Computers & Industrial Engineering 169 (2022) 108244

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

A centralized matching scheme to solve the role-partner allocation problem in collaborative networks

Javier Andrade-Garda [a], Juan Ares-Casal [a], Marta Hidalgo-Lorenzo [a], Juan-Alfonso Lara [b], David Lizcano [b], Sonia Suárez-Garaboa [a,*]

[a] *Grupo ISLA, Dep. Ciencias da Computación e Tecnoloxías da Información, Fac. Informática, Universidade da Coruña, 15071 A Coruña, Spain*
[b] *Universidad a Distancia de Madrid, Vía de Servicio 15, 28400 Collado-Villalba, Spain*

ARTICLE INFO

Keywords:
Centralized matching scheme
Collaborative network
Matching problem
Partner
Role

ABSTRACT

In the current constantly changing business and economic environment, partners (i.e., individuals and/or enterprises) create Collaborative Networks to join efforts and undertake new projects together, thus allowing them to face business opportunities that would not be possible if attempted by them individually. In this situation, an assignment problem arises, since these projects involve the performance of a group of tasks or processes (named roles) that have to be distributed among the partners. Specifically, this problem, called the Role-Partner Allocation (RPA) problem in Collaborative Networks is a two-sided matching problem with lower and upper quotas on the partner's side, and incomplete and partially ordered preference lists on both sides. A matching problem, and thus also the RPA problem, should be solved by a centralized matching scheme. However, allocations in Collaborative Networks continue to be mainly created by *ad hoc* arrangements, which takes a long time and is hard work. Looking for a reliable and faster way of distributing roles among partners in a Collaborative Network, the existing centralized matching schemes expected to solve the RPA problem (e.g., DA algorithm, SOSM, CA-QL algorithm, and EADAM) are studied in this paper, concluding that none of them obtain a matching that properly meets the requirement of the RPA problem. Therefore, a new centralized matching scheme to solve the RPA problem is proposed, discussed and exemplified.

1. Introduction

As defined in Camarinha-Matos, Afsarmanesh, Galeano, and Molina (2009), a collaborative network (CN) is a network consisting of a variety of entities (e.g., organizations and people) that are largely autonomous, geographically distributed, and heterogeneous in terms of their operating environment, culture, social capital, and goals, but that collaborate to better achieve common or compatible goals, thus jointly generating value, and whose interactions are supported by computer network.

CNs born as a way of facing the significant challenges arising from the rapidly changing and ever more globalized business environments (Appio, Martini, Massa, & Testa, 2017) that have become increasingly volatile, uncertain, complex, and ambiguous (vom Brocke et al., 2018). Dealing with the new challenges requires organizations and people to increase their flexibility, agility, and speed. To do this, in an interconnected world thanks to the advances in the information and communication technologies (ICT) where the barriers of time and distance disappear, new ways of working and new forms of dividing labor arise (Brynjolfsson & McAfee, 2014). ICT have allowed enterprises to move from highly data-driven environments to more cooperative information/knowledge-driven environments (Camarinha-Matos et al., 2009).

In this business atmosphere, CNs have increasingly become a mainstay of business operations (Camarinha-Matos, Fornasiero, Ramezani, & Ferrada, 2019; Camarinha-Matos, 2014; Durugbo, 2016). It shows the growth trend in enterprises and professionals in seeking joint activities to allow them to participate in competitive business opportunities for innovative developments.

In CNs, partners seek to undertake a business opportunity together in a given field that would not be possible or would have a higher cost if attempted by them individually. Typically, there are one or more initiators—that is the partner who initially detects the business opportunity—and one or more partners interested in participating in the project

* Corresponding author.
E-mail addresses: javier.andrade@udc.es (J. Andrade-Garda), juan.ares@udc.es (J. Ares-Casal), marta.hidalgo@udc.es (M. Hidalgo-Lorenzo), juanalfonso.lara@udima.es (J.-A. Lara), david.lizcano@udima.es (D. Lizcano), sonia.suarez@udc.es (S. Suárez-Garaboa).

https://doi.org/10.1016/j.cie.2022.108244
Received 23 October 2019; Received in revised form 27 January 2022; Accepted 10 May 2022
Available online 14 May 2022

by joining the CN. Usually, the CN initiator configures the project (i.e., it decomposes the entire process into sub-processes) and the parties interact somehow in order to specify the operational requirements of the CN and to create it. Process decomposition is a complex task that can be approached in multiple ways. See, for example, Milani, Dumas, Matulevičius, Ahmed, and Kasela (2016) for detailed information about decomposition heuristics, as well as about criteria and associated metrics to assess the "goodness" of a given decomposition.

Hence, facing a business opportunity by creating a CN implies undertaking a new project that involves the performance of a group of processes or activities (here referred to as roles) that have to be distributed among the partners, thus an assignment problem arises. As indicated in Malaguti and Medina Durán (2019), assignment problems are about the best way of assigning a first set of n elements to a second set of n elements, where each element of the first set has to be assigned to exactly one element of the second set, and vice versa, each element of the second set has assigned exactly one element of the first set. When the cardinality n of the first set is larger than the cardinality of the second set, let us denote it by m, then we are facing a Semi-Assignment problem (SAP), where each element i of the first set has to be assigned to exactly one element j of the second set, while each element j of the second set has assigned a given number $q_j \geq 1$ of elements of the first. In the CNs environment, such first set is the set of n roles and such second set is the set of m partners ($n \geq m$). As the total number of roles to be assigned usually exceeds that of the partners, a SAP arises in most cases.

In particular, the SAP of allocating roles among partners in a CN is an example of the so-called two-sided matching problem (Gale, 2001). Following the definition in Abraham, Irving, and Manlove (2007), in this type of assignment problems the input entities (in this case, the roles and the partners) are partitioned into two disjoint sets (in this case, the set of roles and the set of partners), and the aim is to pair elements of one set with elements of the other set subject to various criteria such as capacity constraints and preference lists.

The assignment of roles to partners is commonly addressed in real life in CNs by negotiation between the partners. This takes a long time and is hard work. Partners have to present their arguments (i.e., knowledge, capabilities, personal preferences, availability, etc.) and discuss the situation in order to reach an agreement regarding the distribution of the roles of the project among them.

Even if certain information about the partners is stored, the above-mentioned process is complex because it implies human beings, each one with its own personality, requirements, tastes, etc. In the current changing, globalized and competitive business environment, a reliable and faster way of assigning roles to partners is required to minimize the risk of losing the business opportunity. However, there is no such proposal in CNs, where allocations continue to be mainly created by *ad hoc* arrangements. This is despite, as Abraham, Irving, and Manlove (2003) indicate, both historical evidence and economic analysis show that participants involved in matching problems should not be allowed to construct an allocation by approaching one another directly in order to make *ad hoc* arrangements. Instead, the allocation process should be automated by means of a centralized matching scheme, which is an algorithm to solve a matching problem. A centralized matching scheme will greatly contribute to the progress of CNs by solving the SAP of distributing roles among partners, since it will allow the process to be sped up and simplified, thus preventing the complications that occur when constructing an allocation by partners approaching one another directly. The SAP to be solved and its core components have been formally defined in Andrade-Garda et al. (2018) under the name of Role-Partner Allocation (RPA) problem.

Taking this formal definition into account, we firstly looked for an algorithm to solve the RPA problem among the main existing centralized matching schemes that had been successfully applied to solve other two-sided matching problems. However, as discussed below, the current centralized matching schemes do not fulfil the requirements of the RPA problem in a comprehensive manner. Thus, they cannot therefore be directly applied to solve it. Because of this, it is necessary to propose a new scheme (algorithm) for distributing roles among partners in a CN (i.e., for solving the RPA problem).

The remainder of this paper is structured as follows. Section 2 briefly presents and discusses the formal definition of the RPA problem and its core components. Section 3 analyses the different existing approaches expected to solve it, concluding that none of them properly satisfy the specified conditions. As a first step towards the definition of the new algorithm, Section 4 presents the three main properties that must be considered for a matching algorithm, and after which Section 5 presents the new algorithm to find a stable matching giving an instance of the RPA problem and considering the previous three properties. Section 6 presents an example to illustrate its applicability and, finally, Section 7 presents the most relevant conclusions and future work.

2. RPA problem overview

As defined in Andrade-Garda et al. (2018), the RPA problem is an example of the two-sided matching problem that, according to Gale (2001) and Abraham et al. (2007), is a class of problems in which the input set of entities can be partitioned into two disjoint sets A and B, and the aim is to find a matching M of members of A to members of B subject to various criteria. These criteria usually involve preference lists and capacity constraints, such is the case with the RPA problem. A brief summary of the RPA problem and its core components is presented in the following paragraphs in order to make this paper self-contained.

In this case, the two characteristic disjoint sets (generically denoted as A and B in the above-mentioned definition) of the two-sided matching problems are the set of partners and the set of roles. Specifically, let $I \equiv \{i_0, i_1, \cdots, i_{n-1}\}$ with $|I| \geq 2$ denote the finite set of partners and $R \equiv \{r_0, r_1, \cdots, r_{m-1}\}$ the finite set of roles. A generic element in I is denoted by i and a generic element in R is denoted by r.

Each partner i defines an incomplete and partially ordered preference list $\beta(i) \subseteq [r_0, r_1, \cdots, r_k]$ with the roles that it is willing to perform in the project ($k \leq m-1$), and each role r also defines an incomplete and partially ordered preference list $\alpha(r) \subseteq [i_0, i_1, \cdots, i_t]$ with the partners appropriate to perform the role ($t \leq n-1$). Note that a role is not a human being, so it has no personal preferences or individual tastes. Thus, the preference list of each role is created using objective criteria (i.e., the partner with most knowledge or capabilities for the role will be on the top of the list). However, partners are human beings (or sets of human beings, like enterprises), so that besides their knowledge and capabilities they may also have personal preferences and individual tastes, which will be reflected in their preference lists.

Preference lists on both sides are incomplete because not all partners have to be empowered to perform all roles (i.e., to do any kind of activity), and similarly not all partners have to want to or be able to perform all roles (e.g., because of a lack of knowledge or different personal preferences and individual tastes). In addition, lists are partially ordered because they include ties. That is to say, partner i strictly prefers $\beta(i)[x]$ to $\beta(i)[x+1] (x = 0..|\beta(i)|)$ or is indifferent between them (the same applies to roles).

Each partner i defines $l(i)$ as the number of roles that i requires at least in order to join the project and $u(i)$ as the maximum number of roles that i is willing to perform in the project. It doesn't mean that the aim of i is to play $u(i)$ roles. The aim of i is to play at least $l(i)$ roles (it meets its needs) and by no means will accept to play more than $u(i)$ roles (it is more work than the partner can handle). Note that only if the total number of roles is no less than the minimum total number of roles required by the partners (i.e., $|R| \geq \sum_{i \in I} l(i)$) can a feasible allocation exist. Similarly, only if $\sum_{i \in I} u(i) \geq |R|$ a solution is possible.

Let $M(i)$ be the set of roles assigned to partner i after the allocation process, then $0 < l(i) \leq |M(i)| \leq u(i) \leq |R| - \sum_{j \in I} l(j), j \neq i$. That is to say, each partner has to take the responsibility of a minimum of $l(i)$ roles (with $l(i)$ at least 1; that is, a piece of the project) and a maximum of roles such that each other partner j has the possibility to play a minimum of $l(j)$ roles. Finally, each role has to be played by only one partner (i.e., $|M(r)| = 1$ for all roles).

3. Centralized matching schemes

There are multiple matching mechanisms to solve a two-sided matching problem and are, therefore, potentially applicable to solve the RPA problem. The simplest approach could be a first-come-first-served (FCFS) allocation mechanism. This way, and respecting the lower and upper bounds, the first acceptable partner (role) that asks for a role (partner) is the one that will obtain it. However, as Diebold, Aziz, Bichler, Matthes, and Schneider (2014) conclude in the case of course allocation problems, FCFS doesn't care about equity and preferences can be violated, so another type of mechanism is needed.

Gale and Shapley (1962) presented the central bipartite matching problem with two-sided preferences: The Stable Marriage (SM) problem. An instance of this problem comprises of a set of men and women, and each person ranks each member of the opposite sex in strict order of preference. They suggested the Gale-Shapley deferred acceptance (DA) algorithm to find a stable matching of single men to single women (one-to-one). In the CN context, making an analogy between men and women and roles and partners, this approach is not valid "as it is" as a solution to the RPA problem, basically because (i) not all the partners are valid for playing all the roles, (ii) not all the partners might want to play all the roles, and (iii) a one-to-one matching is not always possible either because the number of roles is greater than the number of partners or because of quota constraints. The classical SM problem with incomplete lists (SMI) (Gusfield & Irving, 1989) allows to tackle (i) and (ii), but (iii) remains unsolved.

The many-to-one generalization of SM was firstly defined by Gale and Shapley under the name College Admissions problem (Gale & Shapley, 1962), where each man corresponds to a student and each woman corresponds to a college which can potentially be assigned multiple students up to a fixed capacity. As indicated in Abdulkadiroglu and Sönmez (2003), the many-to-one version of the DA algorithm, where students are proposing to the colleges, is also referred to as Student Optimal Stable mechanism (SOSM). Likewise, where colleges are proposing to the students, it can be also referred to as College Optimal Stable mechanism. The College Admission problem has been extensively studied—see e.g., Roth and Sotomayor (2008), and Roth (2008) for a survey—, being the most influential application the assignment of residents or students to hospitals by the Hospitals/Residents problem (HR) (Roth, 1986; Roth & Peranson, 1997; Manlove, 2008). In this case, each student corresponds to a resident and each college corresponds to a hospital. A reduction of HR to SMI using the method of "cloning" hospitals exists. That is, replacing each hospital h with capacity $\times$, with $\times$ hospitals denoted $h_1, \ldots, h_x$. However, in practice direct algorithms are applied to HR instances because the cloning technique increases the number of hospitals in a given HR instance by a potentially significant factor (Abraham et al., 2003).

Other SM related problems in different research areas are: the School Choice problem (Abdulkadiroglu & Sönmez, 2003; Chen & Sönmez, 2006), the Course Allocation problem (Diebold et al., 2014; Sönmez & Ünver, 2010), the Student–Project Allocation (SPA) problem (Abraham et al., 2007; Chen & Sönmez, 2006), the Trainee-Project Allocation problem (Gharote, Patil, Lodha, & Raman, 2015), the finding of mentor–mentee matches (Haas & Hall, 2019; Haas, Hall, & Vlasnik, 2018), the spectrum resource allocation problem (Li, Ma, Xu, & Shankaran, 2020), the matching between core nodes and edge nodes for content providers (Qin, Xue, Li, Sun, & Lu, 2021), and, of course, the well-known case with patient-donor pairs (Roth, Sönmez, & Ünver, 2004) — a patient in need of a kidney and a donor (family, friend) who is willing to donate one. Proof of the importance of this field of study is that in 2012 Alvin E. Roth and Lloyd S. Shapley were awarded with the Nobel Prize in Economics for the theory of stable allocations and the practice of market design.

The SM related problems that allow an element of the first set (e.g., hospitals) to has assigned more than one element of the second set (e.g., residents) are more similar to the RPA problem and match with the need of managing upper quotas. However, the management of lower quotas is also required. Lower quotas appear with the College Admission with Lower Quotas (CA-LQ) problem (Biró, Fleiner, Irving, & Manlove, 2010). In a given instance of that problem, each college c has the classical (upper) quota of many-to-one type problems, denoted by $u(c)$, and a lower quota $l(c)$. A matching of applicants to colleges in this context requires every college c satisfying $|M(c)| = 0$ or $l(c) \leq |M(c)| \leq u(c)$. The authors say that c is *closed* if $|M(c)| = 0$, and *open* otherwise. Note that, despite having similar purposes, this algorithm is not directly applicable to the RPA problem. This is because in the case of the RPA problem "closed partners" are entirely unacceptable. An algorithm that solves the RPA problem has to be defined in such a way that a feasible matching involves all partners participating in the project and lower quotas are respected for all partners. Otherwise, the algorithm must return that no solution meets the requirements of the RPA problem. In this regard, Hamada, Iwama, and Miyazaki (2011) also studied the HR problem with lower quota bounds; that is a CA-LQ related problem with similar motivations. In this case, they require the matching to satisfy all lower quotas (i.e., no hospital can be closed in their model). However, they assume that each applicant has a complete preference list (i.e., the underlying bipartite graph is complete), that is not the case in the RPA problem. In addition, they define a feasible matching as an assignment of residents to hospitals satisfying the upper and lower quotas but possibly, leaving some residents unassigned. This is not valid for the RPA problem since no role can remain unassigned in a feasible matching. Additionally, the matching admits blocking pairs (i.e., two participants that are not partners in the matching and prefer each other to its assigned partner), and they proved that the problem of finding a matching with the minimum number of blocking pairs is NP-hard. Nevertheless, as later discussed, for CN purposes a matching with blocking pairs is not acceptable. It is preferable that the partners on their own initiative modify their requirements and/or preferences in order to obtain a comprehensive agreement on how to undertake the project and to ensure that none of them does feel (rightly or mistakenly) discriminated against.

Therefore, a new proposal is required to solve the RPA problem. This is because the RPA problem accepts upper and lower quotas in the partners' side, and incomplete preference lists with ties in both sides; in addition to fulfilling the properties and constraints defined for the RPA problem in Andrade-Garda et al. (2018) and summarized in Section 2.

4. Stability, pareto-efficiency and strategy-proofness

As stated above, the RPA problem is an example of the two-sided matching problem. Therefore, the three main properties of this type of matching problems—namely stability, Pareto-efficiency and strategy-proofness (Kesten, 2010)—have to be taking into account when defining the mechanism for constructing a matching M of roles to partners.

As indicated in Gale and Shapley (1962), a matching M is unstable if there is a pair *(a, b)* that are not partners in M and prefer each other to its partner in the matching. Such a pair is said to block, or to be a blocking pair for, the matching. Naturally, a matching for which there is no blocking pair is said to be *stable*. A mechanism is stable if it always produces a stable matching. This is the general definition of stability. However, if ties are allowed in the participants' preference lists, as is the case of the RPA problem, the following three different stability definitions are possible (Irving, Manlove, & Scott, 2003; Irving, 1994):

1. A matching will be called *weakly stable* unless there is a couple each of whom strictly prefers the other to its partner in the matching. It is not hard to see that if ties in preference lists are broken arbitrarily,

any matching that is stable in the resulting (strict) instance is weakly stable in the original instance.

2. A matching is *strongly stable* if there is no couple *(a, b)* such that *a* strictly prefers *b* to its partner, and *b* either strictly prefers *a* to its partner or is indifferent between them.
3. A matching is *super-stable* if there is no couple each of whom either strictly prefers the other to its partner or is indifferent between them.

A matching *M* is Pareto-efficient if there is no other matching for which all participants (i.e., roles/partners in this case) are at least as well off, and at least one participant better off. A mechanism is Pareto-efficient if it always produces a Pareto-efficient matching.

Finally, if a mechanism is strategy-proof, then no participant can benefit by lying, irrespective of its beliefs regarding the announcements of other participants.

Although these three properties (stability, Pareto-efficiency, and strategy-proofness) are important, unfortunately no mechanism exists to satisfy all of them. There are Pareto-efficient and strategy-proof (but not stable) mechanisms (Abdulkadiroglu & Sönmez, 2003), which are based on the top trading cycle algorithm (Shapley & Scarf, 1974); there are stable and strategy-proof (but not Pareto-efficient) mechanisms like the Gale–Shapley SOSM and—as indicated in Diebold et al. (2014)—Kesten (2010) demonstrates that there cannot exist a mechanism that always returns a Pareto optimal and stable matching (i.e., no stable and Pareto-efficient mechanism can exist).

Knowing this, it is necessary to determine which properties are essential in the context of the RPA problem in order to develop a matching scheme to satisfy them. Thus, as stated in Roth (2002), the empirical evidence is clear that stability is important to the success of matches in practice. Stable mechanisms have mostly (but not always) succeeded, and unstable mechanisms have mostly (but not always) failed. Moreover, as indicated in Abraham et al. (2007), and accordingly to Roth (1984), it has been convincingly agreed that, when preference lists exist on both sides, the key property that a matching constructed by a matching scheme should satisfy is that of stability. In fact, in the CN domain stability is an essential need: if partners cannot be confident that their priorities will not be violated, then the advantages of a central mechanism over making *ad hoc* arrangements disappear. Likewise, if the CN initiator cannot be confident that the role's priorities—that are objective priorities—will be met then it could not trust the mechanism. Thus, as stability is required, one of the three stability options has to be selected for the RPA problem. In the CN domain, the goal is to achieve a matching that makes it possible to address the business opportunity respecting the needs of each part (roles and partners). That is to say, avoiding strict preference violation. A *weakly stable* matching is, therefore, enough in the context presented in this paper and this is the type of stability that will be intended in the following. Thus, ties in preference lists will be broken via a random draw since any matching that is stable in the resulting strict instance is weakly stable in the original instance. Consequently, in a *weakly stable* matching *M* for an RPA instance there is no $r' \notin M(i)$ such that i strictly prefers r' to all roles in $M(i)$ and r' strictly prefers i to $M(r')$. At worst, i equally prefers the roles in $M(i)$ to any other role, and each role in $M(i)$ equally prefers i to any other partner.

Therefore, as mentioned above, if stability is required then Pareto-efficiency has to be given up. In this situation, the idea behind SOSM may provide the basis for developing the new algorithm required for solving the RPA problem. In doing so, it will be stable but not Pareto-efficient. Nonetheless, as Kesten (2010) states, the fact of caring about equity (stability) should not mean that the welfare (efficiency) aspects of the problem can be totally neglected. This raises a question about the price one needs to pay for achieving stability. In the School Choice problem, Kesten (2010) illustrates with an example a striking situation in which every student is unsatisfied at the most favorable stable matching (for students) that one can possibly find by the application of SOSM. In general, it is possible to arbitrarily construct problems for which a stable mechanism results in high welfare losses. Even more when, as in the case of the CN domain, weak stability is considered as sufficient and ties in preference lists are broken via some random draw.

In view of this, Kesten (2010) proposed the Efficiency Adjusted Deferred Acceptance Mechanism (EADAM). The central idea in EADAM is that of identifying interrupter students—those who interrupt desirable settlements among other students at no gain to themselves—and neutralizing their adverse effect on the outcome by asking them for consent to waive their priorities for crucial schools. Thus, EADAM closely mimics SOSM and makes adjustments to recover artificial welfare losses caused only by those interrupters who give consent to priority waiving. EADAM Pareto dominates SOSM; that is, no student is ever worse off under EADAM than for his/her assignment under SOSM and, when all students consent, the EADAM outcome is Pareto-efficient. With respect to stability, a student's priority for a particular school may be violated but this is the case only if he/she consents so it's no longer considered a priority violation in practice. Moreover, a consenting student is never hurt by consenting: no interrupter student ever gains anything by choosing not to consent. He/she can at best prevent others from improving. In addition, EADAM eliminates welfare losses due to randomly breaking ties in priorities. All of this is gained at the expense of strategy-proofness. However, trust telling of students is a Bayesian Nash equilibrium in this mechanism (Diebold et al., 2014). Even if there are no dominant strategies, possibilities to strategically misrepresent preferences are minimal in most applications. This is in line with the working philosophy in CNs, where the aim is not so much the individual but the collective success, since partners come together to address business opportunities that they could not undertake individually. The success of the project (through the success of the collaborative work) is the success of everybody, so strategic behavior is not the expected behavior. On the face of it, EADAM is taken as the basis for the definition of the new algorithm for solving the RPA problem.

5. Proposed centralized matching scheme

5.1. Preconditions

For the RPA problem, in summary, the following is true: (i) there are objective preferences over the partners most appropriate to play each role (e.g., on the basis of its knowledge), (ii) each partner has also preferences over the roles that it is willing to perform (on the basis of its knowledge, individual tastes, and/or personal preferences), (iii) preference lists are incomplete and partially ordered (with ties) on both sides (partners and roles), (iv) ties in preference lists are arbitrary broken since a weakly stable matching is enough, (v) there are upper bounds on the number of roles that can be assigned to a particular partner, and (vi) there are also lower bounds on the number of roles that a given partner is willing to perform.

Before presenting the algorithm, let us suppose that a matching is, at least in theory, possible. That is to say, in addition to the constraints previously presented, (i) there is no role that is unacceptable for all partners, and (ii) each partner i is acceptable for at least $l(i)$ roles. Otherwise a matching is not possible. Also, in order to minimize the number of useless computations, it is recommended to prune preference lists if necessary to achieve consistency. That is, if partner i is not acceptable for role r then r must not be in i's preference list and vice versa, since no matching will pair i and r. Analogously, if partner i doesn't want to play role r then i must not be in r's preference list.

5.2. Proposing side

As stated above, for the definition of this algorithm, EADAM is used as a reference. It is also well known that EADAM applies SOSM at its heart, that is the generalization of the Shapley-Gale student proposing DA algorithm to solve the College Admission Problem. In the College Admission problem, each student is assigned to one college and each college can receive many students. Making an analogy between this problem and the RPA problem, each role corresponds to a student and each partner corresponds to a college since each role is assigned to one partner and each partner can receive many roles (i.e., pieces of project). In this regard, the Shapley-Gale algorithm (Gale & Shapley, 1962) showed that when students (colleges) are proposing, the resulting matching M is student (college) optimal in the sense that every student (college) is at least as well off under M as it is under any other stable matching. Unfortunately, the student (college) optimal matching is college (student) pessimal in the sense that no college (student) is worse off in any other stable matching. Consequently, if the student optimal solution is the same as the college optimal solution, the solution is unique.

In referring to which side must be the proposing one in the RPA problem, as stated above, in the CN domain partners create alliances to face business opportunities that would not be possible or would have a higher cost if attempted by them individually. In this context, the success of the joint project is the success for all partners and the best way to ensure the success of the project is obtaining a matching in which each piece of project (i.e., role) is performed by the most appropriate worker (i.e., partner). Thus, it is expected that a role optimal matching might be the best option since it is the best matching for roles and hence for the project. For this reason, a role proposing algorithm is presented by default. However, if a partner optimal solution is required it would be enough to use the college optimal stable matching algorithm (instead of SOSM) as the core of EADAM and to ask partners instead of roles to consent for recovering welfare losses.

In the role proposing algorithm, interrupter roles are detected and asked for consent to waive their priorities for certain partners. In this case, it is assumed that all interrupter roles consent. A non-consenting role can only harm other roles preventing them from being performed by a better partner, which is not a collaborative work philosophy and makes no sense when looking for the best for the project. Also, note that roles are not human beings so role consenting actually means that the partners consent to seek the best for the project and not for themselves as individuals.

5.3. Algorithm

The first step of the matching process may be informally expressed in terms of a sequence of proposals from the roles to the partners. In brief, whilst a role r that is unmatched exists, r makes a proposal to each partner i on its preference list until it becomes provisionally assigned to a partner or its preference list has been exhausted. Partners only accept proposals if they are currently undersubscribed or if they are fully subscribed and strictly prefer the proposing role to their least favored assignee.

Note that what has been indicated up to now is the direct execution of SOSM. However, remember that the new algorithm must ensure that no partner i remains without playing $l(i)$ roles (lower bound) at the end of the allocation process. For example, consider a problem with three roles (r_0, r_1, r_2), and two partners (i_0, i_1), with $l(i) = 1$ and $u(i) = 3$ for all partners. In this case, if 3 roles are assigned to i_0 (or i_1) then i_1 (or i_0) will remain totally unsubscribed, which would not be acceptable. In order to avoid this type of situations, the proposal in this paper is to reduce the partners' upper quotas following certain criteria. This is to ensure that all partners meet their participation requirements by finding a matching in which each partner i plays at least $l(i)$ roles. Remember that the meaning of $l(i)$ and $u(i)$ is that the aim of i is to play at least $l(i)$ roles (it meets its needs) and by no means will accept to play more than $u(i)$ roles. Thus, reducing $u(i)$ does not harm the interests of partner i as long as $l(i)$ is guaranteed. If such a matching doesn't exist, partners must be asked in order to modify their preference lists and/or upper and lower bounds if they want to try to form an alliance to undertake the business opportunity. If they agree, the algorithm must be applied again. Otherwise, it would not be possible to reach an agreement in order to take advantage of the business opportunity.

Once a matching satisfying all lower quotas is obtained, EADAM extension to recover welfare losses (due both to SOSM and to random tie breaking in preference lists) is applied. This is necessary because, as previously indicated, when roles are proposing, the resulting matching M is role optimal in the sense that every role is at least as well off under M as it is under any other stable matching. However, as indicated in (Kesten, 2010) it is possible to arbitrarily construct problems for which a stable mechanism results in high welfare losses, in which every role is unsatisfied at the most favorable stable matching for roles that one can possibly find by the application of SOSM. That is to say, a matching where roles are played by partners in low or very low positions on their preference lists, which are not the most appropriate partners since the preference list of each role is created using objective criteria (i.e., the partner with the most knowledge or capabilities for the role will be on the top of the list). Such practices could have detrimental effects for the success of the project since, as previously indicated, the best for the project would be for roles to be played by the best, or next best, partners. Therefore, the welfare of the roles must not be neglected. Applying EADAM for recovering the welfare losses of the roles provides this important benefit in exchange for asking roles to waive their priorities for certain partners (thus possibly losing stability). However, no consenting role is ever worse off under EADAM than it is for its assignment under SOSM. That is, no piece of a project will be played by a less appropriate partner, so roles, and therefore the project, are not negatively affected in practice by consent.

Thus, interrupting roles are asked to consent and, as stated above, all of them do, so the outcome is Pareto-efficient. The new algorithm can be viewed therefore as an upper bound reduction mechanism (UBReM in the following) that applies EADAM at its heart. UBReM is synthesized in Fig. 1, and operates as follows with an RPA problem instance A:

1. Run SOSM to obtain a matching M. This matching must involve all roles, since it is not possible to undertake a project in which there are project pieces that will not be carried out by anyone. Thus, to continue with the next step of UBReM, no role can be unassigned in M. Otherwise, a stable matching for the RPA problem instance A is not possible even when applying the rest of the steps in the procedure. This directly derives from the condition $\sum_{i \in I} u(i) \geq |R|$ above presented in Section 2. In other words, partners have capacity for at least all roles ($|R|$ roles), but at least one role r is unassigned in the resulting matching M. When this happens after applying SOSM, this means that no partner in a stable matching can play this role. If r is "unplayable" with the original upper bounds, it will continue to be "unplayable" with smaller upper bounds, as preference lists do not change.
2. If all roles are matched but M doesn't satisfy all lower quotas, then obtain A' by reducing upper quotas. The aim is to balance the assignation of roles to partners respecting priorities and quotas. For this:
    2.1. Considering that no partner by no means will play more than $u(i)$ roles, the partner with the highest "margin" is the partner with the highest value of $u(i) - l(i)$ (henceforth referred to as *gap*). Thus, obtain the partner i with the highest *gap* and set $u(i) = u(i) - 1$.
    2.2. If there is more than one partner with the same (maximum) *gap*, then obtain the "less preferred" one among them and set $u(i) = u(i) - 1$. Partner i is preferred over partner j if it is a better option for more roles. If a partner is highly preferred, then it is better for the project not to reduce its upper quota in order to make it possible for it to obtain more roles. Even knowing this, note that in step 2.1 the upper bound of a highly preferred partner could be reduced because of its *gap*. This is done to prevent most preferred partners to "pre-empt the market" and generate highly imbalanced outcomes harming the alliance interests.

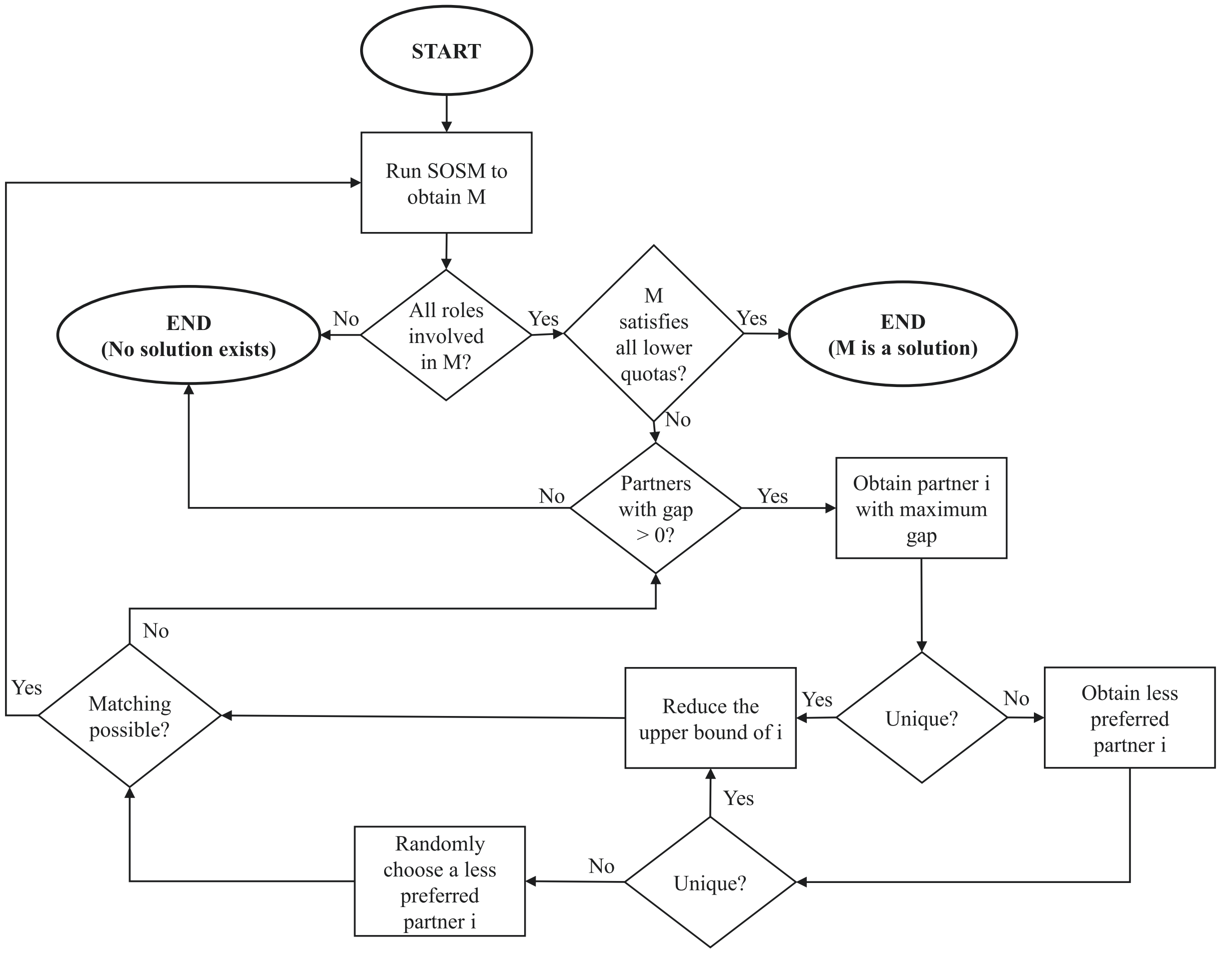

Fig. 1. Flowchart for UBReM algorithm.

2.3. If several partners are equally less preferred, then randomly choose a partner i among them and set $u(i) = u(i) - 1$.

3. Rerun SOSM over the new RPA problem instance *A'*. Prior to performing this step, it is necessary to check if *A'* satisfies the minimal conditions for an existing matching. These are: (i) $|R| \geq \sum_{i \in I} l(i)$, (ii) $\sum_{i \in I} u(i) \geq |R|$, (iii) there is no role that is unacceptable for all partners, and (iv) each partner i is acceptable for at least $l(i)$ roles. As mentioned above, this does not guarantee that a solution to the RPA problem exists, but it makes it possible (i.e., a necessary but not sufficient condition).

4. The process is repeated until: (i) M satisfies all lower quotas, or (ii) M doesn't satisfy all lower quotas and there is no partner with actual *gap* greater than 0. In the first case, EADAM extension to recover welfare losses must be applied if there are interrupter roles. In the second case, a stable matching satisfying lower and upper quotas doesn't exist and the process finishes.

Note that in steps 2.1 to 2.3, if $|M(i)| \leq u(i)$ being i the partner reducing upper quota, then running SOSM again will not produce a different outcome. This is because each partner is assigned the same number of roles in any stable matching (Gale & Sotomayor, 1985) and, therefore, any partner that is under-subscribed in one stable matching is assigned precisely the same set of roles in all stable matching (Roth, 1986). Thus, for example, let A be an RPA instance with two partners i_0 and i_1 ($l(i_0) = 1, l(i_1) = 2, u(i_0) = u(i_1) = 3$) and three roles, with $|M(i_0)| = 2$ and $|M(i_1)| = 1$ after step 1. The partner with the highest *gap* is i_0 so it reduces the upper bound in one unit. Running again SOSM with the new upper bounds will produce exactly the same solution. Thus, steps 2.1 to 2.3 have to be repeated until new upper bounds are reached making it possible to obtain a different matching (i.e., $|M(i)|\rangle u(i)$ for partner i reducing $u(i)$). This way, new role allocations are possible only when partners reducing their upper bounds lose their less preferable assigned roles that are now free to be assigned to other partners. Also, note that partner i reducing its upper quota continue to be guaranteed to receive at least $l(i)$ roles. This is because preference lists do not change, so if it is a preferable partner for at least $l(i)$ roles it will continue to be so.

In order to clarify the proposed algorithm, the data structures and the main functions defined for UBReM are presented below in pseudocode (Pseudocode 1 to 3) followed by their explanation.

Data Structures

```
partner: record
  id: integer
  l: integer
  u: integer
  gap: integer
  beta: list
  M: list
role: record
  id: integer
  alpha: list
  M: integer
partners: list of n elements of type partner
roles: list of m elements of type role
```

Pseudocode 1 UBReM (Main procedure)

```
1    Input: partners
2    Output: partners with upper bound reduced
```

(*continued on next page*)

(*continued*)

```
3    Variables /*counting variables are omitted*/
4      stop /*true if new upper bounds allowing a new matching are reached*/
5      mgp /*subset of partners with the maximum gap value*/
6      lpp /*subset of partners with the less preferred partners*/
7    Begin procedure
8      stop = false
     /*Partners not exceeding l assigned roles will not reduce upper bound, gap
   must be 0 instead of u-l*/
9      for i=0 to (size of partners – 1) do
10       if size of (partners[i].M) ≤ partners[i].l then
11         partners[i].gap = 0
12   while stop is false do
13     mgp = partners with maximum gap /*pseudocode omitted for simplicity*/
14     if size of mgp = 1 then
15       stop = reduce(partners[mgp[0]].id)
16     else
17       lpp = lessPreferredPartners(mgp)
18       if size of lpp = 1 then
19         stop = reduce(partners[lpp[0]].id)
20       else
21         stop = reduce(randomly choose a partner in lpp)
22   End Procedure
```

Pseudocode 2 reduce

```
1    Input: p /*the id of the partner reducing upper bound*/
2    Output: false if it is needed to continue reducing upper bounds
       true otherwise
3    Begin procedure
4    partners[p].u = partners[p]-u - 1
5      if partners[p].u < size of (partners[p].M) then
6        return true
7      else
8        return false
9    End procedure
```

Pseudocode 3 lessPreferredPartners

```
1    Input: mgp /*list of partners satisfying lower quotas with maximum gap*/
2    Output: the list of less preferred partners
3    Variables /*counting and control variables are omitted*/
4    v /*partners at position x in the alpha list of each role*/
5      oc /*occurrences of a list item*/
6      min /*the elements of the list with minimum value*/
7      lpp /*list of less preferred partners*/
8    Begin procedure
9      stop = false
10     i = 0
11     while stop is false and i < size of partners do
12       k = 0
13       lpp = empty list
14       for j=0 to (size of roles - 1) do
15         if i < size of (roles[j].alpha) then
             /*preference lists are incomplete*/
16           v[k] = roles[j].alpha[i]
17           k = k + 1
18     for j=0 to (size of mgp – 1) do
19   oc[j] = occurrences(mgp[j].v) /*pseudocode omitted for simplicity*/
20   min = minValue(oc)/*pseudocode omitted for simplicity*/
21   t = 0
22   for j=0 to (size of oc – 1) do
23   if oc[j] = min then
24       lpp[t] = mgp[j]
25       t = t + 1
26   if size of lpp = 1 then
27       stop = true
28   else
29     i = i + 1
30   return lpp
31   End procedure
```

The data structures for partners and roles are defined as records with a series of elements. In the case of partners, these elements are the *id* (e.g., 0 to represent i_0), the value of l (i.e., the lower quota), the value of u (i.e., the upper quota), the value of *gap* (i.e.,$u - l$), the preference list *beta* of roles' *ids*, and the list M with the *ids* of the roles assigned to the partner in the final matching. Likewise, the data structure for roles is composed of *id*, the preference list *alpha* of partners' *ids*, and M as the *id* of partner assigned to the role in the final matching. Thus, the final matching M is represented by the two elements M in the previous records. Finally, a list of n elements of type partner (namely *partners*) and a list of m elements of type role (namely *roles*) are defined to represent the RPA problem participants. Note that *beta* and *alpha* are represented as strict preference lists because ties are broken arbitrary since weak stability is enough in the CN domain.

Regarding the functions, after running SOSM (step 1 of UBReM), if a matching involving all roles is obtained but lower quotas are not satisfied, the UBReM procedure starts (see Pseudocode 1). It represents the new functionality proposed in this paper for solving the RPA problem. Upper quota reduction is needed, so the partners with the highest *gap* have to be obtained (step 2.1 of UBReM). If there is only one partner with the highest *gap,* then this is the partner reducing the upper quota, which is done through the function *reduce* (see Pseudocode 2). Otherwise, the less preferred partner among the partners with maximum *gap* have to be calculated (step 2.2 of UBReM). This is done by the function *lessPreferredPartners* (see Pseudocode 3). Again, if there is only one less preferred partner then this is the one reducing the upper quota. Otherwise, as indicated at the end of the UBReM main procedure, the partner reducing the upper quota is randomly chosen from the list of less preferred partners (step 2.3 of UBReM). After upper bound reduction, if the new instance of the RPA problem satisfies the minimal conditions for an existing matching (pseudocode omitted for simplicity), the process is repeated (step 3 of UBReM). This continues until a matching satisfying all lower bounds is obtained. After that, the EADAM extension to recover welfare loses must be applied as indicated in (Kesten, 2010). Otherwise, a matching satisfying all lower quotas is not possible.

Regarding the complexity of UBReM, it firstly consists of a running of SOSM, which is a polynomial procedure (see e.g., Gusfield & Irving, 1989). After that, the *UBReM* procedure is applied. As may be deduced from the previous pseudocodes presented, in the worst of cases it runs in quadratic time. Finally, the EADAM extension to recover welfare losses is applied, that is a computationally simple polynomial-time algorithm (Kesten, 2010). Therefore, the computational complexity of the entire matching mechanism continues to be polynomial (i.e., UBReM runs in polynomial time).

6. Illustrative example

In order to illustrate UBReM, a representative example is presented below. Mathematical instead of algorithmic notation will be used to make the example more readable.

Example. *Consider an instance of the RPA problem with twelve roles* $R = \{r_0, r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9, r_{10}, r_{11}\}$, *and five partners* $I = \{i_0, i_1, i_2, i_3, i_4\}$, *with* $l(i) = 2$ *for all partners,* $u(i_0) = u(i_2) = 4$, *and* $u(i_1) = u(i_3) = u(i_4) = 5$. *This is an adaptation of the Example 2 used by Kesten (2010) to show the inefficiency of SOSM. The preference lists (*$\alpha(r)$ *for role* r*, and* $\beta(i)$ *for partner* i*) are as indicated in Tables 1 and 2.*

In this case, i_0 is indifferent between r_9 and r_7; i_1 is indifferent between r_1 and r_9; r_2 is indifferent between i_3 and i_4; and r_0 is indifferent between i_0 and i_1. Random tie breaking produces the preference lists $\alpha(r)$ and $\beta(i)$ in Tables 1 and 2. The following tables illustrate the steps of SOSM (step 1 of UBReM) applied to this problem. The columns of the tables represent the partners, and the rows represent the steps of the algorithm. Any role tentatively placed at a partner at a particular step is shown underlined at the corresponding entry of corresponding table. Table 3 shows the first matching obtained from SOSM in a single step.

The resulting matching doesn't satisfy lower quotas for i_4. Let $s(i)$ be the *gap* of partner i. Thus, $s(i_0) = s(i_2) = 2$ and $s(i_1) = s(i_3) = 3$. Partner i_4 is the deficit partner, so its upper quota is not reduced (it would be

Table 1
Preference lists for partners.

$\alpha(r_0)$	$\alpha(r_1)$	$\alpha(r_2)$	$\alpha(r_3)$	$\alpha(r_4)$	$\alpha(r_5)$	$\alpha(r_6)$	$\alpha(r_7)$	$\alpha(r_8)$	$\alpha(r_9)$	$\alpha(r_{10})$	$\alpha(r_{11})$
i_0	i_0	i_0	i_2	i_1	i_1	i_2	i_2	i_3	i_0	i_3	i_3
i_1	i_2	i_1	i_3	i_2	i_2	i_1	i_1	i_1	i_2	i_2	i_0
i_2	i_3	i_3	i_4	i_0	i_4	i_0	i_0	i_4	i_1	i_4	i_4
i_4	i_1	i_4				i_4	i_4				

Table 2
Preference lists for roles.

$\beta(i_0)$	$\beta(i_1)$	$\beta(i_2)$	$\beta(i_3)$	$\beta(i_4)$
r_4	r_1	r_{10}	r_3	r_{11}
r_6	r_9	r_0	r_2	r_8
r_{11}	r_8	r_5	r_1	r_5
r_7	r_6	r_9	r_8	r_2
r_9	r_2	r_4	r_{10}	r_0
r_0	r_0	r_3	r_{11}	r_3
r_1	r_7	r_1		r_7
r_2	r_4	r_6		r_{10}
	r_5	r_7		r_6

Table 3
First execution of SOSM.

Step	i_0	i_1	i_2	i_3	i_4
0	$\underline{r_0}, \underline{r_1}, \underline{r_2}, \underline{r_9}$	$\underline{r_4}, \underline{r_5}$	$\underline{r_3}, \underline{r_6}, \underline{r_7},$	$\underline{r_8}, \underline{r_{10}}, \underline{r_{11}}$	

futile since partner i_4 will never obtain more than $l(i_4)$ roles in any stable matching). Both i_1 and i_3 are the partners with the highest *gap*, but i_3 is the preferred partner (it is the first option for more roles) so that $u(i_1) = u(i_1) - 1 = 4$. Before rerunning SOSM with the new upper bounds the possible production of a different outcome must be checked. In this case, $|M(i_1)| = 4 \geq u(i_1)$ so that the resulting matching will not change. That is, it is possible for i_1 to obtain the same roles again—the same as to i_3 because its upper bound has not been reduced—and, as a consequence, i_4 will remain unmatched. Therefore, the quota reduction process has to be repeated. This will continue until upper bounds allowing a new matching are reached, going through the following steps:

1. i_3 reduces its upper bound as the partner with the highest *gap*.
2. The new *gap* of all partners is 2. i_1 reduces its upper bound as the less preferred partner.
3. Randomly choose i_3 to reduce the upper bound (i_2 and i_3 are equally less preferred partners).
4. i_2 reduces its upper bound as the less preferred partner with the highest *gap*.
5. i_0 reduces its upper bound as the partner with the highest *gap*.

The upper bounds $u(i_0) = u(i_1) = u(i_2) = u(i_3) = 3$ are achieved, which allows to generate a different matching. This is because $|M(i_0)| = 4 > u(i_0) = 3$ so that i_0 will lose its less preferred assigned role. The new matching obtained by rerunning SOSM is $M = \{(i_0, r_0), (i_0, r_1), (i_0, r_9), (i_1, r_4), (i_1, r_5), (i_2, r_2), (i_2, r_3), (i_2, r_6), (i_2, r_7), (i_3, r_8), (i_3, r_{10}\}, (i_3, r_{11})\}$.

Again, M doesn't satisfy lower quotas for i_4 as it remains with 0 assigned roles. The application of the upper bounds reduction process results in $u(i_1) = u(i_1) - 1 = 2$ and a new matching $M = \{(i_0, r_4), (i_0, r_6), (i_0, r_{11}), (i_1, r_9), (i_1, r_8), (i_2, r_{10}), (i_2, r_0), (i_2, r_5), (i_3, r_3), (i_3, r_2), (i_3, r_1\}, (i_4, r_7)\}$ is obtained by rerunning SOSM. In this example it would have been possible to know without running SOSM again that the new matching was not going to satisfy lower quotas for i_4. This is because the quota reduction process frees only one role to be reassigned (being assigned to i_4 in the best case) but i_4 needs two roles to satisfy its lower quota. Therefore, without changes in the preference lists, the selected partners have to reduce their upper bounds in such quantities that the deficit partners can achieve their minimums due to the free roles reallocation.

Randomly choosing i_3 to reduce the upper bound (i_2 and i_3 are now equally less preferred partners with highest *gap*) the solution $M = \{(i_0, r_4), (i_0, r_6), (i_0, r_{11}), (i_1, r_9), (i_1, r_8), (i_2, r_{10}), (i_2, r_0), (i_3, r_3), (i_3, r_2), (i_3, r_1\}, (i_4, r_7), (i_4, r_5)\}$ shown in Table 4 arises.

The resulting matching is a stable matching satisfying all lower quotas. It is stable because it is the result of running SOSM, which returns a stable matching. Lower quotas are satisfied thanks to the upper bound reduction process presented in UBReM. That is, if after running SOSM the resulting matching doesn't satisfy all lower quotas, a new instance *A'* of the RPA problem is obtained by reducing upper quotas (step 2 of UBReM) and SOSM is applied again with *A'* (step 3 of UBReM). Only when all lower quotas are satisfied will the corresponding matching be accepted. Otherwise, UBReM returns that no matching exists satisfying the RPA problem instance requirements.

In the resulting matching the preferred partners (i.e., i_0 and i_3) receive an extra role ($l(i) + 1$). However, note that i_0 is the most preferable partner. Thus, if i_0 doesn't reduce its upper bound (by eliminating the *gap* checking from UBReM, and then with $u(i_0) = 4$ and $u(i_1) = u(i_2) = u(i_3) = 2$) it will obtain four roles (i.e., $u(i_0)$) and the solution $M = \{(i_0, r_4), (i_0, r_6), (i_0, r_{11}), (i_0, r_7), (i_1, r_9), (i_1, r_1), (i_2, r_{10}), (i_2, r_0), (i_3, r_3), (i_3, r_2), (i_4, r_8), (i_4, r_5)\}$ arises.

In this case, r_7 gets better off, but by making r_1 and r_8 worse off. Thus, this alternative solution doesn't Pareto-dominate the first one. Moreover, and also very important, it floods partner i_0 by assigning to it the maximum possible number of partners (presumably $u(i_0)$) since it always maintains the upper bound, and leaves the rest of partners with few roles (presumably $l(i)$). It may be counter-productive to reach an agreement on creating the alliance and undertaking the business opportunity since in the context of CNs the “common good” is the achievement of a solution in which all partners get a number of roles between $l(i)$ and $u(i)$ in an equitable distribution. Knowing that highly imbalanced outcomes can arise from the centralized allocation mechanism could predispose partners to not participate the process. The *gap* criterion has the intention of preventing such situations.

The stable matching M in Table 4 is the solution to the RPA problem in the example. However, it is clearly Pareto-inefficient since it places each role at either its last choice or its next to last choice. In a role-proposing algorithm this should not occur. It is easy to see that in this case there are interrupter roles, so the application of the EADAM extension to recover welfare losses is required. Thus, the last

Table 4
Fourth execution of SOSM.

Step	i_0	i_1	i_2	i_3	i_4
0	$\underline{r_0}, \underline{r_1}, r_2, \underline{r_9}$	$\underline{r_4}, \underline{r_5}$	$\underline{r_3}, \underline{r_6}, \underline{r_7}$	$\underline{r_8}, \underline{r_{10}}, \underline{r_{11}}$	
1	$\underline{r_0}, \underline{r_1}, \underline{r_9}$	$r_5, r_4, \underline{r_2}, \underline{r_7}$	$\underline{r_3}, \underline{r_6}$	$\underline{r_8}, \underline{r_{10}}, \underline{r_{11}}$	
2	$\underline{r_0}, \underline{r_1}, \underline{r_9}$	$\underline{r_2}, \underline{r_7}$	$r_3, r_6, \underline{r_5}, \underline{r_4}$	$\underline{r_8}, \underline{r_{10}}, \underline{r_{11}}$	
3	$\underline{r_0}, \underline{r_1}, \underline{r_9}$	$\underline{r_6}, \underline{r_2}, r_7$	$\underline{r_4}, \underline{r_5}$	$\underline{r_8}, \underline{r_{10}}, r_{11}, \underline{r_3}$	
4	$r_0, r_1, \underline{r_9}, \underline{r_7}, \underline{r_{11}}$	$\underline{r_6}, \underline{r_2}$	$\underline{r_4}, \underline{r_5}$	$\underline{r_8}, \underline{r_{10}}, \underline{r_3}$	
5	$\underline{r_9}, \underline{r_7}, \underline{r_{11}}$	$\underline{r_6}, \underline{r_2}$	$r_4, \underline{r_5}, \underline{r_0}$	$\underline{r_8}, r_{10}, \underline{r_3}, \underline{r_1}$	
6	$r_9, \underline{r_7}, \underline{r_{11}}, \underline{r_4}$	$\underline{r_6}, \underline{r_2}$	$r_5, \underline{r_0}, \underline{r_{10}}$	$\underline{r_8}, \underline{r_3}, \underline{r_1}$	
7	$\underline{r_7}, \underline{r_{11}}, \underline{r_4}$	$\underline{r_6}, \underline{r_9}, r_2$	$\underline{r_0}, \underline{r_{10}}$	$\underline{r_8}, \underline{r_3}, \underline{r_1}$	$\underline{r_5}$
8	$\underline{r_7}, \underline{r_{11}}, \underline{r_4}$	$\underline{r_9}, \underline{r_6}$	$\underline{r_0}, \underline{r_{10}}$	$r_8, \underline{r_3}, \underline{r_1}, \underline{r_2}$	$\underline{r_5}$
9	$\underline{r_7}, \underline{r_{11}}, \underline{r_4}$	$\underline{r_9}, r_6, \underline{r_8}$	$\underline{r_0}, \underline{r_{10}}$	$\underline{r_3}, \underline{r_1}, \underline{r_2}$	$\underline{r_5}$
10	$\underline{r_7}, \underline{r_{11}}, \underline{r_4}$	$\underline{r_9}, \underline{r_8}$	$\underline{r_0}, \underline{r_{10}}$	$\underline{r_3}, \underline{r_1}, \underline{r_2}$	$\underline{r_5}$
11	$r_7, \underline{r_{11}}, \underline{r_4}, \underline{r_6}$	$\underline{r_9}, \underline{r_8}$	$\underline{r_0}, \underline{r_{10}}$	$\underline{r_3}, \underline{r_1}, \underline{r_2}$	$\underline{r_5}$
12	$\underline{r_{11}}, \underline{r_4}, \underline{r_6}$	$\underline{r_9}, \underline{r_8}$	$\underline{r_0}, \underline{r_{10}}$	$\underline{r_3}, \underline{r_1}, \underline{r_2}$	$\underline{r_5}, \underline{r_7}$

interrupting pair is (r_7, i_0) at step 11. This is because role r_7 is tentatively placed at i_0 at step 4 and rejected from it at step 11. Role r_9 is rejected by i_0 while role r_7 is tentatively placed at it. Assuming that r_7 is a consenting role, partner i_0 is removed from the preference list of role r_7 and, in order to maintain consistency in preference lists, role r_7 is removed from the preference list of i_0.

The matching obtained after the application of SOSM with the new preference lists is $M' = \{(i_0, r_1), (i_0, r_9), (i_0, r_4), (i_1, r_2), (i_1, r_6), (i_2, r_{10}), (i_2, r_0), (i_3, r_8), (i_3, r_{11}), (i_3, r_3\}, (i_4, r_7), (i_4, r_5)\}$, where the consenting role r_7 continues to be assigned to i_4 (it is not adversely affected) while r_1, r_2, r_6, r_8, r_9, and r_{11} get better off. The full execution of the EADAM extension results in the matching (marked in bold in Tables 5 and 6) $M' = \{(i_0, r_1), (i_0, r_9), (i_0, r_0), (i_1, r_2), (i_1, r_4), (i_2, r_3), (i_2, r_6), (i_3, r_8), (i_3, r_{10}), (i_3, r_{11}), (i_4, r_7), (i_4, r_5)\}$ that clearly Pareto-dominates M (underlined in Tables 5 and 6). M' places each role at either its first choice or its next to first choice. Only r_5 and r_7 continue to be assigned to i_4 (their last choice). Thus, the preference lists of these two roles were preventing other roles to be assigned to their first or its next to first choice by no gain for r_5 and r_7. The solution M' is not stable from a theoretical point of view, but all blocking pairs involve consenting roles and no role is adversely affected, so it is no longer considered a priority violation in practice.

7. Conclusion and future research

This paper has proposed a computationally polynomial-time matching algorithm called UBReM to solve the RPA problem, which is an example of the two-sided matching problem. This is an inherent problem in CNs, since it addresses the distribution of roles among the partners in a CN when they want to start a new project together. Due to the specific characteristics of the RPA problem, it cannot be directly solved by the existing matching mechanisms used to solve other examples of the two-sided matching problem.

To face these specific requirements, UBReM manages lower and upper quotas on the partners' side correctly, finding for a stable matching outcome satisfying all quotas, if it exists. To do so, it is assumed that when a partner sets a lower quota it means that it needs to play at least this amount of roles, so a matching that doesn't satisfy that requirement is not valid. Likewise, when a partner specifies an upper quota it doesn't mean that it wants to play this amount of roles, but it represents that by no means will it accept to play more than this amount of roles. This way, a solution satisfying lower quotas in which each partner receives a number of roles less than or equal to its upper quota is an acceptable solution. Knowing this, UBReM starts running SOSM to obtain a role-optimal outcome (i.e., roles are proposing to partners) and, if the outcome doesn't satisfy lower quotas, an upper quota reduction process begins for those partners exceeding lower quotas. It doesn't damage partners as lower quotas are not changed; it only implies that some partners will not receive as many roles as if lower quotas did not have to be satisfied for all partners, but still will receive a number of roles greater than or equal to their lower quota.

Once a solution satisfying all lower quotas is obtained, and knowing that it can be highly inefficient, the EADAM extension to recover welfare losses is applied assuming that all interrupter roles consent. It also allows to recover welfare losses due to random tie breaking in preference lists, as preference lists in the RPA problem are incomplete and partially ordered (with ties). Thus, UBReM is a Pareto-efficient mechanism in which a role's priority for a particular partner may be violated (i.e., the mechanism is theoretically non stable), but this is the case only if roles consent. It is known that no role is ever worse off under EADAM than for its assignment under SOSM, so it is assumed that roles consent and it's no longer considered a priority violation in practice.

Finally, the mechanism is not strategy-proof, but strategic behavior is not the expected behavior in the RPA context, that is the CN domain.

Therefore, UBReM satisfies the requirements of the RPA problem and can be used to solve that problem in CNs.

Thus, as EADAM closely mimics SOSM and makes adjustments to recover artificial welfare losses, UBReM closely mimics SOSM and makes adjustments to find a matching satisfying lower and upper quotas.

The innovative idea of addressing role allocation in CNs by means of a centralized matching scheme will simplify the difficult and time consuming task of making *ad hoc* arrangements by partners approaching one another directly, thus minimizing the risk of losing the business opportunity.

The authors are also working on how to address the case in which the UBReM outcome is that a matching satisfying the RPA problem requirements doesn't exist. The aforementioned would lead to think that the partners in the CN are not able to play the roles defined for the new project that they want to undertake together (e.g., due to lack of knowledge). However, this might not necessarily be the case. The trouble may lie in the way in which the CN initiator defines the set of roles to be performed by the partners. If the characteristics and constraints of the partners are not taken into account when defining those roles, the business opportunity might be unnecessarily lost (e.g., there is no partner that satisfies all the requirements of a role, but by splitting the role into sub-roles there is at least one partner able to perform each sub-role). For this reason, an approach (possibly in the form of a decision support system) is being researched to help the CN initiator to configure the roles of the project that the partners in the CN want to undertake together. Alternatively, if the UBReM outcome is that a matching doesn't exist it could be because the partners are not able to play the defined roles in a many-to-one allocation (e.g., no partner alone has the required knowledge to play a given role). However, roles could be properly performed in a many-to-many allocation. That is to say, allowing partners to join efforts to play the roles instead of requiring roles to be performed only by individual partners. Thus, the definition and resolution of the many-to-many generalization of the RPA problem is something the authors are currently researching.

Finally, the authors are also working on the next step required to allow partners in the CN to take advantage of a business opportunity. That is, once a matching of roles to partners satisfying the RPA problem requirements is obtained, the new project that partners will undertake

Table 6
UBReM resulting matching of roles to partners.

$\beta(i_0)$	$\beta(i_1)$	$\beta(i_2)$	$\beta(i_3)$	$\beta(i_4)$
$\underline{r_4}$	r_1	$\underline{r_{10}}$	$\underline{r_3}$	r_{11}
$\underline{r_6}$	$\underline{r_9}$	$\underline{r_0}$	$\underline{r_2}$	r_8
$\underline{r_{11}}$	$\underline{r_8}$	r_5	$\underline{r_1}$	$\underline{\boldsymbol{r_5}}$
r_7	r_6	r_9	$\boldsymbol{r_8}$	r_2
$\boldsymbol{r_9}$	$\boldsymbol{r_2}$	r_4	$\boldsymbol{r_{10}}$	r_0
$\boldsymbol{r_0}$	r_0	$\boldsymbol{r_3}$	$\boldsymbol{r_{11}}$	r_3
$\boldsymbol{r_1}$	r_7	r_1		$\underline{\boldsymbol{r_7}}$
r_2	$\boldsymbol{r_4}$	$\boldsymbol{r_6}$		r_{10}
	r_5	r_7		r_6

Table 5
UBReM resulting matching of partners to roles.

$\alpha(r_0)$	$\alpha(r_1)$	$\alpha(r_2)$	$\alpha(r_3)$	$\alpha(r_4)$	$\alpha(r_5)$	$\alpha(r_6)$	$\alpha(r_7)$	$\alpha(r_8)$	$\alpha(r_9)$	$\alpha(r_{10})$	$\alpha(r_{11})$
$\boldsymbol{i_0}$	$\boldsymbol{i_0}$	i_0	$\boldsymbol{i_2}$	$\boldsymbol{i_1}$	i_1	$\boldsymbol{i_2}$	i_2	$\boldsymbol{i_3}$	$\boldsymbol{i_0}$	$\boldsymbol{i_3}$	$\boldsymbol{i_3}$
i_1	i_2	$\boldsymbol{i_1}$	$\underline{i_3}$	i_2	i_2	i_1	i_1	$\underline{i_1}$	i_2	$\underline{i_2}$	$\underline{i_0}$
$\underline{i_2}$	$\underline{i_3}$	$\underline{i_3}$	i_4	$\underline{i_0}$	$\underline{\boldsymbol{i_4}}$	$\underline{i_0}$	i_0	i_4	$\underline{i_1}$	i_4	i_4
i_4	i_1	i_4				i_4	$\underline{\boldsymbol{i_4}}$				

together must be planned. This means first to create a project schedule. In this case, the aim would mainly be to minimize the project time subject to certain restrictions (task dependences, milestones, partners' time availability, market requirements, etc.). For that purpose, various approaches such as branch and bound methods (see e.g., Ross & Soland, 1975; Brucker, Jurisch, & Sievers, 1994) and argumentation techniques (see e.g., Rahwan, Ramchurn, & Jennings, 2004) are being considered.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Abdulkadiroglu, A., & Sönmez, T. (2003). School choice: A mechanism design approach. *American Economic Review, 93*(3), 729–747. https://doi.org/10.7916/D8WS95FB

Abraham, D. J., Irving, R. W., & Manlove, D. F. (2003). The student-project allocation problem. *Lecture Notes in Computer Science, 2906*, 474–484. https://doi.org/10.1007/978-3-540-24587-2_49

Abraham, D. J., Irving, R. W., & Manlove, D. F. (2007). Two algorithms for the student-project allocation problem. *Journal of Discrete Algorithms, 5*(1), 73–90. https://doi.org/10.1016/j.jda.2006.03.006

Andrade-Garda, J., Anguera, A., Ares-Casal, J., Hidalgo-Lorenzo, M., Lara, J. A., Lizcano, D., & Suárez-Garaboa, S. (2018). Definition and core components of the role-partner allocation problem in collaborative networks. *International Journal of Economics and Management Engineering, 12*(1), 96–100. https://doi.org/10.5281/zenodo.1315585

Appio, F. P., Martini, A., Massa, S., & Testa, S. (2017). Collaborative network of firms: Antecedents and state-of-the-art properties. *International Journal of Production Research, 55*(7), 2121–2134. https://doi.org/10.1080/00207543.2016.1262083

Biró, P., Fleiner, T., Irving, R. W., & Manlove, D. F. (2010). The college admissions problem with lower and common quotas. *Theoretical Computer Science, 411*(34–36), 3136–3153. https://doi.org/10.1016/j.tcs.2010.05.005

Brynjolfsson, E., & McAfee, A. (2014). *The second machine age: Work, progress, and prosperity in a time of brilliant technologies*. New York: Norton.

Brucker, P., Jurisch, B., & Sievers, B. (1994). A branch and bound algorithm for the job-shop scheduling problem. *Discrete Applied Mathematics, 49*(1–3), 107–127. https://doi.org/10.1016/0166-218X(94)90204-6

Camarinha-Matos, L. M. (2014). Collaborative Networks: A Mechanism for Enterprise Agility and Resilience. In K. Mertins, F. Bénaben, R. Poler, & J. P. Bourrières (Eds), *Enterprise Interoperability VI* (pp. 3-11). Springer. https://doi.org/10.1007/978-3-319-04948-9_1.

Camarinha-Matos, L. M., Afsarmanesh, H., Galeano, N., & Molina, A. (2009). Collaborative networked organizations – Concepts and practice in manufacturing enterprises. *Computers & Industrial Engineering, 57*, 46–60. https://doi.org/10.1016/j.cie.2008.11.024

Camarinha-Matos, L. M., Fornasiero, R., Ramezani, J., & Ferrada, F. (2019). Collaborative networks: A pillar of digital transformation. *Applied Sciences, 9*(24), 5431. https://doi.org/10.3390/app924543

Chen, Y., & Sönmez, T. (2006). School choice: An experimental study. *Journal of Economic Theory, 127*(1), 202–231. https://doi.org/10.1016/j.jet.2004.10.006

Diebold, F., Aziz, H., Bichler, M., Matthes, F., & Schneider, A. (2014). Course Allocation via Stable. Matching. *Business & Information Systems Engineering 6(2)*, 97–110. https://doi.org/10.1007/s12599-014-0316-6.

Durugbo, C. (2016). Collaborative networks: A systematic review and multi-level framework. *International Journal of Production Research, 54*(12), 3749–3776. https://doi.org/10.1080/00207543.2015.1122249

Gale, D. (2001). The two-sided matching problem: Origin, development and current issues. *International Game Theory Review, 3*(2–3), 237–252. https://doi.org/10.1142/S0219198901000373

Gale, D., & Shapley, L. S. (1962). College admissions and the stability of marriage. *The American Mathematical Monthly, 69*(1), 9–15. https://doi.org/10.2307/2312726

Gale, D., & Sotomayor, M. (1985). Some remarks on the stable matching problem. *Discrete Applied Mathematics, 11*, 223–232. https://doi.org/10.1016/0166-218X(85)90074-5

Gharote, M., Patil, R., Lodha, S., & Raman, R. (2015). Assignment of trainees to software project requirements: A stable matching based approach. *Computers & Industrial Engineering, 87*, 228–237. https://doi.org/10.1016/j.cie.2015.05.017

Gusfield, D., & Irving, R. W. (1989). *The stable marriage problem: Structure and algorithms*. Cambridge: MIT Press.

Haas, C., & Hall, M. (2019). Two-Sided Matching for mentor-mentee allocations—Algorithms and manipulation strategies. *PLoS ONE, 14*(3), e0213323.

Haas, C., Hall, M., & Vlasnik, S. (2018). Finding optimal mentor-mentee matches: A case study in applied two-sided matching. *Heliyon, 4*(6), e00634.

Hamada, K., Iwama, K., & Miyazaki, S. (2011). The Hospitals/Residents problem with quota lower bounds. *Lecture Notes in Computer Science, 6942*, 180–191. https://doi.org/10.1007/978-3-642-23719-5_16

Irving, R. W. (1994). Stable marriage and indifference. *Discrete Applied Mathematics, 48*(3), 261–272. https://doi.org/10.1016/0166-218X(92)00179-P

Irving, R. W., Manlove, D. F., & Scott, S. (2003). Strong stability in the hospitals/residents problem. *Lecture Notes in Computer Science, 2607*, 439–450. https://doi.org/10.1007/3-540-36494-3_39

Kesten, O. (2010). School choice with consent. *The Quarterly Journal of Economics, 125*(3), 1297–1348. https://doi.org/10.1162/qjec.2010.125.3.1297

Li, X., Ma, L., Xu, Y., & Shankaran, R. (2020). Resource allocation for D2D-based V2X communication with imperfect CSI. *IEEE Internet of Things Journal, 7*(4), 3545–3558. https://doi.org/10.1109/JIOT.2020.2973267

Malaguti, E., & Medina Durán, R. (2019). Computing *k* different solutions to the assignment problem. *Computers & Industrial Engineering, 135*, 528–536. https://doi.org/10.1016/j.cie.2019.06.025

Manlove, D. F. (2008). Hospitals/residents problem. In M. Y. Kao (Ed.), *Encyclopedia of Algorithms* (pp. 390–394). New York: Springer.

Milani, F., Dumas, M., Matulevičius, R., Ahmed, N., & Kasela, S. (2016). Criteria and heuristics for business process model decomposition: review and comparative evaluation. *Business & Information Systems Engineering, 58*(1), 7–17. https://doi.org/10.1007/s12599-015-0413-1

Qin, J., Xue, K., Li, J., Sun, Q., & Lu, J. (2021). Service prioritization in information centric networking with heterogeneous content providers. *IEEE Transactions on Network and Service Management, 18*(4), 476–4488. https://doi.org/10.1109/TNSM.2021.3105198

Rahwan, I., Ramchurn, S. D., & Jennings, N. R. (2004). Argumentation-based negotiation. *The Knowledge Engineering Review, 18*(4), 343–375. https://doi.org/10.1017/S0269888904000098

Ross, G. T., & Soland, R. M. (1975). A branch and bound algorithm for the generalized assignment problem. *Mathematical Programming, 8*, 91–103. https://doi.org/10.1007/BF01580430

Roth, A. E. (1984). The evolution of the labor market for medical interns and residents: A case study in game theory. *Journal of Political Economy, 92*(6), 991–1016. https://doi.org/10.1086/261272

Roth, A. E. (1986). On the allocation of residents to rural hospitals: A general property of two-sided matching markets. *Econometrica, 54*(2), 425–427. https://doi.org/10.2307/1913160

Roth, A. E. (2002). The economist as engineer: game theory, experimentation, and computation as tools for design economics. *Econometrica, 70*(4), 1341–1378. https://doi.org/10.1111/1468-0262.00335

Roth, A. E. (2008). Deferred acceptance algorithms: History, theory, practice, and open questions. *A Collection of Papers Dedicated to David Gale on the Occasion of His 85th Birthday, Special Issue, International Journal of Game Theory 36(3-4)*, 537-569. https://doi.org/10.3386/w13225.

Roth, A. E., & Peranson, E. (1997). The effects of a change in the NRMP matching algorithm. *Journal of the American Medical Association, 278*(9), 729–732. https://doi.org/10.1001/jama.1997.03550090053032

Roth, A. E., & Sotomayor, M. (2008). *Two-sided matching: A study in game theoretic modeling and analysis*. New York: Cambridge University Press.

Roth, A. E., Sönmez, T., & Ünver, M. U. (2004). Kidney exchange. *The Quarterly Journal of Economics, 119*, 457–488. https://doi.org/10.1162/0033553041382157

Shapley, L. S., & Scarf, H. (1974). On cores and indivisibility. *Journal of Mathematical Economics, 1*, 23–28. https://doi.org/10.1016/0304-4068(74)90033-0

Sönmez, T., & Ünver, M. U. (2010). Course bidding at business schools. *International Economic Review, 51*(1), 99–123. https://doi.org/10.1111/j.1468-2354.2009.00572.x

vom Brocke, J., Maaß, W., Buxmann, P., Maedche, A., Leimeister, J. M., & Pecht, G. (2018). Future work and enterprise systems. *Business & Information Systems Engineering, 60*(4), 357–366. https://doi.org/10.1007/s12599-018-0544-2