CS 357: Algorithmic Game Theory **Spring 2025**

Assignment 3 (due 04/11/2025)

Instructor: Shikha Singh

LaTeX Template (Source): https://www.overleaf.com/read/wprzjfbgxxft#6f2d2b

Instructions. This is a **partner assignment**. You must use the LaTeX solution template provided to write your answers and submit a joint PDF with your partner on Gradescope. Points will be awarded for *clarity, correctness and completeness* of the answers. Assignments are different from homeworks and **formal proofs** are expected for each question. This assignment is due Friday (04/11) at noon on Gradescope.

Stable Matching with Partial Preferences

Consider a stable matching market with n candidates and n jobs. In class, we assumed that each candidate provides a complete ranking of the n jobs and each job provides a complete ranking of n candidates. In reality, this is often infeasible or impractical. For example, the National Residency Matching program (NRMP) asks doctors to submit a ranked list of their top 20 hospitals.[1]

Consider the stable matching problem where agents can submit incomplete preference lists. To define stability in this setting, it is assumed that a candidate c prefers to be unmatched than be matched to a job j they did not include in their list, and similarly a job j prefers to be unmatched than be matched to a candidate c they did not include in their list. When incomplete lists are allowed, then some agents may now be left unmatched in a stable matching.

Fortunately, the Gale Shapley deferred acceptance algorithm naturally extends to partial preferences. If each candidate has d jobs on their preference list where $d < n$, then in a candidate-proposing deferred acceptance algorithm, if a candidate is rejected by all d jobs on their list then they remain unmatched. The extended procedure is summarized in Algorithm 1. Notice that the CPDA algorithm terminates when all candidates are either matched or have exhausted their preference list. Moreover, all candidates in $\mathcal{R}$ at the end of Algorithm 1 remain unmatched.

We proved in class that the CPDA algorithm outputs the best-possible matching for each candidate. In particular, let J_c denote the set of all stable partners of a candidate c on an instance I of stable matching. Recall that best(c) denotes the job that is most preferred by c among J_c. We proved in class that the CPDA algorithm matches each candidate c to best(c). This property continues to hold in the extended procedure when there are incomplete lists.

[1] Candidates submit 20 ranks for free; each additional rank costs $30 up to a maximum of 300 [2].

Algorithm 1: Candidate Proposing Deferred-Acceptance (CPDA) Algorithm

Fix an ordering over $\mathcal{C}$; Initialize $\mathcal{U} \leftarrow \mathcal{C}$ and $\mathcal{R} \leftarrow \emptyset$ and $\mu(j) \leftarrow \emptyset$ for all $j \in \mathcal{J}$
while $\mathcal{U} \setminus \mathcal{R} \neq \emptyset$ **do**
  Pick a candidate $c \in \mathcal{U}$ with the lowest index
  if $c \notin \mathcal{R}$ **then**
    Let j be the most preferred job on c's list that c has not yet proposed to
    if j is the last job on c's list **then**
      add c to $\mathcal{R}$
    end
    c **proposes** to j; Let $c' = \mu(j)$
    if j prefers c to c' **then**
      j **rejects** c'
      if $c' \neq \emptyset$ **then**
        add c' to $\mathcal{U}$
      end
      j **accepts** c;
      Set $\mu(j) = c$ and remove c from $\mathcal{U}$
    end
  end
end

Lone Wolf Theorem. When incomplete preferences are allowed, stable matchings admit a somewhat surprising property which is referred to as the *Lone Wolf Theorem* in the literature.

This property says that the set of matched agents are the same in *any* stable matching. That is, no matter what algorithm to used to generate the matching, the people who are unmatched in one are unmatched in all and those who are matched in one are matched in all of them. The question below asks you to formally prove this property.

Problem 1. Let I be an instance of the stable matching problem with incomplete preference list and μ be some stable matching of I. Prove that if a candidate c is unmatched in μ, then c must be unmatched in every stable matching of I. (*Hint. Consider the stable matching μ^* output by the candidate-proposing DA algorithm. Compare the matched pairs in μ and μ^*.*)

Strategyproofness of CDPA. The lone-wolf theorem is useful in proving that the CPDA algorithm is dominant-strategyproof for the candidates.

Problem 2. Prove that the candidate-proposing deferred acceptance algorithm is dominant-strategyproof for the candidates. That is, no candidate can achieve a better match by misreporting their preferences.

An outline of the proof is provided to guide your reasoning. We prove the theorem by contradiction.

Suppose that the CPDA algorithm is not dominant-strataegyproof for the candidates. Then, there exists at least one candidate c that can benefit from misreporting, keeping the preferences of all others fixed. More formally, consider an instance I and a candidate c with a true preference P_c and another instance $I^{(1)}$ which is identical to I except c's preference list

$P'_c \neq P_c$. Let μ and μ' be the matching output by CPDA on instances I and $I^{(1)}$ respectively. Suppose c prefers their match $j' = \mu'(c)$ to their match $j = \mu(c)$.

(a) Consider a new instance $I^{(2)}$ which is identical to I except c's preference list now only consists of a single job j'. Show that the matching μ' output when CPDA is run on $I^{(1)}$ continues to be a stable matching with respect to the instance $I^{(2)}$.

(b) Consider a new instance $I^{(3)}$ which is identical to I except c's truthful preference list P_c is truncated at job j'. That is, all jobs below j' are removed from c's list. Show that the matching μ'' output when CPDA algorithm on $I^{(3)}$ must leave c unmatched.

(c) Show that the matching μ'' output when CPDA algorithm on $I^{(3)}$ is also stable with respect to the instance $I^{(2)}$.

(d) Finally, use the lone-wolf theorem on one of these instances to arrive at a contradiction.

Stategyproof and Stability.

Problem 3. In this question, we will show that no algorithm for two-sided matchings can be both stable and strategyproof (for both sides of the market). To simplify, we allow incomplete preferences as in the previous questions, that is, candidates (jobs) can declare other jobs (candidates) as unacceptable by leaving them off their preference list.

Using this model, prove that for a stable matching instance with two candidates and two jobs, that does not exist an algorithm that is both stable and strategyproof for both students and jobs.

(*Hint.* Suppose such an algorithm exists and consider an instance where in a truthful ordering s_1 prefers h_1 over h_2, s_2 prefers h_2 over h_1, h_1 prefers s_2 over s_1 and h_2 prefers s_1 over s_2. Show that any algorithm that outputs a stable matching on such a instance fails to be dominant-strategyproof. The only assumption you can make is that the algorithm always output a stable matching.)

Popular Matchings. When candidates and jobs have incomplete preferences, requiring stability causes some candidates/jobs will be unmatched. Moreover, such unmatched agents are unlucky as they are unmatched in all stable matchings. Let the size of a matching be the number of matched candidates. It is known that requiring stability can lead to a size of a matching that is much smaller than the largest cardinality bipartite matching. With the goal of matching more agents, researchers [1] have looked at relaxing the requirement with the goal of finding larger matchings.

Consider a bipartite graph $G = (V, E)$ where $V = C \cup J$ where C is the set of n candidates and J is a set of n jobs. An edge $e = (c, j) \in E$ if candidate c appears in j's preference list and j appears in c's preference list. Moreover, each vertex has a strict ranking over its neighbors.

A matching $N \subseteq E$ such that no two vertices in N share an edge. We say a candidate c prefers a matching N_1 to a matching N_2 if (a) c is matched in N_1 and unmatched in N_2, or (c) c is matched in both but prefers their match in N_1 to their match in N_2. A matching N_1 is **more popular** than N_2 if the number of candidates that prefer N_1 to N_2 exceeds the

number of candidates that prefer N_2 to N_1. A matching M is **popular** if and only if there is no matching M' that is more popular than M. In the next problem, show that popularity is a relaxation of stability, that is, each stable matching is popular but there exist popular matchings that are not stable.

Problem 4. (a) Prove that every stable matching is popular.

(b) Give an example of a matching that is popular but is not stable.

References

[1] David J Abraham, Robert W Irving, Telikepalli Kavitha, and Kurt Mehlhorn. Popular matchings. *SIAM Journal on Computing*, 37(4):1030–1045, 2007.

[2] NRMP. How many programs can I rank? — nrmp.org. https://www.nrmp.org/help/item/how-many-programs-can-i-rank/#, 2024. [Accessed 09-02-2025].