LP techniques for MAX-SAT and scheduling

Siert Wieringa

Department of Information and Computer Science
Helsinki University of Technology

March 13, 2008

Maximum Satisfiability - Outline

Goal: A factor $\frac{3}{4}$ approximation algorithm for MAX-SAT.

- Problem definitions.
- A factor $\frac{1}{2}$ approximation algorithm for MAX-SAT.
- A factor $1 - \frac{1}{\mathrm{e}}$ approximation algorithm for MAX-SAT
- Combining the two to obtain the approximation factor $\frac{3}{4}$.

Maximum Satisfiability Problem (MAX-SAT)

Given a conjunctive normal form formula $\mathcal{F}$ on Boolean variables $x_1, \ldots, x_n$, and non-negative weights, w_c, for each clause c of $\mathcal{F}$, find a truth assignment to the variables that maximizes the total weight of satisfied clauses.

- $\mathcal{F} = \bigwedge_{c \in C} c$ with C the set of clauses.
- size(c) is the length of clause c.
- MAX-SAT is NP-hard.

Restricted MAX-SAT: MAX-kSAT

- For any positive integer k, MAX-kSAT is the restriction of MAX-SAT to instances in which each clause has size(c) $\leq$ k.
- MAX-2SAT is NP-hard.
- Note that 2SAT is in P !

Common notation

- Random variable W denotes the total weight of satisfied clauses.
- Random variable W_c denotes the weight contributed by c to W.
- $W = \sum_{c \in C} W_c$

$$E[W_c] = w_c \cdot \Pr[c \text{ is satisfied}]$$

"Trivial" randomized algorithm

Generate a variable assignment τ in which each variable is set to True with probability $\frac{1}{2}$, or False otherwise.

Lemma

If $\mathrm{size(c)} = \mathrm{k}$ and $\alpha_\mathrm{k} = 1 - 2^{-\mathrm{k}}$ then $\mathrm{E}[\mathrm{W_c}] = \alpha_\mathrm{k} \mathrm{w_c}$

Proof.

Clause c is not satisfied by τ if all of its literals are set to False. The probability of that event is α_k $(= \ 1 - 2^{-\mathrm{k}})$ □

Expectation lowerbound for the randomized algorithm

For $k \geq 1$ it holds that $\alpha_k = 1 - 2^{-k} \geq \frac{1}{2}$, so:

$$E[W] = \sum_{c \in C} E[W_c] \geq \frac{1}{2} \sum_{c \in C} w_c \geq \frac{1}{2} OPT$$

Note that if each clause has size(c) ≥ 2 then $E[W] \geq \frac{3}{4} OPT$.

Derandomizing the suggested algorithm (1)

Lemma

The conditional expectation of any node in the self-reducibility tree of $\mathcal{F}$ can be computed in polynomial time.

Proof.

Consider a node corresponding to partial assignment $x_1 = a_1, \ ..., \ x_i = a_i$. Let ϕ be the Boolean formula on variables $x_{i+1}, ..., x_n$ obtained for this node. The expected weight of satisfied clauses of ϕ under a random truth assignment can be computed in polynomial time. Adding to this the total weight of clauses of $\mathcal{F}$ already satisfied by the partial assignment $x_1 = a_1, \ ..., \ x_n = a_n$ gives the answer. □

Derandomizing the suggested algorithm (2)

Lemma

A path from the root to a leaf such that the conditional expectation of each node on this path is $\geq$ E[W] can be calculated in polynomial time.

Proof.

The conditional expectation of a node is the average conditional expectiation of its children. So, the child with the highest conditional expectation of the two has a conditional expectation at least as large as its parent. This proofs that the desired path exists. By the previous lemma we can compute the conditional expectation of any node in the tree in polynomial time this lemma follows. □

The derandomization of the suggested algorithm

1. Compute a path from the root to a leaf such that the conditional expectation of each node on this path is $\geq$ E[W].

2. Output the truth assignment τ found at the leaf node of the computed path.

Definitions for an integer program for MAX-SAT

We will continue to introduce and integer program for MAX-SAT and a second randomized algorithm using its LP-relaxation.

For all $c \in C$:

- Let S_c^+ be the set of variables occuring nonnegated in c.
- Let S_c^- be the set of variables occuring negated in c.

Let $y_i = 1$ denote setting $x_i =$ True, and $y_i = 0$ denote $x_i =$ False.

An integer program for MAX-SAT

$$\begin{aligned}
\text{maximize} \quad & \textstyle\sum_{c \in C} w_c z_c \\
\text{subject to} \quad & \forall c \in C: \quad \textstyle\sum_{i \in S_c^+} y_i + \sum_{i \in S_c^-} (1 - y_i) \geq z_c \\
& \forall c \in C: \quad z_c \in \{0, 1\} \\
& \forall i : \quad y_i \in \{0, 1\}
\end{aligned}$$

LP-Relaxation of the integer program for MAX-SAT

$$\text{maximize} \quad \sum_{c \in C} w_c z_c$$

$$\begin{aligned} \text{subject to} \quad & \forall c \in C: && \sum_{i \in S_c^+} y_i + \sum_{i \in S_c^-} (1 - y_i) \geq z_c \\ & \forall c \in C: && 1 \geq z_c \geq 0 \\ & \forall i: && 1 \geq y_i \geq 0 \end{aligned}$$

Using the LP relaxation

The second randomized algorithm for MAX-SAT:

- Solve the LP program introduced on the previous slide.
- Let (y^*, z^*) denote the optimal solution.
- Set x_i to True with probability y_i^* for $1 \leq i \leq n$.
- Output the resulting truth assignment τ.

Expectation lowerbound for the second randomized algorithm (1)

Recall the random variables W and W_c:

- $E[W_c] = w_c \cdot \Pr[c \text{ is satisfied}]$
- $W = \sum_{c \in C} W_c$

Define for $k > 1$:

$$\beta_k = 1 - \left(1 - \frac{1}{k}\right)^k$$

Expectation lowerbound for the second randomized algorithm (2)

Lemma

If $\mathrm{size}(c) = k$ then $E[W_c] \geq \beta_k w_c z_c^*$.

Proof

Without loss of generality assume that all literals in clause c occur nonnegated. By renaming variables assume that $c = (x_1 \vee ... \vee x_k)$. Clause c is satisfied if not all x_1, ..., x_n are set to False.

Expectation lowerbound for the second randomized algorithm (3)

Proof (continued).

The probability of c being satisfied is:

$$1 - \prod_{i=1}^{k} (1 - y_i) \geq 1 - \left(\frac{\sum_{i=1}^{k} (1 - y_i)}{k} \right)^k \geq 1 - \left(1 - \frac{z_c^*}{k} \right)^k$$

The arithmic-geometric mean inequality states for nonnegative $a_1, \ldots, a_k$:

$$\frac{a_1 + \ldots + a_k}{k} \geq \sqrt[k]{a_1 \cdot \ldots \cdot a_k} \qquad \left(\frac{a_1 + \ldots + a_k}{k} \right)^k \geq a_1 \cdot \ldots \cdot a_k$$

Expectation lowerbound for the second randomized algorithm (3)

Proof (continued).

The probability of c being satisfied is:

$$1 - \prod_{i=1}^{k}(1 - y_i) \geq 1 - \left(\frac{\sum_{i=1}^{k}(1 - y_i)}{k}\right)^k \geq 1 - \left(1 - \frac{z_c^*}{k}\right)^k$$

The arithmic-geometric mean inequality states for nonnegative $a_1, \ldots, a_k$:

$$\frac{a_1 + \ldots + a_k}{k} \geq \sqrt[k]{a_1 \cdot \ldots \cdot a_k} \qquad \left(\frac{a_1 + \ldots + a_k}{k}\right)^k \geq a_1 \cdot \ldots \cdot a_k$$

Expectation lowerbound for the second randomized algorithm (3)

Proof (continued).

The probability of c being satisfied is:

$$1 - \prod_{i=1}^{k} (1 - y_i) \geq 1 - \left(\frac{\sum_{i=1}^{k} (1 - y_i)}{k} \right)^k \geq 1 - \left(1 - \frac{z_c^*}{k} \right)^k$$

The arithmic-geometric mean inequality states for nonnegative $a_1, \ldots, a_k$:

$$\frac{a_1 + \ldots + a_k}{k} \geq \sqrt[k]{a_1 \cdot \ldots \cdot a_k} \qquad \left(\frac{a_1 + \ldots + a_k}{k} \right)^k \geq a_1 \cdot \ldots \cdot a_k$$

Expectation lowerbound for the second randomized algorithm (3)

Proof (continued).

The probability of c being satisfied is:

$$1 - \prod_{i=1}^{k} (1 - y_i) \geq 1 - \left(1 - \frac{\sum_{i=1}^{k} y_i}{k}\right)^k \geq 1 - \left(1 - \frac{z_c^*}{k}\right)^k$$

The arithmic-geometric mean inequality states for nonnegative $a_1, \ldots, a_k$:

$$\frac{a_1 + \ldots + a_k}{k} \geq \sqrt[k]{a_1 \cdot \ldots \cdot a_k} \qquad \left(\frac{a_1 + \ldots + a_k}{k}\right)^k \geq a_1 \cdot \ldots \cdot a_k$$

Expectation lowerbound for the second randomized algorithm (3)

Proof (continued).

The probability of c being satisfied is:

$$1 - \prod_{i=1}^{k} (1 - y_i) \geq 1 - \left(1 - \frac{\sum_{i=1}^{k} y_i}{k}\right)^k \geq 1 - \left(1 - \frac{z_c^*}{k}\right)^k$$

The arithmic-geometric mean inequality states for nonnegative $a_1, \ldots, a_k$:

$$\frac{a_1 + \ldots + a_k}{k} \geq \sqrt[k]{a_1 \cdot \ldots \cdot a_k} \qquad \left(\frac{a_1 + \ldots + a_k}{k}\right)^k \geq a_1 \cdot \ldots \cdot a_k$$

Expectation lowerbound for the second randomized algorithm (4)

Proof (continued).

- The probability of c being satisfied is $\geq 1 - \left(1 - \frac{z_c^*}{k}\right)^k$.
- Consider a function g defined as $g(z) = 1 - \left(1 - \frac{z}{k}\right)^k$.
- Recall $\beta_k = 1 - \left(1 - \frac{1}{k}\right)^k$.
- For $z \in [0, 1]$ it holds that $g(z) \geq \beta_k z$.
- Therefore $\Pr[c \text{ is satisfied}] \geq \beta_k z_c^*$, so $E[W_c] \geq \beta_k w_c z_c^*$. □

Expectation lowerbound for the second randomized algorithm (5)

- $\mathrm{E}[\mathrm{W_c}] \geq \beta_\mathrm{k} \mathrm{w_c} \mathrm{z_c^*}$
- As β_k is a decreasing function of k:

$$\mathrm{E}[\mathrm{W}] = \sum_{\mathrm{c} \in \mathrm{C}} \mathrm{E}[\mathrm{W_c}] \geq \beta_\mathrm{k} \sum_{\mathrm{c} \in \mathrm{C}} \mathrm{w_c} \mathrm{z_c^*} = \beta_\mathrm{k} \mathrm{OPT_f} \geq \beta_\mathrm{k} \mathrm{OPT}.$$

- After derandomization this is a β_k factor approximation algorithm for MAX-kSAT.
- $\forall \mathrm{k} \in \mathrm{Z}^+ \ : \ \left(1 - \frac{1}{\mathrm{k}}\right)^\mathrm{k} > \frac{1}{\mathrm{e}}$
- So, this is a $1 - \frac{1}{\mathrm{e}}$ factor approximation algorithm for MAX-SAT.

A randomized $E[W] \geq \frac{3}{4}OPT$ algorithm

Let b be the flip of a fair coin. If $b = 0$ use the first randomized algorithm, if $b = 1$ use the second randomized algorithm.

- $E[W_c \mid b = 0] \geq \alpha_k w_c \geq \alpha_k w_c z_c^*$.
- $E[W_c \mid b = 1] \geq \beta_k w_c z_c^*$.

$$E[W_c] = \frac{1}{2}(E[W_c \mid b = 0] + E[W_c \mid b = 1]) \geq w_c z_c^* \frac{\alpha_k + \beta_k}{2}.$$

$\alpha_1 + \beta_1 = \alpha_2 + \beta_2 = \frac{3}{2}$, and for $k \geq 3$:

$\alpha_k + \beta_k \geq \frac{7}{8} + (1 - \frac{1}{e}) \geq \frac{3}{2}$. $E[W] \geq \frac{3}{4}OPT$.

A randomized E[W] $\geq \frac{3}{4}$OPT algorithm

Let b be the flip of a fair coin. If $b = 0$ use the first randomized algorithm, if $b = 1$ use the second randomized algorithm.

- $E[W_c \mid b = 0] \geq \alpha_k w_c \geq \alpha_k w_c z_c^*$.
- $E[W_c \mid b = 1] \geq \beta_k w_c z_c^*$.

$$E[W_c] = \frac{1}{2}(E[W_c \mid b = 0] + E[W_c \mid b = 1]) \geq w_c z_c^* \frac{\alpha_k + \beta_k}{2}.$$

$\alpha_1 + \beta_1 = \alpha_2 + \beta_2 = \frac{3}{2}$, and for $k \geq 3$:

$\alpha_k + \beta_k \geq \frac{7}{8} + (1 - \frac{1}{e}) \geq \frac{3}{2}$. $E[W] \geq \frac{3}{4}$OPT.

A randomized E[W] $\geq \frac{3}{4}$OPT algorithm

Let b be the flip of a fair coin. If $b = 0$ use the first randomized algorithm, if $b = 1$ use the second randomized algorithm.

- $E[W_c \mid b = 0] \geq \alpha_k w_c \geq \alpha_k w_c z_c^*$.
- $E[W_c \mid b = 1] \geq \beta_k w_c z_c^*$.

$$E[W_c] = \frac{1}{2}(E[W_c \mid b = 0] + E[W_c \mid b = 1]) \geq w_c z_c^* \frac{\alpha_k + \beta_k}{2}.$$

$\alpha_1 + \beta_1 = \alpha_2 + \beta_2 = \frac{3}{2}$, and for $k \geq 3$:

$\alpha_k + \beta_k \geq \frac{7}{8} + (1 - \frac{1}{e}) \geq \frac{3}{2}$. $E[W] \geq \frac{3}{4}$OPT.

A randomized E[W] $\geq \frac{3}{4}$OPT algorithm

Let b be the flip of a fair coin. If $b = 0$ use the first randomized algorithm, if $b = 1$ use the second randomized algorithm.

- $E[W_c \mid b = 0] \geq \alpha_k w_c \geq \alpha_k w_c z_c^*$.
- $E[W_c \mid b = 1] \geq \beta_k w_c z_c^*$.

$$E[W_c] = \frac{1}{2}(E[W_c \mid b = 0] + E[W_c \mid b = 1]) \geq w_c z_c^* \frac{\alpha_k + \beta_k}{2}.$$

$\alpha_1 + \beta_1 = \alpha_2 + \beta_2 = \frac{3}{2}$, and for $k \geq 3$:

$\alpha_k + \beta_k \geq \frac{7}{8} + (1 - \frac{1}{e}) \geq \frac{3}{2}$. $E[W] \geq \frac{3}{4}$OPT.

A randomized E[W] $\geq \frac{3}{4}$OPT algorithm

Let b be the flip of a fair coin. If $b = 0$ use the first randomized algorithm, if $b = 1$ use the second randomized algorithm.

- $E[W_c \mid b = 0] \geq \alpha_k w_c \geq \alpha_k w_c z_c^*$.
- $E[W_c \mid b = 1] \geq \beta_k w_c z_c^*$.

$$E[W_c] = \frac{1}{2}(E[W_c \mid b = 0] + E[W_c \mid b = 1]) \geq w_c z_c^* \frac{\alpha_k + \beta_k}{2}.$$

$\alpha_1 + \beta_1 = \alpha_2 + \beta_2 = \frac{3}{2}$, and for $k \geq 3$:

$\alpha_k + \beta_k \geq \frac{7}{8} + (1 - \frac{1}{e}) \geq \frac{3}{2}$. $E[W] \geq \frac{3}{4}$OPT.

A randomized E[W] $\geq \frac{3}{4}$OPT algorithm

Let b be the flip of a fair coin. If $b = 0$ use the first randomized algorithm, if $b = 1$ use the second randomized algorithm.

- $E[W_c \mid b = 0] \geq \alpha_k w_c \geq \alpha_k w_c z_c^*$.
- $E[W_c \mid b = 1] \geq \beta_k w_c z_c^*$.

$$E[W_c] = \frac{1}{2}(E[W_c \mid b = 0] + E[W_c \mid b = 1]) \geq w_c z_c^* \frac{\alpha_k + \beta_k}{2}.$$

$\alpha_1 + \beta_1 = \alpha_2 + \beta_2 = \frac{3}{2}$, and for $k \geq 3$:

$\alpha_k + \beta_k \geq \frac{7}{8} + (1 - \frac{1}{e}) \geq \frac{3}{2}$. $\qquad$ $E[W] \geq \frac{3}{4}$OPT.

A deterministic factor $\frac{3}{4}$ approximation algorithm

1. Use the derandomized factor $\frac{1}{2}$ algorithm to get a truth assignment τ_1.

2. Use the derandomized factor $1 - \frac{1}{\mathrm{e}}$ algorithm to get a truth assignment τ_2.

3. Output the better of the two assignments.

One of the two conditional expressions $\mathrm{E}[\mathrm{W} \mid \mathrm{b} = 0]$ and $\mathrm{E}[\mathrm{W} \mid \mathrm{b} = 1]$ is at least as large as $\mathrm{E}[\mathrm{W}]$. So, the total weight of clauses satisfied by the better of τ_1 and τ_2 is at least $\mathrm{E}[\mathrm{W}]$.

Scheduling - Outline

- Problem defintions.
- Relation with Petri's earlier talk.
- Problematic LP relaxation of integer program.
- Useful graph theory concepts.
- Proof of approximation factor 2.

Scheduling on unrelated parallel machines

Given:

- A set J of jobs.
- A set M of machines.
- A $p_{ij} \in Z^+$ for each $j \in J$ and $i \in M$.

Where p_{ij} is the time it takes to execute job j on machine i.

Schedule the jobs on the machines such that the makespan, e.g. the maximum processing time of any machine, is minimized.

Didn't we study this before?

No, we didn't...

- These machines are unrelated as there is no relation between how long a job takes on one machine, and how long it might take on another machine.

- If for all $i \in M$ the value of p_{ij} is the same then the machines are said to be identical. For the restricted problem of scheduling on identical machines a PTAS exists as was discussed by Petri.

- For the generalization of minimum makespan to uniform machines a PTAS also exists. In that case all machines have a speed s_i associated with them and the processing time for job j on machine i is $\frac{p_j}{s_i}$.

An integer program

Let x_{ij} denote whether job j is scheduled on machine i.

$$\begin{aligned} \text{minimize} \quad & t \\ \\ \text{subject to} \quad & \textstyle\sum_{i \in M} x_{ij} = 1, && j \in J \\ & \textstyle\sum_{j \in J} x_{ij} p_{ij} \leq t, && i \in M \\ & x_{ij} \in \{0, 1\}, && i \in M,\ j \in J \end{aligned}$$

This integer program has unbounded integrality gap.

Unbounded integrality gap - example

Suppose there is only one job, with processing time m on each of the m machines. The minimum makespan is thus m.

The optimal solution to the linear relaxation of the suggested integer program is to schedule a fraction $\frac{1}{m}$ of the job on each of the m machines, which leads to $t = 1$. The integrality gap is thus m.

Bounding the integrality gap

- The integrality gap of the LP relaxation would be bounded if we could add the following constraint:

$$\forall i \in M \; j \in J\text{: if } p_{ij} > t \text{ then } x_{ij} = 0.$$

- That is unfortunately not possible as it is not a linear constraint.

- We will parametrize the integer program and use parametric pruning instead.

- Parameter $T \in Z^+$ a guess for a lower bound on the optimal makespan.

Parametrized integer program

Define: $S_T = \{ (i,j) \mid p_{ij} \leq T \}$

And a family of linear programs LP(T):

$$\sum_{i:(i,j)\in S_T} x_{ij} = 1, \qquad j \in J$$

$$\sum_{j:(i,j)\in S_T} x_{ij} p_{ij} \leq T, \qquad i \in M$$

$$x_{ij} \leq 0, \qquad (i,j) \in S_T$$

Using binary search we can find the smallest value of T for which LP(T) has a feasible solution. Let this value be T^*.

Using extreme point solutions (1)

- Let $r = |S_T|$, the number of variables on which LP(T) is defined.
- A feasible solution to LP(T) is an extreme point solution iff it corresponds to setting r lineary independent constraints to equality.

Lemma

Any extreme point solution to LP(T) has at most $n + m$ nonzero variables.

Proof.

An extreme point solution has r lineary independend constraint to equality. Of these at least $r - (n + m)$ must be of the form $x_{ij} \leq 0$. The corresponding variables must be set to 0. So, at most $n + m$ variables might be set to a nonzero value. □

Using extreme point solutions (2)

Lemma

Any extreme point solution to LP(T) must set at least $n - m$ jobs integrally.

Proof.

- Let x be an extreme point solution to LP(T).
- Let α be the number of jobs set integrally by x.
- Let β be the number of jobs set fractionally by x.
- Each of the β jobs is assigned to at least two machines and therefore results in at least two nonzero entries in x.

$$\alpha + \beta = n \qquad \alpha + 2\beta \leq n + m$$

- Therefore, $\beta \leq m$ and $\alpha \geq n - m$. □

LP-Rounding algorithm: Graph construction

Consider extreme point solution x in LP(T).

- Define $G = (J, M, E)$ to be the biparite graph on vertex $J \cup M$ such that $(j, i) \in E$ iff $x_{ij} \neq 0$.
- Let $F \subset J$ be the set of jobs that are fractionally set in x.
- Let H be the subgraph of G induced on vertices $F \cup M$.
- So, (i, j) is an edge in H if $0 < x_{ij} < 1$.
- Graph H has a perfect matching, that is a matching that matches every job $j \in F$ (proof follows).

Approximation algortithm

Let α be the makespan found by a greedy algorithm.

1. By a binary search in the interval $[\frac{\alpha}{\mathrm{m}}, \alpha]$, find the smallest value of $\mathrm{T} \in \mathrm{Z}^+$ for which LP(T) has a feasible solution. Let this value be T^*.

2. Find an extreme point solution x to LP(T^*).

3. Assign all integrally set jobs to machines as in x.

4. Construct the graph H and find a perfect matching M in it (construction follows).

5. Assign fractionally set jobs to machines according to matching M.

Graph theory definitions

- A connected graph on a vertex set V is a pseudo-tree if it contains at most |V| edges.
- So, it is either a tree or a tree plus one edge.
- A tree plus one edge has unique cycle.
- A graph is a pseudo-forest if each of its connected components is a pseudo-tree.

Lemma on graph G

Lemma

Graph G is a pseudo-forest.

Proof.

- To prove this we show the number of edges in each connected component of G is bounded by the number of vertices in it.
- Consider a connected component G_c in G.
- Restrict LP(T) and its extreme point solution x to the jobs and machines in G_c only, to obtain $LP_c(T)$ and x_c.
- x_c must be an extreme point solution to LP(T).
- As any extreme point solution has at most $n + m$ nonzero variables the lemma follows. □

Lemma on graph H (1)

Lemma

Graph H has a perfect matching.

Proof

- Each job that is integrally set in x has exactly one edge incident at it in G.
- Remove these jobs, and their incident edges from G to obtain H.
- As the number of edges and vertices removed is equal H is also a pseudo-forest.
- In H each job has a degree of at least 2, so all leave nodes are machines.

Lemma on graph H (2)

Proof (continued).

- Keep matching a leaf with the job it is incident to, and remove both from the graph.
- We are left with even cycles. Match of the alternate edges of each cycle.
- We have now obtained a perfect matching in H. □

Approximation factor

Lemma

The presented algorithm achieves approximation factor 2 for scheduling on unrelated parallel machines.

Proof.

- $T^* \leq OPT$, since LP(OPT) has a feasible solution.
- The extreme point solution x to LP(T^*) has a fractional makespan of $\leq T^*$.
- The restriction of x to integrally set jobs has an integral makespan of $\leq T^*$.
- Each edge (i, j) of H satisfies $p_{ij} \leq T^*$.
- The perfect matching found in H schedules at most one extra job per machine.
- So, the total makespan is $\leq 2T^* \leq 2 \cdot OPT$. □

Summary

- We have seen a factor $\frac{3}{4}$ approximation algorithm for MAX-SAT.
- Derandomization of randomized algorithms was used to obtain that algorithm.
- We have seen a factor 2 approximation algorithm for schedulling on unrelated parallel machines.
- Parametric pruning and the properties of extreme point solutions where used to obtain that algorith.
- Note: For minimum makespan scheduling on identical machines this approximation guarantee was reached using an almost trivial algorithm.