CS 5114
Solutions to Homework Assignment 7

Meghendra Singh

April 7, 2017

[30] 1. Exercise 34.5-5. The set-partition problem takes as input a set S of numbers. The question is whether the numbers can be partitioned into two sets A and $\overline{A} = S - A$ such that $\sum_{x \in A} x = \sum_{x \in \overline{A}} x$. Show that the set-partition problem is NP-complete.

A. State the SET-PARTITION **problem as a formal decision problem, as we do in class.**

B. Follow the NP-completeness paradigm to show that SET-PARTITION **is NP-complete.**

A. We can define the set-partition problem as follows:

> SET PARTITION
> INSTANCE: A set $S = \{s_1, s_2, ..., s_n\}$ of n integers.
> QUESTION: Is there a subset $A \subset S$ such that the sum of all integers in A is equal to the sum of all integers in $S - A$. $\sum_{x \in A} x = \sum_{x \in S-A} x$.

A set of integers is partitionable if there exists a subset of the set such that, the sum of integers in the subset is equal to the sum of integers, in the original set but not in the subset. We can define the language corresponding to SET-PARTITION as follows:

SET-PARTITION $= \{e(S) | S$ is partitionable$\}$

B. To show that SET-PARTITION is NP-complete, we need to prove:

1. SET-PARTITION $\in NP$.

2. SET-PARTITION is NP-Hard.

1. SET-PARTITION $\in NP$: A certificate for SET-PARTITION is any subset of S. We can compute the sum of elements in S and the sum of elements in the certificate (i.e. $A \subset S$) in linear time. We can easily check if the sum of elements in A is equal to half of the sum of elements in S, (i.e. $\sum_{x \in A} x = \frac{\sum_{x \in S} x}{2}$) in constant time. Hence, we can verify SET-PARTITION in linear time, and therefore SET-PARTITION $\in NP$.

2. SET-PARTITION is NP-Hard: We can show that SUBSET-SUM $\leq_{\mathrm{p}}$ SET-PARTITION by constructing a reduction function f, which reduces an instance of SUBSET-SUM to an instance of SET-PARTITION in polynomial time. SUBSET-SUM $= \{e(S, t) | S$ is a set of positive integers that has a subset $S' \subseteq S$, such that $\sum_{s \in S'} s = t\}$. f

takes $e(S, t)$ (an instance of SUBSET-SUM) and reduces it to $e(S')$ (an instance of SET-PARTITION). We can construct f as follows:

Let $\langle S, t\rangle$ be an instance of SUBSET-SUM. Let $X = \sum_{s\in S} s$ and $S' = S \cup \{X + t, 2X - t\}$. The sum of all elements in S' can be written as $\sum_{s\in S'} s = X + (X + t) + (2X - t) = 4X$. We claim that $e(S')$ is an instance of SET-PARTITION. We prove this claim below.

The sum of all elements in S' is $4X$, SET-PARTITION would partition S' into two sets A and $\overline{A} = S - A$, such that the sum of elements in these two sets would be equal to $2X$ each. Also, the two elements $(X + t)$ and $(2X - t)$ which were added to S for getting S' will never be on the same side of the partition, i.e. if $(X + t) \in A$ then $(2X - t) \notin A$, and if $(2X - t) \in A$ then $(X + t) \notin A$. This is true because adding both $(X + t)$ and $(2X - t)$ to A would make the sum of elements in $A \geq 3X$ (similarly adding both $(X + t)$ and $(2X - t)$ to $\overline{A}$ would make the sum of elements in $\overline{A} \geq 3X$), this will be an incorrect solution, as the sum of elements in both A and $\overline{A}$ should be $2X$. Assume that $(2X - t) \in A$, therefore $(X + t) \in \overline{A}$ and we can write the sum of elements in both A and $\overline{A}$ as follows:

$$\sum_{s\in A} s = (2X - t) + a = 2X \tag{1}$$

$$\sum_{s\in \overline{A}} s = (X + t) + b = 2X \tag{2}$$

Here, a and b are the sums of all the elements except for $(2X - t)$ and $(X + t)$ in A and $\overline{A}$ respectively. Also, $a + b$ is the sum of all elements present in the set S of the original SUBSET-SUM instance $\langle S, t\rangle$, therefore $a + b = X$. From equation (1) above we can deduce that $a = t$, therefore $b + t = X$. So we can conclude that the set S of the original SUBSET-SUM instance $\langle S, t\rangle$ has a subset $S'' \subseteq S$ such that $\sum_{x\in S''} x = t$. Therefore given a certificate and instance $\langle S, t\rangle$ for SUBSET-SUM a verifier would return 1, also given a certificate and instance $\langle S'\rangle$ a SET-PARTITION verifier would also return 1. Conversely, assume that for the set S in the original SUBSET-SUM instance $\langle S, t\rangle$ there is no subset S'' for which $\sum_{x\in S''} x = t$. This would imply that $b + t \neq X$ and $a \neq t$ therefore, equations (1) and (2) above would not hold (i.e. now $(2X - t) + a \neq 2X$ and $(X + t) + b \neq 2X$). So given a certificate and instance $\langle S'\rangle$ a SET-PARTITION verifier would return 0 in this case, also given a certificate and instance $\langle S, t\rangle$ a SUBSET-SUM verifier would return 0. Hence, we have reduced an instance of SUBSET-SUM to SET-PARTITION, and since the reduction only involved appending two extra elements to set S in the SUBSET-SUM instance $\langle S, t\rangle$, we can be sure that this reduction is linear time. Since, SUBSET-SUM $\in$ NP-Complete and SUBSET-SUM $\leq_{\mathrm{p}}$ SET-PARTITION, we have SET-PARTITION is NP-complete.

---

[30] 2. Consider the following decision problem:

PATH SELECTION

INSTANCE: A directed graph $G = (V, E)$; a set $\Pi = \{P_1, P_2, \ldots, P_c\}$ of c directed paths in G; a bound $K \geq 0$.

QUESTION: Is there a subset $\Pi' \subseteq \Pi$ of cardinality $\Pi' \geq K$ such that every pair of distinct paths from Π' is node disjoint?

A. Define the corresponding language, call it PS**.**

B. Show that PS $\in \mathcal{NP}$**.**

C. Give a polynomial-time reduction of 3-CNF-SAT **to** PS**.**

D. Conclude that PS **is** $\mathcal{NP}$**-complete.**

A. We can define the language corresponding to PATH SELECTION as follows:

PS $= \{e(G, \Pi, K) | G = (V, E)$ is a directed graph and Π is a subset of all directed paths in G, there exists a set $\Pi' \subseteq \Pi$ such that Π' contains at-least K node disjoint paths.$\}$

B. To show that PS $\in \mathcal{NP}$, we can consider a set of paths $\Pi' = \{P_1, P_2, ..., P_l\}$ as the certificate. Given an instance $\langle G, \Pi, K \rangle$ and a certificate Π', a verifier can check whether:

(i) $\underline{\Pi' \subseteq \Pi}$: This can be done in polynomial time, by comparing each directed path in Π' with each path in Π.

(ii) $\underline{\forall P'_i, P'_j \in \Pi' \text{ such that } i \neq j,\ P'_i \text{ and } P'_j \text{ are node disjoint}}$: This can also be done in polynomial time by comparing each path in Π' with all other paths in Π'.

(iii) $\underline{|\Pi'| \geq K}$: This can be done in linear time by counting the number of paths in Π'.

If all of the three checks above succeed the verifier would return a 1, and if any of these fail the verifier would return a 0. Hence, we can easily verify PS in polynomial time and therefore, PS $\in \mathcal{NP}$.

C. In order to reduce 3-CNF-SAT to PS, lets consider a boolean formula ϕ in 3-CNF having n variables and k clauses:

$$\phi(x_1, x_2, ..., x_n) = C_1 \wedge C_2 \wedge ... \wedge C_k$$

Each clause C_i in ϕ contains 3 literals, each of which can either be a variable x_j or its negation $\neg x_j$. Now, ϕ is an instance of 3-CNF-SAT, and we need to reduce it to an instance of PS (i.e. $\langle G, \Pi, K \rangle$) in polynomial time such that when ϕ is satisfiable the corresponding instance of PS would contain a set $\Pi' \subseteq \Pi$ such that Π' contains at-least K node disjoint paths. Conversely, when ϕ is not satisfiable corresponding instance of PS would not contain a set $\Pi' \subseteq \Pi$ such that Π' contains at-least K node disjoint paths. To reduce ϕ to $\langle G, \Pi, K \rangle$, we start by constructing the directed graph $G = (V, E)$ as follows:

(i) **Construct variable gadgets:** For each variable $x_i \in \phi$, add two vertices x_i and $\overline{x_i}$ connected by two paths, each going from x_i to $\overline{x_i}$. We will call these paths **true path** and **false path**, so as to represent the boolean values of TRUE and FALSE which can be taken up by the variable, given a truth assignment for ϕ. Figure 1 shows an example of a variable gadget.

(ii) **Construct clause gadgets:** For each clause $C_i \in \phi$ add two vertices C_i and $\overline{C_i}$. In step **(iii)** we will be connecting these two vertices by three paths, each of which would represent the three literals that need to be present in a 3-CNF clause. Figure 2 shows an example of a clause gadget with the three paths.

(iii) **Construct intersection gadgets:** For each literal l in each clause $C_i \in \phi$, if $l = x_j$ insert a vertex n_{ij1} on the **true path** connecting x_j to $\overline{x_j}$. Also, connect a directed edge from C_i to n_{ij1} and another directed edge from n_{ij1} to $\overline{C_i}$. Similarly, if $l = \neg x_j$ insert a vertex n_{ij0} on the **false path** connecting x_j to $\overline{x_j}$. Also, connect a directed edge from C_i to n_{ij0} and another directed edge from n_{ij0} to $\overline{C_i}$. We call these n_{ij0} and n_{ij1} vertices, intersection nodes as these are the nodes where the variable and clause gadgets would connect with each other.

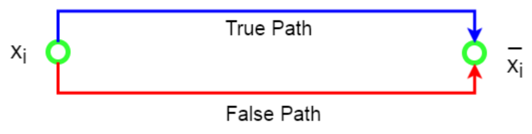

Figure 1: A variable gadget for boolean variable x_i.

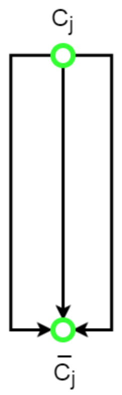

Figure 2: A clause gadget for clause C_j.

The above steps would result in a directed graph G which we will use in the instance of PS reduced from ϕ. Also, in the above process, step **(i)** will take linear time with respect to the number of variables in ϕ, step **(ii)** will also take linear time with respect to the number of clauses in ϕ and step **(iii)** will also take linear time with respect to the number of clauses in ϕ, since here we are creating 3, n_{ij1} vertices for each clause. Hence, we can be sure that the above algorithm creates G using ϕ in linear time. The next step is to construct a set of paths Π, this is done as follows:

(i) Let $\Pi = \{\}$ and $N = \{\}$.

(ii) For each variable $x_i \in \phi$, if the boolean value for x_i in the truth assignment is TRUE, we add the **false path** connecting x_i to $\overline{x_i}$ into Π and add all the intersection nodes n_{ji1} in this **false path** into N. Similarly, if the boolean value for x_i in the truth assignment is FALSE, we add the **true path** connecting x_i to $\overline{x_i}$ into Π and add all the intersection nodes n_{ji0} in this **true path** into N.

(iii) For each clause $C_j \in \phi$, add all the paths connecting C_j to $\overline{C_j}$ into Π, such that the intersection nodes contained in the paths are $\notin N$.

These steps would result in the set of paths Π which we will include in the instance of PS. Once again, in the above process, step **(i)** will take constant time and step **(ii)** will execute for every variable and in the worst case a true or false path would contain k intersection nodes, so this will take polynomial time. Step **(iii)** would execute for every clause, and in the worst case this will also take polynomial time. Hence, we can be sure that the above algorithm creates Π in polynomial time. Next, we set K equal to the sum of the number of clauses and variables in ϕ (i.e. $K = k + n$). All of the above steps (i.e. constructing G, Π and assigning a value to K) would take polynomial time.

For a satisfying truth assignment of ϕ the set of paths Π will always have a subset $\Pi' \subseteq \Pi$, such that number of node disjoint paths in $\Pi' = K$. This is true because a satisfying assignment would lead to at-least one literal being TRUE in each clause of ϕ. In our reduction this corresponds to at-least one intersection node not being included for each clause in the true/false paths for the variables. Since, we only select those paths of clause gadgets into Π, which do not contain an intersection node that is already a part of one of the true/false paths selected in Π, we will select at-least one path for each clause gadget. As all clause gadgets are node disjoint with each other, there will always be k node disjoint paths in Π for a satisfying assignment. Furthermore, each of the n true/false paths selected into Π for each variable in ϕ, will be node disjoint with each other and at-least one of the clause paths for each of the k clause gadgets. Hence, there will at-least be a total of $k + n$ node disjoint paths in Π. So, for a satisfying assignment there will always be a subset $\Pi' \subseteq \Pi$ such that all paths in Π' are node disjoint and $|\Pi'| = K$.

Conversely, if the truth assignment of ϕ is not satisfying, there will be at-least one clause in ϕ which will evaluate to FALSE. In our reduction this corresponds to all the intersection nodes for that clause being already present in the true/false paths for the variables. Since, we only select those paths of clause gadgets into Π, which do not contain an intersection node that is already a part of one of the true/false paths selected in Π, we will not select any path for that particular clause gadget. As a result the number of node disjoint paths in Π will be less than the $k + n$ (or K). Therefore, there won't be a subset $\Pi' \subseteq \Pi$ such that all paths in Π' are node disjoint and $|\Pi'| = K$. As an example consider the following boolean formula in 3-CNF:

$$\phi(x_1, x_2, x_3) = (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$$

A satisfying assignment for ϕ is $x_1 = TRUE, x_2 = FALSE, x_3 = TRUE$. Figure 3 shows the directed graph $G = (V, E)$ constructed by the reduction function for ϕ given the truth assignment. Vertices are shown as green circles, blue edges represent **true paths**, red edges represent **false paths** and black edges represent the clause gadget paths. Solid lines show the paths that have been selected into the set Π and dashed lines are the paths not selected into Π.

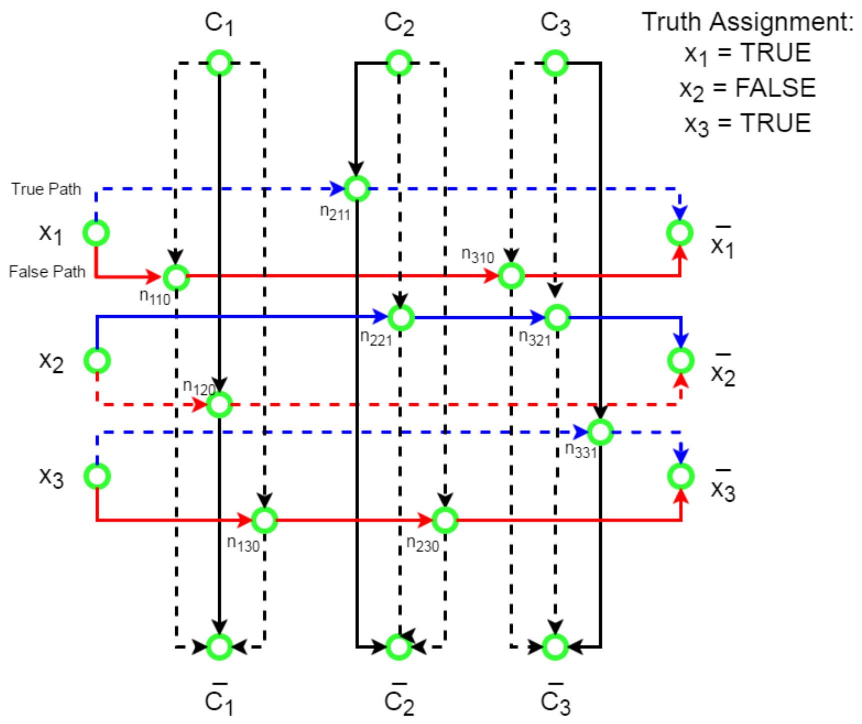

Figure 3: Directed graph G generated by reduction function for ϕ given a satisfying assignment

Here, we can clearly observe that:

$$\Pi = \{< x_1, n_{110}, n_{310}, \overline{x_1} >, < x_2, n_{221}, n_{321}, \overline{x_2} >, < x_3, n_{130}, n_{230}, \overline{x_3} >,$$
$$< C_1, n_{120}, \overline{C_1} >, < C_2, n_{211}, \overline{C_2} >, < C_3, n_{331}, \overline{C_3} >\}$$

All of the six paths in Π are node disjoint and since the number of clauses in $\phi = 3$ and the number of variables is $n = 3$ we have $K = 6$. Therefore, $\Pi' \subseteq \Pi$ exists in this case and a verifier for PS would return 1 for this instance. Similarly, a non-satisfying truth assignment for ϕ is $x_1 = TRUE, x_2 = FALSE, x_3 = FALSE$. Figure 4 shows the directed graph $G = (V, E)$ constructed by the reduction function for ϕ given the truth assignment. In this case we observe that:

$$\Pi = \{< x_1, n_{110}, n_{310}, \overline{x_1} >, < x_2, n_{221}, n_{321}, \overline{x_2} >, < x_3, n_{331}, \overline{x_3} >,$$
$$< C_1, n_{120}, \overline{C_1} >, < C_1, n_{130}, \overline{C_1} >, < C_2, n_{211}, \overline{C_2} >, < C_2, n_{230}, \overline{C_2} >\}$$

Here, even though the cardinality of Π is 7, the number of node disjoint paths in Π is only 5 (vertices $C_1, \overline{C_1}$ and $C_2, \overline{C_2}$ repeat between the last four paths). Since the number of clauses and variables in $\phi = 3$ and 3 respectively, we have $K = 6$. Therefore, $\Pi' \subseteq \Pi$ does not exist in this case and a verifier for PS would return 0 for this instance. This is what we expected as the truth assignment for ϕ in this case was not satisfying. Hence, we can reduce 3-CNF-SAT to PS in polynomial time, i.e. 3-CNF-SAT $\leq_{\text{p}}$ PS.

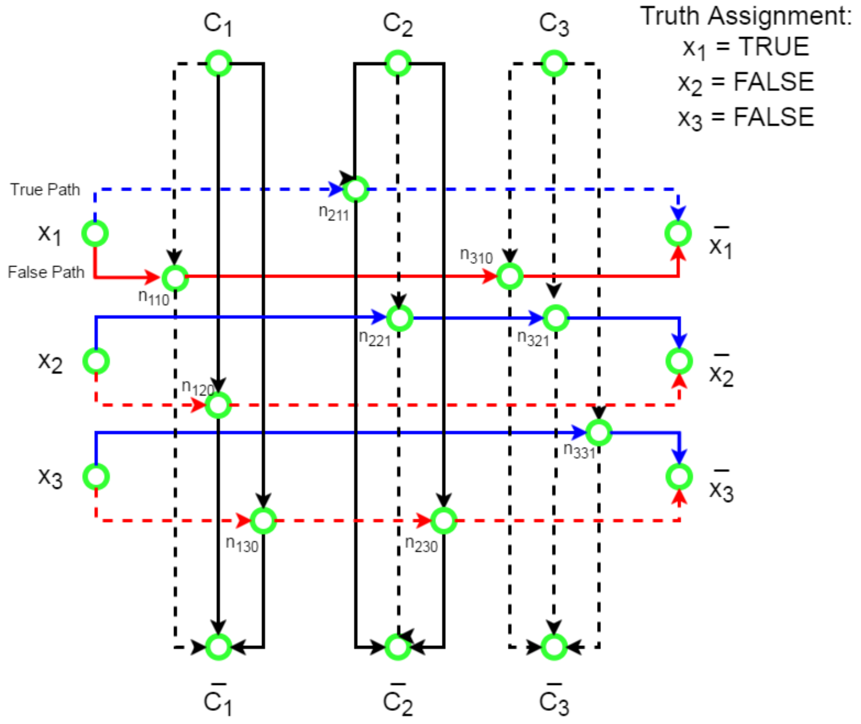

Figure 4: Directed graph G generated by reduction function for ϕ given a non-satisfying truth assignment

D. From the conclusions of sections **B.** and **C.** above we know that PS $\in \mathcal{NP}$ and 3-CNF-SAT $\leq_{\mathtt{p}}$ PS. Since, 3-CNF-SAT is also NP-Complete and 3-CNF-SAT $\leq_{\mathtt{p}}$ PS, we have PS is NP-complete.