CS 260
Design and Analysis of Algorithms
8. Work with *NP*-Hard Problems

Mikhail Moshkov

Computer, Electrical and Mathematical Sciences & Engineering Division
King Abdullah University of Science and Technology

Work with *NP*-Hard Problems

In this section, we will consider two approaches to the work with *NP*-hard problems. The first approach is to study approximate algorithms for the initial problem.

The second one is a contraction of initial *NP*-hard problem up to another *NP*-hard problem which has good approximate algorithms, or even up to a problem from *P*.

In this section, we will work mainly with optimization problems corresponding to decision problems considered in the previous sections.

A Compendium of NP Optimization Problems

Continuously updated catalog of approximability results for *NP* optimization problems:

http://www.nada.kth.se/∼viggo/wwwcompendium/node276.html

Set Cover Problem

First, we study an approximate algorithm for Set Cover problem which is a greedy algorithm.

Let us consider an arbitrary set cover problem U, F where U is a set containing $N > 0$ elements, and $F = \{S_1, \ldots, S_p\}$ is a family of subsets of the set U such that $U = \bigcup_{i=1}^{p} S_i$.

A subfamily $\{S_{i_1}, \ldots, S_{i_t}\}$ of the family F is called a *cover* if $\bigcup_{j=1}^{t} S_{i_j} = U$. The number t is called the *cardinality* of this cover.

We consider here the problem of searching for a cover with minimum cardinality.

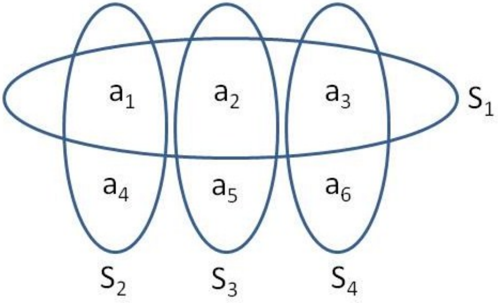
Set Cover Problem

Let us consider the following greedy algorithm: during each step this algorithm chooses a subset $S_i \in F$ with minimal index i which covers maximum number of elements uncovered during previous steps (S_i *covers* all elements from U that belong to S_i).

The algorithm will finish the work when all elements from U are covered.

Set Cover Problem

Example 8.1. Greedy algorithm for Set Cover problem



The constructed cover is $\{S_1, S_2, S_3, S_4\}$.

Set Cover Problem

By $C_{\min}$ we denote the minimum cardinality of a cover.

We denote by C_{greedy} the cardinality of the cover constructed by the greedy algorithm.

Theorem 8.2. $C_{\text{greedy}} \leq C_{\min} \ln N + 1$.

Set Cover Problem

Proof. We denote $m = C_{\min}$. If $m = 1$ then, as it is not difficult to show, $C_{\text{greedy}} = 1$ and the considered inequality holds.

Let $m \geq 2$, and let S_i be a subset with minimum index i for which the cardinality of S_i has maximum value. It is clear that $|S_i| \geq \frac{N}{m}$ (in the opposite case, $C_{\min} > m$ which is impossible).

So after the first step we will have at most $N - \frac{N}{m} = N\left(1 - \frac{1}{m}\right)$ uncovered elements in the set U.

Set Cover Problem

After the first step we will have the following set cover problem: the set $U \setminus S_i$ and the family $\{S_1 \setminus S_i, \ldots, S_p \setminus S_i\}$. For this problem, the minimum cardinality of a cover is at most m.

So after the second step when we choose a set $S_j \in F$ for which the cardinality of $S_j \setminus S_i$ is maximum, the number of uncovered elements in the set U will be at most $N\left(1 - \frac{1}{m}\right)^2$, etc.

After t steps the number of uncovered elements will be at most $N\left(1 - \frac{1}{m}\right)^t$.

Set Cover Problem

Let the greedy algorithm during the process of cover construction make q steps (and constructs a cover of cardinality q). Then after the step number $q-1$ we have at least one uncovered element in the set U.

Therefore $N\left(1-\frac{1}{m}\right)^{q-1} \geq 1$ and $N \geq \left(1+\frac{1}{m-1}\right)^{q-1}$. If we take the natural logarithm of both sides of this inequality, we obtain $\ln N \geq (q-1)\ln\left(1+\frac{1}{m-1}\right)$.

It is known that for any natural r the inequality $\ln\left(1+\frac{1}{r}\right) > \frac{1}{r+1}$ holds. Since $m \geq 2$ we have $\ln N > (q-1)\frac{1}{m}$ and $q < m\ln N + 1$. Taking into account that $m = C_{\mathrm{min}}$ and $q = C_{\mathrm{greedy}}$ we obtain $C_{\mathrm{greedy}} \leq C_{\mathrm{min}}\ln N + 1$. $\square$

Set Cover Problem

The first bound was obtained by Nigmatullin in 1969:

$$C_{\text{greedy}} \leq C_{\min}(1 + \ln N - \ln C_{\min}).$$

Almost exact bounds depending on N only were obtained by Slavik in 1996. If $N \geq 2$ then

$$C_{\text{greedy}} < C_{\min}(\ln N - \ln \ln N + 0.78).$$

For any natural $N \geq 2$ there exists a set cover problem U, F such that $|U| = N$ and

$$C_{\text{greedy}} > C_{\min}(\ln N - \ln \ln N - 0.31).$$

Set Cover Problem

The *approximation ratio* of the greedy algorithm,

$$\max_{U,F:|U|=N} \frac{C_{\text{greedy}}}{C_{\min}},$$

is almost $\ln N - \ln \ln N$.

The following two results show that, under some natural assumptions about NP, the greedy algorithm is very close to the best polynomial approximate algorithms for Set Cover problem.

Set Cover Problem

Feige proved in 1996 that if $NP \not\subseteq DTIME\left(n^{O(\log\log n)}\right)$ then for any ε, $0 < \varepsilon < 1$, there is no polynomial algorithm that for a given set cover problem U, F constructs a cover for U, F which cardinality is at most

$$(1-\varepsilon)C_{\min} \ln |U|.$$

$DTIME\left(n^{O(\log\log n)}\right)$ is the class of languages for each of which there exists an algorithm for language recognition such that the time complexity of the algorithm is bounded from above by $n^{O(\log\log n)}$, where n is the length of input word.

Set Cover Problem

Ras and Safra proved in 1997 that if $P \neq NP$ then there exists $\gamma > 0$ such that there is no polynomial algorithm that for a given set cover problem U, F constructs a cover for U, F which cardinality is at most

$$\gamma C_{\min} \ln |U|.$$

Vertex Cover Problem

Vertex Cover problem is a special case of Set Cover problem. So we can use greedy algorithm to solve Vertex Cover problem.

It is possible to show that for any natural n there exists a graph G_n with at most

$$n\left(1+\frac{1}{2}+\ldots+\frac{1}{n-1}\right) \leq n(\ln n+1)$$

nodes for which $C_{\text{greedy}} > C_{\text{min}}(\ln n - \ln 2 - 1)$ where C_{greedy} is the cardinality of the vertex cover constructed by the greedy algorithm and C_{min} is the minimal cardinality of vertex cover.

Vertex Cover Problem

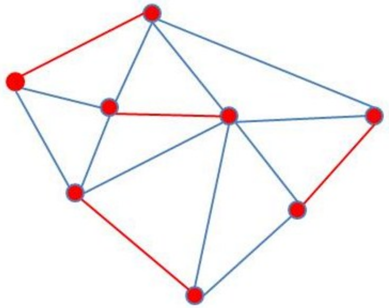
However, there exists more accurate approximate algorithm for Vertex Cover. We will say that a node *covers* edges for which this node is one of the ends.

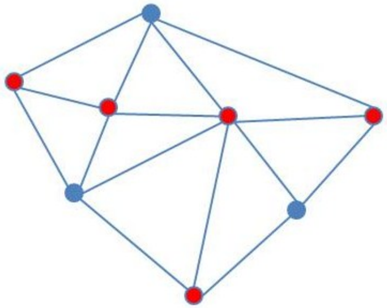
During each step we choose an uncovered edge and add the ends of this edge to the constructed vertex cover. It is clear that at least one end of the chosen edge must be in any vertex cover. So the cardinality of constructed vertex cover is at most 2 times to the minimum cardinality of vertex cover.

Thus, there exists a polynomial approximate algorithm for Vertex Cover problem for which the approximation ratio is at most 2.

Vertex Cover Problem

Example 8.3. Approximate algorithm for Vertex Cover





Vertex Cover Problem

Håstad in 1997 proved that if $P \neq NP$ then for any ε, $0 < \varepsilon < 1$, there is no polynomial approximate algorithm for Vertex Cover problem which approximation ratio is at most $\frac{7}{6} - \varepsilon$.

Maximum Independent Set

In 1999 Håstad proved that if $NP \neq ZPP$ then, for any ε, $0 < \varepsilon < 1$, there is no polynomial approximate algorithm for Maximum Independent Set with approximation ratio $n^{1-\varepsilon}$ where n is the number of nodes.

Here ZPP (Zero-error Probabilistic Polynomial time) is the complexity class of problems for which a probabilistic Turing machine exists which always returns the correct answer, and the running time is polynomial on average for any input.

Note that for Maximum Independent Set the approximation ratio for some algorithm is the maximum value of the cardinality of maximum independent set divided by the cardinality of independent set constructed by the algorithm. We consider maximum among all graphs with n nodes.

Maximum Independent Set

Let us study now the set of trees instead of the set of all undirected graphs. In such a case, there exists a polynomial algorithm for Maximum Independent Set problem. This problem can be solved for trees in linear time using dynamic programming.

Maximum Independent Set

Let $G = (V, E)$ be a tree. We choose a node r of G as the root. Now, each node u of G defines a subtree of G with the root u.

For each u, we consider the subproblem of finding the largest independent set in the subtree with the root u. We denote by $I(u)$ the number of nodes in the largest independent set in this subtree.

Maximum Independent Set

Let S be the largest independent set in the considered subtree. There are two possibilities: $u \in S$ and $u \notin S$.

In the first case, children of u can not be included into S. In the second case we can consider all other nodes of the subtree.

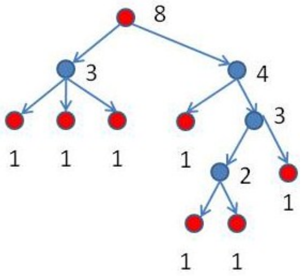
One can show that

$$I(u) = \max \left\{ 1 + \sum_{w \in G(u)} I(w), \sum_{w \in C(u)} I(w) \right\}$$

where $G(u)$ is the set of grandchildren of u and $C(u)$ is the set of children of u.

Maximum Independent Set

Example 8.4. Maximum independent set in tree



Maximum Independent Set

The number of subproblems is exactly the number of nodes. So during the run of the algorithm it will be necessary to make at most $|V|$ comparisons of numbers.

It is not difficult to show that the number of additions is at most $|V| + 2|E|$. So the whole number of operations of addition and comparison of numbers is $O(|V| + |E|)$.

2-SAT

We show that 2-SAT problem belongs to P.

Let $F = F(x_1, \ldots, x_n)$ be a conjunctive normal form (CNF) with variables $x_1, \ldots, x_n$ in which each clause contains exactly two literals that are different and correspond to different variables.

We should recognize if F is *satisfiable*, i.e., if there exist values $\delta_1, \ldots, \delta_n \in \{0, 1\}$ of the variables $x_1, \ldots, x_n$ such that $F(\delta_1, \ldots, \delta_n) = 1$.

2-SAT

We correspond to F a directed graph $G(F)$ with nodes $x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n$: for each clause $a \vee b$ from F, we draw two edges: $\bar{a} \to b$ and $\bar{b} \to a$.

Theorem 8.5. (Aspvall, Plass, and Tarjan, 1979) F *is satisfiable if and only if there is no a strongly connected component in* $G(F)$ *which contains both a variable and its negation.*

The construction of the graph $G(F)$, the construction of all strongly connected components in $G(F)$, and the check of the condition of Theorem 8.5 require a time which is linear depending on the number of clauses in F.

2-SAT

Lemma 8.6. *If F is satisfiable then there is no a strongly connected component in $G(F)$ which contains both a variable and its negation.*

Proof. Let F be satisfiable and $\delta_1, \ldots, \delta_n$ be values of variables $x_1, \ldots, x_n$, respectively, for which $F(\delta_1, \ldots, \delta_n) = 1$.

Let $a \to b$ be an edge in $G(F)$. Then the clause $\bar{a} \vee b$ belongs to F. Since $F(\delta_1, \ldots, \delta_n) = 1$, we have $\bar{a} \vee b = 1$. Therefore if $a = 1$ then $b = 1$.

2-SAT

Let us assume that there exists a strongly connected component in $G(F)$ which contains both a variable x_i and its negation $\bar{x}_i$.

Let $\delta_i = 1$. Since there is a directed path from x_i to $\bar{x}_i$ and $x_i = 1$, we have $\bar{x}_i = 1$ which is impossible.

Let $\delta_i = 0$. Since there is a directed path from $\bar{x}_i$ to x_i and $\bar{x}_i = 1$, we have $x_i = 1$ which is impossible.

We obtain a contradiction. Therefore there is no a strongly connected component in $G(F)$ which contains both a variable and its negation. $\square$

2-SAT

Lemma 8.7. *If there is no a strongly connected component in* $G(F)$ *which contains both a variable and its negation then* F *is satisfiable.*

Proof. Let there be no a strongly connected component in $G(F)$ that contains both a variable and its negation.

We now describe an algorithm which constructs values $\delta_1, \ldots, \delta_n$ of variables $x_1, \ldots, x_n$ such that $F(\delta_1, \ldots, \delta_n) = 1$.

2-SAT

We construct the graph $G(F)$, find all strongly connected components in $G(F)$, construct corresponding meta-graph $G^*(F)$ which nodes are strongly connected components of $G(F)$ (this graph is a directed acyclic graph), and find a topological ordering of $G^*(F)$.

As a result, all strongly connected components of $G(F)$ will be ordered: $V_1, V_2 \ldots$.

Let $i \in \{1, \ldots, n\}$, $x_i \in V_l$ and $\bar{x}_i \in V_k$. Then $\delta_i = 0$ if $l < k$ and $\delta_i = 1$ if $l > k$.

2-SAT

Let $x_1 = \delta_1, \dots, x_n = \delta_n$, and $a \to b$ be an edge in $G(F)$. Let us show that if $a = 1$ then $b = 1$.

Let us assume the contrary: $a = 1$ and $b = 0$. Since the edge $a \to b$ belongs to $G(F)$, the clause $\bar{a} \vee b$ is in F. Therefore the edge $\bar{b} \to \bar{a}$ belongs to $G(F)$.

Let $a \in V_l$, $\bar{a} \in V_k$, $b \in V_p$, and $\bar{b} \in V_q$. Since $a = 1$, we have $l > k$. Since $b = 0$, we have $q > p$. Since $a \to b$ and $\bar{b} \to \bar{a}$ belong to $G(F)$, we have $p \geq l$ and $k \geq q$.

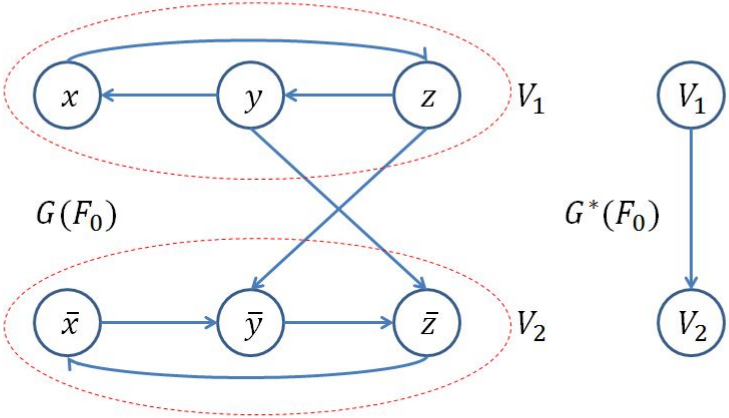
From the inequalities $l > k$, $k \geq q$, and $q > p$ it follows that $l > p$. But it contradicts the inequality $p \geq l$. Therefore, for each edge $a \to b$ in $G(F)$, if $a = 1$ then $b = 1$.

2-SAT

Let us show that $F(\delta_1, \ldots, \delta_n) = 1$. Let us assume the contrary: for a clause $a \vee b$ from F, $a = 0$ and $b = 0$. It is clear that the edge $\bar{a} \to b$ belongs to $G(F)$, $\bar{a} = 1$ and $b = 0$, but this is impossible. Therefore $F(\delta_1, \ldots, \delta_n) = 1$. $\square$

2-SAT

Example 8.8. Let $F_0(x, y, z) = (x \vee \bar{y}) \wedge (y \vee \bar{z}) \wedge (\bar{x} \vee z) \wedge (\bar{y} \vee \bar{z})$. For each variable, the variable and its negation belong to different strongly connected components of the graph $G(F_0)$. The tuple of variable values $(\delta_1, \delta_2, \delta_3)$ described in the proof of Lemma 8.7 is equal to $(0, 0, 0)$. We have $F_0(0, 0, 0) = 1$.



x
y
z
V_1
$G(F_0)$
$\bar{x}$
$\bar{y}$
$\bar{z}$
V_2
$G^*(F_0)$
V_1
V_2