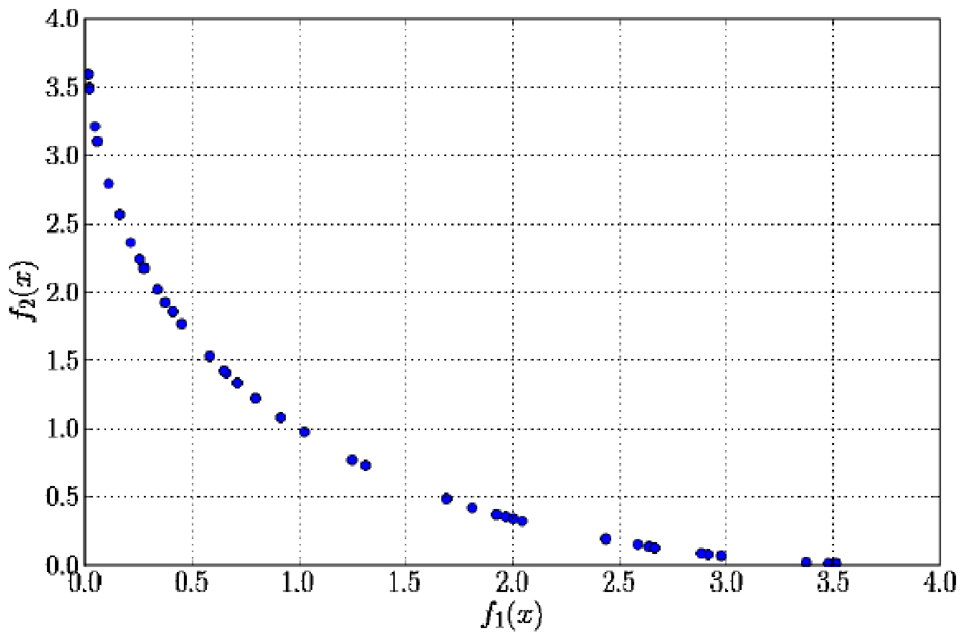


EXAMPLE: MOPSO

- MOPSO with two objective functions

$$f_1(b) = b^2 \quad f_2(b) = (b-2)^2 \qquad b \in [-100, 100]$$

MOPSO: Twenty particles, forty iterations



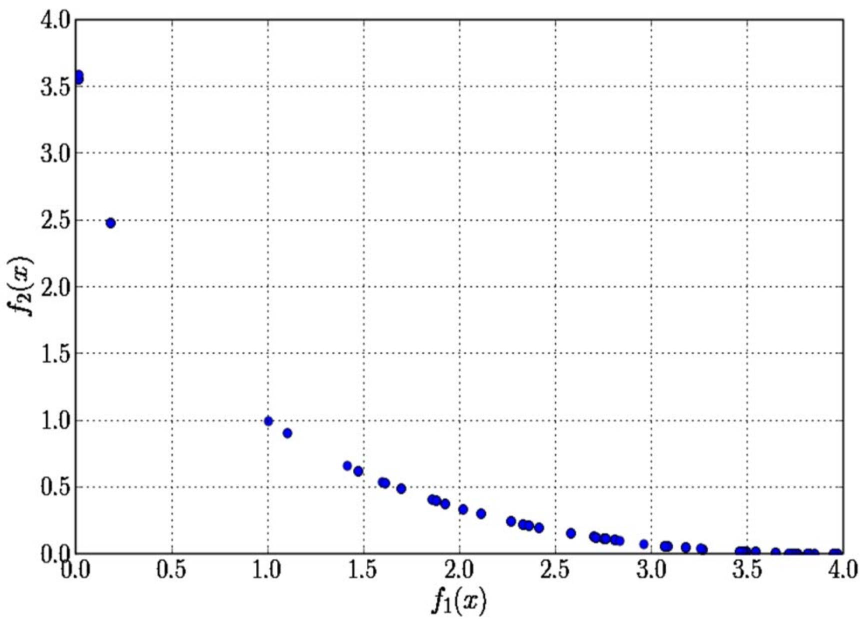
With this large range of b, how come the two objectives have such small values?

EXERCISE: CROWDING DISTANCE

- Generate 100 random b values in [0,2] and the corresponding values of $f_1(b)$ and $f_2(b)$ of the functions on the previous slide. Choose a point with the maximum crowding distance, then the next point with the maximum crowding distance, and so on until you reach 20 points. Plot the resulting Pareto front. Why is it the Pareto front?

CONSTRAINT SPECIALIZED BI-OBJECTIVE PSO

- We are not interested in complete Pareto front, but mostly in part that has low constraint violations.
- Bias algorithm to such points
  - Select leaders based on both constraint violation and crowding distance
  - Binary tournament based on constraint violation
- Goal is to place more particles near feasible domain



EXAMPLE

- Composite laminate design $(\theta_1/\theta_2/\theta_3)_s$, with three angles and three thicknesses as design variables.
  – Maximize the transverse in-plane stiffness A_{22}
  – Constraints on ply angles and effective Poisson's ratio
- Methods compared
  – Standard PSO with penalty function
  – MOPSO algorithm of Reyes-Sierra and Coello-Coello
  – Modified MOPSO algorithm
- 100 runs, each with 30 particles and 100 generations.
- 10% probability of mutation, $w = 0.5$, $c_1 = 1.75$, $c_2 = 2.25$

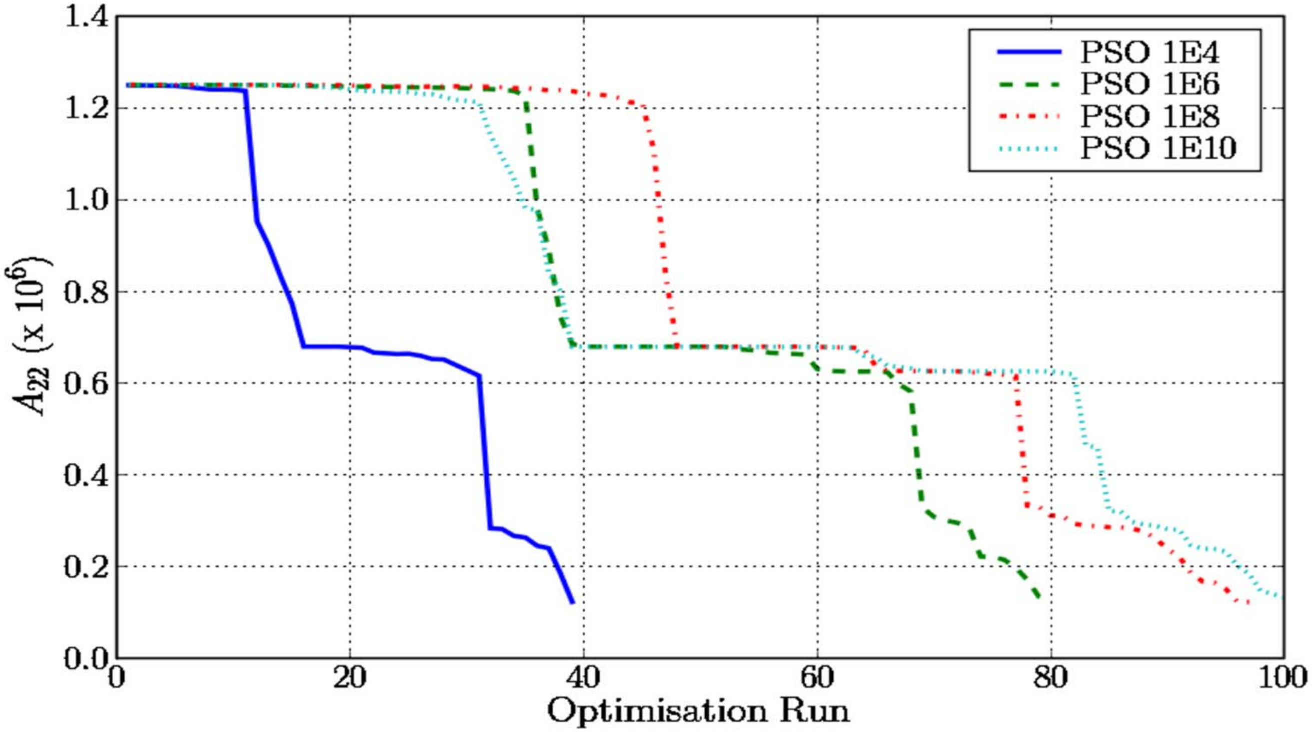
OPTIMIZATION FORMULATION

- See relation between stiffness and Poisson's ratio and design variables in the paper.

$$
\begin{aligned}
&\text{Maximize } A_{22} \\
&\text{subject to } 0.48 \le \nu_{eff} \le 0.52 \\
&\qquad -5^o \le \theta_k \le 5^o \qquad \text{or} \\
&\qquad 40^o \le \theta_k \le 50^o \qquad \text{or} \\
&\qquad 85^o \le \theta_k \le 95^o \\
&\qquad 0.001 \le t_k \le 0.05"
\end{aligned}
$$

RESULTS: PENALTY FUNCTION

- 10^8 is the best penalty multiplier



- Without focus of the Pareto front on the near feasible designs, there are many wasted evaluations on deeply infeasible designs.