Randomized Rounding

Jannis Beese

Ferienakademie im Sarntal 2012
FAU Erlangen-Nürnberg, TU München, Uni Stuttgart

September 2012

Overview

Overview

Why random/approximation?

- Solve hard problems well enough.

Why random/approximation?

- Solve hard problems well enough.
- Using only randomness is not enough: (BPP = P)?

Why random/approximation?

- Solve hard problems well enough.
- Using only randomness is not enough: (BPP = P)?
- Error bounds in deterministic approximation are often bad or hard to estimate.

Goals

For all instances of the problem:

- Fast.

Goals

For all instances of the problem:

- Fast.
- With high probability: Close to an optimal solution.

Performance Ratio of a Randomized Algorithm

Let x be an instance of the Problem P. Let A be an algorithm for solving P, let $m_A(x)$ denote the size of the solution produced by A on x and let $m^*(x)$ denote the size of an optimal solution. The performance ratio is given by

$$\max\{\sup_{x\in P} \frac{m^*(x)}{m_A(x)}, \inf_{x\in P} \frac{m_A(x)}{m^*(x)}\}. \tag{1}$$

Performance Ratio of a Randomized Algorithm

Let x be an instance of the Problem P. Let A be an algorithm for solving P, let $m_A(x)$ denote the size of the solution produced by A on x and let $m^*(x)$ denote the size of an optimal solution. The performance ratio is given by

$$\max\{\sup_{x \in P} \frac{m^*(x)}{m_A(x)}, \inf_{x \in P} \frac{m_A(x)}{m^*(x)}\}. \tag{1}$$

If A uses randomization: $\mathbb{E}[m_A(x)]$

Overview

Boole's inequality

Theorem (Boole's inequality)

Let (A_i) be any countable set of events. Then

$$\mathbb{P}\left[\bigcup_i A_i\right] \leq \sum_i \mathbb{P}\left[A_i\right]. \tag{2}$$

Chernoff-Bound variant

Theorem (Chernoff-Bound variant)

Let $X_1, \ldots, X_n$ be a sequence of independent Bernoulli trials, such that $\mathbb{P}[X_i = 1] = p_i$ and $\mathbb{P}[X_i = 0] = 1 - p_i$. Let S be a subset of $\{1, \ldots, n\}$ and let $s = |S|$. Define $X = \sum_{i \in S} X_i$. Then

$$\mathbb{P}\left[|X - \mathbb{E}[X]| > \sqrt{4s \ln s}\right] \leq \frac{1}{s^2}. \qquad (3)$$

□

Linear Programming

A Linear Program is a problem of the following type:

$$
\begin{array}{rr}
\text{Maximize} & \mathbf{c}^T\mathbf{x} \\
\text{constrained by} & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\
\text{for} & \mathbf{c}, \mathbf{b} \in \mathbb{Q}^n, \mathbf{A} \in \mathbb{Q}^{n \times n}
\end{array}
$$

Linear Programming

A Linear Program is a problem of the following type:

$$\begin{array}{rr} \text{Maximize} & \mathbf{c}^T\mathbf{x} \\ \text{constrained by} & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ \text{for} & \mathbf{c}, \mathbf{b} \in \mathbb{Q}^n, \mathbf{A} \in \mathbb{Q}^{n \times n} \end{array}$$

While Linear Programs are efficiently solvable, Integer Linear Programming (ILP) is known to be **NP** complete.

Overview

1. Rewrite the problem as an Integer Linear Program.

Randomized Rounding

1. Rewrite the problem as an Integer Linear Program.
2. Relax the integrality requirement and compute an optimal solution to the corresponding LP.

Randomized Rounding

1. Rewrite the problem as an Integer Linear Program.
2. Relax the integrality requirement and compute an optimal solution to the corresponding LP.
3. Approximate the optimal LP-solution to obtain an integer-approximation of the ILP-solution.

Randomized Rounding: Analysis

Let $P = (\mathbf{A}, \mathbf{c}, \mathbf{b})$ be an ILP, let $\mathbf{x} = (x_1, \ldots, x_n)$ denote the variable vector of P and let $\widehat{\mathbf{x}} = (\widehat{x_1}, \ldots, \widehat{x_n})$ denote a solution to the corresponding Linear Program.

Randomized Rounding: Analysis

Let $P = (\mathbf{A}, \mathbf{c}, \mathbf{b})$ be an ILP, let $\mathbf{x} = (x_1, \ldots, x_n)$ denote the variable vector of P and let $\widehat{\mathbf{x}} = (\widehat{x_1}, \ldots, \widehat{x_n})$ denote a solution to the corresponding Linear Program.
For each $i = 1, \ldots, n$ define $\overline{x_i}$ equal to 1 with probability $\widehat{x_i}$ and equal to 0 otherwise. Let $\overline{\mathbf{x}} = (\overline{x_1}, \ldots, \overline{x_n})$ be the corresponding vector of random variables.

Randomized Rounding: Analysis

Let $P = (\mathbf{A}, \mathbf{c}, \mathbf{b})$ be an ILP, let $\mathbf{x} = (x_1, \ldots, x_n)$ denote the variable vector of P and let $\widehat{\mathbf{x}} = (\widehat{x_1}, \ldots, \widehat{x_n})$ denote a solution to the corresponding Linear Program.
For each $i = 1, \ldots, n$ define $\overline{x_i}$ equal to 1 with probability $\widehat{x_i}$ and equal to 0 otherwise. Let $\overline{\mathbf{x}} = (\overline{x_1}, \ldots, \overline{x_n})$ be the corresponding vector of random variables.
Then for any row vector $\mathbf{a}$ of $\mathbf{A}$

$$\mathbb{E}[\mathbf{a} \cdot \overline{\mathbf{x}}] = \mathbf{a} \cdot \widehat{\mathbf{x}}.$$

Overview

Lattice Approximation: Problem Statement

Let $\mathbf{A} \in \mathbb{B}^{n \times n} = \{0, 1\}^{n \times n}$ be a $n \times n$ matrix with entries in $\{0, 1\}$ and $\mathbf{p} \in [0, 1]^n$ a vector with entries in the real interval $[0, 1]$.

Lattice Approximation: Problem Statement

Let $\mathbf{A} \in \mathbb{B}^{n \times n} = \{0, 1\}^{n \times n}$ be a $n \times n$ matrix with entries in $\{0, 1\}$ and $\mathbf{p} \in [0, 1]^n$ a vector with entries in the real interval $[0, 1]$.
The goal is to find a 0-1-vector $\mathbf{q}$ such that $\|\mathbf{A} \cdot (\mathbf{p} - \mathbf{q})\|_\infty$ is minimal.

Lattice Approximation: Problem Statement

Let $\mathbf{A} \in \mathbb{B}^{n \times n} = \{0, 1\}^{n \times n}$ be a $n \times n$ matrix with entries in $\{0, 1\}$ and $\mathbf{p} \in [0, 1]^n$ a vector with entries in the real interval $[0, 1]$.
The goal is to find a 0-1-vector $\mathbf{q}$ such that $\|\mathbf{A} \cdot (\mathbf{p} - \mathbf{q})\|_\infty$ is minimal.
This is already stated as LP!

Lattice Approximation: Solution step 2

$$\min_{\mathbf{q}\in\mathbb{F}_2^n} \|\mathbf{A}\cdot(\mathbf{p}-\mathbf{q})\|_\infty$$

- Step 2: Solve the non-integral version of the problem.

Lattice Approximation: Solution step 2

$$\min_{\mathbf{q} \in \mathbb{F}_2^n} \|\mathbf{A} \cdot (\mathbf{p} - \mathbf{q})\|_\infty$$

- Step 2: Solve the non-integral version of the problem.
- Simply choose $\mathbf{q} = \mathbf{p}$.

Lattice Approximation: Solution step 3

$$\min_{\mathbf{q} \in \mathbb{F}_2^n} \|\mathbf{A} \cdot (\mathbf{p} - \mathbf{q})\|_\infty$$

- Choose $\overline{\mathbf{q}}$ with $\overline{q_i} = 1$ with probability p_i as solution.

Lattice Approximation: Solution step 3

$$\min_{\mathbf{q}\in\mathbb{F}_2^n} \|\mathbf{A}\cdot(\mathbf{p}-\mathbf{q})\|_\infty$$

- Choose $\overline{\mathbf{q}}$ with $\overline{q_i} = 1$ with probability p_i as solution.
- $\mathbb{E}[\mathbf{A_i}\cdot\overline{\mathbf{q}}] = \mathbf{A_i}\cdot\mathbf{p}$

Lattice Approximation: Solution step 3

$$\min_{\mathbf{q}\in\mathbb{F}_2^n} \|\mathbf{A}\cdot(\mathbf{p}-\mathbf{q})\|_\infty$$

- Choose $\overline{\mathbf{q}}$ with $\overline{q_i} = 1$ with probability p_i as solution.
- $\mathbb{E}[\mathbf{A_i}\cdot\overline{\mathbf{q}}] = \mathbf{A_i}\cdot\mathbf{p}$
- Chernoff:

$$\mathbb{P}\left[|\mathbf{A_i}\cdot\overline{\mathbf{q}} - \mathbf{A_i}\cdot\mathbf{p}| \geq \sqrt{4n\ln n}\right] \leq \frac{1}{n^2}.$$

Lattice Approximation: Solution analysis

$$
\begin{aligned}
\mathbb{P}\left[\|\mathbf{A}\cdot(\overline{\mathbf{q}}-\mathbf{p})\|_\infty > \sqrt{4n\ln n}\right] &= \mathbb{P}\left[\bigcup_i |\mathbf{A_i}\cdot\mathbf{q}-\mathbf{A_i}\cdot\mathbf{p}| > \sqrt{4n\ln n}\right] \\
&\le \sum_i \mathbb{P}\left[|\mathbf{A_i}\cdot\overline{\mathbf{q}}-\mathbf{A_i}\cdot\mathbf{p}| > \sqrt{4n\ln n}\right] \\
&\le \frac{1}{n}.
\end{aligned}
$$

Lattice Approximation: Solution analysis

$$\begin{aligned}\mathbb{P}\left[\|\mathbf{A}\cdot(\overline{\mathbf{q}}-\mathbf{p})\|_\infty > \sqrt{4n\ln n}\right] &= \mathbb{P}\left[\bigcup_i |\mathbf{A_i}\cdot\mathbf{q}-\mathbf{A_i}\cdot\mathbf{p}| > \sqrt{4n\ln n}\right]\\ &\leq \sum_i \mathbb{P}\left[|\mathbf{A_i}\cdot\overline{\mathbf{q}}-\mathbf{A_i}\cdot\mathbf{p}| > \sqrt{4n\ln n}\right]\\ &\leq \frac{1}{n}.\end{aligned}$$

Thus with probability $1-\frac{1}{n}$ we find a solution $\overline{\mathbf{q}}$ with $\|\mathbf{A}\cdot(\overline{\mathbf{q}}-\mathbf{p})\|_\infty \leq \sqrt{4n\ln n}$.

Overview

MaxSAT: Problem Statement

Let I be an instance of SAT, which without loss of generality is assumed to be in conjunctive normal form (CNF), i.e. a set C of clauses in CNF over a set of boolean variables V.

MaxSAT: Problem Statement

Let I be an instance of SAT, which without loss of generality is assumed to be in conjunctive normal form (CNF), i.e. a set C of clauses in CNF over a set of boolean variables V.
A solution to this problem is an assignment $A : V \to \mathbb{B}$, such that the number of clauses $m_A(I)$ that evaluate to true is maximal.

MaxSAT: Solution Step 1

- Clause $C_j \rightarrow z_j$ with $z_j = 1$, iff C_j evaluates to true.

MaxSAT: Solution Step 1

- Clause $C_j \rightarrow z_j$ with $z_j = 1$, iff C_j evaluates to true.
- Variable $x_i \rightarrow y_i$ with $y_i = 1$, iff x_i is set to true.

MaxSAT: Solution Step 1

- Clause $C_j \rightarrow z_j$ with $z_j = 1$, iff C_j evaluates to true.
- Variable $x_i \rightarrow y_i$ with $y_i = 1$, iff x_i is set to true.
- S_j^+: Set of variables that appear in C_j. S_j^- analoguous.

MaxSAT: Solution Step 1

- Clause $C_j \rightarrow z_j$ with $z_j = 1$, iff C_j evaluates to true.
- Variable $x_i \rightarrow y_i$ with $y_i = 1$, iff x_i is set to true.
- S_j^+: Set of variables that appear in C_j. S_j^- analoguous.

$$
\begin{aligned}
\text{Maximize} \quad & \sum_j z_j \\
\text{constrained by} \quad & \sum_{y_i \in S_j^+} y_i + \sum_{y_i \in S_j^-} (1 - y_i) \geq z_j \forall j \\
\text{and constrained by} \quad & y_i, z_j \in \{0, 1\}.
\end{aligned}
$$

MaxSAT: Solution Step 2+3

We need two algorithms:

MaxSAT: Solution Step 2+3

We need two algorithms:

1. Randomized Rounding. Works well for clauses with few literals:

Lemma

Let C_j be a clause with k literals. Then for the probability p_j that C_j evaluates to true, the following holds:

$$p_j \geq (1 - (1 - \frac{1}{k})^k)\widehat{z}_j.$$

MaxSAT: Solution Step 2+3

We need two algorithms:

1. Randomized Rounding. Works well for clauses with few literals:

Lemma

Let C_j be a clause with k literals. Then for the probability p_j that C_j evaluates to true, the following holds:

$$p_j \geq (1 - (1 - \frac{1}{k})^k)\hat{z}_j.$$

2. Assign truth values with equal probability. Works well for large clauses:

$$p_j \geq (1 - \frac{1}{2^k}).$$

MaxSAT: Solution Analysis

Have:

- Randomized Rounding for small clauses.

$$p_j \geq (1 - (1 - \frac{1}{k})^k)\widehat{z}_j.$$

- Random truth assignment for large clauses.

$$p_j \geq (1 - \frac{1}{2^k}).$$

MaxSAT: Solution Analysis

Have:

- Randomized Rounding for small clauses.

$$p_j \geq (1 - (1 - \frac{1}{k})^k)\widehat{z_j}.$$

- Random truth assignment for large clauses.

$$p_j \geq (1 - \frac{1}{2^k}).$$

Simply running both algorithms and choosing the better output already leads to a $\frac{4}{3}$-approximation.

k	$(1 - \frac{1}{2^k})$	$(1 - (1 - \frac{1}{k})^k)$
1	$\frac{1}{2}$	1
2	$\frac{3}{4}$	$\frac{3}{4}$
≥ 3	$\geq \frac{7}{8}$	$\geq 1 - \frac{1}{e} \geq \frac{5}{8}$

k	$(1-\frac{1}{2^k})$	$(1-(1-\frac{1}{k})^k)$
1	$\frac{1}{2}$	1
2	$\frac{3}{4}$	$\frac{3}{4}$
≥ 3	$\geq \frac{7}{8}$	$\geq 1-\frac{1}{e} \geq \frac{5}{8}$

$$\max\{m_1, m_2\} \geq \frac{1}{2}(m_1 + m_2) \tag{4}$$

$$\geq \sum_{k\geq 1} \sum_{c_j \in C_k} \frac{\alpha_k + \beta_k}{2} \hat{z}_j \tag{5}$$

$$\geq \frac{3}{4} \cdot OPT \tag{6}$$

Overview

Semidefinite Programming(1)

$$\min_{x^1,\ldots,x^n \in \mathbb{R}^n} \sum_{i,j=1,\ldots,n} c_{i,j}(x^i \cdot x^j)$$

$$\text{constrained by} \sum_{i,j=1,\ldots,n} a_{i,j,k}(x^i \cdot x^j) \leq b_k \forall k.$$

Semidefinite Programming(1)

$$\min_{x^1,\dots,x^n \in \mathbb{R}^n} \quad \sum_{i,j=1,\dots,n} c_{i,j}(x^i \cdot x^j)$$

$$\text{constrained by} \quad \sum_{i,j=1,\dots,n} a_{i,j,k}(x^i \cdot x^j) \leq b_k \forall k.$$

- Uses dot products of vectors.

Semidefinite Programming(1)

$$\min_{x^1,\dots,x^n \in \mathbb{R}^n} \quad \sum_{i,j=1,\dots,n} c_{i,j}(x^i \cdot x^j)$$

$$\text{constrained by} \quad \sum_{i,j=1,\dots,n} a_{i,j,k}(x^i \cdot x^j) \leq b_k \forall k.$$

- Uses dot products of vectors.
- Called "Semidefinite" as this may also be defined through positive-semidefinite matrices.

Semidefinite Programming(2)

$$\min_{x^1,\ldots,x^n \in \mathbb{R}^n} \quad \sum_{i,j=1,\ldots,n} c_{i,j}(x^i \cdot x^j)$$

$$\text{constrained by} \quad \sum_{i,j=1,\ldots,n} a_{i,j,k}(x^i \cdot x^j) \leq b_k \forall k.$$

- There exist efficient algorithms for solving SDPs (up to an arbitrarily small error).

Semidefinite Programming(2)

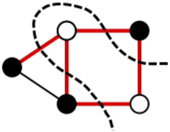
$$\min_{x^1,\ldots,x^n \in \mathbb{R}^n} \quad \sum_{i,j=1,\ldots,n} c_{i,j}(x^i \cdot x^j)$$

$$\text{constrained by} \quad \sum_{i,j=1,\ldots,n} a_{i,j,k}(x^i \cdot x^j) \leq b_k \forall k.$$

- There exist efficient algorithms for solving SDPs (up to an arbitrarily small error).
- Many problems may be stated as a SDP.

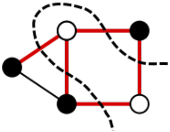
Overview

Maximum Weighted Cut: Problem Statement



Let $G = (V, E)$ be an undirected Graph and let $w : E \to \mathbb{N}$. The goal is to find a Partition $\Pi = (V_1, V_2)$ of V, such that sum of the weights of the edges from V_1 to V_2 is maximized.

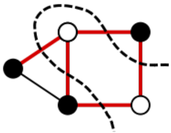
Maximum Weighted Cut: Problem Statement



Let $G = (V, E)$ be an undirected Graph and let $w : E \to \mathbb{N}$. The goal is to find a Partition $\Pi = (V_1, V_2)$ of V, such that sum of the weights of the edges from V_1 to V_2 is maximized.

- MAX-CUT is already NP-Hard.

Maximum Weighted Cut: Problem Statement



Let $G = (V, E)$ be an undirected Graph and let $w : E \to \mathbb{N}$. The goal is to find a Partition $\Pi = (V_1, V_2)$ of V, such that sum of the weights of the edges from V_1 to V_2 is maximized.

- MAX-CUT is already NP-Hard.
- Assuming the Unique Games Conjecture, the following algorithm (for MAX-WCUT) is optimal.

Maximum Weighted Cut: Algorithm Outline

1. Express the problem as a Quadratic Integer Program IQP-CUT.

Maximum Weighted Cut: Algorithm Outline

1. Express the problem as a Quadratic Integer Program IQP-CUT.
2. Relax IQP-CUT into a Quadratic Programming Problem QP-CUT for analysis.

Maximum Weighted Cut: Algorithm Outline

1. Express the problem as a Quadratic Integer Program IQP-CUT.
2. Relax IQP-CUT into a Quadratic Programming Problem QP-CUT for analysis.
3. From QP-CUT derive an equivalent Semidefinite Programming Problem SD-CUT and solve this.

Maximum Weighted Cut: Algorithm Outline

1. Express the problem as a Quadratic Integer Program IQP-CUT.
2. Relax IQP-CUT into a Quadratic Programming Problem QP-CUT for analysis.
3. From QP-CUT derive an equivalent Semidefinite Programming Problem SD-CUT and solve this.
4. Use the solution obtained in the previous step to approximate a solution of the original IQP-CUT.

Maximum Weighted Cut: Constructing IQP-CUT and QP-CUT

$$\text{Maximize} \qquad \frac{1}{2}\sum_{(i,j)\in E} w_{ij}(1 - y_i y_j)$$

$$\text{constrained by} \qquad y_i \in \{-1, 1\} \forall i.$$

Maximum Weighted Cut: Constructing IQP-CUT and QP-CUT

$$\begin{aligned} \text{Maximize} \quad & \frac{1}{2} \sum_{(i,j) \in E} w_{ij}(1 - y_i y_j) \\ \text{constrained by} \quad & y_i \in \{-1, 1\} \forall i. \end{aligned}$$

Partition by $V_1 = \{y_i | y_i = 1\}, V_2 = \{y_i | y_i = -1\}$.

Maximum Weighted Cut: Constructing IQP-CUT and QP-CUT

$$\text{Maximize} \quad \frac{1}{2}\sum_{(i,j)\in E} w_{ij}(1 - y_i y_j)$$

$$\text{constrained by} \quad y_i \in \{-1, 1\} \forall i.$$

Partition by $V_1 = \{y_i | y_i = 1\}$, $V_2 = \{y_i | y_i = -1\}$.
Let $\mathbf{y_i}$ denote a 2-dimensional vector on the unit sphere.

$$\text{Maximize} \quad \frac{1}{2}\sum_{(i,j)\in E} w_{ij}(1 - \mathbf{y_i} \cdot \mathbf{y_j}) \tag{7}$$

$$\text{constrained by} \quad \|y_i\| = 1. \tag{8}$$

Maximum Weighted Cut: Constructing IQP-CUT and QP-CUT

$$\text{Maximize} \quad \frac{1}{2}\sum_{(i,j)\in E} w_{ij}(1 - y_i y_j)$$

$$\text{constrained by} \quad y_i \in \{-1, 1\} \forall i.$$

Partition by $V_1 = \{y_i | y_i = 1\}$, $V_2 = \{y_i | y_i = -1\}$.
Let $\mathbf{y_i}$ denote a 2-dimensional vector on the unit sphere.

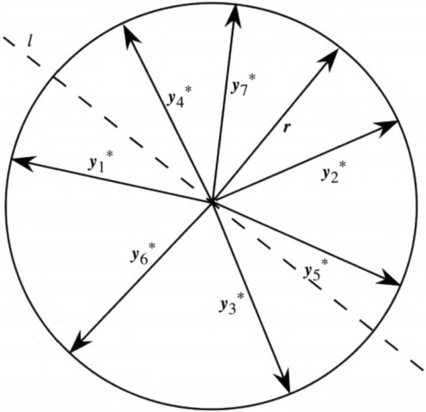
$$\text{Maximize} \quad \frac{1}{2}\sum_{(i,j)\in E} w_{ij}(1 - \mathbf{y_i} \cdot \mathbf{y_j}) \tag{7}$$

$$\text{constrained by} \quad \|y_i\| = 1. \tag{8}$$

Assume QP-CUT is efficiently solvable.

Maximum Weighted Cut: Randomized Rounding

Randomly choose a straight line through the origin and partition according to the side on which each vector lies on:



Maximum Weighted Cut: Analysis

- The same can be done in higher dimensions, leading to SD-CUT:

$$
\begin{aligned}
\text{Maximize} \quad & \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(1 - M_{i,j}) \\
\text{constrained by} \quad & M \text{ is positive semidefinite} \\
\text{and} \quad & M_{i,i} = 1.
\end{aligned}
$$

Maximum Weighted Cut: Analysis

- The same can be done in higher dimensions, leading to SD-CUT:

$$
\begin{aligned}
\text{Maximize} \quad & \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(1 - M_{i,j}) \\
\text{constrained by} \quad & M \text{ is positive semidefinite} \\
\text{and} \quad & M_{i,i} = 1.
\end{aligned}
$$

- This can be solved efficiently.

Maximum Weighted Cut: Analysis

- The same can be done in higher dimensions, leading to SD-CUT:

$$\begin{aligned} \text{Maximize} \quad & \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(1 - M_{i,j}) \\ \text{constrained by} \quad & M \text{ is positive semidefinite} \\ \text{and} \quad & M_{i,i} = 1. \end{aligned}$$

- This can be solved efficiently.
- Straight forward analysis shows that this is an $0.878\ldots-$approximation.

Summary

- Linear Programming and Randomized Rounding are applicable to a wide variety of problems.
- Algorithms are fairly easy to understand.
- Often leads to suprisingly efficient solutions.

Questions?

Any Questions?

Questions?

Any Questions?
Thank you for your attention!