Metaheuristic Optimization

16. Constraint Handling

Thomas Weise · 汤卫思

tweise@hfuu.edu.cn · http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Faculty of Computer Science and Technology
Institute of Applied Optimization
230601 Shushan District, Hefei, Anhui, China
Econ. & Tech. Devel. Zone, Jinxiu Dadao 99

合肥学院 南艳湖校区/南2区
计算机科学与技术系
应用优化研究所
中国 安徽省 合肥市 蜀山区 230601
经济技术开发区 锦绣大道99号

Outline

website

Section Outline

Constraints

- Besides optimization criteria, there often are feasibility critera defining whether a candidate solution makes sense or not

Constraints

- Besides optimization criteria, there often are feasibility critera defining whether a candidate solution makes sense or not

Definition (Feasibility)

In an optimization problem, $q \geq 0$ inequality constraints g and $r \geq 0$ equality constraints h may be imposed additionally to the objective functions. Then, a candidate solution x is *feasible*, if and only if it fulfils all constraints:

$$\text{isFeasible}(x) \Leftrightarrow \begin{array}{rcll} g_i(x) & \geq & 0 & \forall i \in 1 \ldots q \wedge \\ h_j(x) & = & 0 & \forall j \in 1 \ldots r \end{array} \tag{1}$$

Section Outline

Genotype/Phenotype Methods

- Ensure that all phenotypes are always feasible

Genotype/Phenotype Methods

- Ensure that all phenotypes are always feasible
  1. Coding: Use a GPM that constructs feasible phenotypes only [1–3]

Genotype/Phenotype Methods

- Ensure that all phenotypes are always feasible
  1. Coding: Use a GPM that constructs feasible phenotypes only [1–3]
  2. Repairing: Repair infeasible phenotypes, i.e., , turn them feasible [3, 4]

Genotype/Phenotype Methods

- Ensure that all phenotypes are always feasible
  1. Coding: Use a GPM that constructs feasible phenotypes only [1–3]
  2. Repairing: Repair infeasible phenotypes, i.e., , turn them feasible [3, 4]

✔ Feasibility "removed" from optimization process $\Longrightarrow$ problem complexity kept low

Genotype/Phenotype Methods

- Ensure that all phenotypes are always feasible
  1. Coding: Use a GPM that constructs feasible phenotypes only [1–3]
  2. Repairing: Repair infeasible phenotypes, i.e., , turn them feasible [3, 4]

✔ Feasibility "removed" from optimization process $\Longrightarrow$ problem complexity kept low

✘ Requires knowledge about "what makes a candidate solution feasible"

Death Penalty

- Throw away infeasible candidate solutions [5, 6]

Death Penalty

- Throw away infeasible candidate solutions [5, 6]

✔ Very easy to implement

Death Penalty

- Throw away infeasible candidate solutions [5, 6]

✔ Very easy to implement

✘ Only possible if many (most) candidate solutions are feasible, otherwise

Death Penalty

- Throw away infeasible candidate solutions [5, 6]

✔ Very easy to implement

✘ Only possible if many (most) candidate solutions are feasible, otherwise
  - Much effort may be wasted just to discover feasible individuals

Death Penalty

- Throw away infeasible candidate solutions [5, 6]

✔ Very easy to implement

✘ Only possible if many (most) candidate solutions are feasible, otherwise
  - Much effort may be wasted just to discover feasible individuals
  - The transition from one feasible individual to another one may be unlikely, the objective landscape becomes rugged with large neutral planes at the worst possible fitness levels

Death Penalty

- Throw away infeasible candidate solutions [5, 6]

✔ Very easy to implement

✘ Only possible if many (most) candidate solutions are feasible, otherwise
  - Much effort may be wasted just to discover feasible individuals
  - The transition from one feasible individual to another one may be unlikely, the objective landscape becomes rugged with large neutral planes at the worst possible fitness levels
  - The information gained from sampling infeasible individuals is lost!

Penalty Function

- Apply evolutionary pressure to guide the search from infeasible to feasible individuals

Penalty Function

- Apply evolutionary pressure to guide the search from infeasible to feasible individuals
- Idea by Courant [7] for single-objective optimization in the 1940s: Add a penalty to original objective or fitness value ν' [7–11]

Penalty Function

- Apply evolutionary pressure to guide the search from infeasible to feasible individuals
- Idea by Courant [7] for single-objective optimization in the 1940s: Add a penalty to original objective or fitness value ν' [7–11]
- Examples
  1. If $h > 0$ or $h < 0$, there should be a penalty:

$$\nu(p) = \nu'(p) + \sum_{i=1}^{r} z_i * [h_i(p.x)]^2 \tag{2}$$

Penalty Function

- Apply evolutionary pressure to guide the search from infeasible to feasible individuals
- Idea by Courant [7] for single-objective optimization in the 1940s: Add a penalty to original objective or fitness value ν' [7–11]
- Examples
  1. If $h > 0$ or $h < 0$, there should be a penalty:

$$\nu(p) = \nu'(p) + \sum_{i=1}^{r} z_i * [h_i(p.x)]^2 \tag{2}$$

  2. The closer g gets to 0, the larger should the penalty be (works if g is always > 0)

$$\nu(p) = \nu'(p) + \sum_{i=1}^{q} z * [g_i(p.x)]^{-1} \tag{3}$$

Penalty Function

- Apply evolutionary pressure to guide the search from infeasible to feasible individuals
- Idea by Courant [7] for single-objective optimization in the 1940s: Add a penalty to original objective or fitness value ν' [7–11]
- Examples
  1. If $h > 0$ or $h < 0$, there should be a penalty:
  2. The closer g gets to 0, the larger should the penalty be (works if g is always > 0)

Penalty Function

- Apply evolutionary pressure to guide the search from infeasible to feasible individuals
- Idea by Courant [7] for single-objective optimization in the 1940s: Add a penalty to original objective or fitness value ν' [7–11]
- Examples
  1. If $h > 0$ or $h < 0$, there should be a penalty:
  2. The closer g gets to 0, the larger should the penalty be (works if g is always > 0)

Penalty Function

- Apply evolutionary pressure to guide the search from infeasible to feasible individuals
- Idea by Courant [7] for single-objective optimization in the 1940s: Add a penalty to original objective or fitness value ν' [7–11]

✔ Easy to implement

Penalty Function

- Apply evolutionary pressure to guide the search from infeasible to feasible individuals
- Idea by Courant [7] for single-objective optimization in the 1940s: Add a penalty to original objective or fitness value ν' [7–11]

✔ Easy to implement

✘ Knowledge about behavior of objective functions, fitness, and constraint functions necessary

Penalty Function

- Apply evolutionary pressure to guide the search from infeasible to feasible individuals
- Idea by Courant [7] for single-objective optimization in the 1940s: Add a penalty to original objective or fitness value ν' [7–11]

✔ Easy to implement

✘ Knowledge about behavior of objective functions, fitness, and constraint functions necessary

- Good and a nice method for single-objective optimization, but a bit harder to understand in multi-objective scenarios

Constraints as Objectives

- Treat constraints as additional objectives

Constraints as Objectives

- Treat constraints as additional objectives
- Examples
  1. The objective representing the "greater-equal constraint" g as should be 0 (best) as long as g is met (≥ 0) and > 0 otherwise:

$$f^{\geq}(x) = -\min\{g(x), 0\} \tag{2}$$

Constraints as Objectives

- Treat constraints as additional objectives
- Examples
  1. The objective representing the "greater-equal constraint" g as should be 0 (best) as long as g is met (≥ 0) and > 0 otherwise:

$$f^{\geq}(x) = -\min\{g(x), 0\} \tag{2}$$

  2. The objective representing "equality constraint" should be ϵ (or 0) as long as the constraint is met with a given precision ϵ, and larger otherwise

$$f^{=}(x) = \max\{|h(x)|, \epsilon\} \text{ with an } \epsilon > 0 \tag{3}$$

Constraints as Objectives

- Treat constraints as additional objectives
- Examples
  1. The objective representing the "greater-equal constraint" g as should be 0 (best) as long as g is met (≥ 0) and > 0 otherwise:

$$f^{\geq}(x) = -\min\{g(x), 0\} \tag{2}$$

  2. The objective representing "equality constraint" should be ϵ (or 0) as long as the constraint is met with a given precision ϵ, and larger otherwise

$$f^{=}(x) = \max\{|h(x)|, \epsilon\} \text{ with an } \epsilon > 0 \tag{3}$$

✔ Very easy to realize in a MOEA

Constraints as Objectives

- Treat constraints as additional objectives
- Examples
  1. The objective representing the "greater-equal constraint" g as should be 0 (best) as long as g is met (≥ 0) and > 0 otherwise:

$$f^{\geq}(x) = -\min\{g(x), 0\} \tag{2}$$

  2. The objective representing "equality constraint" should be ϵ (or 0) as long as the constraint is met with a given precision ϵ, and larger otherwise

$$f^{=}(x) = \max\{|h(x)|, \epsilon\} \text{ with an } \epsilon > 0 \tag{3}$$

✔ Very easy to realize in a MOEA

✘ Too many objectives may make the problem very hard to solve (many-objective optimization optimization [3, 12–19])

Method of Inequalities

- Method of Inequalities (MOI) [20–25]: instead of defining constraints explicitly, goal ranges $[\underline{r}i, \overline{r}i]$ are defined for each of the n objective function f_i

Method of Inequalities

- Method of Inequalities (MOI) [20–25]: instead of defining constraints explicitly, goal ranges $[\underline{r}i, \overline{r}i]$ are defined for each of the n objective function f_i
- Pohlheim [26] combines this with Pareto ranking by introducing three classes of candidate solutions

Method of Inequalities

- Method of Inequalities (MOI) [20–25]: instead of defining constraints explicitly, goal ranges $[\underline{r}i, \overline{r}i]$ are defined for each of the n objective function f_i
- Pohlheim [26] combines this with Pareto ranking by introducing three classes of candidate solutions: a candidate solution either...

Method of Inequalities

- Method of Inequalities (MOI) [20–25]: instead of defining constraints explicitly, goal ranges $[\underline{r}i, \overline{r}i]$ are defined for each of the n objective function f_i
- Pohlheim [26] combines this with Pareto ranking by introducing three classes of candidate solutions: a candidate solution either...
  1. fulfills all goals, i.e.,

$$\underline{r}i \leq f_i(x) \leq \overline{r}i \forall i \in 1 \ldots n \tag{4}$$

Method of Inequalities

- Method of Inequalities (MOI) [20–25]: instead of defining constraints explicitly, goal ranges $[\underline{r}i, \overline{r}i]$ are defined for each of the n objective function f_i
- Pohlheim [26] combines this with Pareto ranking by introducing three classes of candidate solutions: a candidate solution either...
  1. fulfills all goals, i.e.,

$$\underline{r}i \leq f_i(x) \leq \overline{r}i \forall i \in 1 \ldots n \tag{4}$$

  2. fulfills some (but not all) of the goals

$$(\exists i \in 1 \ldots n : \ \underline{r}i \leq f_i(x) \leq \overline{r}i) \wedge (\exists j \in 1 \ldots n : \ (f_j(x) < \underline{r}j) \vee (f_j(x) > \overline{r}j)) \tag{5}$$

Method of Inequalities

- Method of Inequalities (MOI) [20–25]: instead of defining constraints explicitly, goal ranges $[\underline{r}i, \overline{r}i]$ are defined for each of the n objective function f_i
- Pohlheim [26] combines this with Pareto ranking by introducing three classes of candidate solutions: a candidate solution either...
  1. fulfills all goals, i.e.,

$$\underline{r}i \leq f_i(x) \leq \overline{r}i \forall i \in 1 \ldots n \tag{4}$$

  2. fulfills some (but not all) of the goals

$$(\exists i \in 1 \ldots n : \ \underline{r}i \leq f_i(x) \leq \overline{r}i) \wedge (\exists j \in 1 \ldots n : \ (f_j(x) < \underline{r}j) \vee (f_j(x) > \overline{r}j)) \tag{5}$$

  3. fulfills none of the goals, i.e.,

$$(f_i(x) < \underline{r}i) \vee (f_i(x) > \overline{r}i) \, \forall i \in 1 \ldots n \tag{6}$$

Method of Inequalities

- New comparison mechanism

Method of Inequalities

- New comparison mechanism
  1. Solutions that fulfill all goals are preferred to solutions which fulfill only some goals

Method of Inequalities

- New comparison mechanism
  1. Solutions that fulfill all goals are preferred to solutions which fulfill only some goals
  2. Solutions which fulfill only some goals are preferred to solutions which fulfill no goals

Method of Inequalities

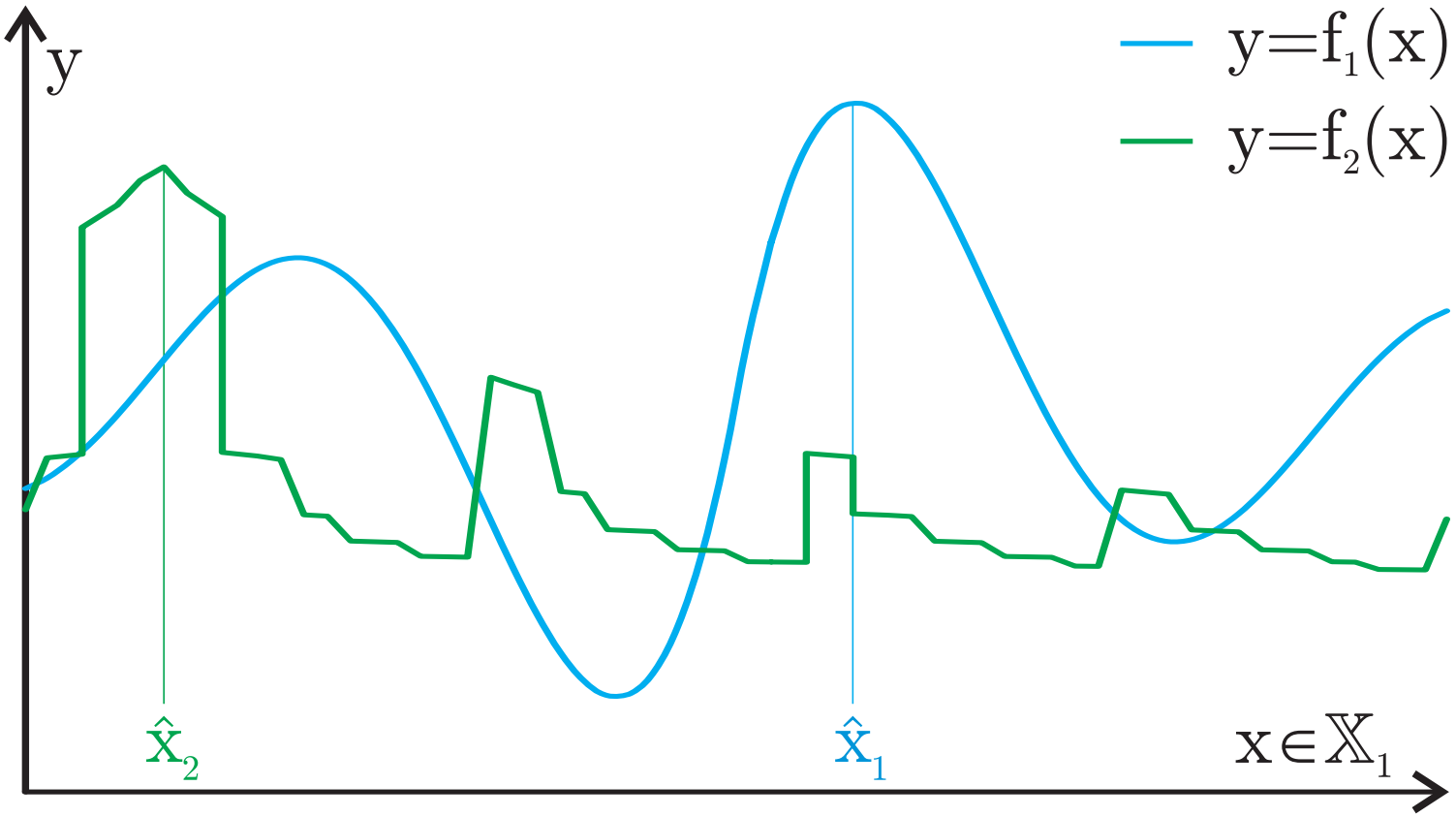
- New comparison mechanism
  1. Solutions that fulfill all goals are preferred to solutions which fulfill only some goals
  2. Solutions which fulfill only some goals are preferred to solutions which fulfill no goals
  3. Only solutions in the same group are compared according to the Pareto relationship

Method of Inequalities

- New comparison mechanism
  1. Solutions that fulfill all goals are preferred to solutions which fulfill only some goals
  2. Solutions which fulfill only some goals are preferred to solutions which fulfill no goals
  3. Only solutions in the same group are compared according to the Pareto relationship
- Pareto ranking is performed based on this comparison method

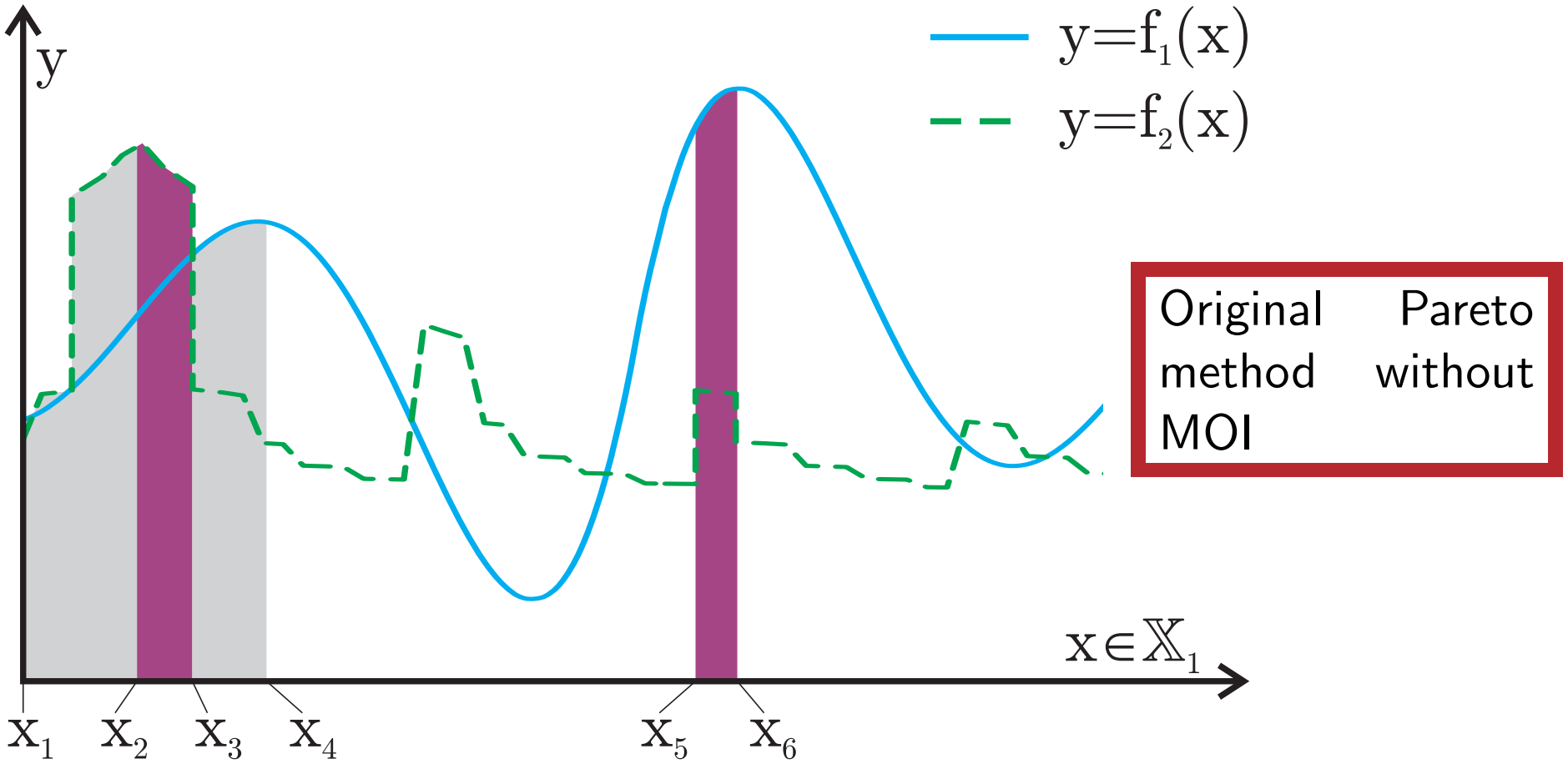
Example A: Two 1d-Functions

- Two 1-dimensional functions subject to maximization:
  $\vec{f} = \{f_1, f_2\}, f_i : \mathbb{R} \mapsto \mathbb{R} \, \forall i \in \{1, 2\}$



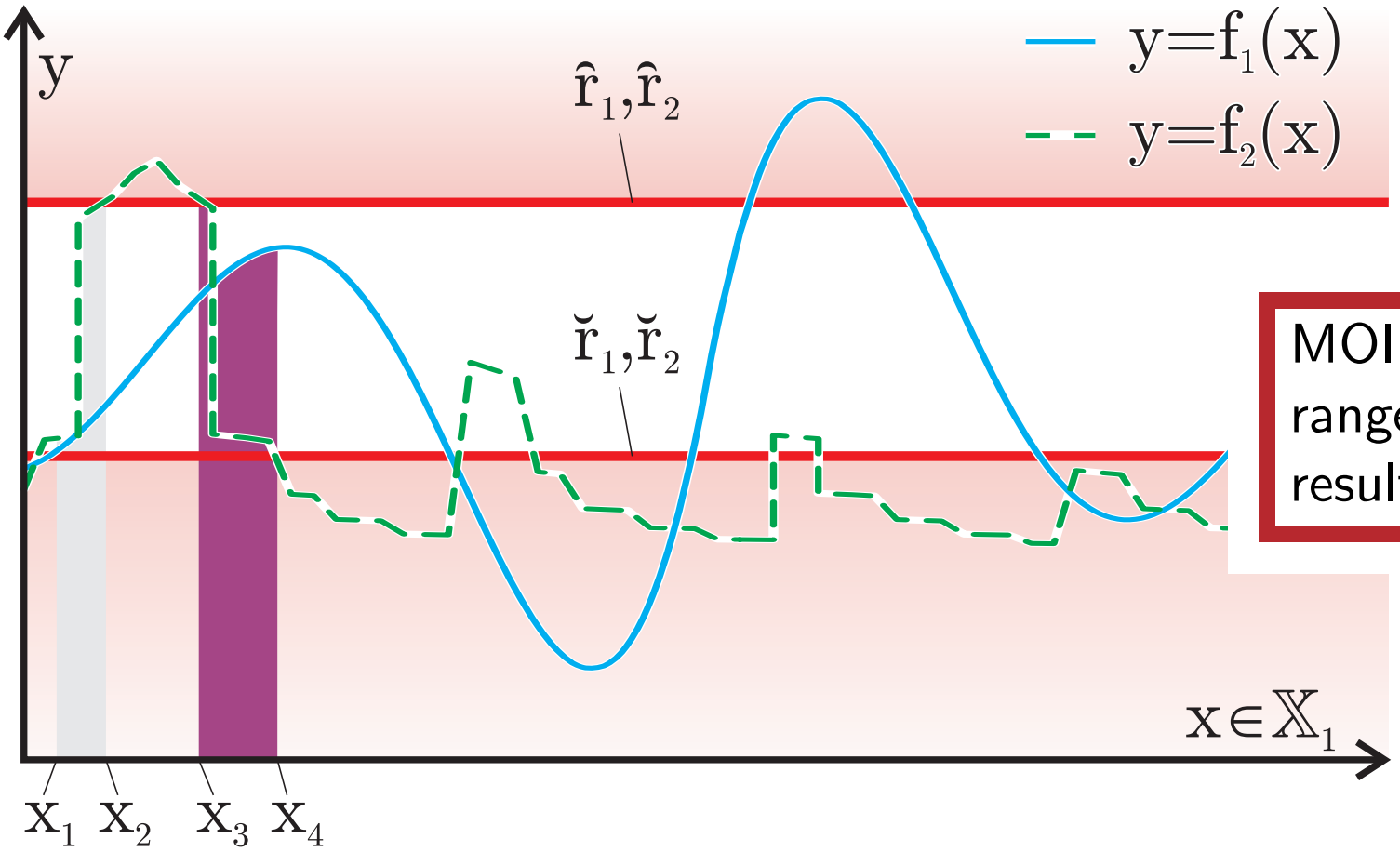
Example A: Two 1d-Functions

- Two 1-dimensional functions subject to maximization:
  $\vec{f} = \{f_1, f_2\}, f_i : \mathbb{R} \mapsto \mathbb{R} \, \forall i \in \{1, 2\}$



Example A: Two 1d-Functions

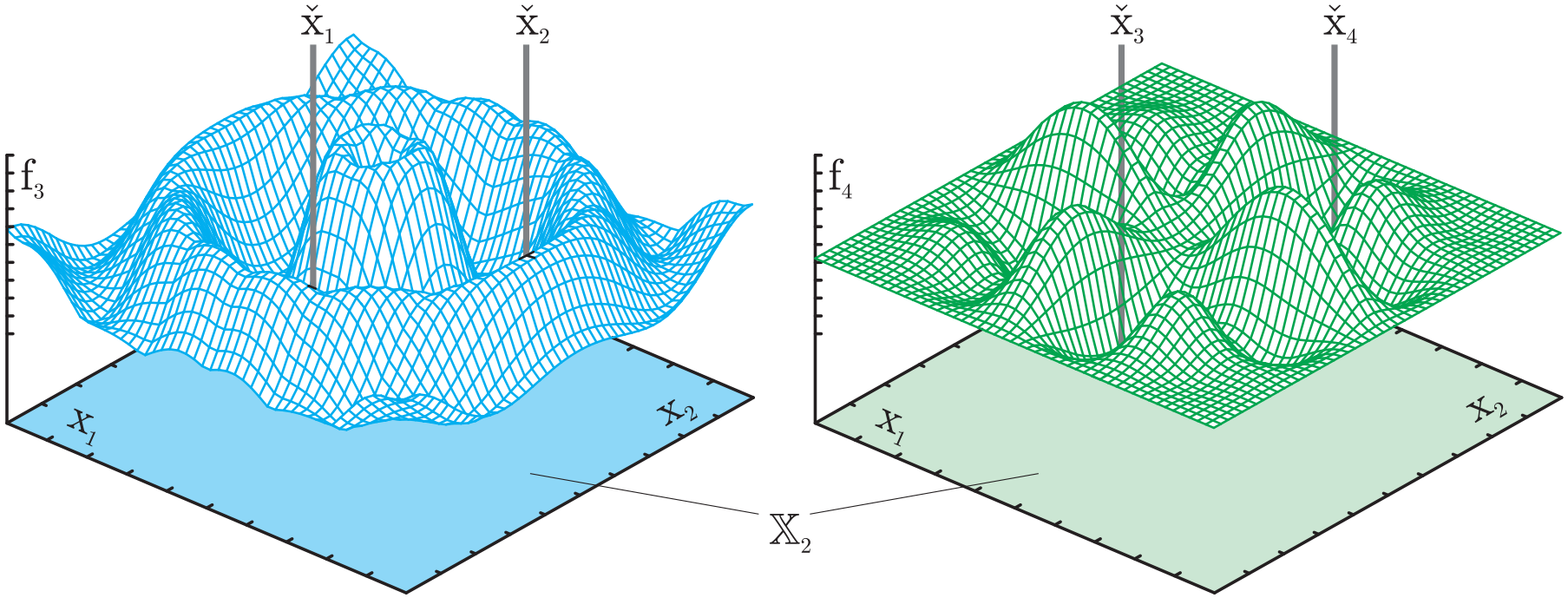
- Two 1-dimensional functions subject to maximization: $\vec{f} = \{f_1, f_2\}, f_i : \mathbb{R} \mapsto \mathbb{R} \, \forall i \in \{1, 2\}$



MOI with goal ranges $[\underline{r}i, \overline{r}i]$ – results change

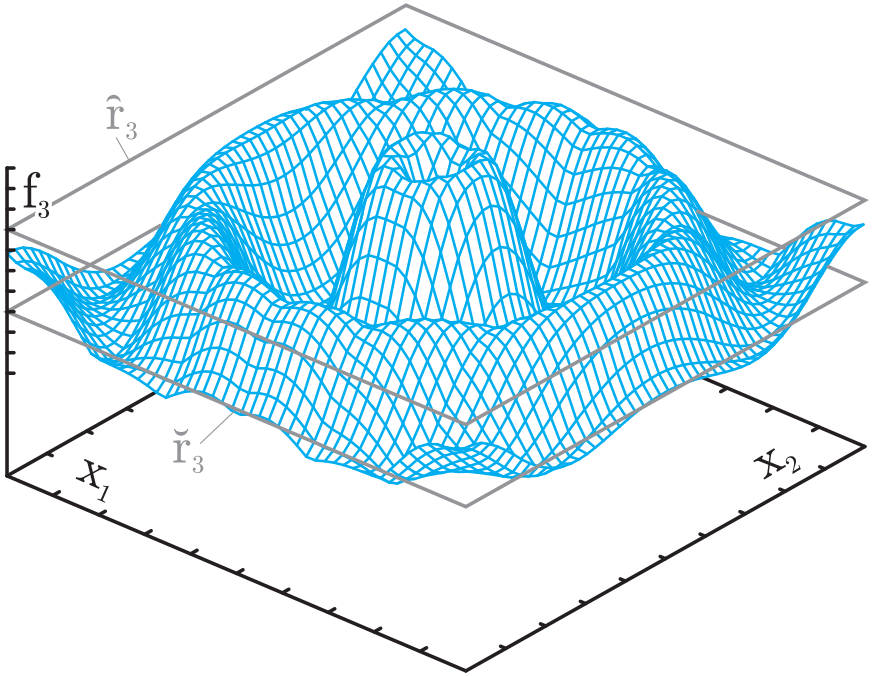
Example B: Two 2d-Functions

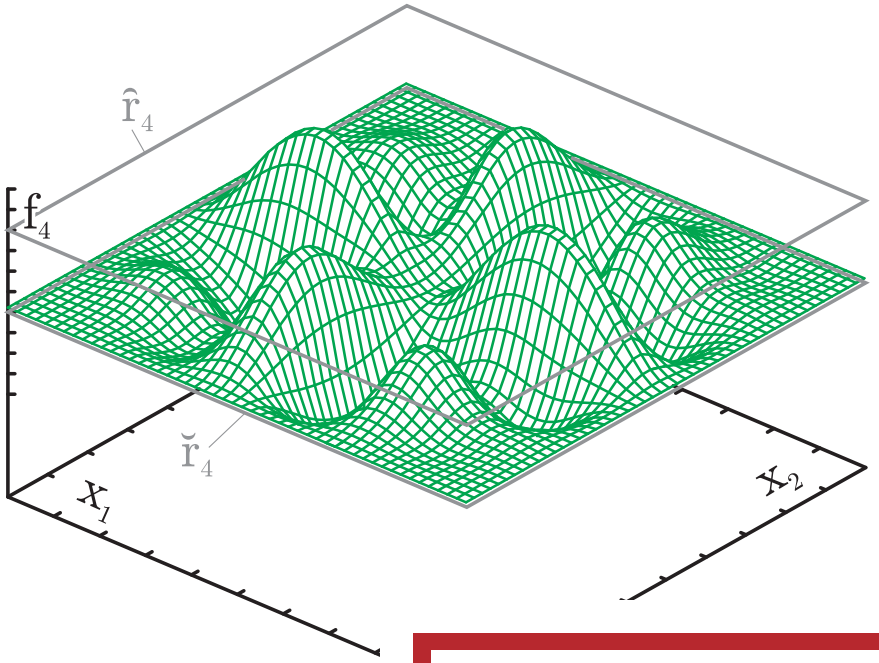
- Two 2-dimensional functions to minimization:
  $\vec{f} = \{f_3, f_4\}, f_i : \mathbb{R}^2 \mapsto \mathbb{R}\, \forall i \in \{3, 4\}$



Example B: Two 2d-Functions

- Two 2-dimensional functions to minimization:
  $\vec{f} = \{f_3, f_4\}, f_i : \mathbb{R}^2 \mapsto \mathbb{R}\, \forall i \in \{3, 4\}$

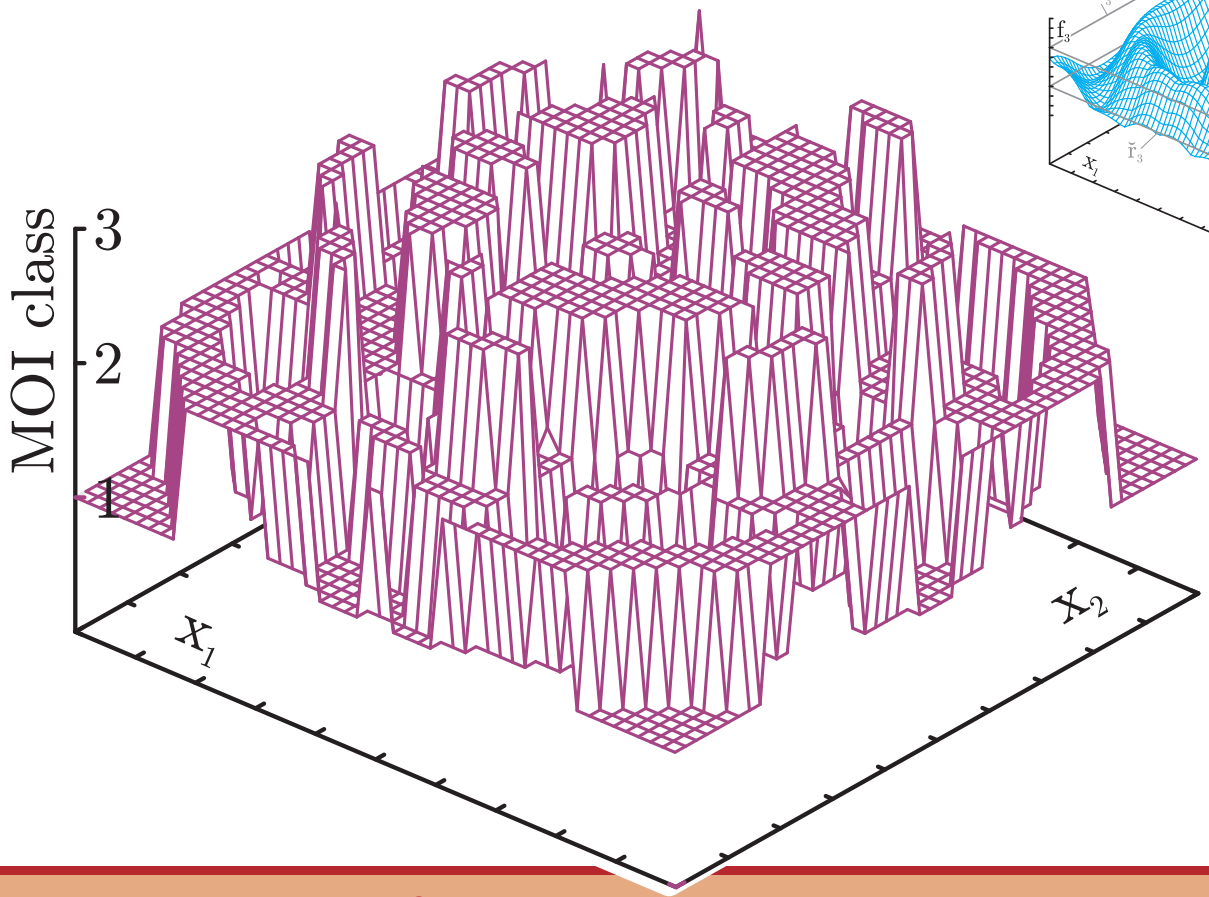


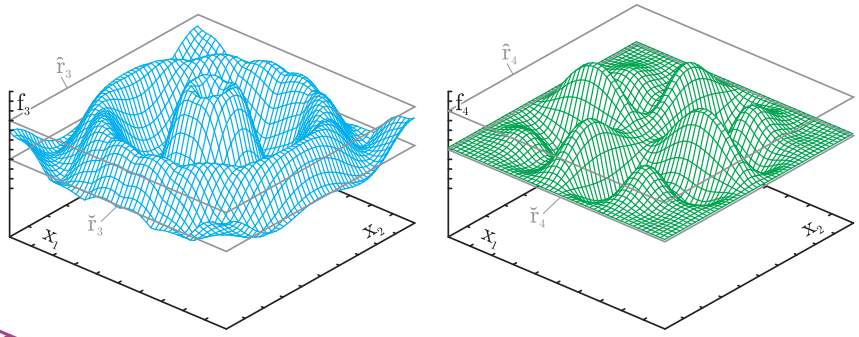


Goal ranges $[\underline{r}i, \overline{r}i]$

Example B: Two 2d-Functions

- Two 2-dimensional functions to minimization:
  $\vec{f} = \{f_3, f_4\}, f_i : \mathbb{R}^2 \mapsto \mathbb{R}\, \forall i \in \{3, 4\}$

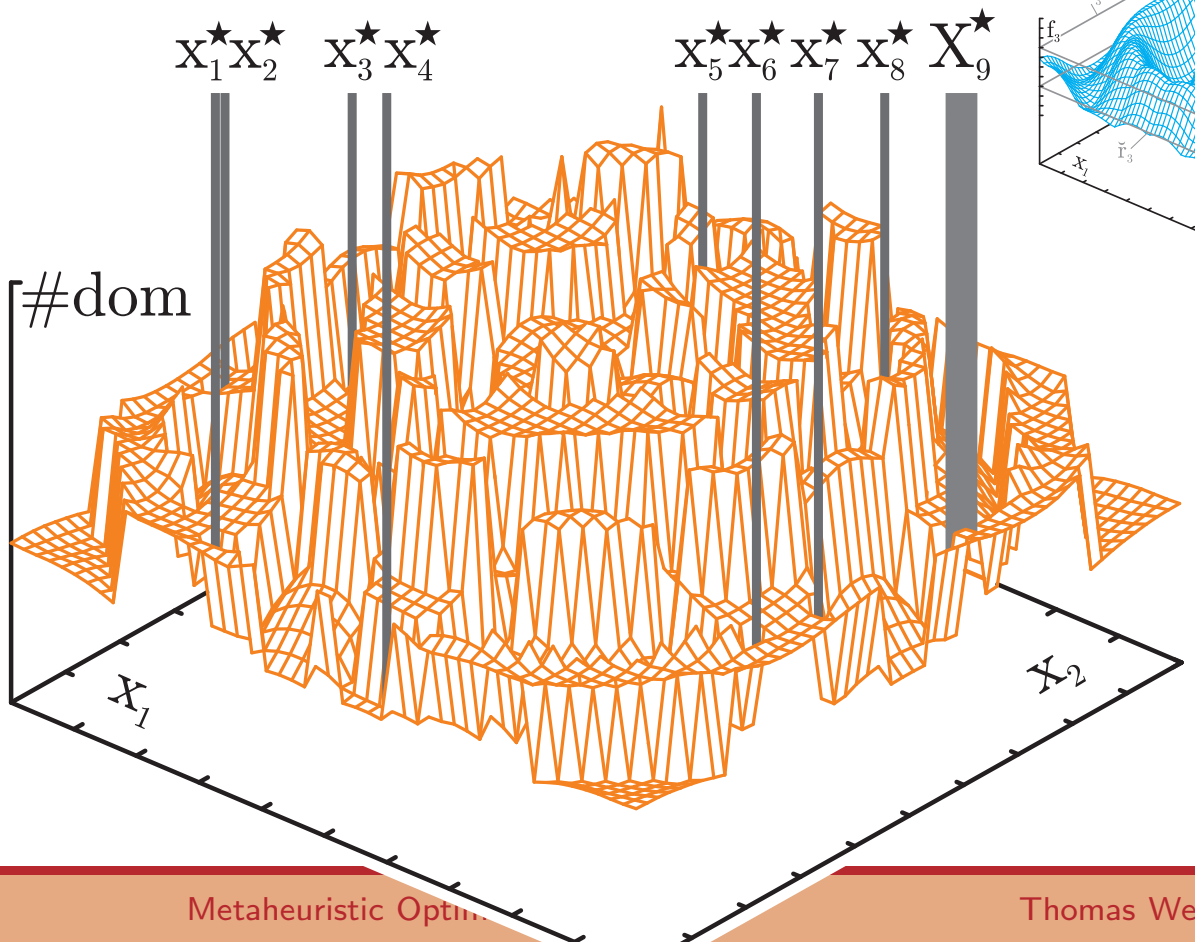


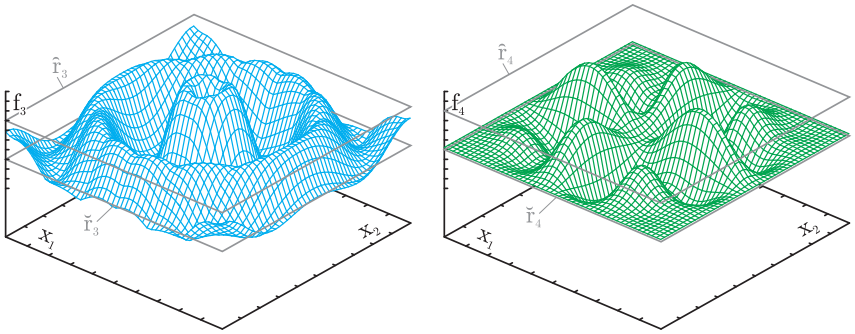


The three different classes

Example B: Two 2d-Functions

- Two 2-dimensional functions to minimization:
  $\vec{f} = \{f_3, f_4\}, f_i : \mathbb{R}^2 \mapsto \mathbb{R}\, \forall i \in \{3, 4\}$





The MOI-domination ranks and optima

$$\#\mathbf{dom}(\mathbf{x}, \mathbf{X}) = |\{\mathbf{x}' : \ (x' \in X) \wedge (x' \dashv_c x)\}|$$

Method of Inequalities

✔ Easy to implement

Method of Inequalities

- ✔ Easy to implement
- ✔ Fits perfectly well to Pareto ranking / can easily be integrated into this process

Method of Inequalities

- ✔ Easy to implement
- ✔ Fits perfectly well to Pareto ranking / can easily be integrated into this process
- ✘ Constraints must be formulated as objective function ranges

Constraint Domination

- Constraint Domination [27–29] adapts the Pareto comparison to also consider constraints

Constraint Domination

- Constraint Domination [27–29] adapts the Pareto comparison to also consider constraints
- A candidate solution x_1 is prefered in comparison to an element x_2 if...

Constraint Domination

- Constraint Domination [27–29] adapts the Pareto comparison to also consider constraints
- A candidate solution x_1 is prefered in comparison to an element x_2 if...
  1. x_1 is feasible while x_2 is not,

Constraint Domination

- Constraint Domination [27–29] adapts the Pareto comparison to also consider constraints
- A candidate solution x_1 is prefered in comparison to an element x_2 if...
  1. x_1 is feasible while x_2 is not,
  2. x_1 and x_2 both are infeasible but x_1 has a smaller overall constraint violation, or

Constraint Domination

- Constraint Domination [27–29] adapts the Pareto comparison to also consider constraints
- A candidate solution x_1 is prefered in comparison to an element x_2 if...
  1. x_1 is feasible while x_2 is not,
  2. x_1 and x_2 both are infeasible but x_1 has a smaller overall constraint violation, or
  3. x_1 and x_2 are both feasible but x_1 dominates x_2.

Constraint Domination

- Constraint Domination [27–29] adapts the Pareto comparison to also consider constraints
- A candidate solution x_1 is prefered in comparison to an element x_2 if...
  1. x_1 is feasible while x_2 is not,
  2. x_1 and x_2 both are infeasible but x_1 has a smaller overall constraint violation, or
  3. x_1 and x_2 are both feasible but x_1 dominates x_2.

✔ Easy to implement

Constraint Domination

- Constraint Domination [27–29] adapts the Pareto comparison to also consider constraints
- A candidate solution x_1 is prefered in comparison to an element x_2 if...
  1. x_1 is feasible while x_2 is not,
  2. x_1 and x_2 both are infeasible but x_1 has a smaller overall constraint violation, or
  3. x_1 and x_2 are both feasible but x_1 dominates x_2.

✔ Easy to implement

✔ Fits perfectly well to Pareto ranking and existing MOEAs

Constraint Domination

- Constraint Domination [27–29] adapts the Pareto comparison to also consider constraints
- A candidate solution x_1 is prefered in comparison to an element x_2 if...
  1. x_1 is feasible while x_2 is not,
  2. x_1 and x_2 both are infeasible but x_1 has a smaller overall constraint violation, or
  3. x_1 and x_2 are both feasible but x_1 dominates x_2.

✔ Easy to implement

✔ Fits perfectly well to Pareto ranking and existing MOEAs

✔ Constraints can be of arbitrary nature

Section Outline

Summary

- Constraints represent limitations on the possible solutions: require special treatment too

Summary

- Constraints represent limitations on the possible solutions: require special treatment too
- They are different from objectives: Objectives put "always" optimization pressure, whereas constraints only put pressure as long as they are not satisfied.

谢谢

Thank you

Thomas Weise [汤卫思]
tweise@hfuu.edu.cn
http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Institute of Applied Optimization
Shushan District, Hefei, Anhui,
China

Caspar David Friedrich, “Der Wanderer über dem Nebelmeer”, 1818
http://en.wikipedia.org/wiki/Wanderer_above_the_Sea_of_Fog

Bibliography

Bibliography I

1. Katharine Jane Shaw and Peter J. Fleming. Including real-life problem preferences in genetic algorithms to improve optimisation of production schedules. In Ali M. S. Zalzala, editor, *Proceedings of the Second IEE/IEEE International Conference On Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'97)*, volume 1997 of *IET Conference Publications*, pages 239–244, Glasgow, Scotland, UK: University of Strathclyde, September 2–4, 1997. Stevenage, Herts, UK: Institution of Engineering and Technology (IET).
2. Katharine Jane Shaw. *Using Genetic Algorithms for Practical Multi-Objective Production Schedule Optimisation*. PhD thesis, Sheffield, UK: University of Sheffield, Department of Automatic Control and Systems Engineering, 1997.
3. Robin Charles Purshouse. *On the Evolutionary Optimisation of Many Objectives*. PhD thesis, Sheffield, UK: University of Sheffield, Department of Automatic Control and Systems Engineering, September 2003. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.8816`.
4. Gwoing Tina Yu and Peter John Bentley. Methods to evolve legal phenotypes. In Ágoston E. Eiben, Thomas Bäck, Marc Schoenauer, and Hans-Paul Schwefel, editors, *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature (PPSN V)*, volume 1498/1998 of *Lecture Notes in Computer Science (LNCS)*, pages 280–291, Amsterdam, The Netherlands, September 27–30, 1998. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/BFb0056843. URL `http://www.cs.ucl.ac.uk/staff/p.bentley/YUBEC2.pdf`.
5. Zbigniew Michalewicz. Genetic algorithms, numerical optimization, and constraints. In Larry J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA'95)*, pages 151–158., Pittsburgh, PA, USA: University of Pittsburgh, July 15–19, 1995. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. URL `http://www.cs.adelaide.edu.au/~zbyszek/Papers/p16.pdf`.
6. Zbigniew Michalewicz and Girish Nazhiyath. Genocop iii: A co-evolutionary algorithm for numerical optimization with nonlinear constraints. In *Second IEEE International Conference on Evolutionary Computation (CEC'95)*, volume 2, pages 647–651, Perth, WA, Australia: University of Western Australia, November 29–December 1, 1995. Los Alamitos, CA, USA: IEEE Computer Society Press. doi: 10.1109/ICEC.1995.487460. URL `http://www.cs.cinvestav.mx/~constraint/papers/p24.ps`.
7. Richard Courant. Variational methods for the solution of problems of equilibrium and vibrations. *Bulletin of the American Iris Society*, 49(1):1–23, 1943. doi: 10.1090/S0002-9904-1943-07818-4. URL `http://www.ams.org/bull/1943-49-01/S0002-9904-1943-07818-4/home.html`.
8. Anthony V. Fiacco and Garth P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. New York, NY, USA: John Wiley & Sons Ltd. and Fort Belvoir, VA, USA: Defense Technical Information Center (DTIC), January 1969. ISBN 0471258105 and 978-0471258100. URL `http://books.google.de/books?id=Hjp7OwAACAAJ`.

Bibliography II

9. Alice E. Smith and David W. Coit. Penalty functions. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, Computational Intelligence Library, chapter C 5.2. New York, NY, USA: Oxford University Press, Inc., Dirac House, Temple Back, Bristol, UK: Institute of Physics Publishing Ltd. (IOP), and Boca Raton, FL, USA: CRC Press, Inc., January 1, 1997. URL `http://www.cs.cinvestav.mx/~constraint/papers/chapter.pdf`.
10. Charles W. Carroll. *An Operations Research Approach to the Economic Optimization of a Kraft Pulping Process*. PhD thesis, Appleton, WI, USA: Institute of Paper Chemistry, June 1959. URL `http://hdl.handle.net/1853/5853`.
11. Charles W. Carroll. The created response surface technique for optimizing nonlinear, restrained systems. *Operations Research (Oper. Res.)*, 9(2):169–184, March–April 1961. doi: 10.1287/opre.9.2.169.
12. Vineet Khare, Xin Yao, and Kalyanmoy Deb. Performance scaling of multi-objective evolutionary algorithms. In Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele, editors, *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO'03)*, volume 2632/2003 of *Lecture Notes in Computer Science (LNCS)*, pages 367–390, Faro, Portugal: University of the Algarve, April 8–11, 2003. Berlin, Germany: Springer-Verlag GmbH. doi: 10.1007/3-540-36970-8_27.
13. Vineet Khare. Performance scaling of multi-objective evolutionary algorithms. Master's thesis, Birmingham, UK: University of Birmingham, School of Computer Science, September 21, 2002. URL `http://www.lania.mx/~ccoello/EMOO/thesis_khare.pdf.gz`.
14. Thomas Weise, Michael Zapf, Raymond Chiong, and Antonio Jesús Nebro Urbaneja. Why is optimization difficult? In Raymond Chiong, editor, *Nature-Inspired Algorithms for Optimisation*, volume 193/2009 of *Studies in Computational Intelligence*, chapter 1, pages 1–50. Berlin/Heidelberg: Springer-Verlag, April 30, 2009. doi: 10.1007/978-3-642-00267-0_1.
15. Carlos Artemio Coello Coello, Gary B. Lamont, and David A. van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*, volume 5 of *Genetic Algorithms and Evolutionary Computation*. Boston, MA, USA: Springer US and Norwell, MA, USA: Kluwer Academic Publishers, 2nd edition, 2002. ISBN 0306467623, 0387332545, 978-0306467622, 978-0-387-33254-3, and 978-0-387-36797-2. doi: 10.1007/978-0-387-36797-2. URL `http://books.google.de/books?id=sgX_Cst_yTsC`.
16. Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos Artemio Coello Coello, and David Wolfe Corne, editors. *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO'01)*, volume 1993/2001 of *Lecture Notes in Computer Science (LNCS)*, Zürich, Switzerland: Eidgenössische Technische Hochschule (ETH) Zürich, March 7–9, 2001. Berlin, Germany: Springer-Verlag GmbH. ISBN 3-540-41745-1 and 978-3-540-41745-3. doi: 10.1007/3-540-44719-9. URL `http://books.google.de/books?id=d52UjQ5pfcIC`.

Bibliography III

17. Marco Farina and Paolo Amato. On the optimal solution definition for many-criteria optimization problems. In James M. Keller and Olfa Nasraoui, editors, *Proceedings of the Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS'00)*, pages 233–238, New Orleans, LA, USA: Tulane University, June 27–29, 2002. Piscataway, NJ, USA: IEEE Computer Society. doi: 10.1109/NAFIPS.2002.1018061. URL `http://www.lania.mx/~ccoello/farina02b.pdf.gz`.
18. Robin Charles Purshouse and Peter J. Fleming. Evolutionary many-objective optimisation: An exploratory analysis. In Ruhul Amin Sarker, Robert G. Reynolds, Hussein Aly Abbass Amein, Kay Chen Tan, Robert Ian McKay, Daryl Leslie Essam, and Tom Gedeon, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'03)*, volume 3, pages 2066–2073, Canberra, Australia, December 8–13, 2003. Piscataway, NJ, USA: IEEE Computer Society. doi: 10.1109/CEC.2003.1299927. URL `http://www.lania.mx/~ccoello/EMOO/purshouse03b.pdf.gz`.
19. Evan J. Hughes. Evolutionary many-objective optimisation: Many once or one many? In David Wolfe Corne, Zbigniew Michalewicz, Robert Ian McKay, Ágoston E. Eiben, David B. Fogel, Carlos M. Fonseca, Günther R. Raidl, Kay Chen Tan, and Ali M. S. Zalzala, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05)*, volume 1, pages 222–227, Edinburgh, Scotland, UK, September 2–5, 2005. Piscataway, NJ, USA: IEEE Computer Society. doi: 10.1109/CEC.2005.1554688. URL `http://www.lania.mx/~ccoello/hughes05.pdf.gz`.
20. Vladimir Zakian. New formulation for the method of inequalities. *Proceedings of the Institution of Electrical Engineers*, 126:579–584, 1979.
21. Vladimir Zakian. Perspectives on the principle of matching and the method of inequalities. Control Systems Centre Report 769, Manchester, UK: University of Manchester Institute of Science and Technology (UMIST), Control Systems Centre, 1992.
22. Vladimir Zakian and U. Al-Naib. Design of dynamical and control systems by the method of inequalities. *IEE Proceedings D: Control Theory – Control Theory and Applications*, 120(11):1421–1427, 1973.
23. Satoshi Sato, Tadashi Ishihara, and Hikaru Inooka. Systematic design via the method of inequalities. *IEEE Control Systems Magazine*, pages 57–65, October 1996. doi: 10.1109/37.537209.
24. James F. Whidborne, Da-Wei Gu, and Ian Postlethwaite. Algorithms for the method of inequalities – a comparative study. In *Proceedings of the 1995 American Control Conference (ACC)*, volume 5, pages 3393–3397, Seattle, WA, USA, June 21–23, 1995. Piscataway, NJ, USA: IEEE Computer Society. FA19 = 9:15.
25. P. Zhang and A.H. Coonick. Coordinated synthesis of pss parameters in multi-machine power systems using the method of inequalities applied to genetic algorithms. *IEEE Transactions on Power Systems*, 15(2):811–816, May 2000. doi: 10.1109/59.867178. URL `http://www.lania.mx/~ccoello/EMOO/zhang00.pdf.gz`.

Bibliography IV

26. Hartmut Pohlheim. *GEATbx Introduction – Evolutionary Algorithms: Overview, Methods and Operators*, version 3.7 edition, November 2005. URL http://www.geatbx.com/download/GEATbx_Intro_Algorithmen_v38.pdf. Documentation for GEATbx version 3.7 (Genetic and Evolutionary Algorithm Toolbox for use with Matlab).
27. Kalyanmoy Deb, Amrit Pratab, Samir Agrawal, and T Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation (IEEE-EC)*, 6(2):182–197, April 2002. doi: 10.1109/4235.996017. URL http://dynamics.org/~altenber/UH_ICS/EC_REFS/MULTI_OBJ/DebPratapAgarwalMeyarivan.pdf.
28. Min-Rong Chen, Yong-zai Lu, and Gen-ke Yang. Multiobjective extremal optimization with applications to engineering design. *Journal of Zhejiang University – Science A (JZUS-A)*, 8(12):1905–1911, November 2007. doi: 10.1631/jzus.2007.A1905.
29. V. Ling Huang, Ponnuthurai Nagaratnam Suganthan, Alex Kai Qin, and S. Baskar. Multiobjective differential evolution with external archive and harmonic distance-based diversity measure. Technical report, Singapore: Nanyang Technological University (NTU), 2006. URL http://www3.ntu.edu.sg/home/epnsugan/index_files/papers/Tech-Report-MODE-2005.pdf.