DM872

Mathematical Optimization at Work

Multiobjective Optimization

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Multi-Objective Optimization: Basics

Multiobjective optimization problem with p objective functions to be minimized over a discrete, nonempty, set X of feasible solutions:

$$\min \{f_1(\boldsymbol{x}), \ldots, f_p(\boldsymbol{x}) \mid \boldsymbol{x} \in X\}$$

- $\boldsymbol{x} \in X$ a feasible solution, $\boldsymbol{y} = [f_1(\boldsymbol{x}), \ldots, f_p(\boldsymbol{x})]$ a feasible point (image)
- $Y = f(X) \subseteq \mathbb{R}^p$ set of images of all feasible solutions in objective space.
  Assume Y is bounded. We can rewrite:

$$\min \{(y_1 \ldots, y_p) \mid \boldsymbol{y} \in Y\}$$

- Given a point $\boldsymbol{y} \in \mathbb{R}^p$, $\boldsymbol{y}_{-k}$ is its orthogonal projection on the subspace $\mathbb{R}^{p-1}$ i.e.

$$\boldsymbol{y}_{-k} = (y_1, \ldots, y_{k-1}, y_{k+1}, \ldots, y_p)$$

Multi-Objective Optimization: Basics

- Objective vector $\boldsymbol{f} = (f_1, \ldots, f_p)$
- We want to minimize $\boldsymbol{f}$ but what does it mean?
  - Scalarization (eg, Weighted sum)
  - Lexicographic
  - Pareto optimality (without previous knwoledge on component importance)

Basic Notions

Partial orders on $\mathbb{R}^p$: let $\boldsymbol{u}$ and $\boldsymbol{v}$ be vectors in $\mathbb{R}^p$:

weak component-wise order	$\boldsymbol{u} \leqq \boldsymbol{v}$	$u_i \leq v_i,\ i = 1, \ldots, p$;
component-wise order	$\boldsymbol{u} \preceq \boldsymbol{v}$	$u_i \leq v_i,\ i = 1, \ldots, p$ and $\boldsymbol{u} \neq \boldsymbol{v}$;
strict component-wise order	$\boldsymbol{u} \prec \boldsymbol{v}$	$u_i < v_i,\ i = 1, \ldots, p$

Let $\boldsymbol{x}$ and $\boldsymbol{x}'$ be two feasible solutions and $\boldsymbol{y} = f(\boldsymbol{x})$ and $\boldsymbol{y}' = f(\boldsymbol{x}')$:

$\boldsymbol{y}$ weakly dominates $\boldsymbol{y}'$	iff $\boldsymbol{y} \leqq \boldsymbol{y}'$
$\boldsymbol{y}$ (Pareto) dominates $\boldsymbol{y}'$	iff $\boldsymbol{y} \preceq \boldsymbol{y}'$
$\boldsymbol{y}$ strictly dominates $\boldsymbol{y}'$	iff $\boldsymbol{y} \prec \boldsymbol{y}'$

- $f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$ are non-weakly dominated if $f(\boldsymbol{x}) \not\leqq f(\boldsymbol{x}')$ and $f(\boldsymbol{x}') \not\leqq f(\boldsymbol{x})$
- $f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$ are non-dominated if $f(\boldsymbol{x}) \not\preceq f(\boldsymbol{x}')$ and $f(\boldsymbol{x}') \not\preceq f(\boldsymbol{x})$

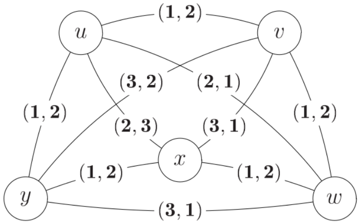
- A solution $\boldsymbol{x}$ is called efficient (or Pareto global optimum solution) iff its image is not dominated: there is no $\boldsymbol{x}' \in X$ such that $\boldsymbol{f}(\boldsymbol{x}') \preceq \boldsymbol{f}(\boldsymbol{x})$
- A solution $\boldsymbol{x}$ is called weakly efficient iff its image is not strictly dominated.

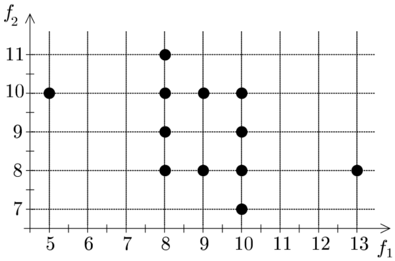
Basic Notions

- A set of solutions S is a Pareto global optimum set iff it contains only and all Pareto global optimum solutions.
- Efficient set (or Pareto frontier) is the image of the Pareto global optimum set in the objective space.
- $X^* \subseteq X$ is strict Pareto global optimum set iff:
  - it contains only Pareto global optimum solutions
  - the corresponding set of objective function value vectors coincides with the efficient set and its elements are unique.

Let $Y_{ND} \subseteq Y$ be the set of points that are non-dominated (ie, the efficient set). We want an algorithm to generate Y_{ND} and provide one of the corresponding efficient solutions for each point of this set.

Example – Multi-objective TSP





Pareto global optimum set:

π	$f(\pi)$
$\pi = [u, v, w, x, y]$	$[5, 10]$
$\pi = [u, w, v, x, y]$	$[8, 8]$
$\pi = [u, v, w, x, y]$	$[10, 7]$

strict Pareto global optimum set

if all edges had weights, eg, $(3, 3)$, then all $\binom{5}{2}$ solutions would have cost $[5 \cdot 3, 5 \cdot 3]$ and would be in the Pareto global optimum set. However, both the efficient set and the strict Pareto global optimum set would have one single solution, which is any of the feasible ones.

Basic Notions

Computational class #P: concerned with counting the number of solutions.

A counting problem belongs to #P if there is a polynomial nondeterministic algorithm such that for any instance of the problem, it computes a number of yes-answers that is equal to the number of distinct solutions of that instance.

Class #P-complete: a problem $\mathcal{P}_1$ is #P-complete if it belongs to #P and for all problems $\mathcal{P}_2$ in #P there exists a polynomial transformation from $\mathcal{P}_1$ to $\mathcal{P}_2$ such that any instance of $\mathcal{P}_1$ is mapped into an instance of $\mathcal{P}_2$ with the same number of yes-answers as the instance of $\mathcal{P}_1$.

Pareto Optimal Solutions

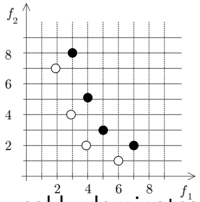
How do we find the set of Pareto Optimal solutions?

- evolutionary algorithms
- scalarization method
- enumeration (branch and bound and dynamic programming) [Przybylski A, Gandibleux X (2017)]
- ϵ-constraint method

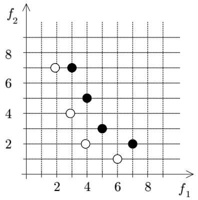
Basic Notions

Given two arbitrary sets of objective function value vectors in a Q-dimensional objective space, $A = \{\boldsymbol{a}^1, \dots \boldsymbol{a}^m\}$ and $B = \{\boldsymbol{b}^1, \dots \boldsymbol{b}^n\}$

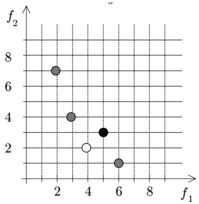
strictly dominates



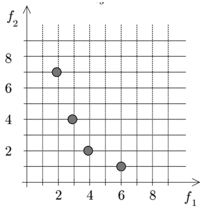
dominates



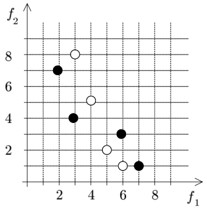
better than



weakly dominates



incomparable



Evolutionary Algorithms Approaches

- strength Pareto EA (SPEA) [Zitzler and Thiele, 1998]
  maintain an external population at every generation storing all non-dominated solutions discovered so far beginning from the initial population. This external population participates in genetic operations. At each generation, a combined population with the external and the current population is first constructed. All non-dominated solutions in the combined population are assigned a fitness based on the number of solutions they dominate and dominated solutions are assigned fitness worse than the worst fitness of any non-dominated solution. A deterministic clustering technique is used to ensure diversity among non-dominated solutions.

- Pareto-archived evolution strategy (PAES) [Knowles and Gome, 1999]
  one parent and one child, the child is compared with respect to the parent. If the child dominates the parent, the child is accepted as the next parent and the iteration continues. If the parent dominates the child, the child is discarded and a new mutated solution (a new child) is found. If the child and the parent do not dominate each other, the choice between the child and the parent is made by comparing them with an archive of best solutions found so far. Both domination and diversity are considered.

Evolutionary Algorithms Approaches

- elitist GA [Rudolph, 1999]
  systematic comparison of individuals from parent and offspring populations.
  The non-dominated solutions of the offspring population are compared with parent solutions to form an overall non-dominated set of solutions, which becomes the parent population of the next iteration.
  If the size of this set is not greater than the desired population size, other individuals from the offspring population are included.

- Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II [Deb, Agrawal, Pratap, Meyarivan, 2000]

NSGA-II: Main Components

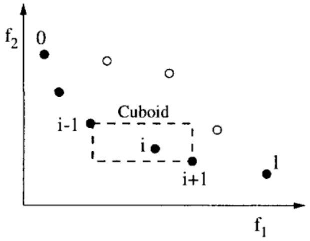
Systematic comparison between solutions to find the non-dominated fronts.
It requires $O(MN^3)$ if only one solution in each front. (M number of objectives, N population size)

Fast non-dominated sorting approach in $O(MN^2)$

```
fast-nondominated-sort($P$)
$P' = \{1\}$                                      include first member in $P'$
for each $p \in P \wedge p \notin P'$                     take one solution at a time
    $P' = P' \cup \{p\}$                              include $p$ in $P'$ temporarily
    for each $q \in P' \wedge q \neq p$                  compare $p$ with other members of $P'$
        if $p \prec q$, then $P' = P' \backslash \{q\}$        if $p$ dominates a member of $P'$, delete it
        else if $q \prec p$, then $P' = P' \backslash \{p\}$   if $p$ is dominated by other members of $P'$,
                                                  do not include $p$ in $P'$
```

NSGA-II: Main Components

Density estimation (crowding distance) of a particular point in the population: average distance of the two points on either side of this point along each of the objectives



```
crowding-distance-assignment($\mathcal{I}$)
$l = |\mathcal{I}|$                                          number of solutions in $\mathcal{I}$
for each $i$, set $\mathcal{I}[i]_{distance} = 0$            initialize distance
for each objective $m$
   $\mathcal{I} = \text{sort}(\mathcal{I}, m)$               sort using each objective value
   $\mathcal{I}[1]_{distance} = \mathcal{I}[l]_{distance} = \infty$   so that boundary points are always selected
   for $i = 2$ to $(l-1)$                                    for all other points
      $\mathcal{I}[i]_{distance} = \mathcal{I}[i]_{distance} + (\mathcal{I}[i+1].m - \mathcal{I}[i-1].m)$
```

NSGA-II: Main Components

The crowded comparison operator $\prec_n$ guides the selection process towards a uniformly spread-out Pareto-optimal front.

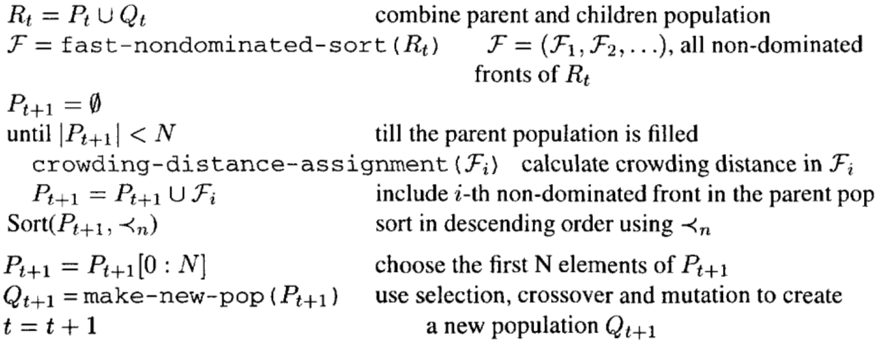
Let us assume that every individual i in the population has two attributes.

1. Non-domination rank (s_{rank})
2. Local crowding distance (s_{distance})

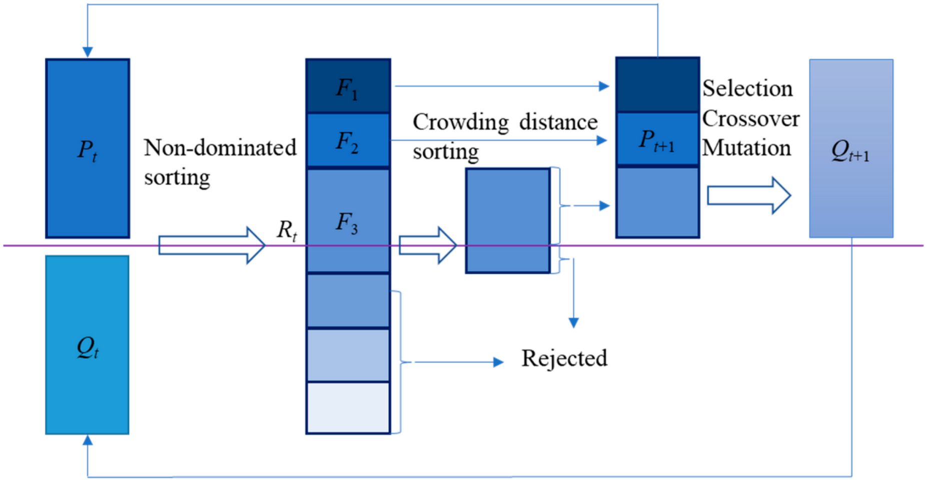
Partial order $\prec_n$:
$s \prec_n s'$ if $(s_{\text{rank}} < s'_{\text{rank}})$ or $((s_{\text{rank}} = s'_{\text{rank}}) \wedge (s_{\text{distance}} > s'_{\text{distance}}))$

NSGA-II: Algorithm

- Initially, a random parent population P_0 is created. The population is sorted based on the non-domination. Each solution is assigned a fitness equal to its non-domination level (1 is the best level). Thus, minimization of fitness is assumed.
- Binary tournament selection, recombination, and mutation operators are used to create a child population Q_0 of size N.
- At each iteration $t > 1$ and for a particular generation an elitism procedure is used:



```
$R_t = P_t \cup Q_t$                                   combine parent and children population
$\mathcal{F} =$ fast-nondominated-sort($R_t$)          $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \ldots)$, all non-dominated
                                                            fronts of $R_t$
$P_{t+1} = \emptyset$
until $|P_{t+1}| < N$                                  till the parent population is filled
    crowding-distance-assignment($\mathcal{F}_i$)      calculate crowding distance in $\mathcal{F}_i$
    $P_{t+1} = P_{t+1} \cup \mathcal{F}_i$             include $i$-th non-dominated front in the parent pop
Sort($P_{t+1}, \prec_n$)                               sort in descending order using $\prec_n$
$P_{t+1} = P_{t+1}[0 : N]$                             choose the first N elements of $P_{t+1}$
$Q_{t+1} =$ make-new-pop($P_{t+1}$)                    use selection, crossover and mutation to create
$t = t + 1$                                                 a new population $Q_{t+1}$
```



Machines 2021, 9, p. 156

Iteration Complexity

At each generation, the basic operations being performed and the worst case complexities associated with it are as follows:

1. Non-dominated sort is $O(MN^2)$,
2. Crowding distance assignment is $O(MN \log N)$
3. Sort on $\prec_n$ is $O(2N \log(2N))$

The overall complexity of the single iteration is $O(MN^2)$.