CMSC 641 Design & Analysis of Algorithms — Spring 2023
Section 01 — Homework 10

Faisal Rasheed Khan

VB02734

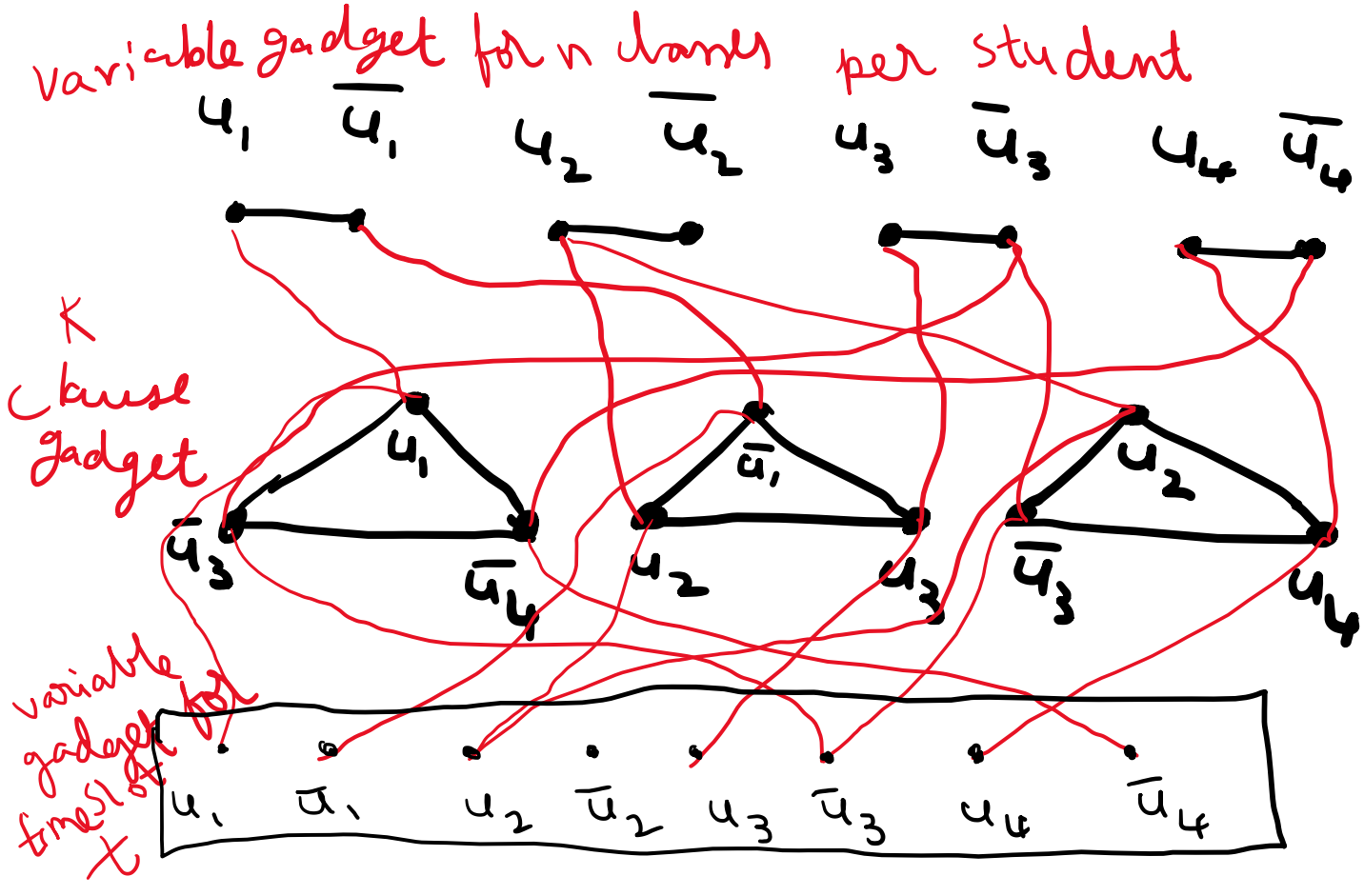
vb02734@umbc.edu

1 Class Scheduling

At some universities, class scheduling proceeds in a different manner than at UMBC. One process starts with each department producing a list of classes that will be offered the next semester. (This list just has the classes offered and does not include any information about timing.) Next, the students indicate which classes they will take. Finally, the university Registrar assigns one of the standard time slots to each class. Suppose that there are n classes, m students, t standard time slots and that each student is allowed to sign up for at most k classes. The Registrar wants to determine if it is possible to assign a standard time slot to each class so that none of the students have a scheduling conflict — i.e., none of the students have two of the classes they want to take assigned to the same time slot. Below, you will show that the Registrar's Class Scheduling Problem (RCSP) is NP-complete by selecting one of the problems we have shown NP-complete during lecture and reducing that problem to RCSP.

1. Pick X to be one of 3SAT, 3-Color, k-Clique, Vertex Cover, Partition, Traveling Salesman or 3-Dimensional Matching. Devise and describe a polynomial-time function f that takes an instance of X and produces an instance of RCSP. Do briefly justify that your function f runs in polynomial time.

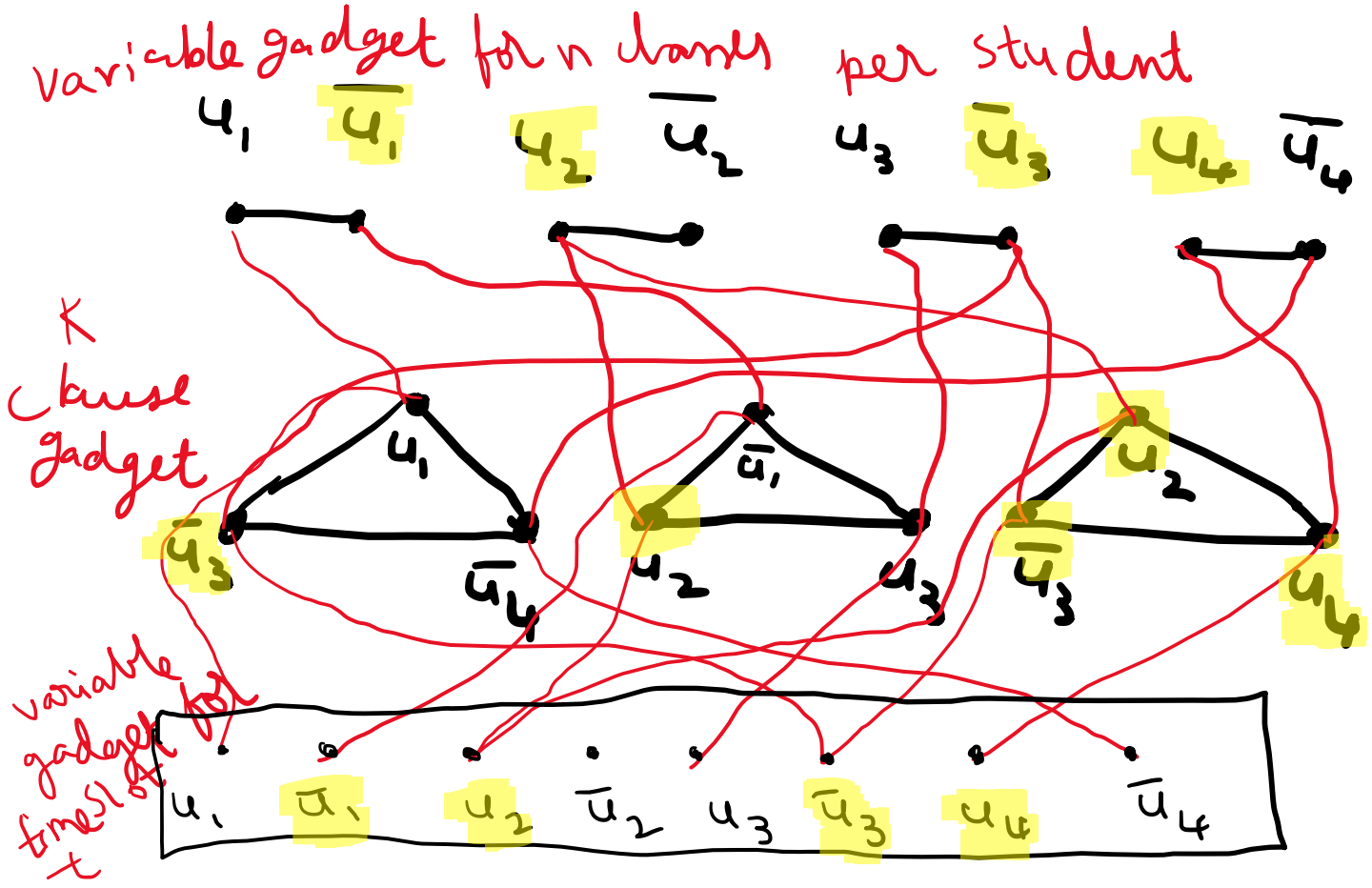
A. If we reduce 3 satisfiability to any function then it belongs to NP.
$3SAT \leq_m^P RCSP$
Consider boolean function, $\phi = (u_1 \vee \bar{u}_3 \vee \bar{u}_4) \wedge (\bar{u}_1 \vee u_2 \vee u_3) \wedge (u_2 \vee \bar{u}_3 \vee u_4)$



ϕ has n variables and k clauses per student
G has 2n+3k+2n nodes and n+6k edges
C=n+3k+2n
C=3n+3k
n classes has 2n nodes which are used to select or to not select the class, here one variable should be either true or false
Based on the class variable true literal the time slot is assigned such that no overlapping of the time is there since we are assigning unique time slot for the literal.
We have 4n+3k nodes and n+6k edges for every student which runs in polynomial time as 3SAT reduces into polynomial time.
Running time=(5n+9k)*m which is a polynomial time, where m =total students

2. Prove that for all instances x of X that if x ∈ X then f(x) ∈ RCSP.

A. Claim: *ϕ ∈ 3 SAT <=> G has assignment of a student with n classes (atmost k classes assigned such that the timeslot do not over lap)*
Proof:
Suppose ϕ is satisfiable
Consider a satisfying assignment if u_i is true, pick u_i in variable gadget otherwise pick $\bar{u}_i$
Each clause must have a true literal, pick true node in clause gadget.
u_1=false, u_2=true, u_3=false, u_4,=true



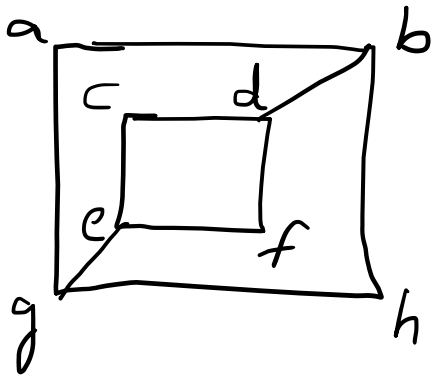
Each node at variable gadget should have edge either to u_i *or* $\bar{u}_i$ *, if time slot should not overlap.*
The above conditions of ensuring true literal at clause ensures that a student is assigned to atmost k classes and their time slots do not overlap by checking the timeslot variable gadget.
The above condition satisfies the satisfying assignment x ∈ X and the gadget ensures that f(x) ∈ RCSP

3. Prove that for all instances x of X that if f(x) ∈ RCSP then x ∈ X.

A. For f(x) ∈ RCSP, it means that there is no conflict with the time slot for the classes. This means that we have classes assigned to time slot without overlapping.
The students don't have time overlap for classes we have timeslot variable gadget which has at least a true literal for either u_i *or* $\bar{u}_i$.
This ensures that each variable clause has a true literal, the literal which is selected has the edge to the clause and the clause has edge to time slot gadget such that no 2 or more class schedules of a student overlap.
All the requirements are fulfilled for the RCPS and its satisfying assignment.
For f(x) ∈ RCSP we have true literal for every variable and each clause has a true variable.
Therefore x ∈ X.

2 Lucky Cover

The performance of the 2-approximation algorithm for Vertex Cover is very much dependent on which edges are chosen, since the endpoints of the chosen edge are added to the approximate vertex cover. Consider the graph below.



Questions:

1. What is the best possible sequence of edges that the algorithm might have picked? (Here best means the resulting vertex cover is smallest possible for the algorithm.)

A. The best possible sequence of edges that the algorithm is edge d,b and edge e,g.
The above vertices d,b,e,g covers all the edges for the graph i.e. every edge is covered by the vertices d,b,e,g.
d-b covers b-a, b-h, d-c, d-f
e-g covers e-c, e-f, g-a, g-h

2. What is the worst possible sequence of edges that the algorithm might have picked? (Similarly, worst means the resulting vertex cover is the largest possible for the algorithm.)

A. The worst possible sequence of edges that the algorithm is not including edge d,b and edge e,g. As it has the best possible sequence of edges.
Our worst possible sequence contains every possible vertices in the graph.
We pick edge a-b, edge g-h, edge c-d, edge e-f
a-b covers a-g, b-h, b-d
g-h covers g-a, g-e, h-b
c-d covers c-e, d-b, d-f
e-f covers e-g, e-c, f-d

3 Approximating Three-Dimensional Matching

In the three-dimensional matching problem (3DM) we are given three sets X, Y and Z such that the sizes of the three sets are the same. Let $m = |X| = |Y| = |Z|$. We are also given a set of triples $T \subseteq X \times Y \times Z$. The NP-complete problem we discussed in class asks if there exists a perfect matching — that is, a subset of triples $T' \subseteq T$ such that $|T'| = m$ and every $x \in X$ appears in a triple in T' exactly once, every $y \in Y$ appears in a triple in T' exactly once and every $z \in Z$ appears in a triple in T' exactly once. A matching $M \subseteq T$ (not necessarily a perfect one) just requires that every $x \in X$ appears in a triple in M at most once, every $y \in Y$ appears in a triple in M at most once and every $z \in Z$ appears in a triple in M at most once. So, a matching enforces that triples in M do not share components, but does not require that every x, y and z appear in some triple in M. The optimization version of 3DM asks us to find a matching that contains the largest number of triples. We are still required to make sure that none of the selected triples overlap. For example, we are not allowed to select two triples (x, y1, z1) and (x, y2, z2) that share the same value in the first component.

Assignment: Consider the following naive (some might say stupid) approximation algorithm for 3DM: Pick an arbitrary triple (x, y, z) from T. Then, remove from T any triple that contains x, y or z. We continue picking an arbitrary triple in this manner until T is empty. Argue that this naive algorithm achieves an approximation factor of 3

A. Given $m = |X| = |Y| = |Z|$
$T \subseteq X \times Y \times Z$
$T' \subseteq T$ such that $|T'| = m$ and every $x \in X$ appears in a triple in T' exactly once, every $y \in Y$ appears in a triple in T' exactly once and every $z \in Z$ appears in a triple in T' exactly once.
There is a match M,
$M \subseteq T$ (not necessarily a perfect one) just requires that every $x \in X$ appears in a triple in M at most once, every $y \in Y$ appears in a triple in M at most once and every $z \in Z$ appears in a triple in M at most once.
Matching M ensures that the triples in M do not share the components.
Naïve approximation algorithm:
    Pick arbitrary triple (x, y, z), each x,y,z at one time
    Repeat for x,y,z
        Remove all x,_,_ from T for x
        Remove all _,y,_ from T for y
        Remove all _,_,z from T for z
To argue that the naïve algorithm achieves we need to prove:
i. Triples do not share the components

ii. Size of Matching M produced by naïve algorithm is at most 3 times the optimum solution

Claim1: Triples do not share the components

Proof:

Suppose the algorithm outputs a set of triples M. We need to show that M does not contain any overlapping triples. Suppose there exist two triples in M that share a component. let's assume that these triples are (x, y1, z1) and (x, y2, z2), i.e., they both contain the value x in their first component. Then, by the way the algorithm works, we know that both (x, y1, z1) and (x, y2, z2) were selected from T, which means that when either of them was selected, the algorithm removed from T all triples that contained x, y1, or z1, as well as all triples that contained x, y2, or z2. This means that neither (x, y1, z2) nor (x, y2, z1) can be in T, and so the algorithm cannot have selected them later. Therefore, M does not contain any overlapping triples.

Claim2: Size of Matching M produced by naïve algorithm is at most 3 times the optimum solution

Proof:

From the above claim we remove overlapping triples. consider any maximum matching optimum solution. Since every triple in optimum solution covers exactly one element from each of x, y, and z, it follows that optimum covers at most m triples. Therefore, OPT has size at most m.

Our Matching M according to the above claim1 has at most 3m size of the triples of optimum solution because of the naïve algorithm.

M≤3m and

optimum solution≤m

$\frac{M}{\text{optimum solution}} \leq \frac{3m}{m}$

$\frac{M}{\text{optimum solution}} \leq 3$

$\frac{|c|}{|c^*|} \leq 3$, Hence the algorithm achieves approximation factor of 3.

References:

https://archive.org/details/introduction-to-algorithms-by-thomas-h.-cormen-charles-e.-leiserson-ronald.pdf/page/427/mode/2up

https://archive.org/details/AlgorithmDesign1stEditionByJonKleinbergAndEvaTardos2005PDF/page/n201/mode/2up

16-NP-Completeness-post.pdf - Google Drive

17.1-SAT-vs-Coloring-post.pdf - Google Drive

18.3-Hamiltonian-Cycle-post.pdf - Google Drive

19.1-Clique-post.pdf - Google Drive

20.2-3D-Matching-post.pdf - Google Drive

21.1-Approximation-Algorithms-post.pdf - Google Drive

21.2-Approximating-TSP-post.pdf - Google Drive

22-Bin-Packing-post.pdf - Google Drive