Communications and Transport Systems
Department of Science and Technology **Fall 2018**
Linköping University

Christiane Schmidt

Design and Analysis of Algorithms Part 2 - Approximation and online algorithms homework 4, 05.11.2018

Problem 1 (NP-Completeness of the Dominating Set Problem):
Dominating Set Problem:
Instance: Graph $G = (V, E)$, positive integer $K \leq |V|$.
Question: Is there a subset $V' \subseteq V$ such that $|V'| \leq K$ and such that every vertex $v \in V \setminus V'$ is joined to at least one member of V' by an edge in E?

Vertex Cover Problem:
Instance: Graph $G = (V, E)$, positive integer $C \leq |V|$.
Question: Does G contain a vertex cover of size at most C?

Show the Dominating Set Problem to be NP-complete by reducing Vertex Cover to it.

Problem 2 (First-Fit-Decreasing for Bin Packing):
Show that the First-Fit-Decreasing Algorithm for Bin Packing presented in class has an approximation factor of 3/2.

Problem 3 (Bin Packing II):
Consider another algorithm for MIN BIN PACKING: the next fit algorithm. At each step there is exactly one open bin B_j. The next item is packed into B_j if it fits, otherwise, a new bin B_{j+1} gets opened, B_j gets closed and will never be opened again.

(a) Show that the next fit algorithm has an approximation factor of 2.

(b) Show that the bound from (a) cannot be improved.

Problem 4 (MAX CUT):
We consider the problem MAX CUT:
Input: an undirected graph $G = (V, E)$ with vertex set V and edge set E.
Output: a partition $(S, V \setminus S)$ of the vertex set, such that the size $w(S)$ of the cut, that is, the number of edges between S and $V \setminus S$, is maximized.

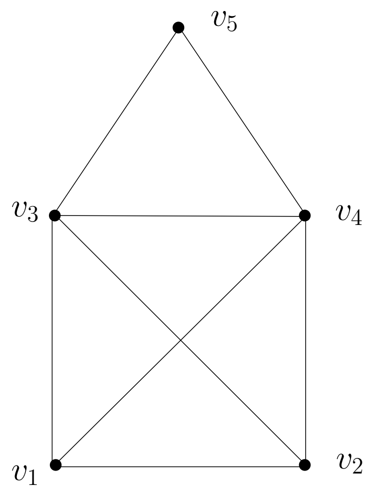

Figure 1: Graph G.

(a) Consider the example graph G from Figure 1. Give a MAX CUT S for G. What is its size?

The problem MAX CUT is NP-hard, hence, we consider the following approximation algorithm:

Algorithm

1 $S = \varnothing$

2 while $\exists v \in V : w(S\Delta\{v\}) > w(S)$ do

3 $\qquad S = S\Delta\{v\}$

4 return S

Here, Δ gives the symmetric difference of two sets, so:

$$S\Delta\{v\} = \begin{cases} S \cup \{v\} & : \quad v \notin S \\ S \setminus \{v\} & : \quad \text{otherwise} \end{cases}$$

So our algorithms starts with a vertex set S and as long as there exists a vertex that if added or deleted from S increases the current cut, S is adapted accordingly (with a local improvement).

(b) Apply the algorithm to the graph H from Figure 2. In case of ties use the following rule: prefer adding vertices over deleting vertices; in case there still is a tie, use the vertex with the smallest index.

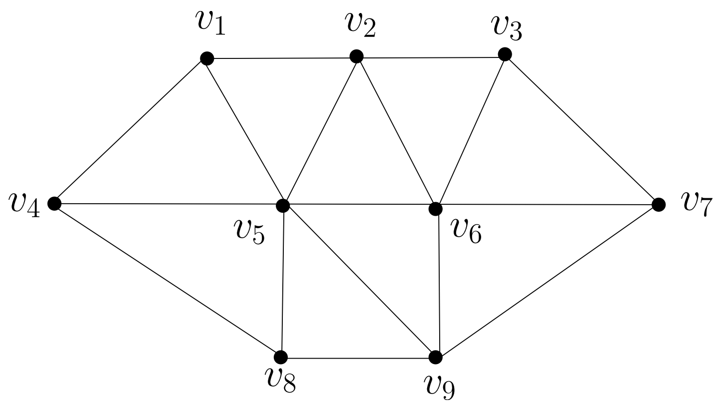

Figure 2: Graph H.

(c) Show: for every given input the algorithm outputs a cut of size $w \geq \frac{1}{2}OPT$, where OPT denotes the size of an optimal cut.

(d) Show that the algorithm has polynomial running time.

(e) Was the analysis from (c) best possible? That is, is there a graph $G = (V, E)$, such that the algorithm finds a feasible solution $S \subseteq V$ with $w(S) = \frac{1}{2} \cdot OPT(G)$? (Give a graph with an arbitrary number of nodes.)

Problem 5 (Greedy for $(0,1)$-Knapsack):
Show that the greedy algorithm, which sorts the objects by decreasing order of the ratio profit to size and then greedily picks objects, can be arbitrarily bad for the $(0,1)$-knapsack problem, in which an object can only be chosen as an entire object or be neglected completely.