The Cross-Entropy Method

Dirk P. Kroese *Reuven Y. Rubinstein

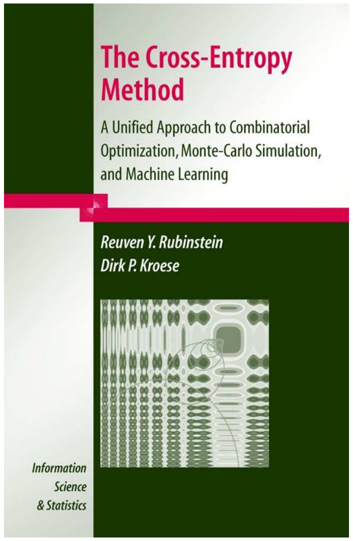
Department of Mathematics, The University of Queensland, Australia
*Faculty of Industrial Engineering and Management, Technion, Israel
and Stanford University (Visiting)

Contents

1. Introduction
2. CE: Methodology
3. CE for the Probability of Rare-Events Simulation (Light and Heavy-Tailed distributions)
4. Applications of Rare-Events: Calculating the Permanent and Counting the Number of 0-1 Tables.
5. CE for NP Combinatorial Optimization: the Max-Cut and TSP Problem.
6. Noisy (Simulation-Based) Optimization with CE
7. The Minimum Cross-Entropy (MCE) Method
8. Some Theory on CE and MCE
9. Conclusion

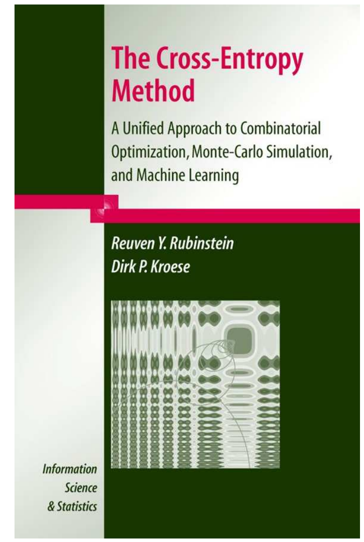
CE Matters

Book: R.Y. Rubinstein and D.P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte Carlo Simulation and Machine Learning*, Springer-Verlag, New York, 2004.



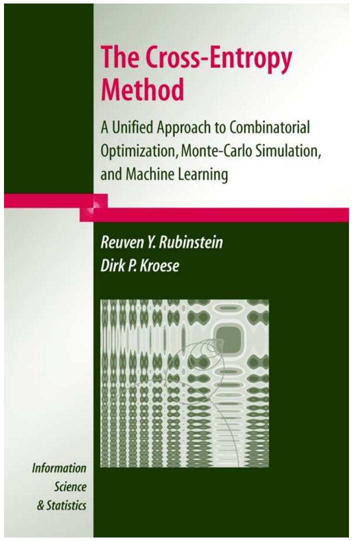
CE Matters

Book: R.Y. Rubinstein and D.P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte Carlo Simulation and Machine Learning*, Springer-Verlag, New York, 2004.



Special Issue: *Annals of Operations Research* (Jan 2005).

CE Matters



Book: R.Y. Rubinstein and D.P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte Carlo Simulation and Machine Learning*, Springer-Verlag, New York, 2004.

Special Issue: *Annals of Operations Research* (Jan 2005).

The CE home page:

`http://www.cemethod.org`

Introduction

The Cross-Entropy Method was originally developed as a simulation method for the estimation of *rare event* probabilities:

$$\boxed{\text{Estimate} \quad \mathbb{P}(S(\boldsymbol{X}) \geq \gamma)}$$

$\boldsymbol{X}$: random vector/process taking values in some set $\mathcal{X}$.
S: function on $\mathcal{X}$.

Introduction

The Cross-Entropy Method was originally developed as a simulation method for the estimation of *rare event* probabilities:

$$\boxed{\text{Estimate} \quad \mathbb{P}(S(\boldsymbol{X}) \geq \gamma)}$$

$\boldsymbol{X}$: random vector/process taking values in some set $\mathcal{X}$.
S: function on $\mathcal{X}$.

It was soon realised that the CE Method could also be used as an *optimization* method:

$$\boxed{\text{Determine} \quad \max_{\boldsymbol{x} \in \mathcal{X}} \; S(\boldsymbol{x})}$$

Applications

- Rare-Event Probability Estimation (Queueing Models), Estimation of Normalization Constant and Counting Problems (Permanent Calculation, Self-Avoiding Random Walks).

Applications

- Rare-Event Probability Estimation (Queueing Models), Estimation of Normalization Constant and Counting Problems (Permanent Calculation, Self-Avoiding Random Walks).
- Combinatorial Optimization, like TSP, Maximal Cut, Scheduling and Production Lines.

Applications

- Rare-Event Probability Estimation (Queueing Models), Estimation of Normalization Constant and Counting Problems (Permanent Calculation, Self-Avoiding Random Walks).
- Combinatorial Optimization, like TSP, Maximal Cut, Scheduling and Production Lines.
- Machine Learning

Applications

- Rare-Event Probability Estimation (Queueing Models), Estimation of Normalization Constant and Counting Problems (Permanent Calculation, Self-Avoiding Random Walks).
- Combinatorial Optimization, like TSP, Maximal Cut, Scheduling and Production Lines.
- Machine Learning
- Pattern Recognition, Clustering and Image Analysis

Applications

- Rare-Event Probability Estimation (Queueing Models), Estimation of Normalization Constant and Counting Problems (Permanent Calculation, Self-Avoiding Random Walks).
- Combinatorial Optimization, like TSP, Maximal Cut, Scheduling and Production Lines.
- Machine Learning
- Pattern Recognition, Clustering and Image Analysis
- DNA Sequence Alignment

Applications

- Rare-Event Probability Estimation (Queueing Models), Estimation of Normalization Constant and Counting Problems (Permanent Calculation, Self-Avoiding Random Walks).
- Combinatorial Optimization, like TSP, Maximal Cut, Scheduling and Production Lines.
- Machine Learning
- Pattern Recognition, Clustering and Image Analysis
- DNA Sequence Alignment
- Simulation-based (noisy) Optimization, like Optimal Buffer Allocation and Optimization in Finance Engineering

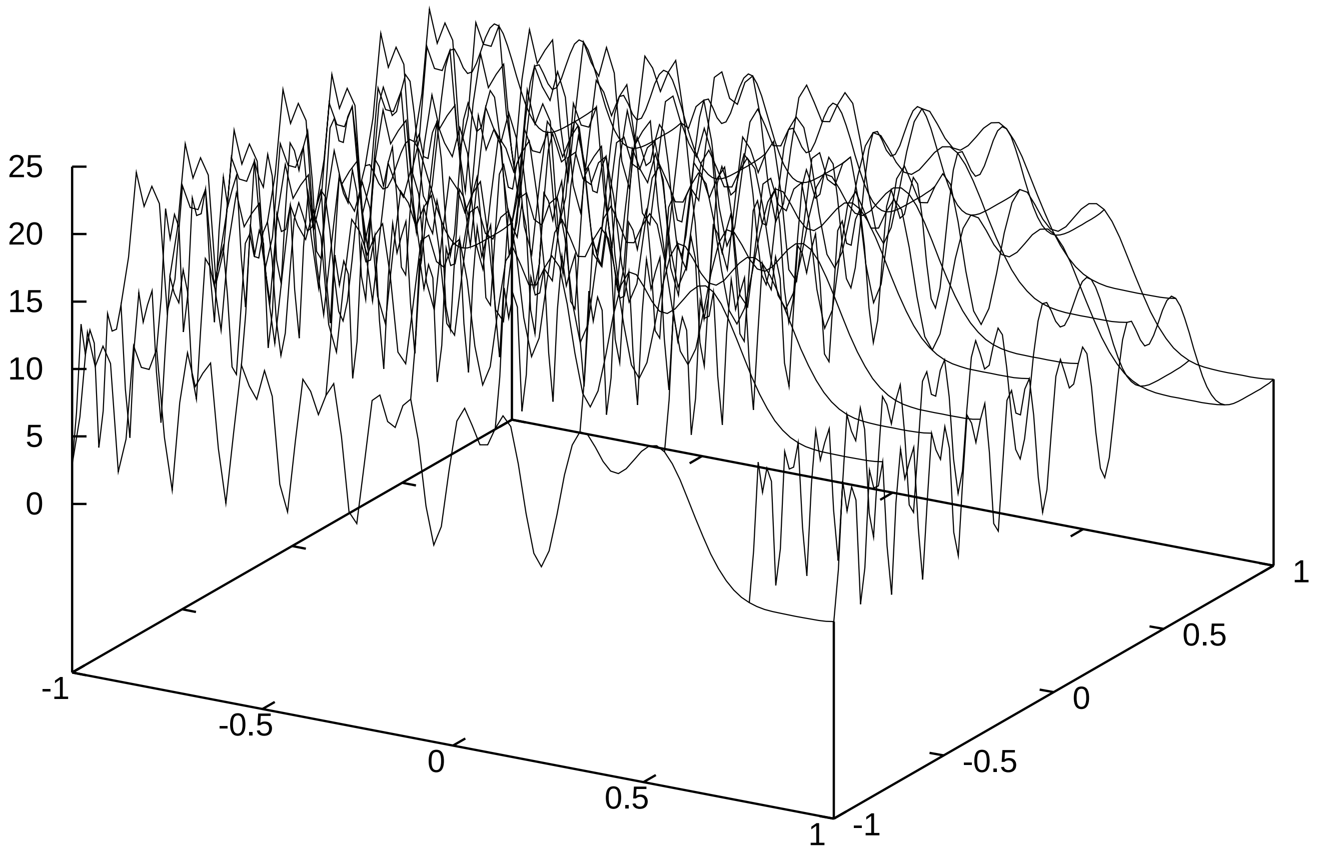
Applications

- Rare-Event Probability Estimation (Queueing Models), Estimation of Normalization Constant and Counting Problems (Permanent Calculation, Self-Avoiding Random Walks).
- Combinatorial Optimization, like TSP, Maximal Cut, Scheduling and Production Lines.
- Machine Learning
- Pattern Recognition, Clustering and Image Analysis
- DNA Sequence Alignment
- Simulation-based (noisy) Optimization, like Optimal Buffer Allocation and Optimization in Finance Engineering
- Multi-extremal Continuous Optimization

Applications

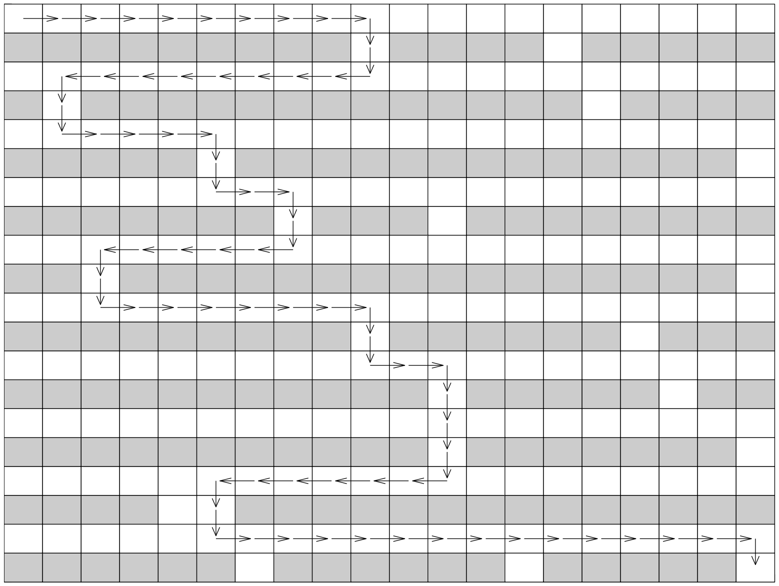
- Rare-Event Probability Estimation (Queueing Models), Estimation of Normalization Constant and Counting Problems (Permanent Calculation, Self-Avoiding Random Walks).
- Combinatorial Optimization, like TSP, Maximal Cut, Scheduling and Production Lines.
- Machine Learning
- Pattern Recognition, Clustering and Image Analysis
- DNA Sequence Alignment
- Simulation-based (noisy) Optimization, like Optimal Buffer Allocation and Optimization in Finance Engineering
- Multi-extremal Continuous Optimization

A Multi-extremal function



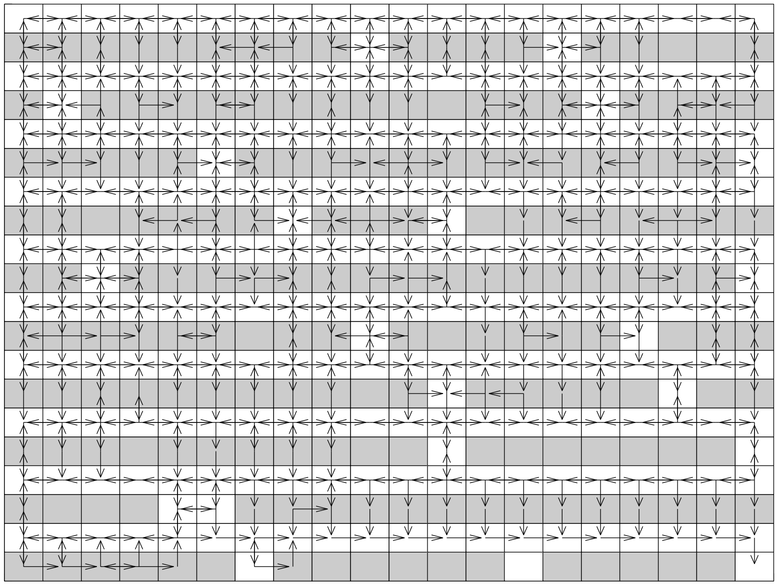
A Maze Problem

The Optimal Trajectory



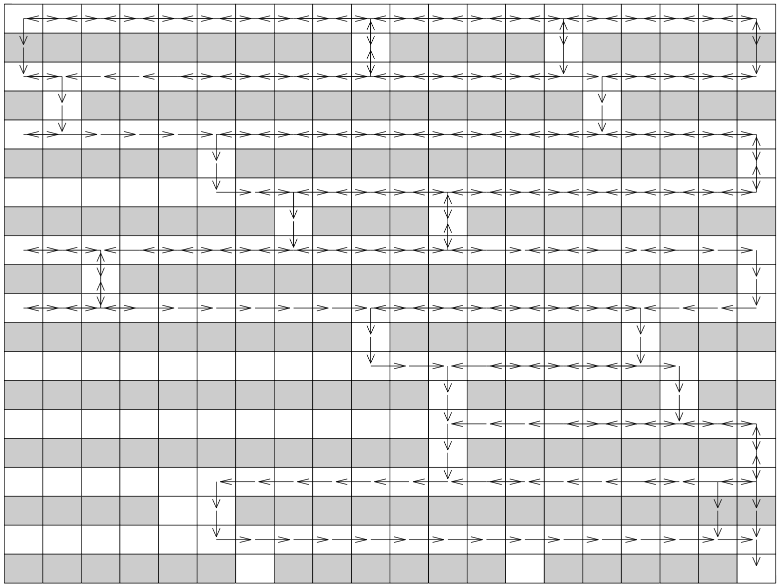
A Maze Problem

Iteration 1:



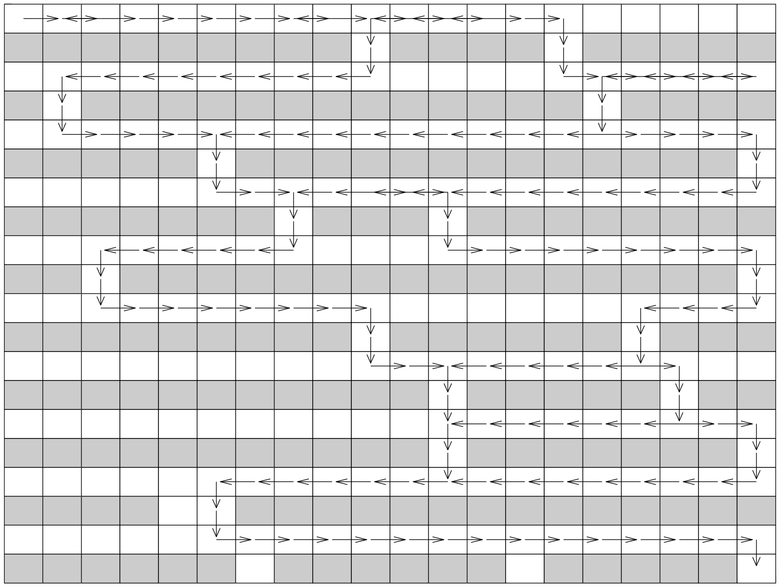
A Maze Problem

Iteration 2:



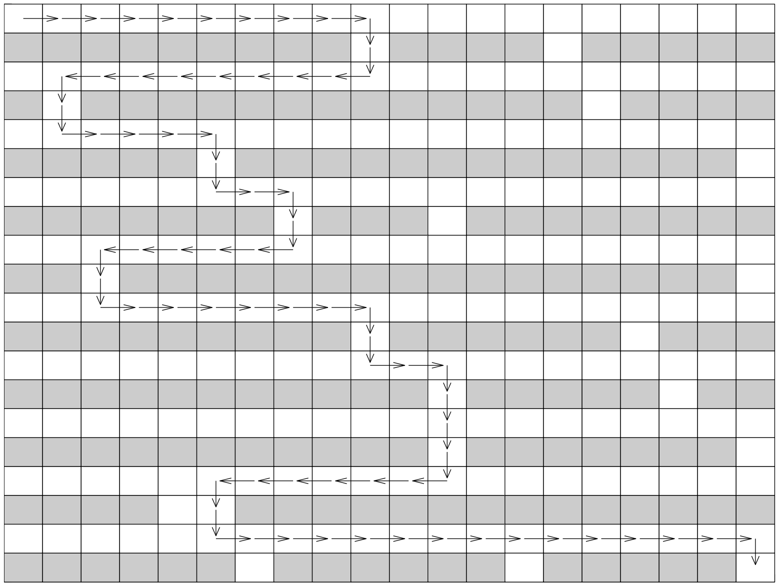
A Maze Problem

Iteration 3:



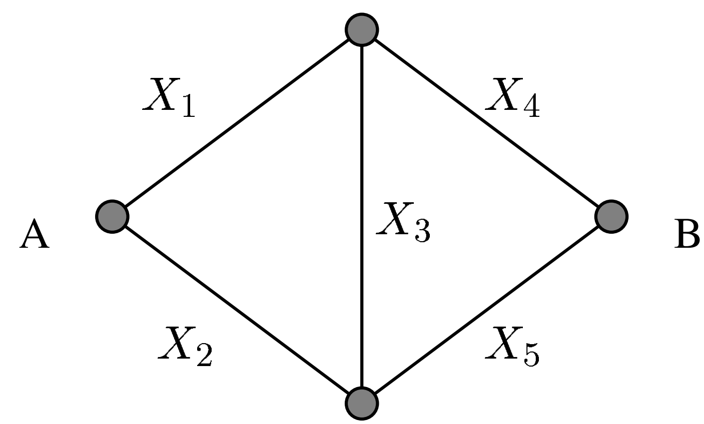
A Maze Problem

Iteration 4:



Example 1: Rare Event Simulation

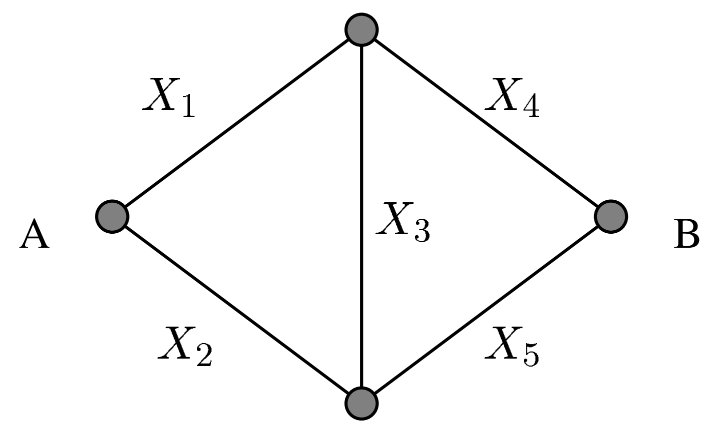
Consider a randomly weighted graph:



The random weights $X_1, \ldots, X_5$ are independent and exponentially distributed with means $u_1, \ldots, u_5$.

Example 1: Rare Event Simulation

Consider a randomly weighted graph:



The random weights $X_1, \ldots, X_5$ are independent and exponentially distributed with means $u_1, \ldots, u_5$.

Find the probability that the length of the shortest path from A to B is greater than or equal to γ.

Crude Monte Carlo (CMC)

Define $\boldsymbol{X} = (X_1, \ldots, X_5)$ and $\boldsymbol{u} = (u_1, \ldots, u_5)$. Let $S(\boldsymbol{X})$ be the length of the shortest path from node A to node B.

Crude Monte Carlo (CMC)

Define $\boldsymbol{X} = (X_1, \ldots, X_5)$ and $\boldsymbol{u} = (u_1, \ldots, u_5)$. Let $S(\boldsymbol{X})$ be the length of the shortest path from node A to node B.

We wish to estimate

$$\ell = \mathbb{P}(S(\boldsymbol{X}) \geq \gamma) = \mathbb{E} I_{\{S(\boldsymbol{X}) \geq \gamma\}} \ .$$

Crude Monte Carlo (CMC)

Define $\boldsymbol{X} = (X_1, \ldots, X_5)$ and $\boldsymbol{u} = (u_1, \ldots, u_5)$. Let $S(\boldsymbol{X})$ be the length of the shortest path from node A to node B.

We wish to estimate

$$\ell = \mathbb{P}(S(\boldsymbol{X}) \geq \gamma) = \mathbb{E} I_{\{S(\boldsymbol{X}) \geq \gamma\}} \; .$$

This can be done via Crude Monte Carlo: sample independent vectors from density $f(\boldsymbol{x}; \boldsymbol{u}) = \prod_{j=1}^{5} \exp(-x_j/u_j)/u_j$, and estimate ℓ via

$$\frac{1}{N} \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \gamma\}} \; .$$

Importance Sampling (IS)

However, for small ℓ this requires a very large simulation effort.

Importance Sampling (IS)

However, for small ℓ this requires a very large simulation effort.

A better way is to use Importance Sampling: draw $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from a different density g, and estimate ℓ via the estimator

$$\widehat{\ell} = \frac{1}{N} \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \gamma\}} W(\boldsymbol{X}_i) \, ,$$

where $W(\boldsymbol{X}) = f(\boldsymbol{X})/g(\boldsymbol{X})$ is called the likelihood ratio.

Which Change of Measure?

If we restrict ourselves to g such that $X_1, \ldots, X_5$ are independent and exponentially distributed with means $v_1, \ldots, v_5$, then

$$W(\boldsymbol{x};\boldsymbol{u},\boldsymbol{v}) := \frac{f(\boldsymbol{x};\boldsymbol{u})}{f(\boldsymbol{x};\boldsymbol{v})} = \exp\left(-\sum_{j=1}^{5} x_j \left(\frac{1}{u_j} - \frac{1}{v_j}\right)\right) \prod_{j=1}^{5} \frac{v_j}{u_j} \; .$$

In this case the "change of measure" is determined by the "reference vector" $\boldsymbol{v} = (v_1, \ldots, v_5)$.

.

Which Change of Measure?

If we restrict ourselves to g such that $X_1, \ldots, X_5$ are independent and exponentially distributed with means $v_1, \ldots, v_5$, then

$$W(\boldsymbol{x}; \boldsymbol{u}, \boldsymbol{v}) := \frac{f(\boldsymbol{x}; \boldsymbol{u})}{f(\boldsymbol{x}; \boldsymbol{v})} = \exp\left(-\sum_{j=1}^{5} x_j \left(\frac{1}{u_j} - \frac{1}{v_j}\right)\right) \prod_{j=1}^{5} \frac{v_j}{u_j} \,.$$

In this case the "change of measure" is determined by the "reference vector" $\boldsymbol{v} = (v_1, \ldots, v_5)$.

Question: How do we find the optimal $\boldsymbol{v} = \boldsymbol{v}^*$?

.

Which Change of Measure?

If we restrict ourselves to g such that $X_1, \ldots, X_5$ are independent and exponentially distributed with means $v_1, \ldots, v_5$, then

$$W(\boldsymbol{x}; \boldsymbol{u}, \boldsymbol{v}) := \frac{f(\boldsymbol{x}; \boldsymbol{u})}{f(\boldsymbol{x}; \boldsymbol{v})} = \exp\left(-\sum_{j=1}^{5} x_j \left(\frac{1}{u_j} - \frac{1}{v_j}\right)\right) \prod_{j=1}^{5} \frac{v_j}{u_j} \,.$$

In this case the "change of measure" is determined by the "reference vector" $\boldsymbol{v} = (v_1, \ldots, v_5)$.

Question: How do we find the optimal $\boldsymbol{v} = \boldsymbol{v}^*$?

Answer: Adaptively by the CE method, by generating a sequence $\{\boldsymbol{v}_t, \gamma_t\}$ until we cross the level γ.

Illustration of the Algorithm

A standard way to find the optimal vector $\boldsymbol{v}$ in the IS density $f(\boldsymbol{x};\boldsymbol{v})$ is to minimize the variance of the estimator

$$\widehat{\ell}(\boldsymbol{v}) = \frac{1}{N}\sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i)\geq\gamma\}}\, W(\boldsymbol{X}_i;\boldsymbol{u},\boldsymbol{v}) \ ,$$

with respect to $\boldsymbol{v}$, that is to solve the program

$$\min_{\boldsymbol{v}} \mathrm{Var}\left\{\frac{1}{N}\sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i)\geq\gamma\}}\, W(\boldsymbol{X}_i;\boldsymbol{u},\boldsymbol{v})\right\},$$

where $\boldsymbol{X}_i \sim f(\boldsymbol{x};\boldsymbol{u})$.

Illustration of the Algorithm

In the CE method we minimize the croos-entropy instead of the variance. The associated program can be written as

$$\min_{\boldsymbol{v}} \left\{ -\frac{1}{N} \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \gamma\}} \ln f(\boldsymbol{X}_i; \boldsymbol{v}) \right\}, \qquad (1)$$

where $\boldsymbol{X}_i \sim f(\boldsymbol{x}; \boldsymbol{u})$.
(Compare with variance minimization)

$$\min_{\boldsymbol{v}} \mathrm{Var} \left\{ \frac{1}{N} \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \gamma\}} W(\boldsymbol{X}_i; \boldsymbol{u}, \boldsymbol{v}) \right\}. \qquad (2)$$

The only advantage of using CE minimization versus variance minimization is that the above CE program can be typically solved analytically, while the variance minimization -

Illustration of the Algorithm

Before proceeding with CE one needs to define the so called the rarity parameter ρ, which is typically choosen $0.1 \leq \rho \leq 0.01$. We illustrate graphically the main steps of the CE Algorithm by considering estimation of

$$\mathbb{P}(X \geq \gamma),$$

where $X \sim \mathsf{Exp}(u)$, that is

$$f(x,u) = u^{-1} e^{-u^{-1}x}.$$

The answer is, of course

$$\mathbb{P}(X \geq \gamma) = e^{-u^{-1}\gamma},$$

but let us do it adaptively (sequentially).

Illustration of the Algorithm

Figure 1 illustrates generation of a sequence of $\{\gamma_t, v_t\}$.

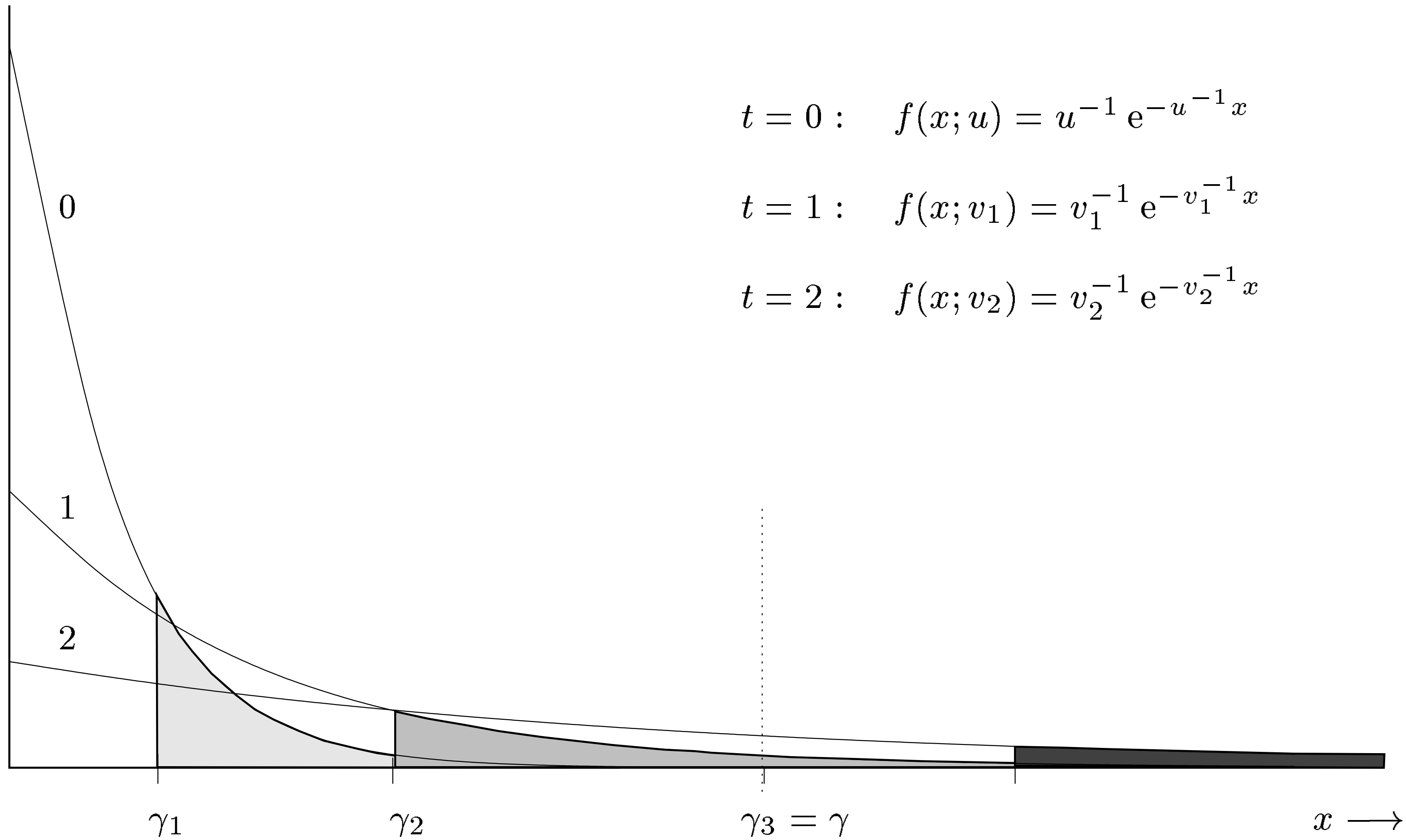


Illustration of the Algorithm

As we shall see, CE Algorithm generates a sequence of tuples $\{\gamma_t, \boldsymbol{v}_t\},\ t \geq 0$, such that γ_t monotonically increases, crossing the level γ after finite number of iterations t, and $\boldsymbol{v}_t,\ (\boldsymbol{v}_0 = \boldsymbol{u}$ converges to the optimal parameter $\boldsymbol{v}^*$ of the IS density $g(\boldsymbol{x}) = f(\boldsymbol{x}, \boldsymbol{v}^*)$.

In our example with $X \sim \exp(u)$ we have that $\ell = P(X \geq \gamma) = e^{-u^{-1}\gamma}$. It can be readily shown that the optimal $v_t = \gamma_t + 1$. For $\gamma = (16,\ 64,\ 256)$ (corresponding to $\ell = e^{-u^{-1}\gamma} \approx (10^{-7},\ 10^{-28},\ 10^{-111})$, such CE algorithm terminates after iteration $T = 2, 3$ and 4, respectively, provided $\rho = 0.001$.

We can next proceed with the actual CE Algorithm.

Generic CE Algorithm

1 Define $\hat{\boldsymbol{v}}_0 := \boldsymbol{u}$. Set $t := 1$ (iteration counter).

Generic CE Algorithm

1 Define $\hat{\boldsymbol{v}}_0 := \boldsymbol{u}$. Set $t := 1$ (iteration counter).

2 Update $\hat{\gamma}_t$: Generate $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ according to $f(\cdot; \hat{\boldsymbol{v}}_{t-1})$. Let $\hat{\gamma}_t$ be the worst of the $\rho \times 100\%$ best performances, provided this is less than γ. Else $\hat{\gamma}_t := \gamma$.

Generic CE Algorithm

1 Define $\hat{\boldsymbol{v}}_0 := \boldsymbol{u}$. Set $t := 1$ (iteration counter).

2 Update $\hat{\gamma}_t$: Generate $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ according to $f(\cdot; \hat{\boldsymbol{v}}_{t-1})$. Let $\hat{\gamma}_t$ be the worst of the $\rho \times 100\%$ best performances, provided this is less than γ. Else $\hat{\gamma}_t := \gamma$.

3 Update $\hat{\boldsymbol{v}}_t$: Use the **same** sample and solve the above CE program. The solution is (coordinate-wise):

$$\hat{v}_{t,j} = \frac{\sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \hat{\gamma}_t\}} W(\boldsymbol{X}_i; \boldsymbol{u}, \hat{\boldsymbol{v}}_{t-1}) \, X_{ij}}{\sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \hat{\gamma}_t\}} W(\boldsymbol{X}_i; \boldsymbol{u}, \hat{\boldsymbol{v}}_{t-1})} \; .$$

Generic CE Algorithm

1 Define $\hat{\boldsymbol{v}}_0 := \boldsymbol{u}$. Set $t := 1$ (iteration counter).

2 Update $\hat{\gamma}_t$: Generate $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ according to $f(\cdot; \hat{\boldsymbol{v}}_{t-1})$. Let $\hat{\gamma}_t$ be the worst of the $\rho \times 100\%$ best performances, provided this is less than γ. Else $\hat{\gamma}_t := \gamma$.

3 Update $\hat{\boldsymbol{v}}_t$: Use the **same** sample and solve the above CE program. The solution is (coordinate-wise):

$$\hat{v}_{t,j} = \frac{\sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \hat{\gamma}_t\}} W(\boldsymbol{X}_i; \boldsymbol{u}, \hat{\boldsymbol{v}}_{t-1})\, X_{ij}}{\sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \hat{\gamma}_t\}} W(\boldsymbol{X}_i; \boldsymbol{u}, \hat{\boldsymbol{v}}_{t-1})} \,.$$

4 If $\hat{\gamma}_t = \gamma$ then proceed to step 5; otherwise set $t := t + 1$ and reiterate from step 2.

Generic CE Algorithm

1 Define $\hat{\boldsymbol{v}}_0 := \boldsymbol{u}$. Set $t := 1$ (iteration counter).

2 Update $\hat{\gamma}_t$: Generate $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ according to $f(\cdot; \hat{\boldsymbol{v}}_{t-1})$. Let $\hat{\gamma}_t$ be the worst of the $\rho \times 100\%$ best performances, provided this is less than γ. Else $\hat{\gamma}_t := \gamma$.

3 Update $\hat{\boldsymbol{v}}_t$: Use the **same** sample and solve the above CE program. The solution is (coordinate-wise):

$$\hat{v}_{t,j} = \frac{\sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \hat{\gamma}_t\}} W(\boldsymbol{X}_i; \boldsymbol{u}, \hat{\boldsymbol{v}}_{t-1}) \, X_{ij}}{\sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \hat{\gamma}_t\}} W(\boldsymbol{X}_i; \boldsymbol{u}, \hat{\boldsymbol{v}}_{t-1})} \, .$$

4 If $\hat{\gamma}_t = \gamma$ then proceed to step 5; otherwise set $t := t + 1$ and reiterate from step 2.

5 Estimate ℓ via the LR estimator, using the final $\hat{\boldsymbol{v}}_T$.

Shortest Path Example

Level: $\gamma = 2$. Fraction of best performances: $\rho = 0.1$. Sample size in steps 2 – 4: $N = 1000$. Final sample size: $N_1 = 10^5$.

t	$\widehat{\gamma}_t$	$\widehat{\boldsymbol{v}}_t$				
0		0.250	0.400	0.100	0.300	0.200
1	0.575	0.513	0.718	0.122	0.474	0.335
2	1.032	0.873	1.057	0.120	0.550	0.436
3	1.502	1.221	1.419	0.121	0.707	0.533
4	1.917	1.681	1.803	0.132	0.638	0.523
5	2.000	1.692	1.901	0.129	0.712	0.564

A similar CE algorithm is used for rare-events in queueing networks.

Shortest Path Example

- The estimate was $1.34 \cdot 10^{-5}$,

Shortest Path Example

- The estimate was $1.34 \cdot 10^{-5}$,
- with an estimated relative error (that is, $\mathrm{Std}(\widehat{\ell})/\mathbb{E}\widehat{\ell}$) of 0.03.

Shortest Path Example

- The estimate was $1.34 \cdot 10^{-5}$,
- with an estimated relative error (that is, $\mathrm{Std}(\widehat{\ell})/\mathbb{E}\widehat{\ell}$) of 0.03.
- The simulation time was only 3 seconds (1/2 second for table).

Another Example: Flipping n Symmetric Coins

Consider an experiment of flipping simultaneously n symmetric coins. Suppose we are interested estimating $\mathbb{P}(S(\boldsymbol{X}) \geq \gamma)$, where $\boldsymbol{X} = (X_1, \ldots, X_n)$ and the random variables $X_1, \ldots, X_n$ are iid each distributed $\mathsf{Ber(p)}$ with $p = 1/2$. Consider the following two case of $S(\boldsymbol{x})$:

$$(a)\ S(\boldsymbol{X}) = \sum_{i=1}^{n} X_i$$

and

$$(b)\ S(\boldsymbol{X}) = \sum_{i=1}^{n-k} X_i - \sum_{j=n-k+1}^{n} X_j .$$

Another Example: Flipping n Symmetric Coins

(a) In this case γ accepts integers values from the set $\{0, \ldots, n\}$. Let $\gamma = n$. Then it is readily seen that for $\gamma = n$ our Algorithm generates $\boldsymbol{p}^* = (1, \ldots, 1)$. If we would set $\gamma = n/2$ then clearly $\boldsymbol{p}^* = (1/2, \ldots, 1/2)$. The corresponding probabilities are $\ell(\boldsymbol{p}^*) = \frac{1}{2}$ and $\ell(\boldsymbol{p}^*) = \frac{1}{2^n}$, respectively. Note that $\mathsf{Ber}(\boldsymbol{p}^*)$ corresponds to the degenerated distribution .

(b) In this case γ accepts integers values from the set $\{-k, \ldots, n-k\}$. For $\gamma = n - k$ our Algorithm generates again $\boldsymbol{p}^* = (1, \ldots, 1)$. The corresponding probability is again $\ell(\boldsymbol{p}^*) = \frac{1}{2^n}$.

Another Example: Flipping n Symmetric Coins

The updating formula for $\boldsymbol{p}$ (at the tth iteration), while solving the CE program

$$\min_{\boldsymbol{v}} \left\{ -\frac{1}{N} \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \gamma\}} \ln f(\boldsymbol{X}_i; \boldsymbol{v}) \right\}, \tag{3}$$

is given by

$$\hat{p}_{t,j} = \frac{\sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \hat{\gamma}_t\}} \, I_{\{X_{ij}=1\}}}{\sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \hat{\gamma}_t\}}} = \frac{\sum_{\boldsymbol{X}_i \in \mathcal{E}} X_{ij}}{|\mathcal{E}|}, \quad j = 2, \ldots, n$$

where $|\mathcal{E}| = \rho N$, the number of elite samples, $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ is a random sample from the density $\mathsf{Ber}(\cdot; \widehat{\boldsymbol{p}}_{\mathsf{t}-1})$ and $\boldsymbol{u} = \boldsymbol{p} = (1/2, \ldots, 1/2)$.

Important Remark

It is important to note that if we employ CE Algorithm to any distributions with finite support without fixing γ in advance, it will iterate until it reaches some γ, which corresponds to the degenerate case and then stops.

Kullback-Leibler Entropy

Idea: Choose $g = f(\cdot; \boldsymbol{v})$ such that the "distance" between the densities g^* and $f(\cdot; \boldsymbol{v})$ is minimal.

Kullback-Leibler Entropy

Idea: Choose $g = f(\cdot;\boldsymbol{v})$ such that the "distance" between the densities g^* and $f(\cdot;\boldsymbol{v})$ is minimal.
The **Kullback-Leibler** or **cross-entropy** distance is defined as:

$$\mathcal{D}(g,h) = \mathbb{E}_g \log \frac{g(\boldsymbol{X})}{h(\boldsymbol{X})}$$
$$= \int g(\boldsymbol{x}) \log g(\boldsymbol{x})\, d\boldsymbol{x} - \int g(\boldsymbol{x}) \log h(\boldsymbol{x})\, d\boldsymbol{x} \; .$$

Kullback-Leibler Entropy

Idea: Choose $g = f(\cdot; \boldsymbol{v})$ such that the “distance” between the densities g^* and $f(\cdot; \boldsymbol{v})$ is minimal.

The **Kullback-Leibler** or **cross-entropy** distance is defined as:

$$\mathcal{D}(g, h) = \mathbb{E}_g \log \frac{g(\boldsymbol{X})}{h(\boldsymbol{X})}$$
$$= \int g(\boldsymbol{x}) \log g(\boldsymbol{x})\, d\boldsymbol{x} - \int g(\boldsymbol{x}) \log h(\boldsymbol{x})\, d\boldsymbol{x} \ .$$

Determine the optimal $\boldsymbol{v}^*$ from $\min_{\boldsymbol{v}} \mathcal{D}(g^*, f(\cdot; \boldsymbol{v}))$, where the best density (zero variance estimator!) is

$$g^*(\boldsymbol{x}) := \frac{I_{\{S(\boldsymbol{x}) \geq \gamma\}} f(\boldsymbol{x}; \boldsymbol{u})}{\ell} \ .$$

Kullback-Leibler Entropy

This is equivalent to solving

$$\max_{\boldsymbol{v}} \mathbb{E}_{\boldsymbol{u}} \, I_{\{S(\boldsymbol{X}) \geq \gamma\}} \log f(\boldsymbol{X}; \boldsymbol{v}) \ .$$

Using again IS, we can rewrite this as

$$\max_{\boldsymbol{v}} \ \mathbb{E}_{\boldsymbol{u}} \, I_{\{S(\boldsymbol{X}) \geq \gamma\}} \ \log f(\boldsymbol{X}; \boldsymbol{v}).$$

Applications to Rare-Events and Normalization Constant Calculation

- Calculating the Permanent.
- Counting the Number of 0-1 Tables for Given row sums and column sums.
- Calculating the Normalization Constant for Queueing Networks in the Product Form.
- Calculating the Partition Function for Boltzmann type Models, like Ising Model.

All the above problems are of deterministic nature, so we shall introduce randomness, typicaly an uniform pdf over the sample space.

Calculating the Permanent

The permanent problem can be stated as follows: Given an n x n matrix $A = (a_{ij})$, where each element a_{ij} is either 0 or 1, and a set of all permutations Π of the sequence $\{1, \ldots, n\}$, find the permanent of A, which is

$$\operatorname{perm}(A) = \sum_{\sigma \in \Pi} \prod_{i=1}^{n} a_{i\sigma(i)}.$$

A 3 x 3 permanent matrix

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

Calculating the Permanent

Here $\sigma \in \Pi$ is a permutation of $\{1, \ldots, n\}$, which can be represented as

$$\begin{pmatrix} 1, & 2, & \ldots, & n \\ \sigma(1), & \sigma(2), & \ldots, & \sigma(n) \end{pmatrix}, \tag{4}$$

$$\mathcal{R}_A = \{\sigma : \; S(\sigma) = 1\}, \tag{5}$$

$$S(\sigma) = \prod_{i=1}^{n} a_{i\sigma(i)}, \tag{6}$$

where $\sigma(i)$ records the new position of label i after the permutation. Note that if one of the elements $a_{i\sigma(i)}, \; i = 1, \ldots, n$ in (6) is zero, then $S(\sigma)$ is zero.

Combinatorial Optimization Problems (COP's)

Let $\mathcal{X}$ be a finite set of *states*, and let S be a real-valued *performance function* on $\mathcal{X}$. We wish to find the maximum of S over $\mathcal{X}$, that is to solve

$$S(\boldsymbol{x}^*) = \gamma^* = \max_{\boldsymbol{x} \in \mathcal{X}} S(\boldsymbol{x}) \ . \tag{7}$$

The starting point in the methodology of the CE method is to associate with the optimization problem (7) a meaningful estimation problem.

A COP Example: Flipping n Coins

Consider again the “flipping coins" example but in the optimization framework (7). Assume that we want to maximize $S(\boldsymbol{x})$, where again:

$$(a)\ S(\boldsymbol{x}) = \sum_{i=1}^{n} x_i$$

and

$$(b)\ S(\boldsymbol{x}) = \sum_{i=1}^{n-k} x_i - \sum_{j=n-k+1}^{n} x_j$$

and where each component of the decision vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ is either 0 or 1. Clearly, the optimal solutions of (7) for the cases (a) and (b) are $\boldsymbol{x}^* = (1, \ldots, 1)$ and $\boldsymbol{x}^* = (1, \ldots, 1, 0, \ldots, 0)$, respectively.

A COP Example: Flipping n Coins

To put this COP into rare-event framework we merely associate each coordinate $x_i,\ i = 1, \ldots, n$ with a Bernoulli random variable $X_i,\ i = 1, \ldots, n$ assuming, say that each $X_i \sim \mathsf{Ber}(1/2)$ and run our Algorithm for rare-events until it reaches γ^*.

All COP's fit in the framework of this toy example: **Put the deterministic COP into ASP (associated stochastic program), generate trajectories and use the algorithm for rare events until you reach γ^*.**

Detailed Generic CE Algorithm

1. Choose an initial parameter vector, $\boldsymbol{v}_0$. Set $t = 1$.
2. Generate $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_N$ from the density $f(\cdot; \boldsymbol{v}_{t-1})$
3. Compute the sample $(1-\rho)$-quantile, $\hat{\gamma}_t = S_{(\lceil (1-\rho)N \rceil)}$, locate the elite samples $\mathcal{E} = \{\boldsymbol{X}_i \,:\, S(\boldsymbol{X}_i) \geq \hat{\gamma}_t\}$, and determine

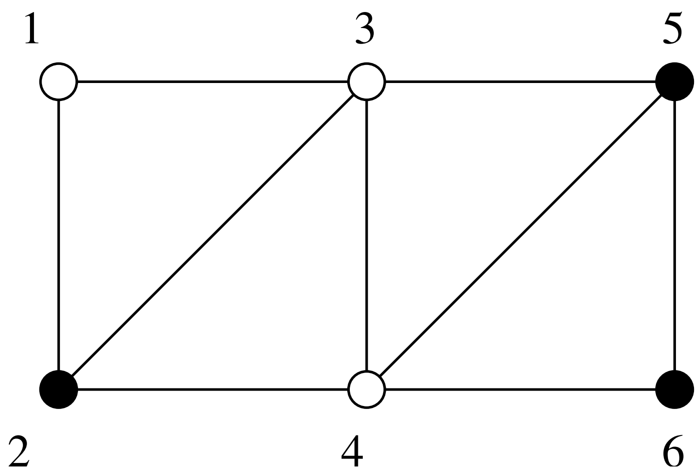
$$\tilde{\boldsymbol{v}}_t = \underset{\boldsymbol{v}}{\operatorname{argmax}} \sum_{\boldsymbol{X}_i \in \mathcal{E}} \ln f(\boldsymbol{X}_i; \boldsymbol{v}).$$

   Set the current estimate for the optimal parameter vector, $\boldsymbol{v}_t$ to: $\boldsymbol{v}_t = \alpha \tilde{\boldsymbol{v}}_t + (1-\alpha)\boldsymbol{v}_{t-1}$.
4. If the stopping criterion is met, stop; otherwise set $t = t + 1$, and return to Step 2.

The Max-Cut problem

Consider a weighted graph G with node set $V = \{1, \ldots, n\}$. Partition the nodes of the graph into two subsets V_1 and V_2 such that the sum of the weights of the edges going from one subset to the other is maximised.

Example



Cost matrix:

$$C = \begin{pmatrix} 0 & c_{12} & c_{13} & 0 & 0 & 0 \\ c_{21} & 0 & c_{23} & c_{24} & 0 & 0 \\ c_{31} & c_{32} & 0 & c_{34} & c_{35} & 0 \\ 0 & c_{42} & c_{43} & 0 & c_{45} & c_{46} \\ 0 & 0 & c_{53} & c_{54} & 0 & c_{56} \\ 0 & 0 & 0 & c_{64} & c_{65} & 0 \end{pmatrix}.$$

$\{V_1, V_2\} = \{\{1,3,4\},\{2,5,6\}\}$ is a possible cut. The cost of the cut is

$$c_{12} + c_{32} + c_{35} + c_{42} + c_{45} + c_{46}.$$

Cut Vector

We can represent a cut via a cut vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, where $x_i = 1$ if node i belongs to same partition as 1, and 0 else.

Cut Vector

We can represent a cut via a cut vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, where $x_i = 1$ if node i belongs to same partition as 1, and 0 else.

For example, the cut $\{\{1, 3, 4\}, \{2, 5, 6\}\}$ can be represented via the cut vector $(1, 0, 1, 1, 0, 0)$.

Cut Vector

We can represent a cut via a cut vector $\boldsymbol{x} = (x_1, \dots, x_n)$, where $x_i = 1$ if node i belongs to same partition as 1, and 0 else.

For example, the cut $\{\{1, 3, 4\}, \{2, 5, 6\}\}$ can be represented via the cut vector $(1, 0, 1, 1, 0, 0)$.

Let $\mathcal{X}$ be the set of all cut vectors $\boldsymbol{x} = (1, x_2, \dots, x_n)$.

Cut Vector

We can represent a cut via a cut vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, where $x_i = 1$ if node i belongs to same partition as 1, and 0 else.

For example, the cut $\{\{1,3,4\},\{2,5,6\}\}$ can be represented via the cut vector $(1,0,1,1,0,0)$.

Let $\mathcal{X}$ be the set of all cut vectors $\boldsymbol{x} = (1, x_2, \ldots, x_n)$.

Let $S(\boldsymbol{x})$ be the corresponding cost of the cut.

Cut Vector

We can represent a cut via a cut vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, where $x_i = 1$ if node i belongs to same partition as 1, and 0 else.

For example, the cut $\{\{1, 3, 4\}, \{2, 5, 6\}\}$ can be represented via the cut vector $(1, 0, 1, 1, 0, 0)$.

Let $\mathcal{X}$ be the set of all cut vectors $\boldsymbol{x} = (1, x_2, \ldots, x_n)$.

Let $S(\boldsymbol{x})$ be the corresponding cost of the cut.

We wish to maximize $S(\boldsymbol{x})$ via the CE method.

General CE Procedure

The CE method cast the original optimization problem of $S(\boldsymbol{x})$ into an associated rare-events probability estimation problem, that estimation of

$$\ell = \mathbb{P}(S(\boldsymbol{X}) \geq \gamma) = \mathbb{E} I_{\{S(\boldsymbol{X}) \geq \gamma\}} \ .$$

and involves the following iterative steps:

- Formulate a parameterized random mechanism to *generate* the objects $\boldsymbol{x} \in \mathcal{X}$ (here, cut vectors).

General CE Procedure

The CE method cast the original optimization problem of $S(\boldsymbol{x})$ into an associated rare-events probability estimation problem, that estimation of

$$\ell = \mathbb{P}(S(\boldsymbol{X}) \geq \gamma) = \mathbb{E} I_{\{S(\boldsymbol{X}) \geq \gamma\}} \ .$$

and involves the following iterative steps:

- Formulate a parameterized random mechanism to *generate* the objects $\boldsymbol{x} \in \mathcal{X}$ (here, cut vectors).
- Give the *updating formulas* for the parameters of the random mechanism (obtained via Cross-Entropy minimization), in order to produce a better sample in the next iteration.

Generation and Updating Formulas

Generation of cut vectors: The most natural and easiest way to generate the cut vectors is to let $X_2, \ldots, X_n$ be independent Bernoulli random variables with success probabilities $p_2, \ldots, p_n$.

Generation and Updating Formulas

Generation of cut vectors: The most natural and easiest way to generate the cut vectors is to let $X_2, \ldots, X_n$ be independent Bernoulli random variables with success probabilities $p_2, \ldots, p_n$.

Updating formulas:

$$\hat{p}_{t,j} = \frac{\sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \hat{\gamma}_t\}} \, I_{\{X_{ij}=1\}}}{\sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \hat{\gamma}_t\}}} = \frac{\sum_{\boldsymbol{X}_i \in \mathcal{E}} X_{ij}}{|\mathcal{E}|}, \quad j = 2, \ldots, n$$

where $|\mathcal{E}| = \rho N$, the number of elite samples:

Algorithm

1 Start with $\hat{\boldsymbol{p}}_0 = (1/2, \ldots, 1/2)$. Let $t := 1$.

Algorithm

1 Start with $\hat{\boldsymbol{p}}_0 = (1/2, \ldots, 1/2)$**. Let** $t := 1$**.**

2 Update $\hat{\gamma}_t$**: Draw** $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ **from** $\mathsf{Ber}(\hat{\boldsymbol{p}}_t)$**. Let** $\hat{\gamma}_t$ **be the worst performance of the** $\rho \times 100\%$ **best performances.**

Algorithm

1 **Start with** $\hat{\boldsymbol{p}}_0 = (1/2, \ldots, 1/2)$**. Let** $t := 1$**.**

2 **Update** $\hat{\gamma}_t$**: Draw** $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ **from** $\mathsf{Ber}(\hat{\boldsymbol{p}}_t)$**. Let** $\hat{\gamma}_t$ **be the worst performance of the** $\rho \times 100\%$ **best performances.**

3 **Update** $\hat{\boldsymbol{p}}_t$**: Use the same sample to calculate**

$$\hat{p}_{t,j} = \frac{\sum_{\boldsymbol{X}_i \in \mathcal{E}} X_{ij}}{|\mathcal{E}|} = \frac{\sum_{i=1}^N I_{\{S(\boldsymbol{X}_i) \geq \hat{\gamma}_t\}} \, I_{\{X_{ij}=1\}}}{\sum_{i=1}^N I_{\{S(\boldsymbol{X}_i) \geq \hat{\gamma}_t\}}} \; ,$$

$j = 1, \ldots, n$**, where** $\boldsymbol{X}_i = (X_{i1}, \ldots, X_{in})$**, and increase** t **by 1.**

Algorithm

1 **Start with** $\hat{\boldsymbol{p}}_0 = (1/2, \ldots, 1/2)$**. Let** $t := 1$**.**

2 **Update** $\hat{\gamma}_t$**: Draw** $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ **from** $\mathsf{Ber}(\hat{\boldsymbol{p}}_t)$**. Let** $\hat{\gamma}_t$ **be the worst performance of the** $\rho \times 100\%$ **best performances.**

3 **Update** $\hat{\boldsymbol{p}}_t$**: Use the same sample to calculate**

$$\hat{p}_{t,j} = \frac{\sum_{\boldsymbol{X}_i \in \mathcal{E}} X_{ij}}{|\mathcal{E}|} = \frac{\sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \hat{\gamma}_t\}} \, I_{\{X_{ij}=1\}}}{\sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \hat{\gamma}_t\}}} \,,$$

$j = 1, \ldots, n$**, where** $\boldsymbol{X}_i = (X_{i1}, \ldots, X_{in})$**, and increase** t **by 1.**

4 **If the stopping criterion is met, then stop; otherwise set** $t := t + 1$ **and reiterate from step 2.**

TSP Problem

The travelling salesman problem (TSP) can be formulated as follows. Consider a weighted graph G with n nodes (cities) labeled $1, 2, \ldots, n$. Each edge from i to j has weight or cost c_{ij}, representing the length of the road. The problem is to find the shortest *tour* that visits all the cities exactly once,that is

$$\min_{\boldsymbol{x} \in \mathcal{X}} S(\boldsymbol{x}) \quad = \quad \min_{\boldsymbol{x} \in \mathcal{X}} \left\{ \sum_{i=1}^{n-1} c_{x_i, x_{i+1}} + c_{x_n, 1} \right\} . \qquad (8)$$

TSP Problem

The cost matrix:

$$C = \begin{pmatrix} c_{(1,1)} & c_{(1,2)} & \cdots & c_{(1,n)} \\ c_{(2,1)} & c_{(2,2)} & \cdots & c_{(2,n)} \\ \vdots & \vdots & \vdots & \vdots \\ c_{(n,1)} & c_{(n,2)} & \cdots & c_{(n,n)} \end{pmatrix}. \quad (9)$$

The associated probability matrix:

$$P = \begin{pmatrix} p_{(1,1)} & p_{(1,2)} & \cdots & p_{(1,n)} \\ p_{(2,1)} & p_{(2,2)} & \cdots & p_{(2,n)} \\ \vdots & \vdots & \vdots & \vdots \\ p_{(n,1)} & p_{(n,2)} & \cdots & p_{(n,n)} \end{pmatrix}. \quad (10)$$

Generation tours in the TSP Problem

1. Define $P^{(1)} = P$ and $X_1 = 1$. Let $k = 1$.
2. Obtain $P^{(k+1)}$ from $P^{(k)}$ by first setting the X_k-th column of $P^{(k)}$ to 0 and then normalizing the rows to sum up to 1. Generate X_{k+1} from the distribution formed by the X_k-th row of $P^{(k)}$.
3. If $k = n - 1$ then **stop**; otherwise set $k = k + 1$ and reiterate from step 2.

Synthetic Max-Cut Problem

Let $n = 2m$. Consider the following symmetric cost matrix:

$$C = \begin{pmatrix} Z_{11} & B_{12} \\ B_{21} & Z_{22} \end{pmatrix} ,$$

where Z_{ii} is Uniform [0,1] matrix and B_{ij} a matrix of ones. By construction, the optimal cut is given by $V^* = \{V_1^*, V_2^*\}$, with

$$V_1^* = \{1, \ldots, m\} \quad \text{and} \quad V_2^* = \{m+1, \ldots, n\} ,$$

and the optimal value of the cut is

$$\gamma^* = m^2 .$$

- Results for the case with $n = 400, m = 200$ nodes are given next.

- Results for the case with $n = 400, m = 200$ nodes are given next.
- Parameters: $\rho = 0.1$, $N = 1000$.

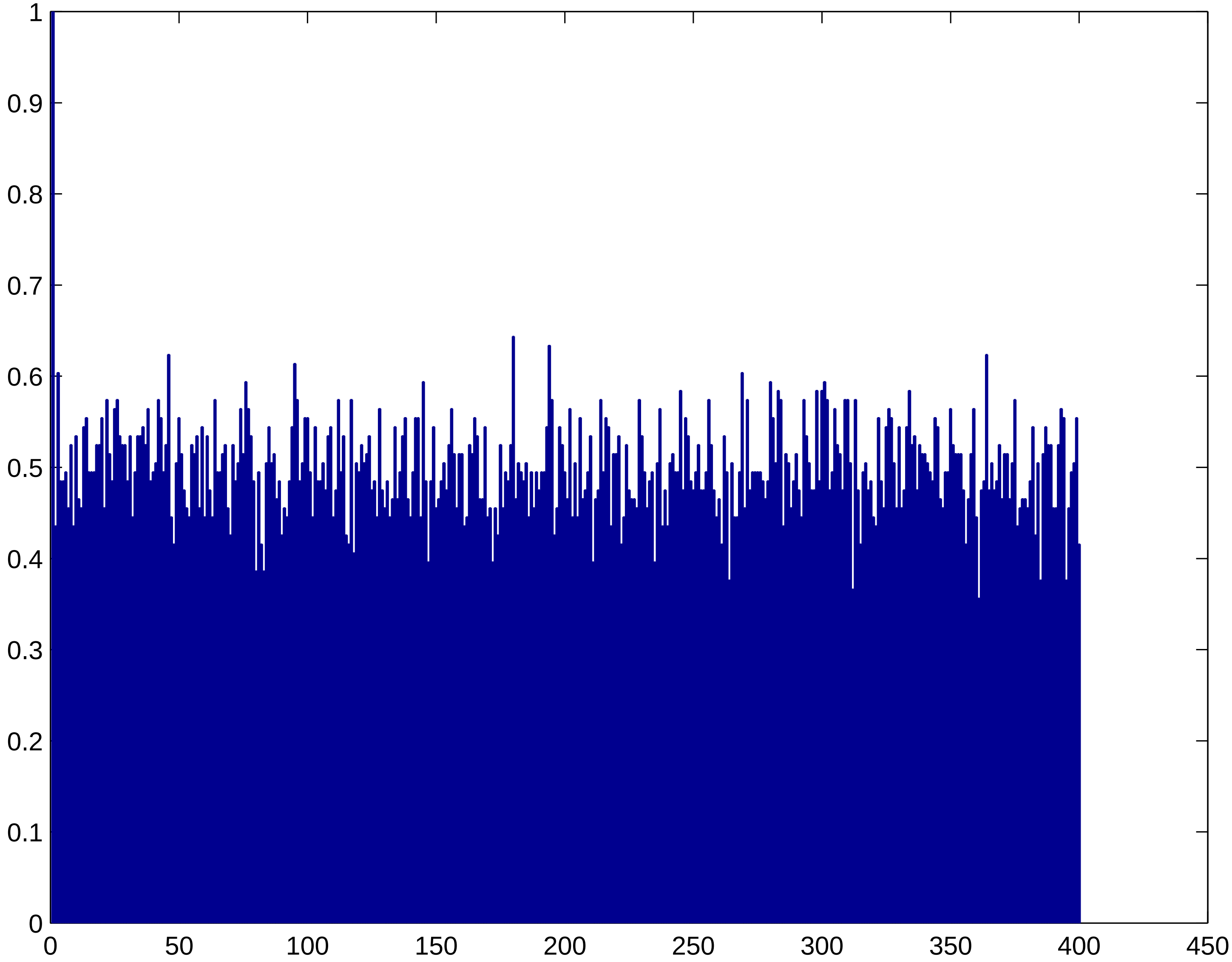
- Results for the case with $n = 400, m = 200$ nodes are given next.
- Parameters: $\rho = 0.1$, $N = 1000$.
- The CPU time was only 100 seconds (Matlab, pentium III, 500 Mhz).

- Results for the case with $n = 400, m = 200$ nodes are given next.
- Parameters: $\rho = 0.1$, $N = 1000$.
- The CPU time was only 100 seconds (Matlab, pentium III, 500 Mhz).
- The CE algorithm converges quickly, yielding the exact optimal solution 40000 in 22 iterations.

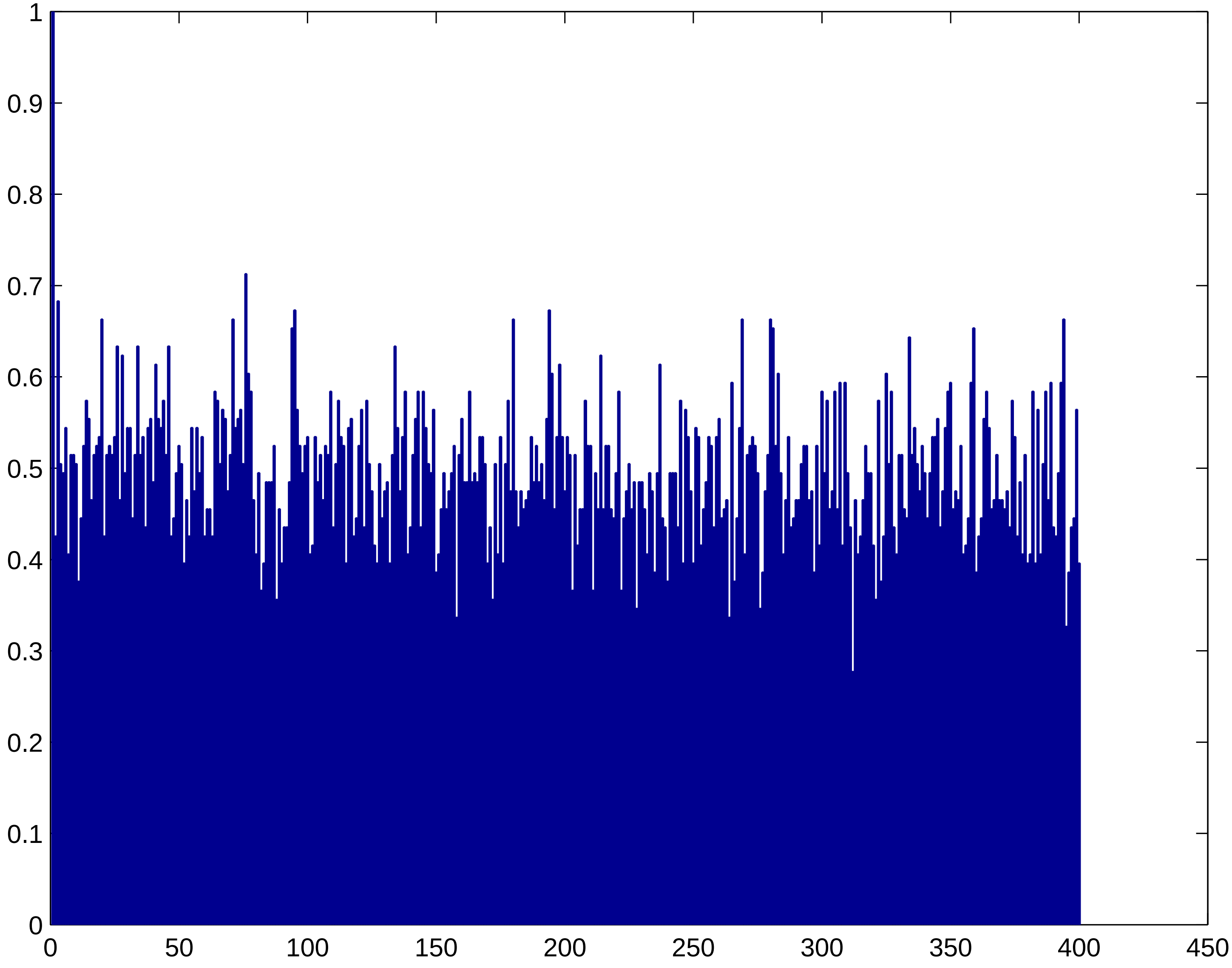
t	$\widehat{\gamma}_t$	$S_{t,(N)}$	$\|\|\hat{\boldsymbol{p}}_t - \boldsymbol{p}^*\|\|$
1	30085.3	30320.9	9.98
2	30091.3	30369.4	10.00
3	30113.7	30569.8	9.98
4	30159.2	30569.8	9.71
5	30350.8	30652.9	9.08
6	30693.6	31244.7	8.37
7	31145.1	31954.8	7.65
8	31711.8	32361.5	6.94
9	32366.4	33050.3	6.27
10	33057.8	33939.9	5.58
11	33898.6	34897.9	4.93

t	$\widehat{\gamma}_t$	$S_{t,(N)}$	$\|\|\hat{\boldsymbol{p}}_t - \boldsymbol{p}^*\|\|$
12	34718.9	35876.4	4.23
13	35597.1	36733.0	3.60
14	36368.9	37431.7	3.02
15	37210.5	38051.2	2.48
16	37996.7	38654.5	1.96
17	38658.8	39221.9	1.42
18	39217.1	39707.8	1.01
19	39618.3	40000.0	0.62
20	39904.5	40000.0	0.29
21	40000.0	40000.0	0.14
22	40000.0	40000.0	0.00

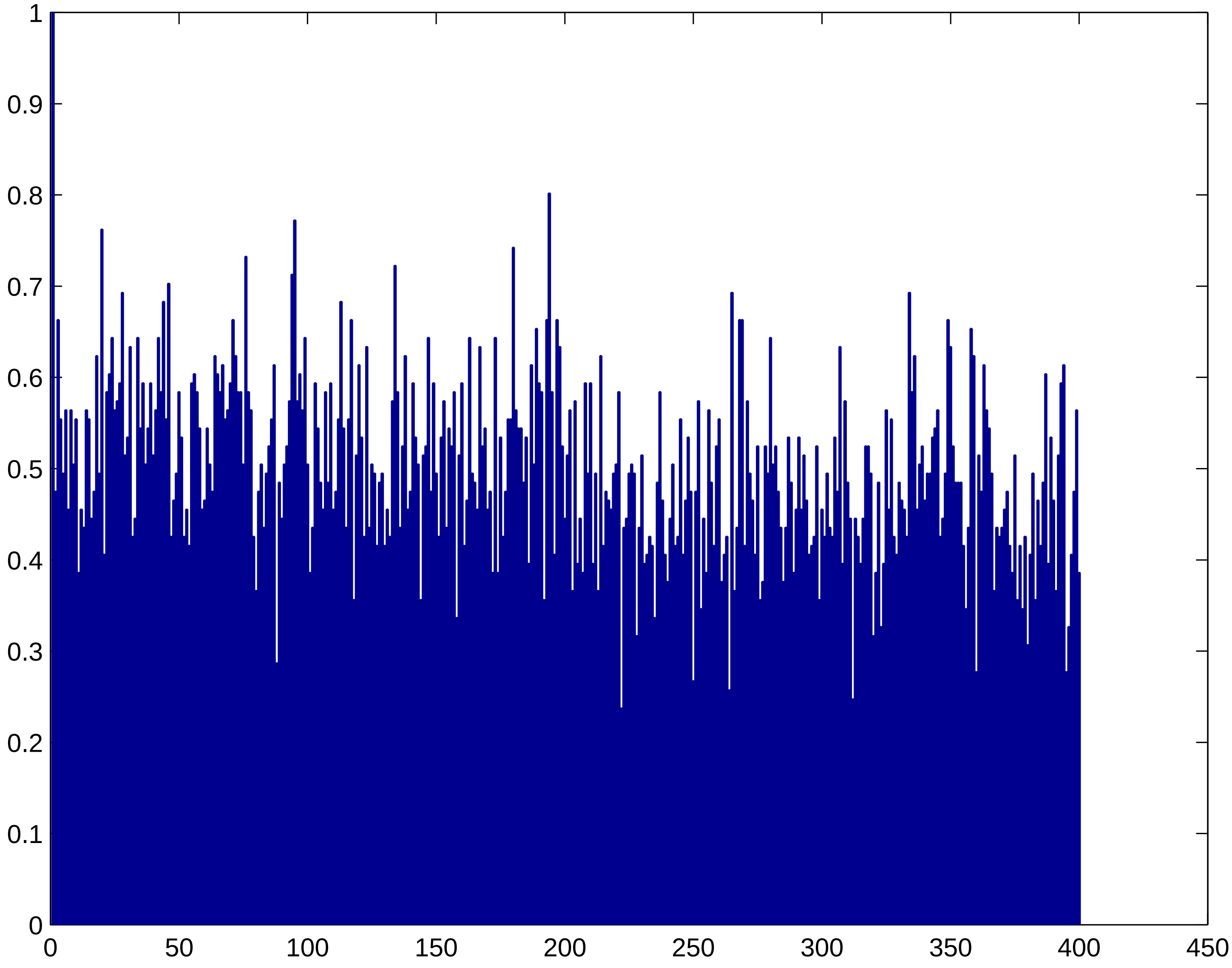
Max-Cut



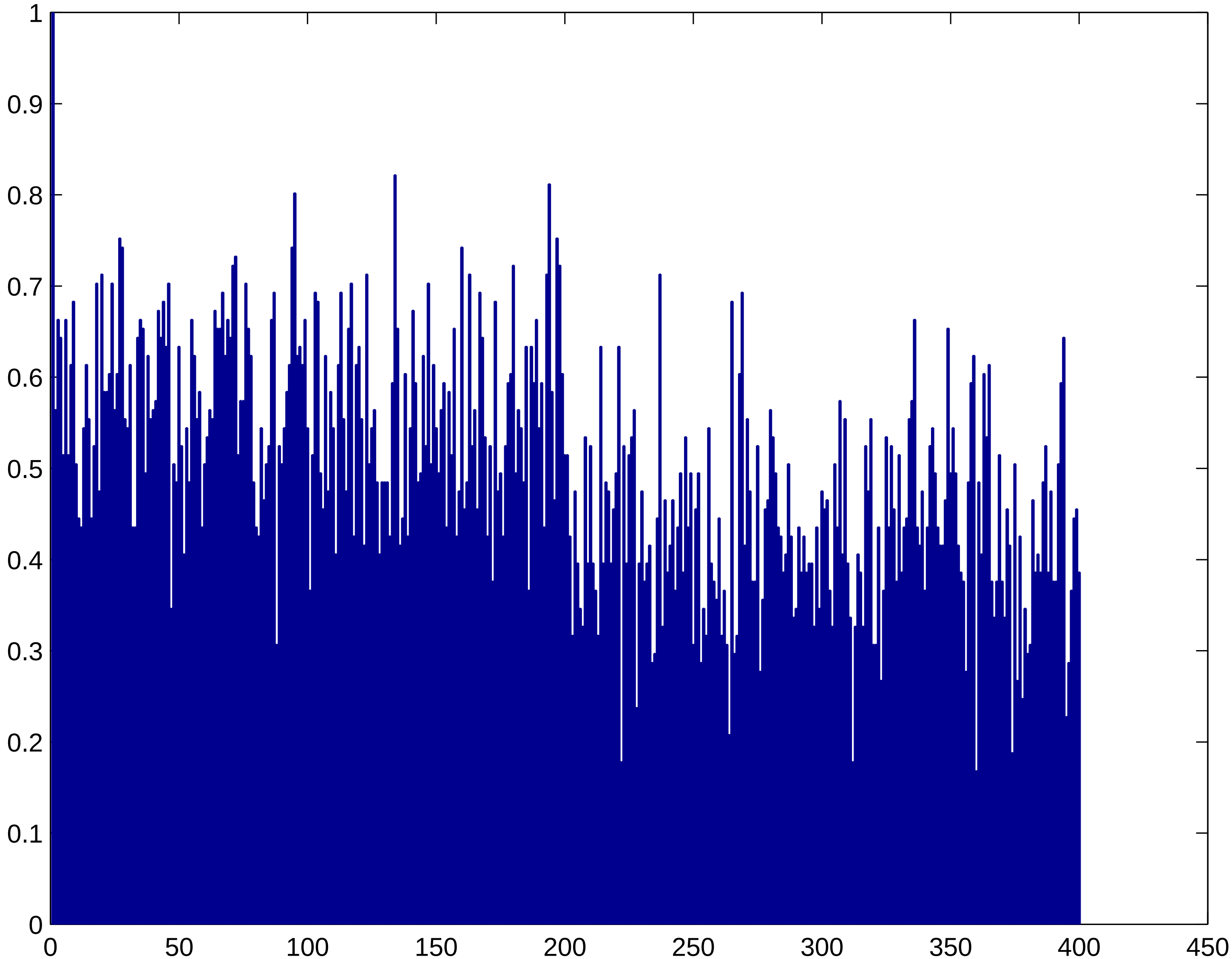
Max-Cut



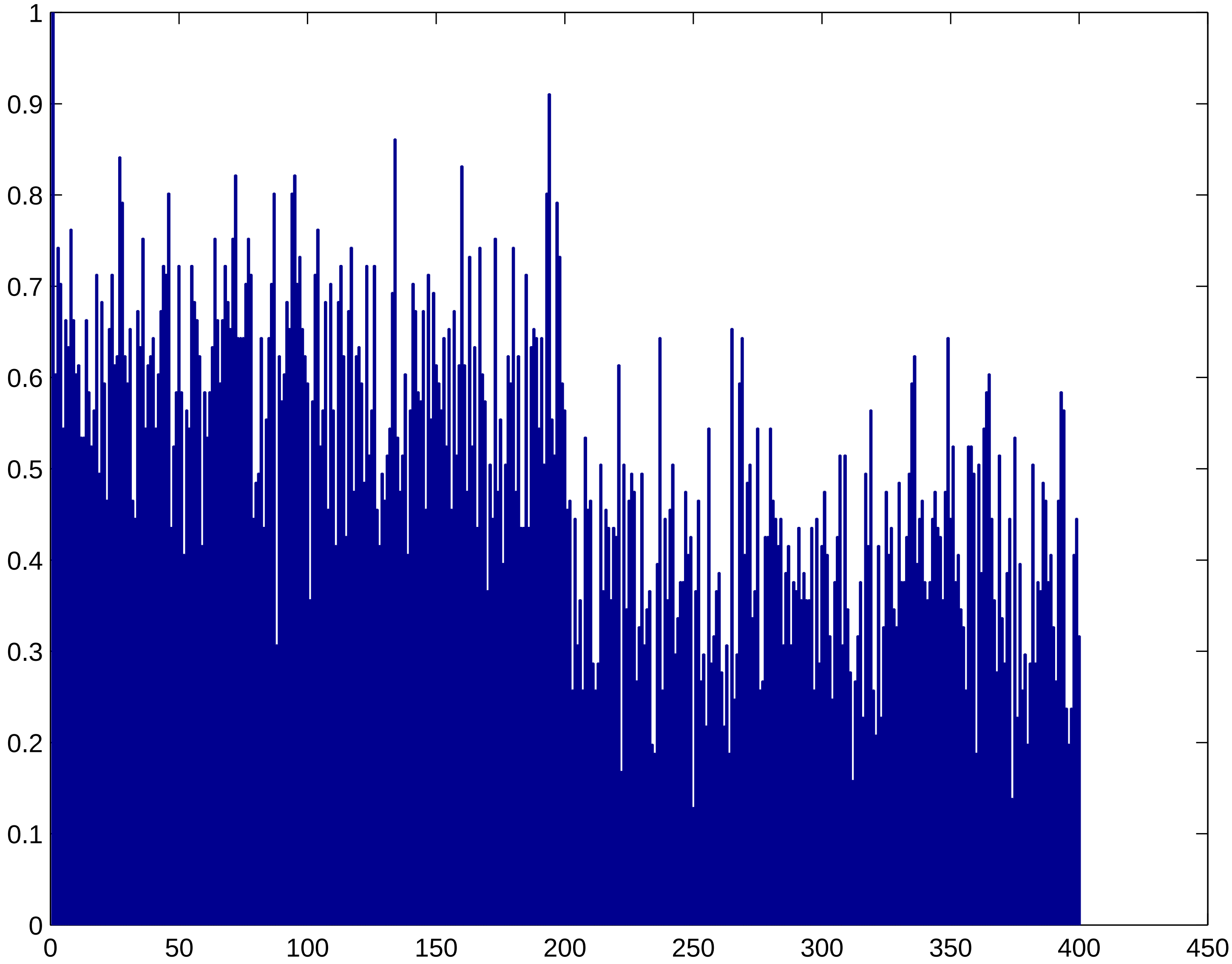
Max-Cut



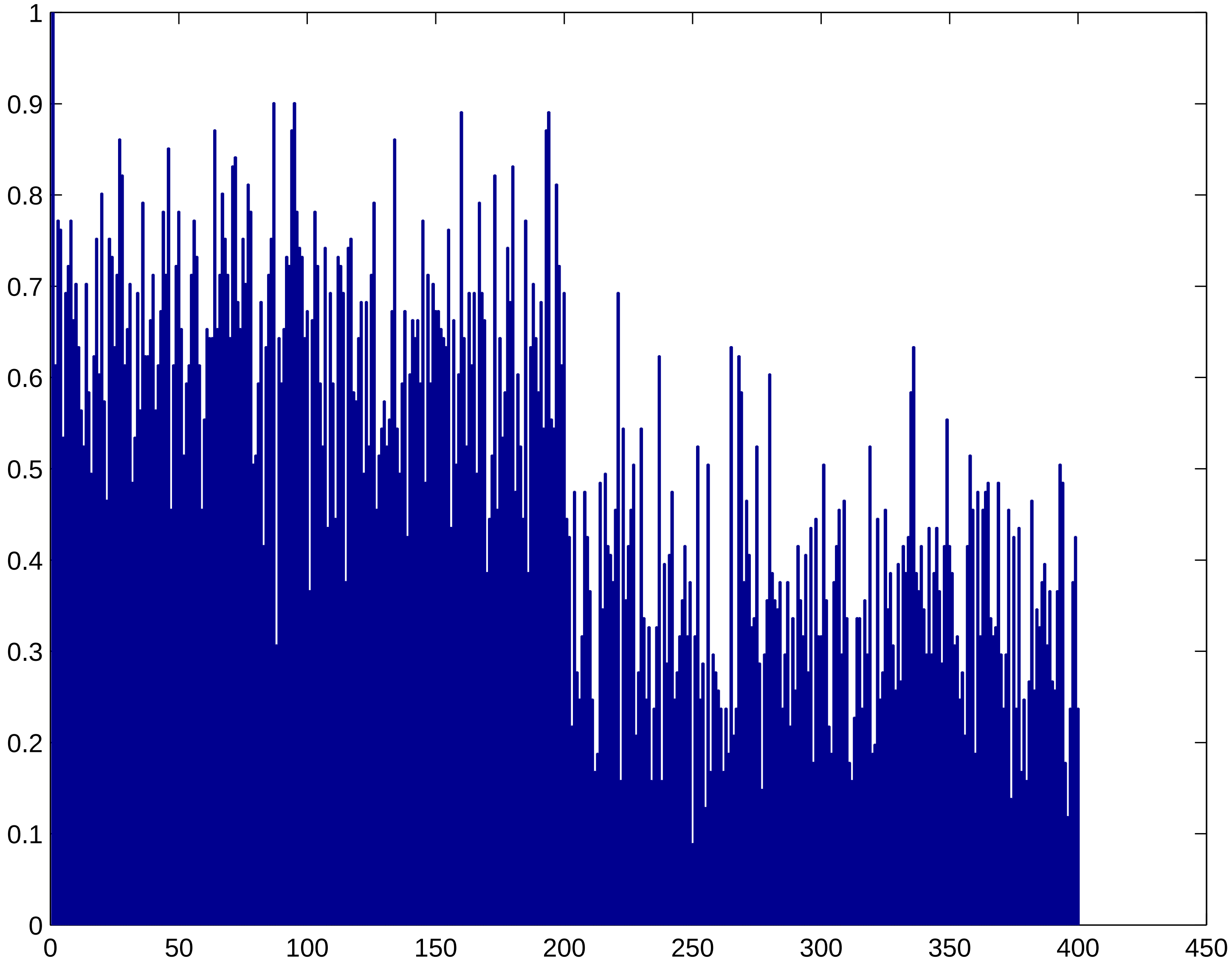
Max-Cut



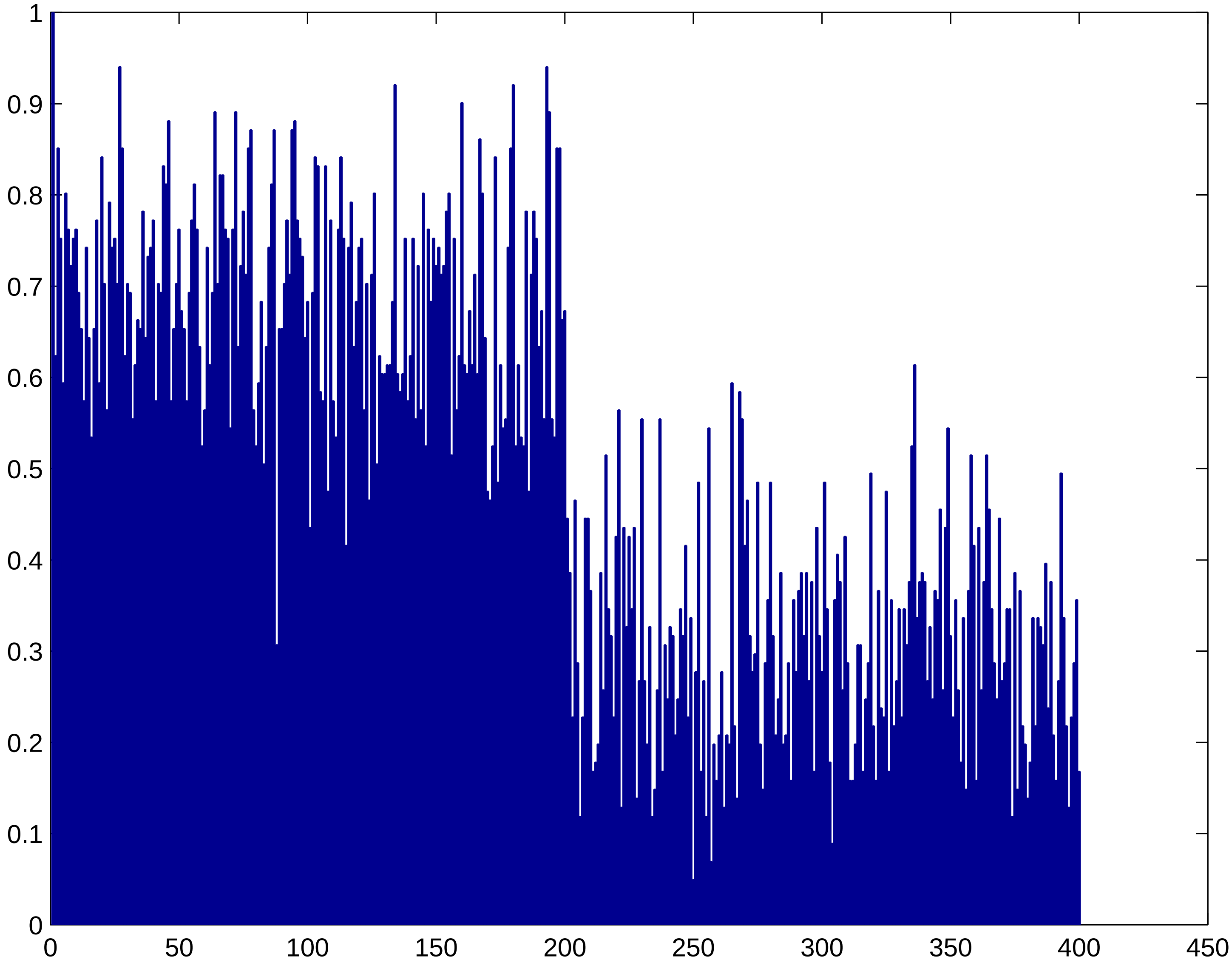
Max-Cut



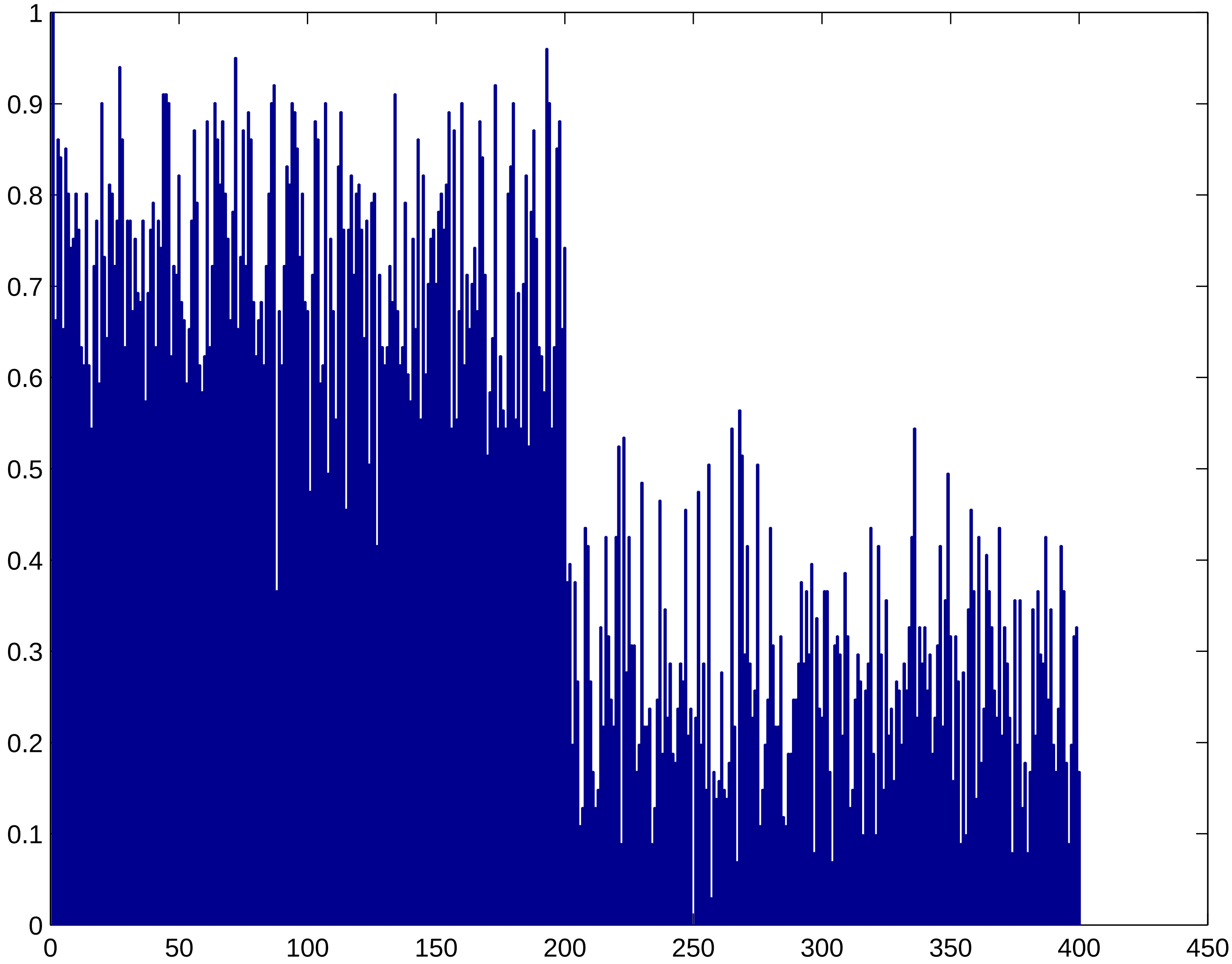
Max-Cut



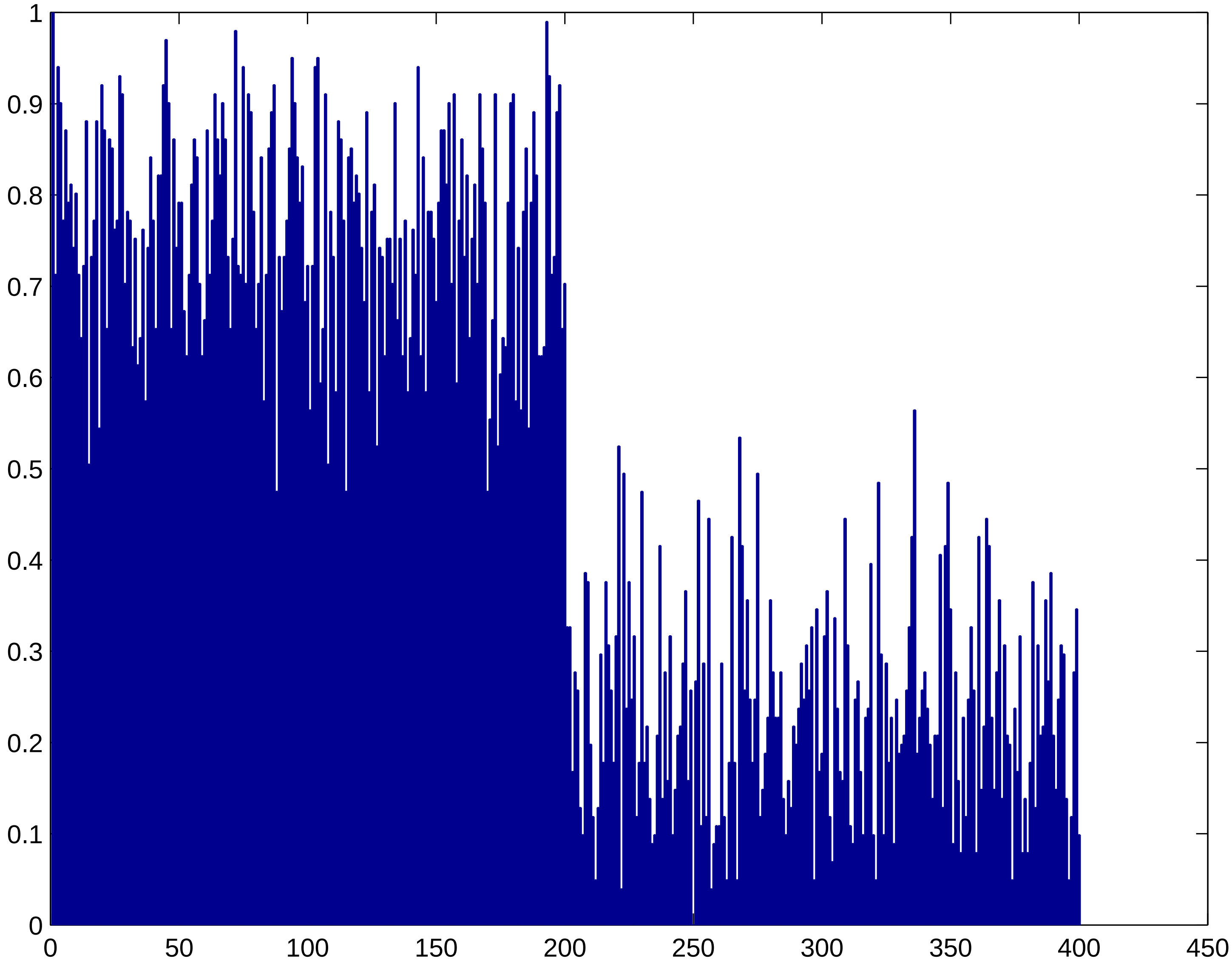
Max-Cut



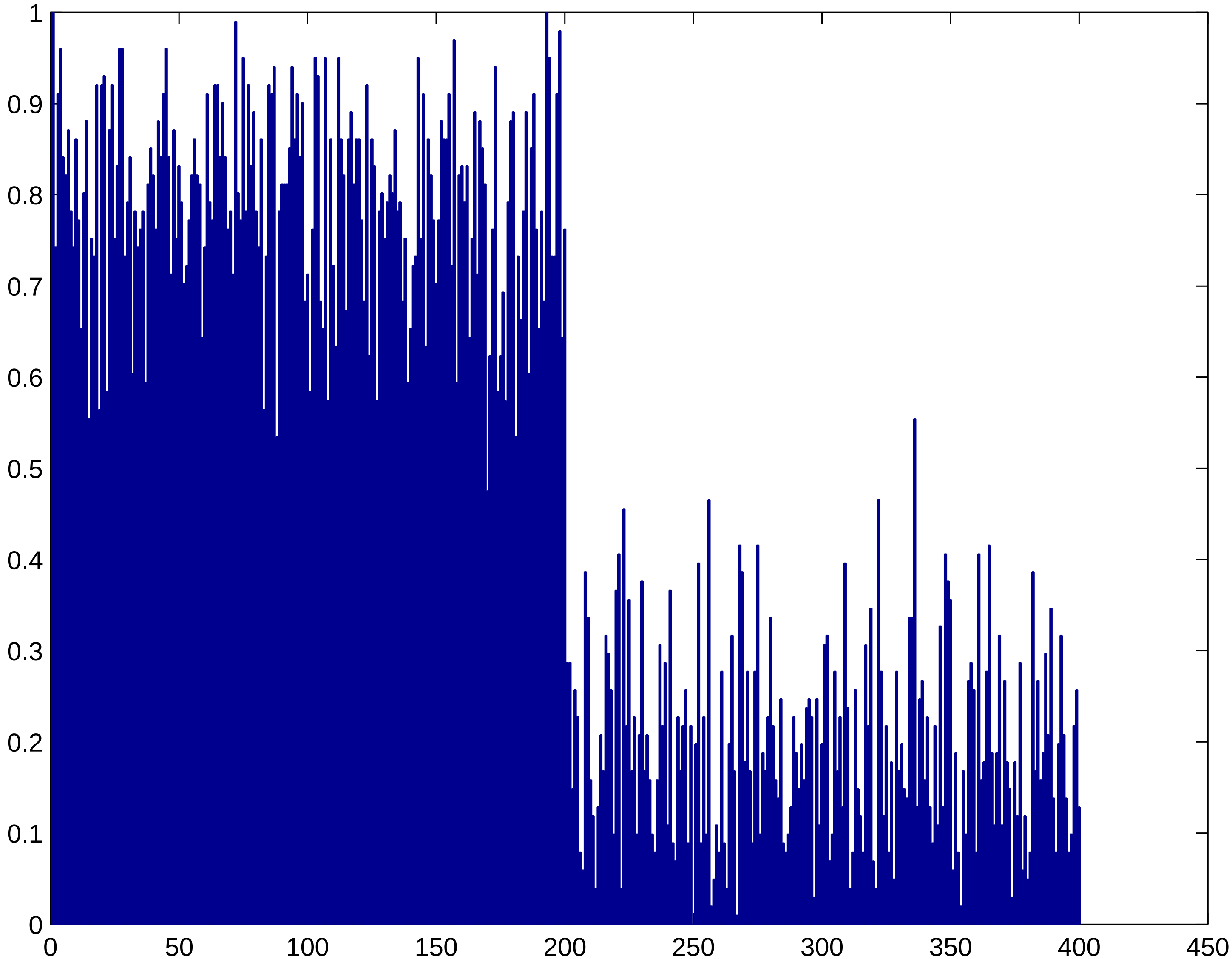
Max-Cut



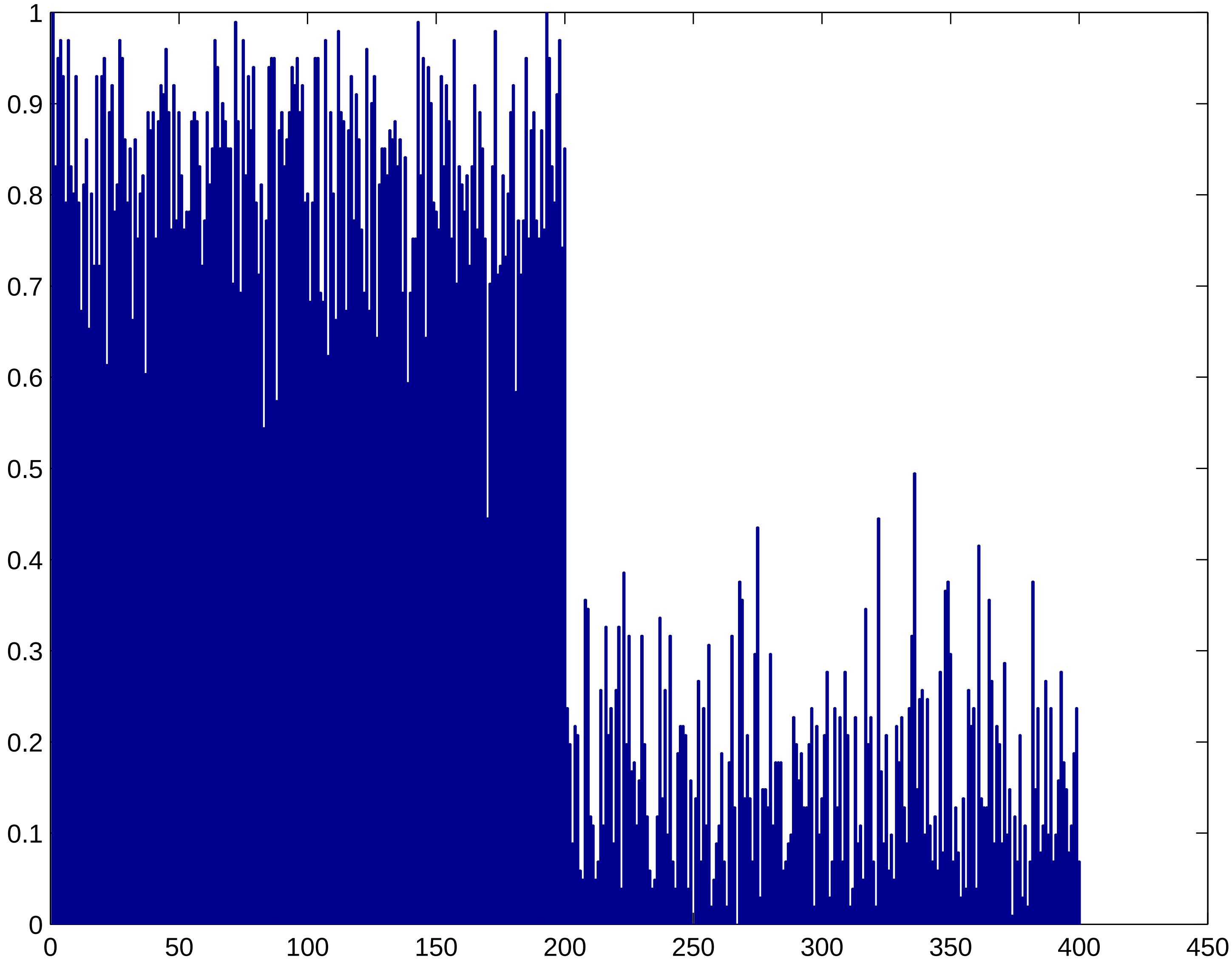
Max-Cut



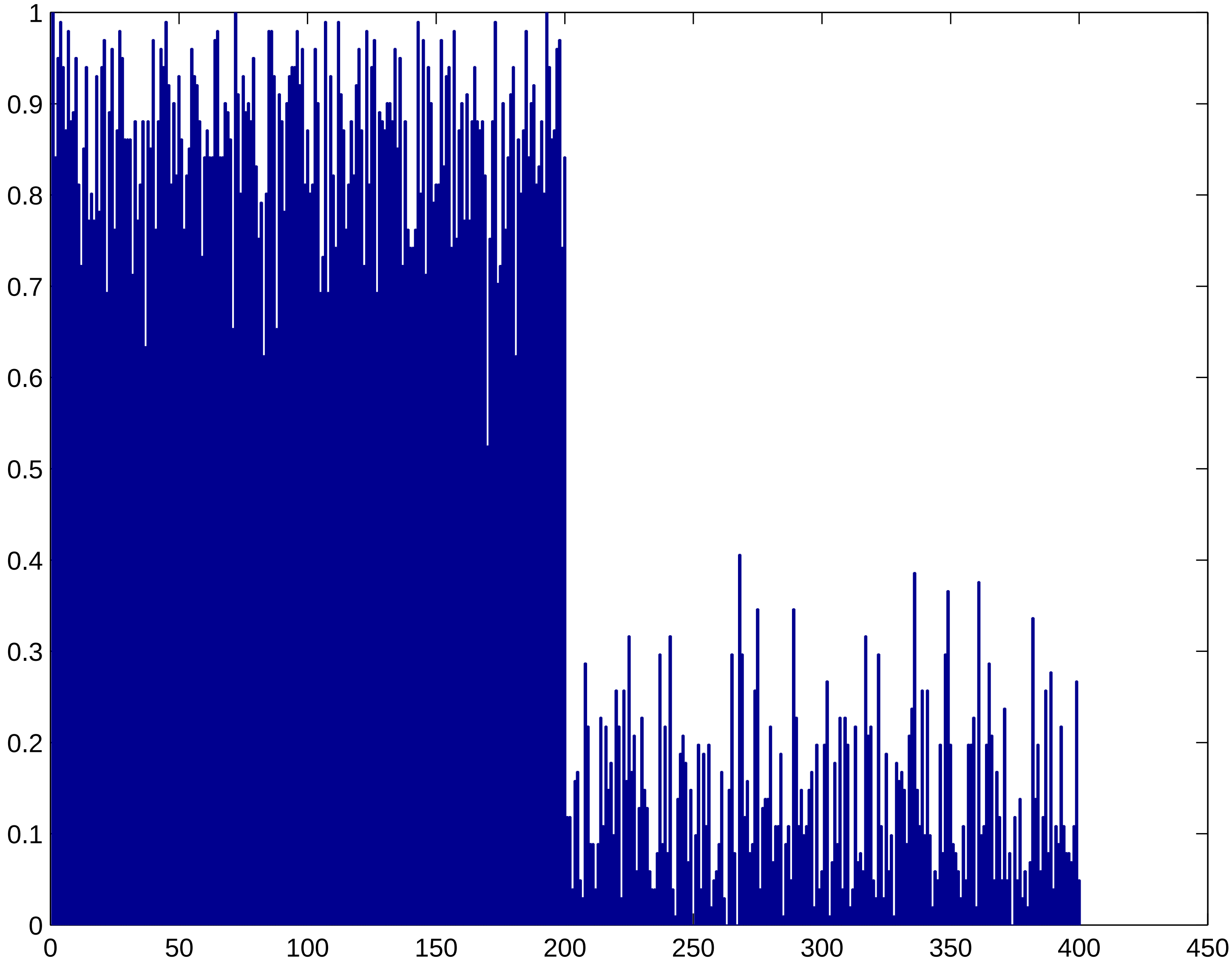
Max-Cut



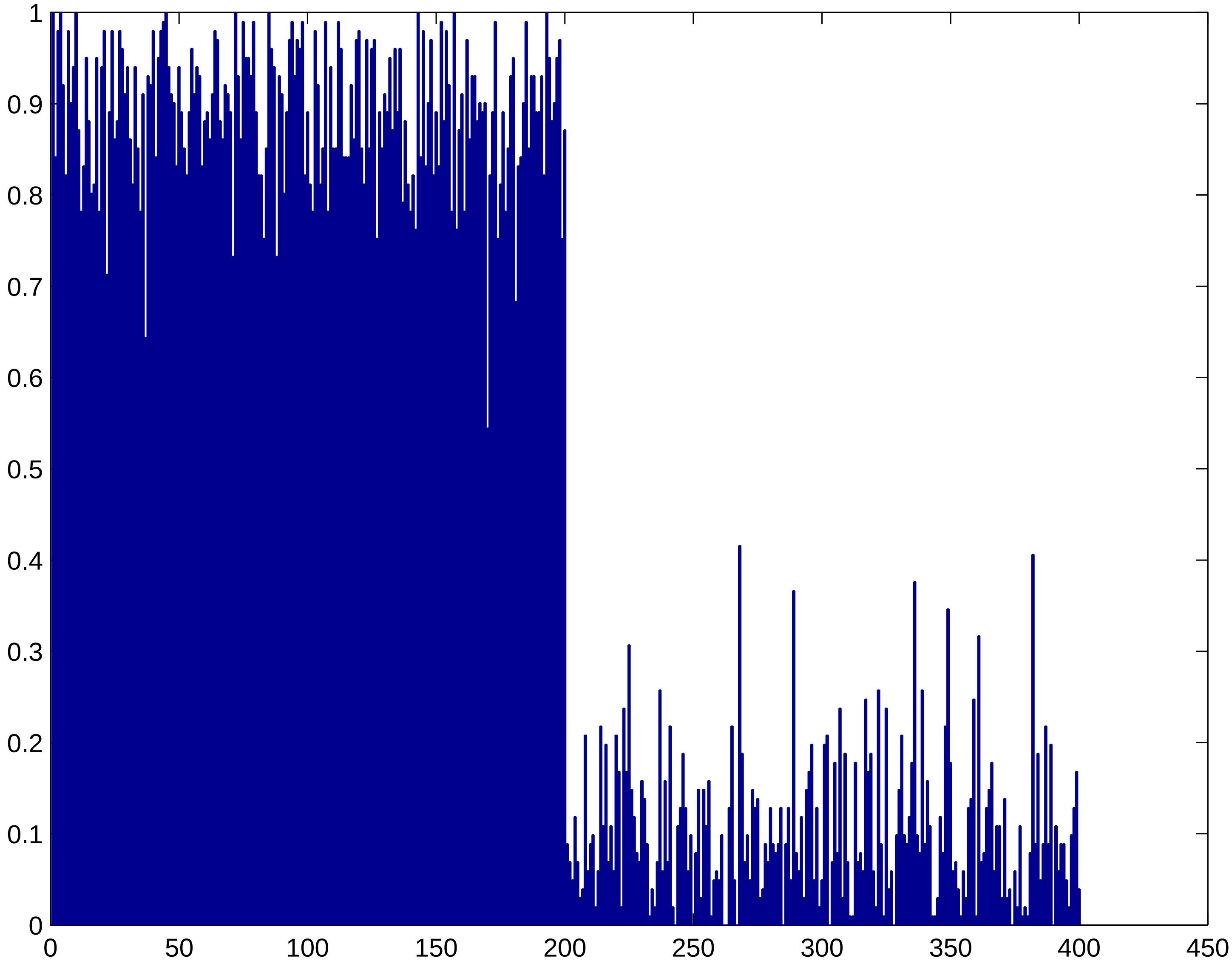
Max-Cut



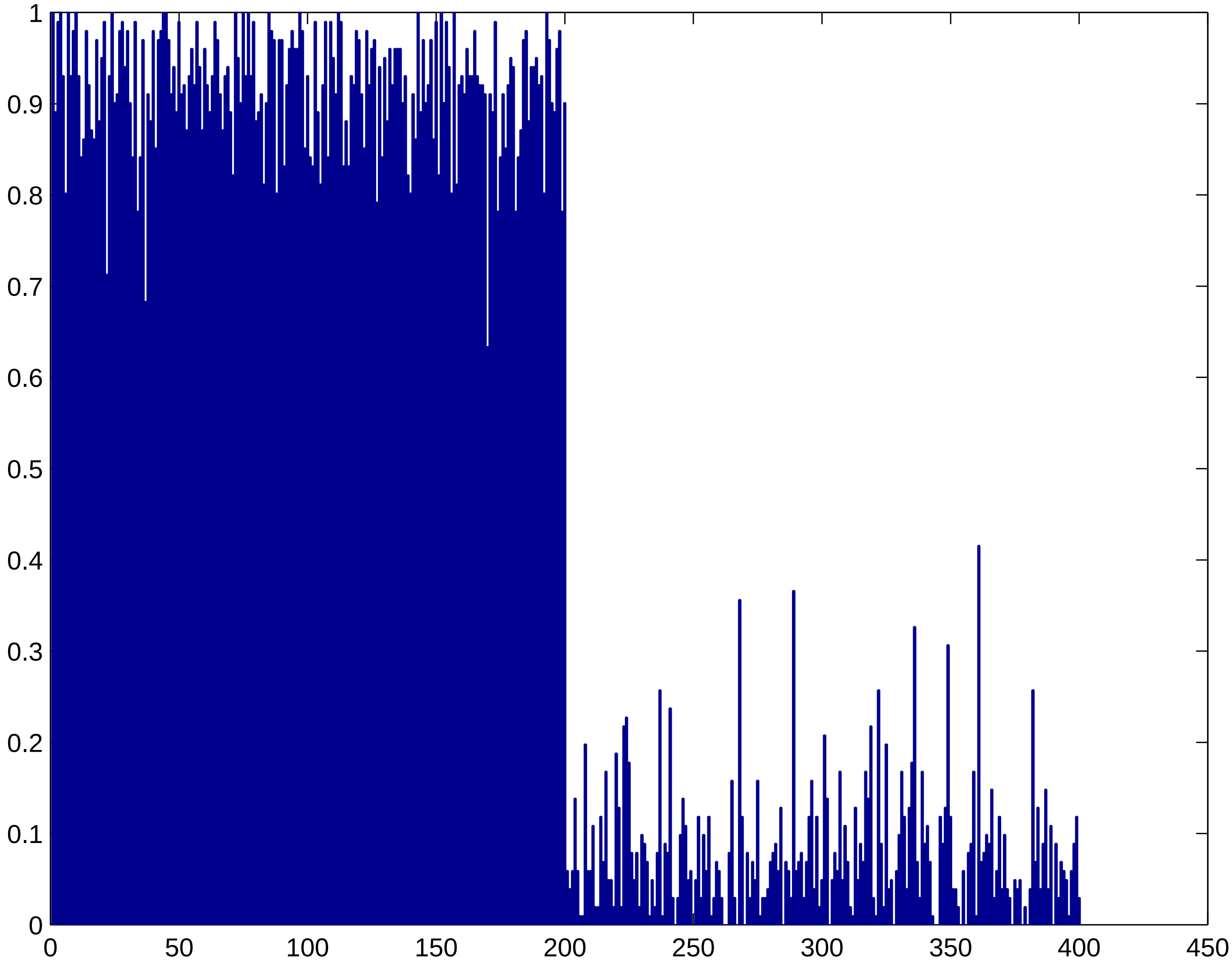
Max-Cut



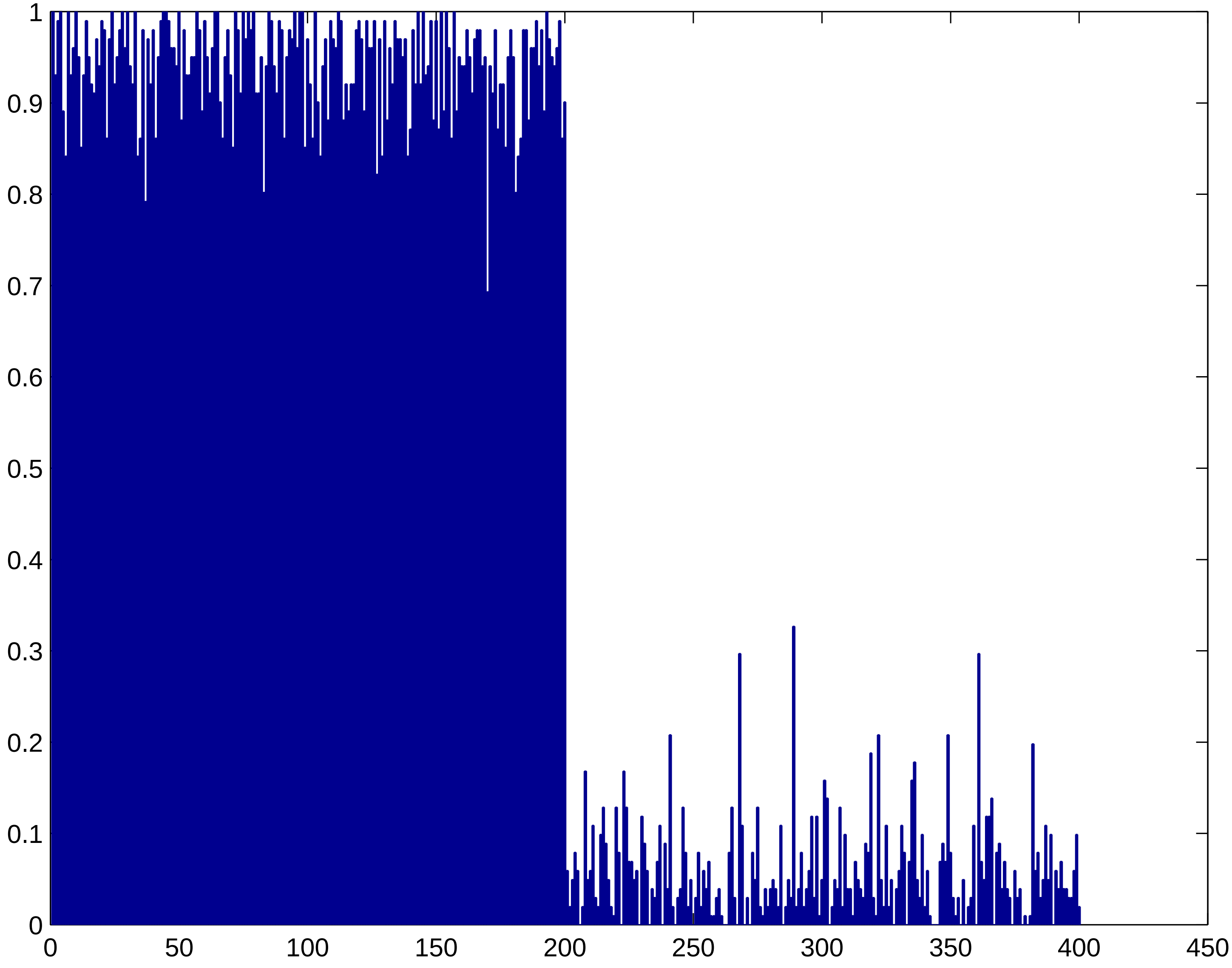
Max-Cut



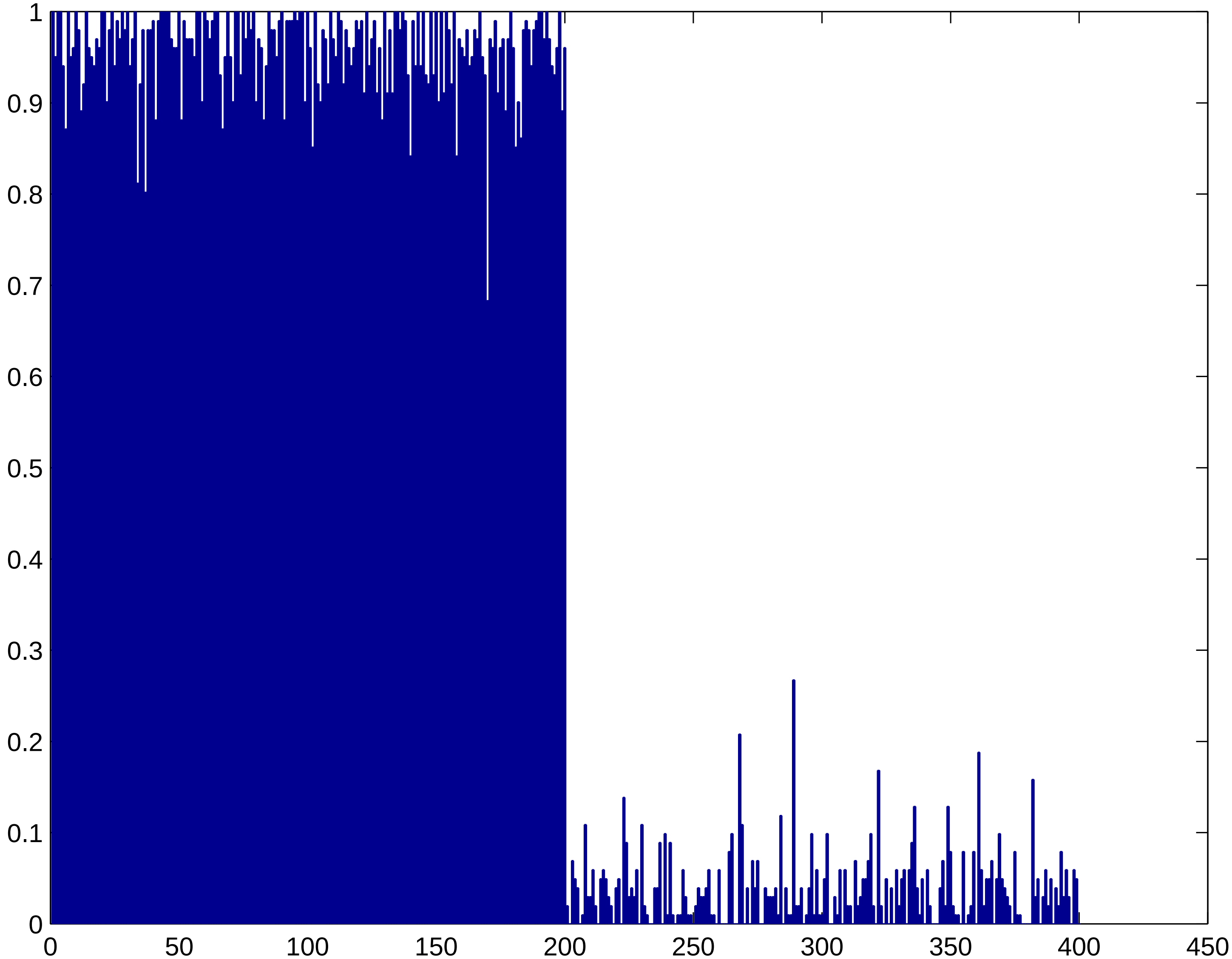
Max-Cut



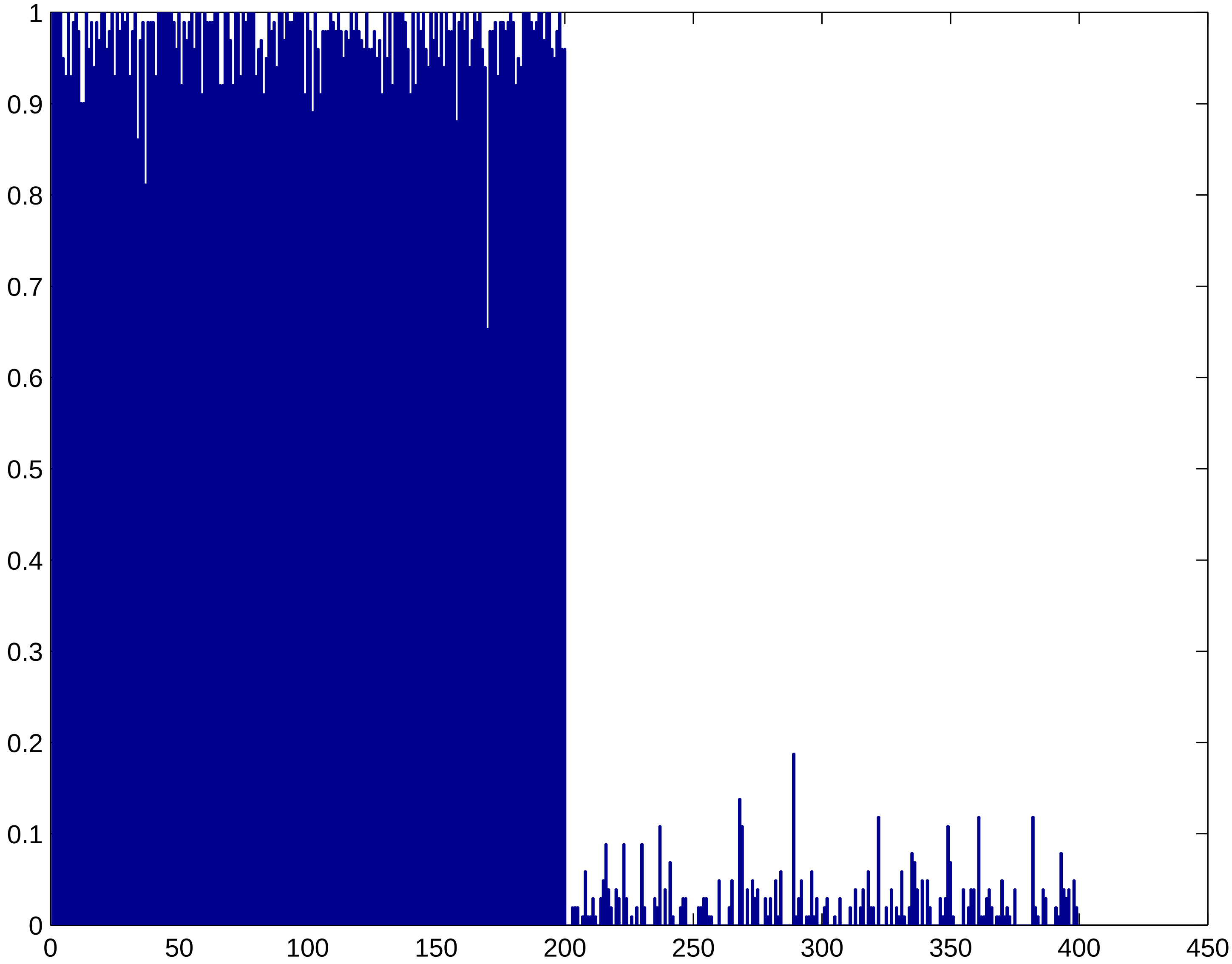
Max-Cut



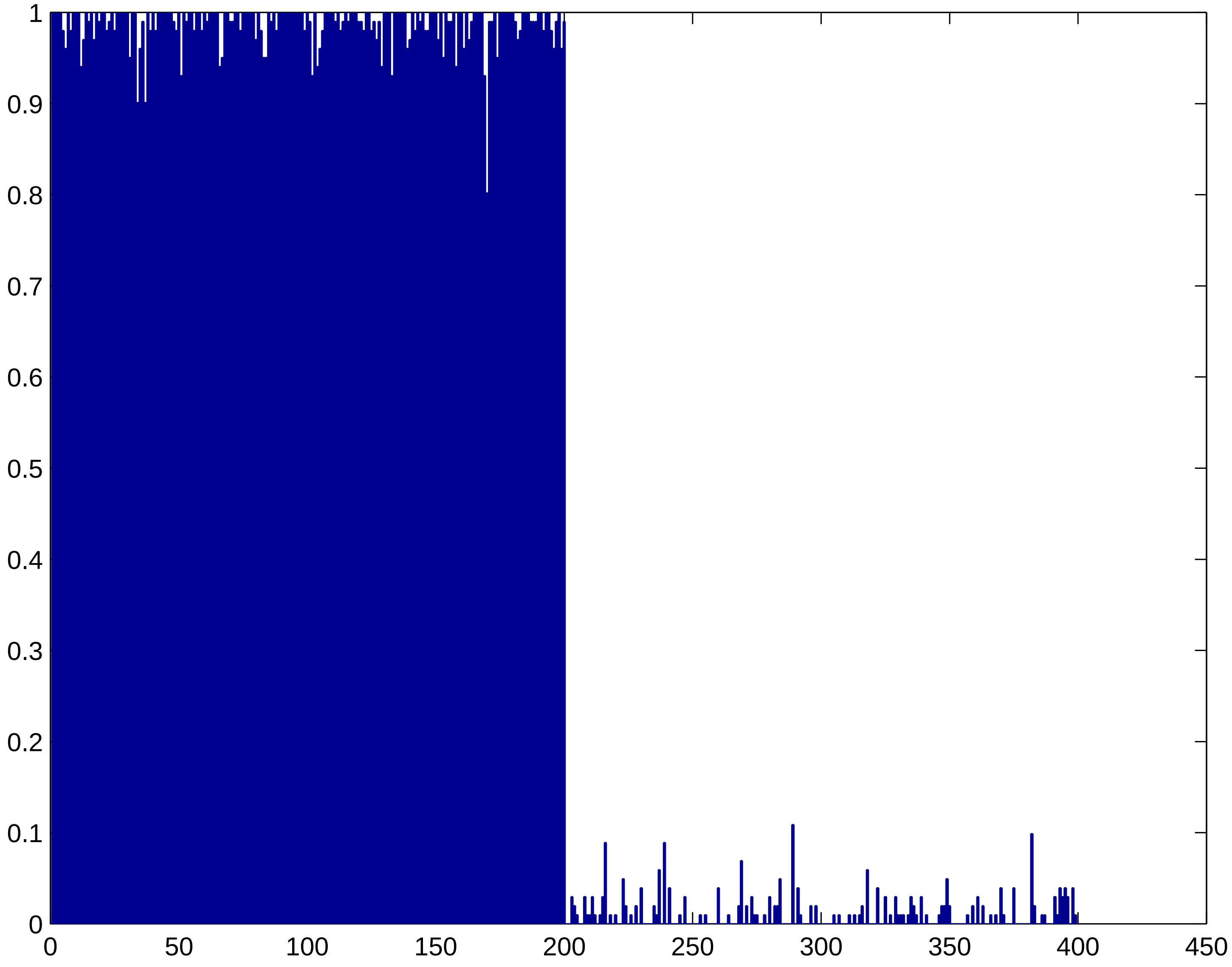
Max-Cut



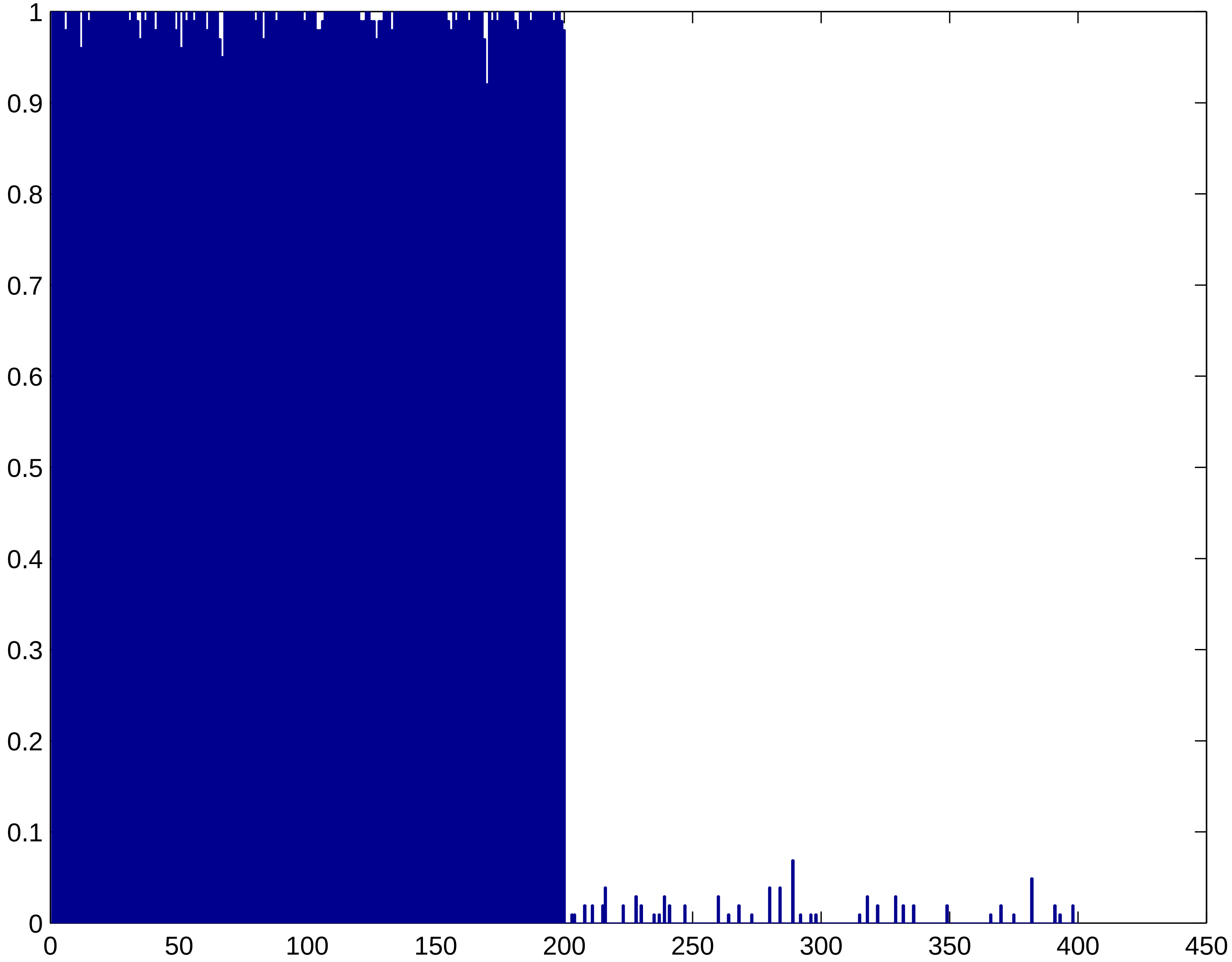
Max-Cut



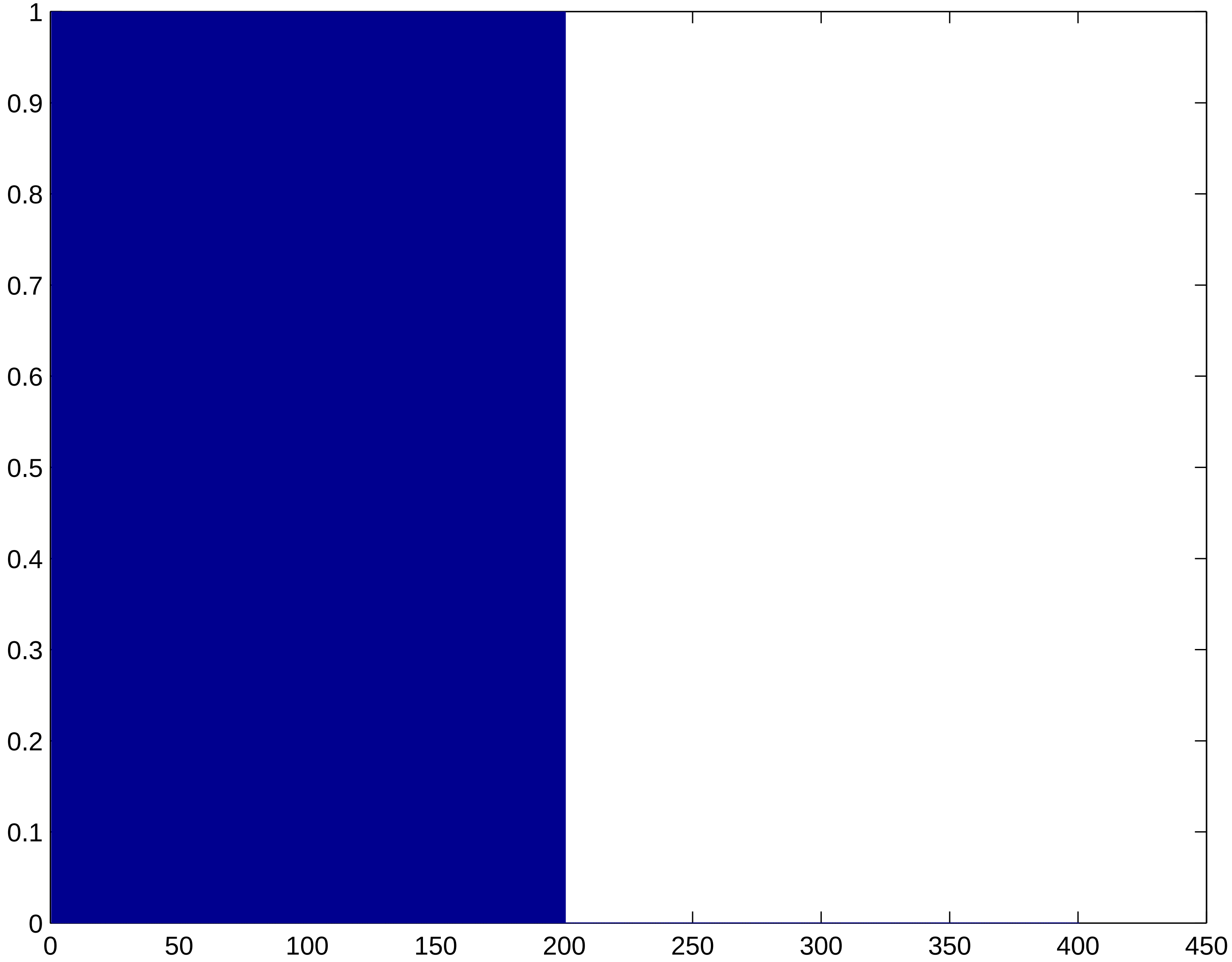
Max-Cut



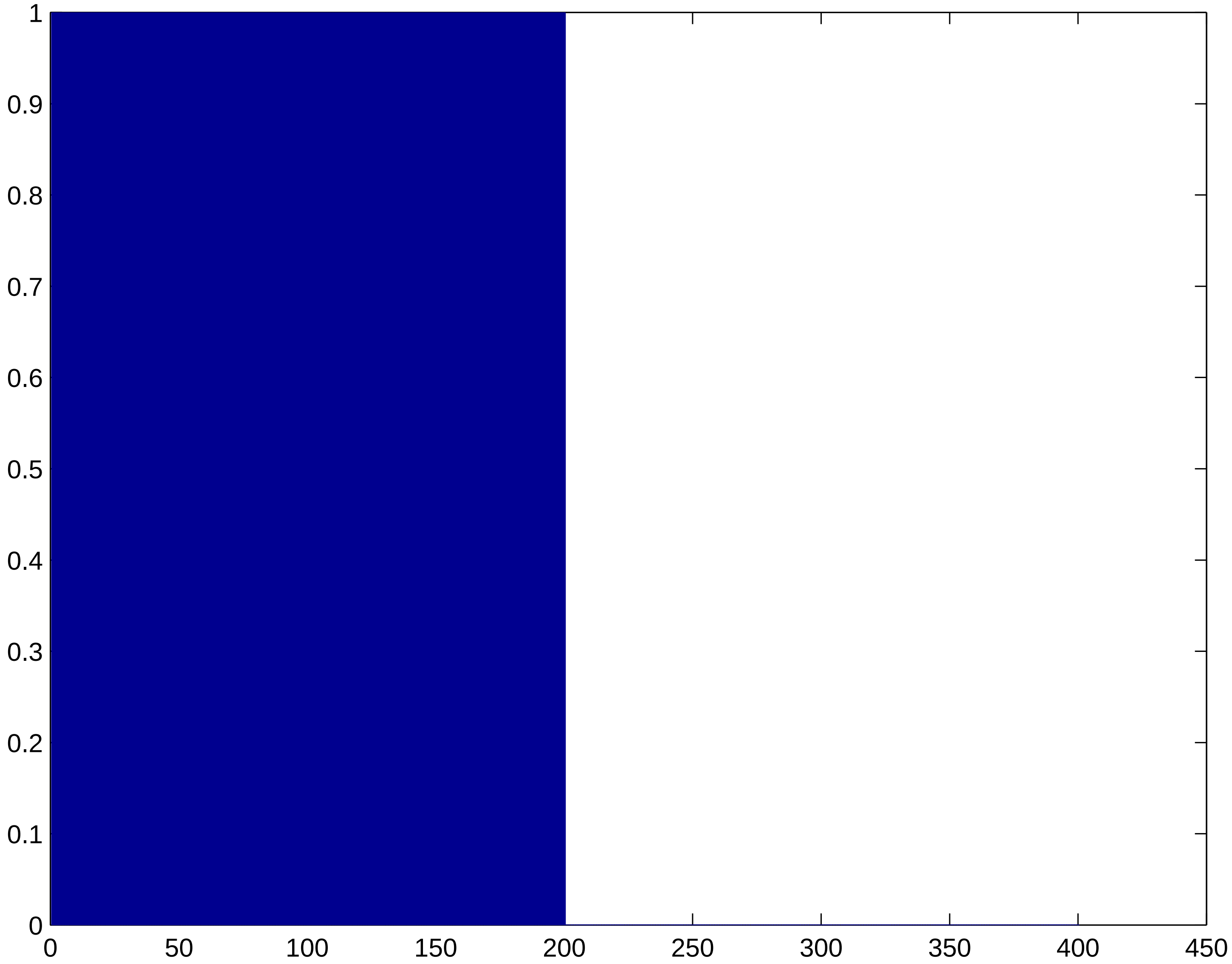
Max-Cut



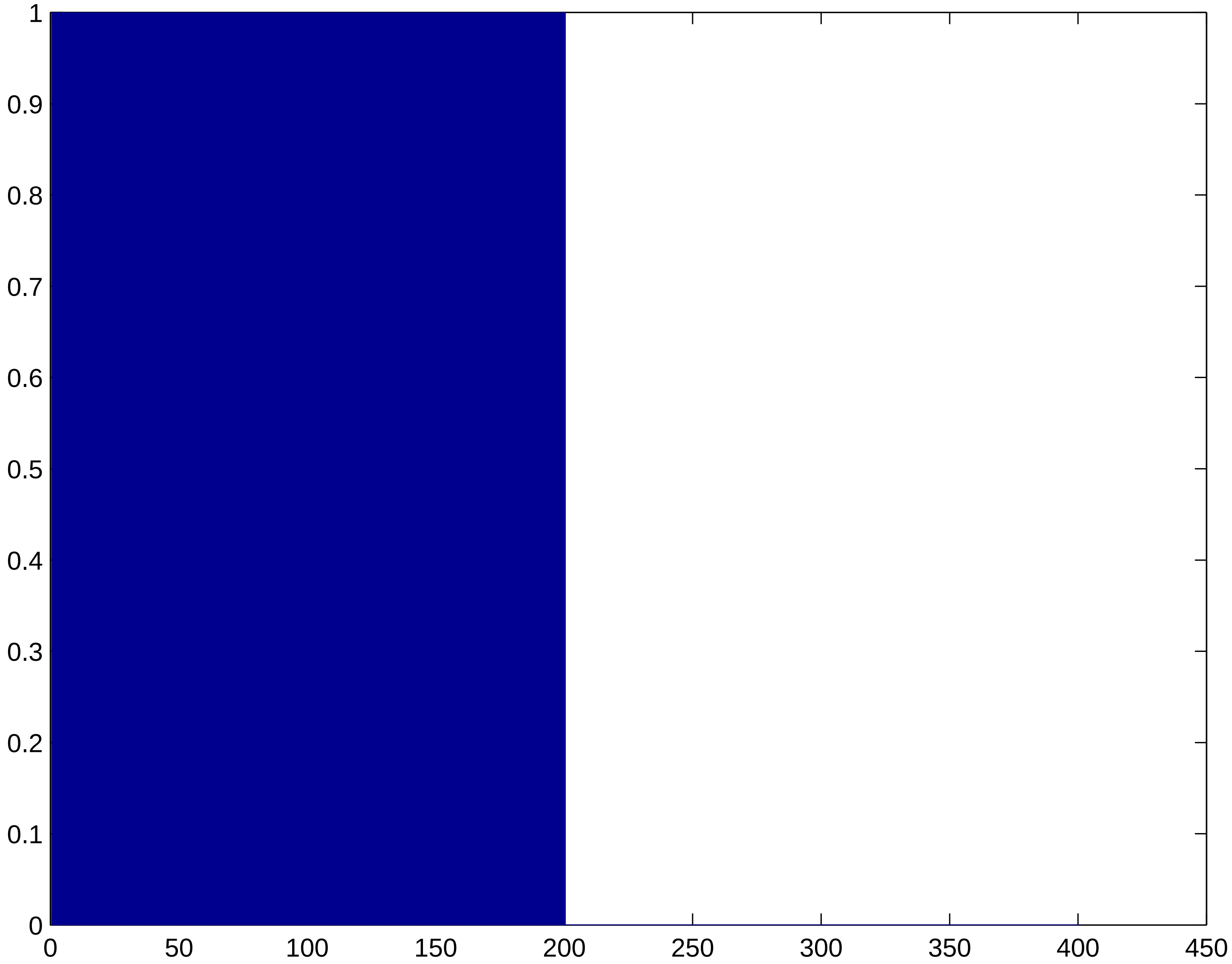
Max-Cut



Max-Cut



Max-Cut



Further research

- Convergence of CE algorithm.

Further research

- Convergence of CE algorithm.
- Noisy optimization.

Further research

- Convergence of CE algorithm.
- Noisy optimization.
- Simulation with heavy tail distributions.

Further research

- Convergence of CE algorithm.
- Noisy optimization.
- Simulation with heavy tail distributions.
- Markov Decision Processes.

THANK YOU