HW12

1 Study Group

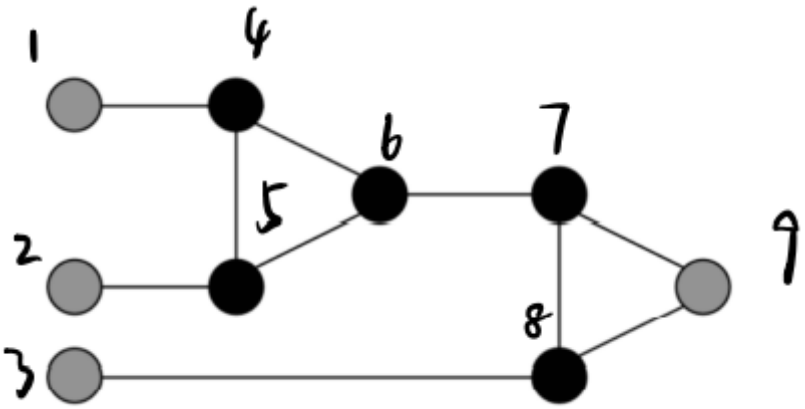
none Yes

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)