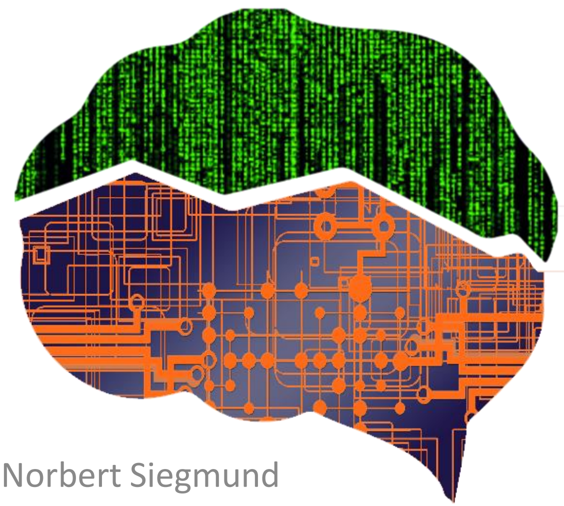
Search-Based Software Engineering

Multi-Objective Optimization



Prof. Dr.-Ing. Norbert Siegmund
Intelligent Software Systems

Bauhaus-Universität Weimar

Black Box Optimization Competition

BBComp

Contents

- Background
- Goals and Scope
- How to Participate
- Competition Rules
- Important Dates
- Evaluation Criteria
- Problem Statistics
- Downloads
- Results
- Documentation
- Frequently Asked Questions
- Related Material

Login*

username

password

login

* Please write an email to blackboxcompetition@gmail.com to register an account. Check the section How to Participate.

https://bbcomp.ini.rub.de/

For the Impatient

What is BBComp?

BBComp — the black box optimization competition — is the first competition in continuous black-box optimization where test problems are truly black boxes for participants. It is also the first web/online optimization competition in the direct search domain.

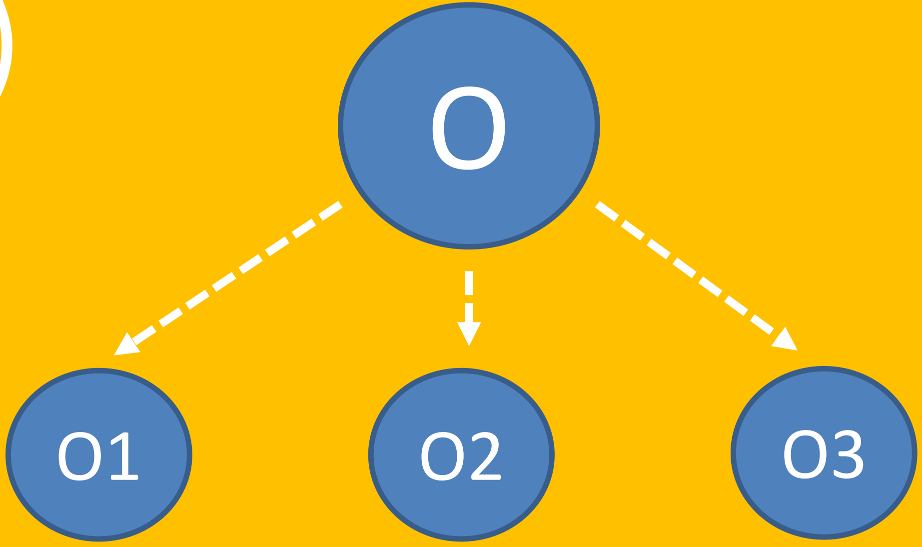
Quick links: [Downloads] [Rules] [FAQ] [Documentation] [Results]

Recap

- Exploit more to improve genetic and evolution algorithms
  - Ellitism
  - Hybrid approaches (ES+HC)
  - Memetic algorithms

- Differential Evolution:
  - Survival selection (do selection of the bred children)
  - Adaptive mutation based on variance in the population
- Particle swarm optimization:
  - Particles store position, velocity, and best positions
  - Particles move based on the velocity and neighbors’ best solutions

Multi-Objective Optimization (MOO)



Introduction to MOO

- A practical problem: Optimize not for a single, but for multiple objectives
  - “I want a cheap, luxurious, fuel-efficient, fast, good-looking family car.”
  - “I want a software system that is fast, reliable, energy-efficient, secure, easy-to-use, bug free, and with low resource consumption.”
  - “I want a set of test cases that cover all paths in my software, are fast to execute, reveal all bugs very quickly, and are easy to understand and maintain.”

Find the best trade-off among multiple, possibly opposing objectives

MOO Formalization

Minimize $O(\vec{x}) = [O_1(\vec{x}), O_2(\vec{x}), \dots, O_k(\vec{x})]$

Subject to $G(\vec{x}) = [g_1(\vec{x}), g_2(\vec{x}), \dots, g_m(\vec{x})] \geq 0$

$H(\vec{x}) = [h_1(\vec{x}), h_2(\vec{x}), \dots, h_r(\vec{x})] = 0$

$\mathrm{x}_\mathrm{i}^\mathrm{L} \leq x_i \leq x_i^U, i = 1, \dots, n$

Where

$\vec{x} = \langle x_1, x_2, \dots, x_n \rangle^T$ is a vector of decision variables;
k is the number of objectives O_i;
m inequality and r equality constraints
x_i^L and x_i^U are respectively the lower and upper bound for each decision variable x_i

Defining the Objective

- Objective might be a vector F of k system responses or characteristics, we are trying to maximize or minimize

$$O = \begin{bmatrix} O_1 \\ O_2 \\ O_3 \\ O_i \\ \dots \\ O_k \end{bmatrix} = \left.\begin{bmatrix} cost\ in\ € \\ -\,range\ in\ km \\ weight\ in\ kg \\ -\,response\ time\ in\ s \\ \dots \\ -\,ROI\ in\ \% \end{bmatrix}\right\}$$

How to compute O?

Simple solution: Weighted sum

Naïve: Weighted Sum

- Idea: Define a linear function to combine all objectives
  - $O = \omega_1 * O_1 + \omega_2 * O_2 + \ldots + \omega_k * O_k$
  - Example: $F = 2 * performance + 5 * security + 0.5 * reliability + 1.3 * energy\ consumption$
- Problems:
  - How to define the weights or how to express how much an objective is more worth than another one?
  - What if the objectives are non-linear (i.e., the performance difference between 2-3s is of lower interest than the performance difference between 8-9s)?
  - How to encode different value ranges of the objectives?
  - Can we move toward the actual trade-off area of interest?

Naïve: Preference Ranking

- Idea: Rank the objectives according to their importance
  - Individual x is better than y if it is superior in a higher ranked objective; if similar, go to the next objective and repeat
  - When comparing two individuals, go through the objectives from most to least important until we find one is clearly superior to the other one

$Best \leftarrow$ individual picked at random from population with replacement
$O \leftarrow \{O_1, O_2, \dots, O_k\}$ objectives
$t \leftarrow$ tournament size, $t \geq 1$
for i from 1 to t **do**
  $Next \leftarrow$ individual picked at random from population with replacement
  for j from 1 to k **do**
    if $ObjectiveValue(O_j, Next) > ObjectiveValue(O_j, Best)$ **then**
      $Best \leftarrow Next$; **break**
    else if $ObjectiveValue(O_j, Next) < ObjectiveValue(O_j, Best)$ **then**
      break
return $Best$

Adaptations to Preference Ranking

- (1) Pick objective at random each time to use for fitness
- (2) Use voting: An individual is preferred if it is better in *more* objectives than another one

$Best \leftarrow$ individual picked at random from population with replacement
$O \leftarrow \{O_1, O_2, \ldots, O_k\}$ objectives
$t \leftarrow$ tournament size, $t \geq 1$
for i from 1 to t **do**
  $Next \leftarrow$ individual picked at random from population with replacement
  $c \leftarrow 0$
  for each objective $O_j \in O$ **do**
    if $ObjectiveValue(O_j, Next) > ObjectiveValue(O_j, Best)$ **then**
      $c \leftarrow c + 1$
    else if $ObjectiveValue(O_j, Next) < ObjectiveValue(O_j, Best)$ **then**
      $c \leftarrow c - 1$
  if $c > 0$ **then**
    $Best \leftarrow Next$
return $Best$

Adaptations to Preference Ranking

- (3) Entrance-Based Tournament Selection
  - Tournament based on one objective
  - The individuals applicable for a tournament selection are selected using tournament selection of a second objective, and recursively further till all objectives have been covered

Entrance-Based Tournament Selection

Different weights are possible

$O \leftarrow \{O_1, O_2, \dots, O_k\}$ objectives
$T \leftarrow \{T_1, T_2, \dots, T_k\}$ tournament sizes for the individual objectives in O, all ≥ 1
return $ObjectiveTournament(O, T)$

Recursive function with changing sets of objectives and tournament sizes

procedure $ObjectiveTournament(O, T)$
  $Best \leftarrow$ individual picked at random from population with replacement
  $n \leftarrow ||O||$

Recursion abort

  if $O - \{O_n\}$ is empty **then**
    $Best \leftarrow$ individual picked at random from population with replacement
  else

Recursion step

    $Best \leftarrow ObjectiveTournament(O - \{O_n\}, T - \{T_n\})$
  for i from 1 to T_n **do**

This is the remaining object, so pick any individual for Next

    if $O - \{O_n\}$ is empty **then**
      $Next \leftarrow$ individual picked at random from population with replacement
    else

Get next individual by recursion as we got Best individual

      $Next \leftarrow ObjectiveTournament(O - \{O_n\}, T - \{T_n\})$
    if $ObjectiveValue(O_n, Next) > ObjectiveValue(O_n, Best)$ **then**
      $Best \leftarrow Next$
return $Best$

Open Problems

- Opposing objectives cancel each other out
  - We get solutions that are neither good in any objective
- We do not know what are *the best options* available
  - Trade-off must be represented by the set of final solutions/individuals
  - I want to be able to chose among them

- Still, I want only the best individuals that are superior to all other solutions in a certain objective
  - Called dominance relation
  - Set of individuals are the Pareto front of the solution space

Vilfredo Pareto

- Italian engineer, sociologist, economist, philosopher, and political scientist (1848-1923)
- First to analyze economic problems with mathematical tools
- Famous for two things:
  - 80/20 rule: For many events, roughly 80% of the effects come from 20% of the causes
    - True for many domains (engineering, economics, sales, politics, etc.)
    - Microsoft reported that by fixing the top 20% of most-reported bugs, 80% of the related errors and crashed would be eliminated
    - 80% of traffic in load testing occurs in 20% of the time
    - 20% of the code has 80% of the errors
    - 80% of use cases are easy to implement and 20% are way harder
  - Pareto front (see next)

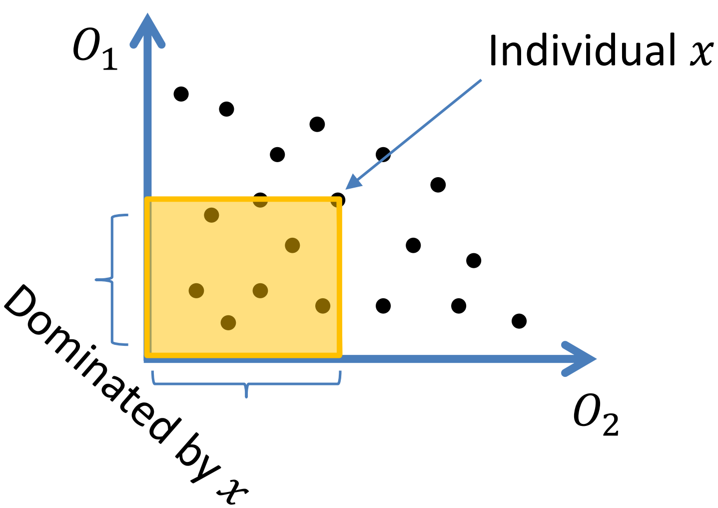
Pareto Front / Optimum

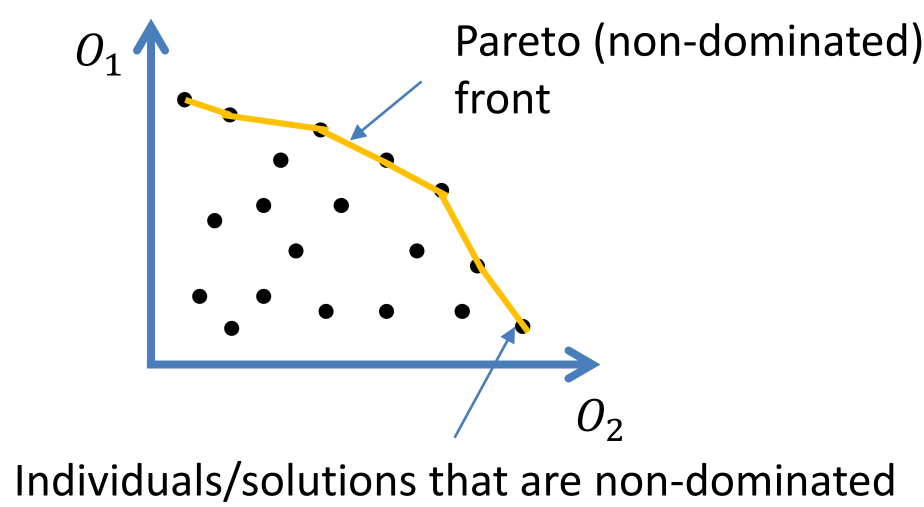
- Pareto Optimum
  - *“The optimum allocation of the resources of a society is not attained so long as it is possible to make at least one individual better off in his own estimation while keeping others as well off as before in their own estimation.”*
  - Reference: Pareto, V., Manuale di Economia Politica, Societa Editrice Libraria, Milano, Italy, 1906.
  - What does this mean?

Pareto Dominance

- Two candidate solutions x and y
- x is **Pareto dominant** to y if x *is at least as good as* y in *all* objectives and *superior* to y in at least one objective
  - Why select y in any case, when x is always as good as y or sometimes even better?

Objective space





Dominance Relation Properties I

- Reflexive
  - Is **not reflexive**, because any solution x does not dominate itself by definition of dominance
- Symmetric
  - Is **not symmetric** because $x \preceq y$ does not imply $y \preceq x$. The opposite is true: if $x \preceq y$, then $y \npreceq x$
- Antisymmetric
  - Since dominance relation is not symmetric and not reflexive, **it cannot be antisymmetric** as well
- Transitive
  - Is **transitive**, because $x \preceq y$ and $y \preceq z$, then $x \preceq z$

Dominance Relation Properties II

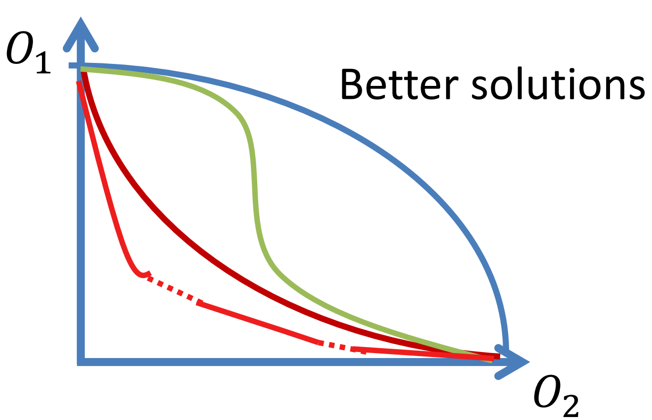
- Consider: x does not dominate y, does it mean that y dominates x?
  - No! Both can be non-dominating!

- Dominance relation qualifies as an ordering relation due to its transitivity property

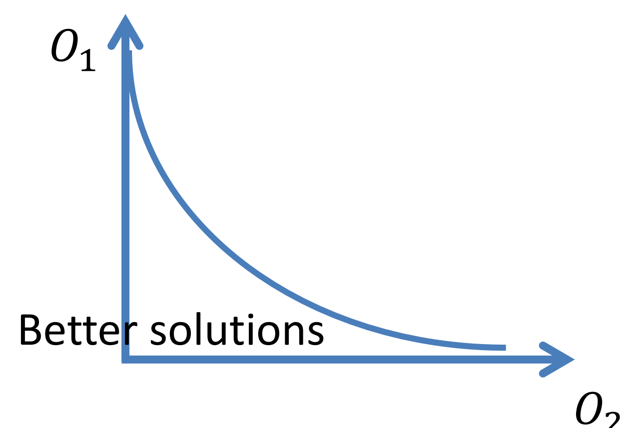
Shapes of Fronts

- Convex: curved outwards towards better solutions
- Concave: curved inwards away from better solutions
- Nonconvex: contains subparts of both kinds
- Discontinuous: regions that are impossible to achieve

Maximization problem



Minimization problem



Open Questions

- Which solutions on the Pareto front to compute?
  - Better have diversity / spread to not have a small group of very similar solutions, but more of the whole front

- What about many objectives (>4) ?
  - Open problem in research (not covered here)
  - Idea: Use hypervolume spanned by the multi-dimensional Pareto front as a metric for diversity
  - E.g. see: HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization, J. Bader and E. Zitzler. In Evolutionary Computation. 2011, Vol. 19, No. 1, Pages: 45-76

Dominance Practice

- Which individuals are non-dominated?

$$\begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \uparrow \\ \downarrow \end{array} \begin{bmatrix} Performance\ in\ s \\ Memory\ in\ MB \\ Energy\ in\ J \\ Reliability\ in\ d \\ Footprint\ in\ KB \end{bmatrix} : \begin{bmatrix} 125 \\ 80 \\ 2150 \\ 238 \\ 1530 \end{bmatrix} \begin{bmatrix} 97 \\ 97 \\ 1850 \\ 138 \\ 2230 \end{bmatrix} \begin{bmatrix} 224 \\ 50 \\ 5150 \\ 538 \\ 2555 \end{bmatrix} \begin{bmatrix} 66 \\ 80 \\ 2005 \\ 268 \\ 1344 \end{bmatrix} \begin{bmatrix} 155 \\ 122 \\ 2553 \\ 156 \\ 1970 \end{bmatrix} \begin{bmatrix} 155 \\ 80 \\ 1450 \\ 256 \\ 1130 \end{bmatrix}$$

- Idea: Pairwise comparison
  - If one individual is always equal and at least one time better, it dominates the other one

Pareto Domination Algorithm

- Idea: Implement tournament selection operator based on Pareto domination

$A \leftarrow$ individual A
$B \leftarrow$ individual B
$O \leftarrow \{O_1, O_2, \dots, O_k\}$ objectives
$a \leftarrow \boldsymbol{false}$
for each objective $O_i \in O$ **do**
  if $ObjectiveValue(O_i, A) > ObjectiveValue(O_i, B)$ **then**
    $a \leftarrow \boldsymbol{true}$
  else if $ObjectiveValue(O_i, A) < ObjectiveValue(O_i, B)$ **then**
    return $\boldsymbol{false}$
return a

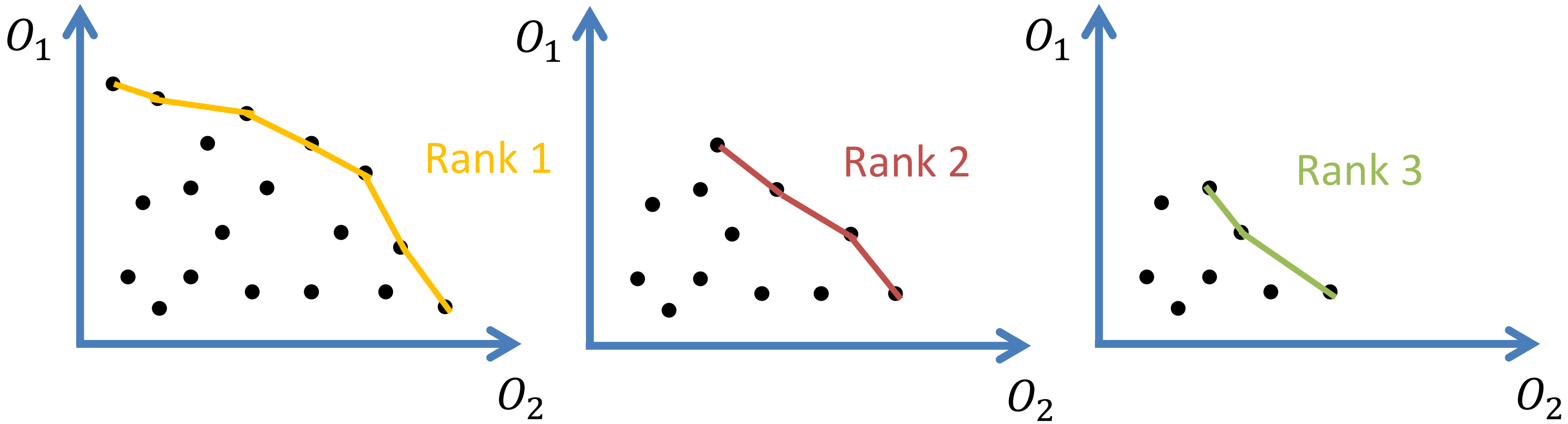
Pareto Domination Binary Tournament Selection

$P \leftarrow$ population
$P_A \leftarrow$ individual picked at random from P with replacement
$P_B \leftarrow$ individual picked at random from P with replacement
if P_A *Pareto Dominates* P_B **then**
  return P_A
else if P_B *Pareto Dominates* P_A **then**
  return P_B
else
  return either P_A or P_B chosen at random

- Improvement: if two individuals do not Pareto dominate each other, we might be interested in the individual that is least dominated by other individuals in the population
- Idea: Use a metric specifying the **closeness** of a solution to the Pareto front

Pareto Front Rank

- Rank defines distance to the Pareto front
  - Individuals in the front have rank 1
- Idea: Remove all rank 1 individuals from the set
  - Now, new individuals form a new Pareto front (rank 2 front)
  - Recursively, remove elements from the population of the current front and build the next front, such that all individuals belong to a certain rank



Computing the Pareto Front

$G \leftarrow \{G_1, G_2, \dots, \mathrm{G_m}\}$ Group of individuals for computing the front
$O \leftarrow \{O_1, O_2, \dots, O_k\}$ objectives
$F \leftarrow \{\}$ Pareto front
for each individual $G_i \in G$ **do**
  $F \leftarrow F \cup \{G_i\}$ Assume G_i is in the Pareto front
  for each individual $F_j \in F$ **do**
    if F_j *Pareto Dominates* G_i given O **then**
      $F \leftarrow F - \{G_i\}$
      break
    else if G_i *Pareto Dominates* F_j given O **then**
      $F \leftarrow F - \{F_j\}$
return F

Check wether G_i can stay in the front or G_i dominates another individual in the front that has to be removed

From Pareto Front To Ranks

- Compute the Pareto front as shown before
- Remove the individuals of the front
- Compute the Pareto front again for the reduced subset
- Repeat until there are no individuals in the population

- Why is this useful?
  – Lower ranked individuals are better (closer to the Pareto front)
  – $Fitness(i) = \frac{1}{1+ParetoFrontRank(i)}$
  – Store each group of individuals separately and store rank in each individual

Non-Dominated Sorting

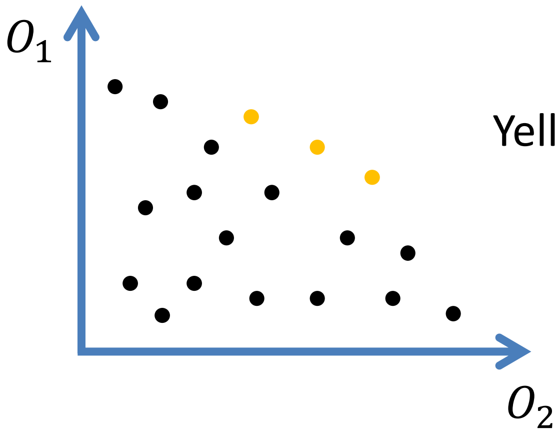
Away from Naïve: Non-Dominated Sorting

- Invented by N. Srinvas and K. Deb in 1994

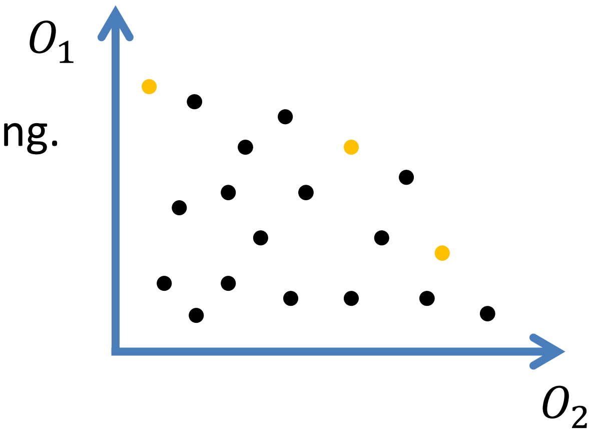
$P \leftarrow$ population
$O \leftarrow \{O_1, O_2, \ldots, O_k\}$ objectives

$P' \leftarrow P$ — Initially, all elements are considered to compute the current front
$R \leftarrow \langle \quad \rangle$ — Empty vector of Pareto front ranks
$i \leftarrow 1$ — Start with the first front
repeat
  $R_i \leftarrow$ Pareto Non–Dominated Front of P' using O — Compute all individuals of the current front that are still in P'
  for each individual $r \in R_i$ **do**
    $ParetoFrontRank(r) \leftarrow i$
    $P' \leftarrow P' - \{r\}$
  — Go through these individuals, store their rank, and remove them from the population
  $i \leftarrow i + 1$
until P' is empty
return R

Spread out the Population: Sparsity

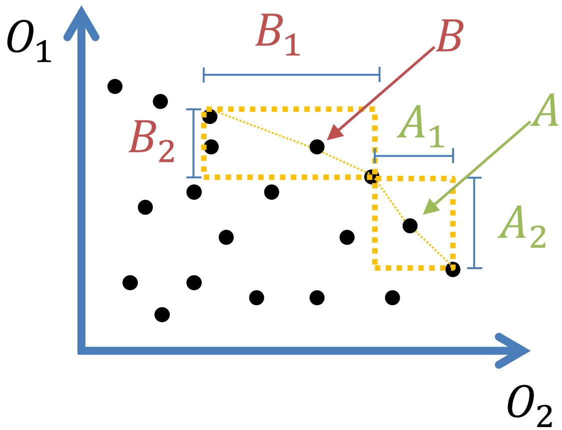


Yellow individuals are used for breeding.
Which selection is better?



- To better show the tradeoff and allow exploration, we want to have individuals with a certain distance to each other
- Idea: Use **sparsity of a region** as a measure of spread

Sparsity using Manhattan Distance



- To get the surrounding region of an individual:
  - Get the direct neighbors of the same Pareto front rank
  - Span the region
- Compute the Manhattan distance over every objective between an individual's left and right neighbors
  - Far end individuals get an infinite sparsity to be always selected
  - Example: $A_1 + A_2 < B_1 + B_2$ so that B is in a sparser region
- Requires the value range of every objective function

Multi-Objective Sparsity Assignment

$F \leftarrow \langle F_1, F_2, \dots, F_m \rangle$ Pareto front rank of individuals
$O \leftarrow \{O_1, O_2, \dots, O_k\}$ objectives
$Range(O_i)$ function providing the range $(max - min)$ of possible values for given objective O_i

for each individual $F_j \in F$ **do**
  $Sparsity(F_j) \leftarrow 0$ First, set sparsity to zero for all individuals
for each objective $O_i \in O$ **do**
  $F' \leftarrow F$ sorted by objective value given objective O_i Sort individuals based on current objective
  $Sparsity(F'_1) \leftarrow \infty$
  $Sparsity(F'_{||F||}) \leftarrow \infty$ Assign infinity to the ends
  for j from 2 to $||F'|| - 1$ **do**

Sparsity based on previous objectives

Region of neighbors

$$Sparsity(F'_j) \leftarrow Sparsity(F'_j) + \frac{ObjectiveValue(O_i, F'_{j+1}) - ObjectiveValue(O_i, F'_{j-1})}{Range(O_i)}$$

Normalization

return F with assigned Sparsities by F'

- How to compute sparsity for the whole population?
  - Break population into ranks and compute for each rank the sparsity

Tournament Selection with Sparsity and Non-Dominated Sorting

- Selected individuals are both close to the Pareto front and spread throughout the front

$P \leftarrow$ population with Pareto front ranks assigned to the individuals
$Best \leftarrow$ picked at random from P
$t \leftarrow$ tournament size, $t \geq 1$

for i from 1 to t **do**
  $Next \leftarrow$ individual picked at random from P with replacement
  if $ParetoFrontRank(Next) < ParetoFrontRank(Best)$ **then**
    $Best \leftarrow Next$
  else if $ParetoFrontRank(Next) == ParetoFrontRank(Best)$ **then**
    if $Sparsity(Next) > Sparsity(Best)$ **then**
      $Best \leftarrow Next$
return $Best$

Non-Dominated Sorting Genetic Algorithm II (NSGA-II)

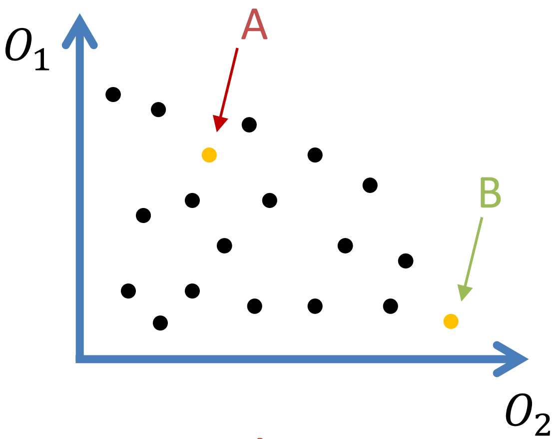
- Developed by K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan in 2000: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II
  - State of the art technique
- Idea: Keep all the best known individuals so far (similar to (μ+λ) or elitist)
  - A is a store of the best n individuals discovered so far
  - Breed a new population P from A and let the individuals of both sets compete to stay in A
  - Highly exploitative?
    - Not really, since we use Sparsity to spread out in the optimization space

NSGA-II Algorithm

$m \leftarrow$ desired population size
$a \leftarrow$ desired archive size, usually $a = m$
$P \leftarrow \{P_1, P_2, \ldots, P_m\}$ population with Pareto front ranks assigned to the individuals
$A \leftarrow \{\ \}$ archive
repeat
  $AssessFitness(P)$ ← Calculate objective values to obtain Pareto front ranks
  $P \leftarrow P \cup A$
  $BestFront \leftarrow$ Pareto front of P
  $A \leftarrow \{\ \}$
  $R \leftarrow$ Compute front ranks of P
  for each front rank $R_i \in R$ **do** ← Go through the front ranks and fill the archive
    Compute sparsities of individuals in R_i
    if $||A|| + ||R_i|| \geq a$ **then** ← Last front rank, whose members can come into A
      $A \leftarrow A \cup$ the sparsest $a - ||A||$ individuals in R_i, breaking ties arbitarily
      break ← Insert only as many as the archive can fill and break the for-loop
    else
      $A \leftarrow A \cup R_i$
  $P \leftarrow Breed(A)$, using tournament selection with sparsity and non−dominated sorting
until $BestFront$ is optimal or out of time
return $BestFront$

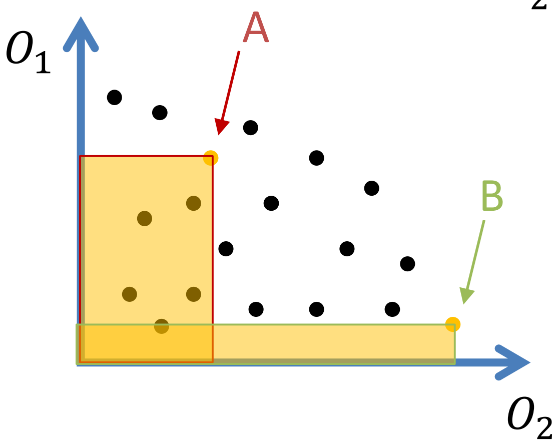
Pareto Strength

- Alternative ~~fact~~ measure to compute the fitness and do parent selection



If you would have to chose between A and B, so far, we would select B, because it has a front rank 1 vs. A with a front rank 2. But is this a good choice?

What if we look at the number of individuals that an individual dominates?



A dominates 5 individuals
B dominates only 1 individual
-> so why don't pick A?

It does not necessarily correspond to closeness to the Pareto front and corner individuals are weak.

Wimpiness

- Use weakness instead:
  - Number of individuals that dominate the current individual
  - Pareto front individuals have weakness of 0
  - Individuals far away from the front have a high weakness
- Improve weakness using the strength of the individuals that dominate the current individual: **Wimpiness**

- $Wimpiness(i) = \sum_{g \in G \text{ that Pareto Dominate } i} Strength(g)$
- $Fitness(i) = \frac{1}{1 + Wimpiness(i)}$

Non-dominated individuals have a fitness of 1

Strength Pareto Evolutionary Algorithm2 (SPEA2)

- Developed by E. Zitzler, M. Laumanns, and L. Thiele in 2002: SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization
  - Similar to NSGA-II, SPEA2 maintains a store of the best known Pareto front individuals + other members found so far
  - Uses Pareto measure (using Wimpiness) and crowding measure (using distance to other individuals in the multi-objective space and no ranks) for its fitness assessment
- Similarity measure computes a distance to other individuals in the population (i.e., to the kth closest individual)
  - Simple solution: Compute distance from everyone to everyone and for each individual sort them to its own distance and take the kth closest individual ($O(n^2 lgn)$), for n individuals

Distance Computation

$P \leftarrow \langle P_1, P_2, \dots, P_m \rangle$ population
$O \leftarrow \{O_1, O_2, \dots, O_n\}$ objectives
$P_l \leftarrow$ individual to compute kth klosest individuals
$k \leftarrow$ desired individual index (the kth individual from l)
global $\boldsymbol{D \leftarrow m\ vectors, each\ of\ size\ m}$ ← D_i holds a vector of distances of various individuals i
global $\boldsymbol{S \leftarrow \{S_1, \dots, S_m\}}$ ← S_i will be true if D_i has already been sorted
perform only once
  for each individual $P_i \in P$ **do**
    $V \leftarrow \{\ \ \}$
    for each individual $P_j \in P$ **do**
      $V \leftarrow V \cup \{\sqrt{\underbrace{\sum_{m=1}^{n} \left(ObjectiveValue(O_m, P_i) - ObjectiveValue(O_m, P_j) \right)}_{\text{Sum the distances over all objectives}}^2}\}$
    $D_i \leftarrow V$
    $S_i \leftarrow \boldsymbol{false}$

Computes the distances among all individuals

perform each time
  if S_l *is* $\boldsymbol{false}$ **then**
    Sort D_l, smallest first
    $S_l \leftarrow \boldsymbol{true}$
  $W \leftarrow D_l$
return W_{k+1} ← Since W_0 is the distance to ourself (i.e., 0), we return the distance of the $k+1$ closest neighbor

SPEA2: Putting Everything Together

- $G_i \leftarrow Wimpiness(i) + \frac{1}{2+d_i}$ where d_i is the distance of i to the $k$the closest neighbar
  - Typically, set $k = \left\lfloor \sqrt{||P||} \right\rfloor$
  - A smaller G_i, is better, because a large distance makes G_i smaller and so we get more diversity and spread and smaller Wimpiness is better, too
- Each iteration of SPEA2 builds an archive/store of size n with the current Pareto front of the population
  - If not enough individuals for n, fill with other fit individuals
  - If too many individuals for n, remove the ones with the smallest distance kth closest distance (starting with $k = 1$, continuing with $k = 2$, etc.)

SPEA2: Archive Construction Algorithm

$P \leftarrow \langle P_1, P_2, \dots, P_m \rangle$ population
$O \leftarrow \{O_1, O_2, \dots, O_n\}$ objectives
$a \leftarrow$ desired archive size
$A \leftarrow$ Pareto non–dominated front of P ← Initialize the archive with Pareto front individuals
$Q \leftarrow P - A$ ← Get the remaining individuals
if $||A|| < a$ **then** ← If not enough individuals in the archive, add fittest individuals to the archive
  Sort Q by fitnes
  $A \leftarrow A \cup$ the $a - ||A||$ fittest individuals in Q, breaking ties arbitarily
while $||A|| > a$ **do** ← If too many individuals in the archive, remove the *k-closest* ones
  $Closest \leftarrow A_1$
  $c \leftarrow$ index of A_1 in P
  for each individual $A_i \in A$ except A_1 **do**
    $l \leftarrow$ index of A_i in P
    for k from 1 to $m - 1$ **do**
      if $DistanceOfKthNearest(k, P_l) < DistanceOfKthNearest(k, P_c)$ **then**
        $Closest \leftarrow A_i$
        $c \leftarrow l$; **break**
      else if $DistanceOfKthNearest(k, P_l) > DistanceOfKthNearest(k, P_c)$ **then**
        break
  $A \leftarrow A - \{Closest\}$
return A

SPEA2: Algorithm

$m \leftarrow$ desired population size
$a \leftarrow$ desired archive size, usually $a = m$
$P \leftarrow \{P_1, P_2, \dots, P_m\}$ population with Pareto front ranks assigned to the individuals
$A \leftarrow \{\quad\}$ archive
repeat
  $AssessFitness(P)$
  $P \leftarrow P \cup A$
  $BestFront \leftarrow$ Pareto front of P
  $A \leftarrow$ Construct SPEA2 archive of size a from P
  $P \leftarrow Breed(A)$, using tournament selection of size 2
until $BestFront$ is optimal or out of time
return $BestFront$

NSGA_II and SPEA2 are both version of (μ+λ) in multi-objective space, combined with a diversity mechanism and a technique for selecting individuals that are closer to the Pareto front

Take Home Message:

- Multiple objectives are more common in practice
- Not a single solution is the optimum, but a set of solutions
- Pareto front represents the individuals that are non-dominated by others
  - Dominance relation: an element dominates another if it is better in at least one objective and not worse in all others
- Ranks of dominance/Pareto fronts can be recursively computed for all elements in the population and represent as a way to assess the fitness
- Another factor for fitness is the diversity of the individuals in the front, so we add a measure based on sparsity to favor individuals that are farther away from other

Take Home Message:

- Combining sparsity and Pareto front ranks plus a storage of the best found individuals so far is realized by NSGA-II
- An alternative approach is SPEA2
  - With a more complex diversity measure, using the distance to the kth nearest individual
  - With a different Pareto front distance measure, using the weakness of individuals (i.e., number of individuals that dominate it)

Next Lecture

- Combinatorial Optimization
  - Greedy randomized adaptive search
  - Ant colony optimization
  - Guided local search