CS 170 HW 12

Due **2021-11-22, at 10:00 pm**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write none.

In addition, we would like to share correct student solutions that are well-written with the class after each homework. Are you okay with your correct solutions being used for this purpose? Answer "Yes", "Yes but anonymously", or "No"

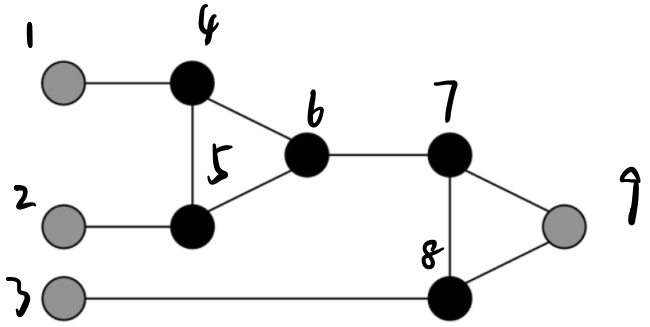
2 Reduction to 3-Coloring

Given a graph $G = (V, E)$, a valid 3-coloring assigns each vertex in the graph a color from $\{0, 1, 2\}$ such that for any edge (u, v), u and v have different colors. In the 3-coloring problem, our goal is to find a valid 3-coloring if one exists. In this problem, we will give a reduction from 3-SAT to the 3-coloring problem. Since we know that 3-SAT is NP-Hard (there is a reduction to 3-SAT from every NP problem), this will show that 3-coloring is NP-Hard (there is a reduction to 3-coloring from every NP problem).

In our reduction, the graph will start with three special vertices, labelled "True", "False", and "Base", and the edges (True, False), (True, Base), and (False, Base).

(a) For each variable x_i in a 3-SAT formula, we will create a pair of vertices labeled x_i and $\neg x_i$. How should we add edges to the graph such that in any valid 3-coloring, one of $x_i, \neg x_i$ is assigned the same color as True and the other is assigned the same color as False?

(b) Consider the following graph, which we will call a "gadget":



Show that in any valid 3-coloring of this graph which does not assign the color 2 to any of the gray vertices, the gray vertex on the right is assigned the color 1 only if one of the gray vertices on the left is assigned the color 1.

(c) We observe the following about the graph we are creating in the reduction:

(i) For any vertex, if we have the edges (v, False) and (v, Base) in the graph, then in any valid 3-coloring v will be assigned the same color as True.

(ii) Through brute force one can also show that in the gadget, for any assignment of colors to gray vertices such that:

(1) All gray vertices are assigned the color 0 or 1

(2) The gray vertex on the right is assigned the color 1

(3) At least one gray vertex on the left is assigned the color 1

Then there is a valid coloring for the black vertices in the gadget.

Using these observations and your answers to the previous parts, give a reduction from 3-SAT to 3-coloring. Prove that your reduction is correct.

3 Multiway Cut

In the multiway cut problem, we are given a graph $G(V, E)$ with k special vertices $s_1, s_2 \dots s_k$. Our goal is to find the smallest set of edges F which, when removed from the graph, disconnect the graph into at least k components, where each s_i is in a different component. When $k = 2$, this is exactly the min s-t cut problem, but if $k \geq 3$ the problem becomes NP-hard.

Consider the following algorithm: Let F_i be the set of edges in the minimum cut with s_i on one side and all other special vertices on the other side. Output F, the union of all F_i. Note that this is a multiway cut because removing F_i from G isolates s_i in its own component.

(a) Explain how each F_i can be found in polynomial time.

(b) Let F^* be the smallest multiway cut. Consider the components that removing F^* disconnects G into, and let C_i be the set of vertices in the component with s_i. Let F_i^* be the set of edges in F^* with exactly one endpoint in C_i. How many different F_i^* does each edge in F^* appear in? How do the sizes of F_i and F_i^* compare?

(c) Using your answer to the previous part, show that $|F| \leq 2|F^*|$. (Challenge: How could you modify this algorithm to output F such that $|F| \leq (2 - \frac{2}{k})|F^*|$?)

(As an aside, consider the minimum k-cut problem, where we want to find the smallest set of edges F whose removal disconnects the graph into at least k components. The following greedy algorithm for minimum k-cut gets a $(2 - \frac{2}{k})$-approximation: Initialize F to the empty set. While $G(V, E - F)$ has less than k components, find the minimum cut within each component of $G(V, E - F)$, and add the edges in the smallest of these cuts to F. Showing this is a $(2 - \frac{2}{k})$-approximation is fairly difficult.)

4 Project

Please answer the following questions after reading the project specification. They will help you build intuition for how to construct your inputs and solvers in the project.

(a)

pick s_i as source

Dinitz $O(V^2E)$

run $|V|-1$ times find the min flow

run BFS on the residual graph

$O(V^3E)$ $O((V+1)^2(E+k))$

choose s_j as source

add dummy node U

connect (s_j, U) and make the edge weight

really big run min cut algorithm

(b) appear 2 times.

$$F \subset F^*$$

(c)

$$F_1 \subseteq F_1^* \quad F = \sum F_i < \sum F_i^* = 2F^*$$

(a) Construct a set of igloos polishing tasks such that the problem, as specified in the project specifications, has a trivial solution. Explain why the input is trivial.

(Hint 1: Your answer should focus on the relationship between the parameters of tasks, instead of the actual numerical values.)

(Hint 2: Try setting the deadlines of all tasks to 1440.)

(b) For this question **do not** use the profit decay function specified in the specification. Instead assume that the profit for polishing is all or nothing. If an igloo is polished by the deadline, the profit gained is equal to the corresponding p_i. Otherwise if an igloo is polished after the deadline, then the profit gained is 0.

Next, find the best sequence of igloos to polish in a total of **100 minutes**, given the following igloo polishing tasks:

- Igloo 1: Duration = 30, Deadline = 65, Profit = 10
- Igloo 2: Duration = 40, Deadline = 100, Profit = 20
- Igloo 3: Duration = 15, Deadline = 80, Profit = 5
- Igloo 4: Duration = 20, Deadline = 30, Profit = 30
- Igloo 5: Duration = 25, Deadline = 40, Profit = 15

(i) Find the optimal sequence of igloos to polish

(ii) If all igloos had identical deadlines, equal to 100, would that change the optimal sequence? Why?

(c) For this question **do not** use the profit decay function specified in the specification. Instead, assume that our profit decays according to the following function, $max(0, p_i - s_i)$, where p_i is the original profit and s_i is the number of minutes by which the task was delayed past the deadline.

(i) Consider a greedy solver, where the task with the **highest benefit** is chosen at every step and scheduled as soon as possible, until we run out of time or tasks to schedule. Construct an input, of exactly 2 igloo polishing tasks, such that the greedy solver chooses a sub-optimal list of igloos. Explain why the greedy algorithm fails.

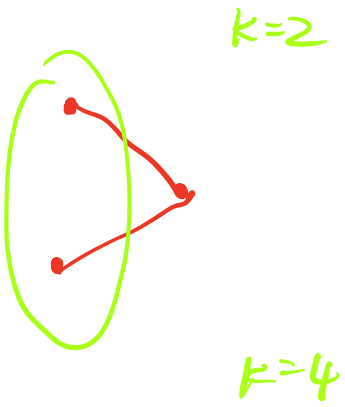
(ii) Consider a greedy solver, where the task with the **next earliest deadline** is chosen at every step, until we run out of time or tasks to schedule. Construct an input, of exactly 2 igloo polishing tasks, such that the greedy solver chooses a sub-optimal list of igloos. Explain why the greedy algorithm fails.

5 (Extra Credit) Approximation Hardness of Independent Set

Recall the maximum independent set problem: Given an undirected graph G, we want to find the largest independent set, i.e. largest set of vertices S such that no two vertices in S are adjacent. An algorithm is a α-approximation algorithm for maximum independent set if

given a graph G, it outputs an independent set of size at least OPT/α, where OPT is the size of the largest independent set.

Show that if there is a polynomial-time $2^{2^{100}}$-approximation algorithm for maximum independent set, there is a polynomial-time 2-approximation algorithm for maximum independent set.



HW12

1 Study Group

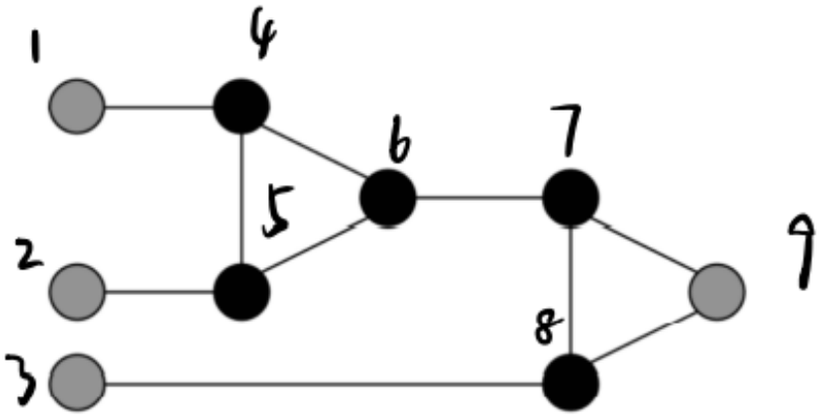
none Yes

2 Reduction to 3-Coloring

(a)

We can add the edge(x_i,¬x_i),(x_i,Base),(¬x_i,Base) into the graph. Since x_i and ¬x_i shouldn't have the same color as Base and they are connected, x_i and ¬x_i will be assigned True in one of them and the other one is False

(b)



Suppose the gray vertices in the left aren't assigned 1, then all of them will be assigned 0. WLOG, we assign 1 and 2 to vertex 4 and 5 respectively. Thus, vertex 6 is assigned 0. Since vertex 3 is 0. If vertex 9 is assigned 1. Then vertex 7 is assigned 2 since it's connected to vertex 6 and 9. There is no strategy to color vertex 8 because it's connected to 3 vertices with 3 different values. Thus, if the vertex 1,2,3 are assigned 0, vertex 9 must be assigned 0.

If one of vertex 1,2,3 switch to 1. If it's vertex 3, we can switch vertex 8 to 0 to satisfy the requirement. Otherwise, WLOG, switch vertex 1 to 1. Switch vertex 4,5 to 2. Switch vertex 7 to 0. Then it will satisfy the requirement. Proved!

(c)

We use 0 as False, 1 as True, 2 as Base in 3-Coloring.

First, we create the graph for the problem

i) For each x_i, we create vertices and edges describe in (a)

ii) For each clause, we first add a new vertex v, 2 edges (v,False) and (v, Base). Then we create a gadget. To achieve this, we set v as the gray vertex on the right and 3 gray vertices on the left as 3 variables in the clause.

We reduce the 3-SAT to 3-Coloring

If there is a 3-SAT solution, then we can assign x_i 1 and ¬x_i 0 if it's true in the 3-SAT or x_i 0 and ¬x_i 1 otherwise. Assign 1 to v since each clause must be true. Moreover, one of the gray vertex in the right must be true because we must have at least one true variable to make the clause to be true. From (b), there exist a 3-Coloring for the gadget.

If there is a 3-Coloring for the graph, then we will have a 3-SAT solution. From (a), we know that x_i and ¬x_i have exactly one 0 and one 1. Thus, we create a valid assignment. From (i), v must be colored 1. Since we should have at least one gray vertex to be assigned with 1 in each gadget, we know that at least one variable in the clause will be true. Thus, it's a valid 3-SAT solution.

Proved!

3 Multiway Cut

(a)

Main Idea

To find F_i, we set s_i as the source. Add a dummy node v as the destination and add edges (s_j,v) (j ∈{1,2,...,i-1,i+1,...,k})into the graph with positive infinite weight. Run max flow algorithm on G. Then run BFS on the residual graph to find the min cut.

Proof of Correctness

After running the algorithm, we will get a min cut with one part containing s_i and other part containing the rest k-1 special vertices. Because the 2 parts must contain s_i and v and the edge cross the min cut with source s_i and sink v will not have (s_j,v) (j not equal to s). Remove v, we will get a min cut in the original graph.

Runtime

The max flow algorithm is O((E+k)V^2) which is covered in previous homework when we always take the the flow with the shortest distance.

Then we run DFS to find the min cut in the residual graph.(All edges which are from a reachable vertex to non-reachable vertex are minimum cut edges) O(V+E)

Total runtime O((E+k)V^2)

(b)

Each edge appears in 2 F_i* because the edge will be counted if it connects 2 components. Thus, it will appear in the 2 components' F_i*s.

Also, by definition F_i* disconnects s_i from other special vertices. Thus, F_i has fewer edges since F_i is the min cut disconnecting s_i from all other special vertices. Thus, |F_i|<=|F_i*|.

(c)

Notice that the size of F is at most as the sum of F_i. From (b), we have

|F|<= sum(|F_i|)<=sum(|F_i*|)=2|F*|

After computing all the F_i. We union all F_i except the one with the largest size(denote by F_j). This will still result in a multiway cut because other F_i guarantee that s_j is disconnected with s_i.

Thus, |F|<=sum(F_i(i≠j))<=(1-1/k)sum(F_i)<=(1-1/k)sum(F_i*)=(2-2/k)|F*|

4 Project

(a)

Let's say we have n igloos.

t_i=t_(i+1)=1440, d_i<=d_(i+1),p_i>=p_(i+1) , i={1,2,...,n-1}

The input is trivial since the igloo with the highest profit always has the shortest duration to finish.

Notice the deadline for each igloo is the same. If we don't choose the igloo in order of their profit, switch the order may cause the decay bigger(because it has a bigger profit) if their total duration exceeds the deadline.

(b)

(i)

4-1-2

Notice that we can only do one igloo in igloo 4 and 5. We will do igloo 4 since it has a shorter duration and larger profit. If we then polish igloo 2, then we only have choice igloo 3. The sum of profit is 55.

If we choose igloo 3 at the second step, then we can only do one of igloo 1 and 2 afterwards. Thus, the sum of profit is 55.

If we choose igloo 1 at the second step, then we can only do one of igloo 2 and 3. Thus, the sum of profit is 60 which is the largest.

(ii)

5-4-3-2

Notice the sum of duration for all igloos is 130. Then we must throw some igloos with sum of duration over 30. We only need to throw one if it's in igloo 1 and 2. The max profit from them is 70 by throwing igloo 1. This is the best strategy since we need to throw 2 igloos if we don't choose igloo 1 or 2 then any sum of profit in 2 of igloo 3,4,5 is over 10.

(c)

(i)

1 80 100 100

2 20 20 50

The output for the greedy solver is 100 while the real highest profit is 150

(ii)

1 20 10 20

2 40 40 100

The output for the greedy solver is 90 while the real highest profit is 100

5 (Extra Credit) Approximation Hardness of Independent Set

First, let I denotes a set of instance and S(I) be the solution set for a optimization on I. A is an algorithm.

In approximation, we want the absolute difference for A(I) and OPT(I) as small as possible.

We define A is a k-absolute approximation algorithm if $|A(I) - OPT(I)|<k$

In the problem, we call S(I) a α-approximate solution if $S(I)>=|OPT(I)|/\alpha$

For a given graph G with size of maximum independent set k(OPT(G)=k), we construct a new graph G' which is k copies of instance G. Then a maximum independent set in G' is in size of k^2(OPT(G')=k^2).

Then we have G has maximum independent set k <==> G' has maximum independent set k^2

In the problem, we can repeat multiple times to turn factor $2^{(2^{100})}$ to 2. Thus, if there is a polynomial time $2^{(2^{100})}$ approximation algorithm for maximum independent set, there is a a polynomial time 2 approximation algorithm for maximum independent set.

Actually, we can find k-absolute approximation is impossible since OPT(G')-k implies we reach one maximum approximation in one of the copies.(G' is k+1 copies here)