CS 170 HW 12

Daniel Deng, SID 3034543526

1 Study Group

(a) Auston Lin, SID 3034554056

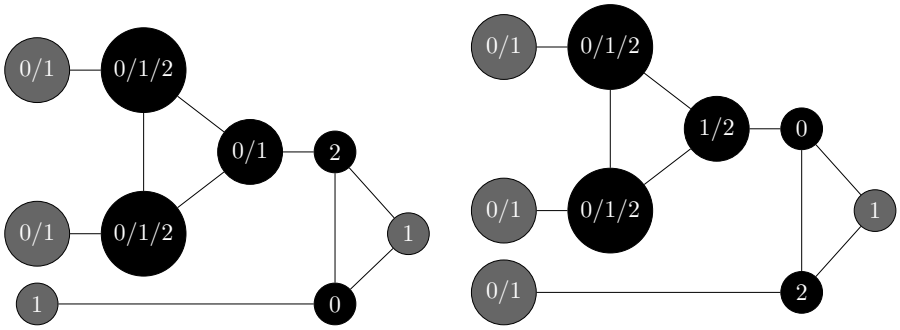
(b) Yes

2 Reduction to 3-Coloring

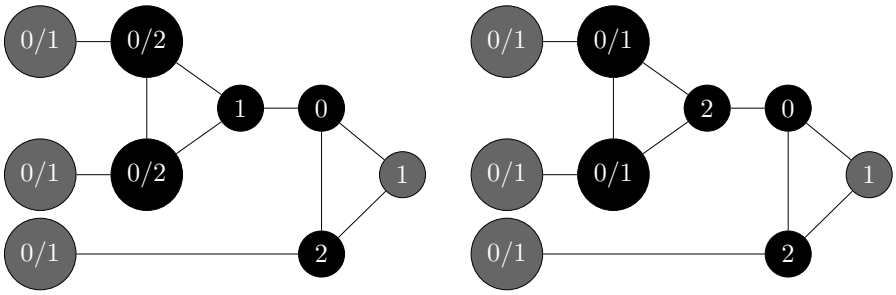
(a) $\forall x_i$, add edges $(x_i, base), (\neg x_i, base), (x_i, \neg x_i)$ to the graph. Connections to the "base" vertex ensures that x_i and $\neg x_i$ can only be "'true" or "false". Connection between x_i and $\neg x_i$ ensures that if one is "true" the other must be "false".

(b) (Right grey vertex = color 1 $\implies$ $\exists$ left grey vertex = color 1)

Proof. After setting the right grey vertex to be color 1 and assigning its immediate neighbors, we get two cases after pruning the domains



The left case already has one of the left grey vertices be of color 1. For the right case, we perform another round of assignment, and get another two cases



In both cases, since at least one of the black vertices in the left triangular structure has to take color 0, one of the two top left grey vertices must take color 1.

Therefore, 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 given the constraint. □

(c) **Algorithm Description:** Given a 3-SAT instance with n variables and m clauses.

First, for each clause, construct a gadget such that the left grey vertices correspond to the variables or negations of variables in the clause.

Then, connect all left grey vertices from all gadgets to "Base", and all right grey vertices to "Base" and "False".

Finally, pairwise connect the left grey vertices from all gadgets such that one of the left grey vertex in a pair corresponds to variable x_i and the other corresponds to $\neg x_i$.

Define "True" = color 1, "False" = color 2, and "Base" = color 3 and find a 3-Coloring assignment for the constructed graph. If the graph is not 3-colorable, the 3-SAT instance is not satisfiable; otherwise, the color assignment of the left grey vertices is the true/false assignment of the variables and variable negations in the clauses.

Proof of Correctness: First, note that in the construct graph all right grey vertices are forced to have color 1 due to their connection to both "Base" and "False" (based on *observation(i)*) and any pair of left grey vertices corresponding to x_i and $\neg x_i$ but be that one is assigned color 1 and the other is assigned color 0 (as shown in *part(a)*).

- If the 3-SAT instance has a satisfying assignment, it means that at least one of the variables (or variable negations) in all clauses is true. This would mean that in the corresponding 3-Coloring graph, at least one of the left grey vertex in any gadget has color 1. Based on *observation(ii)*, there is a valid coloring for all black vertices in all gadgets. Therefore, 3-Coloring will also have a satisfying assignment.
- If the 3-Coloring graph has a satisfying assignment, since all right grey vertices have color 1, based on what we've shown in *part(b)*, at least one of the left grey vertices must also have color 1 in all gadgets. This means that at least one variable (or variable negation) is true in all corresponding clauses, indicating that the 3-SAT instance can also be satisfied by the assignment.

Therefore, the reduction is correct.

3 Multiway Cut

(a) The problem of finding F_i can be reduced the minimum s,t cut problem. Assuming that G is undirected and unweighted, we can construct a flow graph from G that replaced each undirected edge with a forward and a backward directed edge with edge capacities 1. We then add a dummy sink node that contains an in-edge with infinity edge capacity from each special vertex except s_i. Run the minimum s,t cut algorithm, which is polynomial time, on the flow graph, and we will find F_i in polynomial time. This algorithm is correct because the minimum cut must result in all non-s_i special vertices occurring in the same set as t as the newly added edges with infinite edge capacity will never be included be included in the min-cut.

(b) Each edge in F^* must only appear in 2 different F_i^* (one for each endpoint) since one vertex cannot appear in multiple C_i.

$|F_i| \leq |F_i^*|$ because F_i is the minimum cut that separates s_i from other special vertices; therefore, $|F_i^*|$ can only be as small as $|F_i|$.

(c)

$$F = \cup_i F_i \implies |F| \leq \sum_i |F_i|$$

$$\text{Each edge in } F^* \text{ appear in 2 } F_i^* \implies 2|F^*| = \sum_i |F_i^*|$$

$$|F_i| \leq |F_i^*| \implies |F| \leq \sum_i |F_i| \leq \sum_i |F_i^*| = 2|F^*|$$

4 Project

(a) A set of n igloo polishing tasks such that each task has the same duration of d, the same profit p, and the same 1440 deadline has the trivial solution of picking any sequence of tasks in whatever order as long as the total time does not exceed 1440. Since the order to execute identical tasks is irrelevant as long as we maximize the number of tasks to be completed by the end of the day, we can pick the tasks in whatever way we want, and the solution is always trivial.

(b) (i) The optimal sequence of igloos to polish is: $4 \to 1 \to 2$.

(ii) Yes. The optimal tasks to be completed would be $2, 3, 4, 5$ in whatever order, since we can schedule the tasks into the 100-minute interval as tightly as possible without the constraint of individual deadlines.

(c) (i)
- Igloo 1: Duration = 10, Deadline = 1, Profit = 10
- Igloo 2: Duration = 2, Deadline = 2, Profit = 2

The greedy solver would choose the sequence $1 \to 2$, since task 1 has a higher profit than task 2, resulting in an overall profit of 1 since it is past deadline for task 1 by 9 minutes, whereas the optimal sequence is $2 \to 1$, resulting in an overall profit of 2.

(ii)
- Igloo 1: Duration = 10, Deadline = 1, Profit = 1
- Igloo 2: Duration = 2, Deadline = 2, Profit = 10

The greedy solver would choose the sequence $1 \to 2$, since task 1 has an earlier deadline, resulting in a total profit of 0, whereas the optimal sequence is $2 \to 1$, resulting in a total profit of 10.