On Bounding Fronts and Bounding Vectors in Multi-objective Optimization

Julius Žilinskas

Vilnius University, Lithuania

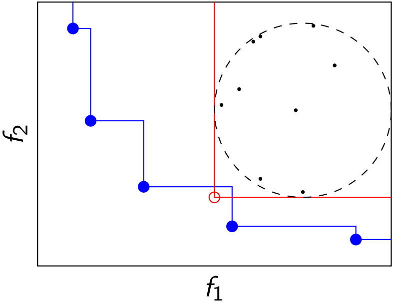
Clustering and Search Techniques in Large Scale Networks
November 5, 2014

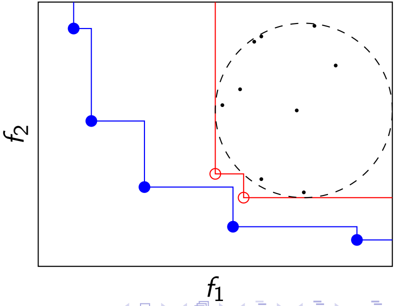
Multi-objective branch and bound algorithm

- In multi-objective optimization, the subset cannot contain Pareto optimal solutions if each bounding vector $\mathbf{b} \in B$ in bounding front B is dominated by at least one already known decision vector $\mathbf{a}$ in the current solution set S:

$$\forall \mathbf{b} \in B \; \exists \mathbf{a} \in S : \begin{array}{l} \forall i \in \{1, 2, \ldots, d\} : \; f_i(\mathbf{a}) \leq b_i \; \& \\ \exists j \in \{1, 2, \ldots, d\} : \; f_j(\mathbf{a}) < b_j. \end{array}$$

- The simplest bounding front consists of a single ideal vector composed of lower bounds for each objective function.

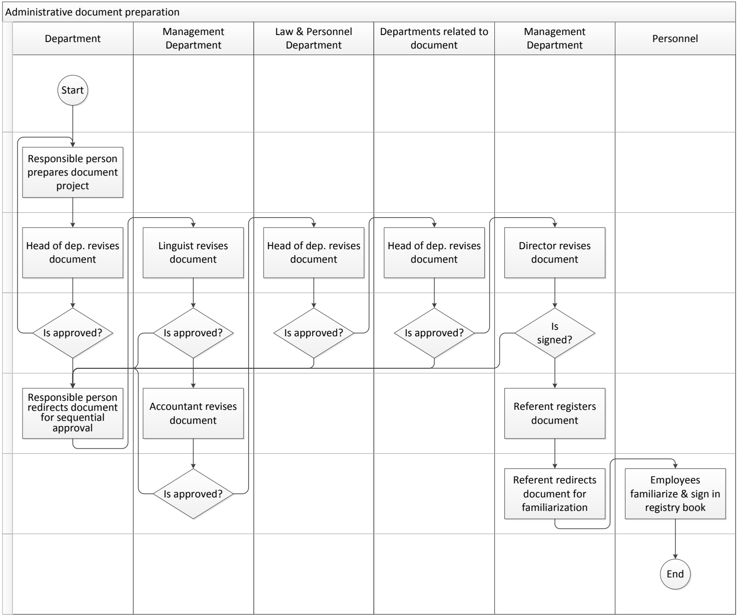




Aesthetic visualization of business process diagrams

- A business process diagram consists of elements (e.g., activities, events, and gateways) which should be drawn according to the rules of Business Process Modeling Notation.
- The elements are drawn as shapes which are allocated in a pool divided by the swimlanes according to function or role.
- Several algorithms for the aesthetic drawing of connectors were proposed assuming the location of shapes is fixed. Such a situation occurs in case a business process diagram is drawn in an interactive mode, and a user selects sites for shapes.
- After an interactive session is completed it is reasonable to draw the final aesthetically appealing diagram. In such a situation the complete drawing problem (allocation of shapes and drawing of connectors) could be considered.
- We propose to decompose the problem into two stages: allocation of shapes and drawing of connectors.
- In the present talk the problem of aesthetic allocation of shapes is attacked by multi-objective optimization.

An example of a business process diagram



Aesthetic allocation of shapes in business process diagrams

- In this talk we are interested only in the allocation of shapes, i.e. we ignore the interpretation of the diagram in terms of the visualized business process.
- It is requested to allocate shapes in swimlanes, and the swimlines with regard to each other aiming at aesthetical appeal of the drawing.
- The connectors show the sequence flow, two flow objects are connected if one directly precedes another.
- The shapes are allocated in such a way that the connected shapes were close to each other and that the flow would direct from left to right and from top to bottom.
- In this talk the bi-objective problem is considered taking into account two simultaneously optimized objectives:
  - Minimization of total length of connectors: The sum of city block distances between connected shapes is minimized.
  - Minimization of the number of right down flow violations: The number of times the preceding shape in the connection is not higher than and is to the right from the following shape is minimized.

Shape allocation: notations

- The shapes are allocated in a grid of predefined number of rows and columns (swimlanes). Let us denote the number of rows by n_r and the number of columns by n_c.
- The data of the problem are assignment of shapes to the roles (or functions) and the list of connections.
  - Let us denote the number of shapes by n and the roles corresponding to shapes by $\mathbf{d}$, where $d_i,\ i = 1, \ldots, n$ define the role number of each shape.
  - The connections are defined by $n_k \times 2$ matrix $\mathbf{K}$ whose rows define connecting shapes and k_{i1} precedes k_{i2}.
- The shapes assigned to the same role should be shown in the same column (swimlane), however the columns may be permuted. Let us denote the assignment of roles to columns by $\mathbf{y}$ which is a permutation of $(1, \ldots, n_c)$ and y_i defines the column number of ith role.
- Another part of decision variables define assignment of shapes to rows. Let us denote this assignment by $\mathbf{x}$, where x_i defines the row number of ith shape.

Shape allocation: objectives

- We model the potential length of connector as a city block distance between shapes. Therefore the total length of connectors is calculated as

$$f_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n_k} |x_{k_{i1}} - x_{k_{i2}}| + |y_{d_{k_{i1}}} - y_{d_{k_{i2}}}|.$$

- The number of right down flow violations is calculated as

$$f_2(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n_k} v_d(k_{i1}, k_{i2}) + v_r(k_{i1}, k_{i2}),$$

where down flow (v_d) and right flow (v_r) violations are

$$v_d(i,j) = \begin{cases} 1, & x_i \geq x_j, \\ 0, & \text{otherwise,} \end{cases} \quad v_r(i,j) = \begin{cases} 1, & y_{d_i} > y_{d_j}, \\ 0, & \text{otherwise.} \end{cases}$$

The connection of two shapes in the same row violates down flow since the bottom or side of preceding shape connects to the top of the following shape.

Separation of problem

- In such a definition objective functions are separable into two parts, one is dependent only on decision variables $\mathbf{x}$ and another on $\mathbf{y}$:

$$f_1(\mathbf{x},\mathbf{y}) = f_{1x}(\mathbf{x}) + f_{1y}(\mathbf{y}),$$

$$f_{1x}(\mathbf{x}) = \sum_{i=1}^{n_k} |x_{k_{i1}} - x_{k_{i2}}|,\ f_{1y}(\mathbf{y}) = \sum_{i=1}^{n_k} |y_{d_{k_{i1}}} - y_{d_{k_{i2}}}|,$$

$$f_2(\mathbf{x},\mathbf{y}) = f_{2x}(\mathbf{x}) + f_{2y}(\mathbf{y}),$$

$$f_{2x}(\mathbf{x}) = \sum_{i=1}^{n_k} v_d(k_{i1}, k_{i2}),\ f_{2y}(\mathbf{y}) = \sum_{i=1}^{n_k} v_r(k_{i1}, k_{i2}).$$

- Therefore the problem can be decomposed into two: find non-dominated vectors (f_{1x}, f_{2x}) representing assignments of shapes to rows and non-dominated vectors (f_{1y}, f_{2y}) representing assignments of roles to columns.
- The non-dominated solutions of two problems are then aggregated and non-dominated solutions of the whole problem are retained.

Size of the problem

- The number of solutions of the first problem is

$$\prod_{i=1}^{n_c} \frac{n_r!}{(n_r - n_i)!},$$

  where n_i is the number of shapes assigned to ith role.
- The number of solutions of the second problem is $n_c!$.
- For example, if we have 3 roles, there are 4 objects in one role and 6 objects in each other two roles, and we want to fit the diagram in 7 rows, the number of solutions of the second problem is $3! = 6$ and the number of solutions of the first problem is

$$\frac{7!}{3!} \times 7! \times 7! = 21\,337\,344\,000$$

  which is a big number.
- Decomposition of the problem into two reduces the number of solutions from the product of two numbers to the sum of these.

Branch and bound for shape allocation

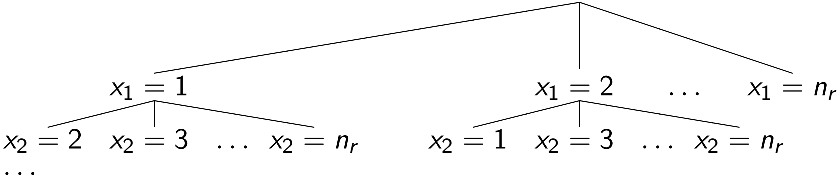
- We will represent a set of solutions of multi-objective problem for allocation of the shapes in business process diagrams as a partial solution where only some shapes are assigned to rows.
- Therefore, the partial solution is represented by the assignment $\mathbf{x}'$ of $n' < n$ shapes to rows.
- The bounds for objective functions include direct contribution from the partial solution and most favorable contribution from completing the partial solution.
- Let us denote bounding vector for objective functions as

$$\mathbf{b}(\mathbf{x}, n') = \left(\sum_{i=1}^{n_k} c_1(i, \mathbf{x}, n'), \sum_{i=1}^{n_k} c_2(i, \mathbf{x}, n') \right),$$

  where $c_1(i, \mathbf{x}, n')$ and $c_2(i, \mathbf{x}, n')$ denote contribution of ith connector to the bounds.
- When connecting shapes are assigned in the partial solution, direct contribution of the connector can be computed.
- In the contrary case, favorable contribution may be estimated.

Search tree of the branch and bound for shape allocation

- The levels of the tree represent different shapes (flow objects).
- The branches of the tree represent assignment of the flow objects to rows of business process diagram.
- Of course the shape cannot be assigned to the row where another shape of the same role (swimlane) is already assigned.
- We build a branch and bound algorithm for multi-objective problem for allocation of the shapes in business process diagrams using the depth first selection to save memory required for storing of candidate sets.



Bound $\mathbf{b}^1(\mathbf{x}, n')$

- Bounds can be computed involving only direct contribution:

$$
\begin{aligned}
c_1^1(i, \mathbf{x}, n') &= \begin{cases} |x_{k_{i1}} - x_{k_{i2}}|, & \text{if } k_{i1} \le n',\ k_{i2} \le n', \\ 0, & \text{otherwise,} \end{cases} \\
c_2^1(i, \mathbf{x}, n') &= \begin{cases} 1, & \text{if } k_{i1} \le n',\ k_{i2} \le n',\ x_{k_{i1}} \ge x_{k_{i2}}, \\ 0, & \text{otherwise.} \end{cases}
\end{aligned}
$$

A single ideal vector may be used as a bounding vector

$$
\mathbf{b}^1(\mathbf{x}, n') = \left(b_1^1(\mathbf{x}, n'),\ b_2^1(\mathbf{x}, n')\right)
$$

composed of two lower bounds for each objective function:

$$
\begin{aligned}
b_1^1(\mathbf{x}, n') &= \sum_{i=1}^{n_k} c_1^1(i, \mathbf{x}, n'), \\
b_2^1(\mathbf{x}, n') &= \sum_{i=1}^{n_k} c_2^1(i, \mathbf{x}, n').
\end{aligned}
$$

Bound $\mathbf{b}^2(\mathbf{x}, n')$

- If connected shapes belong to the same role the vertical distance between them cannot be zero because two shapes cannot be assigned to the same row in the same column.
- Therefore, connectors contribute at least one to the vertical distance if they connect two shapes of the same role:

$$c_1^2(i, \mathbf{x}, n') = \begin{cases} |x_{k_{i1}} - x_{k_{i2}}|, & \text{if } k_{i1} \le n',\ k_{i2} \le n', \\ 1, & \text{if } k_{i1} > n' \text{ or } k_{i2} > n',\ d_{k_{i1}} = d_{k_{i2}}, \\ 0, & \text{otherwise.} \end{cases}$$

- In such a case the bounding vector may be

$$\mathbf{b}^2(\mathbf{x}, n') = \left(b_1^2(\mathbf{x}, n'),\ b_2^1(\mathbf{x}, n')\right),$$

where

$$b_1^2(\mathbf{x}, n') = \sum_{i=1}^{n_k} c_1^2(i, \mathbf{x}, n').$$

Bound $\mathbf{b}^3(\mathbf{x}, n')$

- A favorable contribution of the connector may be estimated by looking at available places for not yet assigned shape:

$$c_1^3(i,\mathbf{x},n') = \begin{cases} |x_{k_{i1}} - x_{k_{i2}}|, & \text{if } k_{i1} \le n',\ k_{i2} \le n', \\ \min_{x \ne x_j,\ d_j = d_{k_{i2}}} |x_{k_{i1}} - x|, & \text{if } k_{i1} \le n',\ k_{i2} > n', \\ \min_{x \ne x_j,\ d_j = d_{k_{i1}}} |x - x_{k_{i2}}|, & \text{if } k_{i1} > n',\ k_{i2} \le n', \\ 1, & \text{if } k_{i1} > n' \text{ and } k_{i2} > n', d_{k_{i1}} = d_{k_{i2}} \\ 0, & \text{otherwise}, \end{cases}$$

$$c_2^3(i,\mathbf{x},n') = \begin{cases} 1, & \text{if } k_{i1} \le n',\ k_{i2} \le n',\ x_{k_{i1}} \ge x_{k_{i2}}, \\ 1, & \text{if } k_{i1} \le n',\ k_{i2} > n',\ \nexists x > x_{k_{i1}} :\ x \ne x_j,\ d_j = d_{k_{i2}}, \\ 1, & \text{if } k_{i1} > n',\ k_{i2} \le n',\ \nexists x < x_{k_{i2}} :\ x \ne x_j,\ d_j = d_{k_{i1}}, \\ 0, & \text{otherwise.} \end{cases}$$

- The bounding vector involving such contributions

$$\begin{aligned} \mathbf{b}^3(\mathbf{x}, n') &= \left(b_1^3(\mathbf{x}, n'),\ b_2^3(\mathbf{x}, n')\right), \\ b_1^3(\mathbf{x}, n') &= \sum_{i=1}^{n_k} c_1^3(i,\mathbf{x},n'), \\ b_2^3(\mathbf{x}, n') &= \sum_{i=1}^{n_k} c_2^3(i,\mathbf{x},n'). \end{aligned}$$

Bounding front $\mathbb{B}^4(\mathbf{x}, n')$

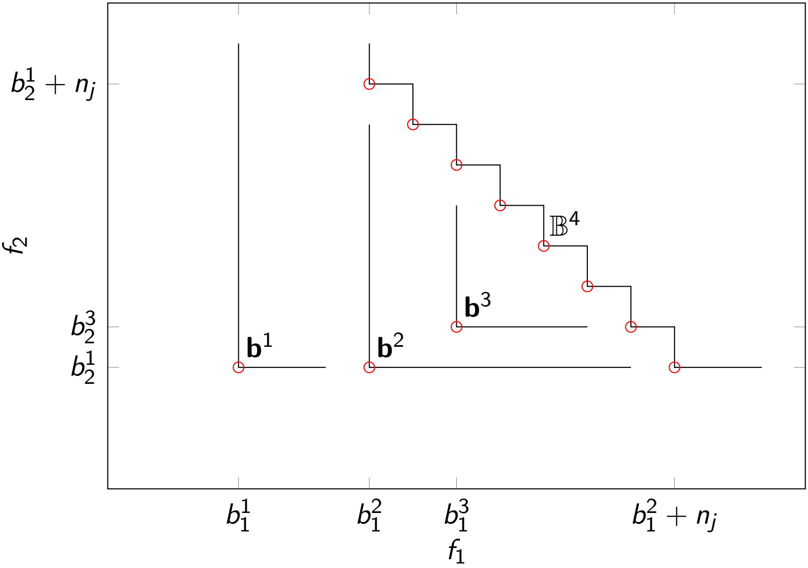
- The most favorable contribution of the connector to the vertical distance is zero – when the connected shapes belong to different roles and assigned to the same row. However in such a situation there is down flow violation because bottom or side of one shape connects to the top of the other shape.
- On the contrary, the most favorable contribution of the connector to down flow violation is zero when preceding shape is higher than the following shape ($x_{k_{i1}} < x_{k_{i2}}$). However in such a situation the vertical distance between the shapes would be at least one.
- Taking this into account a bounding front may be built:

$$\mathbb{B}^4(\mathbf{x}, n') = \left\{ \left(b_1^2(\mathbf{x}, n') + j,\ b_2^1(\mathbf{x}, n') + n_j - j \right) :\ j = 0, \ldots, n_j \right\},$$

where n_j is the number of connectors where at least one of the shapes is not assigned in the partial solution and the shapes belong to different roles:

$$n_j = \left| \left\{ i : k_{i1} > n' \text{ or } k_{i2} > n',\ d_{k_{i1}} \neq d_{k_{i2}},\ i = 1, \ldots, n_k \right\} \right|.$$

Bounding vectors and bounding front



Algorithm for multi-objective allocation of the shapes

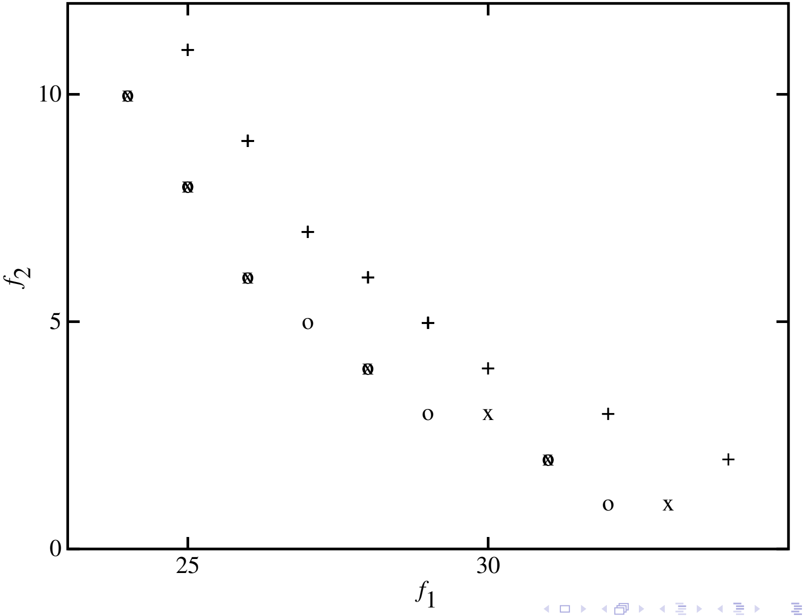
1. Form the first assignment in $\mathbf{x}$. Set $n' \leftarrow n + 1$
2. Repeat while $n' > 0$
   - If the current solution is complete ($n' \geq n$)
     - Set $n' \leftarrow n$.
     - Compute objective functions $f_{1x}(\mathbf{x})$ and $f_{2x}(\mathbf{x})$
     - If no solutions in the current approximation S of the efficient set dominate the current solution $\mathbf{x}$, add it to S.
     - If there are solutions in the current approximation S of the efficient set dominated by the current solution, remove them.
   - Otherwise
     - **Bounding**: Compute $\mathbf{b}^1(\mathbf{x}, n')$, $\mathbf{b}^2(\mathbf{x}, n')$, $\mathbf{b}^3(\mathbf{x}, n')$, or $B^4(\mathbf{x}, n')$.
     - **Pruning**: If $\mathbf{b}^1(\mathbf{x}, n')$, $\mathbf{b}^2(\mathbf{x}, n')$, $\mathbf{b}^3(\mathbf{x}, n')$, or every $\mathbf{b} \in B^4(\mathbf{x}, n')$ is dominated by a solution from the current approximation S of the efficient set, reduce n'.
   - **Branching or retracting** (depth first search): Update $x_{n'}$ by available number and increase n' or reduce n' if there are no further numbers available.
3. Find non-dominated solutions of the second problem.
4. Aggregate non-dominated solutions of two problems, and retain non-dominated solutions of the whole problem.

Numerical comparison

	$\mathbf{b}^1(\mathbf{x}, n')$		$\mathbf{b}^2(\mathbf{x}, n')$		$\mathbf{b}^3(\mathbf{x}, n')$		$\mathbb{B}^4(\mathbf{x}, n')$	
n_r	t, s	NFE	t, s	NFE	t, s	NFE	t, s	NFE
Example problem, $n_c = 3,\ n_1 = 6,\ n_2 = 6,\ n_3 = 4$								
6	0.15	237,440	0.06	930,469	0.08	704,048	0.01	187,343
7	1.44	28,118,029	0.33	6,074,083	0.49	4,576,302	0.04	656,290
8	9.49	192,605,603	1.54	29,855,682	2.26	23,101,727	0.18	2,593,238
Middle size problem, $n_c = 6,\ n_1 = 5,\ n_2 = 4,\ n_3 = 2,\ n_4 = 2,\ n_5 = 4,\ ,n_6 = 2$								
5	0.87	11,846,524	0.14	2,292,133	0.19	2,110,564	0.04	518,681
6	8.76	87,341,601	0.86	15,097,449	1.26	14,111,040	0.21	2,993,714
7	20.10	267,553,983	2.21	40,710,474	3.40	38,251,546	0.52	7,370,189
8	37.05	473,246,383	3.41	64,644,742	5.37	60,846,181	0.83	11,886,008
9	76.96	997,982,630	6.72	128,330,033	10.66	120,741,102	1.31	18,437,102
10	193.69	1,946,020,628	13.17	257,442,963	21.22	243,423,005	3.23	47,220,762
11	394.98	3,386,280,514	25.03	487,597,206	39.77	464,519,182	8.50	131,752,014
12	751.75	5,496,804,470	46.13	949,050,115	76.33	914,075,489	24.45	397,440,621
13	1175.44	8,072,969,995	58.66	1,201,936,218	97.00	1,145,782,878	15.70	236,090,687
14	1845.78	11,516,056,991	85.80	1,774,663,616	143.27	1,695,153,806	23.48	353,554,807
15	2746	15,764,528,221	120.29	2,493,528,143	204.61	2,385,705,518	33.12	498,906,138
16	3825	20,848,903,023	161.08	3,363,454,730	270.26	3,222,389,040	44.81	672,931,502
17	5182	26,788,986,132	209.02	4,388,173,880	352.65	4,208,888,470	57.97	876,392,519
18	6817	33,597,007,137	263.56	5,570,100,374	445.95	5,347,708,471	73.53	1,109,646,700
19	8670	41,280,000,441	330.56	6,912,015,181	526.19	6,272,785,312	92.77	1,373,682,420

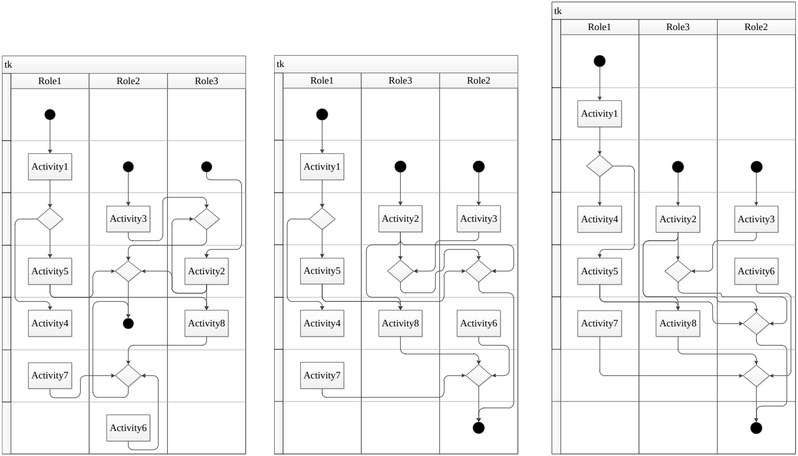
Pareto fronts for the example business process diagram

$n_r = 6$ (+), $n_r = 7$ (x), and $n_r = 8$ (o)



Solutions of example problem of shape allocation

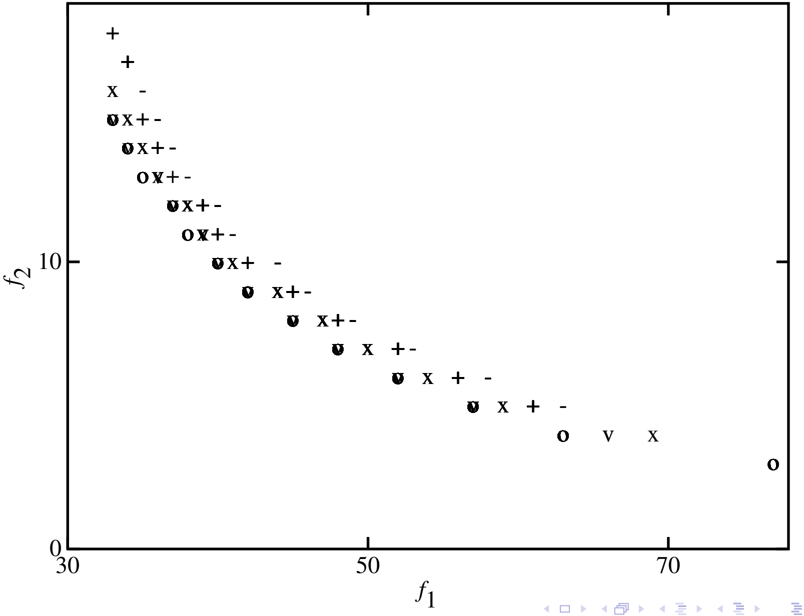
a) shortest total length $(24, 10)$
b) intermediate solution $(28, 4)$
c) the smallest number of violations with 8 rows $(32, 1)$



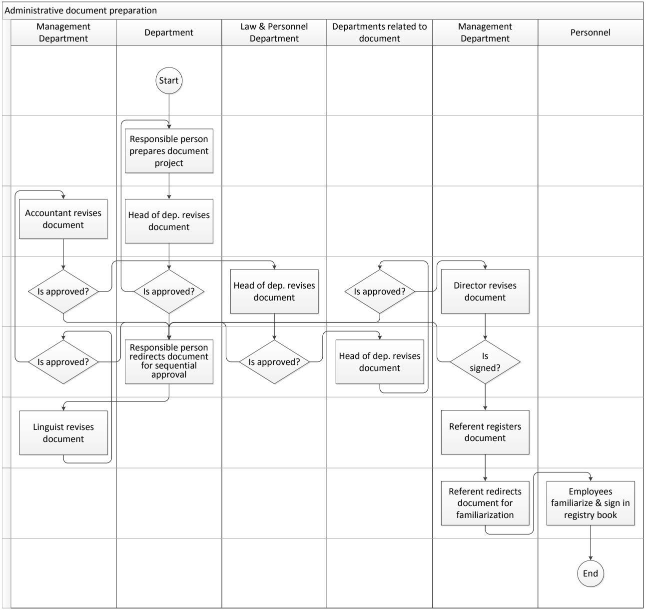
(a) (b) (c)

Pareto fronts for a middle size business process diagram

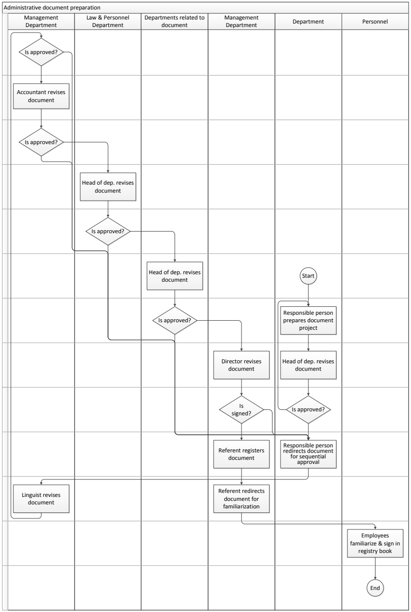
$n_r = 5$ (-), $n_r = 6$ (+), $n_r = 7$ (x), $n_r = 8$ (v), $n_r > 8$ (o)



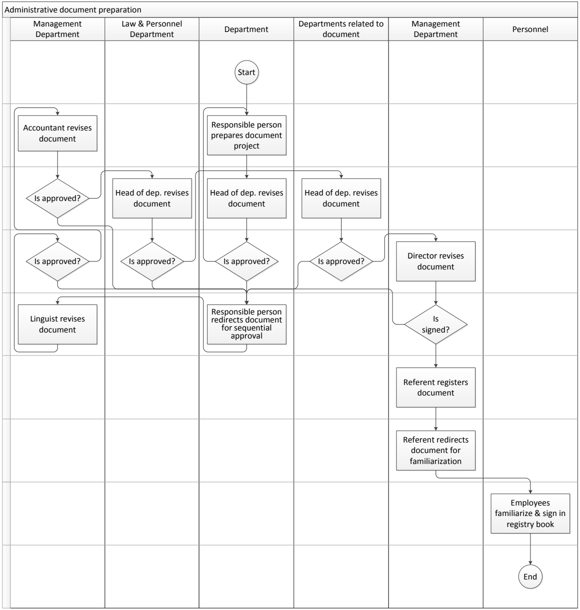
Solution with the shortest total length $(33, 15)$



Solution with the smallest number of flow violations $(77, 3)$



Non-dominated intermediate solution $(38, 11)$

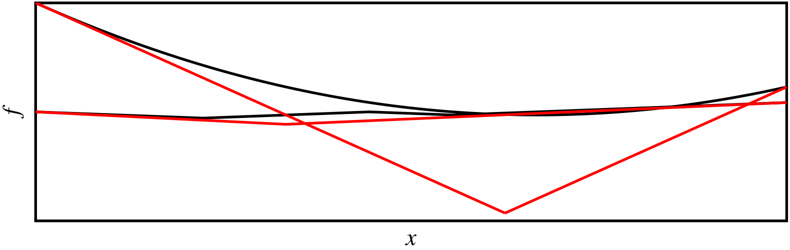


Univariate bi-objective Lipschitzian problems

- A class of Lipschitz objective functions $\Phi(L)$ is considered, i.e. $\mathbf{f}(x) = (f_1(x), f_2(x))^T \in \Phi(L)$, where

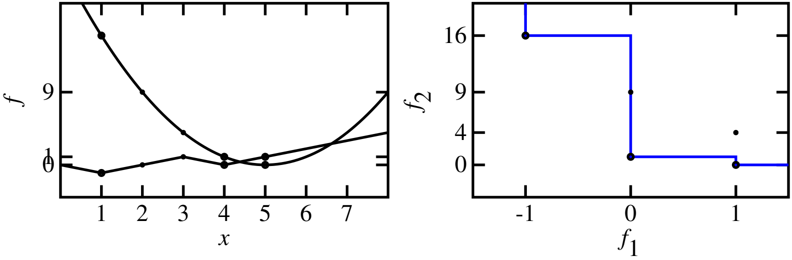
$$|f_k(x) - f_k(t)| \leq L_k \cdot |x - t|, \ k = 1, 2,$$

for $x, t \in \mathbb{A} = [a, b]$, $\mathbf{L} = (L_1, L_2)^T$, $L_k > 0$, $k = 1, 2$.



Upper bound

- Let us denote $\mathbf{Y}^n = (\mathbf{y}_1, \ldots, \mathbf{y}_n)^T$, $\mathbf{y}_i = \mathbf{f}(x_i)$, $i = 1, \ldots, n$. The subset of $\mathbb{D}(\mathbf{Y}^n) = \bigcup_{i=1}^{n} \{\mathbf{z} : \mathbf{z} \in \mathbb{R}^2, \mathbf{z} \geq \mathbf{y}_i\}$ which consists of weakly Pareto optimal solutions is called a trivial upper bound for $\mathbb{P}(\mathbf{f})_O$, and is denoted by $\mathbb{U}(\mathbf{Y}^n) \subset \mathbb{R}^2$.

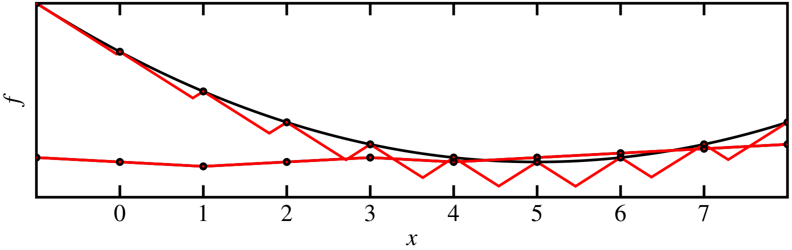


Lower bounding functions

- Functions $g_k(x)$, $k = 1, 2$, define the lower bounds for $f_k(x)$:

$$g_k(x) = \max\left(y_k^{o\,i} - L_k(x - x_{o\,i}),\ y_k^{o\,i+1} - L_k(x_{o\,i+1} - x)\right),$$
$$x_{o\,i} \le x \le x_{o\,i+1}, i = 1, \ldots, n-1,$$

where $x_{o\,i}$, $i = 1, \ldots, n$, denote increasingly ordered points x_i, $y_k^{o\,i}$ denote the corresponding values of the objective functions.

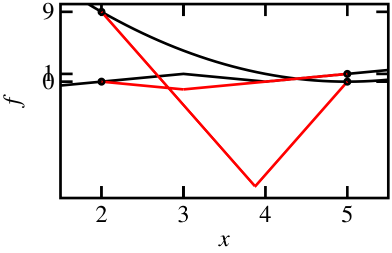


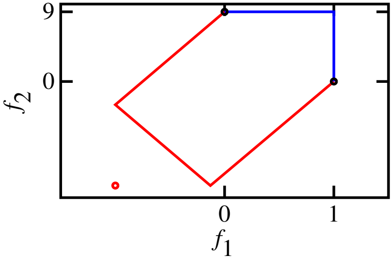
Lipschitz lower bound

- The Pareto front of the bi-objective problem

$$\min_{x_{o\,i} \le x \le x_{o\,i+1}} \mathbf{g}(x),\ \ \mathbf{g}(x) = (g_1(x), g_2(x))^T,$$

is called a local Lipschitz lower bound for $\mathbb{P}(\mathbf{f})_O$, and it is denoted as $\mathbb{V}_j$. The subset of $\bigcup_j \mathbb{V}_j$ constituted of non-dominated points is denoted by $\mathbb{V}(\mathbf{Y}^n)$ and called Lipschitz lower bound for $\mathbb{P}(\mathbf{f})_O$.

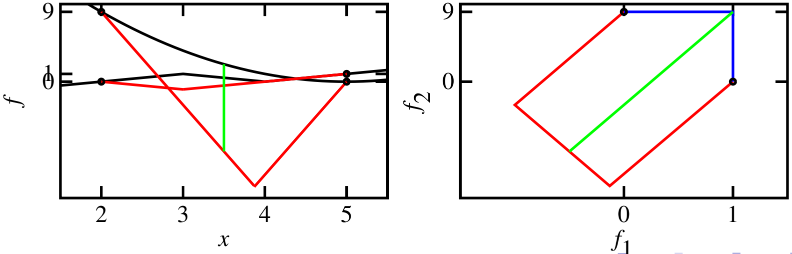




One-step optimal algorithm for univariate bi-objective Lipshitzian problems

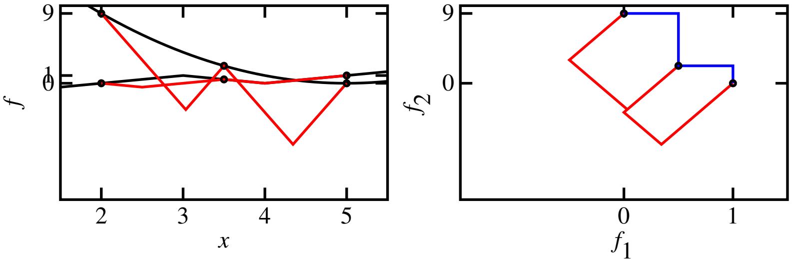
- The idea of the algorithm is to tighten the Lipschitz bounds for the non-dominated solutions, and to indicate the subintervals of $[a, b]$ with dominated objective vectors.
- Let us consider the $n+1$ optimization step where $x_{o\,i}$, $\mathbf{y}^{o\,i}$, $i = 1, \ldots, n$, are known.
- The gap ε_n between $\mathbb{V}(\mathbf{Y}^n)$ and $\mathbb{U}(\mathbf{Y}^n)$ can be computed

$$\varepsilon_n = \max_{\mathbf{Y} \in \mathbb{V}(\mathbf{Y}^n)} \min_{\mathbf{Z} \in \mathbb{U}(\mathbf{Y}^n)} \|\mathbf{Y} - \mathbf{Z}\|.$$



One-step optimal algorithm for univariate bi-objective Lipshitzian problems

- Since ε_n is computed as the maximum of gaps corresponding to the "non-dominated" subintervals, the algorithm is implemented with the idea and the implementation similar to those of the single objective global optimization algorithm by Shubert-Pijavskij.



Multivariate algorithm

- Direct generalization to multidimensional case is difficult. Usually branch and bound algorithm with hyper-rectangular or simplicial partitions is applied in global optimization.
- In this work we use hyper-rectangular partitions and diagonal approach.
- The concept of the algorithm is to tighten iteratively the lower Lipschitz bound, and to indicate the hyper-rectangles $\mathbb{A}_r$ which can be excluded from the further search because the whole $\mathbb{V}_r$ consists of dominated vectors.
- Trisection subdivision is used – the selected hyper-rectangle is subdivided into three parts by two parallel hyper-planes, and computations of $\mathbf{f}(\cdot)$ at two points is sufficient for the continuation of the algorithm.
- The tightness of lower Lipschitz bound $\mathbb{V}(\mathbb{Y}_R, \mathbb{A}_{[R]})$ can be assessed similarly as local lower Lipschitz bounds but using also the information on $\mathbb{U}(\mathbb{Y}_R)$.

Multivariate algorithm

- The global tolerance for $\mathbb{P}(\mathbf{f}, \mathbb{A}_r)_O$ is denoted by $\tilde{\Delta}_r = \tilde{\Delta}(\mathbf{f}(\mathbf{a}(r)), \mathbf{f}(\mathbf{b}(r)), \mathbb{A}_r)$ and defined as follows:
  - if $\mathbf{f}(\mathbf{a}_r)$, $\mathbf{f}(\mathbf{b}_r)$ are not dominated by the elements of $\{\mathbf{f}(\mathbf{a}_i), \ \mathbf{f}(\mathbf{b}_i), i = 1, \dots, R\}$, then $\tilde{\Delta}_r = \Delta_r$,
  - if all vectors belonging to $\mathbb{V}_r$ are dominated by some of elements of $\{\mathbf{f}(\mathbf{a}_i), \mathbf{f}(\mathbf{b}_i), i = 1, \dots, R$, then $\tilde{\Delta}_r = 0$,
  - in other cases the line segment $\mathbb{V}_r$ intersects with $\mathbb{U}(\mathbb{Y}_R)$, and

$$\tilde{\Delta}_r = \max_{\xi \in \mathbb{V}_r} \min_{\zeta \in \mathbb{U}(\mathbb{Y}_R)} \|\xi - \zeta\|.$$

- A hyper-rectangle $\mathbb{A}_{\hat{r}}$ where

$$\hat{r} = \arg\max_r \ \tilde{\Delta}_r,$$

  is selected for partition.
- The algorithm is stopped when

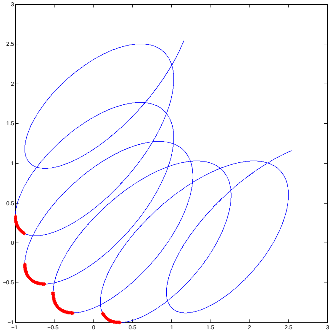
$$\max_r \ \tilde{\Delta}_r < \varepsilon$$

  or after the predefined number of function evaluations.

Illustration on Rastrigin functions

$$\min_{x \in [-1,1]} \mathbf{f}(x),$$

$$\begin{aligned} f_1(x) &= ((x+0.5)^2 - \cos(18(x+0.5)))/21, \\ f_2(x) &= ((x-0.5)^2 - \cos(18(x-0.5)))/21. \end{aligned}$$



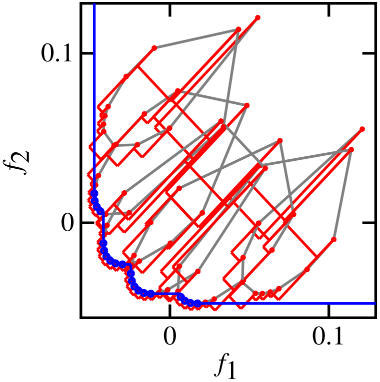
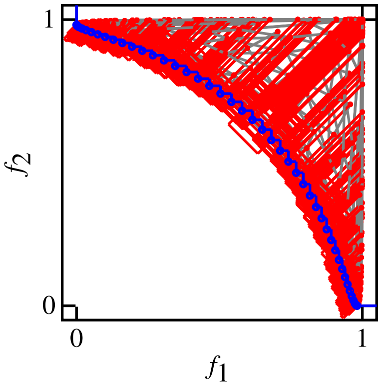


Illustration on two dimensional Fonseca problem

$$\min_{\mathbf{x}\in[-4,4]^2} \mathbf{f}(\mathbf{x}),$$

$$\begin{aligned} f_1(\mathbf{x}) &= 1 - e^{-\sum_{i=1}^{n}(x_i - 1/\sqrt{n})}, \\ f_2(\mathbf{x}) &= 1 - e^{-\sum_{i=1}^{n}(x_i + 1/\sqrt{n})}. \end{aligned}$$



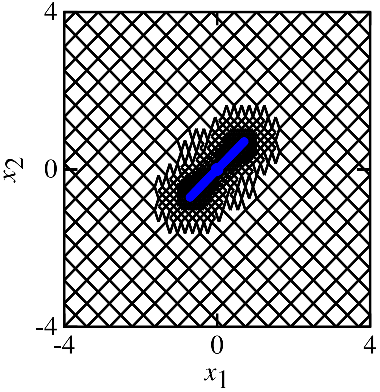


Illustration on example problem 1 from (Evtushenko, Posypkin, 2013)

$$\min_{\mathbf{x}\in[0,2]^2} \mathbf{f}(\mathbf{x}),$$

$$f_1(\mathbf{x}) = x_0,$$
$$f_2(\mathbf{x}) = (\min(|x_0 - 1|, 1.5 - x_0) + x_1 + 1)/2.$$

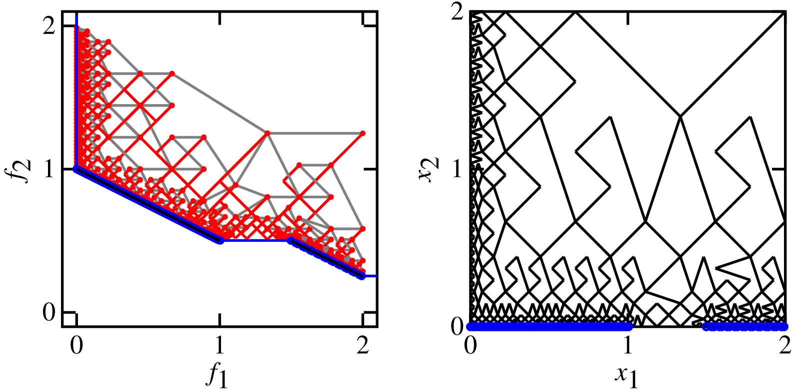
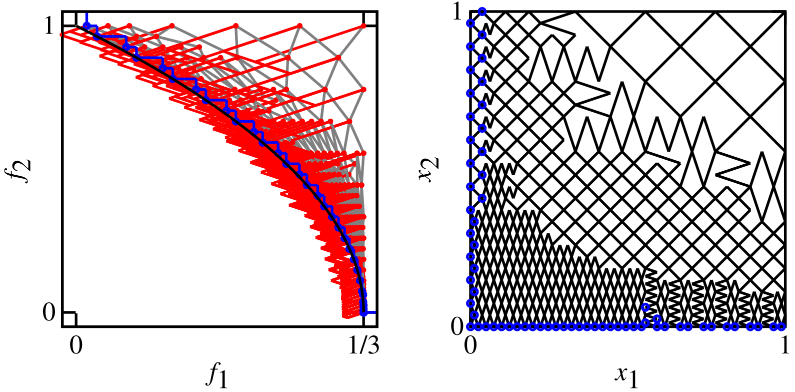


Illustration on example problem 2 from (Evtushenko, Posypkin, 2013)

$$\min_{\mathbf{x}\in[0,1]^2} \mathbf{f}(\mathbf{x}),$$

$$\begin{aligned} f_1(\mathbf{x}) &= ((x_0-1)x_1^2+1)/3, \\ f_2(\mathbf{x}) &= x_1. \end{aligned}$$



Numerical comparison

Method	NFE	ε	ngen	np	hv	ud
			Problem 1			
Genetic algorithm	500		500	221	3.27	
Monte Carlo	500			22	3.38	
Nonuniform covering	490	0.07		36	3.42	
Multiobjective trisection	479	0.035		83	3.61	
			Problem 2			
Genetic algorithm	500		500	104	0.312	1.116
Monte Carlo	500			67	0.300	1.277
Nonuniform covering	515	0.0675		29	0.306	0.210
Multiobjective trisection	513	0.0675		65	0.310	0.178

$$ud = \sqrt{\sum_{i=1}^{np}(d_i - d)^2}, \; d = \frac{1}{np}\sum_{i=1}^{np} d_i, \; d_i = \min_{i \neq j} d_{ij}.$$

Thank you for your attention