Electronic Journal of Graph Theory and Applications **8** (1) (2020), 29–57

Electronic Journal of
Graph Theory and Applications

An efficient implementation of the Gale and Shapley "propose-and-reject" algorithm

Nasia Zacharia[a], Evi Papaioannou[b], Christos Kaklamanis[b]

[a]*University of Patras, GR26504, Rion, Greece*
[b]*University of Patras & CTI "Diophantus", Patras University Campus, Building B, GR26504, Rion, Greece*

zacharia@ceid.upatras.gr, papaioan@ceid.upatras.gr, kakl@ceid.upatras.gr

Abstract

We consider a version of the Hospitals/Residents problem which was first defined in 1962 by Gale and Shapley [9] under the name "College Admissions Problem". In particular, we consider the Firms/Candidates problem, where each Firm wishes to hire at least one Candidate and each Candidate can be finally assigned to a single Firm. We present an efficient implementation of the Gale and Shapley "propose-and-reject" algorithm when applied to the case of the Firms/Candidates problem.

Keywords: stable matching, Gale and Shapley algorithm, the Firms/Candidates problem, efficient implementation, "propose-and-reject" algorithm
Mathematics Subject Classification : 05C85, 05C90
DOI: 10.5614/ejgta.2020.8.1.4

1. Introduction

In terms of graph theory, marriage is expressed as a matching problem. In the simplest form of a matching problem, we are given a graph where the edges indicate compatibility, i.e., vertices connected by an edge can be paired or "married". Given a graph $G = (V, E)$, a matching is a subgraph of G where every node has degree 1, implying that everybody can be married just to one person. When every node is involved in a matching, then this matching is called perfect matching. The need to efficiently compute matchings arises often in a wide range of popular algorithmic problems (see, e.g., [4, 16, 19]) which include scheduling, bipartite matching, independent set, competitive

Received: 16 December 2018, Revised: 10 May 2019, Accepted: 27 June 2019.

facility location and have also been been recently addressed in more theoretical contexts (see, e.g., [2, 5]).

We study a special matching problem known as the Stable Marriage Problem (SMP). In its basic version, we are given N boys and N girls, boys can only be paired to girls and vice versa, each boy (resp. girl) has his own ranked preference list of all the girls (resp. girls), the lists are complete and there are no ties and preferences are not necessarily symmetric and cannot change over time. The goal is to find a perfect matching that is stable. A matching is stable when it does not contain pairs whose members prefer each other to their mates in the matching.

In 1962, David Gale and Lloyd Shapley, partially supported by a grant from the Office of Naval Research, published a paper entitled "College Admissions and the Stability of Marriage" [9, 8]. In that paper, Gale and Shapley highlighted the similarity between college admissions and the marriage market. In the former case, students and universities are trying to pair up to their mutual satisfaction, in the latter case a fixed number of men and women are trying to find a match. The authors proposed a new algorithm for matching an equal number of men and women that achieved as many satisfactory and stable matches between partners as possible from the huge number of potential pairings, a concept that also applies to matching students and universities. According to the Gale and Shapley algorithm, men and women rank their preferred partners. Each man proposes to his highest-ranked woman. Each woman rejects all the proposals she gets except for the highest-ranked among them. However, she does not accept the proposal, in case a man she prefers even more proposes next time. The algorithm is re-executed until all women have a satisfactory proposal. In their 1962 article, Gale and Shapley noted that their algorithm was not particularly complicated, illustrating a larger point about their discipline: "any argument that is carried out with sufficient precision is mathematical".

Since then, the Stable Marriage Problem has been extensively studied in the literature, mainly due to its wide application in several areas and its remarkable variations (see for example [4, 7, 15, 16, 17, 23]). The algorithm suggested by Gale and Shapley brought about significant changes especially in the fields of Economics, Health and Computer Science. Indicative applications include dating problems, where the objective is to match compatible people together, assignment problems, e.g., matching students to schools (e.g., [31]) or interns to hospitals (e.g., [28]), resource allocation problems, e.g., load balancing traffic on the Internet (e.g., [20]), cryptography and data security (e.g., [6]). However, maybe the most fascinating relevant application is about finding compatible kidney donors for patients needing a transplant (e.g., [26, 33]).

Due to the major technology advances having stemmed from their contribution, Lloyd Shapley and Alvin Roth (D. Gale died in 2008) were awarded in 2012 the Nobel Prize in Economic Sciences, as a recognition of their work which has laid the theoretical foundations for analyzing resource allocation, has enhanced understanding of how markets work and has highlighted the usefulness of theory for applied work and policy making [27].

2. Technical description

The Stable Marriage Problem (SMP) was first introduced by Gale and Shapley in 1962 in their seminal paper [9]. Since then, the SMP has attracted the attention of the research community in several areas because of its inherent mathematical structure and its rich real-world applications.

In short, the SMP can be informally stated as follows. Given n men and n women each with their own preference list, provide an algorithm that computes a stable set of matchings, i.e., matchings not containing blocking pairs. A man and a woman form a blocking pair if they are not matched but prefer each other to their current matches, implying that these two persons are likely to cheat on their partners.

Gale and Shapley proposed a polynomial-time deferred acceptance algorithm for the SMP, the so-called Gale and Shapley algorithm, which always finds a stable matching, thus providing a constructive proof of the fact that any instance of the SMP admits at least one stable matching. In particular, their algorithm is a "propose-and-reject" procedure and works as follows. While there are men that are not matched, have them propose to the woman that is next on their preference list, and have each woman select their favorite proposal based on her preference list. The algorithm terminates once every woman has received a proposal.

The Gale and Shapley algorithm always returns a stable, male-optimal and female-pessimal matching. Male-optimal implies that men, going down their preference list, are eventually engaged to their most preferred available women. Female-pessimal implies that women, going up their preference list, are eventually engaged to their most preferred available men. In particular, the (men-oriented version of the) Gale and Shapley algorithm incorporates a sequence of proposals from men to women and computes a stable matching which is uniquely favorable to men, i.e., male-optimal: every man is matched to his best stable female partner. However, in the computed matching each woman is assigned to her least-worst male partner; that is, the algorithm returns a female-pessimal result. Analogously, if women propose to men, then the algorithm computes a stable matching which is uniquely favorable to women, i.e., female-optimal, male-pessimal.

Gale and Shapley also studied the "College Admissions" problem, a many-to-one stable matching problem stated in terms of colleges and students (instead of men and women) as follows [31]. Each college $c \in C$ wishes to be matched to $q(c) \geq 1$ students. Each student is interested in being matched to only 1 college. That is, each student has a preference over colleges, each college has a preference over individual students, and a matching is a function that assigns each student to at most one college, and each college c to at most $q(c)$ students. They observed that essentially the same deferred acceptance algorithm - with college c proposing at each point to its $q(c)$ most preferred students who hadn't yet rejected it in the college-proposing version, or rejecting all but the $q(c)$ most preferred applications it had received at any point of the student-proposing version - would produce a stable matching defined as before. That is, the outcome produced by the algorithm would not admit any student-college blocking pairs defined precisely as for the SMP. Their proof of this result was constructive, i.e., they suggested an algorithm for computing a stable matching.

In [9], the Hospitals/Residents problem (HRP) was also stated. The reference to Hospitals and Residents was due to the application of the "College Admissions problem" to large-scale matching schemes administering the annual match of graduating medical students to hospital appointments in the US (National Resident Matching Program [29, 28]), Canada (Canadian Resident Matching Service [3]) and Scotland (Scottish Pre-registration house officer Allocations- SPA [11]).

An instance of the Hospitals/Residents problem involves a set R of residents and a set H of hospitals. Each resident $r \in R$ is seeking a post at one hospital, and each hospital $h \in H$ offers $q(h) \geq 1$ posts. Each resident in R ranks a subset of H in strict order, and each hospital $h \in H$ ranks its applicants in strict order. An agent $p \in R \cup H$ finds an agent $q \in R \cup H$ acceptable if

q appears on p' s preference list; p finds q unacceptable otherwise. A matching M is a subset of $R \times H$, where $(r, h) \in M$ implies that (i) r, h find each other acceptable, (ii) r is assigned to at most one hospital in M, and (iii) at most $q(h)$ residents are assigned to $h \in M$.

The classical Stable Marriage problem [7, 32, 17] is a restriction of the Hospitals/Residents problem in which each hospital has exactly one post, the number of hospitals equals the number of residents, and all preference lists are complete. For a given instance I of the Hospitals/Residents problem, the Gale and Shapley algorithm for the Stable Marriage problem [9] may be extended in order to find a stable matching for I (such a matching in I always exists) in $O(mn)$ time, where $n = |R|$ and $m = |H|$ [7]. Again, if the residents propose to the hospitals (the resident-oriented algorithm), then a stable matching M is computed which is uniquely favorable to the residents: every resident $r \in M$ is assigned to her best stable partner, and every resident unassigned in M is unassigned in any stable matching. Similarly, if the hospitals propose to the residents (the hospital-oriented algorithm), then a stable matching M is computed which is uniquely favorable to the hospitals: every hospital $h \in H$ is assigned either its $q(h)$ best stable partners, or a set of fewer than $q(h)$ residents; in the latter case, no other resident is assigned to h in any stable matching.

Although an instance of the Hospitals/Residents problem may admit more than one stable matchings, every stable matching has particular properties summarized in the "Rural Hospitals Theorem" [29, 10, 30]. The "Rural Hospitals Theorem" states that for a given instance of the Hospitals/Residents problem, (i) each hospital is assigned the same number of residents in all stable matchings, (ii) exactly the same residents are unassigned in all stable matchings, and (iii) any hospital that is under-subscribed in one stable matching is matched with precisely the same set of residents in all stable matchings. There exists no algorithm that can produce a stable matching changing the allocation of residents to unpopular hospitals. In the experience of the NRMP, these tend to be rural hospitals.

Several other variations of the original problem have been studied [22, 25]. These variations mainly arise in practical contexts. For example, in the original version of the Stable Marriage problem preference lists must be complete and strictly ordered. However, these assumptions do not really hold in real-world applications, especially in a large-scale matching system. Having said that, two natural relaxations of the original problem result in two variants called "Incomplete Preference Lists" and "Incomplete Preference Lists with Ties" [14, 15]; the former allows incompleteness while the latter allows both incompleteness and ties in preference lists and under particular assumptions can be computationally hard [21, 24]. Other interesting variants [15] include (i) the Stable Roommates Problem [7] which is a a non-bipartite extension of the original Stable Marriage problem with applications to the formation of player pairs in chess tournaments [18] and pairwise kidney exchange between incompatible patient-donor pairs [13, 33], (ii) the Man-Exchange Stable Marriage where stability requires also that no two men prefer to exchange their partners making the problem computationally hard [12], (iii) Many-to-Many Stable Marriage, a many-to-many extension of the original Stable Marriage problem, (iv) the Student Project Allocation Problem where students are assigned to projects based on his/her preferences over projects (see for example [1]), (v) the 3-Dimensional Stable Matching proposed by Knuth [17] which involves 3 set of agents, (vi) the One-Sided Preference Lists where only one party has preference lists over the other.

In this work, we consider the Firms/Candidates problem (FCP, in short), a variation of the Hospitals/Residents problem. In particular, given two non-empty sets of Firms and Candidates together

with their complete preference lists with no ties, we use the Gale and Shapley "propose-and-reject" algorithm for producing stable matchings. Our contribution is a "search-without-searching" implementation of the Gale and Shapley algorithm for the Firms/Candidates problem.

The rest of the paper is structured as follows: in Section 3 we overview the Gale and Shapley algorithm and its application to the Firms/Candidates problem. In Section 4, we present in detail our implementation approaches and obtained experimental results. We conclude in Section 5.

3. Problem statements and algorithms

3.1. SMP

3.1.1. SMP statement

An instance I of the classical Stable Marriage problem (SMP) involves n men and n women, each of whom ranks all the members of the opposite sex in strict order of preference. A matching M in I is a one-to-one correspondence between the men and women. We say that a (man, woman) pair (m, w) blocks M, or is a blocking pair with respect to M, if m prefers w to $p_M(m)$, and w prefers m to $p_M(w)$, where $p_M(q)$ denotes the partner of q in M. A matching that admits no blocking pair is said to be stable. It is known that every instance of SMP admits at least one stable matching, and that such a matching can be found in $O(n^2)$ time using the Gale and Shapley algorithm [9].

This version of the Stable Marriage problem (SMP) can be modelled by means of a bipartite graph $G = (U, V, E)$, where U denotes the set of men and V denotes the set of women. Both sets are equally-sized, i.e., $|U| = |V| = n$. Furthermore, each man $m \in U$ (resp., each woman $w \in V$) maintains a strictly ordered preference list, R_m (resp., R_w), containing all women $w \in V$ (resp., all men in $m \in U$). Each such list is complete (i.e., it contains n elements) and shows men's (resp., women's) preferences for all women (resp., men) in decreasing order. For example, let $n = 4$, $U = \{A, B, C, D\}, V = \{a, b, c, d\}$ and $R_A = [d, a, b, c]$. List R_A indicates that A's first preference is woman d, women a and b follow and woman c is A's lowest preference. Then, a solution to the Stable Marriage problem consists in finding a perfect matching M in G, i.e., a subset of n edges of E where each vertex of $U \cup V$ appears exactly once.

3.1.2. The Gale and Shapley algorithm for SMP

Gale and Shapley actually devised a "propose-and-reject" algorithm which works as follows. The "propose-and-reject" procedure takes place over several runs or days. Each day is broken up into three parts: the morning, the afternoon, and the evening. In the morning, each man proposes to his favorite woman who is still on his list. Initially, every woman is on his list. In the afternoon, each women who received at least one proposal picks her favorite man and gets engaged to him. She rejects all other men who proposed to her but have lower priority in her list. That night, any man who was rejected by a woman crosses that woman off his list. This procedure is repeated in every run. When every woman is engaged to at most one man, the algorithm stops and returns the result.

The Gale and Shapley algorithm for SMP can be formally described as follows.

Input.

- A set of n men: $M = \{m_0, m_1, \ldots, m_n\}$
- A set of n women: $W = \{w_0, w_1, \ldots, w_n\}$
- A preference list for each man and woman: for example $R_{m_i} = (w_{10}, w_3, \ldots, w_7)$
- define $R_{w_i}(m_j)$ to be the index of m_j in w_i's preference list
- define $R_{m_i}(w_j)$ to be the index of w_j in m_i's preference list

Note that a smaller index indicates a stronger preference. Specifically, $R_w(m) < R_w(m_0)$ means that w prefers m over m_0.

Output.

- A stable matching of men and women: $S \subseteq M \times W$
- S is a stable matching if all of the following are true:
  - $|S| = n$
  - $\forall m \in M \exists w \in W : (m, w) \in S$
  - $\forall w \in W \exists m \in M : (m, w) \in S$
  - $\neg\exists_{(m,w)\notin S}((m, w') \in S \wedge (m', w) \in S \wedge (R_m(w) < R_m(w')) \wedge (R_w(m) < R_w(m')))$

1. Initially all men and women are free
2. While there exists a free man m, he proposes to her most preferred woman w whom he has not proposed to yet as follows:
   (a) m proposes to w
   (b) If w is free, m and w get engaged and (m, w) is added to S
   (c) If w is currently engaged to another man, m_0, and she prefers m_0 over m, $R_w(m) > R_w(m_0)$: m remains free
   (d) If w is currently engaged to another man, m_0, and she prefers m over m_0, $R_w(m) < R_w(m_0)$: w breaks off the engagement with m_0 and gets engaged to m. (m_0, w) is removed from S and (w, m) is added to S. m_0 becomes free.
3. return S as a stable matching.

Below, we summarize basic statements from regarding correctness and running time analysis of the Gale and Shapley algorithm for the men-oriented version of the SMP. Proofs are omitted but and be found in [9, 7, 11, 16, 23].

Observation 3.1. *Once a woman gets engaged, she never becomes free again and gets engaged to more preferred men.*

Observation 3.2. *Men stay engaged to the same woman or continue getting engaged to worse women.*

Lemma 3.1. *The output of the Gale and Shapley algorithm is always a perfect matching, i.e., a matching of size* n*.*

Lemma 3.2. *The output of the Gale and Shapley algorithm has no instabilities.*

Theorem 3.1. *The Gale and Shapley algorithm always produces a stable matching.*

Lemma 3.3. *There are at most* n^2 *proposals during a run of the Gale and Shapley algorithm.*

Lemma 3.4. *Each proposal can be performed in* $O(1)$ *time.*

3.2. FCP

In their seminal work [9], David Gale and Lloyd Shapley considered a model with two sets of agents, for example workers and firms, that must be paired with each other. If a particular worker is hired by employer A, but this worker would have preferred employer B, who would also have liked to hire this worker (but did not), then there are unexploited gains from trade. If employer B had hired this worker, both of them would have been better off. Gale and Shapley defined a pairing to be stable if no such unexploited gains from trade exist. In an ideal market, where workers and firms have unrestricted time and ability to make deals, the outcome would always be stable. Of course, real-world markets may differ from this ideal in important ways. But Gale and Shapley suggested a "deferred-acceptance" procedure which is easy to understand and always leads to a stable outcome. The procedure specifies how agents on one side of the market (e.g., the employers) make offers to those on the other side, who accept or reject these offers according to certain rules [27].

More precisely, "agents on one side of the market, say the firms, make offers to agents on the other side, the workers. Each worker reviews the proposals she receives, holds on to the one she prefers (assuming it is acceptable), and rejects the rest. A crucial aspect of this algorithm is that desirable offers are not immediately accepted, but simply held on to: deferred acceptance. Any firm whose offer is rejected can make a new offer to a different worker. The procedure continues until no firm wishes to make another offer, at which time the workers finally accept the proposals they hold. In this process, each firm starts by making its first offer to its top-ranked worker, i.e., the worker it would most like to have as an employee. If the offer is rejected, the firm makes an offer to the worker it ranks as number two, etc. Thus, during the operation of the algorithm, the expectations of a firm are lowered as it makes offers to workers further and further down its preference ordering. (Of course, no offers are made to unacceptable firms.) Conversely, since workers always hold on to the most desirable offer they have received, and as offers cannot be withdrawn, each worker's satisfaction is monotonically increasing during the operation of the algorithm. When the decreased expectations of firms have become consistent with the workers' increased aspirations, the algorithm stops" [27].

3.2.1. FCP statement

Based on the Hospitals/Residents problem, we state the Firms/Candidates problem as follows. An instance of the Firms/Candidates problem involves a set C of candidates and a set F of firms. Each candidate $c \in C$ is seeking a post at one firm, and each firm $f \in F$ offers $q(f) \geq 1$ posts. Preference lists of candidates and firms are complete. That is, each candidate $c \in C$ ranks F in

strict order, and each firm $f \in F$ ranks all applicants in strict order. A matching M is a subset of $C \times F$, where $(c, f) \in M$ implies that (i) r is assigned to at most one firm in M, and (ii) at most $q(h)$ candidates are assigned to $h \in M$. A matching M for an instance of the Firms/Candidates problem is stable if M admits no blocking pair. A blocking pair (c, f) for M is a candidate c and firm f such that (i) c either is unassigned or prefers f to her assigned firm in M, and (ii) f either is undersubscribed or prefers c to the worst candidate assigned to it in M. If (c, f) form a blocking pair with respect to a matching M, then (c, f) is said to block M. Also, if $(c, f) \in M$ for some stable matching M, then we say that (c, f) is a stable pair, and c is a stable partner of f (and vice versa).

3.2.2. The Gale and Shapley algorithm for FCP

We consider the firm-oriented version of the FCP. We use the Gale and Shapley algorithm to obtain a stable matching M which is uniquely favorable to firms. Every firm $f \in F$ is assigned either its $q(f)$ best stable candidates, or a set of fewer than $q(f)$ candidates; in the latter case, no other candidate is assigned to h in any stable matching [7].

In particular, the algorithms proceeds in rounds as follows. In each round, (i) each firm f which still has k uncovered posts, proposes to the k candidates who are currently ranked higher on the preference list of the firm and have not rejected it yet, (ii) each candidate accepts the proposal (if any) of the firm which is ranked higher in her preference list and rejects all other proposals (if any). The procedure continues until either all posts are assigned or there are no unassigned candidates.

The Gale and Shapley algorithm for FCP can be formally described as follows (see also [16]).

INITIALIZE M to empty matching

WHILE some firm f is unmatched and has not proposed to every candidate

$c \leftarrow$ first candidate on the list of firm f to whom f has not yet proposed

IF c is unmatched

Add (f, c) to matching M

ELSE IF c prefers f to current firm f'

Replace (f', c) with (f, c) in matching M

ELSE

c rejects f

RETURN stable matching M

Proofs for correctness, stability and complexity follow from the corresponding proofs of the SMP version of the Gale and Shapley algorithm and can be found in [9, 7, 11, 16, 23].

An indicative execution of this algorithm for a given input instance of the FCP is presented in Fig. 1.

The outcome produced by the algorithm does not admit any firm-candidate blocking pairs defined precisely as for the SMP. In the particular example of Fig. 1, all candidates have been assigned to their most preferred firms, so there is no way for a blocking pair to exist.

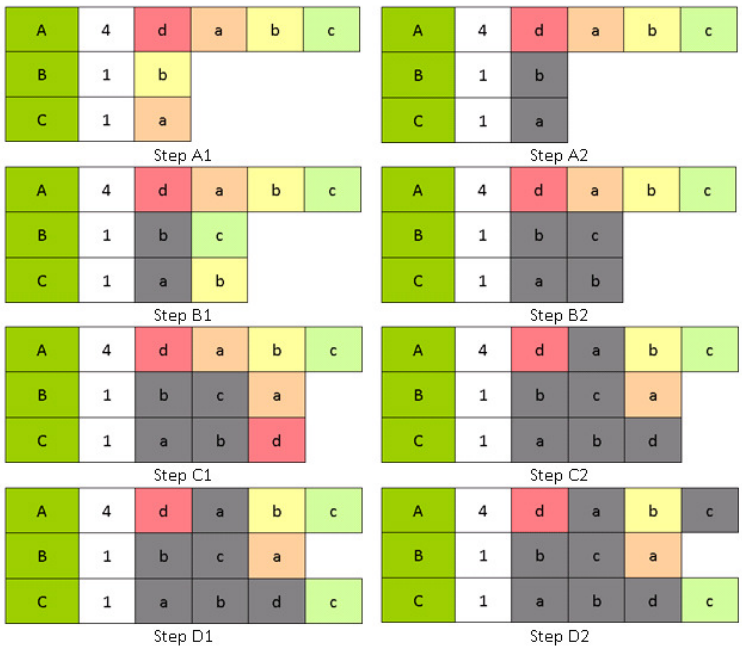

Figure 1. Indicative execution of the Gale and Shapley algorithm for the FCP. Each firm proposes to the k candidates who are currently ranked higher on the preference list of the firm and have not rejected it yet (steps A1, B1, C1, D1). Each candidate accepts the proposal, if any, of the firm which is ranked higher in her preference list and rejects all other proposals, if any (steps A2, B2, C2, D2).

4. Our "search-without-searching" implementation

Given two non-empty sets of Firms and Candidates together with their complete preference lists with no ties, we use the Gale and Shapley "propose-and-reject" algorithm for computing stable matchings. Our contribution is a "search-without-searching" implementation of the Gale and Shapley algorithm for the Firms/Candidates problem.

Let F be the set of firms and C the set of candidates. F and C are not necessarily equally-sized. Each candidate $c \in C$ can be assigned to at most one firm/post, i.e., a candidate cannot be assigned to multiple firms/posts. Related to F, a list J is maintained containing the $q(f) \geq 1$ available posts that each firm $f \in F$ wishes to assign. J is constructed according to the alphabetical order of firms in F. For example, let $F = \{A, B, C, D\}$ be the set of firms. A list $J = [4, 1, 1, 6]$ would indicate that firm A has 4 available posts, firms B and C have 1 available post each, and firm D has 6 available posts (Fig. 2).

A	4
B	1
C	1
D	6

Figure 2. List of available posts containing 4 firms in alphabetical order.

Note that the total number of available posts is not necessarily equal to the total number of candidates.

Each firm f maintains a preference list R_f ordered in descending order, i.e., from the most preferred to the least preferred candidate, which is complete, i.e., contains all candidates. See for example Fig. 3.

A	4	d	a	b	c	R_A
B	1	b	c	a	d	R_B
C	1	a	b	d	c	R_C

Figure 3. Ordered preference lists of firms.

Also, each candidate maintains a preference list ordered in descending order, i.e., from the most preferred to the least preferred firm, which is complete, i.e., contains all firms. See for example Fig. 4.

We consider the firm-proposing version of the problem, i.e., firms propose and candidates accept or reject. In particular, at each round, firm $f \in F$ proposes to its $q(f)$ most preferred

a	B	A	C
b	A	C	B
c	C	A	B
d	A	B	C

Figure 4. Ordered preference lists of of candidates.

candidates who have not yet rejected it. Respectively, in the candidate-proposing version of the problem, at each round, firm f would reject all but the $q(f)$ most preferred applications it would have received.

A valid assignment of candidates to firms must have the following properties.

- Each candidate is assigned to at most 1 firm/post.
- Each firm can be assigned a number of candidates which is at most equal to the number of its available posts.
- Each firm wishes to obtain candidates ranked as high as possible in its preference list. The rank of a candidate on the preference list of a firm is a non-zero, natural number showing the candidate's position in the preference list of a firm which is sorted in decreasing order. If a highly-ranked candidate is not assigned to a proposing firm, then this candidate prefers her current firm more than the proposing one.
- Each candidate $c \in C$ is assigned to the firm with the least-worst rank in her list. If a candidate is not assigned to a highly-ranked firm, then this firm prefers its currently assigned candidates more than c.

For the firm-oriented version of the algorithm, our implementation performs searching in time linear in the number of firms, resulting in $O(n^2)$ total execution time. In particular, we present three slightly different approaches for the implementation of the Gale and Shapley algorithm for the Firms/Candidates problem. The difference among these three approaches consists in the way we implement how each candidate decides which proposal (if any) to accept in each round.

4.1. Implementation 1: "search exhaustively"

A firm $f \in F$ with $q(f) = k$ available posts proposes to the first k candidates ranked higher in its preference list. Then, each of these candidates must decide whether to accept or reject the proposal. More precisely, given the decreasing order of her preference list, a candidate accepts the proposal of a firm if this firm is before her current assignment in her preference list, otherwise the candidate rejects the proposal.

For example, consider a candidate a whose preference list is $r_a = [B, C, D, A]$ and firms B and C whose preference lists sorted in decreasing order are $R_A = [a, d, b, c]$ and $R_C = [a, d, c, b]$, respectively. Initially, candidate a is unassigned. Firm B proposes to candidate a, who accepts the

proposal. Then, firm C proposes to a, who reads her preference list and accepts the proposal of the firm found first in her list. Since C appears first in a's preference list, a accepts the proposal of C (rejecting B). It took a 2 steps to decide that she accepts the proposal of firm C.

A "bad" instance would require reading $n-(j-1)$ positions of the candidate's preference list (of total length n) in each round $j > 1$ for deciding whether the new proposal should be accepted or rejected. For example, consider the following scenario. Candidate's a preference list is depicted in Fig. 5. Initially, firm A proposes to candidate a. In subsequent rounds, candidate a, receives the following proposals: (round 2) firm B, (round 3) firm C and (round 4) firm D.

Figure 5. Candidate a's preference list.

As sketched in Fig. 6, candidate a needs 3 steps to accept the proposal of firm B in round 2 (rejecting firm A), 2 steps to accept the proposal of C in round 3 (rejecting firm B), and 1 step to accept the proposal of D in round 4 (rejecting firm C).

Thus, following this "search exhaustively" approach, it would take each candidate a total of at most $\Sigma_{i=1}^{n-1} i$, i.e., $O(n^2)$, comparison steps to make her final decision, where $n = |F|$, given that candidate preference lists are complete. So, overall, this approach requires $O(n^3)$ steps for computing stable matchings.

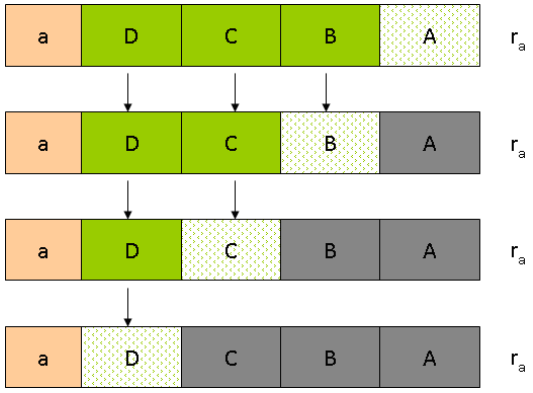

Figure 6. Indicative execution of Implementation 1: "search exhaustively".

4.2. Implementation 2: "search partially"

We, now, present a more efficient implementation of the procedure followed by candidates in order to decide whether to accept or reject a proposal. More precisely, reading the candidate's preference list (i.e., performing comparisons) is still required. However, now, the number of required "reads" is decreased via the exploitation of two facts: (a) the decreasing order of the candidate's preference list and (b) the "rank" of the candidate's current firm, i.e., the position of the currently assigned firm (if any) on the candidate's preference list. Again, accepting or rejecting a proposal depends on whether the proposing firm is (respectively) before or after the candidate's current firm in her preference list. The following example sketches the main idea behind this optimization.

Let $r_a = [E, D, C, B, A]$ be the preference list of candidate a, sorted in decreasing order. Assume that candidate a, who is initially not engaged to any firm, receives a proposal by firm A. Subsequently, she receives proposals by firms B and C. Candidates not engaged to some firm unconditionally accept the first proposal they receive by a firm. So, candidate a accepts the proposal of firm A (round 1). When she receives a proposal by firm B (round 2), candidate a has to "read" the first 4 positions of r_a (4 steps) for deciding whether to accept the proposal of firm B, with $rank_B = 4$ (rejecting firm A). When she receives a proposal by firm C (round 3), in order to decide whether to accept or reject the proposal candidate a must determine whether firm C is before or after her current firm B in her preference list. Given the decreasing order of a's preference list, the relative position of C and B can be determined by the a's answer to the following "Yes/No" question: "is C on the left of B in your preference list?" If Yes, accept the proposal of C (rejecting B), otherwise reject the proposal of C.

Candidate a knows that (i) her preference list is sorted in decreasing order, (ii) the size of her preference list is $|r_a| = 5$, and (iii) that for her current firm B, $rank_B = 4$. Since $rank_B \geq \frac{|r_a|}{2}$, candidate a concludes that her current firm lies in the 2nd half of her ordered preference list. So, it suffices that a checks the single position on the right of B (instead of the 3 positions on the left of B) in her preference list about whether it contains firm C: if Yes, a rejects the proposal of C (since it is a low-preference firm compared to C), otherwise a accepts the proposal of C (rejecting B). In this particular example (Fig. 7), firm C does not lay on the right of B, therefore candidate a accepts the proposal of C without calculating $rank_C$. That is, knowing that $rank_B = 4$, a was required to perform 1 comparison involving 1 list element (on the right of B) in position 5 of her preference list instead of 3 comparisons involving 3 list elements in positions 1, 2 and 3 - from the beginning of the list to the position of C - in order to decide whether to accept or reject the new proposal by firm C.

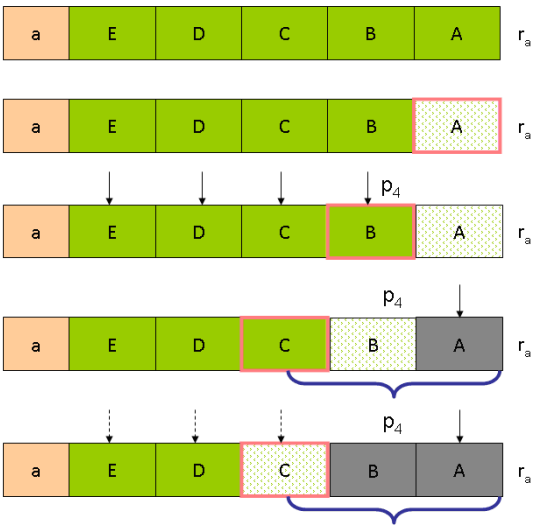

Figure 7. Indicative execution of Implementation 2: "search partially".

Overall, it can be easily observed that this implementation approach requires at most a single full "read" of the n elements of the candidate's preference list every second round. This suggests

an improvement compared to the previous approach which required a full "read" of the n elements of the candidate's preference list in each round.

More precisely, in even rounds (rounds 2, 4, 6, etc), acceptance or rejection of a new proposal requires reading a number, say t, of positions of the candidate's preference list, from its beginning up to the position of either her current or the proposing firm. However, during the following odd rounds (rounds 3, 5, 7, etc, respectively), the candidate already knows that the $rank$ of her current firm is t. Thus, for accepting or rejecting the new proposal, it suffices to "read" only the smallest part of her preference list either on the left (of length $t-1$) or on the right (of length $n-t$) of the position of her current firm. Despite the fact that this improvement does not have an important impact on the order of magnitude of the necessary comparisons, it requires half the number of "reads" comparing to the number of "reads" of the candidates' preference lists needed by Implementation 1.

The algorithm for implementing this approach proceeds as follows:

1. For each firm F with a least 1 available post which has not reached the end of its preference list
2. If F proposes to a candidate c not yet engaged then c accepts the proposal (having no knowledge about her preference for this particular firm), otherwise if F proposes to an already engaged candidate c
   (a) If c has no knowledge of the rank of her current firm $rank_{current}$ in her preference list, c searches her preference list from its beginning until one of the following is true:
      i. If c finds her current firm first, c "remembers" $rank_{current}$ and rejects the proposal of F
      ii. If c finds firm F first, c "remembers" $rank_F$ and accepts the proposal of F
   (b) If c has knowledge of the rank of her current firm $rank_{current}$ in her preference list,
      i. if $rank_{current} \geq \frac{n}{2}$ (where n equals the number of firms or, equivalently, the size of c's preference list) that is, if her current firm is a low-preference choice for c, c searches positions $[rank_{current}, n]$ starting from the position $rank_{current}$ to the end of the preference list
         A. If c finds F then she rejects the proposal of F (since she prefers F less than her current firm)
         B. If c does not find F then she accepts the proposal of F (since she prefers F to her current firm) without knowing $rank_F$
      ii. if $rank_{current} < \frac{n}{2}$ (where n equals the number of firms or, equivalently, the size of c's preference list) that is, if her current firm is a high-preference choice for c, c searches positions $[1, rank_{current}]$ starting from the beginning of the preference list to the position $rank_{current}$
         A. If c finds F then she accepts the proposal of F (since she prefers F to her current firm) and updates $rank_{current}$ (i.e., $rank_F \rightarrow rank_{current}$)
         B. If c does not find F then she rejects the proposal of F (since she prefers F less than her current firm)

This procedure continues until either all firms cover all posts or are rejected by all candidates on their preference lists.

A "bad" instance would require reading all n positions of a candidate's preference list during each pair of successive rounds in order for a candidate to decide whether to accept or reject new proposals.

For example (Fig. 8), let $r_a = [E, D, F, C, H, B, G, A]$ be the candidate's a preference list. Initially, a (not engaged) receives a proposal by firm A. Then, she receives proposals by firms G, B, H and C. Candidates always accept the first proposal they receive. So, a accepts the proposal of firm A (round 1). When she receives a proposal by firm G (round 2), a must read the first 7 positions of her preference list (7 steps) for deciding whether to accept the proposal of G with $rank_G = 7$ (rejecting A). When she receives a proposal by firm B (round 3), in order to decide whether to accept or reject the proposal a must determine whether B proceeds (i.e., acceptance of its proposal) or follows (i.e., rejection of its proposal) her current firm G in her ordered preference list. Therefore, in order to decide whether to accept or reject the proposal of B, a needs to determine the relative position of B with regard to the position of G in her preference list. Since her preference list is strictly ordered, the exact order of B and G results from the answer to the "Yes/No" question: "is B on the left of G in a's preference list?" If Yes, a must accept the proposal of B, otherwise a must reject the new proposal.

a knows that (i) her preference list is sorted in decreasing order, (ii) the size of her preference list is $|r_a| = 8$, and (iii) that $rank_G = 7$. Since $rank_G = 7 \geq 4 = \frac{|r_a|}{2}$, a concludes that her current firm lies in the 2nd half of her ordered preference list. Therefore, she only needs to check the 1 position on the right of G (instead of the 6 positions on the left of G) in her preference list about whether it contains firm B: if Yes, a rejects the proposal of B (since she prefers it less than G), otherwise a accepts the proposal of B (rejecting G). In this particular example, firm B lies on the left of G, therefore a accepts the proposal of B without calculating $rank_B$. To sum up, knowing that $rank_G = 7$, a performed a single comparison with the single element (on the left of G) in position 8 of her preference list instead of 6 comparisons involving the 6 elements in positions 1, 2, 3, 4, 5 and 6 - from the beginning of her preference list to G - in order to decide whether to accept or reject the proposal of firm B. Similarly, when a receives a proposal by firm H (round 4), she knows that her current firm is B, though she does not know $rank_B$ since it was not calculated during the previous round (round 3). Now, for deciding whether to accept or reject the proposal of H, a cannot avoid searching from the beginning of r_a until either B (current firm) or H occurs. After checking the first 5 positions of r_a, a finds firm H, accepts its proposal and "remembers" its rank ($rank_a(H) = 5$). When a receives a proposal by C (round 5), given that she knows that $rank_C = 5 \geq 4 = \frac{|r_a|}{2}$, it suffices to examine the last 3 positions of her preference list (instead of the first 4 ones) about whether they contain firm C. The last 3 positions of r_a do not contain C. Therefore, a accepts the proposal of C (rejecting H). The procedure continues until a knows $rank_{current}$ and $rank_{current} < \frac{|r_a|}{2}$.

Regarding the maximum required number of comparisons, observe that during each pair of successive rounds (even-odd round) $|F|$ comparisons are performed. For instance, in the example of Fig. 8, processing the pair of successive proposals by G and B required 8 comparisons (7 for G and 1 for B), processing the pair of successive proposals by H and C required 8 comparisons

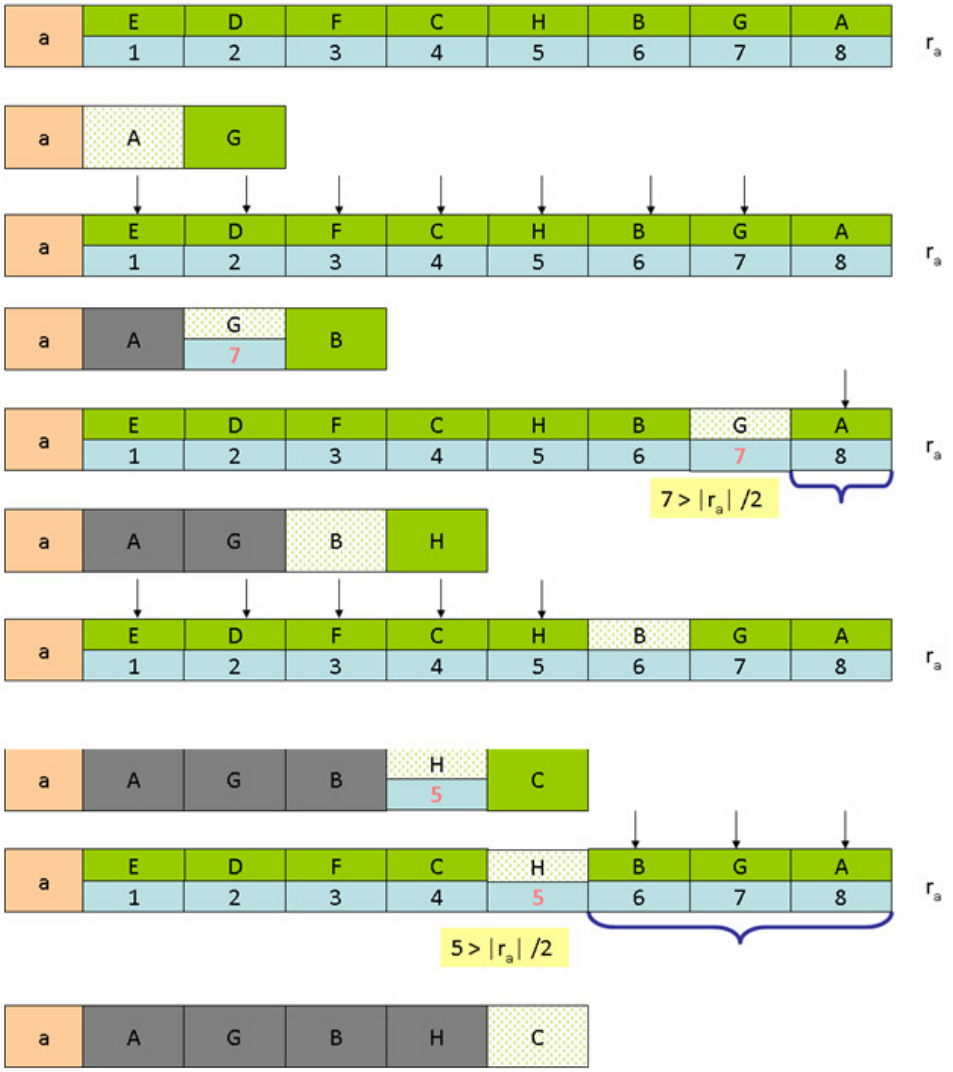

Figure 8. A bad instance for “partial search”.

(5 for H and 3 for C), and so on. Therefore, since the procedure continues until $rank_{current} = \frac{|F|}{2}$ and the number of comparisons required by each pair of successive rounds is $|F|$, at most $\frac{|F|}{4} \cdot |F|$ comparisons are needed for reaching the middle of the preference list. If a candidate receives proposals by all firms in her preference list, $\Sigma_{i=1}^{\frac{|F|}{2}} i$ additional comparisons will be needed. Overall, at most $\frac{|F|}{4} \cdot |F| + \Sigma_{i=1}^{\frac{|F|}{2}} i$ comparisons may be performed by a single candidate. Since $|F| = n$, the number of required comparison per candidate is $\frac{n}{4}n + \Sigma_{i=1}^{\frac{n}{2}} i$.

In order to determine for which values of n this second implementation approach is more efficient, we have to evaluate the following inequality:

$$\Sigma_{i=1}^{n-1} i > \frac{n}{4}n + \Sigma_{i=1}^{\frac{n}{2}} i \Leftrightarrow \frac{n(n-1)}{2} > \frac{n^2}{4} + \frac{\frac{n}{2}(\frac{n}{2}+1)}{2} \Leftrightarrow \frac{n^2}{2} - \frac{n}{2} > \frac{3n^2}{8} + \frac{n}{4} \Leftrightarrow$$

$$\frac{n^2}{2} - \frac{n}{2} - \frac{3n^2}{8} - \frac{n}{4} > 0 \Leftrightarrow \frac{n^2}{8} - \frac{3n}{4} > 0 \Leftrightarrow \frac{1}{8}(n(n-6)) > 0$$

Thus, for $n > 6$, partial search (i.e., Implementation 2) requires fewer comparisons than exhaustive search (i.e., Implementation 1) for a candidate receiving proposals by all firms in her preference list.

To be exact, when a candidate receives proposals by all firms in her preference list, the total number of required comparisons is at most $\lceil \frac{n}{4} \rceil n + \Sigma_{i=1}^{(\lfloor \frac{n}{2} \rfloor - (1 + \lceil \frac{n}{2} \rceil \mod 2))} i$, if $n \mod 4 \neq 1$ and $\lceil \frac{n}{4} \rceil n + \Sigma_{i=1}^{\lfloor \frac{n}{2} \rfloor} i$, if $n \mod 4 = 1$. This is because there are 4 different cases depending on whether (i) n is even or odd and (ii) a pair of sum n is produced when the middle of the preference list is reached.

4.3. Implementation 3: "search without searching"

In the context of our third implementation approach, we focused on minimizing the number of comparisons required for solving the Firms/Candidates problem using the algorithm of Gale and Shapley. For the previous two implementation approaches, we needed to maintain for each candidate a sorted preference list containing all firms. However, candidates do not know the rank of each firm on their preference lists. This implies that whenever a candidate a receives a new proposal by some firm F, a needs to search for F in her preference list r_a and determine whether it comes before or after her current firm in order to decide whether to accept or reject the proposal of F. However, if a knew $rank_F$ and $rank_{current}$, she could just compare these two values and make a straightforward decision on whether to accept (if $rank_F < rank_{current}$) or reject (if $rank_F > rank_{current}$) the proposal of F.

This can be possible if we abandon the idea of the sorted candidates' preference list. In particular, in the context of this third implementation approach, each candidate's a preference list r_a is replaced with a new list r'_a with the following characteristics: (i) r'_a has a position for each firm, (ii) firms are sorted in alphabetical order and are placed in r'_a accordingly (e.g., firm A is placed in the 1st position of r'_a, firm B is placed in the 2nd position of r'_a, and so on) and (iii) each position i of r'_a contains a discrete integer value in $[1, |F|]$ reflecting the rank of firm i in candidate's a

preference list. For example (Fig. 9), let a be a candidate with preference list $r_a = [B, C, D, A]$ sorted in decreasing order. Then $r'_a = [4, 1, 2, 3]$ since firm A is ranked 4th, firm B is ranked 1st, firm C is ranked 2nd and firm D is ranked 3rd in r_a.

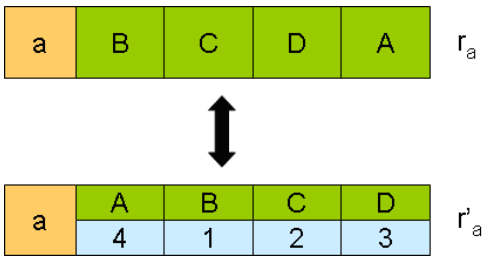

Figure 9. $r'a$ replaces r_a for each candidate a.

In this way, candidate a decides whether to accept or reject the proposal of a firm F by simply retrieving and comparing the value of her current firm $r'_a(current)$ and the value $r'_a(F)$ of firm F in r'_a. Candidate a accepts the proposal of F if $r'_a(F) < r'_a(current)$, and rejects it otherwise. Following this approach, the example discussed in Section 4.2 would evolve as follows. Candidate a accepts proposals by firms A, G, B and H, successively. The modified list r'_a maintained by a is depicted in Fig. 10. Candidates always accept the first proposal they receive. Therefore, a accepts the proposal by firm A (round 1). Upon receiving a proposal by firm G (round 2), a retrieves and compares $r'_a(A)$ and $r'_a(G)$, whose values are 8 and 7, respectively. Since 7 is the smaller of the two values, a (having performed a single comparison) decides to accept the proposal of G (rejecting A). Similarly, upon receiving a proposal by firm B (round 3), a retrieves and compares $r'_a(G)$ and $r'_a(B)$, whose values are 7 and 6, respectively. Since 6 is the smaller of the two values, a (again, having performed a single comparison) decides to accept the proposal of B (rejecting G). Finally, when a receives a proposal by H (round 4), she retrieves and compares $r'_a(B)$ and $r'_a(H)$, with values 6 and 5, respectively. Since 5 is the smaller of these two values, a decides (having performed a single comparison) to accept the proposal of H (rejecting B). This indicative run is illustrated in Fig. 11.

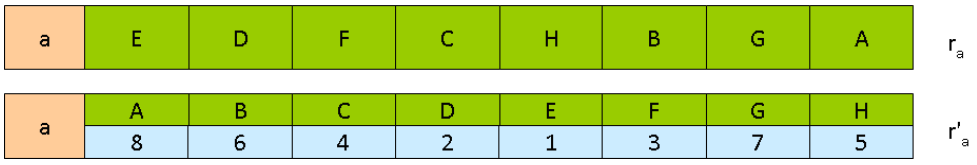

Figure 10. $r'a$ for candidate a.

A "bad" instance would result if each candidate received the maximum number of proposals, i.e., n proposals (where n equals the number of firms). Then, exactly $n - 1$ comparisons would be required for each candidate to decide whether to accept or reject the proposal received in each round. This implementation approach drastically decreases the overall number of comparisons required for the execution of the Gale and Shapley algorithm for the Firms/Candidates problem. In particular, the $O(n^2)$ comparisons required by each candidate in the context of the first two

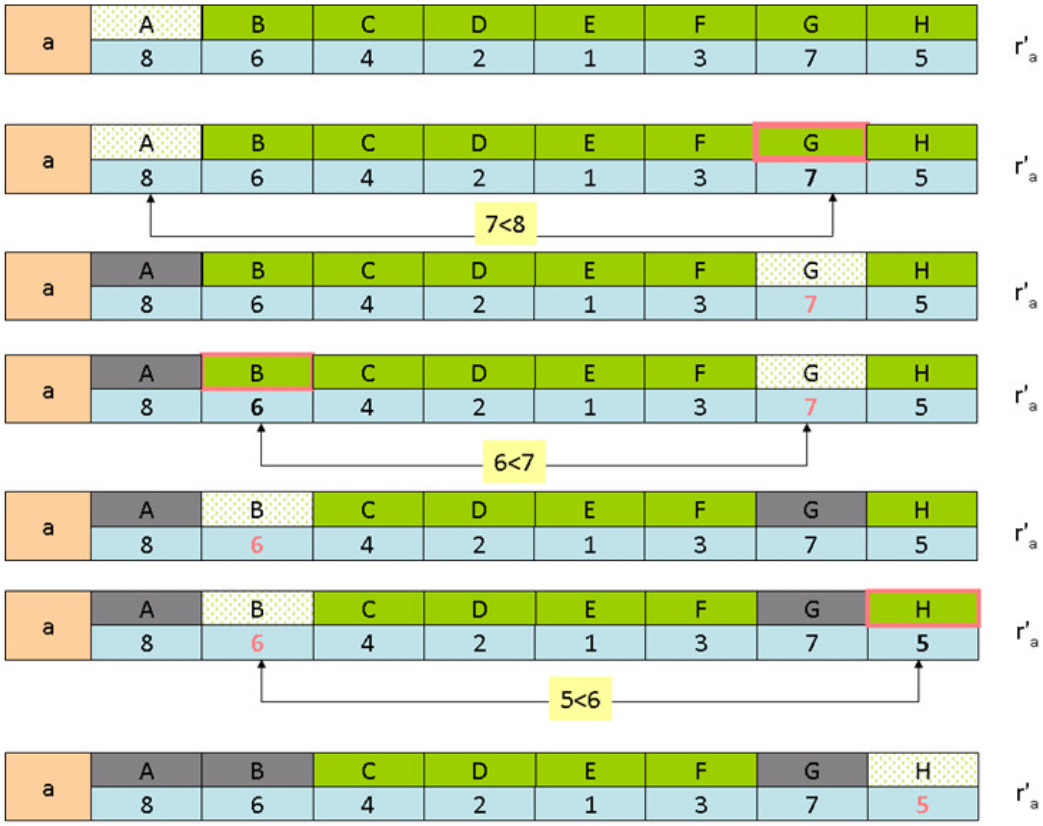

Figure 11. The “search without searching” implementation approach.

implementation approaches have now been decreased to $O(n)$, thus yielding a significant overall reduction in the number of required comparisons (from $O(n^3)$ to $O(n^2)$). Despite the fact that the theoretical upper bound remains polynomial in the number of firms, this reduction can be of great practical importance.

Termination and Stability. All three suggested approaches actually implement the original Gale and Shapley algorithm. The difference among them consists in the number of comparisons required for candidates to decide whether to accept or reject newly received proposals. Therefore, proofs regarding termination and stability of the result are the same as they appear in the original work of Gale and Shapley [9].

However, we also experimentally validated the stability of the results returned by our implementations. In particular, we developed "Check", a routine which verifies that the assignments computed by our implementations do not contain blocking pairs. "Check" receives as input the candidates' preference lists, the preference lists of firms and the computed assignments of candidates to firms and outputs an error message if a blocking pair is detected.

Initially, we define "stability", a Boolean variable which becomes false if a blocking pair is found. For each candidate c, firms in her preference list are exhaustively checked until either her current firm is found or the end of the preference list is reached (i.e., c is not assigned to any firm). For each firm $f \in F$ in c's preference list, we check whether c precedes the last candidate assigned to f on the preference list of f. If this is true, then (i) f prefers c more than at least one other candidate already assigned to it and (ii) c prefers f more than her current firm. Thus, (f, c) form a blocking pair and "stability" becomes false. This procedure continues until either "stability" becomes false or all candidates are checked. If the procedure terminates and "stability" is true, then the examined assignment does not contain blocking pairs. This is because all firms that candidate c prefers more than her current firm do not prefer c more than their currently assigned candidates. That is, no firm occurring in c's preference list earlier than her current firm would prefer replacing a currently assigned candidate with c. Therefore, c cannot participate in any blocking pair.

"Check" also detects incomplete assignments by comparing the number of uncovered posts with the number of unassigned candidates. Since all preference lists are complete, a non-zero value of uncovered posts or unassigned candidates indicates an incomplete assignment. When the number of available posts equals the number of candidates, uncovered posts and unassigned candidates are equal to 0.

4.4. Experimental framework and results

In this section, we present our experimental results regarding the comparison of the three suggested implementation approaches for the Gale and Shapley algorithm for the Firms/Candidate problem, in terms of required number of comparisons and execution time. Obtained results indicate that the "search without searching" approach (i.e., Implementation 3) outperforms the previous two.

Code was developed using C++ and Visual Studio 2017 version 15.7.5.

For the development and the experimentation a computer with the following characteristics was used.

- Processor Intel Core i7-7500U CPU 2.70GHz

- RAM 4.00GB (3.88GB usable)
- System Type 64-bit Operating System, x64-based processor
- Windows 10 Home Operating System

In order to compare the three suggested implementation approaches for the Gale and Shapley algorithm for the Firms/Candidates problem in terms of required number of comparisons and execution time, we implemented a program which generates arbitrary preference lists for Firms and Candidates. More precisely, this program receives as input the number of Firms, the number of vacancies per Firm as well as the number of Candidates. It outputs the list of vacancies for each Firm as well as the preference lists for Firms and Candidates.

In each trial, all three approaches were executed for the same set of input instances. We started by considering small sets of decades of Firms/Candidates and continued by gradually increasing set size to 100, 500 and 1000 Firms/Candidates. For each of these set sizes, we evaluated the performance of the three approaches for worst-case instances, i.e., input sets maximizing the number of proposals and rejections.

Furthermore, we developed an auxiliary routine which transforms candidates' preference lists (r_c) to lists indicating the rank of each firm on the preference list of each candidate (r'_c) used by the "search without searching" implementation approach.

Comparisons. Figures 12 and 13 show the number of comparisons performed by each of the three suggested implementation approaches for the Gale and Shapley algorithm for the Firms/Candidates problem. In particular, Fig. 12 shows the number of comparisons required when arbitrary preference lists are provided as input to the three suggested implementation approaches. Fig. 13 shows the number of comparisons required when input instances involve maximum number for proposals/rejections. It can be observed that in both cases, especially as the size of the input increases, the "search without searching" approach (i.e., Implementation 3) requires a significantly smaller number of comparisons compared to the other two approaches.

Execution time. As expected, the number of required comparisons significantly affects execution time. Regardless of the kind of candidates' preference lists provided as part of the input (i.e., preference lists that maximize the number of proposals/rejections or arbitrary preference lists), it is observed that the "search without searching" implementation approach outperforms the other two. In particular, Fig. 14 shows the execution time for each of the three suggested implementation approaches receiving as input preference lists of size 10, 100, 500 and 1000 which maximize the number of proposals and rejections. Fig. 15 shows the execution time for each of the three suggested implementation approaches when arbitrary preference lists of size 10, 100, 500 and 1000 are used. Note that execution time of the "search without searching" implementation approach does not include the time needed for the transformation of candidates' preference lists (r_c) to lists indicating the rank of each firm on the preference list of each candidate (r'_c). Fig. 16 shows the execution time for each of the three suggested implementation approaches when arbitrary preference lists of size 10, 100, 500 and 1000 are used, but, now, time required for the transformation of candidates' preference lists has been taken into account for the "search without searching" implementation approach. However, it is realistic to assume that in many practical contexts, the required

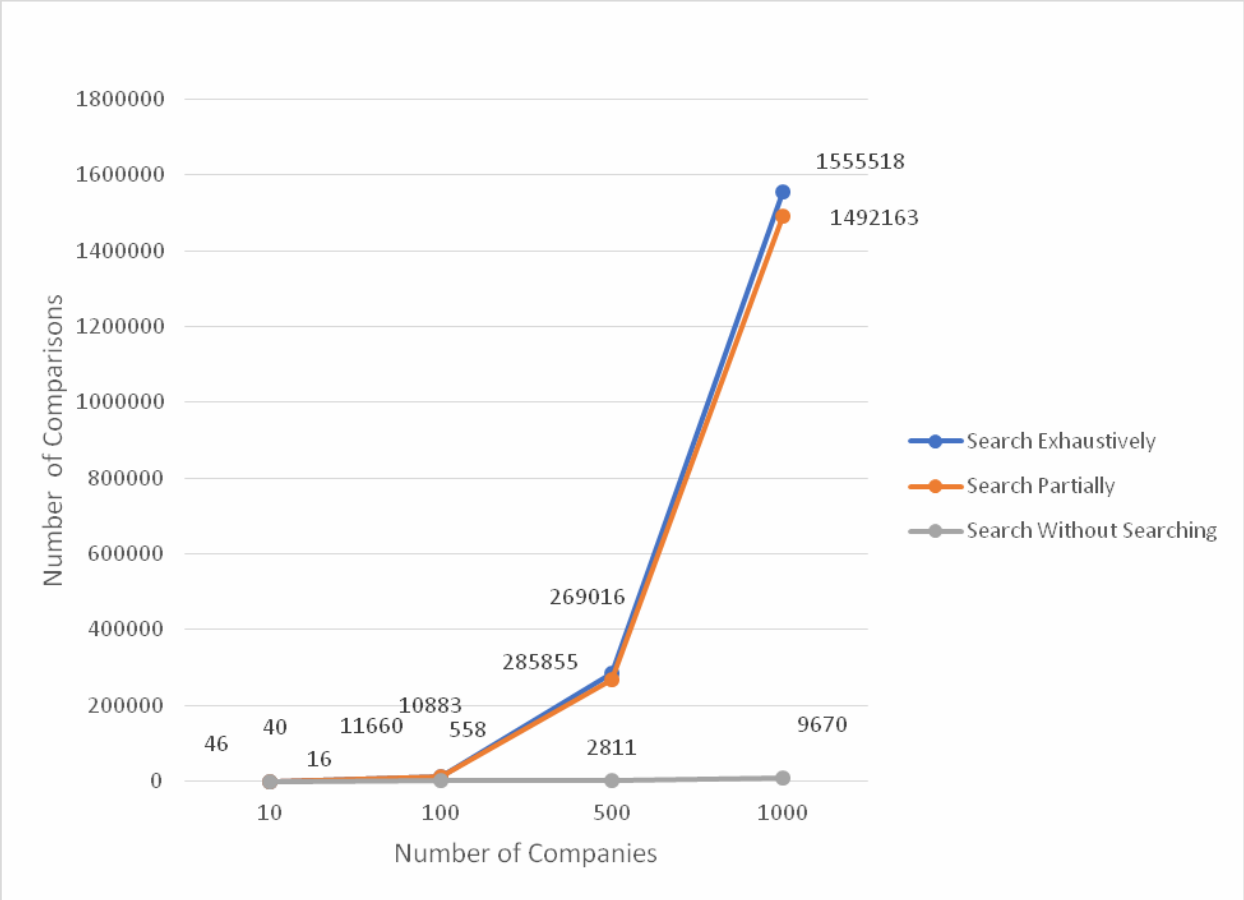

Figure 12. Input instances with arbitrary preference lists: number of comparisons required by the suggested implementation approaches.

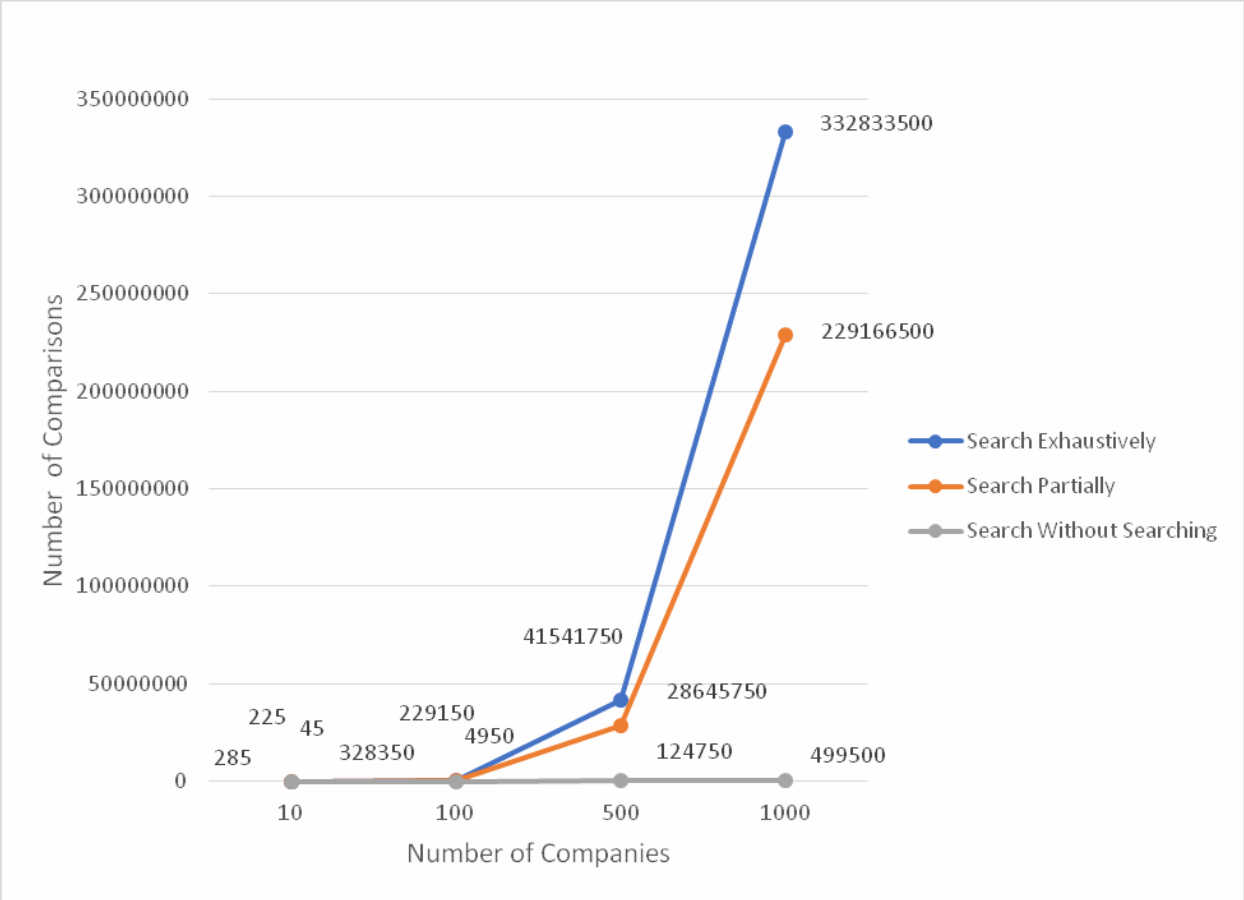

Figure 13. Input instances with preference lists maximizing proposals/rejections: number of comparisons required by the suggested implementation approaches.

list transformation takes place independently, before the execution of the the Gale and Shapley algorithm. Then, it can be observed that the "search without searching" approach (i.e., Implementation 3) significantly outperforms the other two approaches, especially as the size of the input (i.e., number of firms and candidates) increases.

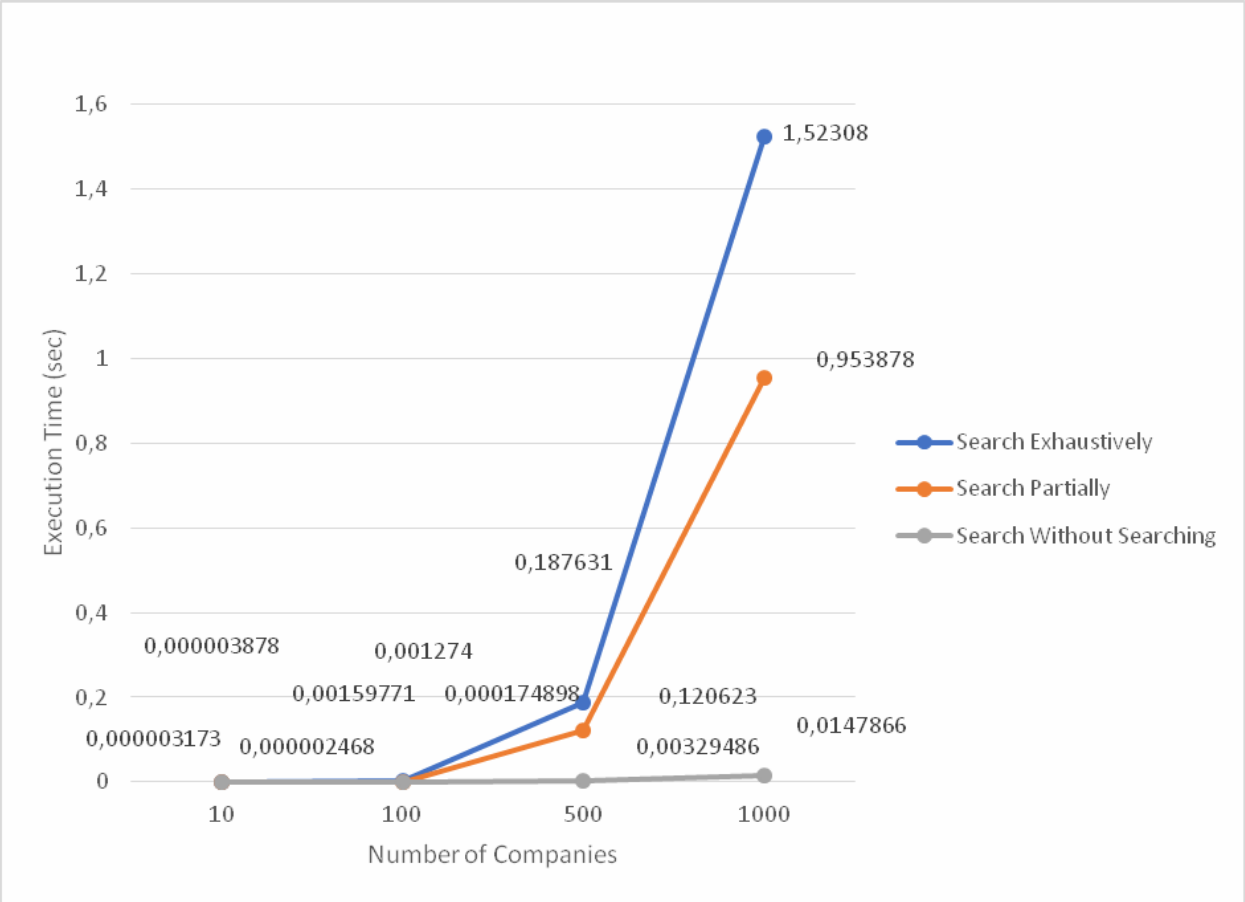

Figure 14. Input instances with preference lists maximizing proposals/rejections: execution time required by the suggested implementation approaches.

5. Concluding Remarks

In this work, we focused on a version of the Hospitals/Residents problem first defined in 1962 by Gale and Shapley [9] under the name "College Admissions Problem". In particular, we considered the Firms/Candidates problem, where each Firm wishes to hire at least one Candidate and each Candidate can be eventually assigned to a single Firm. Preference lists of candidates and firms are complete, i.e., firms rank all candidates and candidates rank all firms. The Gale and Shapley "propose-and-reject" algorithm computes a stable matching for the Firms/Candidates problem. We suggested an efficient implementation of the Gale and Shapley algorithm. The efficiency of our implementation results from replacing candidates' preference lists with lists indicating the rank of

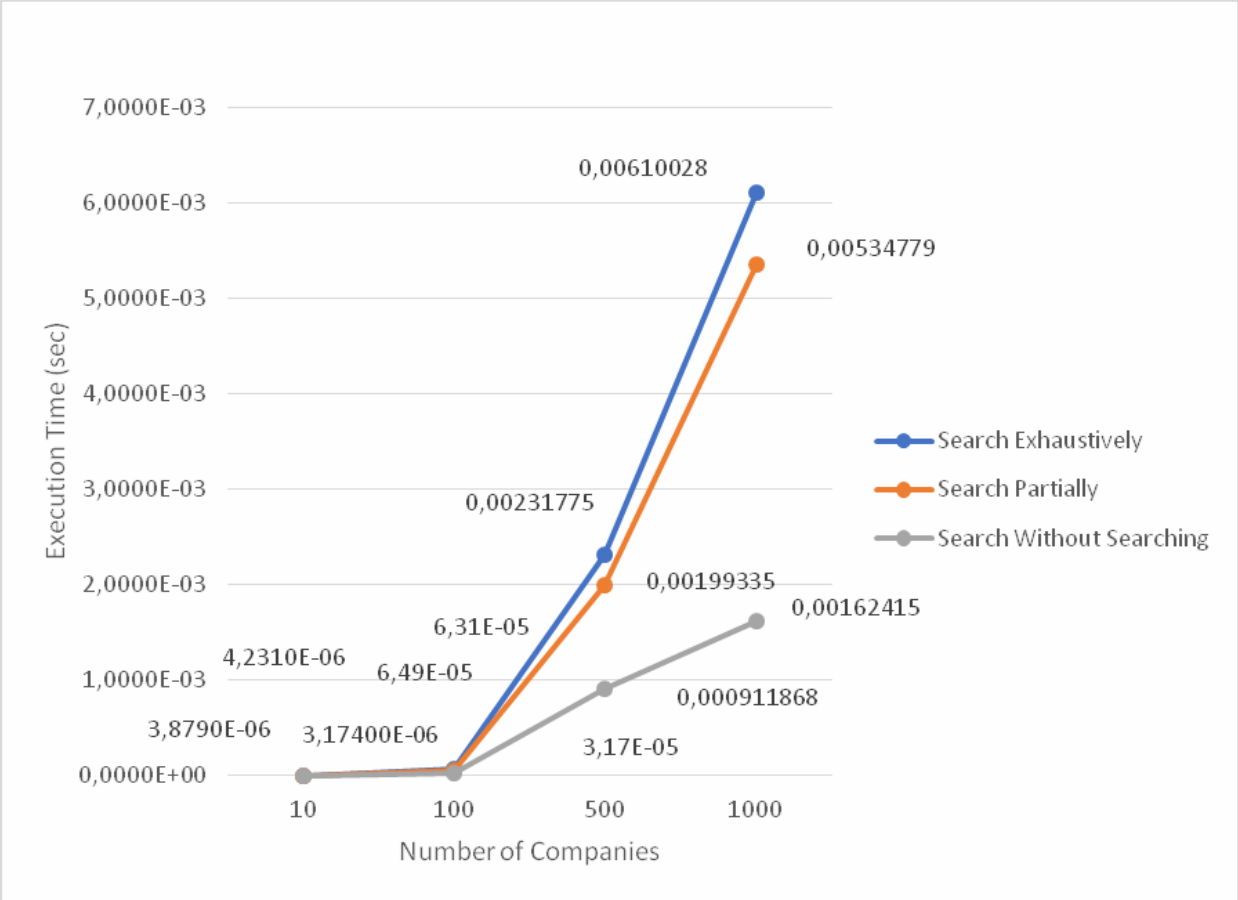

Figure 15. Input instances with arbitrary preference lists: execution time required by the suggested implementation approaches (not including list transformation time).

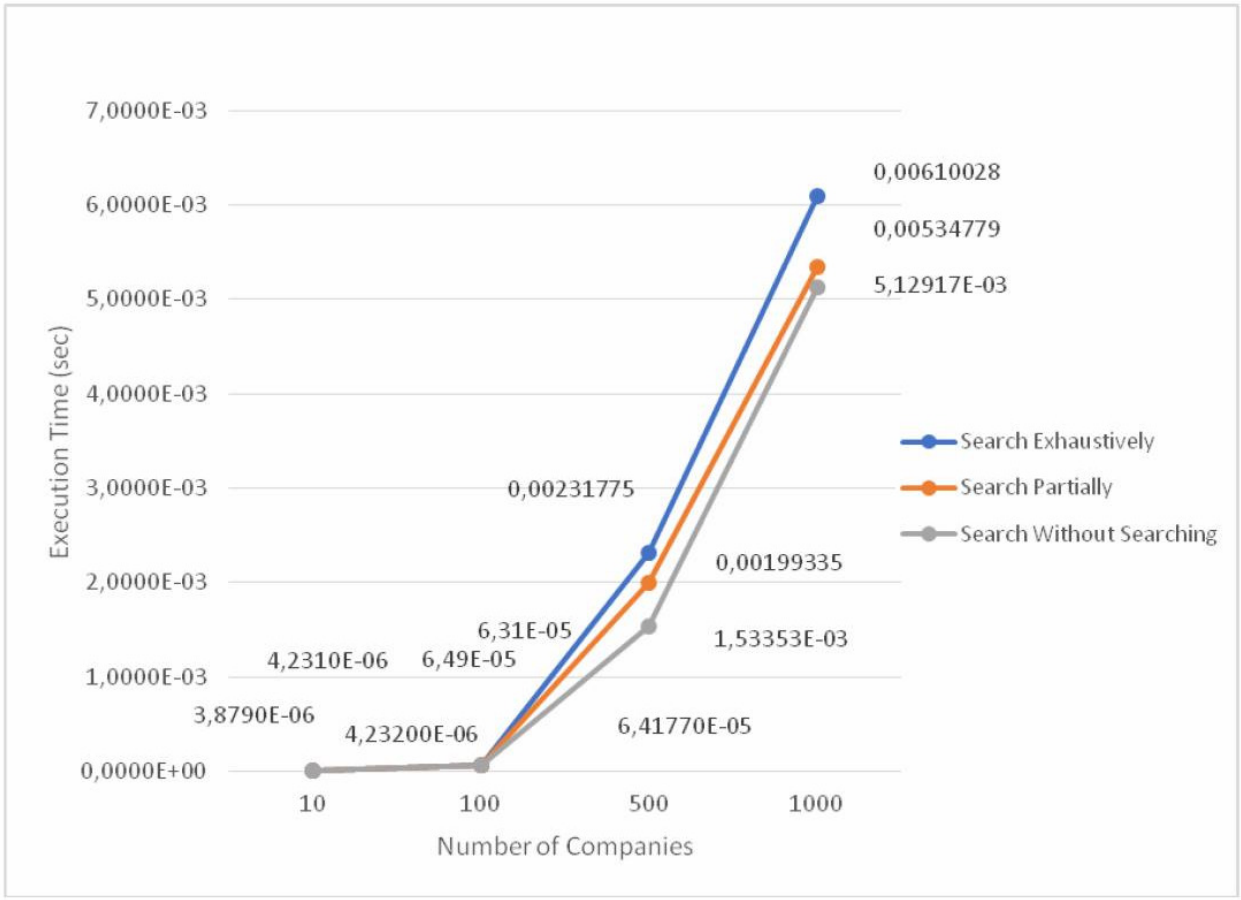

Figure 16. Input instances with arbitrary preference lists: execution time required by the suggested implementation approaches (including list transformation time for the "search without searching" implementation).

each firm on the preference list of each candidate. In this way, searching is avoided and the number of comparisons required by candidates in order to decide whether to accept or reject a proposal by a firm is significantly decreased. This reduction, even not affecting the polynomial execution time of the Gale and Shapley algorithm, can still be of great practical importance.

References

[1] D. J. Abraham, R. W. Irving and D. F. Manlove, The student-project allocation problem, *In Proceedings of the 14th International Symposium on Algorithms and Computation (ISAAC 2003)* (2003), LNCS Springer **2906**, 474–484.

[2] H. Assiyatun, B. Rahadjeng, E. T. Baskoro, The connected size Ramsey number for matchings versus small disconnected graphs, *Electron. J. Graph Theory Appl.* **7** (1) (2019), 113–119.

[3] Canadian Resident Matching Service, How the matching algorithm works. *Web document available at https://www.carms.ca/the-match/how-it-works/.*

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to algorithms, 3rd ed. *MIT press*, 2009.

[5] P.-T. Do, N.-K. Le, V.-T. Vu, Efficient maximum matching algorithms for trapezoid graphs, *Electron. J. Graph Theory Appl.* **5** (1) (2017), 7–20.

[6] P. Golle, A Private Stable Matching Algorithm, *In Proceedings of the 2006 International Conference on Financial Cryptography and Data Security (FC 2006)* (2006), LNCS Springer **4107**, 65–80.

[7] D. Gusfield and R.W. Irving, *The Stable Marriage Problem: Structure and Algorithms*, MIT Press, 1989.

[8] D. Gale and L. S. Shapley, College admissions and the stability of marriage, *Office of Naval Research*, 1960–1961.

[9] D. Gale, L. S. Shapley, College admissions and the stability of marriage, *American Mathematical Monthly* **69** (1) (1962), 9–15.

[10] D. Gale and M. Sotomayor, Some remarks on the stable matching problem, *Discrete Appl. Math.* **11** (1985), 223--232.

[11] R. W. Irving, Matching medical students to pairs of hospitals: a new variation on an old theme, *In Proceedings of the 6th European Symposium on Algorithms (ESA 1998)* (1998), LNCS Springer **1461**, 381–392.

[12] R. W. Irving, Man-exchange stable marriage, University of Glasgow, *Computing Science Department Research Report*, TR-2004-177, 2004.

[13] R. W. Irving, The cycle roommates problem: a hard case of kidney exchange, *Information Processing Letters* **103** (1) (2007), 1–4.

[14] R. W. Irving, D. F. Manlove, S. Scott, The Hospitals/Residents Problem with Ties, *In Proceedings of the 7th Scandinavian Workshop on Algorithm Theory (SWAT2000)* (2000), 259–271.

[15] K. Iwama, S. Miyazaki, A Survey of the Stable Marriage Problem and Its Variants, *In Proceedings of the 2008 IEEE International Conference on Informatics Education and Research for Knowledge-Circulating Society* (2008), 131–136.

[16] J. Kleinberg, É. Tardos, Algorithm Design. *Pearson*, 2006.

[17] D. E. Knuth, Stable Marriage and its Relation to Other Combinatorial Problems, *In CRM Proceedings and Lecture Notes*, volume 10, American Mathematical Society (1997).

[18] E. Kujansuu, T. Lindberg, and E. Mäkinen, The stable roommates problem and chess tournament pairings, Divulgaciones Matemáticas, **7** (1) (1999), 19-28.

[19] László Lovász and M. D. Plummer, Matching Theory, *Annals of Discrete Mathematics* **121** (1986).

[20] B. M. Maggs, R. K. Sitaraman, Algorithmic Nuggets in Content Delivery, *ACM SIGCOMM Computer Communication Review*, **45** (3) (2015), 52–66.

[21] V. S. Malhotra, On the stability of multiple partner stable marriages with ties, *In Proceedings of the 12th Annual European Symposiumon Algorithms (ESA 2004)* (2004), LNCS Springer **3221**, 508-519.

[22] D. Manlove, Hospitals/Residents Problem, *Encyclopedia of Algorithms*, Chapter 180 (2008), 390–394.

[23] D. F. Manlove, Algorithms of Matching under preferences, *World Scientific Publishing* (2013), p. 524.

[24] D. F. Manlove, R. W. Irving, K. Iwama, S. Miyazaki, and Y. Morita, (2002) Hard variants of stable marriage, *Theoretical Computer Science* **276** (1-2) (2002), 261–279.

[25] I. McBride and D. F. Manlove, The Hospitals / Residents Problem with Couples: Complexity and Integer Programming models. *Technical report, Computing Research Repository, Cornell University Library* (2013).

[26] National Kidney Foundation, Organ Donation and Transplantation Statistics (https://www.kidney.org/news/newsroom/factsheets/Organ-Donation-and-Transplantation-Stats), Accessed November 7, 2018.

[27] Stable allocations and the practice of market redesign, Scientific Background on the Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel 2012 compiled by the Economic Sciences Prize Committee of the Royal Swedish Academy of Sciences (2012).

[28] National Resident Matching Program, http://www.nrmp.org/

[29] A. E. Roth, The evolution of the labor market for medical interns and residents: a case study in game theory, *Journal of Political Economy* **92** (6) (1984), 991-1016.

[30] A. E. Roth, On the allocation of residents to rural hospitals: a general property of two-sided matching markets, *Econometrica*, **54** (1986), 425–427.

[31] A. E. Roth, Deferred Acceptance Algorithms: History, Theory, Practice, and Open Questions, *International Journal of Game Theory*, Special Issue in Honor of David Gale on his 85th birthday, **36** (2008), 537–569.

[32] A. E. Roth and M. A. O. Sotomayor, Two-sided matching: a study in game-theoretic modeling and analysis, *Econometric Society Monographs* **18** (1990), Cambridge University Press.

[33] A. E. Roth, T. Sonmez, and M. U. Unver, Pairwise kidney exchange, *Journal of Economic Theory* **125** (2) (2005), 151–188.