Introduction to NP-Completeness	Lecture 2
Lecturer: S. S. Ravi	Date: Oct. 29, 2020

1 Introduction

In the previous lecture, we discussed the procedure for proving **NP**-completeness results. In this lecture, we will present additional examples of **NP**-completeness results. We will also briefly discuss two methods to cope with **NP**-completeness in practice.

2 Additional Proofs of NP-Completeness

2.1 NP-Completeness of Minimum Set Cover

A formal definition of the **Minimum Set Cover** (MSC) problem is as follows.

Instance: A universal set $U = \{u_1, u_2, \ldots, u_n\}$, a collection $S = \{S_1, S_2, \ldots, S_m\}$, where each S_j is a subset of U $(1 \leq j \leq m)$ and an integer $r \leq m$.

Question: Is there is a subcollection S' of S such that $|S'| \leq r$ and the union of the sets in S' is equal to U?

An example to illustrate the problem is given below.

Example of MSC: Let $U = \{u_1, u_2, u_3, u_4, u_5\}$ and let $S = \{S_1, S_2, S_3, S_4\}$, where $S_1 = \{u_1, u_3\}$, $S_2 = \{u_2, u_4\}$, $S_3 = \{u_3, u_5\}$, and $S_4 = \{u_2, u_5\}$. Further, let $r = 3$. This is a "YES" instance of MSC since we can choose the subcollection $\{S_1, S_2, S_3\}$ as a solution. On the other hand, if $r = 2$, you can verify that we have a "NO" instance of MSC.

Applications of MSC: The Minimum Set Cover (MSC) problem has applications in several areas, including software testing and computational epidemiology. We briefly mention an application in software testing. For an application in epidemiology, see [4].

Suppose we have some software whose source code consists of N lines, numbered 1 through N. For a test input t_i, one can identify the set S_i of lines of the source code reached by this test input. Suppose we have a test set $T = \{t_1, t_2, \ldots, t_m\}$ consisting of m tests such that if we run all the tests in T, at least one test will reach each line of source code. If T is large, some of the tests may be redundant; that is, there may be smaller test set T' such that it is enough to run the tests in T' to make sure that each source line is reached. This is exactly the MSC problem where the universal set $U = \{1, 2, \ldots, N\}$, the set $S_i \subseteq U$ corresponds to test t_i $(1 \leq i \leq n)$ and we are required to choose a subcollection of $\{S_1, S_2, \ldots, S_m\}$ of minimum cardinality that covers U.

We now show that MSC is **NP**-complete.

Theorem 2.1 *The MSC problem is **NP**-complete.*

Proof: We first show that MSC is in **NP**. To do this, notice that a proposed solution is a subcollection S' of S. Given S', one can efficiently verify that $|S'| \leq k$ and that the union of all the sets in $S' = U$.

To prove **NP**-hardness, we use a reduction form the Minimum Vertex Cover (MVC) problem. Recall that each instance I of MVC consists of a graph $G(V, E)$ and an integer k. From this instance I we need to construct an instance I' of MSC.

Intuitive idea: MVC covers edges using vertices while MSC covers elements using sets. So, it will be useful to make edges and vertices in the MVC instance to correspond to elements and sets in the MSC instance.

Given an instance I of MVC, we construct an instance I' of MSC as follows.

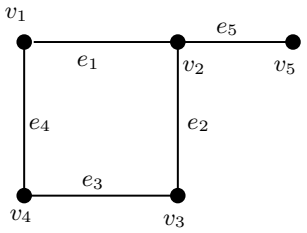
(a) Let $E = \{e_1, e_2, \ldots, e_m\}$ denote the edge set, where $m = |E|$. The universal set $U = \{u_1, u_2, \ldots, u_m\}$ in the MSC instance I' is in one-to-one correspondence with E; that is, element u_i corresponds to edge e_i, $1 \leq i \leq m$.

(b) Let $V = \{v_1, v_2, \ldots, v_n\}$. The set collection $S = \{S_1, S_2, \ldots, S_n\}$ is in one-to-one correspondence with V; that is, set S_j corresponds to vertex v_j, $1 \leq j \leq n$.

(c) For $1 \leq j \leq n$, the elements of set S_j are chosen as follows. Consider each edge e_i that is incident on vertex v_j and add the corresponding element to S_j. (Thus, the elements in S_j are exactly the elements of U corresponding to the edges incident on vertex v_j.)

(d) The bound on the number of sets r is set to the bound k on the vertex cover size.

An example of this construction is shown below.



An instance of MVC with $k = 2$

$U = \{u_1, u_2, u_3, u_4, u_5\}$
$S = \{S_1, S_2, S_3, S_4, S_5\}$, where
$S_1 = \{u_1, u_4\}$ $\quad S_2 = \{u_1, u_2, u_5\}$
$S_3 = \{u_2, u_3\}$ $\quad S_4 = \{u_3, u_4\}$
$S_5 = \{u_5\}$
Bound r on the number of sets $= 2$

We now show the above construction can be carried out in polynomial time. Step (a) and Step (b) can be done in time $O(|E|)$ and $O(|V|)$ respectively. In Step (c), set S_j can be constructed in $O(\text{degree}(v_j))$ time; thus, the time used in constructing all the sets is $O(\sum_{j=1}^{n} \text{degree}(v_j)) = O(|E|)$ (since the sum of the degrees of all the vertices in a graph is equal to twice the number of

edges). Step (d) uses $O(1)$ time. So, the overall time for the construction is $O(|V| + |E|)$, which is linear in the size of the MVC instance.

We now show that there is a solution to the MVC instance I iff there is solution to the MSC instance I'.

Part 1: Suppose there is a solution S' to the MSC instance I'. We need to show that there is a solution to the MVC instance I.

Without loss of generality, let $S' = \{S_1, S_2, \ldots, S_\ell\}$ for some $\ell \leq r$. We claim that the set $V' = \{v_1, v_2, \ldots, v_\ell\}$ is a solution to the MVC instance I. To see this, note that $|V'| = \ell \leq r = k$. Thus, V' satisfies the size constraint. To prove that V' is a vertex cover, consider any edge $e_x = \{v_i, v_j\} \in E$. By our construction, the element u_x corresponding to e_x appears only in sets S_i and S_j. Since S' is a solution to MSC, S' includes at least one of S_i and S_j. Thus, by our choice of V', at least one of v_i and v_j appears in V'. In other words, V' is a solution to the MVC instance.

Part 2: Suppose there is a solution V' to the MVC instance I. We need to show that there is a solution to the MSC instance I'.

Without loss of generality, let $V' = \{v_1, v_2, \ldots, v_\ell\}$ for some $\ell \leq k$. We claim that the set $S' = \{S_1, S_2, \ldots, S_\ell\}$ is a solution to the MSC instance I'. To see this, note that $|S'| = \ell \leq k = r$. Thus, S' satisfies the size constraint. To prove that S' is a set cover, consider any element u_x. This element corresponds to edge $e_x = \{v_i, v_j\} \in E$. Since V' is a vertex cover, at least one of v_i and v_j appears in V'. By our construction, element u_x appears in both the sets S_i and S_j. Thus, by our choice of S', at least one of the sets S_i and S_j is included in S' to cover the element u_x. In other words, S' is a solution to the MSC instance, and this completes our proof. ■

2.2 A Reduction from SAT: NP-Completeness of Maximum Independent Set

The 3SAT problem is the version of SAT in which each clause contains exactly three literals. This version is known to be **NP**-complete [2] and is very useful in doing reductions to show **NP**-completeness. We will illustrate one such reduction which shows that the Maximum Independent Set (MIS) problem is **NP**-complete. This proof is from [3].

Theorem 2.2 *The MIS problem is **NP**-complete.*

Proof: Showing that MIS is in **NP** is straightforward and is left as an exercise to the student.

To prove **NP**-hardness, we use a reduction 3SAT. Any instance I of 3SAT consists of a set $X = \{x_1, x_2, \ldots, x_n\}$ of variables and a set $C = \{C_1, C_2, \ldots, C_m\}$ of clauses, where each clause has exactly 3 literals. The resulting MIS instance should consist of a graph $G(V, E)$ and an integer $\ell \leq |V|$. The intuition behind the reduction is as follows.

- From each clause we construct subgraph of G.

- 3SAT requires at least one literal have the value 1 in each clause. We want this to corresponds to choosing one vertex from each subgraph.
- In choosing satisfying assignments, we need to make sure that a variable x and its complement $\overline{x}$ are not both set to 1. (That would be a **conflict**.) We need to enforce this by adding edges in G between suitable pairs of nodes so that both nodes of such a pair are not chosen in the independent set.

We now present the construction.

1. For each clause C_j, $1 \leq j \leq m$, we construct the following subgraph G_j of G: G_j has three nodes, denoted by $v_{j,1}$, $v_{j,2}$ and $v_{j,3}$ and the three edges that connect these nodes into a triangle. Each node of G_j corresponds to a literal of C_j. (Note that from each subgraph G_j only one node can be chosen in any independent set for G.)
2. Consider any pair of subgraphs G_p and G_q: if there are two nodes $v_{p,a}$ and $v_{q,b}$ such that these nodes correspond to a variable and its complement, we add the edge $\{v_{p,a}, v_{q,b}\}$ to G. (We refer to these as **conflict** edges.)
3. We set the required size ℓ of the independent set to m (the number of clauses).

An example to illustrate the above construction is given below.

Instance of 3SAT:

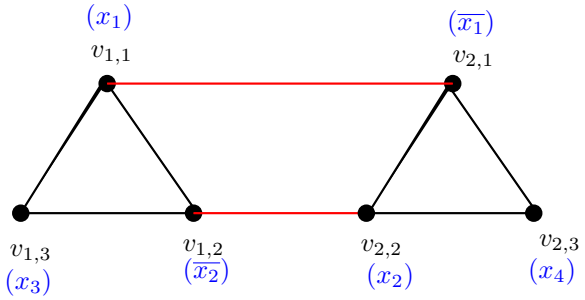
$X = \{x_1, x_2, x_3, x_4\}$

$C = \{C_1, C_2\}$, where

$C_1 = (x_1 \vee \overline{x_2} \vee x_3)$

$C_2 = \{\overline{x_1} \vee x_2 \vee x_4)$

Resulting MIS Instance:



Size of independent set $\ell = 2$

Note: For each node, the corresponding literal is shown in **blue**. The edges representing conflict are shown in **red**.

Time used by the reduction:

- Since each clause in the 3SAT instance has exactly 3 literals and there are m clauses, the size of the 3SAT instance is $O(m)$.

- Step 1 of the reduction produces a subgraph with 3 nodes and 3 edges for each clause C_j. Since there are m clauses, this step produces $3m$ nodes and $3m$ edges. So, the time used by this step is $O(m)$.
- Since each subgraph has 3 nodes, there can be at most 9 conflict edges between any pair of subgraphs. Since the number of pairs of subgraphs is $m(m-1)/2 = O(m^2)$, the total number of conflict edges added in Step 2 is $O(m^2)$. So, Step 2 uses $O(m^2)$ time.
- Step 3 uses $O(1)$ time. So, the overall time for the reduction is $O(m^2)$, which is a polynomial function of the size of the 3SAT instance.

We now prove the correctness of the reduction.

Part 1: Suppose the 3SAT instance I has a solution. We must show that there is an independent set of size m for the MIS instance I'.

Consider any satisfying assignment to the 3SAT instance. For each clause C_j, the assignment sets at least one of the literals in each clause to 1. Choose one such literal arbitrarily. Note that the chosen literal in C_j has a corresponding node in the subgraph G_j. Construct the set V' by choosing one such node from each subgraph G_j, $1 \leq j \leq m$. Thus, $|V'| = m$. We can prove by contradiction that V' is an independent set. Suppose there are two nodes v and w in V' such that $\{v, w\}$ is an edge. Since V' contains exactly one node from each subgraph, v and w are from different subgraphs. In other words, $\{v, w\}$ is a conflict edge. This implies that the literals, say a and b corresponding to v and w are complements of each other. Since both v and w correspond to literals which are set to 1, the given solution to 3SAT is invalid since it sets two complementary literals a and b to 1. This is a contradiction. Hence V' is a solution to the MIS instance I'.

Part 2: Suppose the MIS instance I' has a solution V'. We must show that the 3SAT instance I has a solution. We construct a solution to 3SAT as follows.

1. For each node $v \in V'$, let a_v denote the literal corresponding to v. Set a_v to 1 (thus also setting $\overline{a_v}$ to 0).
2. After Step 1, if any variable has not been assigned a value, set the value of the variable to 0.

Example: In the graph produced from the 3SAT instance, suppose $V' = \{v_{1,2}, v_{2,3}\}$. The literal corresponding to $v_{1,2}$ is $\overline{x_2}$ and that corresponding to $v_{2,3}$ is x_4. So, the above procedure sets $\overline{x_2}$ to 1 (i.e., it sets variable x_2 to 0) and x_4 to 1. The other two variables (i.e., x_1 and x_3) are set to 0.

To prove that this is a satisfying assignment, first notice that every variable has been assigned a value. Since V' is an independent set, the chosen assignment does not have any conflicts (i.e., a variable and its complement are not assigned the same value). Since $|V'| = m$, for each subgraph G_j, V' has node, say v_j from G_j. Now, consider any clause C_j and let v_j be the node in V' from subgraph G_j. Since the literal a corresponding to v_j occurs in clause C_j and a has been set to 1, C_j is satisfied. Thus, there is a solution to the 3SAT instance I. This completes the proof. ■

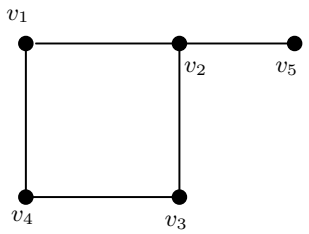
2.3 NP-Completeness of Minimum Dominating Set

A formal definition of the **Minimum Dominating Set** (MDS) problem is as follows.

Instance: An undirected graph $G(V, E)$ and an integer $r \leq |V|$.

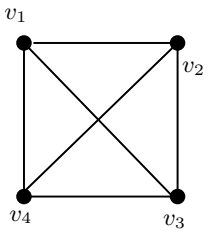
Question: Does G have a **dominating set** of size at most r, that is, is there a subset $V' \subseteq V$ such that $|V'| \leq r$ and for each vertex $v_i \in V - V'$, there is some node $v_j \in V'$ such that $\{v_i, v_j\} \in E$?

When V' is a dominating set, we say that each vertex in $V - V'$ is **dominated** by a vertex in V'. An example to illustrate the notion of dominating sets is given below.



- For the graph G (on the left), $V_1 = \{v_4, v_5\}$ is a dominating set. (Vertex v_4 dominates v_1 and v_3 while v_5 dominates v_2.) It is also a minimum dominating set.
- $V_2 = \{v_1, v_3\}$ is not a dominating set since v_5 is not dominated by any vertex in V_2.

The following example shows the difference between vertex cover and dominating set.



- For the graph G (on the left), any vertex cover must have at least 3 vertices. (Otherwise, there will be an edge that is not covered.)
- However, a dominating set needs only one vertex. (That vertex dominates the other three.)

Applications of MDS: The Minimum Dominating Set (MDS) problem has applications in several areas including fault detection in industrial systems and wireless networks. We will briefly discuss an application in detecting faults in industrial systems. Applications of MDS in wireless networks are discussed in [1].

Large manufacturing plants consist of many subsystems (units) that need to be monitored for faults (e.g., high temperature, high pressure or high humidity in a subsystem). Each monitoring device can handle multiple units. However, each device can monitor only a specified subset of units. Imagine an undirected graph where there two sets of nodes. Each node in one set (V_1) represents a location for placing a monitoring device and each node in the other set V_2 represents the location of a unit to be monitored. There is an edge between a node x in V_1 and node y in V_2 if the device represented by x can monitor the unit represented by y. In this setting, choosing a minimum cardinality subset of nodes from V_1 that dominate all the nodes in V_2 is precisely the MDS problem.

We now show that MDS is **NP**-complete.

Theorem 2.3 *The MDS problem is **NP**-complete.*

Proof: To see that MDS is in **NP**, note that a proposed solution is a subset V' of V. Given V', one can efficiently verify that $|V'| \leq r$ and that each vertex $v_i \in V - V'$ has an edge to some node $v_j \in V'$.

To prove NP-hardness, we use a reduction from the MSC problem.

Intuitive idea: We want to create a graph (for the MDS problem) where there are two sets of vertices, one corresponding to the elements of MSC problem (say, element nodes) and the other corresponding to the sets (say, set nodes). We will make sure that there is a dominating set consisting entirely of set vertices. (So, "a set covers an element" corresponds to a "set node dominates an element node".)

Consider an instance I of MSC given by the universal set $U = \{u_1, u_2, \ldots, u_n\}$, the set collection $S = \{S_1, S_2, \ldots, S_m\}$ and the integer k. We may assume that each element $u_i \in U$ appears in at least one set in S. (Otherwise, there is no way to cover u_i using the sets in S.) An instance I' of the MDS problem consisting of graph $G(V, E)$ and the integer r is constructed as follows.

(a) Let $V = V_1 \cup V_2$ where $V_1 = \{v_1, v_2, \ldots, v_n\}$ (i.e., there is one vertex in V_1 corresponding to each element in U) and $V_2 = \{w_1, w_2, \ldots, w_m\}$ (i.e., there is one vertex in V_2 corresponding to each set in S). We will refer each vertex in V_1 as an **element vertex** and each vertex in V_2 as a **set vertex**.

(b) Let $E = E_1 \cup E_2$ where

$$E_1 \;=\; \{\{v_i, w_j\} \mid u_i \in S_j\} \quad \text{and} \quad E_2 \;=\; \{\{w_i, w_j\} \mid i \neq j\}.$$

Thus, the edges in E_1 represent the membership of elements in subsets and the edges in E_2 connect all the nodes in V_2 into a complete graph (i.e., one in which there is an edge between every pair of vertices in V_2).

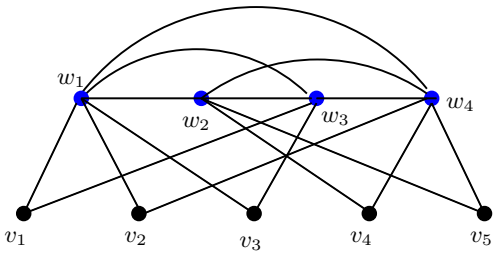
(c) Set r (the dominating set size) equal to k (set cover size).

An example to illustrate the construction is given below.

MSC Instance:

- $U = \{u_1, u_2, u_3, u_4, u_5\}$
- $S = \{S_1, S_2, S_3, S_4\}$ where
  $S_1 = \{u_1, u_2, u_3\}$
  $S_2 = \{u_4, u_5\}$
  $S_3 = \{u_1, u_3\}$
  $S_4 = \{u_2, u_4, u_5\}$
- $k = 2$

MDS Instance:



$r = 2$

Note: Set nodes are in **blue**.

The construction can be carried out in polynomial time since V has $m + n$ nodes and E has $O(m(m + n))$ edges (because $|E_1| \leq mn$ and $|E_2| = m(m - 1)/2$).

We now show that the resulting MDS instance I' has a solution if and only if the MSC instance I has a solution.

Part 1: Suppose the MSC instance I has a set cover of size at most k. We will show that G has a dominating set D of size at most $r = k$.

Without loss of generality, let $S' = \{S_1, S_2, \ldots, S_\ell\}$, where $\ell \leq r$, denote the given set cover. Consider the set $D = \{w_1, w_2, \ldots, w_\ell\}$ of set nodes from G. We claim that D is a dominating set for G. To see this, first note that any node in D (i.e., a set node) dominates all the other set nodes (since the set nodes form a clique). Further, since S' is a set cover, every node in V_1 (i.e., an element node) is adjacent to at least one node in D (i.e., a set node). Thus, D is indeed a dominating set. Further, $|D| = \ell \leq r$ and so D is a solution to the MDS instance I'.

Part 2: Suppose the MDS instance I' has a dominating set of size at most $r = k$. We will show that there is a set cover S' of size at most k for the MSC instance I.

Let D' be a solution to the MDS instance I'. Thus, D' is a dominating set for G and $|D'| = \ell \leq k$. Partition D' into D_1 and D_2 so that $D_1 \subseteq V_1$ and $D_2 \subseteq V_2$. If D_1 is nonempty, we modify D' repeatedly as follows until D_1 becomes empty:

1. Let v be a node (element node) in D_1.
2. Find a node w (a set node) in V_2 such that $\{v, w\} \in E$. Such a node must exist since each element in the MSC instance occurs in at least one set.
3. Delete v from D_1 and add w to D_2 if w is not already in D_2.

After each step of this modification, $D_1 \cup D_2$ continues to be a dominating set since all the (set) nodes dominated by the element node v are also dominated by the set node w. Obviously, this

modification does not increase the size of D'. At the end of this modification, we have $D_2 \subseteq V_2$ is a dominating set of size at most k for G.

Without loss of generality, let $D_2 = \{w_1, w_2, \ldots, w_\ell\}$. Recall that $\ell \leq k$. Consider the subcollection $S' = \{S_,S_, \ldots, S_\ell\}$. We will prove that S' is a set cover. To see this, consider any element $u_x \in U$. We will show that some set in S' contains u_x. Recall that D' is the original dominating set for G. We have two cases to consider.

Case 1: Set D' contains v_x, the node corresponding to u_x.

In this case, the modified dominating set D was obtained by replacing v_x by a node $w_y \in V_2$ such that $\{w_y, v_x\}$ is an edge in G. By our construction and the choice of S', the set S_y corresponding to w_y is in S' and S_y contains u_x.

Case 2: Set D' does not contain v_x, the node corresponding to u_x.

In this case, since D' is a dominating set, there must be a node $w_y \in D_1$ such that $\{w_y, v_x\}$ is an edge in G. By our construction and the choice of S', the set S_y corresponding to w_y is in S' and S_y contains u_x.

Thus, S' is a valid set cover. Further, $|S'| = \ell \leq k$ and so S' is a solution to the MSC instance I. This completes the proof of Part 2 as well as that of the theorem. ■

3 Coping with NP-completeness

3.1 Introduction

We saw that **NP**-completeness implies computational intractability. In other words, there are problem instances which are likely to take a long time to solve. However, such problems arise in practice frequently. We will briefly discuss two methods used by researchers to cope with **NP**-complete problems.

3.2 Using Mathematical Programming Software

Researchers in Mathematics and Operations Research have developed a number of software tools for solving optimization problems expressed using mathematical programming formulations. In such formulations, there is an objective function that uses variables and appropriate constraints are specified on the variables. Examples of some well known tools for finding solutions to such formulations include Gurobi and CPLEX. These software tools use a number of clever heuristics to speed up the search for solutions. In practice, these tools are able to solve fairly large problems.

We will illustrate how a mathematical programming formulation can be developed for the optimization version of the Minimum Vertex Cover (MVC) problem. In other words, our goal is to find a minimum cardinality vertex cover for a given graph $G(V, E)$. In particular, we will develop a formulation for MVC where variables are constrained to take on values from $\{0, 1\}$, the

objective function is linear and the other constraints are also linear. Such a formulation is called a **{0,1}-Integer Linear Program** (or {0,1}-ILP). As you will notice, the transformation from MVC to {0,1}-ILP is actually a reduction. (However, the problems considered here are not decision problems.)

Developing an ILP for MVC: Suppose $G(V, E)$ is the given graph for the MVC problem. We want to develop an ILP whose solution will give us a vertex cover of minimum size.

Let $V = \{v_1, v_2, \ldots, v_n\}$. Note that each vertex cover is a subset of V. For each node v_i, we create a $\{0, 1\}$-valued variable x_i, $1 \leq i \leq n$. The significance of x_i is as follows: $x_i = 1$ if v_i is chosen in the optimal solution and 0 otherwise. Thus, $\sum_{i=1}^{n} x_i$ gives us the number of nodes chosen in the optimal solution. Therefore, the objective of the ILP is:

$$\text{Minimize} \quad \sum_{i=1}^{n} x_i$$

What are the constraints? For each edge $e = \{v_i, v_j\}$, at least one of v_i and v_j must be in the solution; that is, at least one of x_i and x_j must be set to 1. This gives the following constraint:

$$x_i + x_j \geq 1 \quad \text{for each edge } \{v_i, v_j\}$$

So, the complete ILP for the MVC problem is:

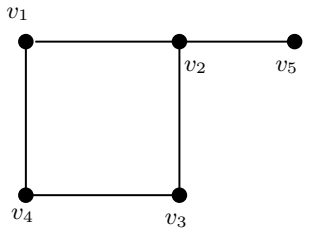
$$\text{Minimize} \quad \sum_{i=1}^{n} x_i \quad \text{subject to the following constraints:}$$

$$x_i + x_j \geq 1, \quad \text{for each edge } \{v_i, v_j\}$$

$$x_i \in \{0, 1\}, \quad 1 \leq i \leq n.$$

Exercise: Show that every solution to the above ILP is a minimum vertex cover for G and vice versa.

An example of a graph and the corresponding ILP formulation is given below.



$$\text{Minimize} \quad x_1 + x_2 + x_3 + x_4 + x_5$$

subject to the following constraints:

$$x_1 + x_2 \geq 1$$
$$x_1 + x_4 \geq 1$$
$$x_2 + x_3 \geq 1$$
$$x_2 + x_5 \geq 1$$
$$x_3 + x_4 \geq 1$$
$$x_i \in \{0, 1\}, \quad 1 \leq i \leq 5$$

Note: A solution to the above ILP is to set $x_2 = x_4 = 1$ and the other variables to 0. The corresponding minimum vertex cover is $\{v_2, v_4\}$.

The ILP formulation provides other advantages as well. Two such advantages are as follows.

1. Suppose there is a known cost c_i associated with choosing node v_i in the vertex cover, $1 \leq i \leq n$, and the goal is to choose a vertex cover of minimum cost. This can be done by simply changing the minimization objective to $\sum_{i=1}^{n} c_i\, x_i$. Since each c_i is a constant, the formulation is still linear.

2. Suppose we are required to find a solution that includes a node x_i but excludes but node x_j. These conditions can be handled by adding the constraints $x_i = 1$ and $x_j = 0$.

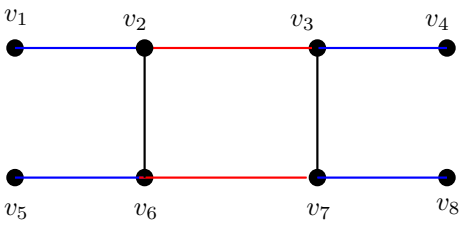
3.3 Approximation Algorithms

Often, one does not need exact solutions to some optimization problems that are **NP**-complete. For example, instead of getting a vertex cover of minimum size, it may be sufficient in practice to get a vertex cover that is near-optimal. When the goal is to obtain a near-optimal solution for a problem, it may be possible to devise a polynomial time algorithm for the problem. Such an algorithm is called an **approximation algorithm** or a **heuristic**. For some approximation algorithms, one can rigorously establish a **performance guarantee** of the following form: "for every instance, the solution produced by the approximation algorithm is within a factor of 2 of the optimal solution". In other cases, researchers study the performance of a heuristic experimentally and observe that the heuristic performs well in practice.

An Approximation Algorithm for MVC: Here, we consider the MVC optimization problem and present an approximation algorithm that provides a performance guarantee. To do this, we need a few graph theoretic concepts.

Definition: Given an undirected graph $G(V, E)$, a **matching** M in G is a subset of edges such that no two edges in M share an end point. A matching M is **maximal** if no edge can be added to it without violating the matching property. A matching M^* is a **maximum matching** if it has the largest number of edges among all the matchings of G.

The following figure illustrates the above definition.



- For the graph G (on the left), the set M consisting of the edges $\{v_1, v_2\}$ and $\{v_2, v_6\}$ is *not* a matching since the two edges share the end point v_2.
- The set M_1 consisting of the **blue** edges is a **maximum** matching with 4 edges.
- The set M_2 consisting of the two **red** edges is a **maximal** matching.

For any graph $G(V, E)$, a *maximal* matching can be found efficiently through the following simple algorithm.

1. Let $M = \emptyset$. (M will contain a maximal matching at the end of the algorithm.)
2. **while** $E \neq \emptyset$ **do**

   (i) Choose any edge $\{x, y\}$ from E and add it to M.

   (ii) Delete all the edges in E that have either x or y as an end point.
3. Output M.

Exercise: Explain how the above algorithm can be implemented to run in $O(|V| + |E|)$ time.

There is a simple approximation algorithm for the MVC problem using the above algorithm for finding a maximal matching. The algorithm is as follows.

1. Given $G(V, E)$, find a maximal matching M of G.
2. Let V' consist of both the end points of each edge in M. Output V' as the vertex cover.

We have the following theorem.

Theorem 3.1 *For any undirected graph $G(V, E)$, the set V' produced by the above algorithm is a vertex cover. Further, if V^* is any minimum vertex cover for G, then $|V'| \leq 2|V^*|$.*

Proof: We first prove that V' is a vertex cover. Let M be the maximal matching from which V' was constructed. Consider the node set $V - V'$. For any two nodes x and y in $V - V'$, if the edge $\{x, y\}$ is in G, then M' cannot be a maximal matching. Therefore, the nodes in $V - V'$ form an independent set in G. Consequently, V' is a vertex cover for G.

For the second part, let V^* be a minimum vertex cover for G. Let q be the number of edges in the maximal matching M. Note that V^* must contain at least one node from each edge in M. (If $\{x, y\} \in M$ and V^* does not contain either x or y, the edge is not covered by V^*.) In other words, $|V^*| \geq q$. On the other hand, $|V'| = 2q$ since V' uses both the end points of the edges in M. Therefore, $|V'| \leq 2|V^*|$. ■

We say that the above algorithm provides a performance guarantee of 2 for the MVC (optimization) problem. Currently, no algorithm with a better performance guarantee is known for general graphs.

Approximation algorithms for **NP**-hard problems is an active area of research. A nice chapter on approximation algorithms appears in [3]. Several text books that discuss approximation algorithms for many optimization problems are also available (e.g., [6, 7]).

A Joke on Reductions

To our knowledge, this joke initially appeared in a text book by Eugene Lawler [5].

A theoretical computer scientist, who is very fond of using reductions, is being interviewed by a reporter. The interviewer's questions and the scientist's answers are given below.

Question 1: You are given a jug with cold water, a cup, a tea bag and a microwave. How do you make tea?

Answer: I will pour water into the tea cup, put the tea bag into the cup and get the cup heated using the microwave.

Question 2: Good answer! Suppose I change the situation mentioned in Question 1 so that the tea cup already contains cold water. How will you make tea now?

Answer: I will pour the water from the cup into the jug so that the problem reduces to the one considered in Question 1!

Suggested Exercises

1. Show that the following version of 2SAT called **Special 2SAT** (S2SAT) is **NP**-complete.

   Instance: A set $X = \{x_1, x_2, \ldots, x_n\}$ of Boolean variables, a set $C = \{C_1, C_2, \ldots, C_m\}$ of clauses, where each clause has at most two literals and an integer $r \leq n$.

   Question: Is there an assignment to the variables in X such that all clauses are satisfied and at most r variables in X are set to 1?

   Hint: Use a reduction from MVC.

2. Suppose a system has a set $R = \{r_1, r_2, \ldots, r_k\}$ of k resources and there is a collection $C = \{R_1, R_2, \ldots, R_p\}$ of resource requests, where each request R_i is a subset of R, $1 \leq i \leq p$. We say that two requests R_i and R_j $(i \neq j)$ **conflict** if $R_i \cap R_j \neq \emptyset$; that is, R_i and R_j have at least one resource in common. A subcollection C' of C is **feasible** if there is no conflict between any pair of requests in C'.

   Example: Suppose $R = \{r_1, r_2, r_3, r_4, r_5\}$ and $C = \{R_1, R_2, R_3\}$, where $R_1 = \{r_1, r_2\}$, $R_2 = \{r_2, r_4\}$ and $R_3 = \{r_1, r_3, r_5\}$. Note that R_1 conflicts with both R_2 and R_3. However, R_2 and R_3 don't conflict. Thus, the subcollection $C' = \{R_2, R_3\}$ is feasible.

   Show that the following problem, called the **Maximum Feasible Request Set** (MFRS) problem, is **NP**-complete.

   Instance: A set $R = \{r_1, r_2, \ldots, r_k\}$ of k resources, a collection $C = \{R_1, R_2, \ldots, R_p\}$ of p resource requests, where $R_i \subseteq R$, $1 \leq i \leq p$, and an integer $q \leq p$.

   Question: Is there a subcollection C' of C such that C' is feasible and the number of requests in C' is at least q?

   Hint: Use a reduction from MIS.

3. Show that the following problem, called the **Close Neighbor** (CN) problem, is **NP**-complete.

   Instance: A complete graph $G(V, E)$ with a positive integer weight $\omega(u, v)$ for each edge $\{u, v\} \in E$, a positive integer $k \leq |V|$ and a positive integer q.

   Question: Is there a subset $V' \subseteq V$ such that $|V'| \leq k$ and for each node $x \in V - V'$, there is a node $y \in V'$ with $\omega(x, y) \leq q$?

   Hint: Use a reduction from MDS.

4. Show that the following problem, which we call **Directed Path Coverage** (DPC), is NP-complete.

   Instance: A directed graph $G(V, A)$, a collection $P = \{p_1, p_2, \ldots, p_r\}$ of r directed simple paths in G and an integer $k \leq |V|$.

   Question: Is there a subset V' of nodes such that $|V'| \leq k$ and every path in P contains at least one node from V'?

5. Develop a {0,1}-ILP for the MDS problem.

References

[1] H. Du, L. Ding, W. Wu, D. Kim, P. M. Pardalos, and J. Willson. Connected dominating set in wireless networks. In P. M. Pardalos, D. Z. Du, and R. L. Graham, editors, *Handbook of Combinatorial Optimization*. Springer, Berlin, Germany, 2013.

[2] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman & Co., San Francisco, 1979.

[3] J. Kleinberg and E. Tardos. *Algorithm Design*. Addison Wesley, Reading, MA, 2006.

[4] Chris J Kuhlman, VS Anil Kumar, Madhav V Marathe, SS Ravi, and Daniel J Rosenkrantz. Inhibiting diffusion of complex contagions in social networks: Theoretical and experimental results. *Data Mining and Knowledge Discovery*, 29(2):423–465, 2015.

[5] E. L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, New York, NY, 1976.

[6] V. V. Vazirani. *Approximation Algorithms*. Springer, New York, NY, 2001.

[7] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, New York, NY, 2010.