10-708 Probabilistic Graphical Models

Machine Learning Department
School of Computer Science
Carnegie Mellon University

MAP Inference by MILP

Matt Gormley
Lecture 9
Mar. 1, 2021

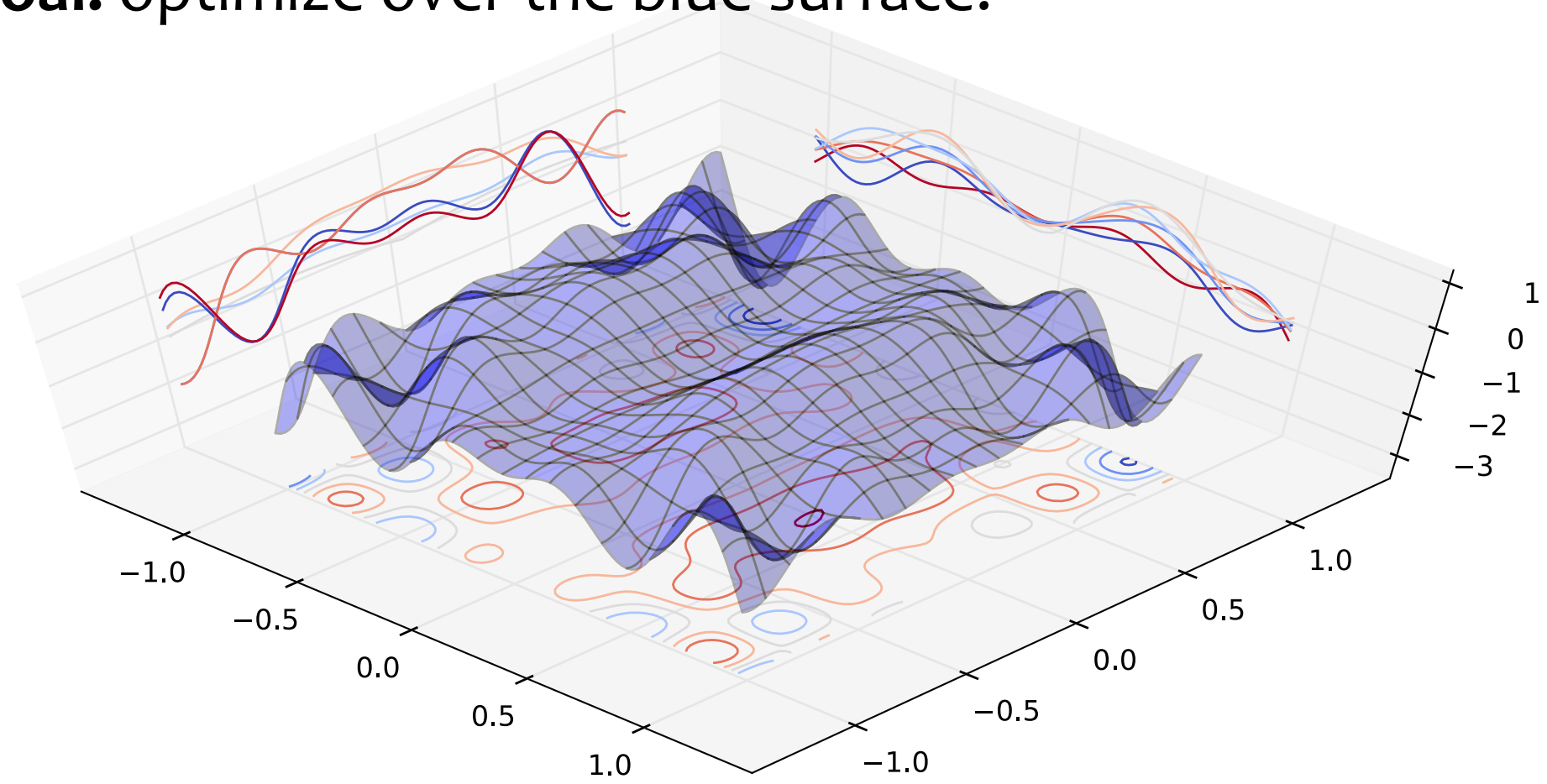
Reminders

- Quiz 1:
  - Fri, Mar. 05 during class time
  - Material: Lectures 1 – 8 only
  - Logistics: see forthcoming Piazza post
- Homework 2: Exact inference and supervised learning (CRF+RNN)
  - Out: Wed, Feb. 24
  - Due: Wed, Mar. 10 at 11:59pm

BRANCH AND BOUND

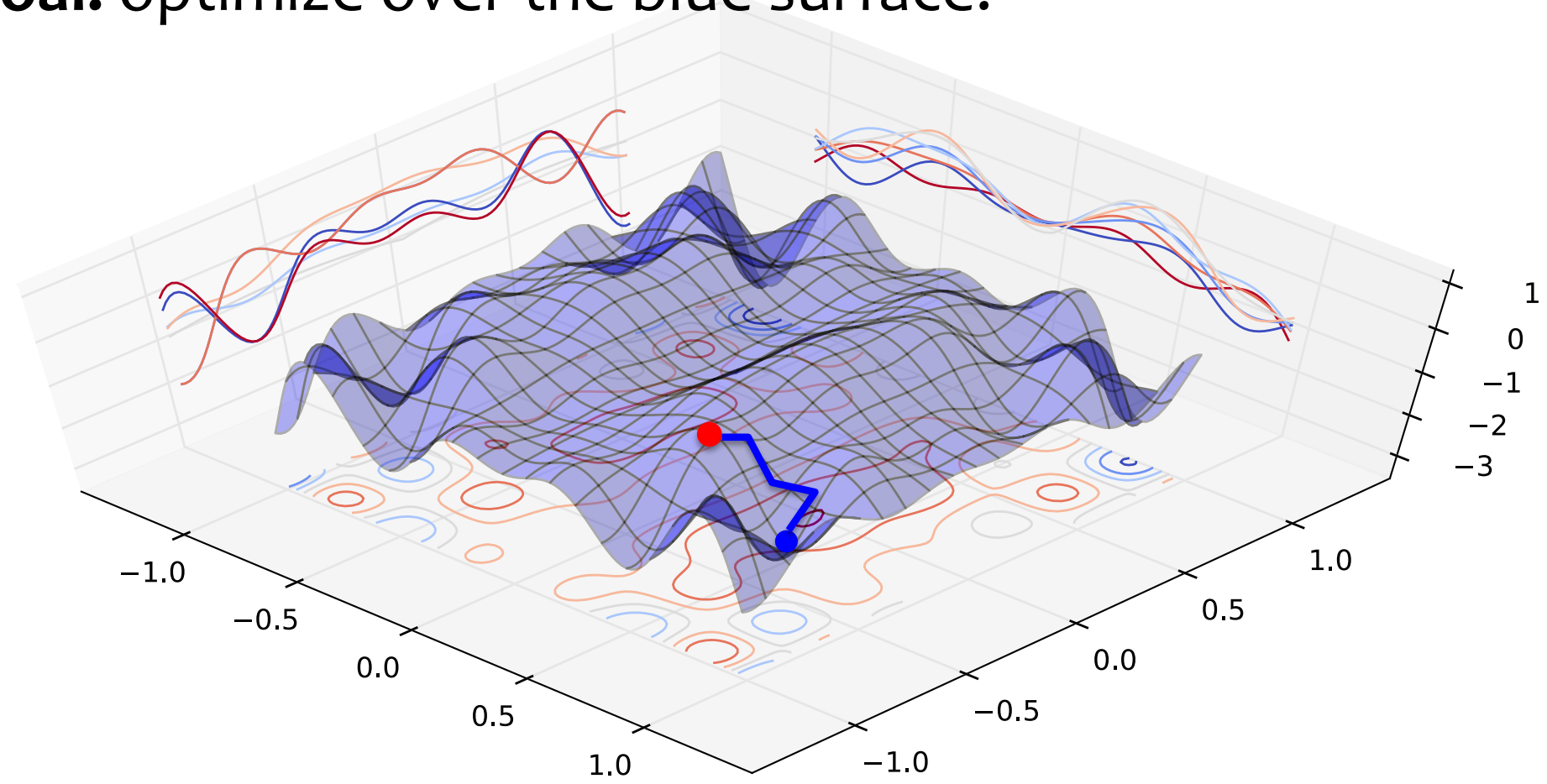
Background: Nonconvex Global Optimization

Goal: optimize over the blue surface.



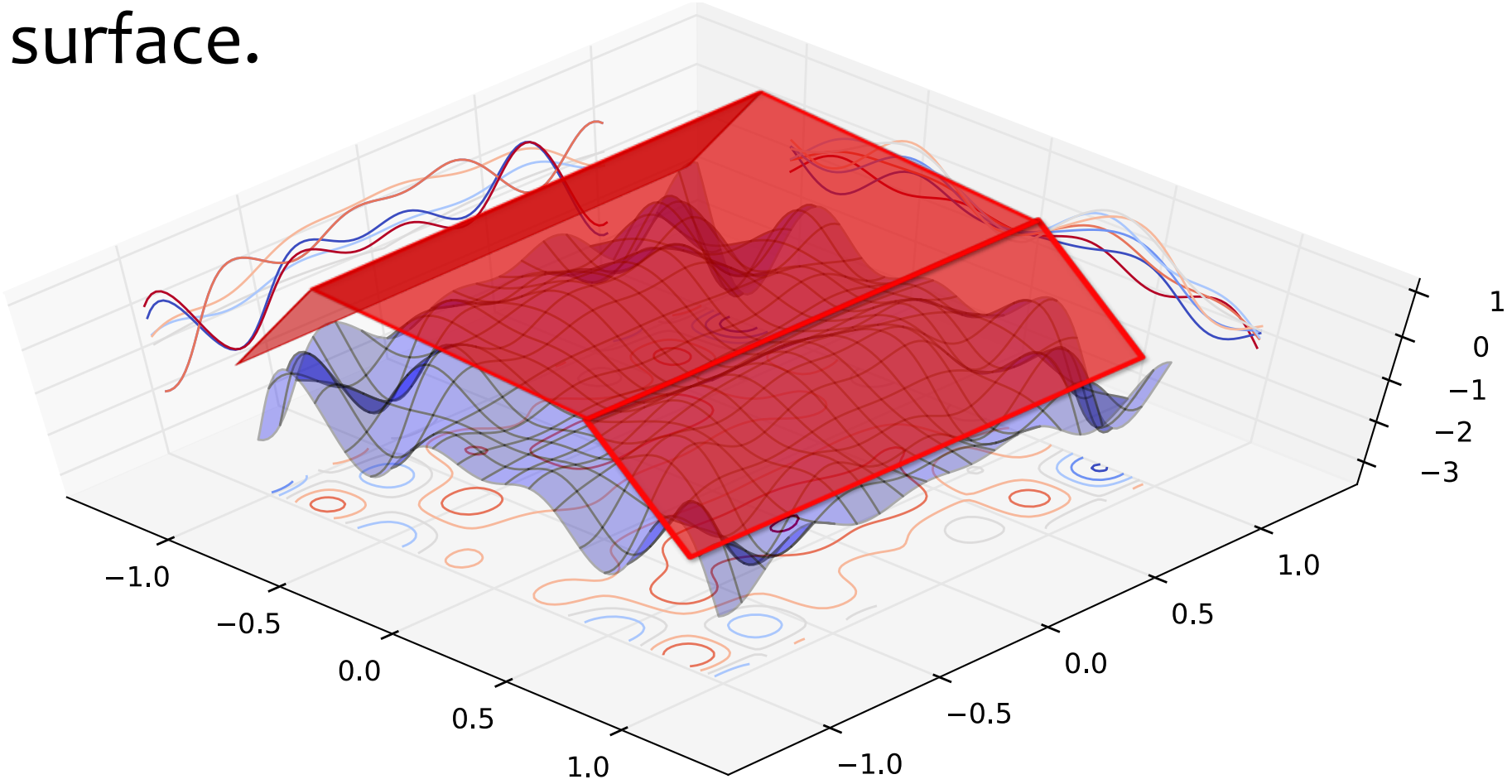
Background: Nonconvex Global Optimization

Goal: optimize over the blue surface.



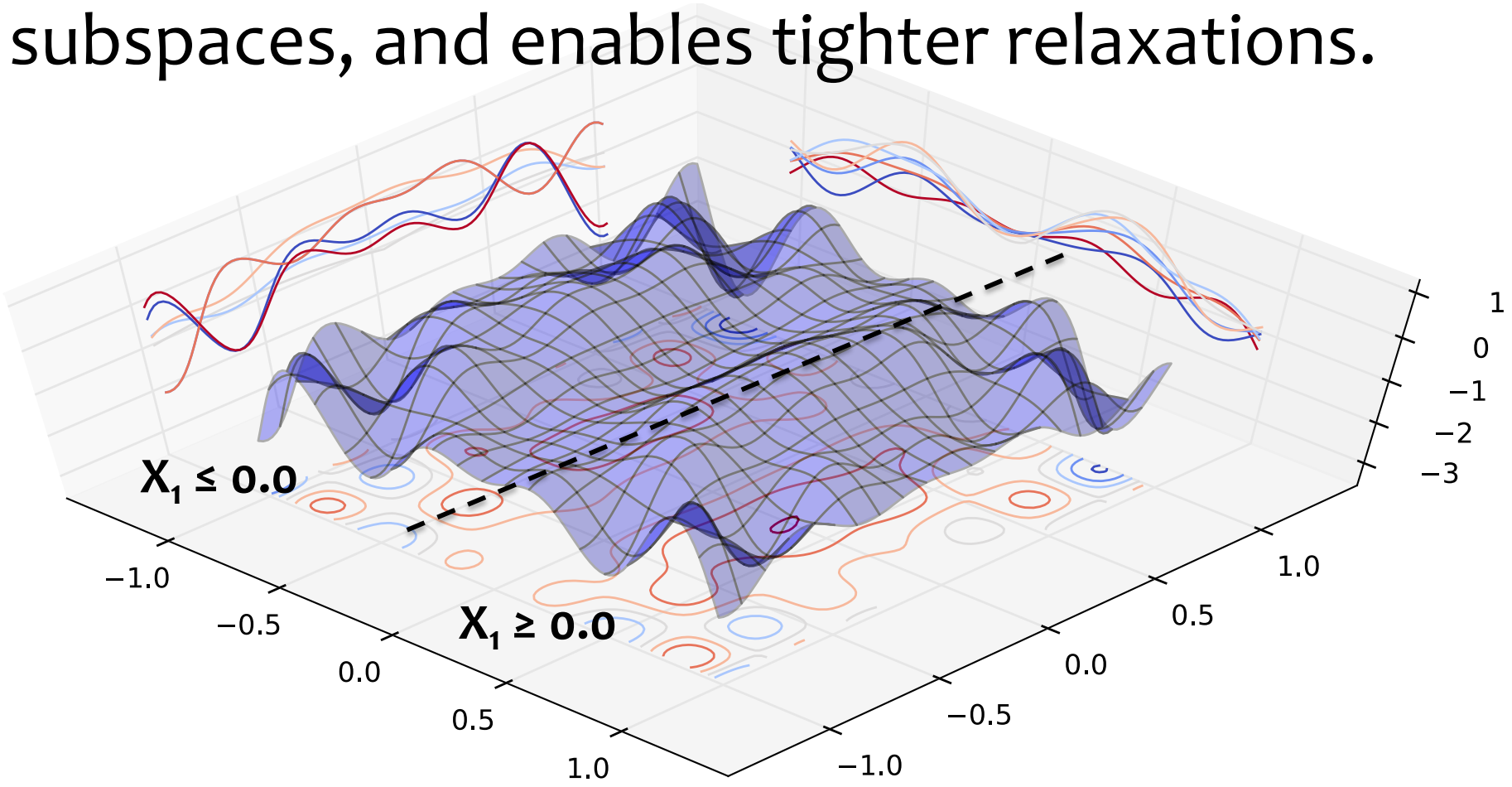
Background: Nonconvex Global Optimization

Relaxation: provides an upper bound on the surface.



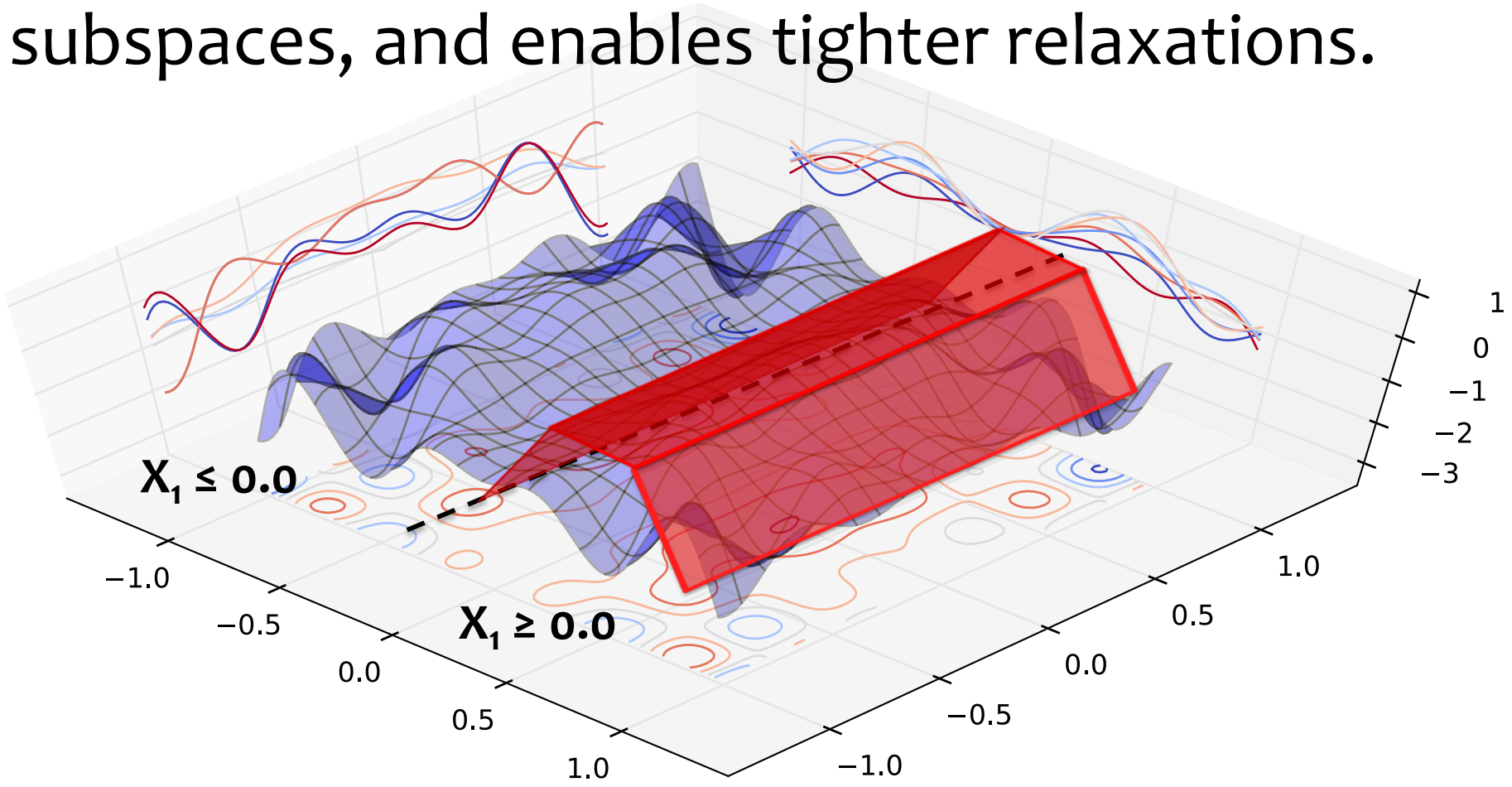
Background: Nonconvex Global Optimization

Branching: partitions the search space into subspaces, and enables tighter relaxations.



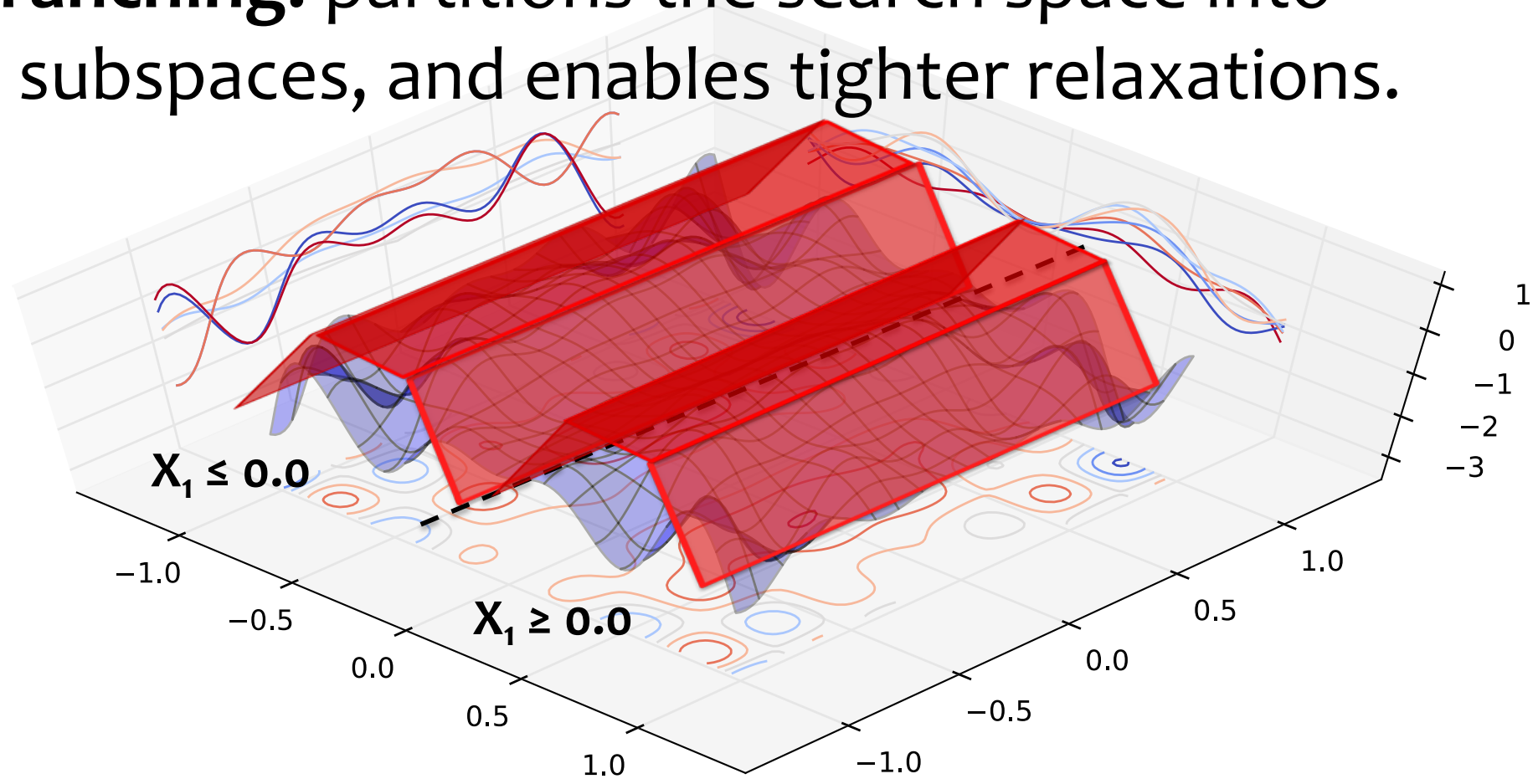
Background: Nonconvex Global Optimization

Branching: partitions the search space into subspaces, and enables tighter relaxations.



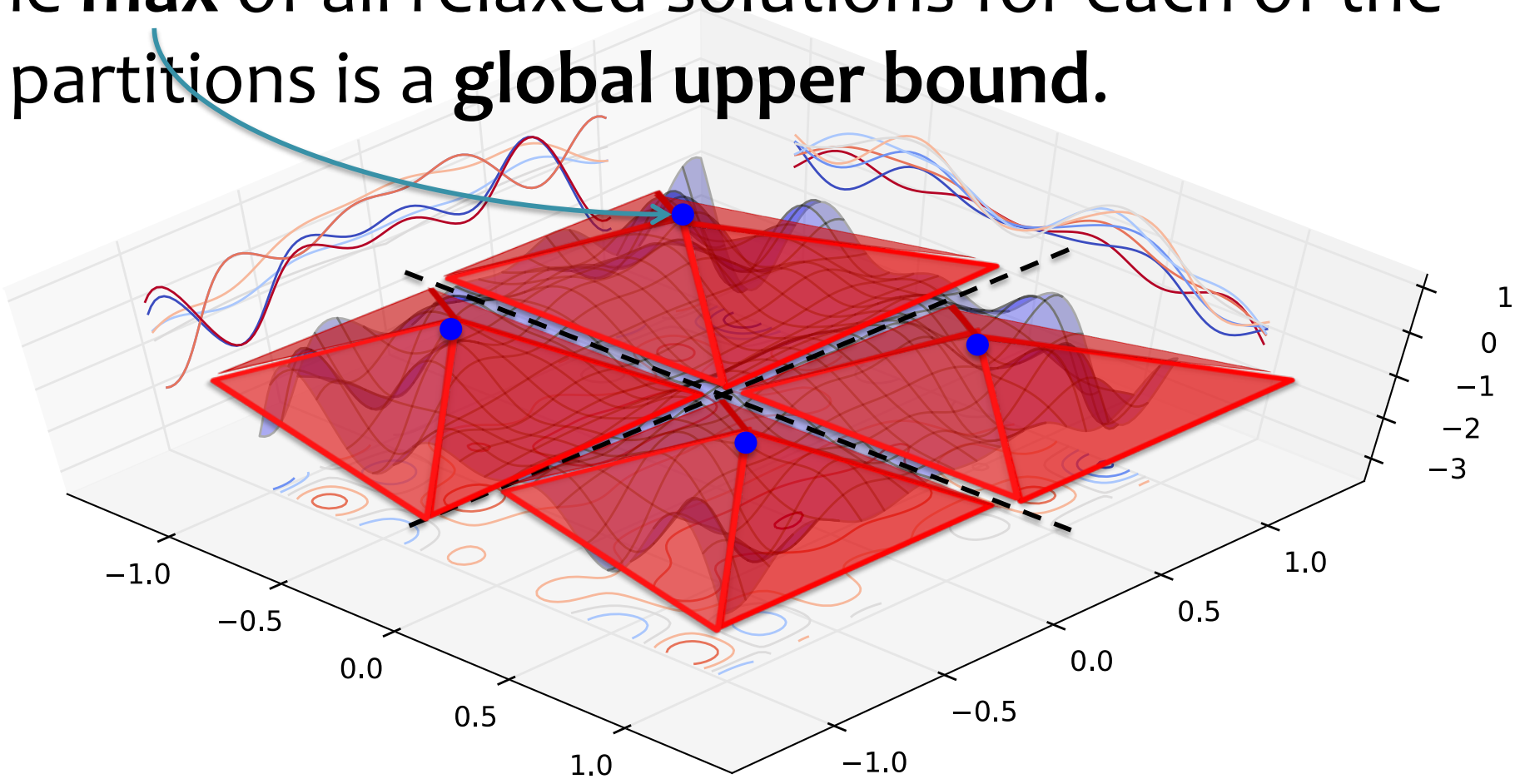
Background: Nonconvex Global Optimization

Branching: partitions the search space into subspaces, and enables tighter relaxations.



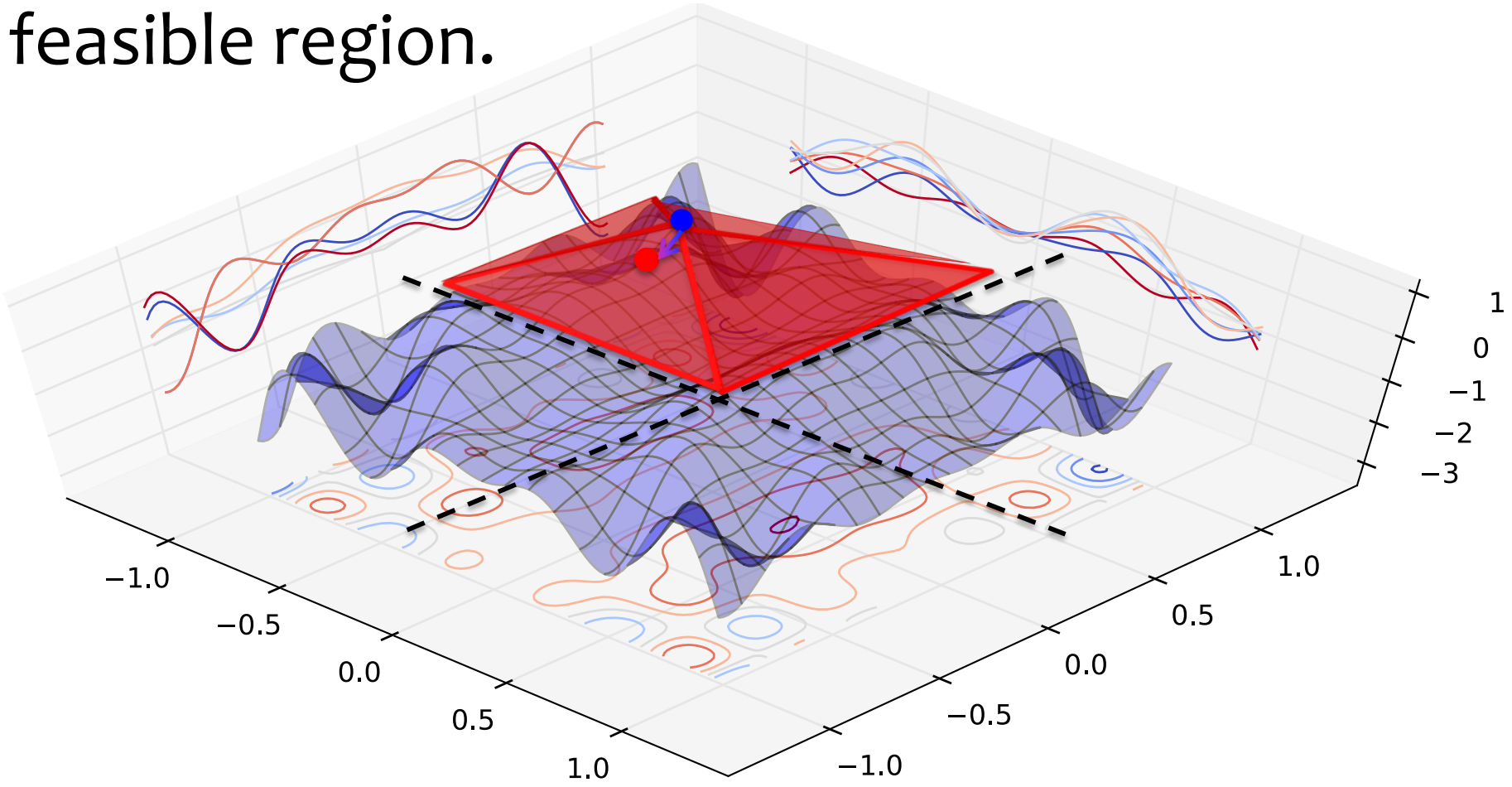
Background: Nonconvex Global Optimization

The **max** of all relaxed solutions for each of the partitions is a **global upper bound.**



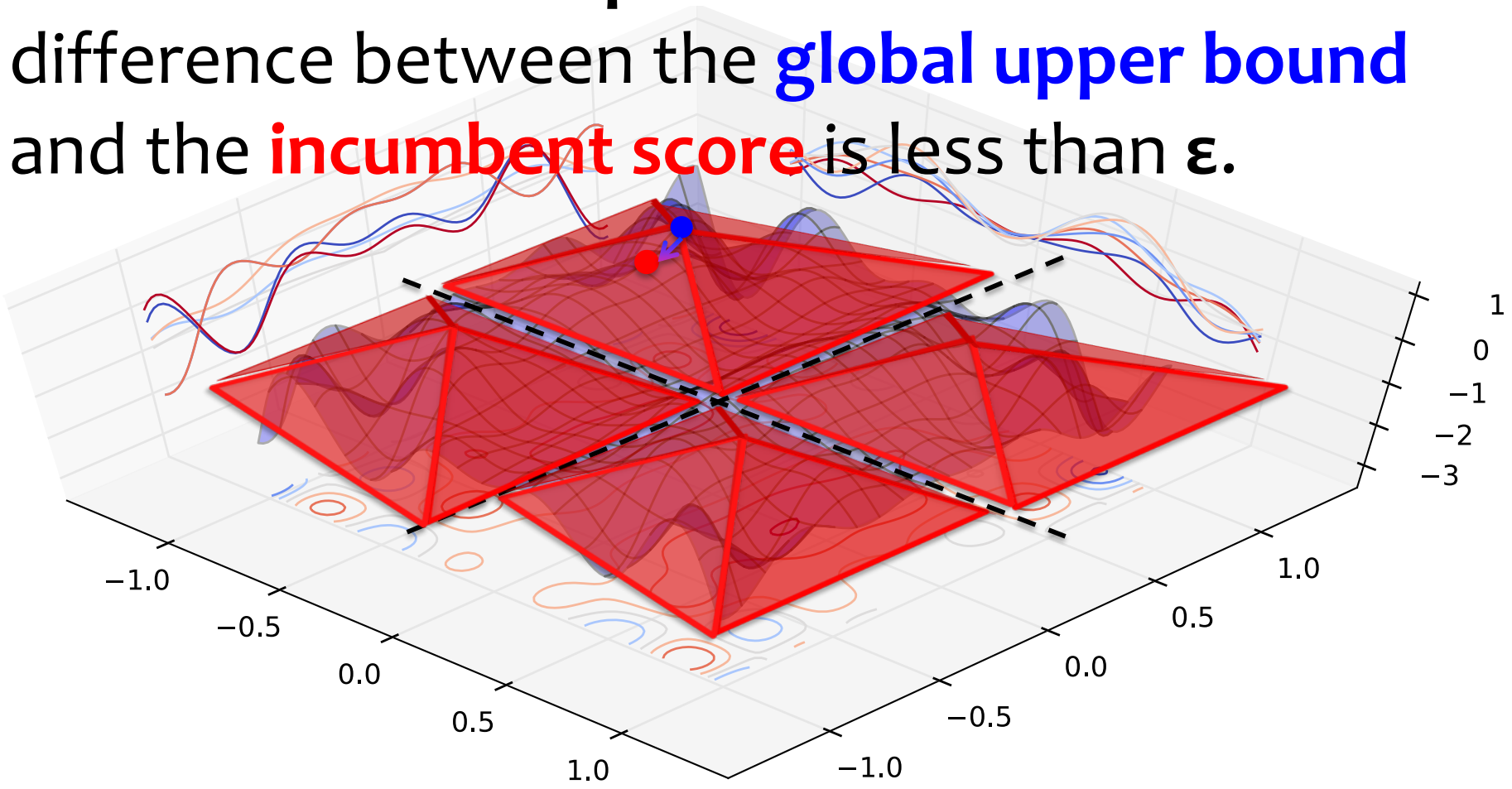
Background: Nonconvex Global Optimization

We can **project** a relaxed solution onto the feasible region.



Background: Nonconvex Global Optimization

The **incumbent** is **ε-optimal** if the relative difference between the **global upper bound** and the **incumbent score** is less than **ε**.



How much should we subdivide?

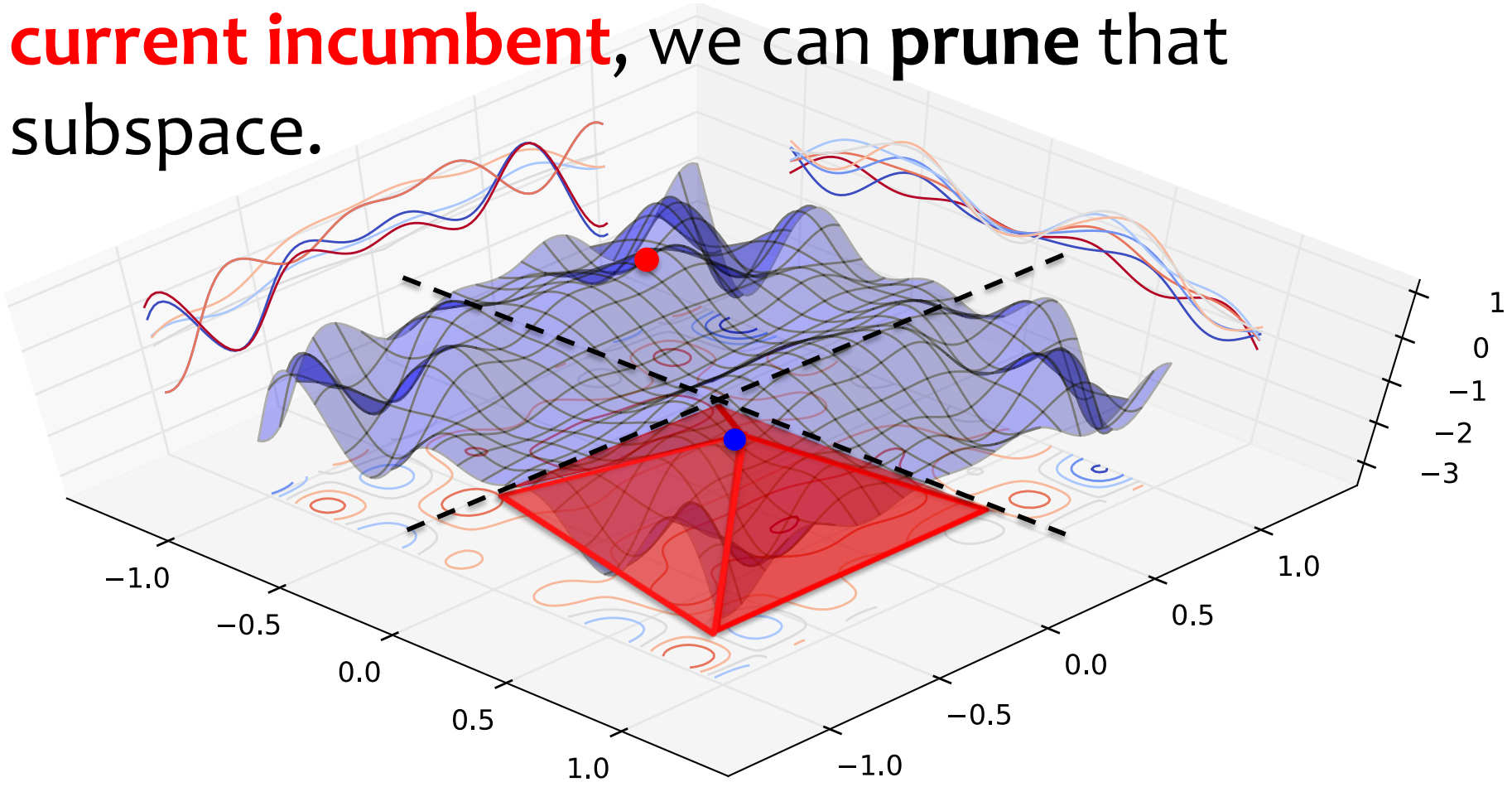
How much should we subdivide?

BRANCH-AND-BOUND

- Method for **recursively subdividing** the search space
- **Subspace order** can be determined heuristically (e.g. best-first search with depth-first plunging)
- **Prunes** subspaces that can't yield better solutions

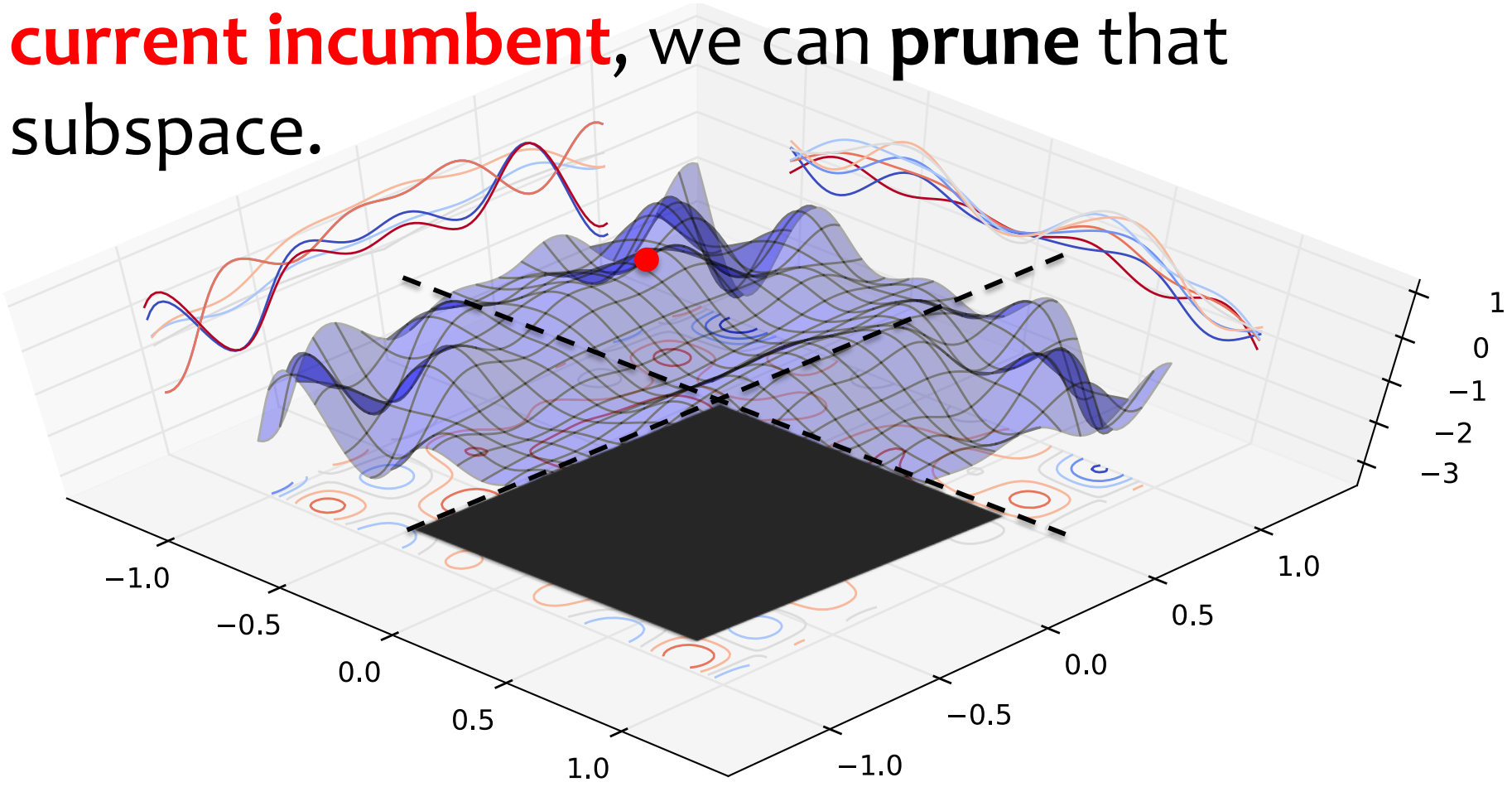
Background: Nonconvex Global Optimization

If the **subspace upper bound** is worse than the **current incumbent**, we can **prune** that subspace.



Background: Nonconvex Global Optimization

If the **subspace upper bound** is worse than the **current incumbent**, we can **prune** that subspace.



Limitations:
Branch-and-Bound

Branch-and-bound

- Kind of tricky to get it right…
  - Lots of hyperparameters to tune
- Curse of dimensionality kicks in quickly
- Applications to problems other than ILP (e.g. QP) rather unsuccessful
  - Nonconvex quadratic optimization by LP-based branch-and-bound usually fails with more than 80 variables (Burer and Vandenbussche, 2009)

BRANCH-AND-BOUND INGREDIENTS

Mathematical Program

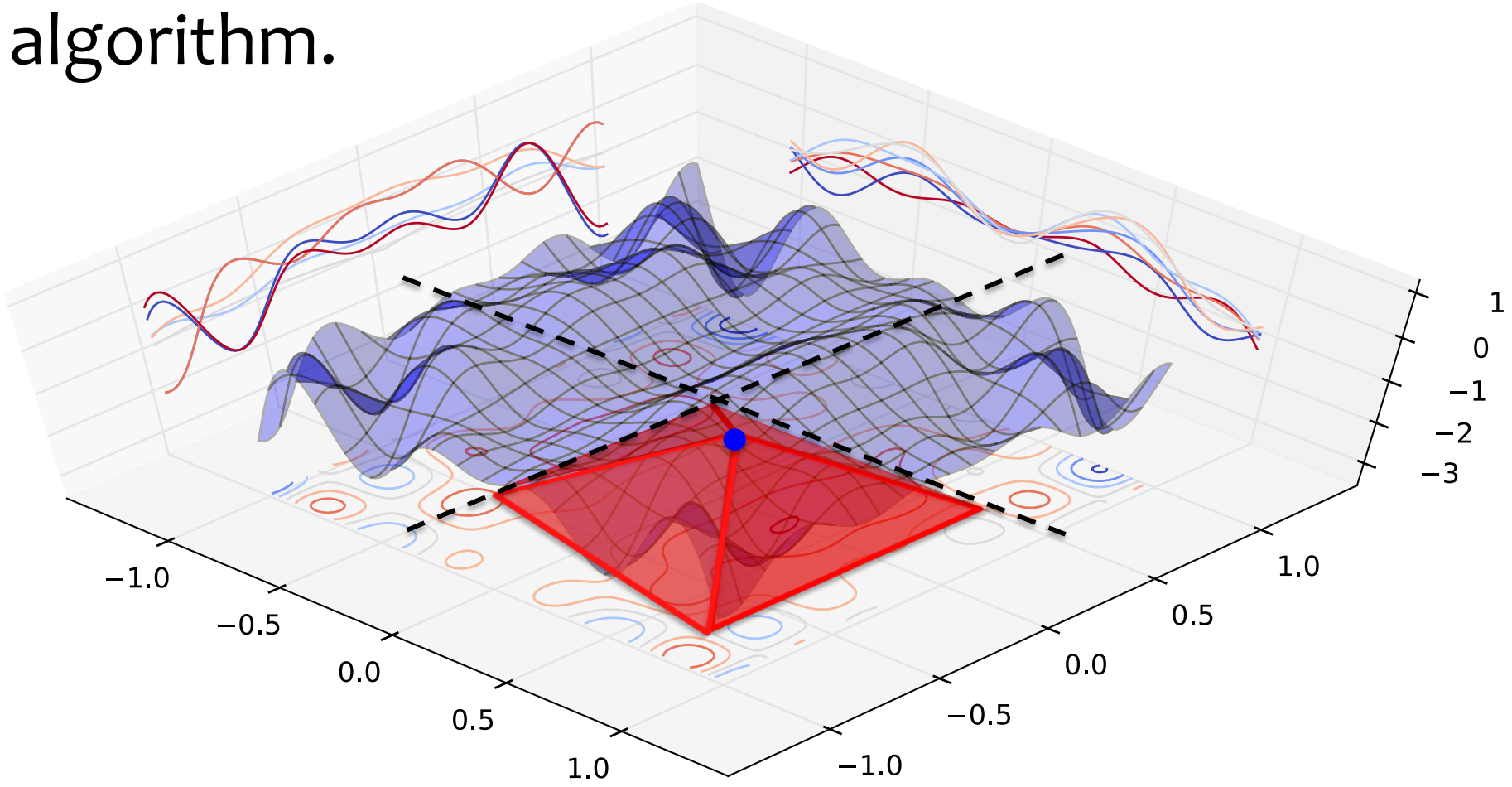
Relaxation

Projection

(Branch-and-Bound Search Heuristics)

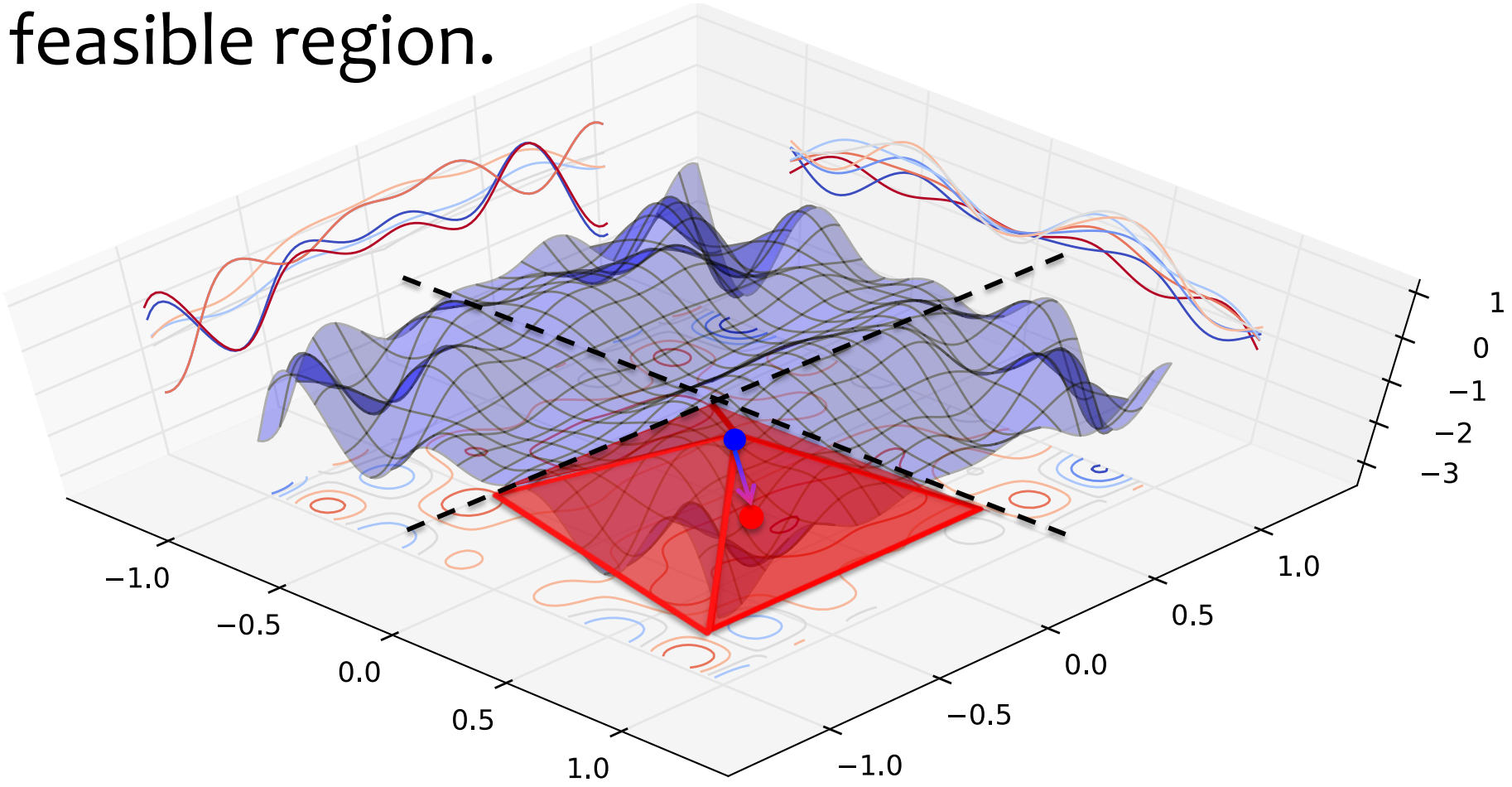
Background: Nonconvex Global Optimization

We solve the **relaxation** using the Simplex algorithm.



Background: Nonconvex Global Optimization

We can **project** a relaxed solution onto the feasible region.



Integer Linear Programming

Whiteboard

– Branch and bound for an ILP in 2D

Branch and Bound

Algorithm 2.1 Branch-and-bound

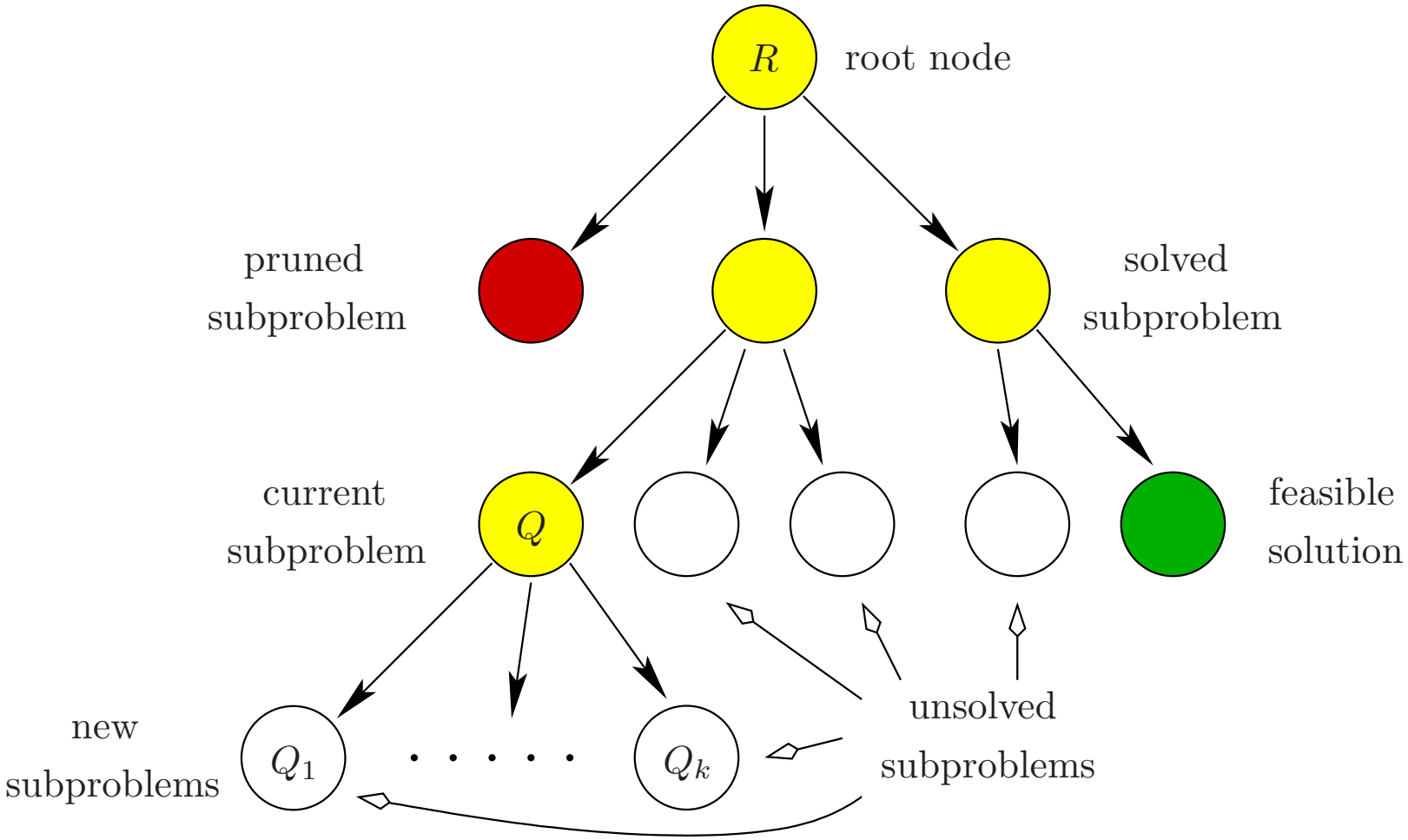
Input: Minimization problem instance R.

Output: Optimal solution $x^\star$ with value $c^\star$, or conclusion that R has no solution, indicated by $c^\star = \infty$.

1. Initialize $\mathcal{L} := \{R\}$, $\hat{c} := \infty$. [*init*]
2. If $\mathcal{L} = \emptyset$, stop and return $x^\star = \hat{x}$ and $c^\star = \hat{c}$. [*abort*]
3. Choose $Q \in \mathcal{L}$, and set $\mathcal{L} := \mathcal{L} \setminus \{Q\}$. [*select*]
4. Solve a relaxation Q_{relax} of Q. If Q_{relax} is empty, set $\check{c} := \infty$. Otherwise, let $\check{x}$ be an optimal solution of Q_{relax} and $\check{c}$ its objective value. [*solve*]
5. If $\check{c} \geq \hat{c}$, goto Step 2. [*bound*]
6. If $\check{x}$ is feasible for R, set $\hat{x} := \check{x}$, $\hat{c} := \check{c}$, and goto Step 2. [*check*]
7. Split Q into subproblems $Q = Q_1 \cup \ldots \cup Q_k$, set $\mathcal{L} := \mathcal{L} \cup \{Q_1, \ldots, Q_k\}$, and goto Step 2. [*branch*]

Slide from Achterberg (thesis, 2007)

Branch and Bound



Slide from Achterberg (thesis, 2007)

Branch and Bound

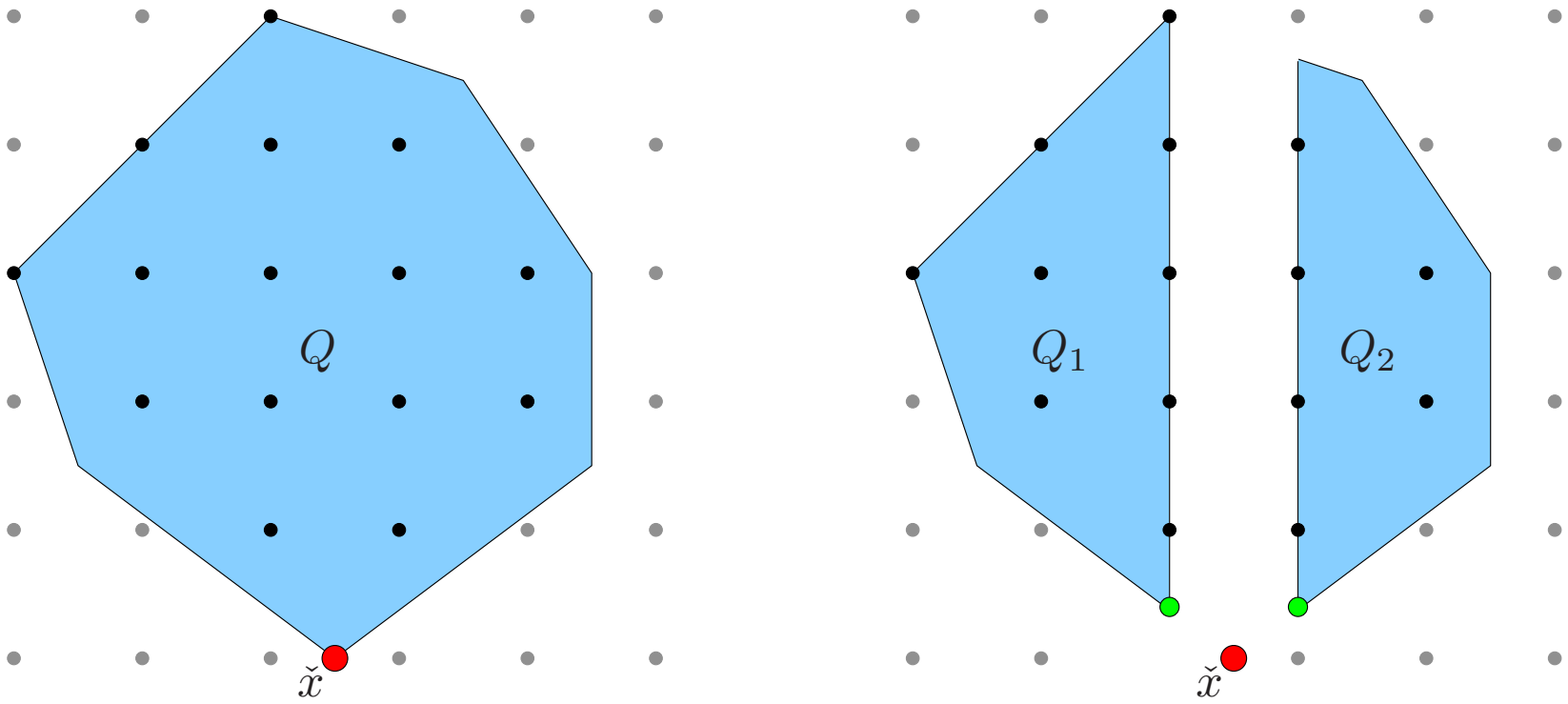


Figure 2.2. LP based branching on a single fractional variable.

Slide from Achterberg (thesis, 2007)

Case #1: Binary Variables

MAP INFERENCE AS MATHEMATICAL PROGRAMMING

Exact Inference

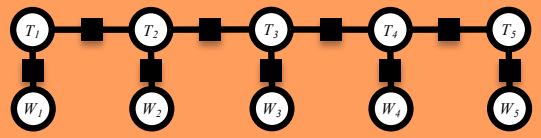
Recall...

1. Data

$$\mathcal{D} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$$

2. Model

$$p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

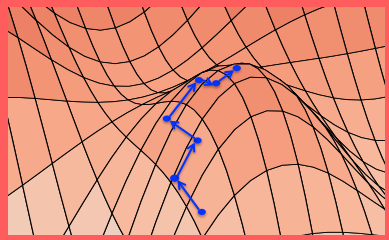


3. Objective

$$\ell(\theta; \mathcal{D}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}^{(n)} \mid \boldsymbol{\theta})$$

4. Learning

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \ell(\boldsymbol{\theta}; \mathcal{D})$$



5. Inference

1. Marginal Inference

$$p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

2. Partition Function

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

3. MAP Inference

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmax}}\; p(\boldsymbol{x} \mid \boldsymbol{\theta})$$

Recall...

5. Inference

Three Tasks:

1. Marginal Inference

Compute marginals of variables and cliques

$$p(x_i) = \sum_{\boldsymbol{x}' : x'_i = x_i} p(\boldsymbol{x}' \mid \boldsymbol{\theta}) \qquad p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

2. Partition Function

Compute the normalization constant

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

3. MAP Inference (NP-Hard in the general case)

Compute variable assignment with highest probability

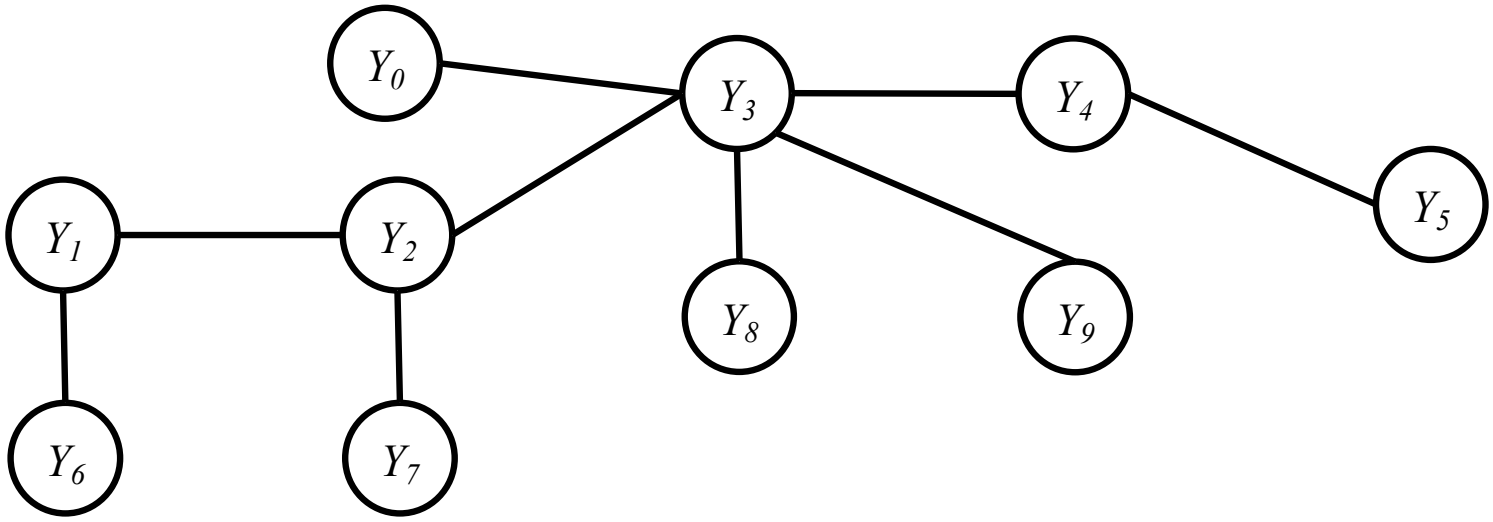
$$\hat{\boldsymbol{x}} = \operatorname*{argmax}_{\boldsymbol{x}} \; p(\boldsymbol{x} \mid \boldsymbol{\theta})$$

MAP Inference

Suppose we want to predict the highest likelihood structure y, given observations x and parameters w.

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \log p_w(y|x)$$

$$= \underset{\mathbf{y}}{\operatorname{argmax}} \sum_j \mathbf{w}^T f_{\text{node}}(x_j, y_j) + \sum_{j,k} \mathbf{w}^T f_{\text{edge}}(\mathbf{x}_{jk}, y_j, y_k)$$



MAP Inference

Suppose we want to predict the highest likelihood structure y, given observations x and parameters w.

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \log p_w(y|x)$$

$$= \underset{\mathbf{y}}{\operatorname{argmax}} \sum_j \mathbf{w}^T f_{\text{node}}(x_j, y_j) + \sum_{j,k} \mathbf{w}^T f_{\text{edge}}(\mathbf{x}_{jk}, y_j, y_k)$$

Idea:

1. Reformulate the problem as an integer linear program (ILP) – note that this is just going to be a new way of writing down the problem: y → z
2. Then remove the integer constraints (i.e. solve the linear program (LP) relaxation)

Lemma: (Wainwright et al., 2002) If there is a unique MAP assignment, the LP relaxation of the ILP above is guaranteed to have an integer solution, which is exactly the MAP solution!

Integer Linear Programming

Whiteboard

- MAP Inference for a Binary Pairwise MRF as an ILP