CS 331 Assignment #5

Greg Plaxton

March 31, 2014

NOTE ADDED 3/31/14. At my office hour today, it was pointed out that Exercise 2 should include an assumption that the MCM M in Algorithm A is an MWMCM. Accordingly, I have added the sentence "Assume that the MCM M of G associated with this execution of Algorithm A is an MWMCM of G" to the statement of Exercise 2. END OF NOTE.

There are three parts to this assignment. The first part, described in Section 1.2 below, consists of six paper-and-pencil exercises. You are required to turn in solutions to any two of these six exercises. (If you turn in more than two solutions, the grader will arbitrarily choose two of them to grade.) You are encouraged to attend the discussion sections and office hours to get hints on how to solve these exercises. You are also permitted to ask for clarifications on Piazza. You are not allowed to work with other students on these exercises. This part is due *at the start of class* on Wednesday, April 9. Please note that no extension will be given on this part of the assignment!

The second part, described in Section 1.3 below, consists of a programming task that is due by 8pm on Monday, April 14. As in preceding assignments, you should follow the instructions in the document entitled "Guidelines for Programming Tasks", which may be found in the *Assignments* section of the class website. For this programming assignment, the secondary deadline is 8pm on Thursday, April 17.

The third part, described in Section 2 below, consists of recommended exercises. There is nothing to be turned in for this part of the assignment, as these exercises will not be graded. You are encouraged to work on the recommended exercises in order to prepare for the tests.

1 Programming & Problem Solving

In this part of the assignment we build on the results established in the Programming & Problem Solving part of Assignment 4. (Accordingly, you may wish to review the definitions appearing in Assignment 4.) Our goal is to determine suitable selling prices for the items in a unit-demand auction. In the programming part of this assignment, you will compute such prices for the auction of Assignments 1 through 3, which as we have seen is a special case of a unit-demand auction.

1.1 A Pricing Algorithm

Consider the following iterative algorithm, which we refer to as Algorithm A. Algorithm A takes as input a configuration $G = (U, V, E)$, an MCM M of G, and a price vector p for G. We now describe a general iteration of Algorithm A. If (M, p) is a stable solution for G, then Algorithm A halts "successfully" and returns the price vector p. Otherwise, at least one of the three conditions in the definition of a stable solution (see the Assignment 4 handout) is violated. If Stability Condition 1 is violated, then Algorithm A halts "unsuccessfully". Otherwise, Algorithm A selects an arbitrary violation of one of Stability Conditions 2 and 3, and completes the current iteration according to the following two cases.

Case 1: The selected violation is of Stability Condition 2, and hence specifies an edge (u, v_0) in M and an edge (u, v_1) in E such that $w(u, v_0) - p_{v_0} < w(u, v_1) - p_{v_1}$. In this case, Algorithm A increments p_{v_1}.

Case 2: The selected violation is of Stability Condition 3, and hence specifies an edge (u, v_0) in E such that u is unmatched in M and $p_{v_0} < w(u, v_0)$. In this case, Algorithm A increments p_{v_0}.

Algorithm A then proceeds to the next iteration.

1.2 Analysis of Algorithm A

For any configuration $G = (U, V, E)$, any MCM M of G, and any price vector p, we define the following predicates.

- The predicate $P_0(G, p)$ holds if $p \leq q$ for all stable price vectors q for G. (Note: The inequality $p \leq q$ means that $p_v \leq q_v$ for all v in V.)
- The predicate $P_1(G, M, p)$ holds if for every item v in V, there exists a bid u in U such that u is unmatched in M and there is a directed path from u to v in $\mathit{digraph}(G, M, p)$.
- The predicate $P_2(G, M, p)$ holds if $p_v \leq w(u, v)$ for all edges (u, v) in M. (In other words, $P_2(G, M, p)$ holds if (M, p) satisfies Stability Condition 1.)
- The predicate $P(G, M, p)$ holds if $P_0(G, p)$, $P_1(G, M, p)$, and $P_2(G, M, p)$ hold.

Exercise 1. Let $G = (U, V, E)$ be a configuration and let p be the price vector for G such that p_v is equal to the start price of v for all items v in V. Assume that the MCM M of G associated with this execution of Algorithm A is an MWMCM of G. Prove that $P(G, M, p)$ holds. Hint: Make use of Exercise 1 from Assignment 4.

Exercise 2. Consider an iteration of Algorithm A on a configuration $G = (U, V, E)$ that increments the price of an item due to a violation of Stability Condition 2. Assume that the MCM M of G associated with this execution of Algorithm A is an MWMCM of G. Let p (resp., p') denote the price vector of G maintained by Algorithm A just before (resp., after) this iteration. Prove that if $P_0(G, p)$ holds, then so does $P_0(G, p')$. Hint: Make use of Lemma 1 of Assignment 4.

Exercise 3. Consider an iteration of Algorithm A on a configuration $G = (U, V, E)$ that increments the price of an item due to a violation of Stability Condition 2. Assume that the MCM M of G associated with this execution of Algorithm A is an MWMCM of G. Let p (resp., p') denote the price vector of G maintained by Algorithm A just before (resp., after) this iteration. Prove that if $P_1(G, M, p)$ holds, then so does $P_1(G, M, p')$. Hint: Make use of Exercise 4 from Assignment 4.

Exercise 4. Consider an iteration of Algorithm A on a configuration $G = (U, V, E)$ that increments the price of an item due to a violation of Stability Condition 2. Assume that the MCM M of G associated with this execution of Algorithm A is an MWMCM of G. Let p (resp., p') denote the price vector of G maintained by Algorithm A just before (resp., after) this iteration. Prove that if $P_1(G, M, p)$ and $P_2(G, M, p)$ hold, then $P_2(G, M, p')$ holds. Hint: Make use of Lemma 2 from Assignment 4.

Combining the results of Exercises 2 through 4, we immediately obtain the following lemma.

Lemma 1. *Consider an iteration of Algorithm A on a configuration G that increments the price of an item due to a violation of Stability Condition 2. Assume that the MCM M of G associated with this execution of Algorithm A is an MWMCM of G. Let p (resp., p') denote the price vector of G maintained by Algorithm A just before (resp., after) this iteration. If $P(G, M, p)$ holds, then so does $P(G, M, p')$.*

We can prove the following analogous lemma for the case of a violation of Stability Condition 3. In this case, the proof details (which are omitted here) are substantially simpler.

Lemma 2. *Consider an iteration of Algorithm A on a configuration G that increments the price of an item due to a violation of Stability Condition 3. Assume that the MCM M of G associated with this execution of Algorithm A is an MWMCM of G. Let p (resp., p') denote the price vector of G maintained by Algorithm A just before (resp., after) this iteration. If $P(G, M, p)$ holds, then so does $P(G, M, p')$.*

Combining Lemmas 1 and 2, we immediately obtain the following lemma.

Lemma 3. *Consider an iteration of Algorithm A on a configuration G that increments the price of an item. Assume that the MCM M of G associated with this execution of Algorithm A is an MWMCM of G. Let p (resp., p') denote the price vector of G maintained by Algorithm A just before (resp., after) this iteration. If $P(G, M, p)$ holds, then so does $P(G, M, p')$.*

Exercise 5. Consider an execution of Algorithm A on a configuration $G = (U, V, E)$. Assume that the associated MCM M of G is an MWMCM of G, and that the initial price vector p for G is such that $P(G, M, p)$ holds. Prove that Algorithm A is guaranteed to halt successfully within a finite number of iterations. Hint: Make use of Lemma 3.

Exercise 6. Prove that there exists a minimum stable price vector for G, i.e., a stable price vector p for G such that $p \leq q$ for all stable price vectors q for G. Hint: Make use of Exercises 1 and 5.

The preceding exercises show us that Algorithm A can be used to compute the minimum stable vector of an arbitrary configuration. Algorithm A can be slow because it only increases the item prices by one unit at a time. In an actual implementation, the following simple technique can be used to speed up Algorithm A: When Algorithm A would increment the price of an item v due to the violation of a stability condition, we instead increase the price of v by the minimum amount necessary to eliminate this violation.

The following lemma is useful for efficiently updating the minimum stable price vector of a configuration when a new bid is added.

Lemma 4. *Let G be a configuration, and let G' be the configuration obtained from G by adding a new bid. Let M denote an MWMCM of G', and let p denote the minimum stable price vector for G. Then $P(G', M, p)$ holds.*

You should seek to exploit Lemma 4 in the program that you are asked to write in Section 1.3.

1.3 Programming Task

We now describe the programming task associated with this assignment. At a high level, the task is similar to that of Assignments 1 through 3. There are three differences.

The first difference is that each item now has a start price in addition to a quality and a reserve price. The input format for a line specifying an item is the same as before, except there is an extra integer at the end of the line to specify the start price. The start price is guaranteed to be at most the reserve price. In what follows, let n denote the number of items in the input instance.

The second difference is that when a “3” command is encountered, the minimum stable price vector for the current configuration is printed, rather than the weight of an MWMCM followed by an MWMCM. The minimum stable price vector is printed as a sequence of n blank-separated integers on a single line.

The third and final difference is that we will provide an MWMCM as part of the input whenever a new bid is added. In more detail, when the input line contains a “1” command (i.e., add a single-item bid) or a “2” command (i.e., add a linear bid), the format of the input line is the same as in Assignments 1 through 3 except that a sequence of n extra integers is appended to the line. This sequence of n integers encodes an MWMCM of the new configuration. (The scheme used for encoding the MWMCM is the same as that used to encode an output MWMCM in the programs of Assignments 1 through 3.) The motivation for including an MWMCM in the input is to help students who did not get the Assignment 3 program working. (Even if you did get the Assignment 3 program working, it is fine if you prefer to read in the provided MWMCM instead of using your own code to compute an MWMCM.)

A point of clarification. Recall that in Assignments 1 through 3, the output encoding of an MWMCM used a bid index of -1 to indicate that an item is allocated to the dummy bid corresponding to the reserve price of the item. Now that we have two dummy bids for each item, one might think that we need two special codes, say -1 to indicate the dummy bid corresponding to the reserve price, and -2 to indicate the dummy bid corresponding to the start price. However, since the start price is at most the reserve price, we can assume without loss of generality that an item v is never won by the dummy bid corresponding to the start price of v. Hence a single special code of -1 is sufficient.

2 Textbook Exercises

1. In class we discussed the problem of determining whether a given graph is 3-colorable, and we referred to this decision problem as 3-COL. We also presented a polynomial-time transformation from 3-SAT to 3-COL as part of a proof that 3-COL is NP-complete (see also Section 8.7 of the textbook). The other part of the proof involved arguing that 3-COL belongs to NP, which is straightforward. Given a graph G, we can also ask whether G is 4-colorable; let us call this decision problem 4-COL. Give a polynomial-time transformation from 3-SAT to 4-COL, justifying your answer. (Since it is easy to argue that 4-COL belongs to NP, we conclude that 4-COL is NP-complete.) Hint: Make suitable modifications to the polynomial-time transformation from 3-SAT to 3-COL that we presented in class.

2. Problem 8.13, page 511. Hint: Use a reduction from the *Independent Set* problem (see page 454 for a definition), which is known to be NP-complete.

3. Problem 8.17, page 513. Hint: Use a reduction from the *Subset Sum* problem (see page 499 for a definition), which is known to be NP-complete.

4. Problem 9.1, page 550.