International Journal of Game Theory (2019) 48:835–862
https://doi.org/10.1007/s00182-019-00664-6

ORIGINAL PAPER

Paths to stable allocations

Ágnes Cseh[1] · **Martin Skutella**[2]

Accepted: 17 February 2019 / Published online: 21 February 2019

Abstract
The stable allocation problem is one of the broadest extensions of the well-known stable marriage problem. In an allocation problem, edges of a bipartite graph have capacities and vertices have quotas to fill. Here we investigate the case of uncoordinated processes in stable allocation instances. In this setting, a feasible allocation is given and the aim is to reach a stable allocation by raising the value of the allocation along blocking edges and reducing it on worse edges if needed. Do such myopic changes lead to a stable solution? In our present work, we analyze both better and best response dynamics from an algorithmic point of view. With the help of two deterministic algorithms we show that random procedures reach a stable solution with probability one for all rational input data in both cases. Surprisingly, while there is a polynomial path to stability when better response strategies are played (even for irrational input data), the more intuitive best response steps may require exponential time. We also study the special case of correlated markets. There, random best response strategies lead to a stable allocation in expected polynomial time.

Keywords Stable matching · Stable allocation · Paths to stability · Best response strategy · Better response strategy · Correlated market

A short version of this paper has appeared in the proceedings of SAGT 2014, the 7th International Symposium on Algorithmic Game Theory. This work was supported by the Cooperation of Excellences Grant (KEP-6/2018), by the Ministry of Human Resources under its New National Excellence Programme (ÚNKP-18-4-BME-331), the Hungarian Academy of Sciences under its Momentum Programme (LP2016-3/2016), its János Bolyai Research Fellowship, OTKA grant K128611, and the Cluster of Excellence MATH+ (EXC 2046/1, project ID: 390685689).

✉ Ágnes Cseh
cseh.agnes@krtk.mta.hu

Martin Skutella
skutella@math.tu-berlin.de

[1] Institute of Economics, Centre for Economic and Regional Studies, Hungarian Academy of Sciences, Tóth Kálmán u. 4., 1097 Budapest, Hungary

[2] TU Berlin, Institut für Mathematik, Straße des 17. Juni 136, 10623 Berlin, Germany

1 Introduction

Capacitated matching markets without prices model various real-life problems such as, e. g., employee placement, task scheduling or admission procedures. Research on markets with ordinal preferences focuses on maximizing social welfare instead of profit, which is usually expressed in cardinal terms. Stability is probably the most widely used optimality criterion in that case.

Finding equilibria in markets that lack a central authority of control is another widely studied, challenging task. Besides modeling uncoordinated markets such as third-generation (3G) wireless data networks (Goemans et al. 2006) and ride-sharing systems (Wang et al. 2017), selfish and uncontrolled agents can also represent modifications in coordinated markets, e. g., the arrival of a new agent or slightly changed preferences (Blum et al. 1997). In our present work, those two topics are combined: we study uncoordinated capacitated matching markets.

1.1 Stability in matching markets

The theory of stable matchings has been investigated for decades. Gale and Shapley (1962) introduced the notion of stability on their well-known *stable marriage problem*. An instance of this problem consists of a bipartite graph where the two vertex groups symbolize men and women, respectively. Each agent has a preference list of their acquaintances of the opposite gender. A set of marriages (a matching) is *stable*, if no pair blocks it. A *blocking pair* is an unmarried pair so that the man is single or he prefers the woman to his current wife and vice versa, the woman is single or she prefers the man to her current husband. The Gale-Shapley algorithm was the first proof for the existence of stable matchings.

A natural extension of matching problems arises when capacities are introduced. The stable allocation problem is defined in a bipartite graph with edge capacities and quotas on vertices. The exact problem formulation and a detailed example are provided in Sect. 2.

1.2 Better and best response steps in uncoordinated markets

Central planning is needed in order to produce a stable solution with the Gale–Shapley algorithm. In many real-life situations, however, such a coordination is not available. Yet stability is a naturally desirable property of uncoordinated markets. A stable matching seems to be the best reachable solution for all agents, because they cannot find any partnership that could improve their own position. In uncoordinated markets, agents play their selfish strategy, trying to reach the best possible solution.

A *path to stability* is a series of myopic operations, each of which can occur without any central coordination. The intuitive picture of a myopic operation is the following. If a man and a woman block a marriage scheme, then they both agree to form a couple together, even if they divorce their current partners to that end. The recently divorced agents may induce new blocking pairs. In a path to stability, such changes are made until a stable matching is reached.

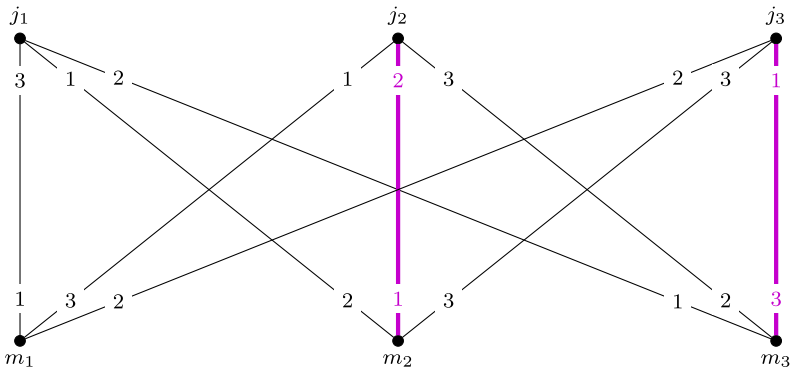

Fig. 1 A stable marriage instance and a cycle of best response blocking edges. Starting with the unstable matching (j_2m_2, j_3m_3), and saturating the blocking edges j_1m_3, j_2m_1, j_3m_1, j_1m_2, j_2m_2, j_3m_3 in this order leads back to the same unstable matching. In each round, the chosen blocking edge is the best blocking edge of the corresponding vertex j_i

The study of uncoordinated matching processes has a long history. In the case of one-to-one matchings, two different concepts have been studied: better and best response dynamics. One of the agent groups is chosen to be the *active* side. These vertices submit proposals to the *passive* vertices. According to *best response dynamics*, the best blocking edge of an active vertex is chosen to perform myopic changes along. In *better response dynamics*, any blocking edge can play this role. Observe that the Gale–Shapley algorithm itself can be seen as series of best response steps, with men being the active side.

The core questions regarding uncoordinated processes rise naturally. Can a series of myopic changes result in returning back to the same unstable matching? If yes, is there a way to reach a stable solution? How do random procedures behave? The first question about uncoordinated two-sided matching markets was brought up by Knuth (1997) in 1976. He also gives an example of a matching problem where better response dynamics cycle. More than a decade later, Roth and Vande Vate (1990) came up with the next result on the topic. They show that random better response dynamics converge to a stable matching with probability one. Analogous results for best response dynamics were published in 2011 by Ackermann et al. (2011). They also show an instance in which best response dynamics cycle (see Fig. 1), give a deterministic algorithm for reaching a stable solution in polynomial time and prove that the convergence time is exponential in both random cases.

Besides these works on the classical stable marriage problem, there is a number of papers investigating it from the paths-to-stability point of view. One dynamic approach is to add or remove agents of the stable marriage instance one by one or in groups. Blum et al. (1997) and Blum and Rothblum (2002) develop a natural procedure which reflects the greedy behavior of agents and leads to a stable matching in the new instance. A path to stability also exists in the bipartite matching case with payments where flexible salaries and productivity are taken into account (Chen et al. 2016). Closely related optimality concepts, such as socially stable, locally stable, friendship matching,

and considerate matching have also been investigated in uncoordinated one-to-one matching markets (Hoefer and Wagner 2014).

Various extensions of the stable marriage problem also have been studied in the dynamic setting. For the stable roommates problem, the non-bipartite version of the stable marriage problem, it is known that there is a series of myopic operations that leads to a stable solution, if one exists (Diamantoudi et al. 2004). In a special one-to-many matching scenario, the hospitals/residents assignment problem with couples, the existence of such a path is only guaranteed if the preferences are weakly responsive (Klaus and Klijn 2007). Weak responsiveness ensures consistence between the preferences of each partner and the couple's preference list on pairs of hospitals. In many-to-many markets, supposing substitutable preferences on one side and responsive preferences on the other side, a path to stability can be found (Kojima and Ünver 2008). Both substitutable and responsive preferences are defined in instances where preferences are given over sets of vertices. In the matching with contracts model, a path to stability exists for a many-to-many matching model with contracts, if the preferences on one side are substitutable and the preferences on the other side satisfy substitutability and two further assumptions (Millán and Risma 2018).

Our setting is a many-to-many bipartite instance with strictly ordered preferences over agents. The preference structure we work with is stricter than some other investigated structures, in particular, it satisfies the conditions in Kojima and Ünver (2008) and Millán and Risma (2018). However, while all previous work is valid for matchings or b-matchings, where an edge is either fully inside or fully outside a matching, our setting allows edges to be capacitated, and be partially contained in a solution. Up to our knowledge, although many variants of the stable marriage problem have been studied, and the range of investigated preference structures is quite broad, no paper discusses the case of capacitated allocations. Our present work makes an attempt to fill this gap in the literature.

Structure of the paper In the next section, the essential theoretical basis is provided: besides stable allocations, better and best response modifications are also defined formally. In Sect. 3, allocation instances with characteristic preference profiles are investigated. We show that although random best response processes generally run in exponential time, in the case of correlated markets, polynomial convergence is expected. Better and best response dynamics in the general case on rational input are extensively studied in Sect. 4. We describe two deterministic algorithms that generalize the result of Ackermann et al. on one-to-one matching markets to stable allocation instances and also show algorithmic differences between better and best response strategies. In the case of random procedures, convergence is shown for both strategies. Sect. 5 focuses on running time efficiency and contains our main result. There, a better response algorithm is presented that terminates with a stable solution in $O(|V|^2|E|)$ time in a graph with $|V|$ vertices and $|E|$ edges, even for irrational input data. A counterexample proves that such an acceleration cannot be reached for the best response dynamics. Our contribution is summarized in Table 1.

Applied to a matching instance, our best-response algorithm (in Sect. 4) performs the same steps as the two-phase best response algorithm of Ackermann et al. Our better-response variant can also be interpreted as an extended version of the above mentioned

Table 1 Our results for a shortest and a random path to a stable allocation in instances with rational input

	Shortest path to stability	Random path to stability
Best response dynamics	Exponential length	Converges with probability 1
Better response dynamics	Polynomial length	Converges with probability 1

method. The only difference is that while our first phase is better response, while theirs is best response. However, this seems to be a minor difference, as their proof is also valid for a better response first phase, and our proof still holds if only best blocking edges are chosen. Moreover, stable allocations might be the most complex model in which this approach brings results. The most intuitive extension of Ackermann et al.'s algorithm for stable flows, defined by Fleiner (2014), does not even result in feasible myopic changes.

On the other hand, our accelerated better-response algorithm (in Sect. 5) generalizes another known method, the polynomial algorithm that finds a stable allocation. Applied directly to an instance with empty allocation, our accelerated Phase II performs augmentations like the augmenting path algorithms of Baïou and Balinski (2002), and of Dean and Munshi (2010). Since our accelerated Phase II is a slightly modified variant of our first algorithm, our solution concept offers a bridge between two known methods for solving two different problems, namely the paths to stability problem in stable marriage instances and the stable allocation problem, providing a solution to both of them.

2 Preliminaries

In this section we define stable allocations formally, and then proceed to the description of better and best response myopic changes in stable allocation instances.

2.1 Stable allocations

The stable marriage problem has been extended in several directions. A great deal of research effort has been spent on *many-to-one* and *many-to-many matchings*, sometimes also referred to as b-matchings. Their extension is called the *stable allocation problem*, also known as the ordinal transportation problem, since it is a direct analog of the classical cost-based transportation problem. In this problem, the vertices of a bipartite graph $G = (V, E)$ have *quotas* $q : V(G) \to \mathbb{R}_{\geq 0}$, while edges have *capacities* $c : E(G) \to \mathbb{R}_{\geq 0}$. Both functions are *real-valued*, unlike the respective functions in many-to-many instances, where capacities are unit, while quotas are integer-valued. Therefore, allocations can model more complex problems, for example where goods can be divided unequally between agents. In order to avoid confusion caused by terms associated with the marriage model, we call the vertices of the first side *jobs* and the remaining vertices *machines*. For each machine, its quota is the maximal time spent working. A job's quota is the total time that machines must spend on the job in order

to complete it. In addition, machines have a limit on the time spent on a specific job; this is modeled by edge capacities. A feasible allocation is a set of contracts where no machine is overloaded and no job is worked on after it has been completed.

Definition 1 (allocation) Function $x : E(G) \to \mathbb{R}_{\geq 0}$ is called an *allocation* if for every edge $e \in E(G)$ and every vertex $v \in V(G)$:

1. $x(e) \leq c(e)$;
2. $x(v) := \sum_{e \in \delta(v)} x(e) \leq q(v)$, where $\delta(v)$ is the set of edges incident to v.

For an edge e with $x(e) > 0$ we say that e is in x. To define stability we need *preference lists* as well. All vertices rank their incident edges strictly. Vertex v prefers uv to wv, if uv is ranked better on v's preference list than wv: $\text{rank}_v(uv) < \text{rank}_v(wv)$. In this case we say that uv *dominates* wv at v. A stable allocation instance consists of four elements: (G, q, c, O), where O is the set of all preference lists.

Definition 2 (blocking edge, stable allocation) An allocation x is *blocked* by an edge jm if all of the following properties hold:

1. $x(jm) < c(jm)$;
2. $x(j) < q(j)$ or j prefers jm to its worst edge with positive value in x;
3. $x(m) < q(m)$ or m prefers jm to its worst edge with positive value in x.

A feasible allocation is *stable* if no edge blocks it.

In other words, edge jm is blocking if it is unsaturated and neither end vertices of jm has filled up its quota with at least as good edges as jm. If an unsaturated edge fulfills the second criterion, then we say that it *dominates* x at j. Similarly, if the third criterion is fulfilled for an unsaturated edge, then we talk about an edge dominating x at m.

Example 1 Figure 2 illustrates a stable allocation instance. We use the same example throughout the entire paper to demonstrate different notions defined here. For the sake of simplicity, all edge capacities are unit. The numbers within parenthesis over and under the vertices represent the quota function. The preferences can be seen on the edges: the more preferred edges carry a better rank, i.e., a smaller number. For example, machine m_1's most preferred job is j_2, its second choice is j_3, while its least preferred, but still acceptable job is j_1. The function $x = 1$ on the colored edges and $x = 0$ on the remaining edges is a feasible allocation, since no quota or capacity constraint is violated. The unique blocking edge is easy to find: j_3m_1 blocks x, because it is unsaturated and both end vertices have free quota.

Baïou and Balinski (2002) prove that stable allocations always exist. They also give two algorithms for finding them, an extended version of the Gale–Shapley algorithm and an inductive algorithm. The worst case running time of the first algorithm is exponential, but the latter one runs in strongly polynomial time. Dean and Munshi (2010) speed up the polynomial algorithm using sophisticated data structures: their version runs in $O(|E| \log |V|)$ time for any real-valued instance.

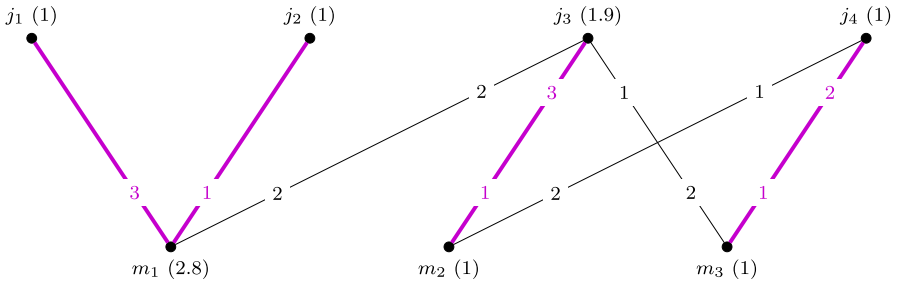

Fig. 2 A stable allocation instance with unit capacities and a feasible, but unstable allocation, marked by colored edges

2.2 Better and best response steps for allocations

First, we provide some basic definitions and notations we will use throughout the entire paper. A feasible, but possibly unstable allocation x is given at the beginning, thus the instance can be written as $\mathcal{I} = (G, q, c, O, x)$. In our instance $\mathcal{I}$, jobs form the active side J, while machines M are passive players. For the sake of simplicity we denote the residual capacity $c(jm) - x(jm)$ of edge jm by $\bar{x}(jm)$ and similarly, the residual quota $q(v) - x(v)$ of vertex v by $\bar{x}(v)$. The definition of better and best response strategies is not as straightforward as it is in the matching instance with unit quotas and capacities. Here, the possible outcomes for a player are ordered lexicographically. We say that machine m prefers allocation x_1 to allocation x_2 if $x_1(j'm) > x_2(j'm)$ the for the best ranked edge $j'm$ among edges with $x_1(jm) \neq x_2(jm)$.

Although lexicographic order seems to be a natural choice, it is somewhat against the convention when discussing stable allocations. In most cases, when comparing the position of an agent in two stable allocations, the so called *min-min criterion* is used Baïou and Balinski (2002). According to this rule, the agent prefers the allocation in which its worst edge in x is ranked better. In order to make use of such an ordering relation, each vertex has to have the same allocation value in all stable solutions. Therefore here, when studying and comparing arbitrary feasible allocations, this concept proves to be counter-intuitive.

An active player j having some blocking edges is chosen to perform a *best response step* on the current allocation x. Amongst j's blocking edges, let jm be the one ranked best on j's preference list. The aim of player j is to reach its best possible lexicographic position via increasing $x(jm)$. To this end, j is ready to allocate all its remaining quota $\bar{x}(j)$ to jm, moreover, it may reassign allocation from all edges worse than jm to jm. Thus, j aims to increase $x(jm)$ by $\bar{x}(j)+x$(edges dominated by jm at j). To preserve feasibility, $x(jm)$ is not increased by more than $\bar{x}(jm)$. The passive player m agrees to increase $x(jm)$ as long as it does not lose allocation on better edges. This constraint gives the third upper bound, $\bar{x}(m) + x$(edges dominated by jm at m). To summarize this, in a best response step $x(jm)$ is increased by the following amount.

$$A := \min\{\bar{x}(j) + x(\text{edges dominated by } jm \text{ at } j), \bar{x}(jm), \\ \bar{x}(m) + x(\text{edges dominated by } jm \text{ at } m)\}$$

Once this A and the new $x(jm)$ is determined, j and m fill their remaining quota, then refuse allocation on their worst allocated edges, until x becomes feasible.

Better response steps are much less complicated to describe. The chosen active vertex j increases the allocation on an arbitrary blocking edge jm. Both j and m are allowed to refuse allocation on worse edges than jm. This rule guarantees that j's lexicographic situation improves and that the change is myopic for both vertices. By definition, best response steps are always better response steps at the same time. The execution of a single better response step consists of modifications on at most $|\delta(j)| + |\delta(m)| - 1 \leq |V| - 1$ edges.

Example 2 In our example above, j_3 and m_1 mutually agree to allocate value 1 to j_3m_1. If best response strategies are played, m_1 refuses 0.2 amount of allocation from j_1m_1, while j_3 reduces $x(j_3m_2)$ to 0.9. Through this step, they induce blocking elsewhere in G: now j_4m_2 blocks the new x, because m_2 lost some allocation. Thus, another myopic change would now be to increase $x(j_4m_2)$, and so on. A better response step of the same vertex j_3 would be for example to increase $x(j_3m_1)$ to 1, while refusing j_3m_2 entirely. To keep feasibility, m_1 has to refuse 0.2 amount of allocation on j_1m_1.

3 Correlated markets

Before tackling the general paths to stability problem, we first restrict ourselves to instances with characteristic preference profiles. In this section, we study the case of stable allocations on an uncoordinated market with correlated preferences. Later we will prove that the convergence time of random best and better response strategies is exponential in general instances. By contrast, here we show that on correlated markets, random best response strategies terminate in expected polynomial time, even in the presence of irrational data. At the end of this section we also elaborate on the behavior of better response dynamics.

Definition 3 (*correlated market*) An allocation instance is *correlated*, if there is a function $f : E(G) \to \mathbb{N}$ such that $\text{rank}_v(uv) < \text{rank}_v(wv)$ if $f(uv) < f(wv)$ for every $u, v, w \in V(G)$ and no two edges have the same f value.

Correlated markets are also called *instances with globally ranked pairs* or *acyclic markets*. The latter property means that there is no cycle of incident edges such that every edge is preferred to the previous one by their common vertex. Abraham et al. (2008) show that acyclic markets are correlated and vice versa. The instance depicted in Fig. 2 is not correlated: edges (j_3m_3, j_4m_3, j_4m_2, j_3m_2) form a preference cycle. Ackermann et al. (2011) were the first to prove that random better and best response dynamics reach a stable matching on correlated markets in expected polynomial time. Using a similar argumentation, we extend their result to allocation instances.

Theorem 1 *In correlated allocation instances with real-valued input data, random best response dynamics reach a stable solution in expected time* $\mathcal{O}(|V|^2|E|)$.

Proof Before studying paths to stability we show that in correlated instances, the set of stable solutions has cardinality one. There is an absolute minimum of $f(jm)$. The

single edge jm with this minimal f value must be in all stable allocations with value $\min\{c(jm), q(j), q(m)\}$, otherwise it is blocking. Fixing x on jm and decreasing the quotas of j and m respectively leads to another correlated allocation instance. In this instance, the stable solutions are exactly the stable solutions of the original instance without jm. This leads to an inductive algorithm that proves that there is a unique stable allocation on correlated markets. We now turn to showing that random best response dynamics reach this unique solution in expected polynomial time.

Whenever a job j with an unsaturated edge jm of an absolute minimal $f(jm)$ is chosen to submit an offer, its best response strategy is to increase x on jm. Due to this single best response operation performed by j, $x(jm) = \min\{c(jm), q(j), q(m)\}$ is reached. The probability that a vertex $j \in J$ is chosen to take the next step is at least $\frac{1}{|J|}$. As mentioned in Sect. 2.2, one best response step requires at most $\mathcal{O}(|V|)$ modifications. Thus, in order to reach $x(jm) = \min\{c(jm), q(j), q(m)\}$ on the best edge in G, $|J| \cdot |V| = \mathcal{O}(|V|^2)$ modifications are needed in expectation. After this, the edge jm with minimal f value will have reached its final position in the unique stable allocation. From this point on, $x(jm)$ will never be reduced, because neither j, nor m have a better incident edge. Thus, $x(jm)$ can be fixed, and a new minimum of f can be chosen for the same procedure as before. The number of iterations is bounded from above by the number of edges in the graph. The unique stable allocation is thus reached in $\mathcal{O}(|V|^2|E|)$ time in expectation. □

In order to establish a similar result for better response dynamics in real-valued correlated instances, an exact interpretation of random events would be needed. In the matching case, best and better response dynamics differ exclusively in the rank of the chosen blocking edge: when playing best response strategy, the best blocking edge is chosen by an active vertex j. In contrast to this, here, better response steps differ also in the amount of modification and in the edges chosen to refuse allocation along. The first factor indicates a continuous state space.

However, if we assume that any better response step results in reassigning the highest possible allocation value to an arbitrary blocking edge, an analogous proof can be derived.

Theorem 2 *On correlated allocation instances with real-valued input data, random better response dynamics reach a stable solution in expected time* $\mathcal{O}(|V|^3|E|)$.

Proof The only difference to the setting with best response steps is that after j is chosen, the expected time of reaching $x(jm) = \min\{c(jm), q(j), q(m)\}$ is larger. In this case, j chooses jm with probability at least $\frac{1}{|\delta(j)|}$. This implies that reaching the stable allocation value on the best edge takes $|\delta(j)| \cdot (|\delta(j)| + |\delta(m)| - 1) = \mathcal{O}(|V|^2)$ steps in expectation. In total, for all vertices $j \in J$ and all edges the algorithm takes $\mathcal{O}(|V|^3|E|)$ steps in expectation. □

4 Best and better responses with rational data

In this section, the case of allocations in an uncoordinated market *with rational data* is studied. As already mentioned, better and best response dynamics can cycle

in such instances. We describe two deterministic methods, a better-response and a best-response algorithm that yield stable allocations in finite time. Our best response algorithm is by definition a better response algorithm as well, yet we present a different, better but not best response strategy in Sect. 4.1, because it can be accelerated to reach a stable solution in polynomial time, while the best response strategy cannot, as shown in Sect. 4.2. The main idea of our algorithms is to distinguish between blocking edges based on the type of blocking at the job: dominance or free quota.

A blocking edge can be of two types. Recall point 2 of Definition 2: if jm blocks x, then $x(j) < q(j)$ or j prefers jm to its worst edge with positive value in x. We talk about *blocking of type I* in the latter case, if jm blocks x because j prefers jm to its worst edge having positive value in x. *Blocking of type II* means that j has no allocated edge that is worse than jm, but j has not filled up its quota yet, $x(j) < q(j)$. Note that the reason of the blocking property at m is not involved when defining the two types.

Example 3 Recall our example in Fig. 2. The unique blocking edge j_3m_1 is of type I, because j_3, its active vertex, prefers edge j_3m_1 to its worst allocated edge j_3m_2.

4.1 Better response dynamics

First, we provide a deterministic algorithm that constructs a finite path to stability from any feasible allocation. In the first phase of our algorithm, only blocking edges of type I are chosen to perform myopic changes along. The active vertices (jobs) choose one of their blocking edges of type I, not necessarily the best one. In all cases, withdrawal is executed along worst allocated edges. The amount of new allocation added to the blocking edge is determined in such a way that at least one edge or a vertex becomes saturated or empty. Thus, in the first phase, active vertices replace their worst edges with better ones, even if they have free quota. When no blocking edge of type I remains, the second phase starts. The allocation value is increased on blocking edges of type II such that they cease to be blocking.

The runtime of our algorithm is exponential. Later, in Sect. 5 we will show that this algorithm can be accelerated such that a stable solution is reached in strongly polynomial time.

Theorem 3 *For every allocation instance with rational data and a given rational feasible allocation x, there is a finite sequence of better responses that leads to a stable allocation.*

The main idea of the proof is the following. We need to keep track of the change in the size of the allocation and in the lexicographic position of the active vertices simultaneously. In one step of the first phase along edge jm, either both j and m refuse edges, thus, the size of the allocation $|x| = \sum_{j \in J} x(j)$ decreases, or only j does so, leaving $|x|$ unchanged and improving j's situation lexicographically. Since both procedures are monotone and the second one does not impair the first one, the first phase terminates. Termination of the second phase is implied by the fact that passive vertices improve their lexicographic situation in each step. The technical details of this proof sketch are presented as Claims 4.1 and 4.1.

In the first phase, the jobs propose along *arbitrary* blocking edges of type I. We will show that this process ends with an allocation where no job has a blocking edge of type I. In the second phase, the jobs propose along their *best* blocking edges of type II. Later we will see that during this phase until termination, no job gets a blocking edge of type I. A pseudocode is provided after the description of both phases.

First phase In one step, an arbitrary blocking edge jm of type I is chosen. Both end vertices, j and m may refuse some allocation along worse edges when increasing x on jm. Job j has a *refusal pointer* $r(j)$ that denotes the worst edge allocated to j, if any exists. Similarly, $r(m)$ stands for the worst currently allocated edge of m. A step of Phase I consists of two or three operations, each along $jm, r(j)$ and possibly along $r(m)$. Two operations take place, if m has not filled up its quota yet. In this case, $x(r(j))$ is decreased by $A := \min\{x(r(j)), \bar{x}(jm), \bar{x}(m)\}$. At the same time, $x(jm)$ is increased by the same amount. Depending on which expression is the minimal one, edge $r(j)$ becomes empty or jm becomes saturated or m fills up its quota. Note that $r(m)$ plays no role because m does not refuse any allocation. In the remaining case, if m has a full quota, three operations take place, since m has to refuse some allocation. The amount of allocation we deal with is now $A := \min\{x(r(j)), \bar{x}(jm), x(r(m))\}$. The allocation on the blocking edge jm will be increased by A, on the other two edges it will be decreased by A, until one of them becomes empty or saturated. We emphasize that whenever a job j with free quota adds a new edge better than its worst allocated edge to x, it withdraws some allocation from the worst edge.

Example 4 We return to our example again. It has already been mentioned that the unique blocking edge j_3m_1 is of type I. The refusal pointer $r(j_3)$ is j_3m_2. Since m_1 has not filled up its quota yet, its refusal pointer j_1m_1 is irrelevant at the moment. Due to the same reason, two operations take place. We augment with the following amount of allocation: $\min\{x(j_3m_2), \bar{x}(j_3m_1), \bar{x}(m_1)\} = 0.8$. After this operation, $x(j_3m_1) = 0.8, x(j_3m_2) = 0.2$, and j_3m_1 is still a Phase I blocking edge. Since $x(m_1) = q(m_1)$ holds now, three operations are executed with $A = \min\{x(j_3m_2), \bar{x}(j_3m_1), x(j_1m_1)\} = 0.2$. Now j_3m_1 is saturated, hence it ceases to be blocking. During the first operation, j_4m_2 became blocking of type I, because m_2 lost allocation. In the next step, one unit of allocation is reallocated to j_4m_2 from j_4m_3. But j_3m_3 then becomes blocking of type I, and so on.

Claim Phase I terminates in finite time.

Proof We use the following potential function in order to show that the process does not cycle:

$$\Theta(x) := \sum_{j \in J} \sum_{jm \in E(G)} x(jm) \operatorname{rank}_j(jm)$$

Recall that $\operatorname{rank}_j(jm)$ stands for the rank of jm on j's preference list. The smaller $\operatorname{rank}_j(jm)$ is, the better m is for j. The expression above is bounded for any feasible allocation x:

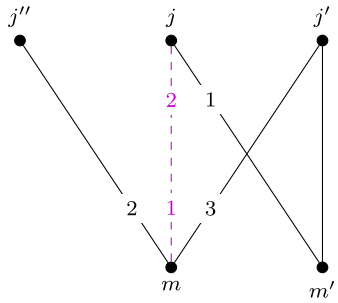

Fig. 3 Edges affected by one myopic operation along the blocking edge jm of type II

$$0 \leq \Theta(x) \leq |J| \cdot \max_{jm \in E(G)} c(jm) \cdot \max_{j \in J} |\delta(j)|.$$

We will show that $\Theta(x)$ decreases in each step of the procedure. The process terminates if the amount of decrement is always greater than a fixed positive constant. If all data are rational, this is guaranteed.

Considering the potential function, we need to keep track of those two jobs that proposed or got refused, since the allocation of all other jobs remains the same, thus their contributions to the summations of $\Theta(x)$ do not change.

As mentioned above, a step consists of either two or three edges changing their value in x. In the first case, when only two edges change their value in x, there is only one job j that modifies its contribution. Thus $\Theta(x)$ decreases, because some allocation will move from a less preferred edge to jm. In the second case, where three edges are involved, there is a job j that improves its lexicographic position, and another job j' that loses allocation. The effect of the first change at j is just as above, $\Theta(x)$ decreases. Losing allocation for j' also decreases $\Theta(x)$, since $x(j')$ decreases. □

Second phase When the first phase terminates, all blocking edges are of type II. In the second phase, we are allowed to increase $x(j)$. When improving the allocation along a blocking edge jm of type II, m may refuse some allocation, but j cannot, since the reason of blocking is that j has not filled up its quota yet. Thus, we do not need the pointer $r(j)$ any more. One step consists of changes along one edge if $x(m) < q(m)$, or along two edges otherwise. If m has not filled up its quota yet, then we simply assign as much allocation to jm as possible without $x(j)$, $x(m)$ and $x(jm)$ exceeding $q(j)$, $q(m)$ and $c(jm)$, respectively. If m has to refuse something from a job j' in order to accept better offers from j, we improve m's position until $j'm$ becomes empty or jm becomes saturated or j's quota is filled up.

Claim No step in Phase II can induce a blocking edge of type I.

Proof One step in Phase II leaves all vertices but j, m and the possibly refused j' unchanged. Thus, if there is a blocking edge of type I after the modification, it must be incident to one of those vertices. The three cases, illustrated in Fig. 3, are the following.

- Edge $j''m$ blocks x. The position of m became lexicographically better, thus, no new blocking edge incident to m was introduced. The existing blocking edges $j''m$ of type II cannot become of type I, because j'''s position remained unchanged.
- Edge jm' (or jm) blocks x. The only change at j is that $x(jm)$ increases, thus, j also improves its lexicographic position. Therefore, no new blocking edge incident to j appeared. Blocking edges of type II can change their type of blocking only if j increased its allocation on a worse edge. But this cannot happen since we chose the best blocking edge jm in Phase II.
- Edge $j'm'$ (or $j'm$) blocks x. The only change in j''s neighborhood is that $x(j'm)$ decreases. After this step, consider an unsaturated edge $j'm'$ preferred by j' to its worst allocated edge. Since no machine worsens its lexicographic position in Phase II, if $j'm'$ dominates the new allocation x, it already dominated the previous allocation. Therefore, $j'm'$ must have been a blocking edge of type II prior to the modification and thus remains of type II.

We have argued that once Phase II has started, Phase I can never return. □

The last step ahead of us is to show that Phase II may not cycle. But this follows from the fact that in each step exactly one machine strictly improves its lexicographic situation, while all other machines maintain the same allocation as before. In case of a rational input, this improvement is bounded from below, thus, the second phase of the algorithm terminates.

With this we finished the proof of Theorem 3.

Algorithm 1 Two-phase better response algorithm

while $\exists j \in J$ with a blocking edge of type I **do**
  IMPROVEMENT_I(j)
end while
while $\exists j \in J$ with a blocking edge of type II **do**
  IMPROVEMENT_II(j)
end while

procedure IMPROVEMENT_I(j)
  $jm \leftarrow$ blocking edge of type I of j
  if $x(m) < q(m)$ **then**
    $A := \min\{x(r(j)), \bar{x}(jm), \bar{x}(m)\}$
    $x(r(j)) := x(r(j)) - A$
    $x(jm) := x(jm) + A$
  else
    $A := \min\{x(r(j)), \bar{x}(jm), x(r(m))\}$
    $x(r(j)) := x(r(j)) - A$
    $x(jm) := x(jm) + A$
    $x(r(m)) := x(r(m)) - A$
  end if
end procedure

procedure IMPROVEMENT_II(j)
  $jm \leftarrow$ best blocking edge of type II of j
  if $x(m) < q(m)$ **then**
    $A := \min\{\bar{x}(jm), \bar{x}(j), \bar{x}(m)\}$
    $x(jm) := x(jm) + A$
  else
    $A := \min\{x(r(m)), \bar{x}(jm), \bar{x}(j)\}$
    $x(jm) := x(jm) + A$
    $x(r(m)) := x(r(m)) - A$
  end if
end procedure

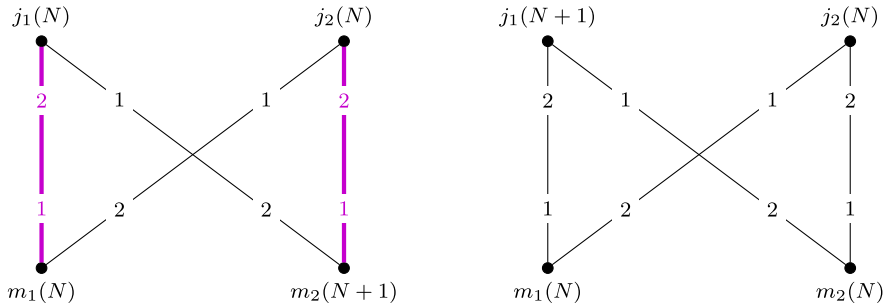

Fig. 4 Worst-case instances for our better response algorithm. On the graph on the left hand-side, Phase I cycles along $\langle j_1m_2, j_2m_2, j_2m_1, j_1m_1\rangle$ N times. In the second instance, Phase II first assigns N amount of allocation to edges j_1m_2 and j_2m_1 and then cycles N times along $\langle j_1m_1, j_2m_1, j_2m_2, j_1m_2\rangle$

Example 5 The duration of both phases strongly depends on the capacities and quotas. The examples in Fig. 4 show two bad instances. The capacity is N on all edges, where N is an arbitrarily large integer. Quotas are marked above and below the vertices. In the first instance, the initial allocation for Phase I is N on j_1m_1 and on j_2m_2 and zero on the remaining two edges. The first phase performs N augmenting steps along the same cycle. Phase II terminates after N iterations in the second instance, starting with the empty allocation.

This algorithm also proves an important result regarding rational random better response processes. If the input is rational (there is a smallest positive number that can be represented as a linear combination of all data), it is clearly worthwhile to restrict the set of feasible better response modifications to the ones that reassign a multiple of this unit. Under this assumption, the set of reachable allocations is finite and they can be seen as states of a discrete time Markov chain. Our algorithm proves that from any state there is a finite path to an absorbing state with a positive probability.

Theorem 4 *In the rational case, random better response strategies terminate with a stable allocation with probability one.*

Polynomial time convergence cannot be shown for random better response strategies, since they need exponential time to converge in expectation even in matching instances (Ackermann et al. 2011).

4.2 Best response dynamics

In this section, we derive analogous results for best response modifications to the ones established for better response strategies. The main difference from the algorithmic point of view is that instances can be found in which no series of best response strategies terminates with a stable solution in polynomial time. A simple example shown on the right in Fig. 4 resembles the instance given by Baïou and Balinski (2002) to prove that the Gale–Shapley algorithm requires exponential time to terminate in stable allocation instances.

Example 6 Let G be a complete bipartite graph on four vertices, with quota $q(j_1) = N+1$, $(j_2) = q(m_1) = q(m_2) = N$, and initial allocation $x(j_1m_1) = x(j_2m_2) = N$ for an arbitrary large number N. If the preference profile is chosen to be cyclic, such that $\text{rank}_{j_1}(m_1) = \text{rank}_{j_2}(m_2) = \text{rank}_{m_1}(j_2) = \text{rank}_{m_2}(j_1) = 2$, the unique series of best response steps consists of $2N$ operations. This example shows that a polynomial path to stability does not exist, not even for rational input data. A path of exponential length to stability can still be found. Our next theorem shows that this is the case in general.

Theorem 5 *For every allocation instance with rational data and a given rational feasible allocation x, there is a finite sequence of best responses that leads to a stable allocation.*

Proof Similar to our method for better response strategies, we prove that there is a two-phase algorithm that terminates with a stable solution.

All blocking edges we take into account are best blocking edges of their job j. Depending on their rank compared to j's worst allocated edge $r(j)$, they are either of type I or type II. A job j's best blocking edge jm is

- of type I(a), if $\text{rank}_j(jm) < \text{rank}_j(r(j))$ and $\bar{x}(j) < \min\{\bar{x}(jm), \bar{x}(m) + x(\text{edges dominated by } jm \text{ at } m)\}$;
- of type I(b), if $\text{rank}_j(jm) < \text{rank}_j(r(j))$ and $\bar{x}(j) \geq \min\{\bar{x}(jm), \bar{x}(m) + x(\text{edges dominated by } jm \text{ at } m)\}$;
- of type II, if $\text{rank}_j(jm) \geq \text{rank}_j(r(j))$.

The intuitive interpretation of the grouping above is given by the steps that we need to execute when jm is chosen to perform a best response operation. If jm is of type I(b), then jm can be saturated without any refusal made by j, since j has sufficient free quota. On the other hand, if j agrees to reduce $x(r(j))$ in order to accommodate more allocation on jm, then jm is a blocking edge of type I(a). The remaining case occurs when jm is not better than $r(j)$, that is, j accepts $\min\{\bar{x}(m), \bar{x}(j), \bar{x}(jm)\}$ allocation from m. In this case, no rejection is called by j.

In Phase I, only best blocking edges of type I(a) and I(b) are selected. Then, when only type II blocking edges remain, Phase II starts. In order to prove finite termination, we introduce two potential functions, $\Theta(x)$ and $\Psi(x)$. When proving termination of the first phase, both of them are used, while the second phase is discussed by analyzing the behavior of $\Psi(x)$ only.

The first function $\Theta(x)$ comprises two components. The first component is the sum consisting of the rank of refusal pointers at jobs. The second term is a sum consisting of the allocation value of refusal pointers at jobs. When we say that $\Theta(x)$ decreases, it is meant in the lexicographic sense. The second function, $\Psi(x)$ is a set of $|M|$ vectors, each of them corresponding to a machine. Each vector contains $|\delta(m)|$ entries, defined as $x(jm)$ for all $jm \in \delta(m)$, ordered as they appear in m's preference list. We denote these vectors by $\mathit{lex}(m)$, because $\mathit{lex}(m)$ increases lexicographically if and only if the lexicographic position of m improves. We added a minus sign in order to keep the terms decreasing. When we say that $\Psi(x)$ decreases we mean that at least one vector in it decreases lexicographically and no vector increases lexicographically.

This also implies that we could add up the i-th elements of these vectors and follow the lexicographic increment of the resulting vector. We choose not to do so for intuitive reasons, but the reader can also think of $\Psi(x)$ as a single vector of $\max_{m \in M} \deg(m)$ scalar components.

$$\Theta(x) := (\Theta_1(x), \Theta_2(x)) := \left(\sum_{j \in J} \text{rank}_j(r(j)), \sum_{j \in J} x(r(j)) \right)$$
$$\Psi(x) := -\left(lex(m_1), lex(m_2), ..., lex(m_{|M|}) \right)$$

Claim The best response step of job j along edge jm of type I(a) decreases $\Theta(x)$.

Proof Due to the type-defining characteristics listed above, there is a rejection on $r(j)$. If $x(r(j))$ becomes 0 through this step, then $\Theta_1(x)$ decreases, while $\Theta_2(x)$ might increase. Otherwise, if $x(r(j)) > 0$ holds even after executing the step, hence $\Theta_1(x)$ remains unchanged, but $\Theta_2(x)$ decreases. Any other decrement in x, such as allocation refused by m on $r(j')$ for some $j' \neq j$ can only further decrease both components of $\Theta(x)$. □

Claim The best response step of job j along edge jm of type I(b) decreases $\Psi(x)$ and does not increase $\Theta(x)$.

Proof Since j does not reject any allocation, $x(r(j))$ remains unchanged. If any other $r(j')$ for some $j' \neq j$ is affected, $\Theta(x)$ is decreased. The only machine whose position changes is m itself: it clearly improves its lexicographic position, thus one component of $\Psi(x)$ decreases, while the remaining vectors remain unchanged. □

For any rational input data, the changes in $\Theta(x)$ or $\Psi(x)$ in each round are bounded from below. Since both functions have an absolute minimum, Phase I terminates in finite time.

Claim The best response step of job j along edge jm of type II decreases $\Psi(x)$. Moreover, no edge becomes blocking of type I(a) or I(b).

Proof During the second phase, no machine loses allocation, thus, their lexicographic position cannot worsen. In addition, for the machine of the current blocking edge jm, $lex(m)$ improves. This also implies that no edge $j'm'$ dominates x at m' that has not already dominated it before the myopic change. Moreover, edges that lost allocation during that step are the worst-choice edges of j, hence they cannot be blocking of type I(a) or I(b). If there is an edge $j'm'$ that became blocking of type I(a) or I(b), then it is better than the worst edge in x at j'. These edges were already unsaturated before the last step and also already dominated x at both end vertices. This contradicts the fact that best blocking edges are chosen in each step. □

The same arguments as above, in Theorem 4, imply the result on random procedures.

Theorem 6 *In the rational case, random best response strategies terminate with a stable allocation with probability one.*

5 Irrational data: the accelerated algorithm

Both of our presented algorithms require exponentially many steps to terminate. Moreover, in our previous section we relied several times on the fact that in each step, x is changed with values greater than a specific positive lower bound. When irrational data are present, e.g., q, c or x are real-valued functions, this can no longer be guaranteed. Hence, our arguments for termination are no longer valid. As a matter of fact, some algorithmic ideas do not work with irrational input data, such as in the case of the well-known Ford-Fulkerson algorithm for finding a maximum flow, which fails when irrational capacities are present: the calculated flow might not even converge towards the maximum flow (Ford and Fulkerson 1962; Zwick 1995). In this section, we describe a fast version of our two-phase better response algorithm that terminates in polynomial time with a stable allocation for irrational input data as well. We also give a detailed proof of correctness for the first phase and show a construction with which all Phase II steps can be interpreted as Phase I operations in a slightly modified instance.

As usual in graph theory, an *alternating path* with respect to an allocation x is a sequence of incident edges that are saturated in x and of those that are unsaturated in x in an alternating manner.

5.1 Accelerated first phase

The algorithm and the proof of its correctness can be outlined in the following way (see also Algorithm 2 below). A helper graph is built in order to keep track of edges that may gain or lose some allocation. A potential function is also defined, which stores information about the structure of the helper graph and the degree of instability of the current allocation. In the helper graph we are looking for paths and cycles to augment along. The amount of allocation we augment with is specified in such a way that the potential function decreases and the helper graph changes. When using paths and cycles instead of proposal-refusal triplets, more than one myopic operation can be executed at a time. Moreover, we also keep track of consequences of locally myopic improvements. For example, we spare running time by avoiding reducing allocation on edges that later become blocking anyway.

First, we elaborate on the structure of the helper graph, define alternating paths and cycles and specify the amount of augmentation. The algorithm, the proof of its correctness, a pseudocode, and an example execution are all described in detail in this section.

Helper graph

Recall that our real-valued input $\mathcal{I}$ consists of a stable allocation instance (G, q, c, O) and a feasible allocation x. First, we define a helper graph $H(x)$ on the same vertices as G. This graph is dependent on the current allocation x and will be changed whenever we modify x. The edge set of $H(x)$ is partitioned into three disjoint subsets. The first subset $\mathcal{P}$ is the set of Phase I blocking edges. Each job j that has at least one edge

with positive x value, also has a worst allocated edge $r(j)$. When a myopic change is made, jobs tend to reduce x along exactly these edges. These *refusal pointers* form $\mathcal{R}$, the second subset of $E(H(x))$. We also keep track of edges that are currently not of blocking type I, but later on they may enter set $\mathcal{P}$. This last subset $\mathcal{P}'$ consists of edges that may become blocking of type I after some myopic changes. An edge $jm \notin \mathcal{P}$ has to fulfill three criteria in order to belong to $\mathcal{P}'$:

1. $c(jm) > x(jm)$;
2. m has at least one refusal edge, i.e., $\delta(m) \cap \mathcal{R} \neq \emptyset$;
3. $\text{rank}_j(jm) < \text{rank}_j(r(j))$.

Such an edge immediately becomes blocking of type I if m loses allocation along one of its refusal edges. Edges in $\mathcal{P}'$ are called *possibly blocking edges*, the set $\mathcal{P} \cup \mathcal{P}'$ forms the set of *proposal edges*. Note that a job j may have several edges in $\mathcal{P}$ and $\mathcal{P}'$, but at most one in $\mathcal{R}$. Moreover, if j has a proposal edge in $H(x)$, it also has an edge in $\mathcal{R}$. Regarding the machines, if m has a $\mathcal{P}'$-edge, it also has an $\mathcal{R}$-edge. Note that $(\mathcal{P} \cup \mathcal{P}') \cap \mathcal{R} = \emptyset$, because both $\mathcal{P}$ and $\mathcal{P}'$ per definition comprises edges that are ranked better by j than $r(j)$. The following lemma provides an additional structural property of $H(x)$.

Lemma 1 *If $jm \in \mathcal{P}$ and $j'm \in \mathcal{P}'$, then $\text{rank}_m(jm) < \text{rank}_m(j'm)$. That is, blocking edges are preferred to possibly blocking edges by their common machine m.*

Proof Since $jm \in \mathcal{P}$ is a blocking edge of type I, jm dominates x at m. If the statement is false, then $\text{rank}_m(jm) > \text{rank}_m(j'm)$ for some unsaturated edge $j'm$ that is better than the worst allocated edge of j'. Then also $j'm$ dominates x at m. This, together with the first and last properties of possibly blocking edges implies that $j'm \in \mathcal{P}$. □

Example 7 Once again we return to our example shown in Fig. 2. The only blocking edge j_3m_1 alone forms $\mathcal{P}$. The set $\mathcal{R}$ contains all four edges with positive allocation value: j_1m_1, j_2m_1, j_3m_2 and j_4m_3. Edges j_3m_3 and j_4m_2 are possibly blocking. Thus, in this case, $G = H(x)$.

Alternating paths and cycles

Our algorithm performs augmentations along alternating paths and cycles, so that the allocation value of refusal edges decreases, while the value of proposal edges increases. This is done in such a way that $\mathcal{R}$, $\mathcal{P}$, or $\mathcal{P}'$ (and thus, $H(x)$) changes. The main idea behind these operations is the same we used in the proof of Theorem 3: reassigning allocation to blocking edges from worse edges, such that the procedure is monotone. The difference between this method and the one presented in Sect. 4.1 is that, while our first algorithm tackles a single blocking edge in each step, here we deal with several blocking edges (forming the alternating path or cycle) at once.

When constructing the alternating proposal-refusal path or cycle ρ to augment along, the following rules have to be obeyed:

1. The first edge j_1m_1 is a $\mathcal{P}$-edge and it is the best proposal edge of m_1.
2. $\mathcal{P}$ and $\mathcal{P}'$-edges are added to ρ together with the refusal edge they are incident with on the active side.

3. Machines choose their best $\mathcal{P}$ or $\mathcal{P}'$-edge.
4. Walk ρ ends at m if

   (i) m has no proposal edge or
   (ii) ρ returns to its starting vertex, that is, $m = m_1$ or
   (iii) m's best proposal edge runs to a job already visited by ρ or
   (iv) m's best proposal edge runs to a job whose refusal pointer points to a machine already on ρ.

As long as there is a blocking edge of type I, the first edge j_1m_1 of such a path or cycle can always be found. Lemma 1 guarantees that if j_1m_1 is the best proposal edge of m_1, then $j_1m_1 \in \mathcal{P}$. After taking $r(j_1)$, all that remains is to continue on best proposal edges of machines and refusal edges of jobs they end at. Since $H(x)$ is a finite graph, ρ either terminates at a machine without any proposal edge or it visits a vertex already listed. These are exactly the cases listed in point 4. According to these rules, proposal-refusal edge pairs are added to the current path until I) there is no pair to add (4(i)) or II) the path reaches a vertex already visited (4(ii)-(iv)). In cases 4(i), 4(iii), and 4(iv), ρ is a path. In the remaining case 4(ii), ρ is a cycle.

Example 8 In our example in Fig. 2, ρ consists of the following edges: m_1j_3, j_3m_2, m_2j_4, j_4m_3. Even if $j_3m_3 \in \mathcal{P}'$, ρ halts at m_3, because j_3 is already on ρ.

Before elaborating on the amount of augmentation, we emphasize that ρ is just a *subset* of the set of edges whose x value changes during an augmentation step. The goal is to reassign allocation from refusal edges to blocking edges, until a stable solution is derived. Naturally, on an alternating path or cycle, refusal edges lose the same amount of allocation that proposal edges gain. However, if augmentations are performed along a path, the first machine m_1 on ρ gains allocation in total (and the last machine on ρ loses allocation). In order to preserve feasibility, m_1 might have to refuse allocation on edges not belonging to ρ. The exact amount of these refusals is discussed later, together with the amount of augmentation along ρ. Since no other vertex gains allocation in an augmentation step, feasibility cannot be violated elsewhere. Thus, these are the only edges not on ρ that need to be modified.

By contrast, if the augmentation is performed along a cycle C, refusals only happen on $r(j) \in C \cap \mathcal{R}$ edges. Even if the machine m_1 that started C has a full quota, it does not need to refuse any allocation, since $x(m_1)$ remains unchanged during the augmentation.

Amount of augmentation

Once ρ is fixed, the amount of allocation A to augment with has to be determined. It must be chosen so that (1) a feasible allocation is derived and (2) at least one refusal edge becomes empty or at least one proposal edge leaves $\mathcal{P} \cup \mathcal{P}'$. These points guarantee that $H(x)$ changes. To fulfill these two requirements, the minimum of the following terms is determined.

1. Allocation value on refusal edges along ρ: $x(r(j))$, where $r(j) \in \rho \cap \mathcal{R}$.
2. Residual capacity on proposal edges along ρ: $\bar{x}(p)$, where $p \in \rho \cap (\mathcal{P} \cup \mathcal{P}')$.

3. If ρ is a path, then m_1 may refuse a sufficient amount of allocation such that j_1m_1 does not become saturated, but it stops dominating x at m_1. In this case, the residual quota of m_1 must be filled up and, in addition, the sum of allocation values on edges worse than j_1m_1 must be refused. With this, j_1m_1 becomes the worst allocated edge of a full machine. Until reaching this point, j_1m_1 may gain $\bar{x}(m_1) + x(\text{edges dominated by } j_1m_1 \text{ at } m_1)$ amount of allocation in total.

To summarize, we augment with

$$A := \min\{x(r(j)), \bar{x}(p) | r(j) \in \rho \cap \mathcal{R},\, p \in \rho \cap (\mathcal{P} \cup \mathcal{P}')\}$$

if ρ is a cycle, because the last case with the starting vertex m_1 does not occur. Otherwise, the amount of augmentation is

$$A := \min\{x(r(j)), \bar{x}(p), \bar{x}(m_1) + x(\text{edges dominated by } j_1m_1 \text{ at } m_1) | \\ r(j) \in \rho \cap \mathcal{R},\, p \in \rho \cap (\mathcal{P} \cup \mathcal{P}')\}.$$

We now provide a pseudocode for our algorithm. To simplify notation, we refer to the vertices occurring on path ρ as $V(\rho)$.

Algorithm 2 Accelerated Phase I

```
while |P| > 0 do
  FindWalk(H(x))
  if ρ is a cycle then
    AugmentCycle(ρ)
  else
    AugmentPath(ρ)
  end if
  update R, P, P'
end while
```

```
procedure FindWalk(H(x))
  i := 1, ρ := ∅, find any m_1 ∈ M with a P-edge
  while m_i has a best proposal edge j_i m_i and r(j_i) ∩ (V(ρ) \ m_1) = ∅ do
    ρ := ρ ∪ {j_i m_i} ∪ {r(j_i)}
    if m_i ≠ m_1 then
      j_i m_{i+1} := r(j_i), i := i + 1
    end if
  end while
end procedure
```

```
procedure AugmentCycle(ρ)
  A := min{x(r(j)), x̄(p) | r(j) ∈ ρ ∩ R, p ∈ ρ ∩ (P ∪ P')}
  for p ∈ ρ ∩ (P ∪ P') do
    x(p) := x(p) + A
  end for
```

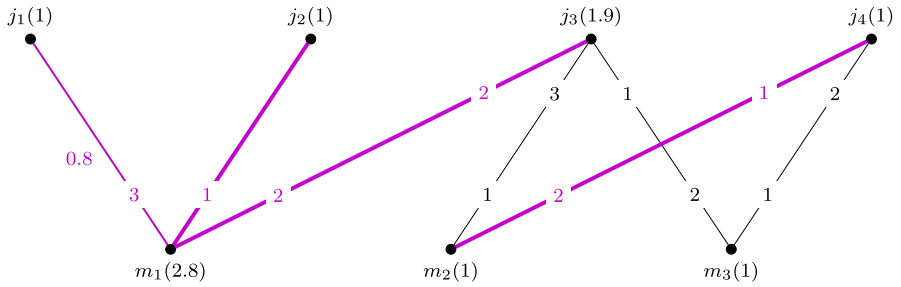

Fig. 5 After the first round of the accelerated Phase I algorithm

for $r(j) \in \rho \cap \mathcal{R}$ **do**
$x(r(j)) := x(r(j)) - A$
end for
end procedure

procedure AUGMENTPATH(ρ)
$A := \min\{x(r(j)), \bar{x}(p), \bar{x}(m_1) + x(\text{edges dominated by } j_1m_1 \text{ at } m_1) | r(j) \in \rho \cap \mathcal{R}, p \in \rho \cap (\mathcal{P} \cup \mathcal{P}')\}$
if $A - \bar{x}(m_1) > 0$ **then**
m_1 refuses $A - \bar{x}(m_1)$ allocation from its worst edges
end if
for $p \in \rho \cap (\mathcal{P} \cup \mathcal{P}')$ **do**
$x(p) := x(p) + A$
end for
for $r(j) \in \rho \cap \mathcal{R}$ **do**
$x(r(j)) := x(r(j)) - A$
end for
end procedure

Example 9 Recall the example instance in Fig. 2 again. Checking both proposal and both refusal edges on $\rho = \langle m_1j_3, j_3m_2, m_2j_4, j_4m_3\rangle$, the residual capacity of m_1, and the allocation on m_1's worst edges, one can compute that $A = 1$. Thus, allocation x shown in Fig. 5 is obtained after the first augmentation. Edge j_3m_1 leaves $\mathcal{P}$, and j_3m_3 enters it. The set of refusal edges consists of all edges with positive allocation value. $\mathcal{P}'$ is empty. In the second round, ρ is easy to find: it is $\langle m_3j_3, j_3m_1\rangle$. After reassigning allocation of value 1 to j_3m_3, Phase I ends. The allocation derived is not yet stable: j_1m_1 and j_3m_2 block it, but they are both of type II.

Note that executing several local myopic steps greedily, like in our first algorithm (Algorithm 1), would lead to a different output. A simple example for that can be seen on a slightly modified version of the first instance in Fig. 4, depicted in Fig. 6.

Example 10 Let us assume that $q(m_2) = N + 2$ and m_2 has an edge j_3m_2 ranked third, where $c(j_3m_2) = 1$. Let us start with the allocation $x(j_1m_1) = x(j_2m_2) = N, x(j_3m_2) = 1$. Edge $j_1m_2 \in \mathcal{P}$, so we can start ρ at m_1, augment along

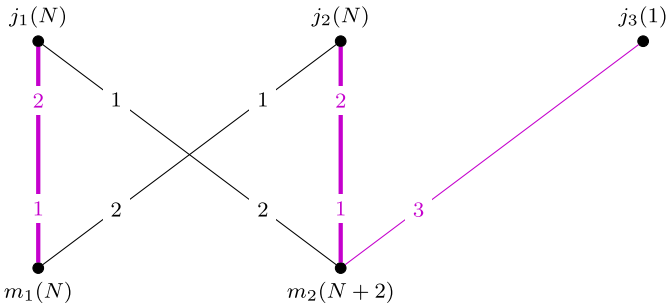

Fig. 6 An example for a cycle augmentation in the accelerated better response algorithm. In the accelerated version, j_3m_2 remains intact, while in the step-by-step version (Algorithm 1) it gets deleted and then added again

$\langle m_2j_1, j_1m_1, m_1j_2, j_2m_2\rangle$ with allocation value 1 and arrive at a stable solution. On the other hand, our step-by-step better response algorithm presented in Sect. 4.1 would first reject j_3m_2 fully, augment along the same cycle and, in Phase II, add j_3m_2 again.

It is easy to see that the emptying and then again adding edges like j_3m_2 can cause more superfluous rounds if the instance is more complex. Generally speaking, here we avoid m_1 refusing edges, knowing that it loses allocation later. As a result of that, m_1 would go under its quota, and would possibly create new blocking edges. Both strategies are better response, the difference is that our second algorithm keeps track of changes made as a consequence of a myopic operation.

We will later show that the first phase of our algorithm can be easily transformed into a second phase. We now turn to proving the correctness and running time of the first phase in detail.

Theorem 7 *For every real-valued allocation instance and given feasible allocation, there is a sequence of better responses leading to an allocation blocked by edges of type II exclusively. The sequence can be executed in $\mathcal{O}(|V|^2|E|)$ time.*

Proof **Potential function.** We show with the help of the following bicriteria potential function that the procedure is monotone and finite:

$$\Theta(x) := (\Theta_1(x), \Theta_2(x)) :=$$
$$\Big(\sum_{j\in J} \text{rank}_j(r(j)), \quad -\sum_{m\in M} \text{rank}_m(\text{best proposal edge at } m)\Big).$$

If $x(j) = 0$, then $\text{rank}_j(r(j))$ can be interpreted as a large number, for example as $|M| + 1$. In lack of proposal edges, the expression in the second component can also be interpreted as a large constant, for example as $|J| + 1$. In order to keep both terms decreasing, a minus sign is added to the second expression. When function $\Theta(x)$ decreases, it does so in the lexicographic sense.

Since $\Theta(x)$ is a bounded, integer-valued function, any procedure that modifies it strictly monotonically, is finite. We first show that each augmentation step strictly decreases $\Theta(x)$. Later, we elaborate on the running time of our algorithm.

Finiteness. The amount of allocation we augment with depends only on the extreme points of the min function determining the amount of augmentation A. Recall the three points we listed when defining A.

1. $x(r(j))$
   The worst allocated edge of j becomes empty. Thus, $\Theta_1(x)$ decreases unless j gains an edge in x that is worse than $r(j)$ was.
2. $\bar{x}(p)$
   If one of the proposal edges reaches its capacity, it stops being a proposal edge. Since p was the best proposal edge of its machine, $\Theta_2(x)$ decreases, unless p's machine gains a proposal edge better than p. Moreover, $\Theta_1(x)$ does not increase unless j gains an edge in x that is worse than $r(j)$ was.
3. In case of a path ρ: $\bar{x}(m_1) + x$(edges dominated by $j_1 m_1$ at m_1)
   The first blocking edge on ρ, m_1's best proposal edge ceases to dominate x at m_1, hence $\Theta_2(x)$ decreases, unless m_1 gains a proposal edge better than p. Moreover, $\Theta_1(x)$ does not increase unless j gains an edge in x that is worse than $r(j)$ was.

To summarize this, Θ strictly decreases after a round unless a job gets a worse refusal edge than before or a machine gets a better proposal edge than before without any refusal edge improving. The upcoming two claims eliminate the possibility of these two cases.

Claim In our algorithm, no job j gains an edge in x that is worse than $r(j)$.

Proof The only edges x is increased on are proposal edges in augmenting paths and cycles. These are by definition better than $r(j)$ in j's list. □

Claim If a machine m gains a proposal edge that is better than m's best proposal edge in a round of our algorithm, then the same round shifted the refusal edge of some job to m.

Proof Assume that $j'm$ became a better proposal edge of m than the previous best proposal edge jm was. Since the preference lists are fixed, this is only possible, if $j'm$ was not in $\mathcal{P} \cup \mathcal{P}'$ before. Proposal edges by definition have to fulfill three criteria, out of which at least one was not fulfille before the augmentation.

1. $j'm$ became unsaturated
   - One of the two possibilities for an edge to lose allocation occurs when $j'm \in \rho \cap \mathcal{R}$. Since $j'm$ is already the worst allocated edge of j', it cannot become a blocking edge of type I or possibly blocking, unless an even worse edge gains allocation, which is not possible, since it is only best proposal edges that do so.
   - Even if $j'm \notin \rho$, it can lose allocation, but only if $x(j'm)$ was reduced by $m = m_1$, the starting vertex of an alternating path. This $j'm$ is worse than jm, which contradicts our assumption.
2. $r(j')$ became worse than $j'm$
   Claim 5.1 shows that $r(j')$ never becomes worse during Phase I.

3. m gained an allocated edge worse than $j'm$ or $x(m)$ dropped below $q(m)$ (necessary for a blocking edge of type I) or
   m gained a refusal pointer (necessary for a possibly blocking edge)
   - In the first case, m increased x along an edge worse than $j'm$. This worse edge was m's best proposal edge, hence any better unsaturated edge from a job with worse edges in x was already in $\mathcal{P} \cup \mathcal{P}'$ too. Thus, $j'm$ cannot enter $\mathcal{P} \cup \mathcal{P}'$ solely because m gained an allocated edge worse than $j'm$. The previous two points eliminate the possibility of the two cases that could be combined with it in order to make $j'm$ enter $\mathcal{P} \cup \mathcal{P}'$.
   - If m lost some allocation, then it was the last vertex on ρ and thus, it had a refusal pointer. According to our definitions, any unsaturated edge of m from a job with worse edges in x was already in $\mathcal{P} \cup \mathcal{P}'$. The same applies as above, $j'm$ cannot enter $\mathcal{P} \cup \mathcal{P}'$ just because m lost allocation.
   - A refusal pointer moves to m, proving our statement.

□

Running time

The helper graph $H(x)$ has at most as many edges and vertices as G. In each iteration, $\Theta(x)$ improves. Consider first the case when only Θ_2 changes. The best proposal edge of each machine m can move along all $|\delta(m)|$ edges of m. Since the procedure is monotone, $|E|$ such steps can be executed in total, for the machines altogether. Then, Θ_1 has to improve. Just like Θ_2, Θ_1's monotone behavior also allows $|E|$ steps in total. Yet it is not possible that both components need all $|E|$ rounds. When a refusal pointer $r(j) = jm$ switches to a better edge jm', most of the elements in the sum for Θ_2 remain unchanged.

Upon shifting a refusal pointer, Θ_2 can clearly be increased, since only lexicographic monotonicity of $\Theta(x)$ can be shown. However, Claim 5.1 proves that shifting a single refusal pointer $r(j) = jm$ to jm' might cause m''s sole component in Θ_2 to increase by $|J| < |V|$ at most and leaves all other components in the sum unaffected.

This argumentation shows that the number of iterations can be bounded by $\mathcal{O}(|V||E|)$ from above. Next, we determine how much time is needed to execute a single augmentation. Procedure FINDWALK starts with choosing any machine that has a blocking edge of type I. This can be done in $\mathcal{O}(|V|)$ time. Adding the best proposal edge and the refusal pointer takes constant time, if they are stored for each vertex. Since at most one vertex is visited twice by the walk, after $\mathcal{O}(|V|)$ steps ρ is chosen. Then, either of the two augmenting procedures is called. It modifies x on at most $\mathcal{O}(|V|)$ edges. At last, $\mathcal{R}$, $\mathcal{P}$, and $\mathcal{P}'$ are updated, which involves at most $\mathcal{O}(|V|)$ edges.

In total, the algorithm performs $\mathcal{O}(|V||E|)$ rounds, each of which needs $\mathcal{O}(|V|)$ time to be computed. Thus the accelerated Phase I algorithm runs in $\mathcal{O}(|V|^2|E|)$ time.

Notice that our algorithm does not only take $\mathcal{O}(|V|^2|E|)$ steps as computation time, but each of the $\mathcal{O}(|V||E|)$ iterations also can be seen as a sequence of $\mathcal{O}(|V|)$ better response steps. If ρ is a path, then the first agent acting myopically is j_1 and the other jobs follow it in the order of their occurrence in ρ. Each agent simply sets the value of

its edges in ρ to the value calculated by our algorithm. If ρ is a cycle, then in this first step, j_1 increases $x(j_1m_1)$ by an arbitrary, small amount and decreases $x(r(j))$ so that it reaches its final value calculated by our algorithm. Notice that this is a lexicographic improvement. Now the last proposal edge added by the algorithm is blocking of type I and a better response step sets the value of both of its edges in ρ to the final value. We proceed backwards in the cycle in this manner, until we reach j_1 again, who increases $x(j_1m_1)$ to its final value and we are done. We admittedly do not argue that such a sequence of myopic changes is likely to occur without any central authority of control. Yet it proves that there is a polynomial-length better-response path to stability, nicely complementing our result that such a path for best responses does not exist in all markets.

Our method resembles the well-known notion of *rotations* (Gusfield and Irving 1989). They can be used when deriving a stable solution from another, by finding an alternating cycle of matching and non-matching edges and augmenting along them. In our algorithm, when we are searching for augmenting cycles or walks, we use an approach similar to rotations: jobs candidate their edges better than their worst edge in x, while machines choose the best one out of them. However, two differences can be spotted right away. While rotations are always assigned to a stable solution different from the job-optimal, our method works on an unstable input. Moreover, besides cycles we also augment along paths.

5.2 Accelerated second phase

The second phase can be accelerated in a very similar manner to the first phase. Instead of describing this new algorithm directly and proving its correctness using the same methods as above, we choose a simpler approach. The main idea in this section is that the accelerated second phase of our algorithm is actually the accelerated first phase of the same algorithm in a slightly modified instance. Thus, its correctness and running time have already been proved.

At the beginning of our argumentation we make modifications in the instance $\mathcal{I}$ given at termination of the accelerated Phase I algorithm. We show that the set of blocking edges of type I in the modified instance $\mathcal{I}'$ and the set of blocking edges of type II in $\mathcal{I}$ coincide. Then we let our accelerated Phase I algorithm run in $\mathcal{I}'$. At the end, we argue that its output is stable in $\mathcal{I}$.

Modified instance

After termination of the first phase, an allocation x_0 is given so that all blocking edges are of type II. This input of the second phase is modified in the following way. A dummy job j_d and edges between each machine and j_d are added to G. The capacity of these edges equals the maximum quota amongst all machines, and $q(j_d)$ is their sum. While j_d's preference list can be chosen arbitrarily, the new edges are ranked worst on the preference lists of the machines. The new graph is called G'. Not only the graph, but also the allocation x_0 is slightly modified: machines under their quota assign all their free quota to j_d. In this new allocation x_0' all machines are saturated. The

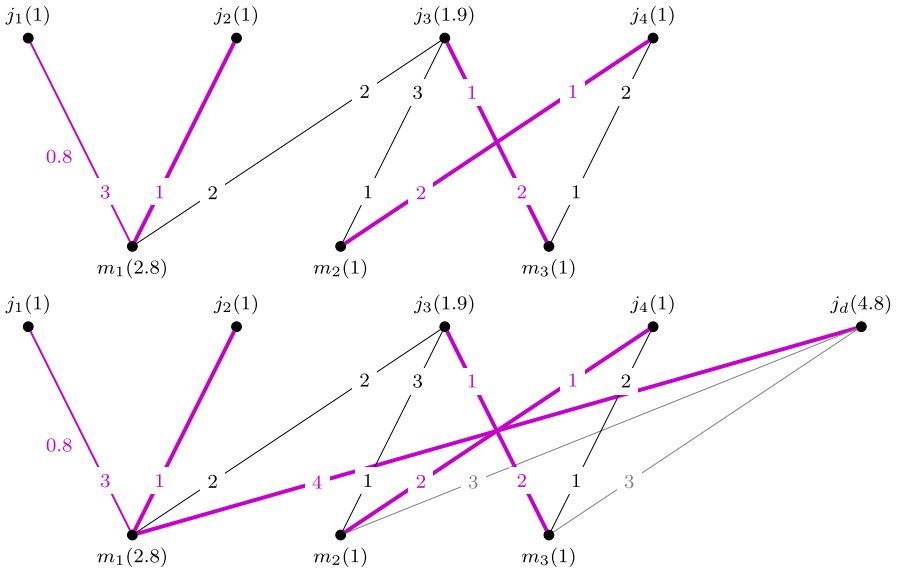

Fig. 7 The top figure depicts allocation x_0 in $\mathcal{I}$, denoted by colored edges, while allocation x'_0 in $\mathcal{I}'$ is denoted by colored edges in the bottom figure

new instance $\mathcal{I}'$ consists of G', q', c', O' and x'_0. An example instance modification is illustrated in Fig. 7.

As mentioned above, our goal is to perform Phase I operations in $\mathcal{I}'$. In order to be able to do so, we swap the two sides: jobs play a passive role, while machines become the active players. Since each active vertex has a filled up quota, all blocking edges are of type I in $\mathcal{I}'$.

Note that $\mathcal{I}'$ was constructed in such a way that—regardless of the type of blocking—each edge blocking x also blocks x' and vice versa. This is due to the fact that the only difference between the two instances is that machines' free quota appears as allocation on their worst edge in $\mathcal{I}'$. The definition of a blocking edge does not distinguish between those two notions. In particular, given a specific allocation x_0 with no blocking edge of type I, the set of (Phase II) blocking edges in $\mathcal{I}$ and the set of (Phase I) blocking edges in $\mathcal{I}'$ trivially coincide.

Let us denote the output of the accelerated Phase I algorithm in $\mathcal{I}'$ by x', and its restriction to $E(G)$ by x.

Claim Allocation x is stable in $\mathcal{I}$.

Proof Suppose edge jm blocks x. In $\mathcal{I}'$, jm is unsaturated and dominates x' at both end vertices, hence jm blocks x' as well. Since x' is the output of the accelerated Phase I algorithm in $\mathcal{I}'$, jm is of type II. Our goal is to show by induction that $x'(m) = q(m)$ for all machines. Thus, a contradiction is derived, because in $\mathcal{I}'$ no Phase II blocking edge can occur.

Initially, $x'(m) = q(m)$ for all machines. The key property of x'_0 is that all unsaturated edges that dominate x'_0 at their (saturated) machine are not better than their

job's worst edge in x'_0. Otherwise, they would be blocking edges of type I for x_0. Augmenting along a blocking edge jm in x'_0 can therefore never result in a refusal by the passive vertex j. Thus, after the first round, $x'(m) = q(m)$ still holds. Alternating walks are chosen in such a manner that jobs increase x' only on their best proposal edges. This guarantees that even after the first round, if jm dominates the current allocation x'_1 at m, it is not better than j's worst edge in x'_1. From here on, induction applies. □

The running time of this phase cannot exceed the running time of the accelerated Phase I algorithm, since the size of $\mathcal{I}'$ does not exceed the size of $\mathcal{I}$ significantly.

With this we finished the proof of the following result.

Theorem 8 *For every allocation instance and given feasible allocation x, there is a sequence of better responses that leads to a stable allocation in $\mathcal{O}(|V|^2|E|)$ time.*

Conclusion and open questions

We solved the problem of uncoordinated processes in stable allocation instances algorithmically. Our first method is a deterministic better response algorithm that finds a stable solution through executing myopic steps. In case of rational input data, the existence of such an algorithm guarantees that random better response strategies terminate with a stable solution with probability one. Analogous results are shown for best response dynamics. We also prove that random best response strategies terminate in expected polynomial time on correlated markets, even in the presence of irrational data. An accelerated version of our first, better-response algorithm is provided as well. For any real-valued instance, it terminates after $O(|V|^2|E|)$ steps with a stable allocation. We also show a counterexample for a possible acceleration for the case of best response dynamics.

Future research may involve more complex stability problems from the paths-to-stability point of view. For example, any of the problem variants listed in Sect. 1.2 can be combined with our general setting of allocations.

Acknowledgements We would like to thank Péter Biró, Tamás Fleiner, and our reviewers for their valuable comments.

References

Abraham DJ, Levavi A, Manlove DF, O'Malley G (2008) The stable roommates problem with globally-ranked pairs. Int Math 5:493–515

Ackermann H, Goldberg PW, Mirrokni VS, Röglin H, Vöcking B (2011) Uncoordinated two-sided matching markets. SIAM J Comput 40:92–106

Baïou M, Balinski M (2002) Erratum: the stable allocation (or ordinal transportation) problem. Math Oper Res 27:662–680

Blum Y, Roth AE, Rothblum UG (1997) Vacancy chains and equilibration in senior-level labor markets. J Econ Theory 76:362–411

Blum Y, Rothblum UG (2002) "Timing is everything" and marital bliss. J Econ Theory 103:429–443

Chen B, Fujishige S, Yang Z (2016) Random decentralized market processes for stable job matchings with competitive salaries. J Econ Theory 165:25–36

Dean BC, Munshi S (2010) Faster algorithms for stable allocation problems. Algorithmica 58:59–81
Diamantoudi E, Miyagawa E, Xue L (2004) Random paths to stability in the roommate problem. Games Econ Behav 48:18–28
Fleiner T (2014) On stable matchings and flows. Algorithms 7:1–14
Ford LR, Fulkerson DR (1962) Flows in Networks. Princeton University Press, Princeton
Gale D, Shapley LS (1962) College admissions and the stability of marriage. Am Math Mon 69:9–15
Goemans MX, Li EL, Mirrokni VS, Thottan M (2006) Market sharing games applied to content distribution in ad hoc networks. IEEE J Sel Areas Commun 24:1020–1033
Gusfield D, Irving RW (1989) The Stable Marriage Problem: Structure and Algorithms. MIT Press, Cambridge
Hoefer M, Wagner L (2014) Matching dynamics with constraints. In: Liu T-Y, Qi Q, Ye Y (eds) 10th International Conference on Web and Internet Economics (WINE). Springer International Publishing, pp 161–174
Klaus B, Klijn F (2007) Paths to stability for matching markets with couples. Games Econ Behav 58:154–171
Knuth D (1997) Mariages stables. Les Presses de L'Université de Montréal, 1976. English translation in Stable Marriage and its Relation to Other Combinatorial Problems, volume 10 of CRM Proceedings and Lecture Notes. American Mathematical Society
Kojima F, Ünver M (2008) Random paths to pairwise stability in many-to-many matching problems: a study on market equilibration. Int J Game Theory 36(3–4):473–488
Millán B, Risma EP (2018) Random path to stability in a decentralized market with contracts. Soc Choice Welf 51(1):79–103
Roth AE, Vande Vate J H (1990) Random paths to stability in two-sided matching. Econometrica 58:1475–1480
Wang X, Agatz N, Erera A (2017) Stable matching for dynamic ride-sharing systems. Transp Sci 52(4):850–867
Zwick U (1995) The smallest networks on which the Ford–Fulkerson maximum flow procedure may fail to terminate. Theor Comput Sci 148(1):165–170