GECCO 2023 Tutorial: Large-Scale Optimization and Learning

Nabi Omidvar[1], Yuan Sun[2], Xiaodong Li[3]
[1] University of Leeds, United Kingdom
[2] La Trobe University, Australia
[3] RMIT University, Australia
M.N.Omidvar@leeds.ac.uk, Yuan.Sun@latrobe.edu.au,
xiaodong.li@rmit.edu.au

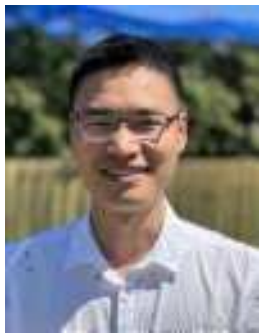
GECCO '23 Companion, July 15 - 19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0120-7/23/07.
doi:10.1145/3583133.3595037

Instructors

Nabi Omidvar is an Assistant Professor of Artificial Intelligence in Financial Services affiliated with Leeds University Business School and School of Computing, University of Leeds, U.K. He received the first bachelors (First Class Hons.) degree in computer science, the second bachelors degree in applied mathematics, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia. His current research interests include large-scale global optimization, high-dimensional machine learning, and AI for financial services.

Yuan Sun is a Lecturer in Business Analytics and Artificial Intelligence at La Trobe University, Australia. He received his BSc in Applied Mathematics from Peking University, China, and his PhD in Computer Science from The University of Melbourne, Australia. His research interest is on artificial intelligence, machine learning, operations research, and evolutionary computation.



Xiaodong Li is a Professor in Artificial Intelligence at the School of Computing Technologies, RMIT University, Australia. He received B.Sc. degree from Xidian University, Xi'an, China, and Ph.D. degree in Artificial Intelligence from University of Otago, Dunedin, New Zealand. His research interests include machine learning, evolutionary computation, multiobjective optimization, multimodal optimization (niching), swarm intelligence, data mining/analytics, deep learning, math-heuristic and hybrid methods for optimization.

Large-Scale Black-Box Continuous Optimization

Optimization

$$\min \quad f(\mathrm{x}),\ \mathrm{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n \tag{1}$$
$$s.t.: \quad \mathrm{g}(\mathrm{x}) \leq 0 \tag{2}$$
$$\mathrm{h}(\mathrm{x}) = 0 \tag{3}$$

Can be converted to unconstrained optimization using:

- Penalty method;
- Lagrangian;
- Augmented Lagrangian.

Our focus is unconstrained optimization. We must learn how to walk before we can run.

Large Scale Global Optimization (LSGO)

How large is large?

- The notion of large-scale is not fix.
- Changes over time.
- Differs from problem to problem.
- The dimension at which existing methods start to fail.

State-of-the-art (EC)

- Binary: $\approx$ 1 billion [*a*].
- Integer (linear): $\approx$ 1 billion [*b*], [*c*].
- Real: $\approx$ 1000-5000.

[*a*] Kumara Sastry et al. (2007). "Towards billion-bit optimization via a parallel estimation of distribution algorithm". In: *Genetic and Evolutionary Computation Conference*. ACM, pp. 577–584.

[*b*] Kalyanmoy Deb and Christie Myburgh (2016). "Breaking the Billion-Variable Barrier in Real-World Optimization Using a Customized Evolutionary Algorithm". In: *Genetic and Evolutionary Computation Conference*. ACM, pp. 653–660.

[*c*] Kalyanmoy Deb and Christie Myburgh (2017). "A population-based fast algorithm for a billion-dimensional resource allocation problem with integer variables". In: *European Journal of Operational Research* 261.2, pp. 460–474.

The Challenge of Large Scale Optimization

Why is it difficult?

- Exponential growth in the size of search space (curse of dimensionality).

Research Goal

- Improving search quality (get to the optimal point).
- Improving search efficiency (get there fast).

Large Scale Global Optimization: Evolutionary Approaches

1. Decomposition and Divide-and-Conquer
2. Hybridization, Memetic Algorithms, and Local Search
3. Sampling and Variation Operators
4. Approximation and Surrogate Modeling
5. Parallelization
6. Initialization

Variable Interaction, Linkage, Epistasis

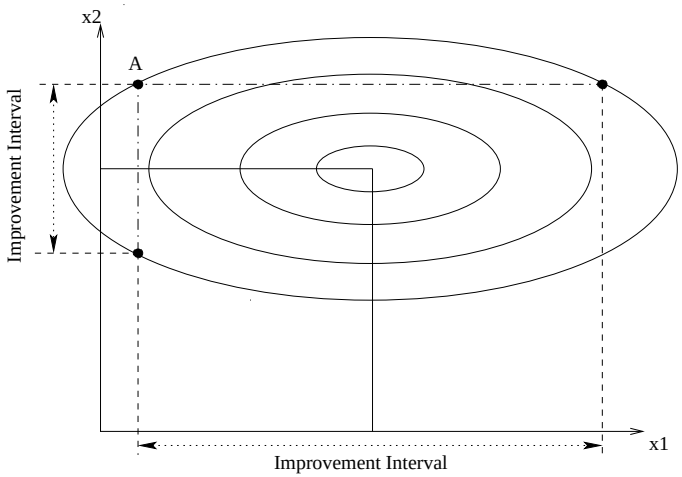
What is variable interaction?

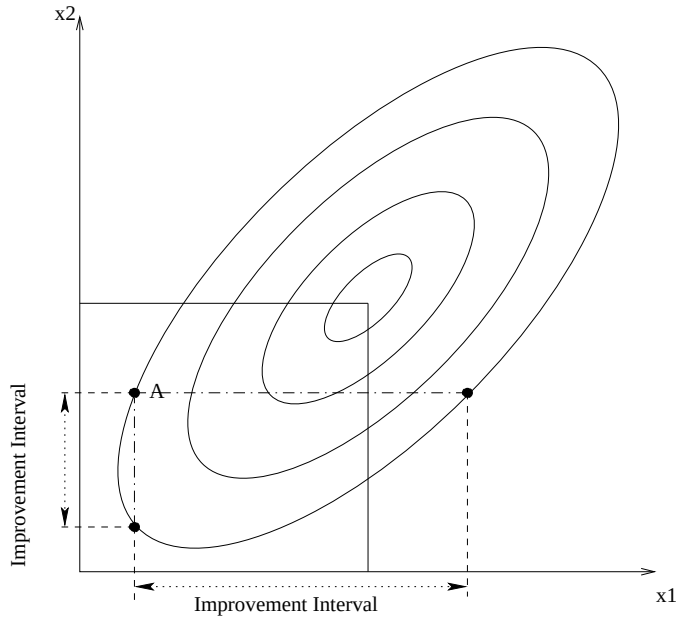
- Genetics: two genes are said to interact with each other if they collectively represent a feature at the phenotype level.
- The extent to which the fitness of one gene can be suppressed by another gene.
- The extent to which the value taken by one gene activates or deactivates the effect of another gene.

Variable Interaction, Linkage, Epistasis

Illustrative Example

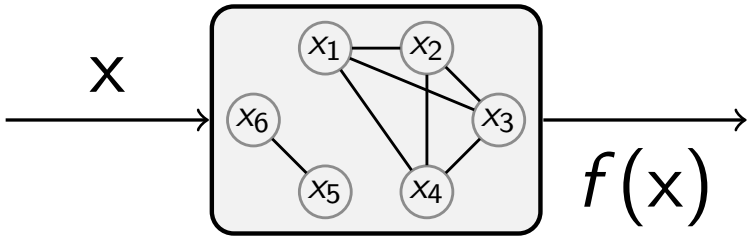
- $f(x_1, x_2) = x_1^2 + \lambda_1 x_2^2$
- $g(x_1, x_2) = x_1^2 + \lambda_1 x_2^2 + \lambda_2 x_1 x_2$





Problem Structure

(a)



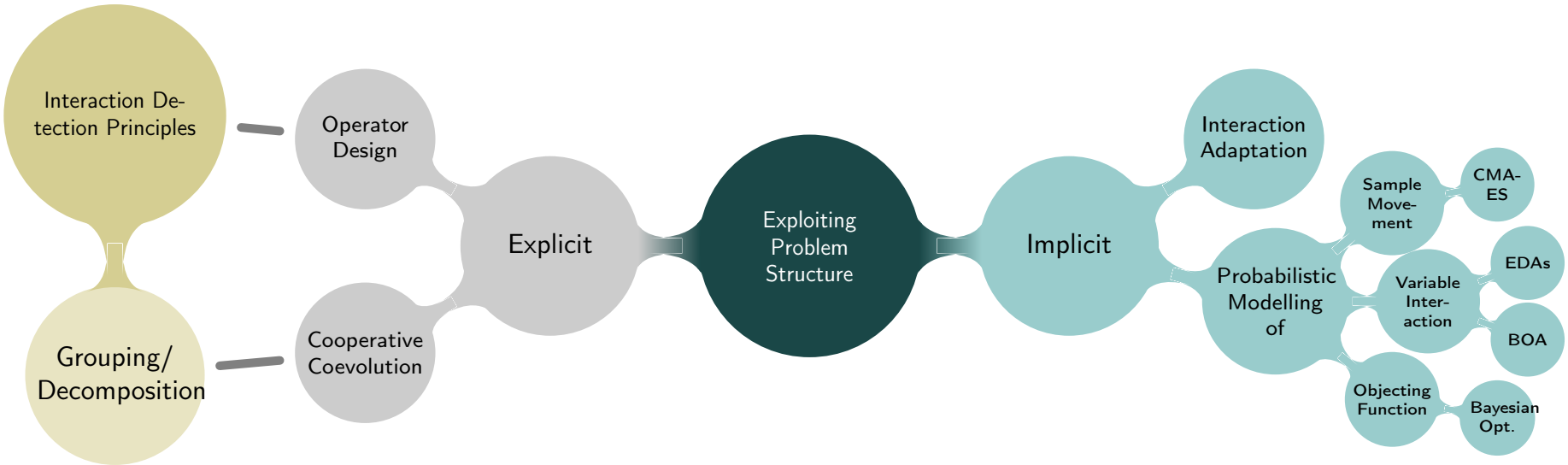
(b)

Effect of Variable Interaction

Why variable interaction?

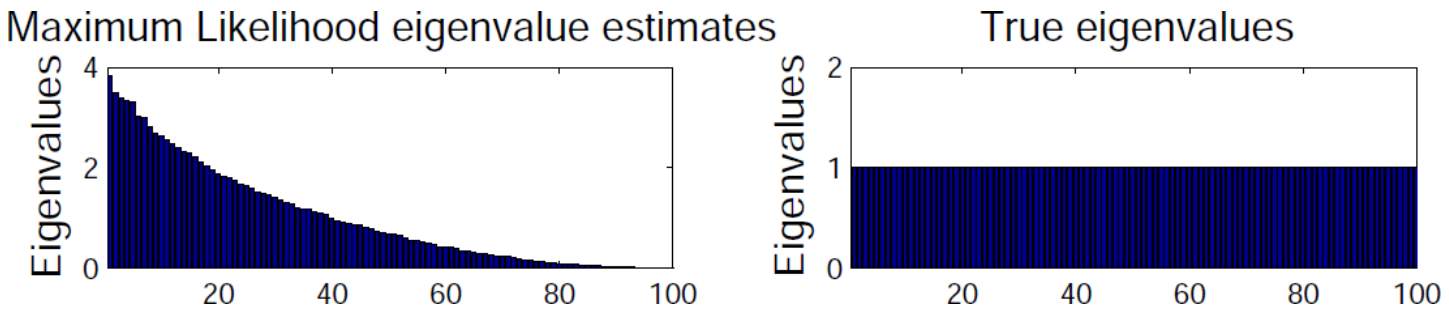
- The effectiveness of optimization algorithms is affected by how much they take variable interaction into account.
- Also applies to classic mathematical programming methods.
- In EC:
  - Problem decomposition / dimensionality reduction.
  - Rotational invariance.
  - Systematic local search in memetic algorithms.
  - More effective approximation and meta-modelling.

Linkage Learning and Exploiting Modularity



Scalability issues of EDAs

- Accurate estimation requires a large sample size which grows exponentially with the dimensionality of the problem [1].
- A small sample results in poor estimation of the eigenvalues [2].
- The cost of sampling from a multi-dimensional Gaussian distribution increases cubically with the problem size [3].



[1] Jerome Friedman et al. (2001). *The elements of statistical learning*. Vol. 1. Springer series in statistics Springer, Berlin.

[2] Roman Vershynin (2010). "Introduction to the non-asymptotic analysis of random matrices". In: *arXiv preprint arXiv:1011.3027*.

[3] Weishan Dong and Xin Yao (2008). "Unified eigen analysis on multivariate Gaussian based estimation of distribution algorithms". In: *Information Sciences* 178.15, pp. 3000–3023.

Implicit Methods

- Space partitioning / dimensionality reduction
  - Model Complexity Control [1].
  - Random Matrix Projection [2].
- Low-rank covariance estimation [3]
- Heavy-tail sampling [4].

---

[1] Weishan Dong, Tianshi Chen, et al. (2013). "Scaling up estimation of distribution algorithms for continuous optimization". In: *IEEE Transactions on Evolutionary Computation* 17.6, pp. 797–822.

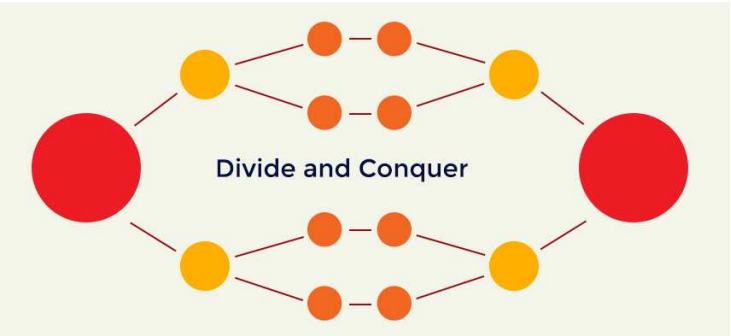
[2] Ata Kabán et al. (2016). "Toward large-scale continuous EDA: A random matrix theory perspective". In: *Evolutionary Computation* 24.2, pp. 255–291.

[3] Zhenhua Li and Qingfu Zhang (2017). "A simple yet efficient evolution strategy for large scale black-box optimization". In: *IEEE Transactions on Evolutionary Computation*; Ilya Loshchilov (2015). "LM-CMA: An Alternative to L-BFGS for Large-Scale Black Box Optimization". In: *Evolutionary Computation*.

[4] Momodou L Sanyang et al. (2016). "How effective is Cauchy-EDA in high dimensions?" In: *IEEE Congress on Evolutionary Computation*. IEEE, pp. 3409–3416.

Explicit Methods

- A large problem can be subdivided into smaller and simpler problems.
- Dates back to René Descartes (*Discourse on Method*).
- Has been widely used in many areas:
  - Computer Science: Sorting algorithms (quick sort, merge sort)
  - Optimization: Large-scale linear programs (Dantzig)



Decomposition in EAs: Cooperative Co-evolution

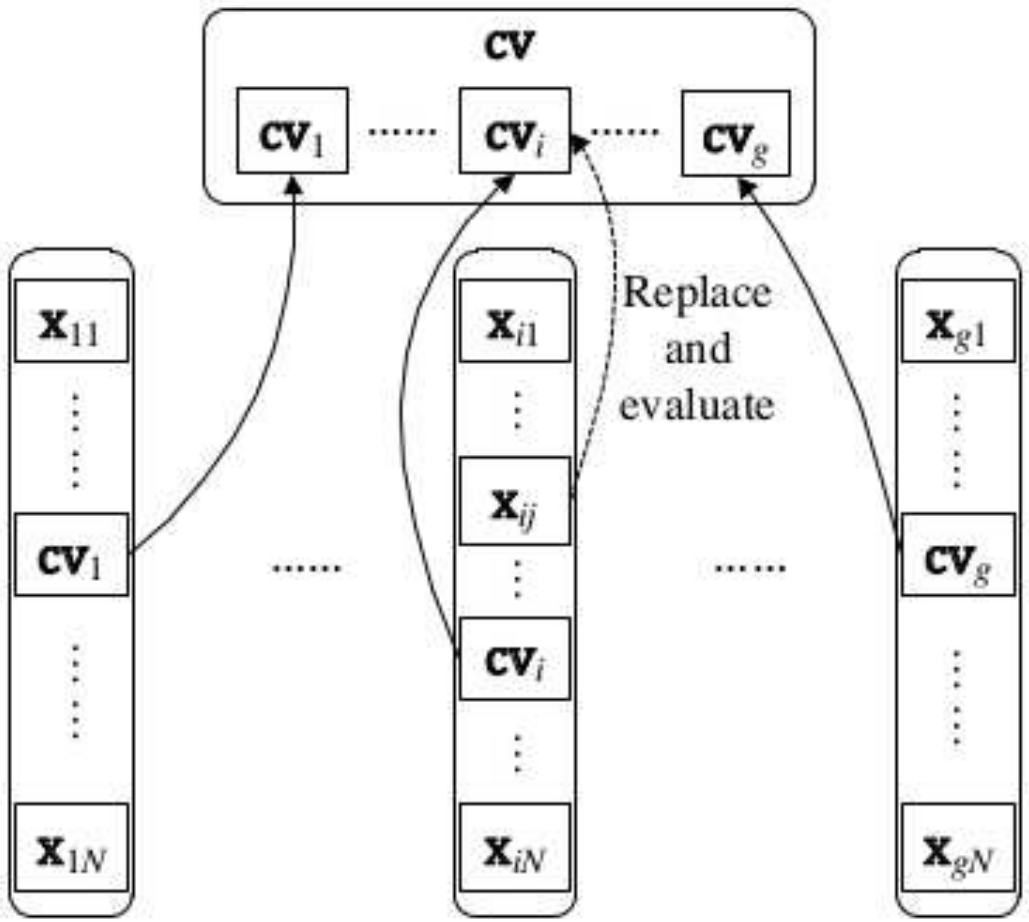


Figure: Cooperative Co-evolution [1]

[1] Mitchell A. Potter and Kenneth A. De Jong (1994). "A cooperative coevolutionary approach to function optimization". In: *Proc. Int. Conf. Parallel Problem Solving from Nature*. Vol. 2, pp. 249–257.

CC is a Framework

CC as a scalability agent:

- CC is not an optimizer.
- Requires a component optimizer.
- CC coordinates how the component optimizer is applied to components.
- A scalability agent.

Challenges of CC

Main Questions

1. How to decompose the problem?
2. How to allocated resources?
3. How to coordinate?

The Decomposition Challenge

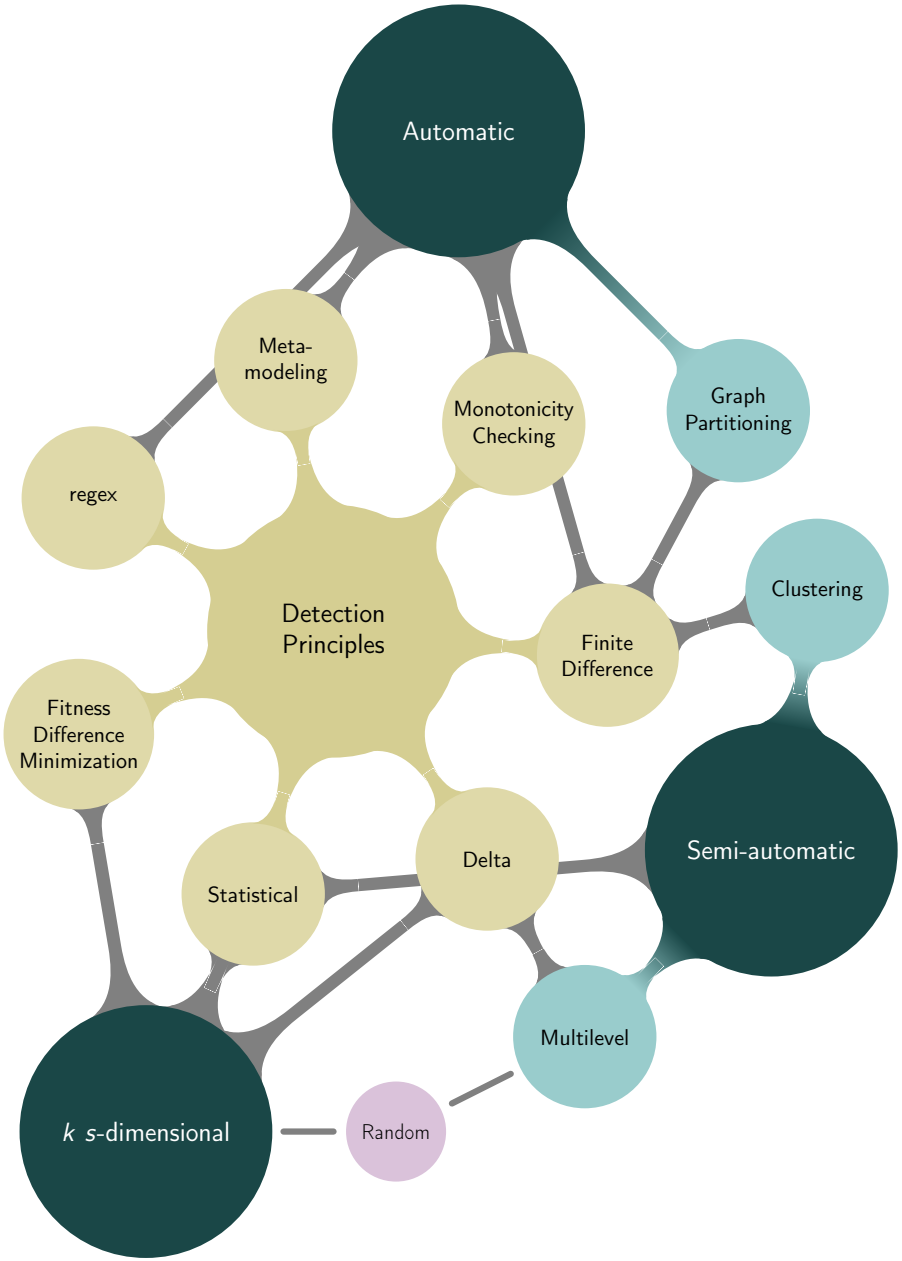
How to decompose?

- There are many possibilities.
- Which decomposition is the best?

Optimal decomposition

- It is governed by the interaction structure of decision variables.
- An optimal decomposition is the one that minimizes the interaction between components.

Overview of Decomposition Methods



Illustrative Example (Canonical CC)

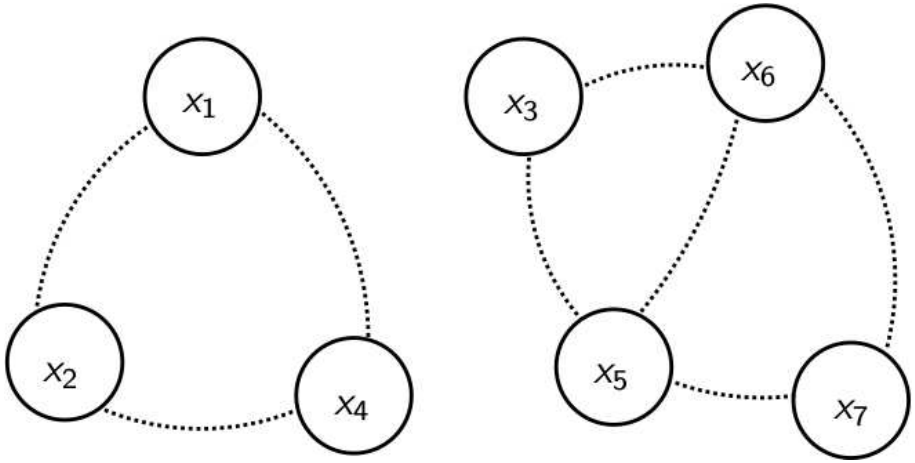


Figure: Variable interaction of a hypothetical function.

- n 1-dimensional components:
  - C_1: $\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}, \{x_6\}, \{x_7\}$
  - C_2: $\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}, \{x_6\}, \{x_7\}$
  - ...
  - C_c: $\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}, \{x_6\}, \{x_7\}$
- Not the best decomposition even for fully separable functions [1].

[1] Mohammad Nabi Omidvar, Yi Mei, et al. (2014). "Effective decomposition of large-scale separable continuous functions for cooperative co-evolutionary algorithms". In: *IEEE Congress on Evolutionary Computation*. IEEE, pp. 1305–1312.

Illustrative Example (Exact Methods)

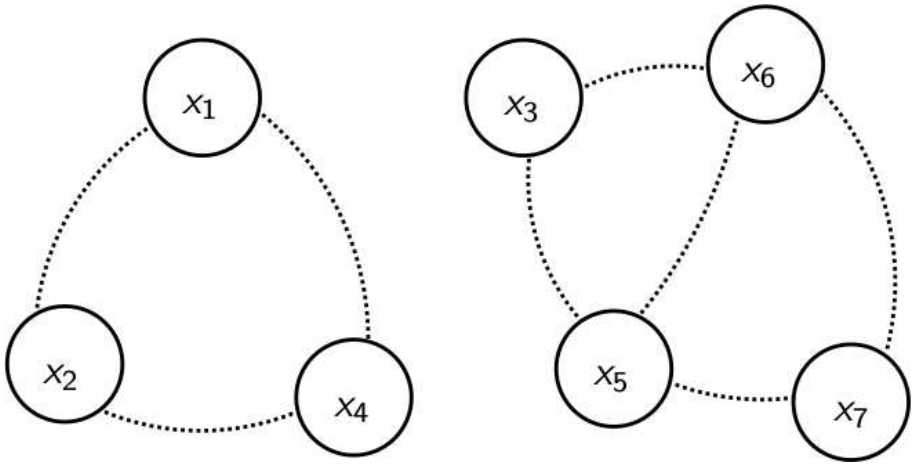
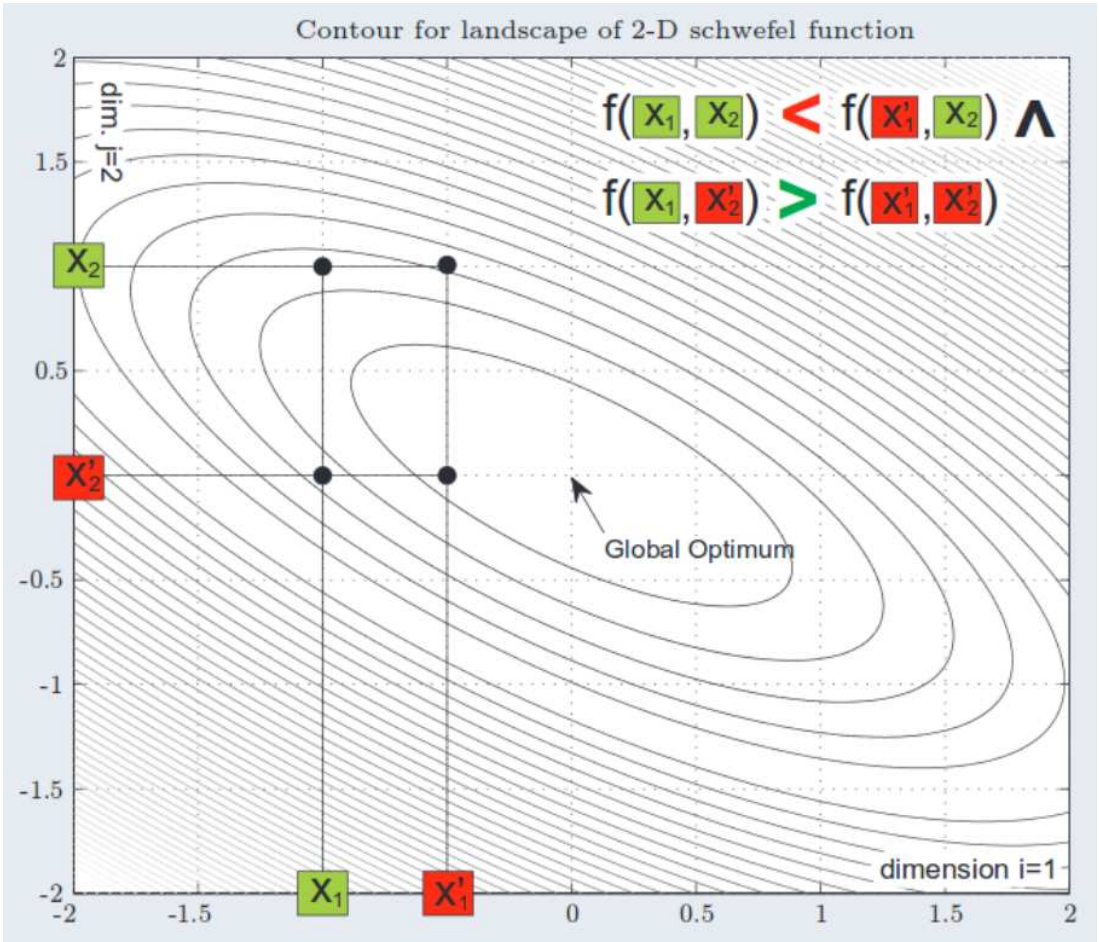


Figure: Variable interaction of a hypothetical function.

- Finite Difference Methods and Monotonicity Detection
  - C_1: $\{x_1, x_2, x_4\}, \{x_3, x_5, x_6, x_7\}$
  - C_2: $\{x_1, x_2, x_4\}, \{x_3, x_5, x_6, x_7\}$
  - ...
  - C_c: $\{x_1, x_2, x_4\}, \{x_3, x_5, x_6, x_7\}$

Monotonicity Check

$$\exists x, x_i', x_j' : f(x_1, ..., x_i, ..., x_j, ..., x_n) < f(x_1, ..., x_i', ..., x_j, ..., x_n) \wedge$$
$$f(x_1, ..., x_i, ..., x_j', ..., x_n) > f(x_1, ..., x_i', ..., x_j', ..., x_n)$$



Monotonicity Check (Algorithms)

- Linkage Identification by Non-Monotonicity Detection [1]
- Adaptive Coevolutionary Learning [2]
- Variable Interaction Learning [3]
- Variable Interdependence Learning [4]
- Fast Variable Interdependence [5]

---

[1] Masaharu Munetomo and David E Goldberg (1999). "Linkage identification by non-monotonicity detection for overlapping functions". In: *Evolutionary Computation* 7.4, pp. 377–398.

[2] Karsten Weicker and Nicole Weicker (1999). "On the improvement of coevolutionary optimizers by learning variable interdependencies". In: *IEEE Congress on Evolutionary Computation*. Vol. 3. IEEE, pp. 1627–1632.

[3] Wenxiang Chen et al. (2010). "Large-scale global optimization using cooperative coevolution with variable interaction learning". In: *Parallel Problem Solving from Nature*. Springer, pp. 300–309.

[4] Liang Sun et al. (2012). "A cooperative particle swarm optimizer with statistical variable interdependence learning". In: *Information Sciences* 186.1, pp. 20–39.

[5] Hongwei Ge et al. (2015). "Cooperative differential evolution with fast variable interdependence learning and cross-cluster mutation". In: *Applied Soft Computing* 36, pp. 300–314.

Differential Grouping

Theorem

Let $f(\mathrm{x})$ be an additively separable function.
$\forall a, b_1 \neq b_2, \delta \in \mathbb{R}, \delta \neq 0$, if the following condition holds

$$\Delta_{\delta,x_p}[f](\mathrm{x})|_{x_p=a,x_q=b_1} \neq \Delta_{\delta,x_p}[f](\mathrm{x})|_{x_p=a,x_q=b_2}, \tag{5}$$

then x_p and x_q are non-separable, where

$$\Delta_{\delta,x_p}[f](\mathrm{x}) = f(\ldots, x_p + \delta, \ldots) - f(\ldots, x_p, \ldots), \tag{6}$$

refers to the forward difference of f with respect to variable x_p with interval δ [a].

---

[a] Mohammad Nabi Omidvar, Xiaodong Li, Yi Mei, et al. (2014). "Cooperative co-evolution with differential grouping for large scale optimization". In: *IEEE Transactions on Evolutionary Computation* 18.3, pp. 378–393.

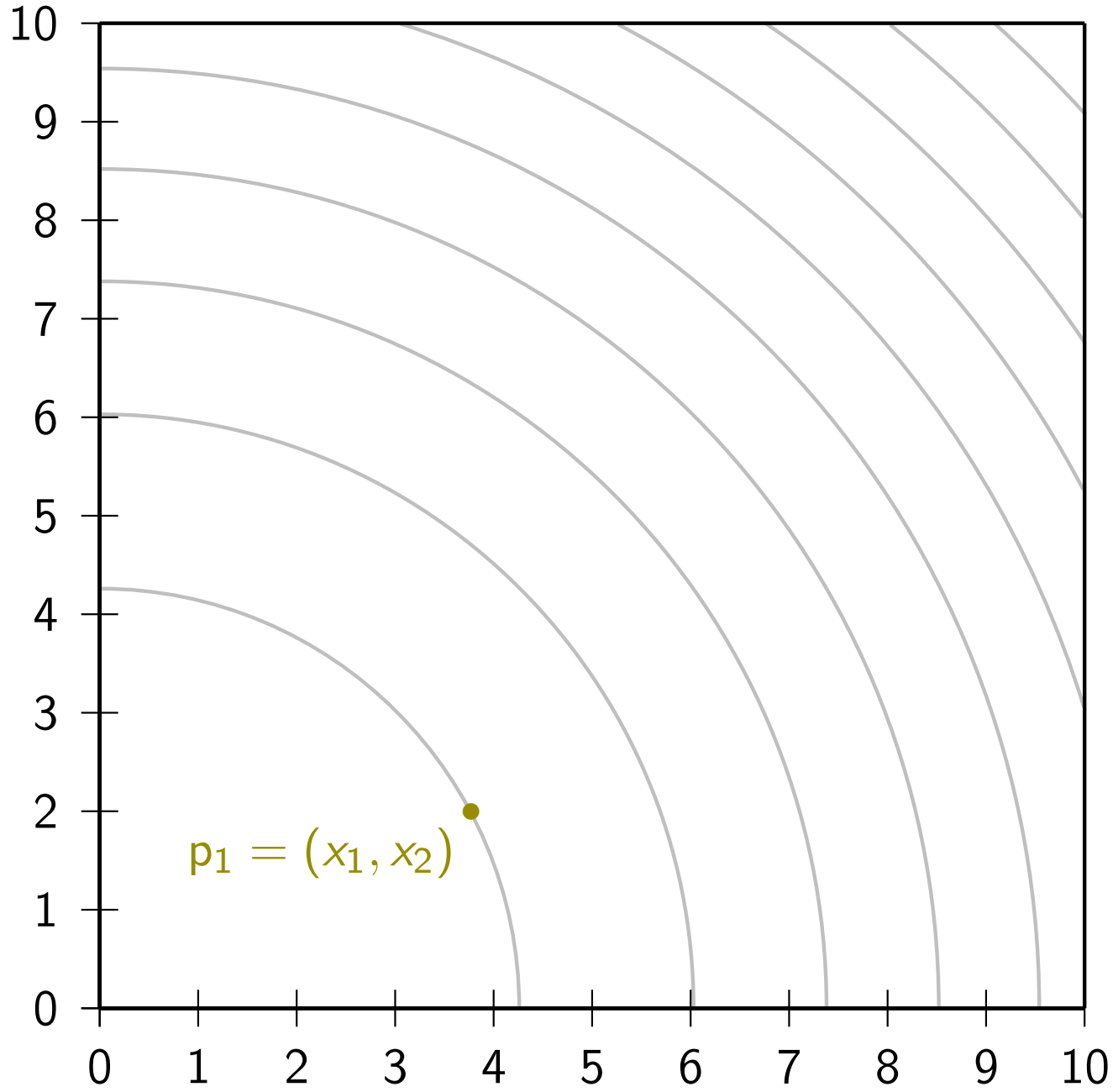


Figure: $f(x_1, x_2) = x_1^2 + x_2^2$

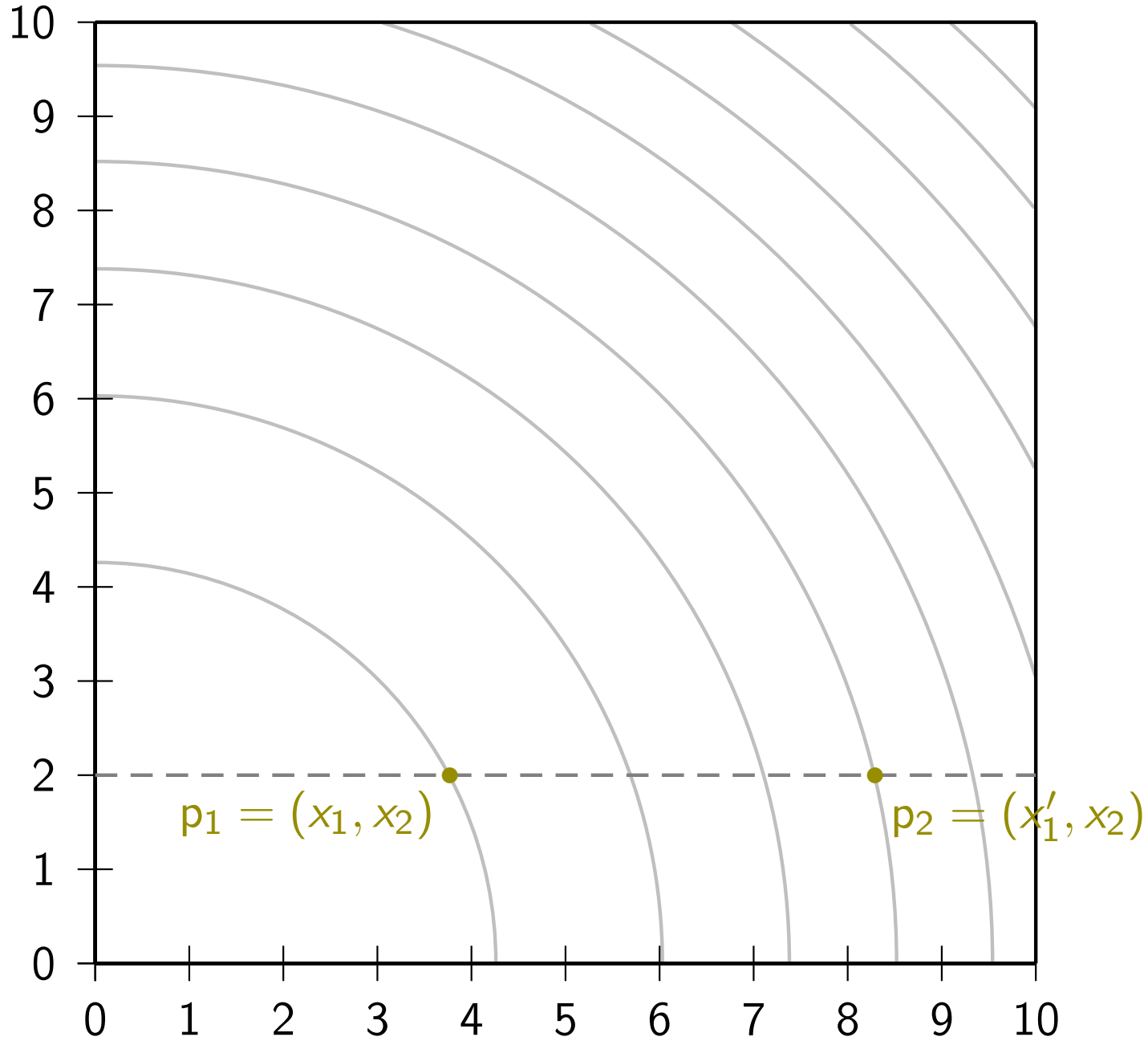


Figure: $f(x_1, x_2) = x_1^2 + x_2^2$

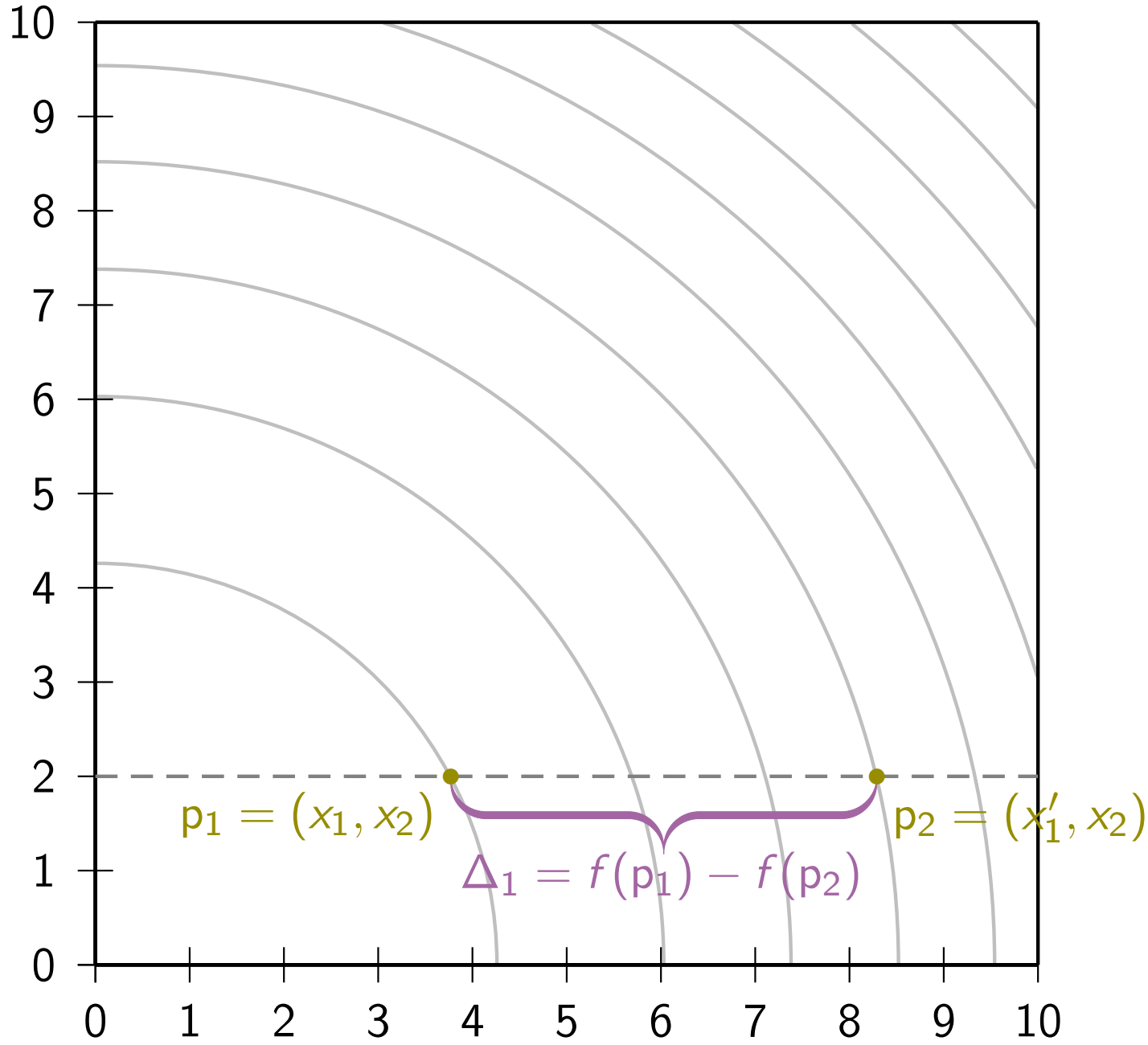


Figure: $f(x_1, x_2) = x_1^2 + x_2^2$

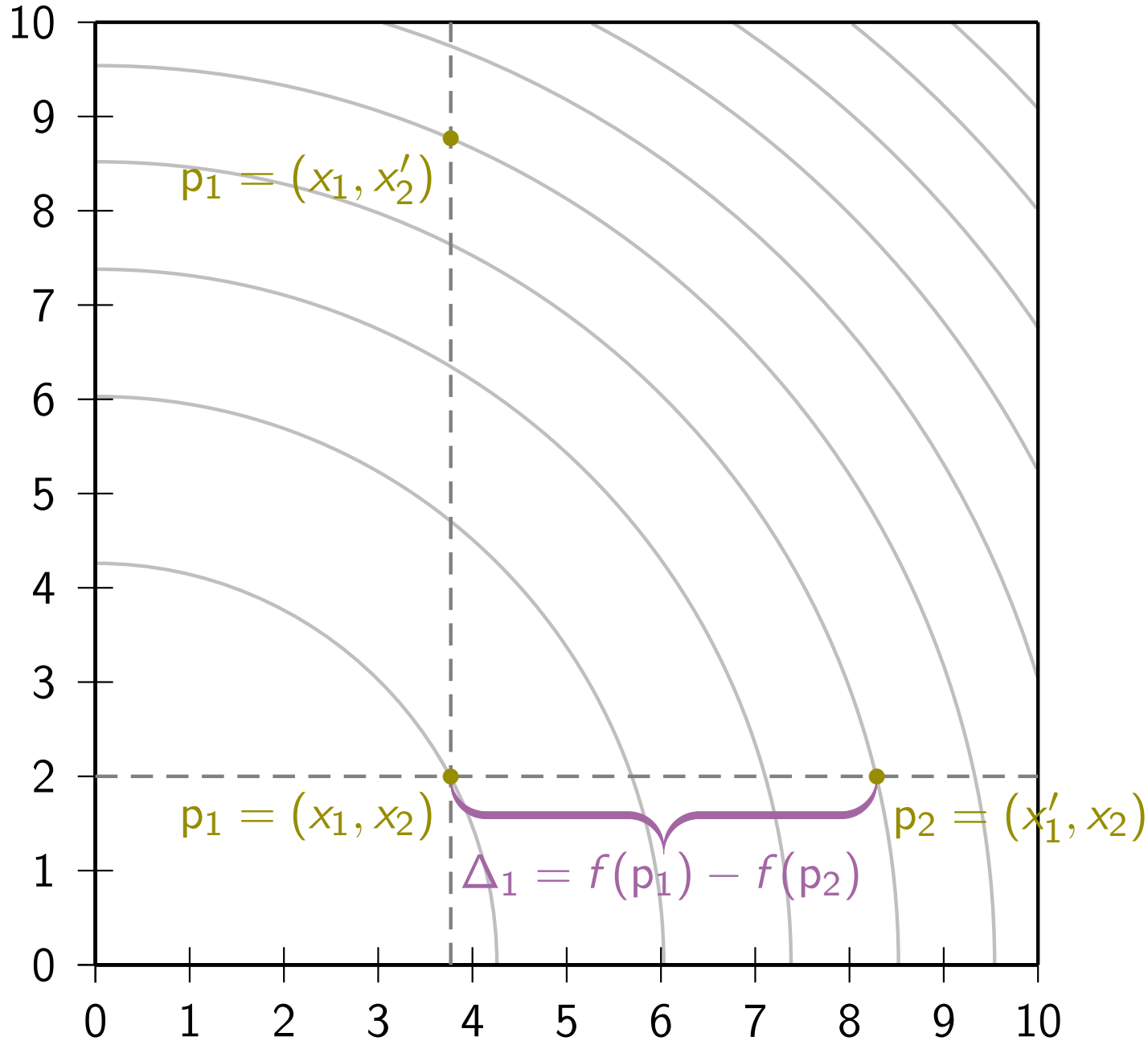


Figure: $f(x_1, x_2) = x_1^2 + x_2^2$

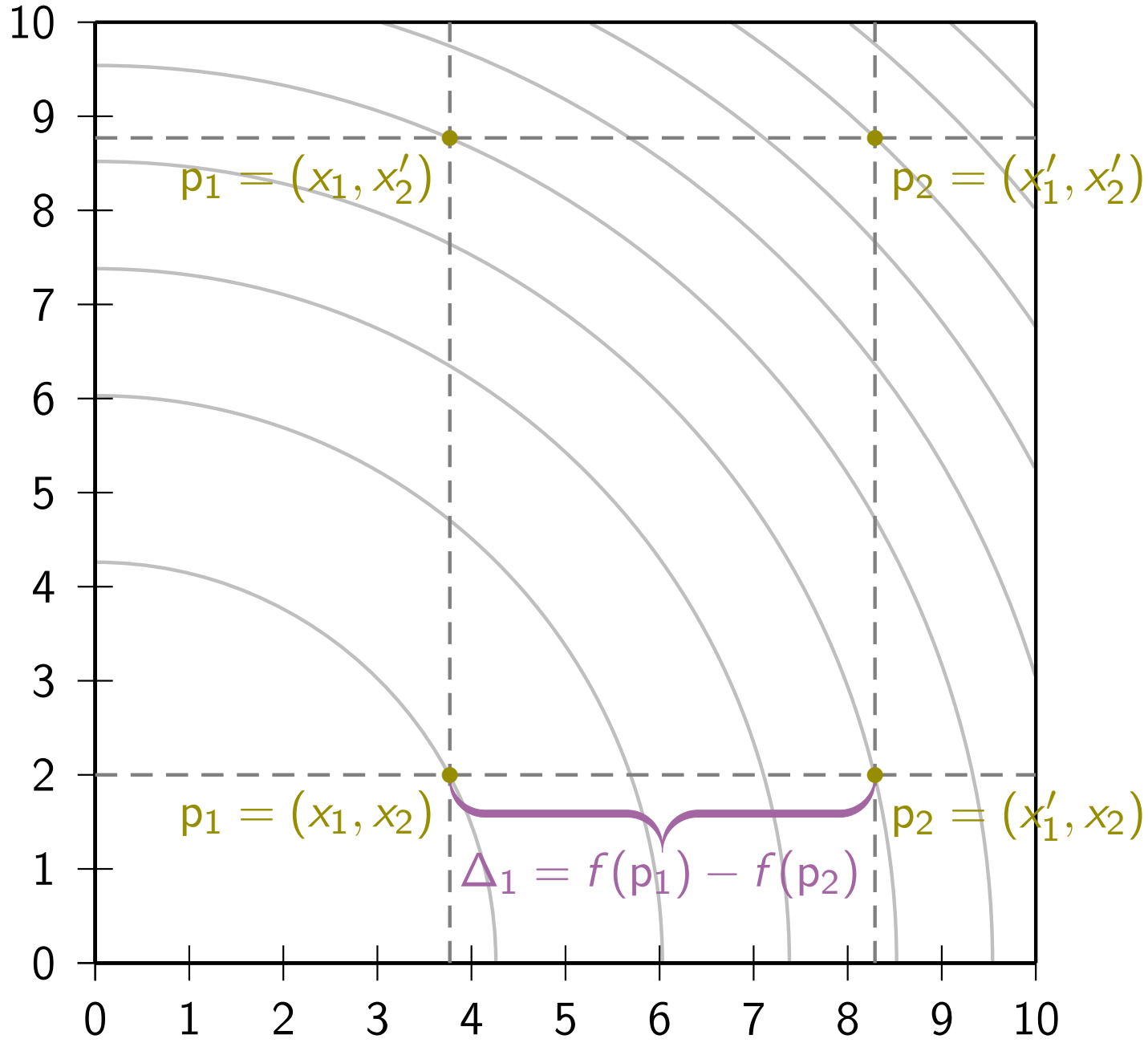


Figure: $f(x_1, x_2) = x_1^2 + x_2^2$

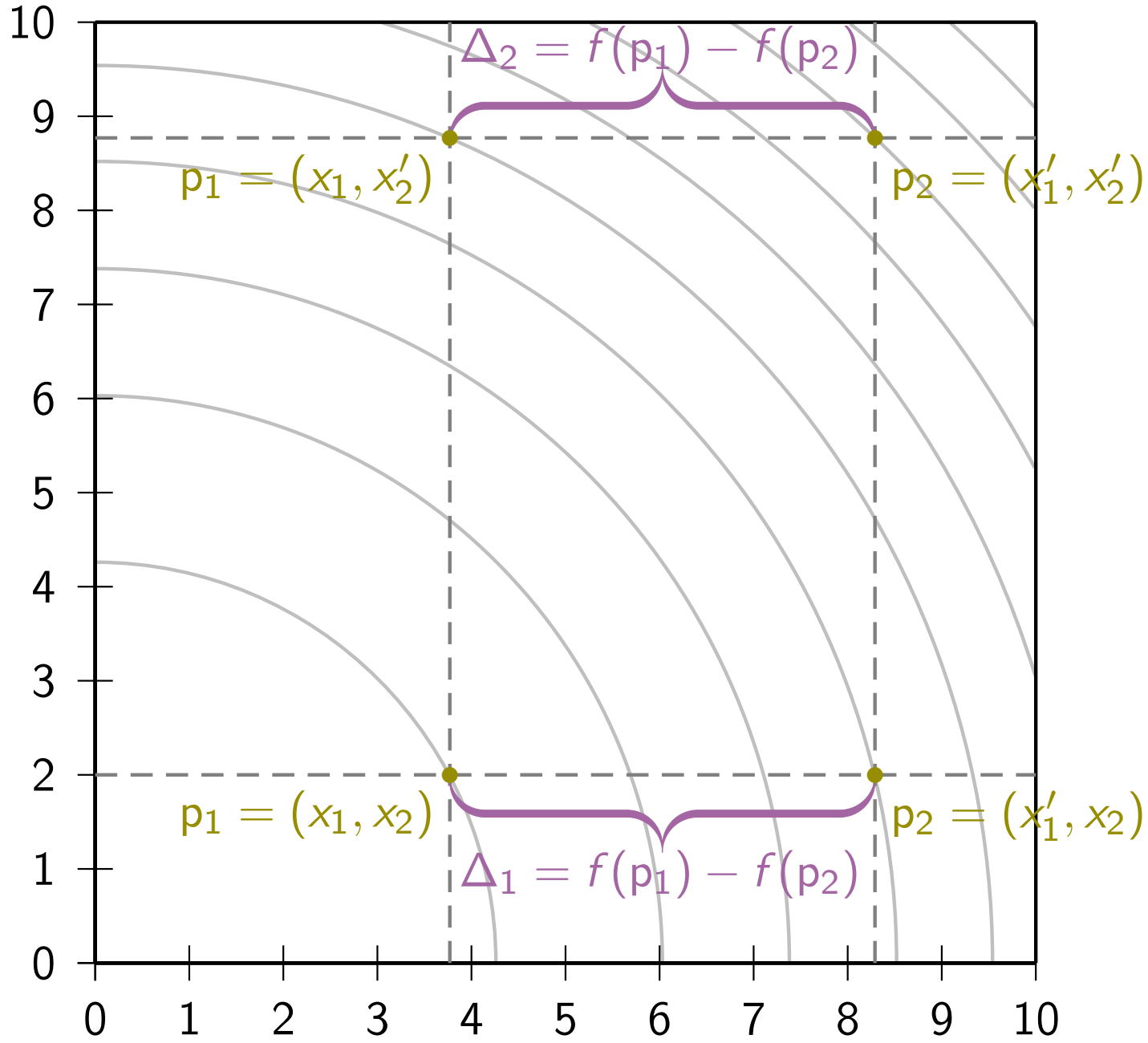


Figure: $f(x_1, x_2) = x_1^2 + x_2^2$

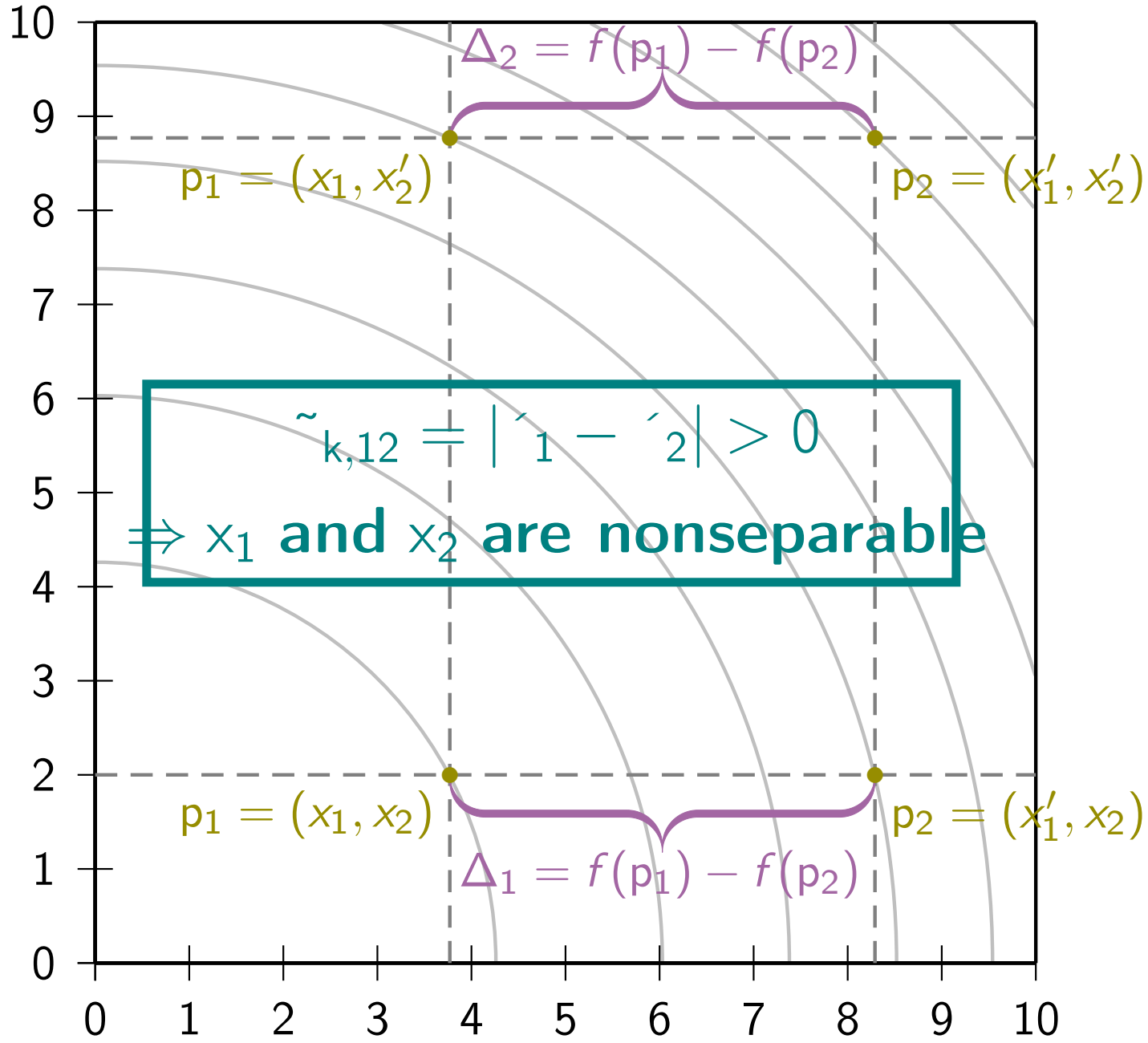


Figure: $f(x_1, x_2) = x_1^2 + x_2^2$

Separability $\Rightarrow \Delta_1 = \Delta_2$

Assuming:

$$f(\mathrm{x}) = \sum_{i=1}^{m} f_i(\mathrm{x}_i)$$

We prove that:

$$\text{Separability} \Rightarrow \Delta_1 = \Delta_2$$

By contraposition ($P \Rightarrow Q \equiv \neg Q \Rightarrow \neg P$):

$$\Delta_1 \neq \Delta_2 \Rightarrow \text{ non-separability}$$

or

$$|\Delta_1 - \Delta_2| > \epsilon \Rightarrow \text{ non-separability}$$

The Differential Grouping Algorithm

Detecting Non-separable Variables

$$|\Delta_1 - \Delta_2| > \epsilon \Rightarrow \text{ non-separability}$$

Detecting Separable Variables

$$|\Delta_1 - \Delta_2| \leq \epsilon \Rightarrow \text{ Separability (more plausible)}$$

Deductive Reasoning

Strong Syllogism

- $A \Rightarrow B$
  A is true
  $\therefore B$ is true
- $A \Rightarrow B$
  B is false
  $\therefore A$ is false

Weak Syllogism

- $A \Rightarrow B$
  A is false
  $\therefore B$ is less plausible
- $A \Rightarrow B$
  B is true
  $\therefore A$ is more plausible

Deductive Reasoning - Example

Strong Syllogism

- Rain $\Rightarrow$ Cloud
  It is rainy
  $\therefore$ It is cloudy

- Rain $\Rightarrow$ Cloud
  It is not cloudy
  $\therefore$ It is not rainy

Weak Syllogism

- Rain $\Rightarrow$ Cloud
  It is not rainy
  $\therefore$ Cloud becomes less likely

- Rain $\Rightarrow$ Cloud
  It is cloudy
  $\therefore$ Rain becomes more likely

Differential Grouping Family of Algorithms

- Linkage Identification by Non-linearity Check (LINC, LINC-R) [1]
- Differential Grouping (DG) [2]
- Global Differential Grouping (GDG) [3]
- Improved Differential Grouping (IDG) [4]

---

[1] Masaru Tezuka et al. (2004). "Linkage identification by nonlinearity check for real-coded genetic algorithms". In: *Genetic and Evolutionary Computation–GECCO 2004*. Springer, pp. 222–233.

[2] Mohammad Nabi Omidvar, Xiaodong Li, Yi Mei, et al. (2014). "Cooperative co-evolution with differential grouping for large scale optimization". In: *IEEE Transactions on Evolutionary Computation* 18.3, pp. 378–393.

[3] Yi Mei et al. (June 2015). "Competitive Divide-and-Conquer Algorithm for Unconstrained Large Scale Black-Box Optimization". In: *ACM Transaction on Mathematical Software* 42.2, p. 13.

[4] Mohammad Nabi Omidvar, Ming Yang, et al. (Sept. 2015). *IDG: A Faster and More Accurate Differential Grouping Algorithm*. Technical Report CSR-15-04. University of Birmingham, School of Computer Science.

Differential Grouping Family of Algorithms

- eXtended Differential Grouping (XDG) [1]
- Graph-based Differential Grouping (gDG) [2]
- Fast Interaction Identification [3]
- Recursive Differential Grouping (RDG1 and RDG2) [4]

---

[1] Yuan Sun, Michael Kirley, and Saman Kumara Halgamuge (2015). "Extended differential grouping for large scale global optimization with direct and indirect variable interactions". In: *Genetic and Evolutionary Computation Conference*. ACM, pp. 313–320.

[2] Yingbiao Ling et al. (2016). "Cooperative co-evolution with graph-based differential grouping for large scale global optimization". In: *International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*. IEEE, pp. 95–102.

[3] Xiao-Min Hu et al. (2017). "Cooperation coevolution with fast interdependency identification for large scale optimization". In: *Information Sciences* 381, pp. 142–160.

[4] Yuan Sun, Michael Kirley, and Saman K Halgamuge (2017). "A recursive decomposition method for large scale continuous optimization". In: *IEEE Transactions on Evolutionary Computation*; Yuan Sun, Mohammad Nabi Omidvar, et al. (2018). "Adaptive threshold parameter estimation with recursive differential grouping for problem decomposition". In: *a a* 5, p. 2.

Shortcomings of Differential Grouping

- Cannot detect the overlapping functions.
- Slow if all interactions are to be checked.
- Requires a threshold parameter (ϵ).
- Can be sensitive to the choice of the threshold parameter (ϵ).

Differential Grouping 2

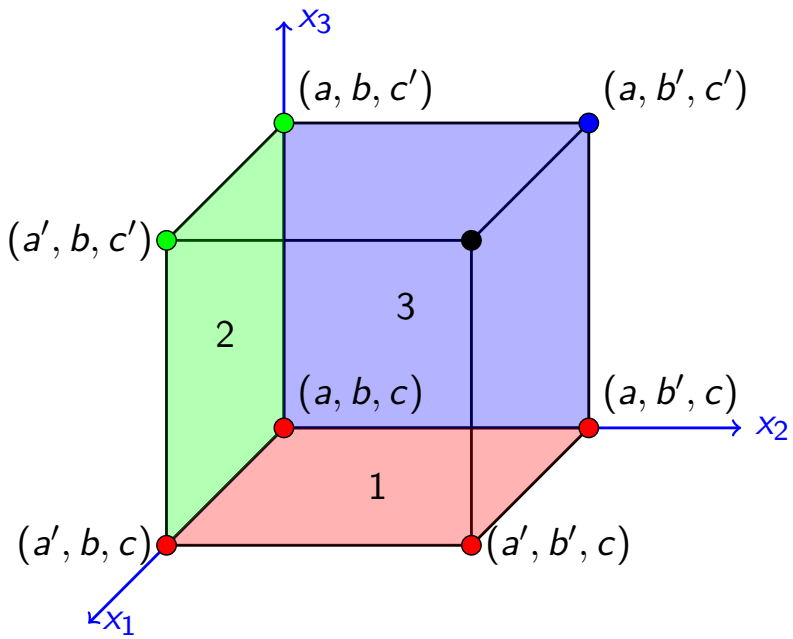


Figure: Geometric representation of point generation in DG2 for a 3D function.

$$x_1 \leftrightarrow x_2: \Delta^{(1)} = f(a', b, c) - f(a, b, c), \Delta^{(2)} = f(a', b', c) - f(a, b', c)$$

$$x_1 \leftrightarrow x_3: \Delta^{(1)} = f(a', b, c) - f(a, b, c), \Delta^{(2)} = f(a', b, c') - f(a, b, c')$$

$$x_2 \leftrightarrow x_3: \Delta^{(1)} = f(a, b', c) - f(a, b, c), \Delta^{(2)} = f(a, b', c') - f(a, b, c'),$$

$$\lambda = |\Delta^{(1)} - \Delta^{(2)}|$$

Differential Grouping 2

$$\tilde{} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{array} \\ \begin{pmatrix} 0 & 2 & 0 & 2 & 0 & 0 & 0 \\ 2 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 2 & 0 \\ 2 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 4 & 2 \\ 0 & 0 & 2 & 0 & 4 & 0 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 0 \end{pmatrix} \end{array}, \tag{7}$$

$$\hat{} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array}. \tag{8}$$

Improving the Efficiency

Minimum Evaluations

The minimum number of unique function evaluations in order to detect the interactions between all pairs of variables is

$$h(n) \geq \frac{n(n+1)}{2} + 1. \tag{9}$$

Improving efficiency beyond the given lower bound is impossible unless:

- Sacrifice on the accuracy (partial variable interaction matrix);
- and/or
- Extending the DG theorem.

Direct/Indirect Interactions

Indirect Interactions

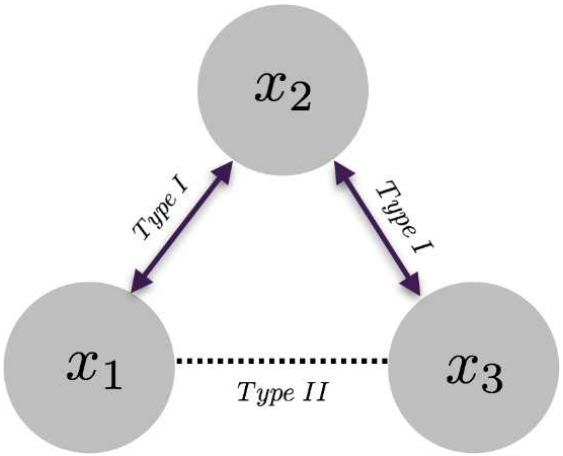
In an objective function $f(x)$, decision variables x_i and x_j interact directly (denoted by $x_i \leftrightarrow x_j$) if

$$\exists a : \left.\frac{\partial f}{\partial x_i \partial x_j}\right|_{x=a} \neq 0,$$

decision variables x_i and x_j interact *indirectly* if

$$\frac{\partial f}{\partial x_i \partial x_j} = 0,$$

and there exists a set of decision variables $\{x_{k1}, ..., x_{ks}\}$ such that $x_i \leftrightarrow x_{l1}, ..., x_{ks} \leftrightarrow x_j$.



Efficiency vs Accuracy

Saving budget at the expense of missing overlaps:

- eXtended Differential Grouping [1].
- Fast Interdependence Identification [2].

Figure: The interaction structures represented by the two graphs cannot be distinguished by XDG.

---

[1] Yuan Sun, Michael Kirley, and Saman Kumara Halgamuge (2015). "Extended differential grouping for large scale global optimization with direct and indirect variable interactions". In: *Genetic and Evolutionary Computation Conference*. ACM, pp. 313–320.

[2] Xiao-Min Hu et al. (2017). "Cooperation coevolution with fast interdependency identification for large scale optimization". In: *Information Sciences* 381, pp. 142–160.

Extending the Theorem

Theorem

Let $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ *be an objective function;* $X_1 \subset X$ *and* $X_2 \subset X$ *be two mutually exclusive subsets of decision variables:* $X_1 \cap X_2 = \emptyset$. *If there exist two unit vectors* $\mathrm{u}_1 \in U_{X_1}$ *and* $\mathrm{u}_2 \in U_{X_2}$, *two real numbers* $l_1, l_2 > 0$, *and a candidate solution* x^* *in the decision space, such that*

$$f(\mathrm{x}^* + l_1\mathrm{u}_1 + l_2\mathrm{u}_2) - f(\mathrm{x}^* + l_2\mathrm{u}_2) \neq f(\mathrm{x}^* + l_1\mathrm{u}_1) - f(\mathrm{x}^*), \qquad (10)$$

there is some interaction between decision variables in X_1 *and* X_2.

Some Algorithms

- Recursive Differential Grouping (RDG) [*a*]
- Fast Interaction Identification (FII) [*b*]

[*a*] Yuan Sun, Michael Kirley, and Saman K Halgamuge (2017). "A recursive decomposition method for large scale continuous optimization". In: *IEEE Transactions on Evolutionary Computation*.

[*b*] Xiao-Min Hu et al. (2017). "Cooperation coevolution with fast interdependency identification for large scale optimization". In: *Information Sciences* 381, pp. 142–160.

DG2: Accuracy

$$n = \pm s \times \beta^{e-p},$$

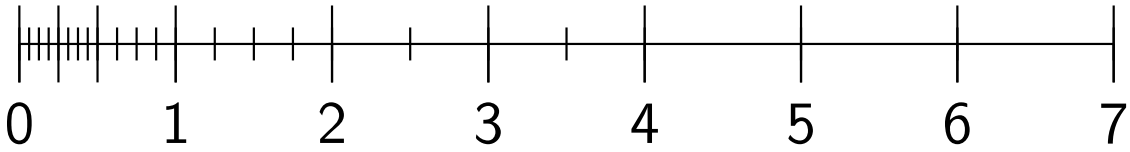


Figure: Non-uniform distribution of floating-point numbers for a hypothetical system ($\beta = 2, e_{\min} = -1, e_{\max} = 3$, and $p = 3$). The vertical bars denote all the representable numbers in this system.

Theorem

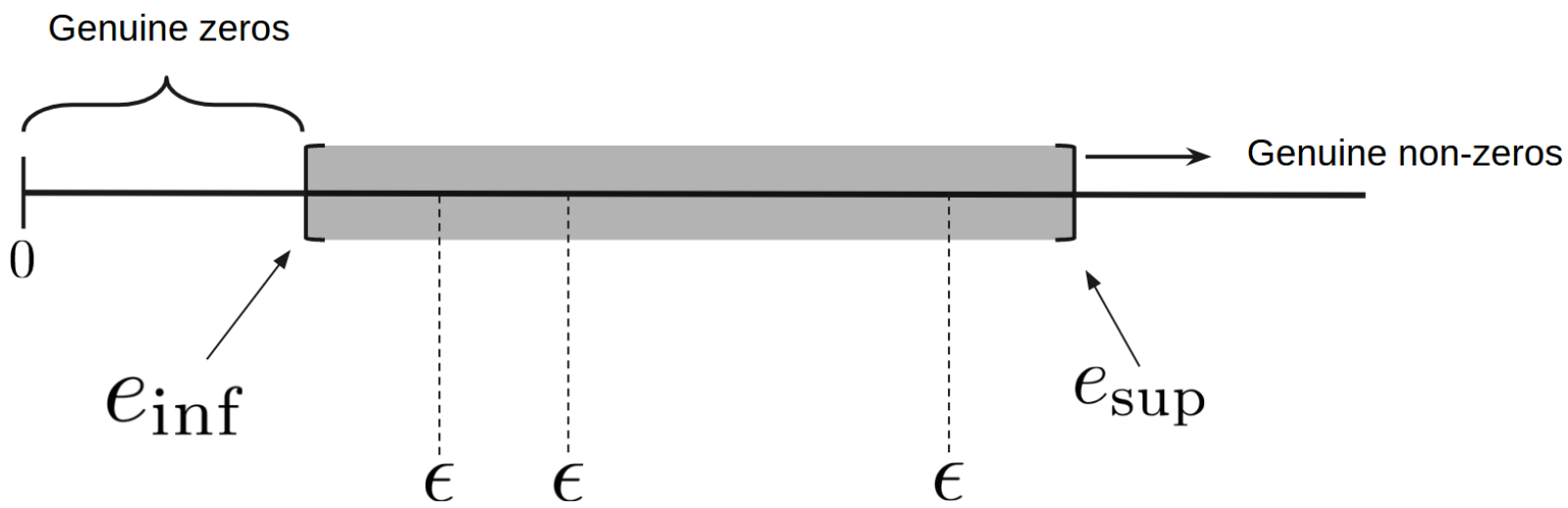
If $x \in \mathbb{R}$ lies in the range of $\mathbb{F}$, then

$$fl(x) = x(1 + \delta), \quad |\delta| < \mu_{\mathrm{M}},$$

where μ_{M} is called the unit roundoff, which is equal to $\frac{1}{2}\beta^{1-p}$.

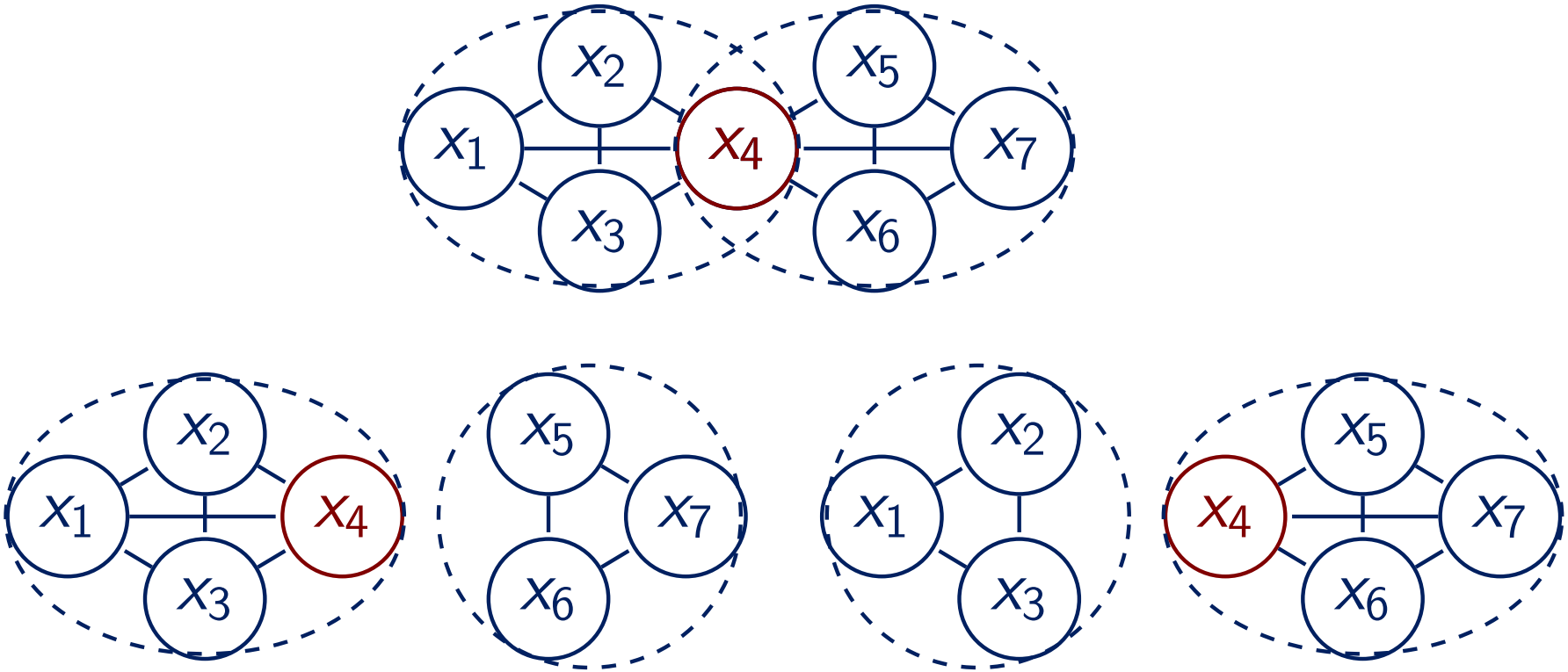
Improving Detection Accuracy: The Idea

$$\lambda = |\Delta_1 - \Delta_2|$$



RDG2 and RDG3

- RDG2 [1]: Efficiency of RDG and accuracy of DG2.
- RDG3 [2]: Decomposition for overlapping problems.



[1] Yuan Sun, Mohammad Nabi Omidvar, et al. (2018). "Adaptive threshold parameter estimation with recursive differential grouping for problem decomposition". In: *a a* 5, p. 2.

[2] Yuan Sun, Xiaodong Li, Andreas Ernst, and Mohammad Nabi Omidvar (2019). "Decomposition for Large-scale Optimization Problems with Overlapping Components". In: *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE.

Popularity of variable interaction detection and decomposition algorithms

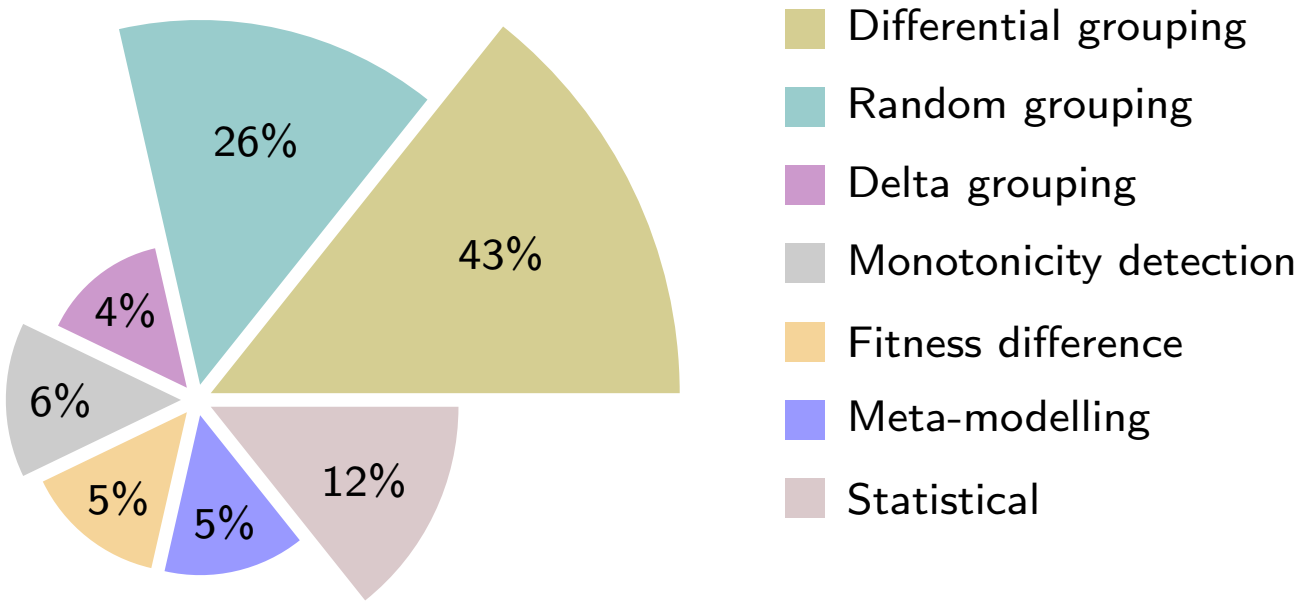
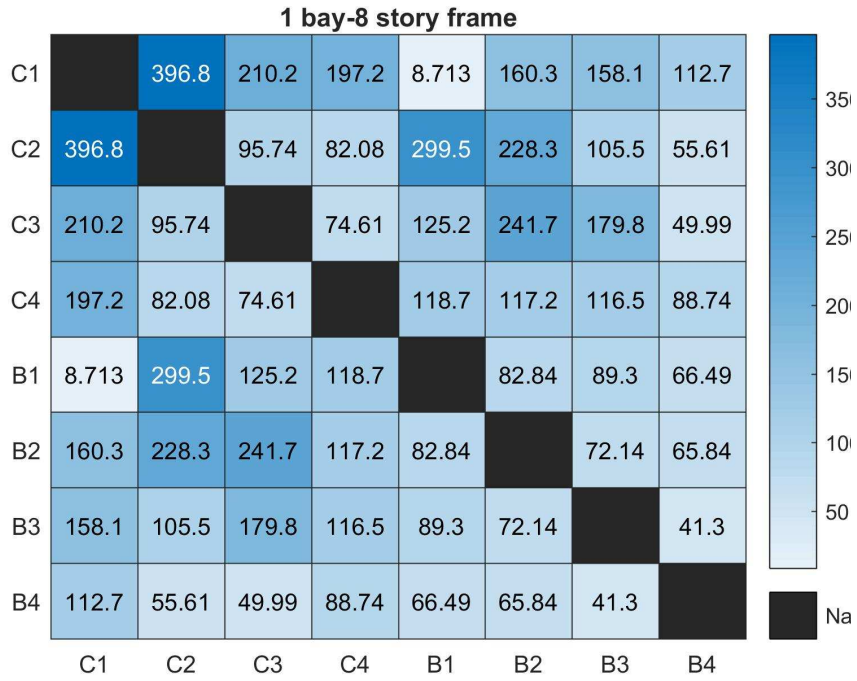
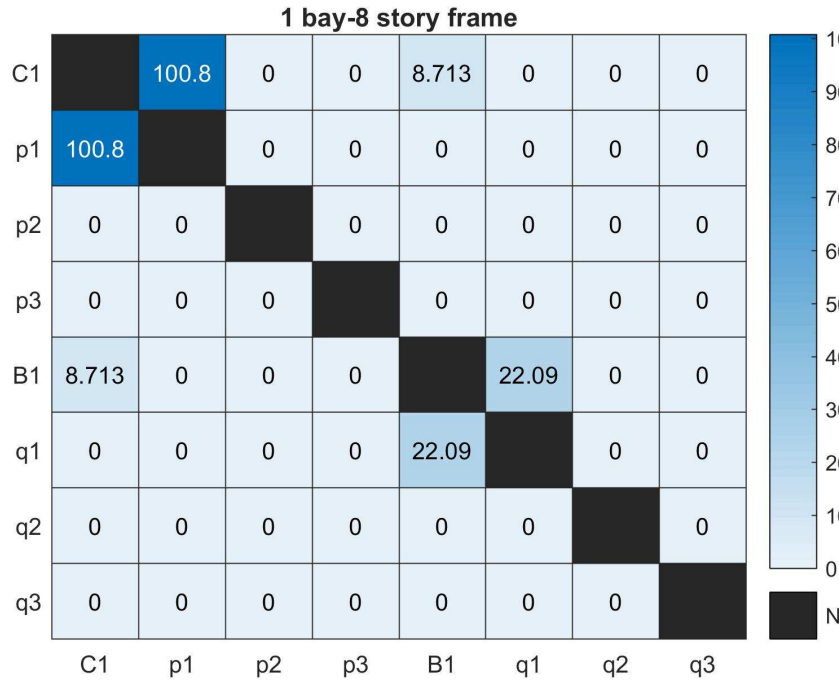


Figure: The most accurate (differential grouping) and the simplest (random grouping) are the most widely used.

DG as an Analysis Tool

Interaction matrix of two different formulations of the same structural engineering problem [1].

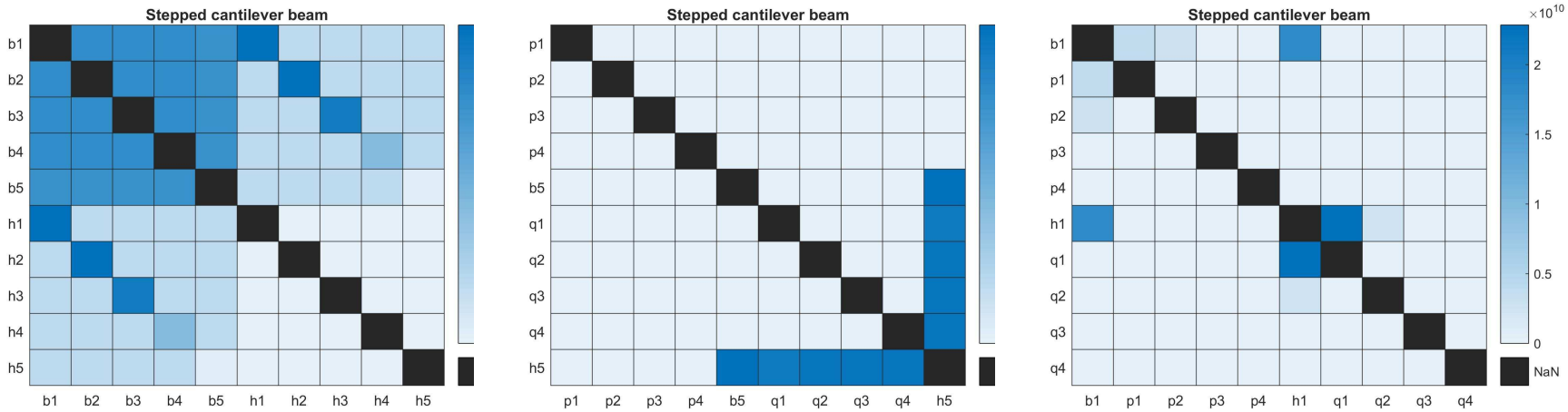




[1] Amir H Gandomi et al. (2019). "Using semi-independent variables to enhance optimization search". In: *Expert Systems with Applications* 120, pp. 279–297.

DG as an Analysis Tool

Interaction matrix of three different formulations of the same structural engineering problem [1].



[1] Amir H Gandomi et al. (2019). "Using semi-independent variables to enhance optimization search". In: *Expert Systems with Applications* 120, pp. 279–297.

Some Auxiliary Topics

- Variable Interaction and Constraint Handling [1], [2], [3]
- Large-Scale Multiobjective Optimization

---

[1] Eman Sayed, Daryl Essam, Ruhul Sarker, and Saber Elsayed (2015). "Decomposition-based evolutionary algorithm for large scale constrained problems". In: *Information Sciences* 316, pp. 457–486.

[2] Adan E Aguilar-Justo and Efrén Mezura-Montes (2016). "Towards an improvement of variable interaction identification for large-scale constrained problems". In: *IEEE Congress on Evolutionary Computation*. IEEE, pp. 4167–4174.

[3] Julien Blanchard et al. (2017). "A cooperative co-evolutionary algorithm for solving large-scale constrained problems with interaction detection". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, pp. 697–704.

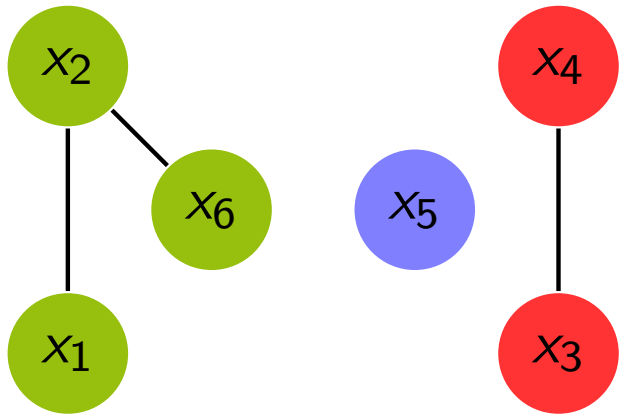
Variable Interaction and Constraint Handling

Idea: analyze the objective **and** the contraints [1].

$$
\begin{aligned}
\min\ f(x) &= x_1^2 x_2 + 4x_5 \\
s.t\ g_1(x) &= \frac{x_3}{x_4^2} + \sqrt{x_5} - x_6 \leq 0 \\
g_2(x) &= x_1 - x_2 e^{-x_6} \leq 0
\end{aligned}
$$

$$
\Theta_0 = \begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}, \Theta_1 = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}, \Theta_2 = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0
\end{pmatrix},
$$

$$
\Theta_{glob} = \begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0
\end{pmatrix}.
$$



[1] Julien Blanchard et al. (2017). "A cooperative co-evolutionary algorithm for solving large-scale constrained problems with interaction detection". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, pp. 697–704.

Large-Scale Multiobjective Optimization

Large-scale multiobjective optimization is growing popularity:

- Benchmark development and analysis:
  - Development of a benchmark [1].
  - Analysis of the existing benchmarks [2].
- Algorithm development:
  - Exploiting modularity using CC [3], [4], [5], [6].
  - Problem transformation [7].

---

[1] Ran Cheng et al. (2016). "Test problems for large-scale multiobjective and many-objective optimization". In: *IEEE Transactions on Cybernetics*.

[2] Ke Li et al. (2016). "Variable Interaction in Multi-objective Optimization Problems". In: *Parallel Problem Solving from Nature*. Springer International Publishing, pp. 399–409.

[3] Luis Miguel Antonio and Carlos A Coello Coello (2013). "Use of cooperative coevolution for solving large scale multiobjective optimization problems". In: *IEEE Congress on Evolutionary Computation*. IEEE, pp. 2758–2765.

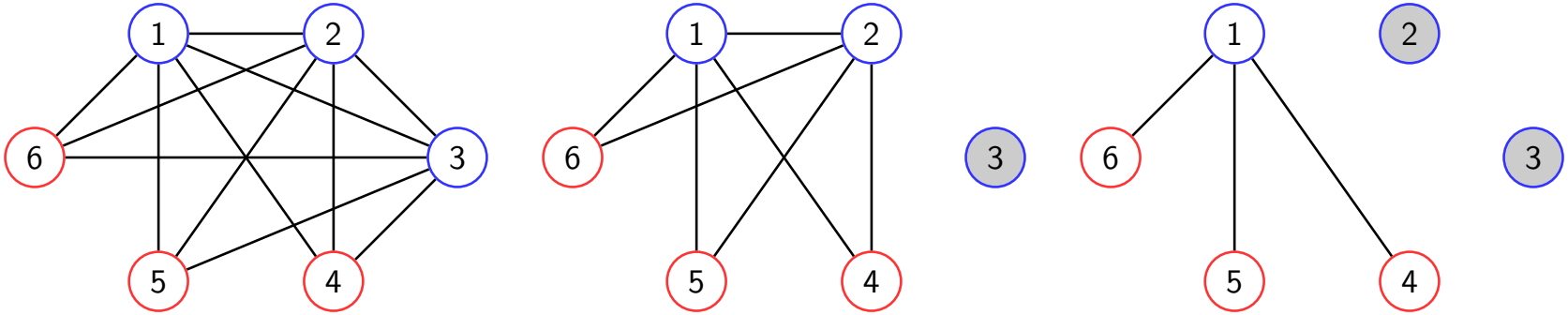
[4] Luis Miguel Antonio and Carlos A Coello Coello (2016). "Decomposition-Based Approach for Solving Large Scale Multi-objective Problems". In: *Parallel Problem Solving from Nature*. Springer, pp. 525–534.

[5] Xiaoliang Ma et al. (2016). "A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables". In: *IEEE Transactions on Evolutionary Computation* 20.2, pp. 275–298.

[6] Xingyi Zhang et al. (2016). "A Decision Variable Clustering-Based Evolutionary Algorithm for Large-scale Many-objective Optimization". In: *IEEE Transactions on Evolutionary Computation*.

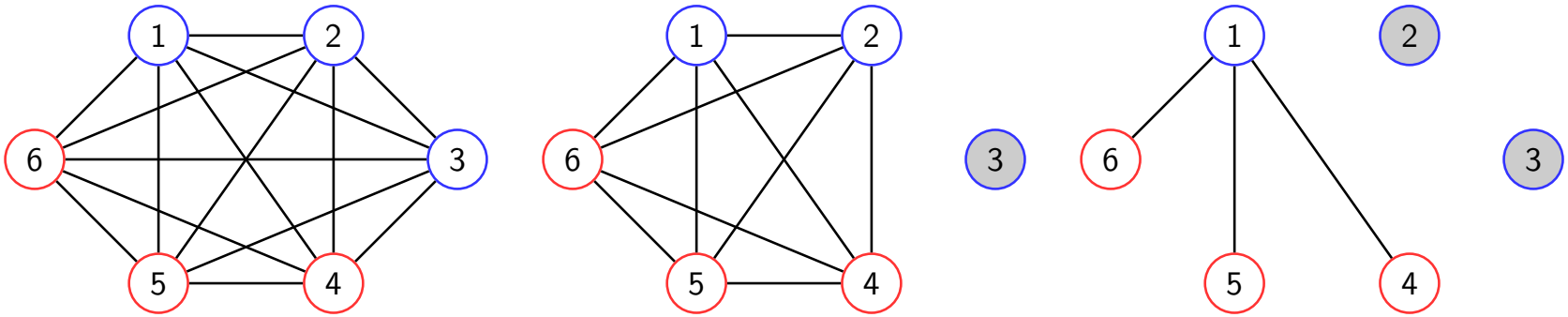
[7] Heiner Zille et al. (2018). "A Framework for Large-Scale Multiobjective Optimization Based on Problem Transformation". In: *IEEE Transactions on Evolutionary Computation* 22.2, pp. 260–275.

Analysis of DTLZ1-DTLZ4



(a) f_1 and f_2 (b) f_3 (c) f_4

Figure: Variable interaction graphs of DTLZ1 to DTLZ4 .



(a) f_1 and f_2 (b) f_3 (c) f_4

Figure: Variable interaction graphs of DTLZ5 and DTLZ6.

Decomposition Based Large-Scale EMO

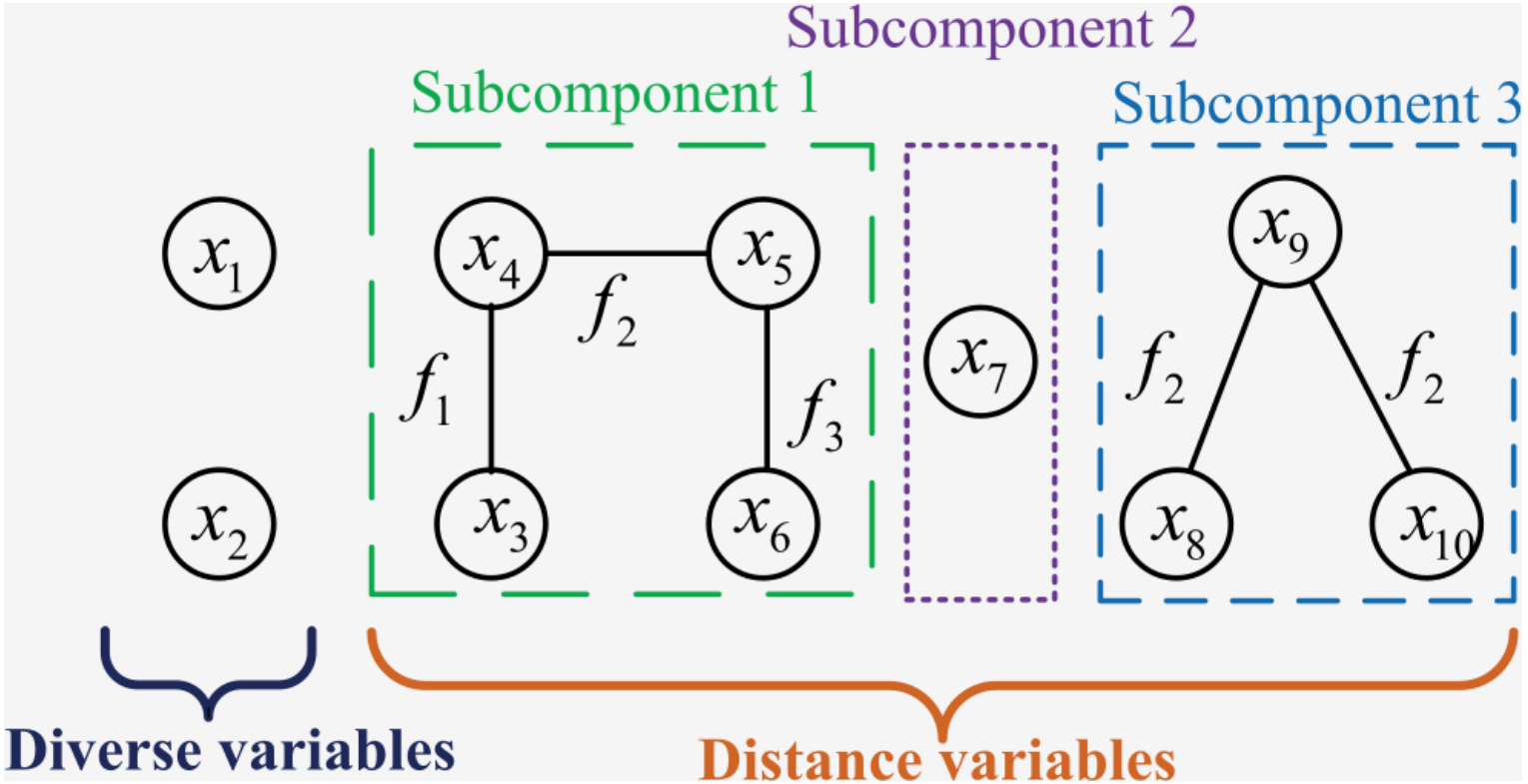


Figure: Image taken from [1]

[1] Xiaoliang Ma et al. (2016). "A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables". In: *IEEE Transactions on Evolutionary Computation* 20.2, pp. 275–298.

Weighted Optimization Framework (WOF)

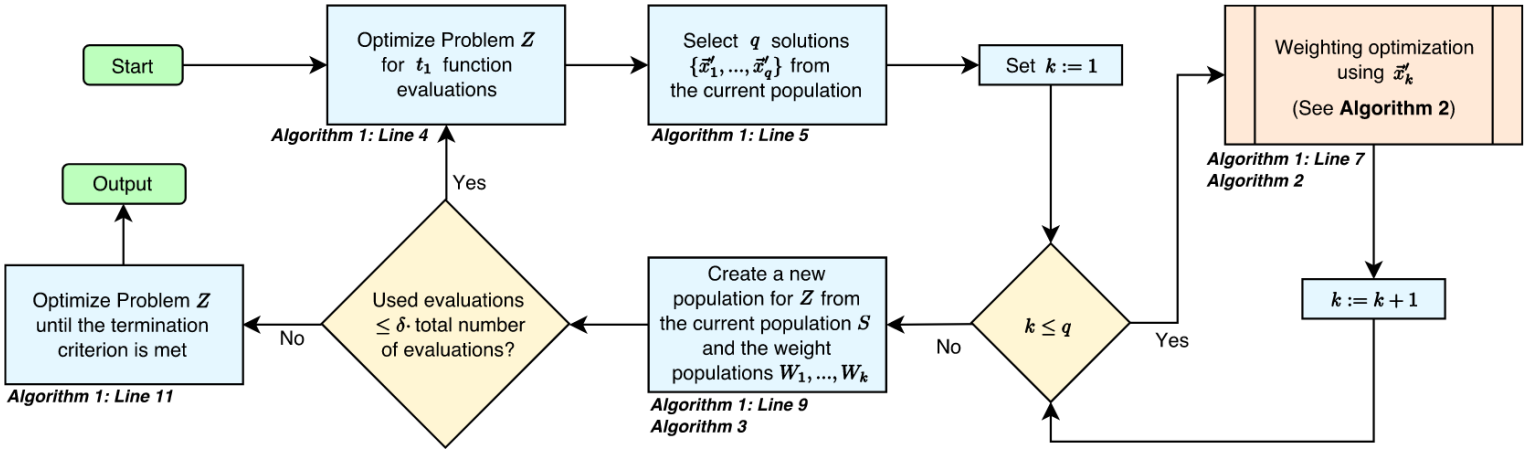


Figure: Weighted Optimization Framework [1], [2]

[1] Heiner Zille et al. (2016). "Weighted Optimization Framework for Large-scale Multi-objective Optimization". In: *Genetic and Evolutionary Computation Conference*. ACM, pp. 83–84.

[2] Heiner Zille et al. (2018). "A Framework for Large-Scale Multiobjective Optimization Based on Problem Transformation". In: *IEEE Transactions on Evolutionary Computation* 22.2, pp. 260–275.

LSGO: Potential Future Directions

1. The synergy between optimization and learning [1], [2].
2. The synergy between metaheuristics and classic mathematical programming:
   - Problem decomposition.
   - Variation operators [3], [4].
3. Exploiting problem structure: overlapping components.
4. Noise, dynamism [5], and uncertainty.
5. Constraint handling.

---

[1] Marcelo Rodrigues de Holanda Maia et al. (2020). "MineReduce: An approach based on data mining for problem size reduction". In: *Computers & Operations Research* 122, p. 104995.

[2] Yuan Sun, Xiaodong Li, and Andreas Ernst (2021). "Using Statistical Measures and Machine Learning for Graph Reduction to Solve Maximum Weight Clique Problems". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.5, pp. 1746–1760.

[3] Angus Kenny, Xiaodong Li, Andreas T Ernst, and Dhananjay Thiruvady (2017). "Towards solving large-scale precedence constrained production scheduling problems in mining". In: *GECCO*, pp. 1137–1144.

[4] Angus Kenny, Xiaodong Li, and Andreas T Ernst (2018). "A merge search algorithm and its application to the constrained pit problem in mining". In: *GECCO*, pp. 316–323.

[5] Danial Yazdani et al. (2019). "Scaling Up Dynamic Optimization Problems: A Divide-and-Conquer Approach". In: *IEEE Transactions on Evolutionary Computation*.

LSGO Resources

- Omidvar, M., Li, X., & Yao, X. (2021). A review of population-based metaheuristics for large-scale black-box global optimization: Part A. IEEE Transactions on Evolutionary Computation.
- Omidvar, M., Li, X., & Yao, X. (2021). A review of population-based metaheuristics for large-scale black-box global optimization: Part B. IEEE Transactions on Evolutionary Computation.

Large-Scale Combinatorial Optimization

Combinatorial Optimization

Many real-world problems can be formulated as combinatorial optimization, e.g., *Routing*, *Scheduling*, *Knapsack*, *Assignment*, *Matching*, *Covering*, *Packing* and *Partitioning*.

General *Integer Linear Programming* (ILP) Formulation:

$$\begin{aligned} &\max_{\boldsymbol{x}} \boldsymbol{cx}, \\ &s.t.\ A\boldsymbol{x} \leq b, \\ &\boldsymbol{x} \in Z^n. \end{aligned}$$

There often exist great opportunities for optimization techniques in real-world applications, e.g., in the mining industry, a small increase in efficiency can translate into millions of dollars in saving [1].

---

[1] Angus Kenny, Xiaodong Li, Andreas T Ernst, and Dhananjay Thiruvady (2017). "Towards solving large-scale precedence constrained production scheduling problems in mining". In: *GECCO*, pp. 1137–1144.

Large-Scale Combinatorial Optimization

Challenges: As the global economy grows, the size of optimization problems arising in industrial applications has increased significantly over the years. From 2003 to 2017, the size of the largest problem in MIPLIB has grown 70 times larger [1]. The dramatic increase in problem size has posed significant challenges to optimization algorithms.

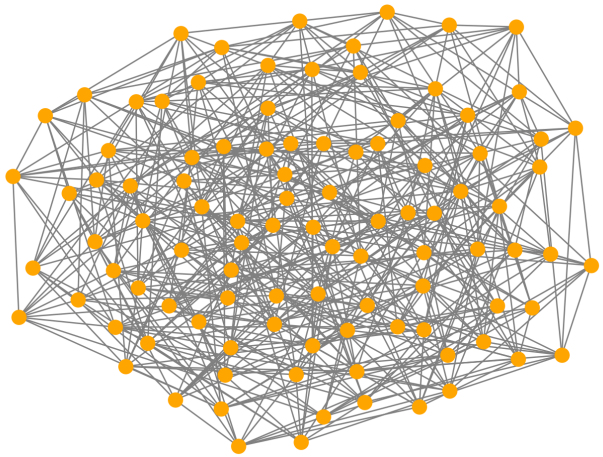
Solution approach: reduce the size of large-scale optimization problems to a point that is manageable by existing optimization algorithms.

[1] Ambros Gleixner et al. (2021). "MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library". In: *Mathematical Programming Computation*, pp. 1–48.

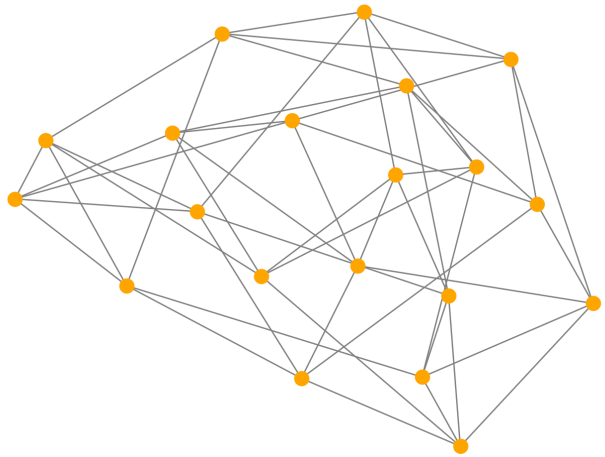
Problem Reduction

Motivation: The optimal solution of many combinatorial optimization problems is determined by only a small fraction of variables.

Aim: (1) significantly reduce the size of a problem, and (2) capture an optimal (or near-optimal) solution in the reduced space.



(a) Before Reduction



(b) After Reduction

Exact Problem Reduction

Exact methods: remove (or fix) decision variables that *cannot* be part of an optimal solution, typically based on mathematical reasoning or computation of an objective bound.

Heuristic methods: remove (or fix) decision variables based on certain rules typically extracted from feasible solutions.

Machine learning-based methods: greedily remove (or fix) decision variables that are *unlikely* to be part of an optimal solution, based on machine learning.

Problem Specific Bounds

Given a weighted graph $G(V, E, W)$, the Maximum Weight Clique (MWC) problem is defined as

$$\max_{\boldsymbol{x}} \sum_{i=1}^{|V|} w_i x_i,$$
$$s.t.\ \ x_i + x_j \leq 1,\ \forall (v_i, v_j) \notin E,$$
$$x_i \in \{0, 1\},\ v_i \in V.$$

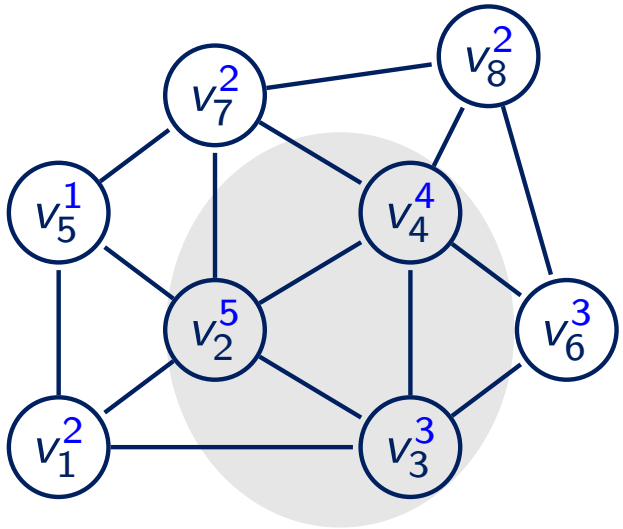


Figure: A MWC is a complete subgraph with maximum total node weights.

$UB(v_i) := w_i + \sum_{v_j \in \mathcal{N}_i} w_j$ ($\mathcal{N}_i := \{\forall\, v_j \,|\, (v_i, v_j) \in E\}$) defines an upper bound on the weight of cliques that include v_i. In other words, if $UB(v_i)$ is less than the incumbent solution, node v_i can be safely pruned.

Linear Relaxation Bounds

Integer Programming Formulation

$$z^{\mathrm{IP}} = \max_{\boldsymbol{x}} \sum_{i=1}^{|V|} w_i x_i,$$
$$s.t.\ x_i + x_j \leq 1,\ \forall (v_i, v_j) \notin E,$$
$$x_i \in \{0, 1\},\ v_i \in V.$$

Linear Programming Relaxation

$$z^{\mathrm{LP}} = \max_{\boldsymbol{x}} \sum_{i=1}^{|V|} w_i x_i,$$
$$s.t.\ x_i + x_j \leq 1,\ \forall (v_i, v_j) \notin E,$$
$$0 \leq x_i \leq 1,\ v_i \in V.$$

z^{LP} is an upper bound on z^{IP}: $z^{\mathrm{IP}} \leq z^{\mathrm{LP}}$.

IP is NP-hard, while LP can be solved in polynomial time.

LP bound is often used in MIP solvers such as CPLEX and Gurobi for prunning nodes in a branch-and-bound search tree.

Lagrangian Relaxation Bounds

Knapsack problem:

1. Number of items: n;
2. Item value: $\{v_1, v_2, \cdots, v_n\}$;
3. Item weight: $\{w_1, w_2, \cdots, w_n\}$;
4. Knapsack capability: C.

$$z^{\mathrm{IP}} = \max_{\boldsymbol{x}} \sum_{i=1}^{n} v_i x_i,$$

$$s.t. \sum_{i=1}^{n} w_i x_i \leq C,$$

$$x_i \in \{0,1\},\ i = 1, 2, \cdots, n.$$

$$L(\lambda) = \max_{\boldsymbol{x}} \sum_{i=1}^{n} v_i x_i + \lambda(C - \sum_{i=1}^{n} w_i x_i) = \max_{\boldsymbol{x}} \sum_{i=1}^{n} (v_i - \lambda w_i) x_i + \lambda C,$$

$$s.t.\ x_i \in \{0,1\},\ i = 1, 2, \cdots, n.$$

The original problem is decomposed into n 1-D problems.
$L(\lambda)$ is an upper bound on z^{IP} for any $\lambda \geq 0$: $L(\lambda) \geq z^{\mathrm{IP}}$.
$\min_{\lambda \geq 0} L(\lambda)$ provides the tightest LR bound.

Dantzig-Wolfe Reformulation

Given a graph $G(V, E)$, the aim of the vertex coloring problem is to assign a color to each vertex, such that the adjacent vertices have different colors and the total number of colors used is minimized.

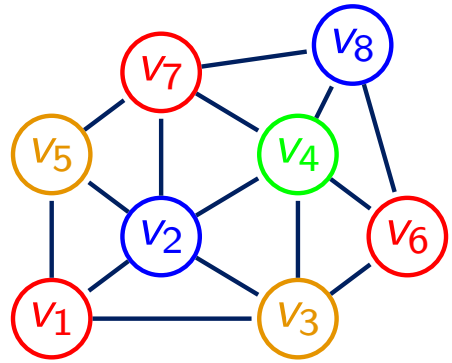


Figure: Vertex coloring example

S: a Maximal Independent Set (MIS).

$\mathbb{S}$: the set of all possible MISs in G.

$\mathbb{S}_v$: the set of MISs that contain $v \in V$.

z_S: indicating whether S is selected.

$$\min_{z} \sum_{S \in \mathbb{S}} z_S,$$

$$s.t. \sum_{S \in \mathbb{S}_v} z_S \geq 1, \quad v \in V;$$

$$z_S \in \{0, 1\}, \quad S \in \mathbb{S}.$$

The Dantzig-Wolfe reformulation with a large number of variables often has a tighter LP bound than the compact MIP formulation.

Column Generation

1: **function** CG($G(V, E)$)
2: Generate an initial subset of MISs $\mathbb{S}'$ (randomly).
3: **repeat**
4: (dual $\boldsymbol{u}^*$, y^*_{RMP}) $\leftarrow$ Solve a restricted master problem with $\mathbb{S}'$.
5: ($\boldsymbol{x}^*$, y^*_{PP}) $\leftarrow$ Solve a pricing problem with the dual value $\boldsymbol{u}^*$.
6: $\mathbb{S}' \leftarrow \mathbb{S}' \cup \boldsymbol{x}^*$ ▷ add the generated column $\boldsymbol{x}^*$ to $\mathbb{S}'$.
7: **until** the minimum reduced cost $y^*_{\mathrm{PP}} \geq 0$.
8: **return** the optimal objective value of the LP y^*_{RMP}.

Restricted Master Problem:

$$\min_{\boldsymbol{z}} \sum_{S \in \mathbb{S}'} z_S,$$
$$s.t. \sum_{S \in \mathbb{S}'_v} z_S \geq 1, \quad v \in V;$$
$$0 \leq z_S \leq 1, \quad S \in \mathbb{S}'.$$

Pricing Problem:

$$\min_{\boldsymbol{x}} 1 - \sum_{i=1}^{|V|} u^*_i \cdot x_i,$$
$$s.t.\ x_i + x_j \leq 1, \quad (i,j) \in E;$$
$$x_i \in \{0,1\}, \quad v_i \in V.$$

CG can be embedded into B&B, i.e., Branch-and-Price method.

Heuristic Problem Reduction

Exact methods: remove (or fix) decision variables that *cannot* be part of an optimal solution, typically based on mathematical reasoning or computation of an objective bound.

Heuristic methods: remove (or fix) decision variables based on certain rules typically extracted from feasible solutions.

Machine learning-based methods: greedily remove (or fix) decision variables that are *unlikely* to be part of an optimal solution, based on machine learning.

Large Neighborhood Search (LNS)

LNS iteratively chooses a subset of variables to optimize while leaving the remainder fixed [1]:

1: **function** LNS(instance I)
2: Generate an initial feasible solution $\boldsymbol{x}$.
3: **while** CPU time limit not reached **do**
4: Select a neighbourhood of $\boldsymbol{x}$.
5: $\boldsymbol{x} \leftarrow$ Optimize the neighbourhood.
6: **return** the best solution found $\boldsymbol{x}$.

[1] David Pisinger and Stefan Ropke (2010). "Large Neighborhood Search". In: *Handbook of Metaheuristics*. Springer, pp. 399–419.

Construct Merge Search & Adapt (CMSA)

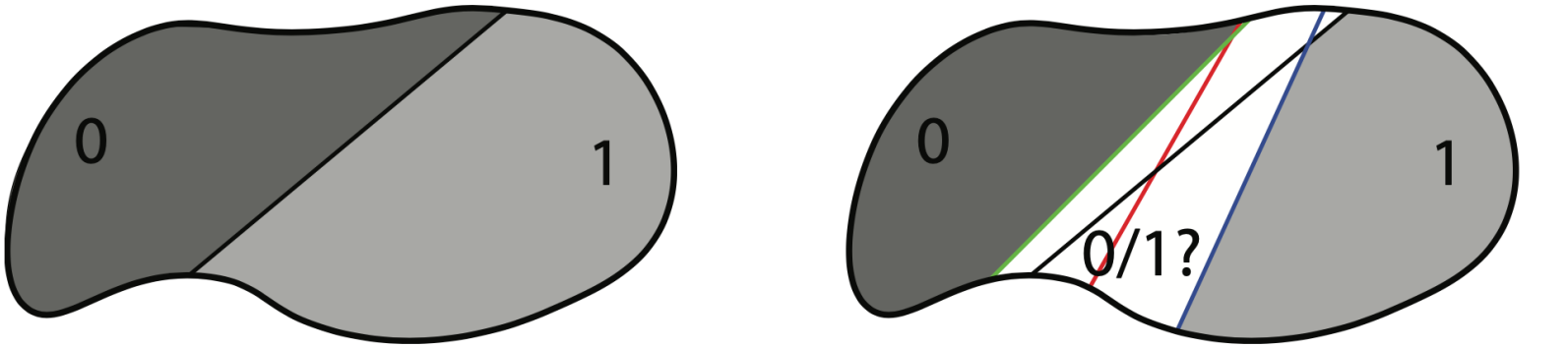
CMSA forms a reduced problem by using the components appearing in sample solutions [1]:

1: **function** CMSA(instance I)
2: Initialize the sub-problem I_s as empty.
3: **while** CPU time limit not reached **do**
4: $S \leftarrow$ Sample solutions using a probabilistic method from I.
5: Add the solution components of S into the sub-problem I_s.
6: Solve the sub-problem I_s to optimality.
7: Remove some low-quality solution components from I_s.
8: **return** the best solution found.

[1] Christian Blum, Pedro Pinacho, et al. (2016). "Construct, merge, solve & adapt a new general algorithm for combinatorial optimization". In: *Computers & Operations Research* 68, pp. 75–88.

Merge Search (MS)

Merge search constrains certain variables to be the same, and replaces multiple variables in the original problem with a single variable [1]



(a) Single solution **(b) Multiple solutions overlaid**

[1] Angus Kenny, Xiaodong Li, and Andreas T Ernst (2018). "A merge search algorithm and its application to the constrained pit problem in mining". In: *GECCO*, pp. 316–323; Angus Kenny, Xiaodong Li, Andreas T Ernst, and Yuan Sun (2019). "An improved merge search algorithm for the constrained pit problem in open-pit mining". In: *GECCO*, pp. 294–302.

Comparison between LNS, CMSA and MS

A series of studies [1] have shown that:

1. CMSA is better than LNS when solutions contain rather few items.
2. LNS is better than CMSA when solutions contain rather many items

MS seems more effective on medium-sized problems, whereas CMSA performs better on large problems [2].

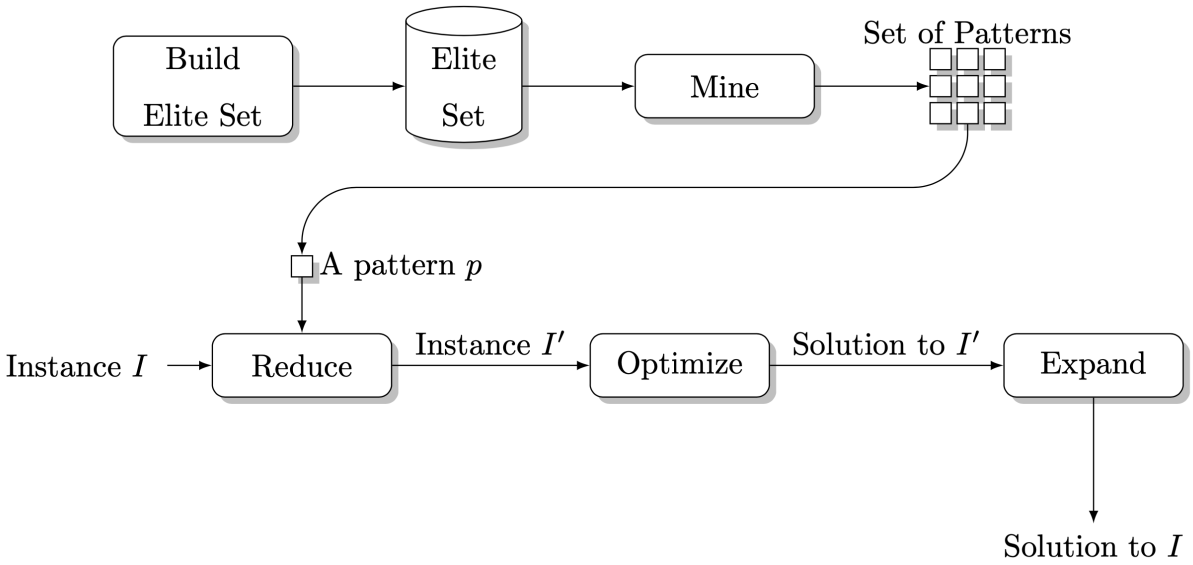
---

[1] Evelia Lizárraga et al. (2017). "Construct, merge, solve and adapt versus large neighborhood search for solving the multi-dimensional Knapsack problem: Which one works better when?" In: *ECECCO*. Springer, pp. 60–74; Christian Blum (2020). "Minimum common string partition: on solving large-scale problem instances". In: *International Transactions in Operational Research* 27.1, pp. 91–111; Christian Blum and Gabriela Ochoa (2021). "A comparative analysis of two matheuristics by means of merged local optima networks". In: *European Journal of Operational Research* 290.1, pp. 36–56.

[2] Dhananjay Thiruvady et al. (2020). "Solution Merging in Matheuristics for Resource Constrained Job Scheduling". In: *Algorithms* 13.10, p. 256.

MineReduce

Reduce problem size based on data mining [1]:



1. The data mining algorithm, FPmax*, is used to mine frequent patterns (i.e., solution components).
2. The mined patterns are used to fix or merge decision variables.

[1] Marcelo Rodrigues de Holanda Maia et al. (2020). "MineReduce: An approach based on data mining for problem size reduction". In: *Computers & Operations Research* 122, p. 104995.

Frequent Pattern-Based Search (FPBS)

Hybridizing meta-heuristics with data mining [1]:

1. Generate a set of high-quality solutions via a greedy method or population-based method.
2. Extract frequent patterns (solution components) from high-quality solutions using data mining techniques.
3. Use the extracted patterns to construct new solutions.

---

[1] Daniel Martins et al. (2018). "Making a state-of-the-art heuristic faster with data mining". In: *Annals of Operations Research* 263.1, pp. 141–162; Yangming Zhou et al. (2020). "Frequent Pattern-Based Search: A Case Study on the Quadratic Assignment Problem". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

Machine Learning-Based Problem Reduction

Exact methods: remove (or fix) decision variables that *cannot* be part of an optimal solution, typically based on mathematical reasoning or computation of an objective bound.

Heuristic methods: remove (or fix) decision variables based on certain rules typically extracted from sample solutions.

Machine learning-based methods: greedily remove (or fix) decision variables that are *unlikely* to be part of an optimal solution, based on machine learning.

Machine Learning-Based Problem Reduction

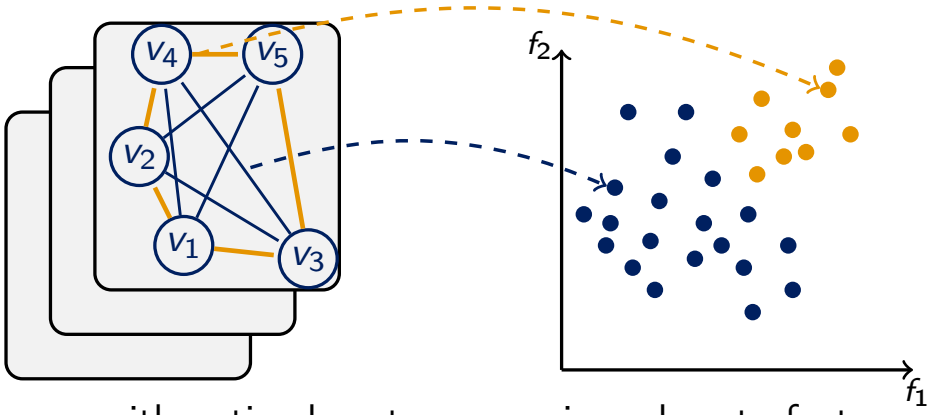
Automatically learn a rule to fix decision variables based on previously solved problem instances [1]:

1. Solve a set of easy problem instances to optimality;
2. Label decision variables based on their optimal solution values;
3. Extract features to characterize each decision variable;
4. Train a machine learning model to predict the optimal solution value for each decision variable in training problem instances;
5. Remove (or fix) the decision variables that are predicted not to be part of an optimal solution for an unseen *test* problem instance.

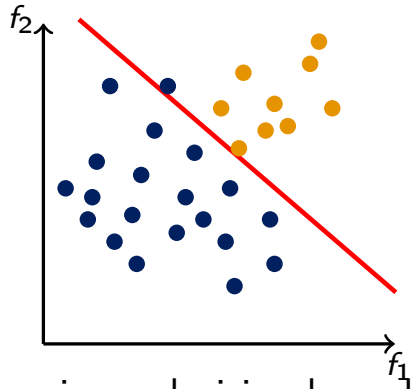
---

[1] Yuan Sun, Xiaodong Li, and Andreas Ernst (2021). "Using Statistical Measures and Machine Learning for Graph Reduction to Solve Maximum Weight Clique Problems". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.5, pp. 1746–1760.

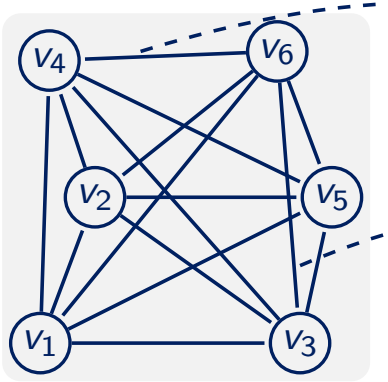
Illustration of Training and Testing



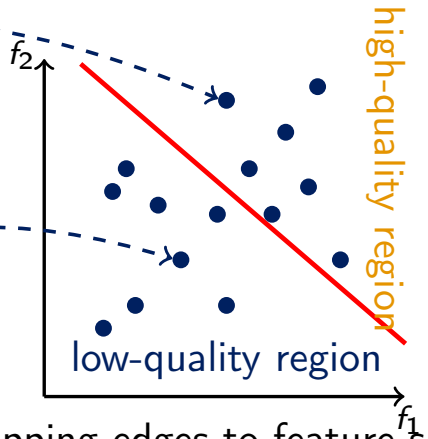
instances with optimal routes mapping edges to feature space



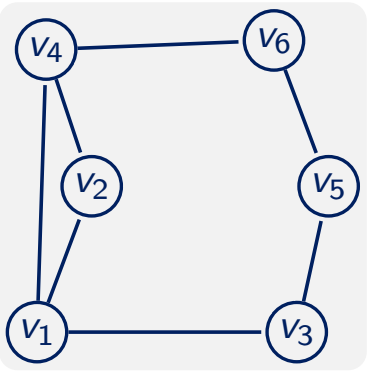
learning a decision boundary



a test problem instance



mapping edges to feature space



removing low-quality edges

Feature Extraction

1. *Problem-specific features* such as vertex or edge weight in a graph [1].
2. *ILP features* extracted from ILP formulations, e.g., cost coefficient or number of non-zeros in constraint matrix [2].
3. *LP features* computed from the LP relaxation of the ILP, e.g., reduced cost [3].
4. *Statistical features* computed from sample solutions, e.g., Pearson correlation coefficient [4].

---

[1] Juho Lauri and Sourav Dutta (2019). "Fine-grained search space classification for hard enumeration variants of subset problems". In: *AAAI*. vol. 33, pp. 2314–2321; Yuan Sun, Xiaodong Li, and Andreas Ernst (2021). "Using Statistical Measures and Machine Learning for Graph Reduction to Solve Maximum Weight Clique Problems". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.5, pp. 1746–1760.

[2] Jian-Ya Ding et al. (2020). "Accelerating Primal Solution Findings for Mixed Integer Programs Based on Solution Prediction". In: *AAAI*, pp. 1452–1459.

[3] Jian-Ya Ding et al. (2020). "Accelerating Primal Solution Findings for Mixed Integer Programs Based on Solution Prediction". In: *AAAI*, pp. 1452–1459; James Fitzpatrick et al. (2021). "Learning to Sparsify Travelling Salesman Problem Instances". In: *arXiv preprint arXiv:2104.09345*.

[4] Yuan Sun, Xiaodong Li, and Andreas Ernst (2021). "Using Statistical Measures and Machine Learning for Graph Reduction to Solve Maximum Weight Clique Problems". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.5, pp. 1746–1760.

Classification Models

1. Support Vector Machine [1]
2. Logistic Regression [2]
3. K-Nearest Neighbor [3]
4. Graph Neural Network [4].

---

[1] Yuan Sun, Xiaodong Li, and Andreas Ernst (2021). "Using Statistical Measures and Machine Learning for Graph Reduction to Solve Maximum Weight Clique Problems". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.5, pp. 1746–1760; Álinson S Xavier et al. (2020). "Learning to solve large-scale security-constrained unit commitment problems". In: *INFORMS Journal on Computing*.

[2] Juho Lauri and Sourav Dutta (2019). "Fine-grained search space classification for hard enumeration variants of subset problems". In: *AAAI*. vol. 33, pp. 2314–2321; Juho Lauri, Sourav Dutta, et al. (2020). "Learning fine-grained search space pruning and heuristics for combinatorial optimization". In: *arXiv preprint arXiv:2001.01230*.

[3] Álinson S Xavier et al. (2020). "Learning to solve large-scale security-constrained unit commitment problems". In: *INFORMS Journal on Computing*.

[4] Zhuwen Li et al. (2018). "Combinatorial optimization with graph convolutional networks and guided tree search". In: *NeurIPS*, pp. 539–548; Jian-Ya Ding et al. (2020). "Accelerating Primal Solution Findings for Mixed Integer Programs Based on Solution Prediction". In: *AAAI*, pp. 1452–1459.

Generalization

How does the trained machine learning model generalize to unseen problem instances [1]*?*

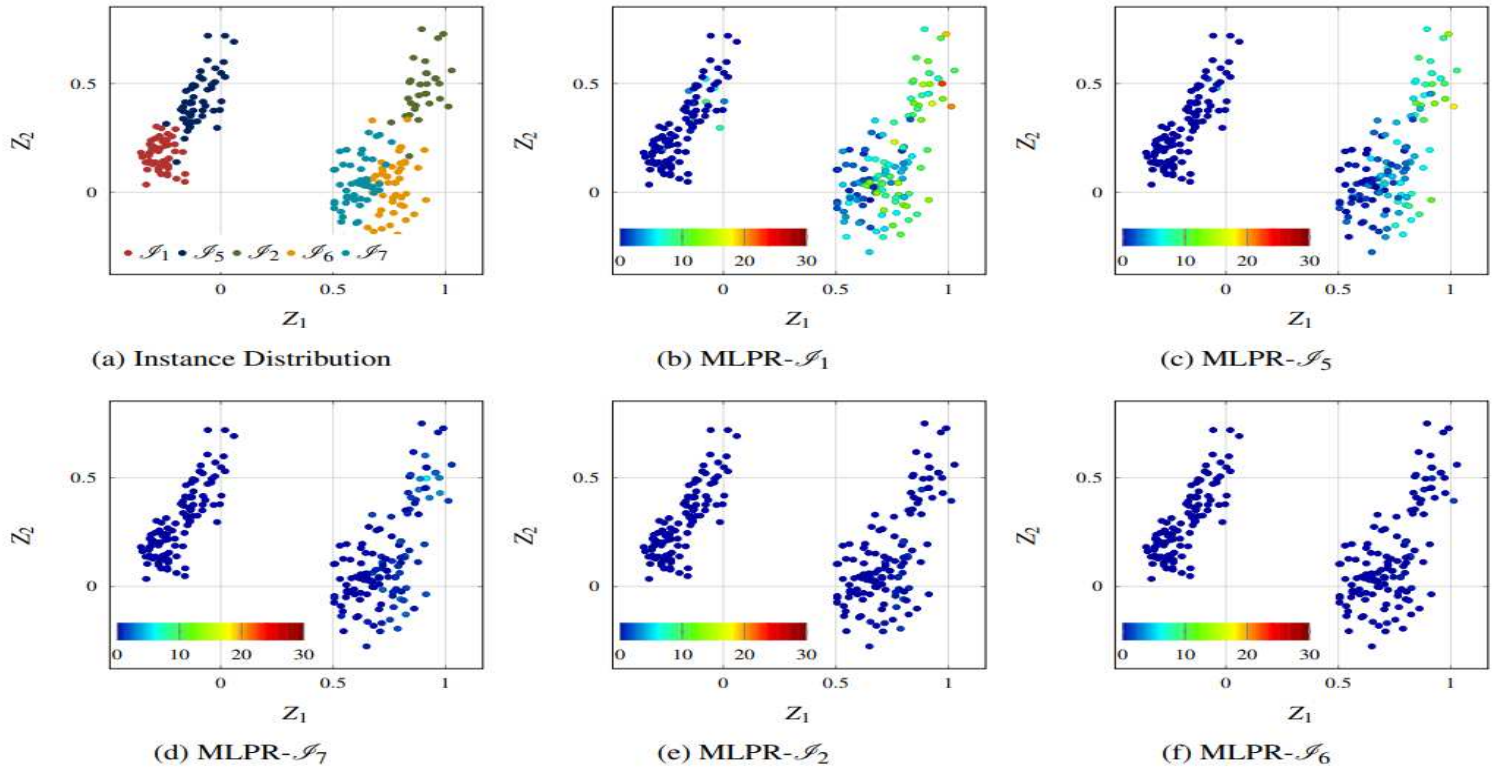


Figure: Each dot represents an instance in the 2-D feature space (Z_1 and Z_2). In figure (a), dot color represents the category where instance is from; while in figure (b) to (f), dot color represents the optimality gap (%) generated by the MLPR models.

---

[1] Yuan Sun, Andreas Ernst, et al. (2020). "Generalization of Machine Learning for Problem Reduction: A Case Study on Travelling Salesman Problems". In: *OR Spectrum*.

Multi-Stage Pruning

Apply the trained ML model to prune the search space of a problem instance **recursively** [1]:

1: **function** Multi-Stage Pruning(instance I)
2: **while** Stopping criterion not met **do**
3: Compute the features for each decision variable in I
4: Apply the trained ML to predict the optimal solution value.
5: $I \leftarrow$ prune the instance I based on the ML predictions.
6: **return** the reduced instance I.

[1] Marco Grassia et al. (2019). "Learning Multi-Stage Sparsification for Maximum Clique Enumeration". In: *arXiv preprint arXiv:1910.00517*; Juho Lauri, Sourav Dutta, et al. (2020). "Learning fine-grained search space pruning and heuristics for combinatorial optimization". In: *arXiv preprint arXiv:2001.01230*.

Feasibility and Optimality Gap

How to ensure that there is at least one feasible solution in the reduced problem? More generally, how to compute an optimality gap for the best solution generated from the reduced problem?

One can guarantee feasibility of the pruned problem instance by adding one feasible solution to the reduced space [1].

If an approximation algorithm exists which can generate a feasible solution with a valid bound, adding that feasible solution to the reduced space can also provide an optimality gap for the best solution in the reduced problem [2].

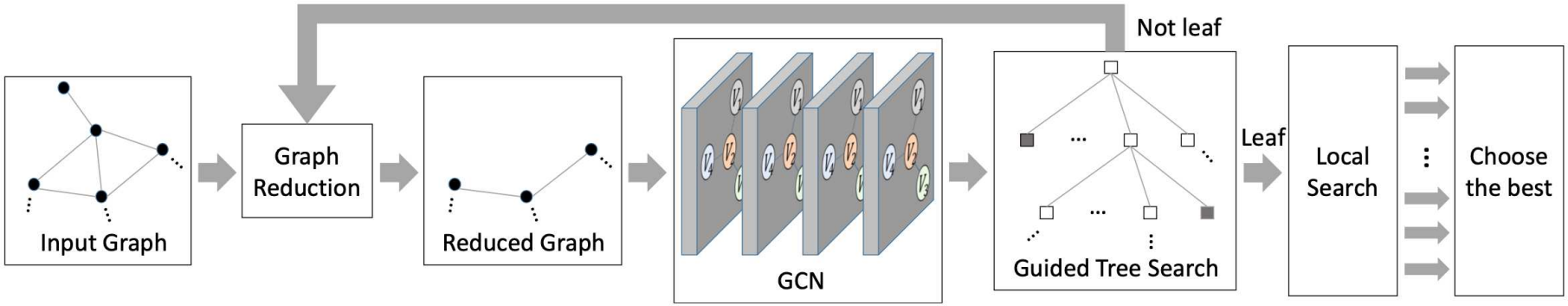
---

[1] Yuan Sun, Andreas Ernst, et al. (2020). "Generalization of Machine Learning for Problem Reduction: A Case Study on Travelling Salesman Problems". In: *OR Spectrum*.

[2] James Fitzpatrick et al. (2021). "Learning to Sparsify Travelling Salesman Problem Instances". In: *arXiv preprint arXiv:2104.09345*.

Guiding Tree-Search Algorithms

A Graph Convolutional Network was trained to predict the optimal solution values for binary variables, which were then used to guide a tree-search algorithm for high-quality solutions [1].



[1] Zhuwen Li et al. (2018). "Combinatorial optimization with graph convolutional networks and guided tree search". In: *NeurIPS*, pp. 539–548.

Generating Cut

A Graph Convolutional Network was trained to predict the optimal solution values for binary variables, which were then used to generate a global inequality constraint to prune the search space [1]:

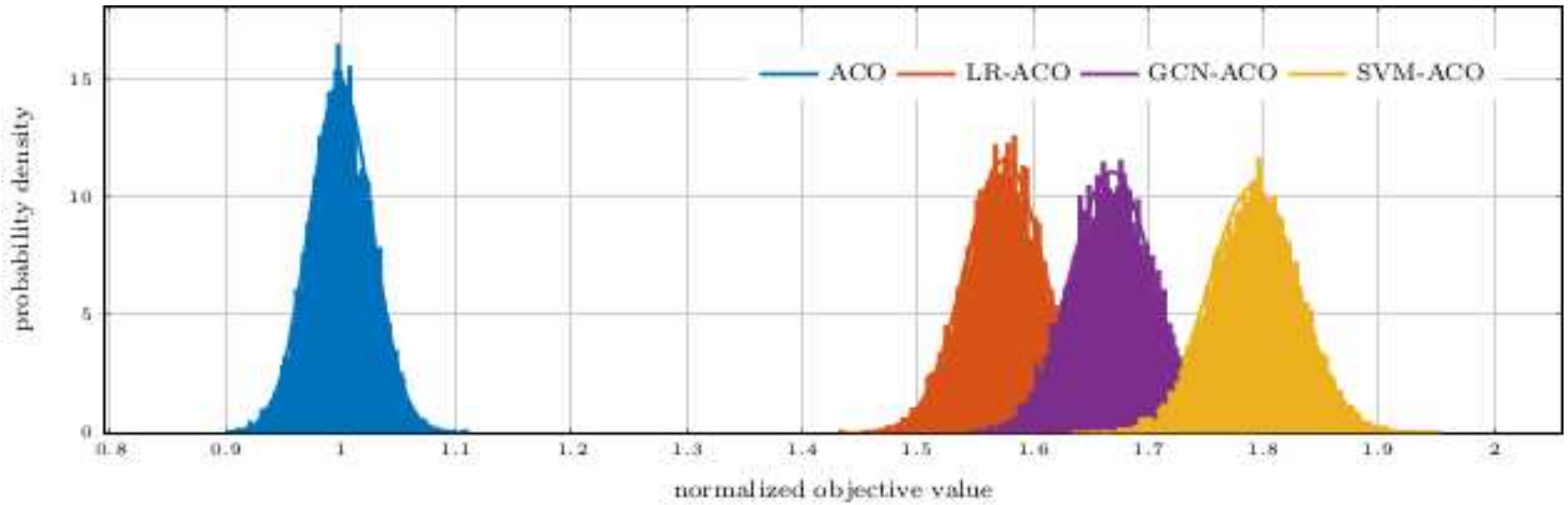
$$\Delta(\boldsymbol{x}, \hat{\boldsymbol{x}}, \mathcal{S}) := \sum_{j \in \mathcal{S}: \hat{x}_j = 0} x_j + \sum_{j \in \mathcal{S}: \hat{x}_j = 1} (1 - x_j) \leq \Phi, \qquad (19)$$

- $\mathcal{S}$ is the set of binary variables.
- $\hat{\boldsymbol{x}}$ are predicted solution values of binary variables $\boldsymbol{x}$;
- Φ is a parameter that controls the size of the reduced problem space.

[1] Jian-Ya Ding et al. (2020). "Accelerating Primal Solution Findings for Mixed Integer Programs Based on Solution Prediction". In: *AAAI*, pp. 1452–1459.

Boosting Meta-Heuristics

An ML model was trained to predict the optimal solution values of binary decision variables, which were then incorporated into the Ant Colony Optimization (ACO) algorithm, to bias its sampling towards high-quality solutions [1].



[1] Yuan Sun, Sheng Wang, et al. (2020). "Boosting Ant Colony Optimization via Solution Prediction and Machine Learning". In: *arXiv preprint arXiv:2008.04213*.

Warm Start

[1] trained a Gated Graph Convolutional Network to predict the solution for the the Boolean satisfiability problem. The predicted solution is then used as the starting point for stochastic local search methods, resulting in a significant performance improvement.

[2] trained a model to predict redundant constraints, good initial feasible solutions and affine subspaces where the optimal solution is likely to lie, leading to significant reduction in problem size.

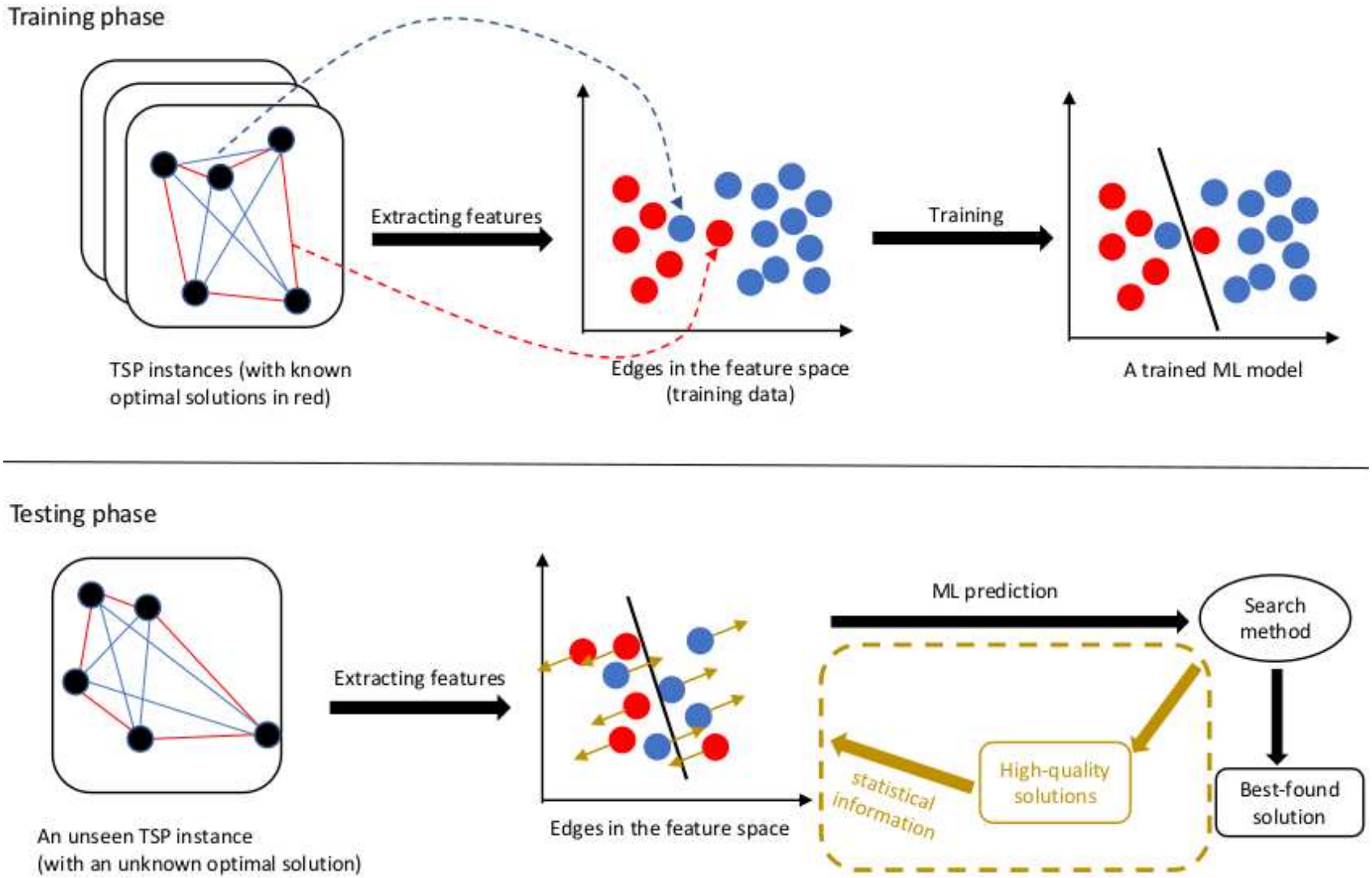
---

[1] Wenjie Zhang et al. (July 2020). "NLocalSAT: Boosting Local Search with Solution Prediction". In: *IJCAI*, pp. 1177–1183.

[2] Álinson S Xavier et al. (2020). "Learning to solve large-scale security-constrained unit commitment problems". In: *INFORMS Journal on Computing*.

Adaptive Solution Prediction

Adaptive solution prediction framework [1].



[1] Yunzhuang Shen et al. (2023). "Adaptive solution prediction for combinatorial optimization". In: *European Journal of Operational Research* 309.3, pp. 1392–1408.

Problem Reduction for Dantzig-Wolfe Reformulation

Learning to predict optimal columns with vertex coloring problem as an example [1].

1: **function** Training(a set of training graphs)
2: Each MIS in a training graph is used as a training instance.
3: Extract features to characterise each MIS.
4: Compute the class label of each MIS.
5: Train a classification algorithm on the training set.

Feature extraction:

- Problem-specific features, e.g., the size of a MIS;
- LP features, e.g., reduced cost and LP solution value;
- Statistical measures based on sample solutions.

Computing class label: A MIS is a positive training instance if it belongs to *any* optimal solution; otherwise it is negative.

Classification algorithms: Any existing classification algorithm can be applied.

---

[1] Yuan Sun, Andreas T Ernst, et al. (2023). "Learning to Generate Columns with Application to Vertex Coloring". In: *International Conference on Learning Representations (ICLR)*.

Problem Reduction for Dantzig-Wolfe Reformulation

ML-based Column Generation [1].

function MLCG($G(V, E)$, N_{it})
    Generate an initial subset of MISs $\mathbb{S}$
    for i from 1 to N_{it} **do**
        Calculate the features for each MIS $S \in \mathbb{S}$;
        Evaluate the quality of each MIS using the trained ML model;
        Update $\mathbb{S}$ by replacing low-quality MISs with new ones.
    Solve the reduced MIP with $\mathbb{S}$ using Gurobi.
    return the best solution found.

Evaluating MIS:

- The prediction speed of the machine learning model matters.
- Linear support vector machine works better than decision tree.

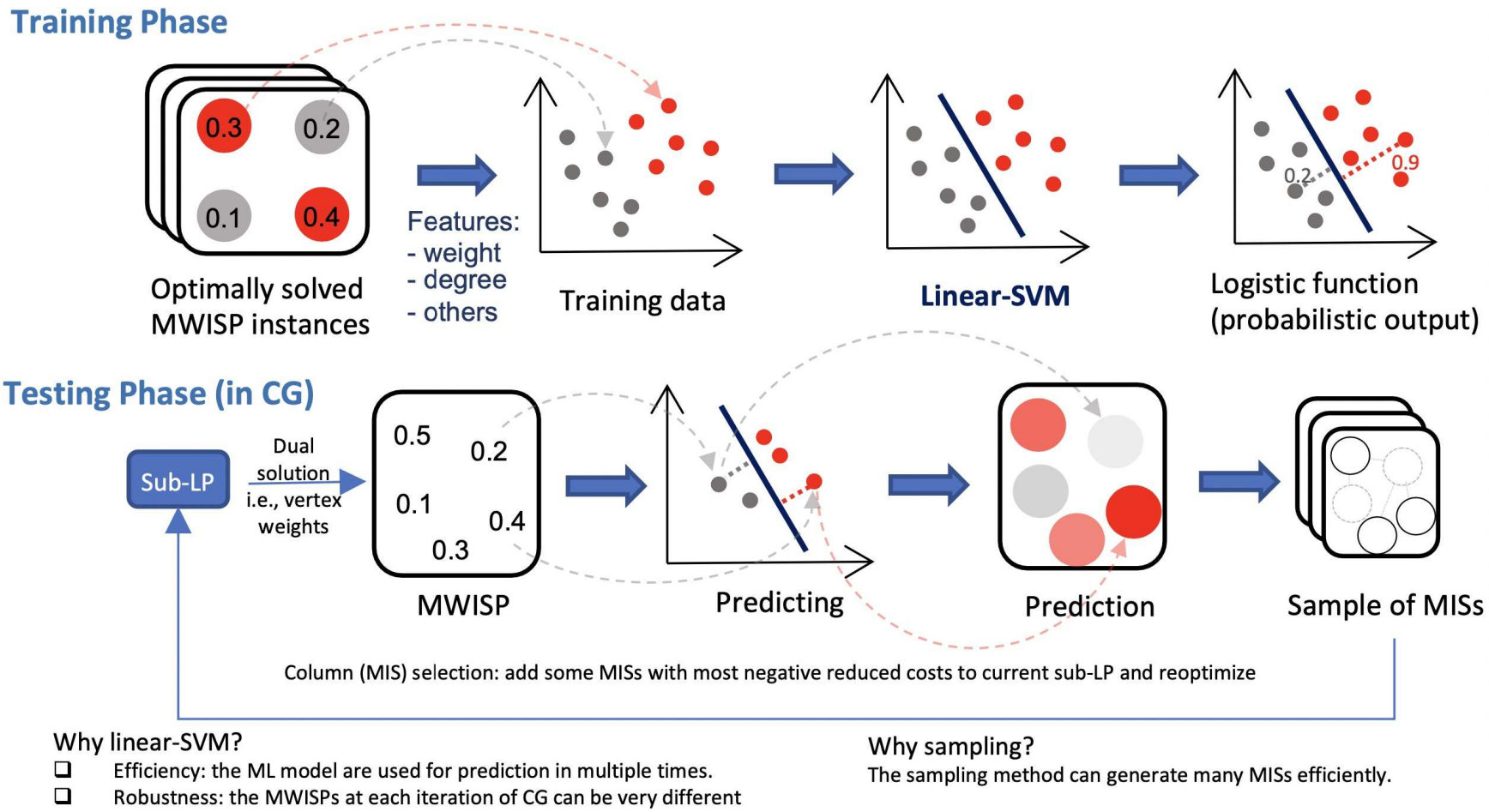
Updating MIS:

- The diversity of $\mathbb{S}$ matters
- Random generation performs similarly with a cross-over type approach.

[1] Yuan Sun, Andreas T Ernst, et al. (2023). "Learning to Generate Columns with Application to Vertex Coloring". In: *International Conference on Learning Representations (ICLR)*.

ML-based Pricing Heuristic for Column Generation

Predict the optimal solution of the pricing problem, then use the prediction to sample multiple high-quality columns [1].



[1] Yunzhuang Shen et al. (2022). "Enhancing column generation by a machine-learning-based pricing heuristic for graph coloring". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 9, pp. 9926–9934.

Predicting Optimal Objective Values

[1] used regression techniques to estimate the production of offshore wind parks.

[2] trained machine learning models to learn bounds on the optimal objective values, which are then used to generate cuts to prune search space.

---

[1] Martina Fischetti and Marco Fraccaro (2019). "Machine learning meets mathematical optimization to predict the optimal production of offshore wind parks". In: *Computers & Operations Research* 106, pp. 289–297.

[2] Franco Peschiera et al. (2020). "A novel solution approach with ML-based pseudo-cuts for the Flight and Maintenance Planning problem". In: *OR Spectrum*, pp. 1–30.

Predicting Magnitude of Changes in Optimal Solutions

In practice, we often need to solve, on a regular basis, many problem instances, of which the parameters change slightly, e.g., routing problem that a navigation company faces.

[1] built a regression model to learn the proportion of optimal solution values that can be reused when the problem instance is perturbed slightly. The predicted information is used as a constraint to reduce the problem size.

---

[1] Andrea Lodi et al. (2020). "Learning to handle parameter perturbations in combinatorial optimization: an application to facility location". In: *EURO Journal on Transportation and Logistics* 9.4, p. 100023.

Predicting Optimal Values for Continous Variables

[1] trained a regression model to estimate the optimal value for *continuous* decision variables, and showed improvement performance in blood supply chain management.

The trained ML model cannot be applied to problem instances with different sizes.

---

[1] Babak Abbasi et al. (2020). "Predicting solutions of large-scale optimization problems via machine learning: A case study in blood supply chain management". In: *Computers & Operations Research*, p. 104941.

Neural Large Neighborhood Search

[1] trained a deep neural network via policy gradient reinforcement learning to repair a destroyed (i.e. incomplete) solution for Large Neighborhood Search (LNS).

[2] showed that one can learn a good neighborhood selector using imitation and reinforcement learning techniques. More specifically, a good decomposition of decision variables for LNS can be learned via ML.

---

[1] André Hottung and Kevin Tierney (2020). "Neural Large Neighborhood Search for the Capacitated Vehicle Routing Problem". In: *ECAI 2020*.

[2] Jialin Song et al. (2020). "A General Large Neighborhood Search Framework for Solving Integer Linear Programs". In: *NeurIPS* 33.

Remarks

Exact methods guarantee that the reduced problem always contains an original optimal solution, but in many cases, they are computationally expensive and often do not significantly reduce the problem size.

Inexact methods (i.e., heuristic and machine learning-based methods) are more aggressive at pruning the search spaces of large-scale problems, but it is difficult to provide an optimality guarantee for those methods.

Challenges and Opportunities: I

Hybrid methods that combine the merits of Mathematical Programming, Meta-heuristics and Machine Learning are often effective in solving large-scale combinatorial optimization problems.

An avenue for future research is to explore effective ways of using machine learning techniques to improve traditional algorithms, such as Neural Large neighborhood Search [1] and Machine Learning Ant Colony Optimization [2].

---

[1] Jialin Song et al. (2020). "A General Large Neighborhood Search Framework for Solving Integer Linear Programs". In: *NeurIPS* 33.

[2] Yuan Sun, Sheng Wang, et al. (2020). "Boosting Ant Colony Optimization via Solution Prediction and Machine Learning". In: *arXiv preprint arXiv:2008.04213*.

Challenges and Opportunities: II

Machine learning-based problem reduction is quite new. It would be interesting to fully test the effectiveness of this approach on various types of combinatorial optimization problems, such as dynamic, stochastic, and multi-objective optimization problems.

Challenges and Opportunities: III

How to develop a strong optimality guarantee for inexact (i.e., heuristic and machine learning-based) problem reduction methods is challenging and important.

May borrow some ideas from statistical learning theory, operations research, or approximation algorithm design.

Challenges and Opportunities: IV

The machine learning-based problem reduction method can be easily adapted to various problems.

However, it is nontrivial to develop a machine learning model that can be used to prune the search space for general ILPs (or at least a class of problems) without the need of re-training the machine learning model.

If this is successful, the generic machine learning model can be incorporated into commercial solvers such as Gurobi and CPLEX as a side package to solve large-scale optimization problems, expected to bring significant benefit.

Thank you!