CSE 6140 / CX 4140

Computational Science & Engineering Algorithms

Homework 4

Please type all answers.

1. (15 points)

A store trying to analyze the behavior of its customers will often maintain a two-dimensional array A, where the rows correspond to its customers and the columns correspond to the products it sells. The entry $A[i, j]$ specifies the quantity of product j that has been purchased by customer i.

Here's a tiny example of such an array A.

	liquid detergent	beer	diapers	cat litter
Raj	0	6	0	3
Alanis	2	3	0	0
Chelsea	0	0	0	7

One thing that a store might want to do with this data is the following. Let us say that a subset S of the customers is *diverse* if no two of the of the customers in S have ever bought the same product (i.e., for each product, at most one of the customers in S has ever bought it). A diverse set of customers can be useful, for example, as a target pool for market research.

We can now define the Diverse Subset Problem as follows: Given an $m \times n$ array A as defined above, and a number $k \leq m$, is there a subset of at least k of customers that is *diverse*?

Show that Diverse Subset is NP-complete.

Hint: Reduce from Independent Set.

Solution: Let's denote the Diverse subset problem by DS. We prove this in two steps as follows:

- **Step 1:** $DS \in NP$**:** Given the matrix A and a subset S (candidate solution), we can verify it efficiently ($O(kn)$ where $|S| = k$) by considering the corresponding k rows of matrix A and counting the number of non-zero entries in each column. If any column has more than one non-zero entry then the subset S is *not diverse* since two or more people have purchased that product. If all columns have at most one non-zero entry then the subset S *is diverse*. Since we are able to verify a candidate solution efficiently, the decision problem $DS \in NP$.

- **Step 2:** `Independent_Set` $\leq_p$ `Diverse_Subset`**:** First, given an instance of Independent Set Problem (IS), we construct an instance of DS:

  Given the graph G and integer k, we construct a $|V| \times |E|$ matrix A where corresponding to each vertex v_i there is a customer c_i and corresponding to each edge e_j there is a product p_j. The entry a_{ij} is 1 if vertex v_i is an endpoint of the edge e_j and otherwise, the entry is zero. So, this matrix A and integer

k will be the corresponding instance of DS. So in the language of DS, each node v corresponds to a customer and each edge $e := (u, v)$ corresponds to a product purchased by both u and v. Also, the same integer k is copied to the constructed instance of DS. Note that each step of this construction can be done in polynomial time.

Suppose IS has a solution S, then G has an independent set of size at least k. So, between any pair of nodes u, v in S there exists no edge e. So, considering the customers S' corresponding to vertices in S, by the construction, we see that for any pair of customers c_i, c_j there exists no product purchased by both the customers. In terms of the matrix A, we pick the rows corresponding to the vertices in S, and we note that every column has at most one non-zero entry. Thus every product is purchased by at most one customer in the subset S', which means that S' is a diverse subset. Hence, IS **is solvable** $\Rightarrow$ DS **is solvable**.

Now, if the constructed instance of DS has a solution for integer k, say S', then we have subset of at least k customers of which no two customers purchased the same product. So, in the constructed matrix A, the rows corresponding to S' have at most one non-zero entry in each column. So, every edge of G has contributed at most one endpoint to S'. This means that no two nodes in S' are connected by an edge, which implies that the subset S' corresponds to an independent set of G of size at least k. So, DS **is solvable** $\Rightarrow$ IS **is solvable**.

So, we have proved that DS is solvable if and only if IS is solvable. If B is a black box that efficiently solves DS, we can take the given IS instance, construct the equivalent DS instance (as done above), and ask if the matrix A has a diverse subset of size at least k. If B answers yes, we can infer from the above equivalence that the IS instance has an independent set of size at least k. Else, no. Thus, we have seen that $IS \leq_p DS$ that is:

$$\texttt{Independent\_Set} \leq_p \texttt{Diverse\_Subset}$$

IS is a known $NP - complete$ problem. Since, $IS \leq_p DS$, we can infer that $DS \in NP - complete$.

---

2. (15 points)

Suppose you're helping to organize a summer sports camp, and the following problem comes up. The camp is supposed to have at least one counselor who's skilled at each of the n sports covered by the camp (baseball, volleyball, and so on). They have received job applications from m potential counselors. For each of the n sports, there is some subset of the m applicants qualified in that sport. The question is: For a given number $k < m$, is it possible to hire at most k of the counselors and have at least one counselor qualified in each of the n sports? We'll call this the *Efficient Recruiting Problem.*

Show that Efficient Recruiting is NP-complete.

Hint: Reduce from Vertex Cover.

Solution: Let us denote the Efficient Recruiting problem as ER. We prove the required result in the following two steps:

- **Step 1:** $ER \in NP$: Given a subset of at most k counselors, we can efficiently check if there is at least one counselor skilled at each of the n sports. For this, we can consider each counselor one at a time and strike off the sports they specialize in. If all sports are struck off, then the subset of at most k counselors is a certificate for ER. So, $ER \in NP$.

- **Step 2:** `Vertex_Cover` $\leq_p$ `Efficient_Recruiting`**:** First, given an instance of Vertex Cover (VC), we construct an equivalent instance of ER as follows:

  Given the graph G and integer $k < |V(G)|$, corresponding to each vertex v_i, assign a counselor c_i, and corresponding to each edge e_j, assign a sport s_j. Also, the counselor c_i specializes in sport s_j if and only if v_i is an endpoint of e_j. Now, the sets $C = \{c_i\}_{i=1}^{|V(G)|}$ and $S = \{s_j\}_{j=1}^{|E(G)|}$ along with the integer k forms the instance of ER. Note that each step of this construction can be done in polynomial time.

  If the VC instance is solvable with a subset V' of at most k nodes, then we have at most k nodes covering all the $|E(G)|$ edges. So, as per construction the corresponding counselors $C' = \{c_i \in C : v_i \in V'\}$ cover all the sports in S. That is, for every sport there is at least one counselor c_i skilled in that sport. So, C' corresponds to a subset of at most k counselors such that for every sport there is at least one counselor skilled in it. So, VC **is solvable** $\Rightarrow$ ER **is solvable**.

  Now, if C^* is a solution to the constructed ER instance, then we have a subset of at most k counselors for which there is at least one counselor who is skilled at each of the $|E(G)|$ sports. Now, consider the set $V^* = \{v_i \in V(G) : c_i \in C^*\}$ which is the set of nodes of G corresponding to the counselors in C^*. Since C^* covers all the sports, the new set of nodes V^* covers all the edges, otherwise there exists at least one sport in which none of the counselors in C^* are skilled, which contradicts that C^* solves ER. Hence, V^* is a vertex cover of graph G of size at most k. So, ER **is solvable** $\Rightarrow$ VC **is solvable**.

So, we have proved that ER is solvable if and only if VC is solvable. If B is a black box that efficiently solves ER, we can take the given VC instance, construct the equivalent ER instance (as done above), and ask if there is a subset of at most k counselors of which there is at least one who is skilled at each of the sports. If B answers yes, we can infer from the above equivalence that the VC instance has a vertex cover of size at most k. Else, no. Thus, we have seen that $VC \leq_p ER$ that is:

`Vertex_Cover` $\leq_p$ `Efficient_Recruiting`

VC is a known $NP - complete$ problem. Since, $VC \leq_p ER$, we can infer that $ER \in NP - complete$.

3. (20 points)

The mapping of genomes involves a large array of difficult computational problems. At the most basic level, each of an organism's chromosomes can be viewed as an extremely long string (generally containing millions of symbols) over the four-letter alphabet $\{A, C, G, T\}$. One family of approaches to genome mapping is to generate a large number of short, overlapping snippets from a chromosome, and then to infer the full long string representing the chromosome from this set of overlapping substrings.

While we won't go into these string assembly problems in full detail, here's a simplified problem that suggests some of the computational difficulty one encounters in this area. Suppose we have a set $S = \{s_1, s_2, \ldots, s_n\}$ of short DNA strings over a q-letter alphabet; and each string s_i has length 2ℓ, for some number $\ell \geq 1$. We also have a library of additional strings $T = \{t_1, t_2, \ldots, t_m\}$ over the same alphabet; each of these also has length 2ℓ. In trying to assess whether the string s_b might come directly after the string s_a in the chromosome, we will look to see whether the library T contains a string t_k so that the first ℓ symbols in t_k are equal to the last ℓ symbols in s_a, and the last ℓ symbols in t_k are equal to the first ℓ symbols in s_b. If this is possible, we will say that t_k *corroborates* the pair (s_a, s_b). (In other words, t_k could be a snippet of DNA that straddled the region in which s_b directly followed s_a.)

Now we'd like to concatenate all the strings in S in some order, one after the other with no overlaps, so that each consecutive pair is corroborated by some string in the library T. That is, we'd like to order the strings in S as $s_{i_1}, s_{i_2}, \ldots, s_{i_n}$, where $i_1, i_2, \ldots, i_n$ is a permutation of $\{1, 2, \ldots, n\}$, so that for each $j = 1, 2, \ldots, n-1$, there is a string t_k that corroborates the pair $(s_{i_j}, s_{i_{j+1}})$. (The same string t_k can be used for more than one consecutive pair in the concatenation.) If this is possible, we will say that the set S has a *perfect assembly*.

Given sets S and T, the *Perfect Assembly Problem* asks: Does S have a perfect assembly with respect to T? Prove that Perfect Assembly is NP-complete.

Example. Suppose the alphabet is $\{A, C, G, T\}$, the set $S = \{AG, TC, TA\}$, and the set $T = \{AC, CA, GC, GT\}$ (so each string has length $2\ell = 2$). Then the answer to this instance of Perfect Assembly is yes: We can concatenate the three strings in S in the order $TCAGTA$ (so $s_{i_1} = s_2$, $s_{i_2} = s_1$, and $s_{i_3} = s_3$). In this order, the pair (s_{i_1}, s_{i_2}) is corroborated by the string CA in the library T, and the pair (s_{i_2}, s_{i_3}) is corroborated by the string GT in the library T.

Hint: Reduce from Hamiltonian Path.

Solution: Let us denote the *Perfect Assembly problem* as PAP. We wish to prove it is NP-complete using the known NP-complete problem of Hamiltonian Path HP, which decides if a given graph G has a path that visits every vertex exactly once (start and end vertices are not the same). We prove the result in two steps:

- **Step 1:** $PAP \in NP$**:** Given the ordering $s_{i_1}, s_{i_2}, s_{i_3}, \ldots, s_{i_n}$ and the set T, we can efficiently verify if the set S has a perfect assembly with set T. We can do this as follows: We pick a consecutive pair $(s_{i_j}, s_{i_{j+1}})$ from the given ordering, form the string $t_j = (s_{i_j}[-l:] + s_{j_{i+1}}[0:l])$ and check if this belongs to T. If the constructed t_j belongs to the set T for all $j = 1, 2, \ldots, (n-1)$, then S has a perfect assembly with respect to T, else no. So, we can conclude that $PAP \in NP$.

- **Step 2:** `Hamiltonian_Path` $\leq_p$ `Perfect_Assembly_Problem`**:** Given a graph G (an instance of HP) we wish to construct a specific instance of PAP. For every vertex v_i take a string s_i of fixed length $2k$ ($k \in \mathbf{Z}^+$) and for every edge $e := (v_p, v_q)$ form the string $t_{p,q} := (s_p[-k:] + s_q[0:k])$. Now, we have formed the sets $S = \{s_i : i = 1, 2, \ldots, |V(G)|\}$ and $T := \{t_{p,q} : (v_p, v_q) \in E(G)\}$. This

forms our specific instance of PAP. Note that each step of this construction can be done in polynomial time. Next, we prove their equivalence.

If the HP instance is solvable, (i.e.,) G has a Hamiltonian path, let the path be v_{i_1}, v_{i_2}, v_{i_3}, ..., $v_{i_{|V(G)|}}$, where $(i_1, i_2, \ldots, i_{|V(G)|})$ is a permutation of $\{1, 2, \ldots, |V(G)|\}$. We claim that the ordering s_{i_1}, s_{i_2}, s_{i_3}, ..., $s_{i_{|V(G)|}}$ is a valid order for the PAP. Note that since every pair of nodes $(v_{i_j}, v_{i_{j+1}})$ is connected by an edge, by construction, for the corresponding pair of consecutive strings $(s_{i_j}, s_{i_{j+1}})$ there is the string $t_{i_j,i_{j+1}} \in T$ that corroborates the pair. Hence, our claim is true and hence PAP is solvable. So, we have shown HP **is solvable** $\Rightarrow$ PAP **is solvable**.

Now, if the PAP instance is solvable, then we have an ordering of the string s in S as s_{i_1}, s_{i_2}, s_{i_3}, ..., $s_{i_{|V(G)|}}$. Note that, by construction of the strings in T, each pair $(s_{i_j}, s_{i_{j+1}})$ is corroborated by $t_{i_j,i_{j+1}}$. So, for the corresponding order of the vertices, v_{i_1}, v_{i_2}, v_{i_3}, ..., $v_{i_{|V(G)|}}$ we note that there is an edge $e := (v_{i_j}, v_{i_{j+1}})$ connecting every consecutive pair of vertices in the above ordering. So, we can say that the above ordering denotes a valid path in G. Since, all the v_{i_j} are distinct in the path and there are $|V(G)|$ vertices in the graph G, the above path visits every vertex exactly once. This means that the above path is a Hamiltonian path in G. So, this solves the HP instance. So, we have shown PAP **is solvable** $\Rightarrow$ HP **is solvable**.

So, we have proved that PAP is solvable if and only if HP is solvable. If B is a black box that efficiently solves PAP, we can take the given HP instance, construct the equivalent PAP instance (as done above), and ask if the set S has a perfect assembly with respect to the set T. If B answers yes, we can infer from the above equivalence that the HP instance has a Hamiltonian path. Else, no. Thus, we have seen that $HP \leq_p PAP$ that is:

$$\texttt{Hamiltonian\_Path} \leq_p \texttt{Perfect\_Assembly\_Problem}$$

HP is a known $NP - complete$ problem. Since, $HP \leq_p PAP$, we can infer that $PAP \in NP - complete$.

---

4. (20 points) Madison is in preschool and has learned to spell some simple words. She has a colorful set of refrigerator magnets featuring the letters of the alphabet (some number of copies of the letter A, some number of copies of the letter B, and so on), and the last time you saw her the two of you spent a while arranging the magnets to spell out words that she knows. The two of you tried to spell out words so as to use up all the magnets in the full set –that is, picking words that she knows how to spell, so that once they were all spelled out, each magnet was participating in the spelling of exactly one of the words. (Multiple copies of words are okay here; so for example, if the set of refrigerator magnets includes two copies each of C, A, and T, it would be okay to spell out CAT twice.)

This turned out to be pretty difficult, and it was only later that you realized a plausible reason for this. Suppose we consider a general version of the problem of

Using Up All the Refrigerator Magnets, where we replace the English alphabet by an arbitrary collection of symbols, and we model Madison's vocabulary as an arbitrary set of strings over this collection of symbols. The goal is the same as in the previous paragraph.

Prove that the problem of Using Up All the Refrigerator Magnets is NP-complete.

Hint: Reduce from 3D matching.

Solution: Let us denote the `Using_Up_All_Refrigerator_Magnets` problem as RM. We wish to show that $RM \in NP$-complete using the known NP-complete problem of 3D-Matching ($3DM$). We prove this in two steps as follows:

- **Step 1:** $RM \in NP$**:** Given a set of words formed, we can count number of each letter used and verify if it is equal to the available number of magnets of that letter. Since this check can be done in polynomial time, we can conclude that $RM \in NP$.
- **Step 2:** `3D-Matching` $\leq_p$ `Using_Up_All_Refrigerator_Magnets`**:** For this, we consider an arbitrary instance of $3DM$ and form a specific instance of RM. Let $W = (X, Y, Z)$ where X, Y and Z are 3 partite sets of the same size (say n) with a set of tuples $M \subseteq X \times Y \times Z$, be an arbitrary instance of $3DM$. Assign a unique letter to each node in each partite set (x_i if node from X, y_j if node from Y and z_k if node from Z). So, we have $3n$ letters and a set of connections $M \subseteq X \times Y \times Z$. A tuple $(x_i, y_j, z_k) \in M$ denotes a word known to Madison. So, each node in W is a letter magnet and the goal is to use up all magnets (i.e.,) cover each node such that every magnet appears in exactly one tuple. Note that this construction can be done in polynomial time.

  If the $3DM$ instance is solvable, then we have n tuples such that every node appears in exactly one tuple. Now, consider the set of words where each word is formed by the letters corresponding to nodes in each tuple (so 3-letter words). In this way, we would have formed words known by Madison and used up all the letter magnets, thus solving the constructed instance of RM. So, $3DM$ **is solvable** $\Rightarrow$ RM **is solvable**.

  If the RM instance is solvable, then we have a set of 3-letter words that uses up all the magnets. Now, we could consider the tuples corresponding to these words. We would get n tuples since each tuple has 3 letters, and there are $3n$ letters in total and each letter appears in exactly one tuple. So, the set of tuples forms the perfect matching for our $3DM$ instance. So, RM **is solvable** $\Rightarrow$ $3DM$ **is solvable**.

So, we have proved that RM is solvable if and only if $3DM$ is solvable. If B is a black box that efficiently solves RM, we can take the given $3DM$ instance, construct the equivalent RM instance (as done above), and ask if all the $3n$ letter magnets can be used up by forming words known to Madison. If B answers yes, we can infer from the above equivalence that the $3DM$ instance has a perfect matching. Else, no. Thus, we have seen that $3DM \leq_p RM$ that is:

`3-Dimensional_Matching` $\leq_p$ `Using_Up_All_Refrigerator_Magnets`

$3DM$ is a known $NP - complete$ problem. Since, $3DM \leq_p RM$, we can infer that $RM \in NP - complete$.

---

5. (15 points)

A convoy of ships arrives at a port and delivers a total of n containers, each containing a different kind of hazardous material. Waiting near the port is a set of m trucks, each of which can hold up to k containers. Any container can be placed in any truck; however, there are certain pairs of containers that cannot be placed together in the same truck. The chemicals they contain may react explosively if brought into contact.

The *Truck Loading Problem* is as follows. Is there a way to load all n containers into the m trucks so that no truck is overloaded, and no two containers are placed in the same truck when they are not supposed to be?

Show that Truck Loading is NP-complete.

Hint: Reduce from 3-Coloring.

Solution: Let us denote the Truck Loading problem by TLP. We prove that $TLP \in NP$-complete using the 3-coloring problem, which is a known NP-complete problem. We prove the result in two steps as follows:

- **Step 1:** $TLP \in NP$**:** Given an assignment of trucks to each of the n containers, we can efficiently verify if the assignment is valid so that no two containers that shouldn't be placed together are assigned to the same truck. We can do this by maintaining m linked lists where the i^{th} linked list holds the label of the containers assigned to the i^{th} truck. Now, for each list, as we populate, we can compare if the new added label is compatible with other labels in the linked list by using hash maps where key-value pairs will be pairs of labels of containers that can't be placed together. If the newly added label isn't compatible, then the given assignment is not a valid certificate, else it is. In this manner, we can efficiently check the validity of the assignment. Hence, $TLP \in NP$.

- **Step 2:** `3-Coloring` $\leq_p$ `Truck_Loading_Problem`**:** For, this we consider an arbitrary instance of the 3-Coloring problem ($3CP$) and form a specific instance of TLP. So, let graph G be an arbitrary instance of $3CP$. Now, for each vertex v_i in the graph G assign a container c_i and for each edge $e := (v_i, v_j)$, we add the constraint that c_i and c_j cannot be loaded into the same truck. Also, assigning a color p to a vertex v_i is equivalent to loading container c_i onto truck p. Since, only 3 colors are allowed for G, for our specific instance we are required to load the $|V(G)|$ containers in 3 trucks. Also, since there is no restriction on the maximum number of vertices that can be assigned a color, in our specific instance every truck has infinite capacity. So, the set of containers $C = \{c_i\}_{i=1}^{|V(G)|}$, the 3 trucks and the set of constraint pairs $T = \{(c_i, c_j) : (v_i, v_j) \in E(G)\}$ form the specific instance of TLP. Note

that each step of this construction can be done in polynomial time.

If $3CP$ is solvable, then a valid assignment of 3 colors $\{1, 2, 3\}$ exists to each of the vertices. That is no two adjacent vertices are assigned the same color. Now, consider the sets $V_1 = \{v \in G : \text{v is assigned color 1}\}$, $V_2 = \{v \in G : \text{v is assigned color 2}\}$ and $V_3 = \{v \in G : \text{v is assigned color 3}\}$. Note that no two vertices in each of these sets are connected by an edge. Now, we load the containers c_i that correspond to vertices v_i in the set V_1, into the first truck. By construction, every pair of these containers can be placed together since their corresponding vertices are not connected by an edge. Similarly loading the second truck with containers c_i whose corresponding vertices are in V_2 and similarly for the third truck, we see that we can load all the $|V(G)|$ containers into 3 trucks and thus solve the specific instance of TLP. So, $3CP$ **is solvable** $\Rightarrow$ TLP **is solvable**.

If TLP is solvable, then we can load the $|V(G)|$ containers into three trucks. Let the sets C_1, C_2, and C_3 be the set of containers loaded into each of the three trucks. Now, we assign colors to the vertices of G as follows: For a vertex v_i color it with color j if the corresponding container c_i is in the set C_j. Note that since no two containers in each of the trucks are a constrained pair (which means they can be placed together), thus no two vertices assigned the same color are connected by an edge. Since this is true for all 3 colors, we have obtained a valid 3-coloring of the graph G. Hence, we have solved the arbitrary instance of $3CP$. So, TLP **is solvable** $\Rightarrow$ $3CP$ **is solvable**.

So, we have proved that TLP is solvable if and only if $3CP$ is solvable. If B is a black box that efficiently solves TLP, we can take the given $3CP$ instance, construct the equivalent TLP instance (as done above), and ask if the $|V(G)|$ containers can be loaded into three trucks such that no two containers that can't be together are in the same truck. If B answers yes, we can infer from the above equivalence that the $3CP$ instance has a valid 3-coloring. Else, no. Thus, we have seen that $3CP \leq_p TLP$ that is:

$$\texttt{3-Coloring\_Problem} \leq_p \texttt{Truck\_Loading\_Problem}$$

$3CP$ is a known $NP - complete$ problem. Since, $3CP \leq_p TLP$, we can infer that $TLP \in NP - complete$.

---

6. (15 points)

One thing that's not always apparent when thinking about traditional "continuous math" problems is the way discrete, combinatorial issues of the kind we're studying here can creep into what look like standard calculus questions.

Consider, for example, the traditional problem of minimizing a one-variable function like $f(x) = 3 + x - 3x^2$ over an interval like $x \in [0, 1]$. The derivative has a zero at $x = \frac{1}{6}$, but this in fact is a maximum of the function, not a minimum; to get the minimum, one has to heed the standard warning to check the values on the boundary of the interval as well. (The minimum is in fact achieved on the boundary, at $x = 1$.)

Checking the boundary isn't such a problem when you have a function in one variable; but suppose we're now dealing with the problem of minimizing a function in n variables $x_1, x_2, \ldots, x_n$ over the unit cube, where each of $x_1, x_2, \ldots, x_n \in [0, 1]$. The minimum may be achieved on the interior of the cube, but it may be achieved on the boundary; and this latter prospect is rather daunting, since the boundary consists of 2^n "corners" (where each x_i is equal to either 0 or 1). Calculus books tend to get suspiciously vague around here, when trying to describe how to handle multivariable minimization problems in the face of this complexity.

It turns out there's a reason for this: Minimizing an n-variable function over the unit cube in n dimensions is as hard as an NP-complete problem. To make this concrete, let's consider the special case of polynomials with integer coefficients over n variables $x_1, x_2, \ldots, x_n$.

To review some terminology, we say a monomial is a product of a real-number coefficient c and each variable x_i raised to some nonnegative integer power a; we can write this as $cx_1^{a_1}x_2^{a_2}...x_n^{a_n}$. (For example, $2x_1^2x_2x_3^4$ is a monomial.) A polynomial is then a sum of a finite set of monomials. (For example, $2x_1^2x_2x_3^4 + x_1x_3 - 6x_2^2x_3^2$ is a polynomial.)

We define the Multivariable Polynomial Minimization Problem as follows: Given a polynomial in n variables with integer coefficients, and given an integer bound B, is there a choice of real numbers $x_1, x_2, \ldots, x_n \in [0, 1]$ that causes the polynomial to achieve a value that is $\leq B$?

Show that 3-SAT $\leq_p$ Multivariable Polynomial Minimization.

Solution: Let us denote the Multivariable Polynomial Minimization problem by MPM. We wish to prove that 3-SAT $\leq_p MPM$. We first take an arbitrary instance of 3-SAT, which is a conjunction of n clauses over k, variables, and construct a specific instance of MPM. So, for each clause C_i, we construct a polynomial P_i over $[0, 1]^3$, which is a product of 3 terms one corresponding to each literal in the clause and P_i evaluates to 0 only when the clause is satisfied else it evaluates to 1. Let $C_i = (l_{i1} \vee l_{i2} \vee l_{i3})$, where l_{i1}, l_{i2} and l_{i3} are the literals.

If the literal $l_{ij} = x_p$ (literal is the true form of the variable), then the corresponding term in P_i will be $(1 - X_p)$, where $X_p \in [0, 1]$. On the other hand, if $l_{ij} = \overline{x_p}$ (negation of variable), then the corresponding term in P_i is X_p, where $X_p \in [0, 1]$. So the polynomial P has one variable for each boolean variable in the boolean formula $F = \bigwedge_{i=1}^{n} C_i$. So if the clause C_i is say:

$$C_i = (x_\alpha \vee \overline{x_\beta} \vee x_\gamma)$$

then the polynomial P_i takes the form:

$$P_i = (1 - X_\alpha)X_\beta(1 - X_\gamma)$$

Once, we construct each P_i for all $i = 1, 2, \ldots, n$, we construct the MPM instance as:

$$P = \sum_{i=1}^{n} P_i$$

Since P is a polynomial in k variables and domain of each is $[0, 1]$, the domain of P is $[0, 1]^k$. So, we wish to find an assignment of $X_1, X_2, \ldots, X_k$ in this hypercube $[0, 1]^k$ such that P attains a value ≤ 0. Note that each of these steps take polynomial time. Next, we prove that the two instances are equivalent.

If the 3-SAT instance is solvable, then there exists a valid assignment of the variables $(x_1, x_2, \ldots, x_k)$ that satisfies all n clauses. Now, we consider the assignment for $(X_1, X_2, \ldots, X_k)$ where each X_i:

$$X_i = \begin{cases} 1 & x_i = true \\ 0 & x_i = false \end{cases}$$

Note that by this assignment and by construction all the n terms in P evaluate to zero. Hence, P attains value ≤ 0 (in fact equality here) in the hypercube (since each term can be 0 or 1) when the 3-SAT instance is solvable. So, **3-SAT solvable** $\Rightarrow$ MPM **is solvable**.

If MPM is solvable, then the constructed polynomial P attains value ≤ 0 in the hypercube. By construction, this happens when P evaluates to zero. This value is attained when each term P_i of P evaluates to 0. Each term P_i in P corresponds to a clause in the arbitrary 3-SAT instance F. Also, we have constructed such that the P_i evaluates to 0 only when C_i is satisfiable. Next, we get the corresponding assignment of the boolean variables as:

$$x_i = \begin{cases} true & X_i > 0 \\ false & X_i = 0 \end{cases}$$

Now, since every term evaluates to zero, every clause in F evaluates to *True* for the assignment defined above. Hence, we get a solution for our 3-SAT instance. So, MPM **is solvable** $\Rightarrow$ **3-SAT is solvable**.

So, we have proved that MPM is solvable if and only if 3-SAT is solvable. If B is a black box that efficiently solves MPM, we can take the given 3-SAT instance, construct the equivalent MPM instance (as done above), and ask if there exists an assignment of the variables $\{X_i\}_{i=1}^k$ in the hypercube where constructed polynomial P attains a value ≤ 0. If B answers yes, we can infer from the above equivalence that the 3-SAT instance has a valid assignment of the boolean variables $\{x_i\}_{i=1}^k$ that satisfies the boolean formula F. Else, no. Thus, we have seen that 3-SAT $\leq_p MPM$ that is:

$$\texttt{3-SAT} \leq_p \texttt{Multivariable\_Polynomial\_Minimization}$$
