Exercise set VII, Algorithms and Networks 2017-2018

Please hand in your solutions either by email to H.L.Bodlaender@uu.nl or on paper during the course or in the mailbox of Hans Bodlaender in the coffeeroom of the 5th floor of the Buys Ballot Gebouw. Note that you may work alone or work in pairs, see the website for further details. The deadline for this exercise set is Monday, January 29th 2017, 23:59.

In case you hand in your work by email:

- Have [AN7] in the header of the email.
- Attach ONE pdf-file to your email. (If you use Word, note that Word can output the text as pdf-file.)
- Make sure that your name(s) and student number(s) is/are in the pdf-file.

You can write either in English or in Dutch. Your work should be well phrased, readable, understandable, and of course, correct. It is not your teachers job to figure out what you might mean with what you have written. As such, both messy work and work that is difficult to understand can be graded with a 0.

1. Approximation Algorithms for Maximum Weight Satisfiability (1 point)

The maximum weight satisfiability problem is a variant of the maximum satisfiability problem where each clause has a positive real weight.

> Maximum Weight Satisfiability
> **Instance**: A set of variables $X = \{x_1, x_2, \ldots, x_n\}$, a collection of clauses $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$, and a set of weights $w_1, w_2, \ldots, w_m$ giving clause C_i weight w_i.
> **Question:** Find a truth assignment to the variables that maximises the sum of the weights of the clauses that are satisfied by the assignment.

Question: Give a 2-approximation algorithm for Maximum Weight Satisfiability. Give a short proof of the approximation guarantee.

2. Inapproximability of the Domatic Number (2 points)

Recall that a dominating set D in an undirected graph $G = (V, E)$ is a subset $D \subseteq V$ such that $\bigcup_{d \in D} N[d] = V$, or, in other words, a vertex subset such that for every $v \in V$ either $v \in D$ or v has in neighbour u that is in D. A *domatic k-partition* of a undirected graph $G = (V, E)$ is a partitioning of the vertices V into k disjoint sets $V_1, V_2, \ldots, V_k$ such that each V_i is a dominating set.

The problem whether a given graph G has a domatic k-partition is:

1. trivial for $k = 1$;

2. polynomial-time solvable for $k = 2$ (if G contains an isolated vertex then there is no such partition, otherwise constructs a maximal independent set I giving the domatic 2-partition $(I, V \setminus I)$);

3. NP-complete for $k \geq 3$.

Note that if G has a domatic k-partition, then it also has a domatic ℓ-partition for all $1 \leq \ell \leq k$ as one can merge dominating sets from the domatic k-partition to obtain a smaller partition that is also domatic.

> Domatic Number
> **Instance**: an undirected graph $G = (V, E)$.
> **Question:** Find the maximum value of k such that a domatic k-partition exists.

Questions:

a) Use the gap technique to show that Domatic Number cannot be approximated with approximation ratio c, for $c < 1\frac{1}{2}$ unless $P = NP$.

b) Prove that there is no PTAS for Domatic Number.

3. APX-Completeness of Maximum Not-All-Equal Satisfiability (3 points)

We consider the maximum Non-All-Equal Satisfiability problem. In this problem, a clause C with literals from a set of variables X is satisfied by an assignment of truth values ϕ to the variables in X, if and only if, the assignment causes at least one literal in C to be $True$ and at least one literal in C to be $False$ (i.e., the values of the literals are *not all equal*).

> Maximum Not-All-Equal Satisfiability
> **Instance**: A set of variables $X = \{x_1, x_2, \ldots, x_n\}$, a collection of clauses $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$.
> **Question:** Find a truth assignment to the variables that maximises the number of clauses in which the values of the literals are not all equal (i.e., maximises the number of satisfied clauses, when satisfied is defined as above).

Questions:

a) Show that the Maximum Not-All-Equal Satisfiability is in APX by giving a constant factor approximation algorithm for it.

Hint: prove that the greedy algorithm below is a 2-approximation by proving that a solution produced by the algorithm satisfies at least $\frac{1}{2}m$ clauses.

Repeat while there are variables without value assigned:
Let $\mathcal{D}$ be the set of clauses that are not yet satisfied (that do not have both a literal set to $True$ and a literal set to $False$ by the assigned variables)
Let x be the unassigned variable that occurs the maximum number of times in clauses in D (break tries arbitrarily).
Assign that value to x such that the most new clauses are satisfied (again, break ties arbitrarity)

b) Prove that Maximum Not-All-Equal Satisfiability is APX-hard by giving an L-reduction from the APX-complete problem Maximum 2-Satisfiability.

Hint: in the reduction you need to add only one additional variable to your instance.

c) Conclude that Maximum Not-All-Equal Satisfiability is APX-complete.

Be careful in this exercise with your proofs. It is easy to make a mistake. We expect that all the basic steps in the required proofs are clearly there.

4. Treewidth of series parallel graphs (2 points)

A special type of graphs are the series parallel graphs.

Series parallel graphs have two special vertices, their *terminals*. G is a series parallel graph with terminals s, t, if one of the following three cases holds:

- G is a single edge $\{s, t\}$.
- There are series parallel graphs G' with terminals s', t', and G'' with terminals s'', t''. We obtain G by taking the disjoint union of G' and G'' and then identifying t' and s''. (Series-composition.)
- There are series parallel graphs G' with terminals s', t', and G'' with terminals s'', t''. We obtain G by taking the disjoint union of G' and G'' and then identifying s' and s'', and identifying t' and t''. (Parallel-composition.)

Show that series parallel graphs have treewidth at most two.

(Hint: use an inductive argument, and pay attention to what vertices belong to the bag at the root of the decomposition tree.)

5. Independent dominating set on trees (2 points)

Give a linear time algorithm that computes the minimum size of an independent dominating set for a given tree.

Hint: use dynamic programming.

What values are you computing for subtrees? Give recursive formulas for these. Show that these give a linear time algorithm.