J Comb Optim (2016) 32:657–671
DOI 10.1007/s10878-015-9889-3

Improved algorithmic results for unsplittable stable allocation problems

Ágnes Cseh[1] · **Brian C. Dean**[2]

Published online: 5 May 2015

Abstract The stable allocation problem is a many-to-many generalization of the well-known stable marriage problem, where we seek a bipartite assignment between, say, jobs (of varying sizes) and machines (of varying capacities) that is "stable" based on a set of underlying preference lists submitted by the jobs and machines. Building on the initial work of Dean et al. (The unsplittable stable marriage problem, 2006), we study a natural "unsplittable" variant of this problem, where each assigned job must be fully assigned to a single machine. Such unsplittable bipartite assignment problems generally tend to be NP-hard, including previously-proposed variants of the unsplittable stable allocation problem (McDermid and Manlove in J Comb Optim 19(3): 279–303, 2010). Our main result is to show that under an alternative model of stability, the unsplittable stable allocation problem becomes solvable in polynomial time; although this model is less likely to admit feasible solutions than the model proposed in McDermid and Manlove (J Comb Optim 19(3): 279–303, McDermid and Manlove 2010), we show that in the event there is no feasible solution, our approach computes a solution of minimal total congestion (overfilling of all machines collectively beyond their capacities). We also describe a technique for rounding the solution of a stable allocation problem to produce "relaxed" unsplit solutions that are only mildly infeasible, where each machine is overcongested by at most a single job.

✉ Ágnes Cseh
cseh@math.tu-berlin.de

Brian C. Dean
bcdean@clemson.edu

[1] Institute for Mathematics, TU Berlin, Strasse des 17. Juni 136, 10623 Berlin, Germany

[2] School of Computing, Clemson University, Box 340974, Clemson, SC 29634-0974, USA

Keywords Stable matchings · Stable allocations · Rotations · Unsplittable assignments

1 Introduction

Consider a bipartite assignment problem over a graph $G = (V = J \cup M, E)$ involving the assignment of a set of jobs J to a set of machines M. Each job $j \in J$ has a processing time $q(j)$, each machine $m \in M$ has a capacity $q(m)$, and there is a capacity $c(jm)$ for each edge $jm \in E$ governing the maximum amount of job j that can be assigned to machine m. A feasible allocation of jobs to machines is described by a function $x : E \to \mathbb{R}_{\geq 0}$ such that

1. $0 \leq x(jm) \leq c(jm)$ for all edges $jm \in E$,
2. $x(j) := \sum_{m \in M} x(jm) \leq q(j)$ for all jobs $j \in J$, and
3. $x(m) := \sum_{j \in J} x(jm) \leq q(m)$ for all machines $m \in M$.

If $x(jm) \in \{0, q(j)\}$ for all $jm \in E$, we say the allocation is *unsplit*, since each assigned job is assigned in its entirety to a single machine. We often forgo the use of edge capacities $c(jm)$ when discussing unsplit allocations, since an edge jm can simply be deleted if $c(jm) < q(j)$.

Problems of the form above have been extensively studied in the algorithmic literature, where typical objectives are to find a feasible assignment or one of maximum weight (maximizing a linear objective function $\sum_{jm \in E} w(jm)x(jm)$, with $w(jm)$ being the weight of edge jm). While the fractional (splittable) variants of these problems are easy to solve in polynomial time via network flow techniques, it is NP-hard to find an unsplit allocation of either maximum total size $|x| = \sum_{jm \in E} x(jm)$ or of maximum weight; the former is a variant of the multiple subset sum problem (Caprara et al. 2000), and the latter is known as the multiple knapsack problem (Chekuri and Khanna 2005).

In contrast to problems with explicit edge costs, the *stable allocation problem* is an "ordinal" problem variant where the quality of an allocation is expressed in a more game theoretic setting via ranked preference lists submitted by the jobs and machines, with respect to which we seek an assignment that is *stable* (defined shortly). In this paper, we study the stable allocation problem in the unsplittable setting, which was shown to be NP-hard in (McDermid and Manlove 2010) using one natural definition for stability. We show here that by contrast, a different and more strict notion of stability, proposed initially in Dean et al. (2006), leads to an $O(|E|)$ algorithm for the unsplit problem. The tradeoff is that under this different notion of stability, it is unlikely that feasible allocations will exist. However, we show that by relaxing the problem to allow mildly infeasible allocations, our algorithm computes a "relaxed" unsplit stable allocation (in which each machine is filled beyond its capacity by at most the allocation of a single job) in which the total amount of overcongestion across all machines, $\sum_{m \in M} \max(0, x(m) - q(m))$, is minimized (so in particular, if there is a feasible allocation with no congestion, we will find it).

Through the work of several former authors (Dinitz et al. 1999; Skutella 2000; Shmoys and Tardos 1993), the "relaxed" model has become relatively popular in the

context of unsplittable bipartite assignment and unsplittable flow problems. The standard approximation algorithm framework (finding an approximately-optimal, feasible solution) typically does not fit these problems, since finding any feasible solution is typically NP-hard. Instead, authors tend to focus on pseudo-approximation results with minimal congestion per machine or per edge. Analogous results were previously developed for unsplit stable allocation problems in Dean et al. (2006), where an unsplit stable allocation can be found in linear time in which each machine is overcongested by at most a single job. The model of stability proposed in Dean et al. (2006) is the one we further develop in this paper, and among all of these prior approaches (including those for standard unsplittable bipartite assignment and flows), it seems to be the only unsplit model studied to date in which minimization of *total* congestion is possible in polynomial time. Hence, there is a substantial algorithmic incentive to consider this model, even though its notion of stability is less natural than in McDermid and Manlove (2010).

The classical stable marriage problem, perhaps the simplest relative of our problem in the domain of ordinal matching, is known to satisfy a number of remarkable mathematical properties. For example, one can always find stable solutions that are "left-hand-side optimal" or "right-hand-side optimal", and the exact same subset of left-hand side and right-hand side elements are matched in every stable solution (the so-called "rural hospital" theorem, named after applications involving the assignment of medical residents to hospitals). We show natural generalizations of all of these structural properties in our relaxed unsplit stable allocation setting (further justifying the utility of this model from a mathematical perspective). For example we show how to compute in $O(|E|)$ time a "job-optimal" allocation that maximizes the total size $|x|$ of all assigned jobs, and a "machine-optimal" allocation that minimizes $|x|$. It is this machine-optimal solution that we show also minimizes total congestion. In order to produce potentially other solutions (e.g., that might be more fair to both sides), we show also a technique for "rounding" a solution of the fractional stable allocation problem to obtain a relaxed unsplit solution.

We review preliminary concepts and background material in the next section, then introduce our structural and algorithmic results for computing relaxed unsplit solutions maximizing or minimizing $|x|$, showing how these can be used to solve the unsplittable stable allocation problem in linear time. Finally, we discuss our rounding method for producing additional relaxed unsplit assignments.

2 Background and preliminaries

2.1 Stable matching and allocation problems

Stable marriage The stable marriage (or stable matching) problem takes place on a bipartite graph with men on one side and women on the other, where each individual submits a strictly-ordered, but possibly incomplete preference list of the members of the opposite sex. The goal is to find a matching that is *stable*, containing no *blocking pair*—an unmatched (man, woman) pair (m, w) where m is either unmatched or prefers w to his current partner, and likewise for w.

In their seminal paper (1962), Gale and Shapley describe a simple $O(|E|)$ algorithm to find a stable matching for any instance. The most typical incarnation of their algorithm generates a solution that is "man-optimal" and "woman-pessimal", where each man is matched with the best possible partner he could receive in any stable matching, and each woman is matched with the worst possible partner she could receive in any stable matching. By reversing the roles of the men and women, the algorithm can also generate a solution that is simultaneously woman-optimal and man-pessimal.

Stable allocation The stable allocation problem was introduced by Baïou and Balinski (2002) as a high-multiplicity variant of the stable matching problem, where we match non-unit elements with non-unit elements—here, we speak of matching jobs of varying size with machines of varying capacity. Just as before, jobs and machines submit strict preferences over their outgoing edges in the bipartite assignment graph. If job $j \in J$ prefers machine $m_1 \in M$ to machine $m_2 \in M$, we write $\text{rank}_j(jm_1) > \text{rank}_j(jm_2)$. A stable allocation in this setting is a feasible allocation (as defined in the introduction) where for every edge $jm \in E$ with $x(jm) < c(jm)$, either j is fully assigned to machines at least as good as m, or m is fully assigned to jobs at least as good as j. That is, there can be no *blocking edge* jm where $x(jm) < c(jm)$ and both j and m would prefer to use more of jm. We say that edges with positive x value are in x. If any machine m has $q(m) > \sum_{j \in J} c(jm)$, then $q(m)$ is set to $\sum_{j \in J} c(jm)$. Machines with $x(m) = q(m)$ are *saturated*. Later, when $x(m) > q(m)$ occurs in the relaxed version of the problem, we talk about *over-capacitated* machines. If any job prefers machine m to any of its allocated machines, then m is called *popular*, otherwise m is *unpopular*. Note that all popular machines must be saturated in any stable allocation.

The stable allocation problem can be solved in $O(|E| \log |V|)$ time (Dean and Munshi 2010). There can be many different solutions for the same instance, but they all have the same total allocation $|x|$, and even stronger, the values of $x(j)$ and $x(m)$ for each job and machine remain unchanged across all stable allocations. This holds for both stable marriage (Gale and Sotomayor 1985) and stable allocation (Baïou and Balinski 2002), moreover, even for stable roommate (Gusfield and Irving 1989), the non-bipartite version of the problem, and is known as the *rural hospital theorem*. A common application of stable matching in practice is the National Resident Matching Program (NRMP) (Roth 2008), where medical school graduates in the USA are matched with residency positions at hospitals via a centralized stable matching procedure. A consequence of the rural hospital theorem is that if a less-preferred (typically rural) hospital cannot fill its quota in some stable assignment, then there is no stable assignment in which its quota will be filled.

Like the stable marriage problem, one can always find job-optimal, machine-pessimal and job-pessimal, machine-optimal solutions. To define these notions for the stable allocation problem, Baïou and Balinski (2002) define an order on stable solutions based on a *min–min criterion*, where a job j prefers allocation x_1 to allocation x_2 if $x_1(jm) < x_2(jm)$ implies $x_1(jm') = 0$ for every jm' worse than jm for j. A similar relation can be defined for machines as well. Stable matchings and stable allocations both are known to form distributive lattices with an ordering relation based on the min–min criterion.

2.2 Unsplittable stable allocation problems

An unsplit allocation x satisfies $x(jm) \in \{0, q(j)\}$ for all $jm \in E$, so each assigned job is assigned in its entirety to one machine. For simplicity, we introduce a "dummy" machine m_d with high capacity, which acts as the last choice for every job. This lets us assume without loss of generality that an unsplittable allocation always exists in which every job is assigned. In this context, we define the size $|x|$ of an allocation so that jobs assigned to m_d do not count, since they are in reality unassigned. In addition to the application of scheduling jobs in a non-preemptive fashion, a motivating application for the unsplittable stable allocation problem is in assigning personnel with "two-body" constraints. For example, in the NRMP, a married pair of medical school graduates might act as an unsplittable entity of size 2 (this particular application has been studied in substantial detail in the literature; see Biró and Klijn 2013 for further reference).

From an algorithmic standpoint, one of the main results of this paper is that how we define stability in the unsplit case seems quite important. In McDermid and Manlove (2010), the following natural definition was proposed: an edge jm is blocking if j prefers m to its current partner, and if m prefers j over $q(j)$ units of its current allocation or unassigned quota. Unfortunately, it was shown in McDermid and Manlove (2010) that this definition makes the computation of an unsplit stable allocation NP-hard. We therefore consider an alternative, stricter notion of stability where edge jm is blocking if j prefers m to its current partner, and if m prefers j over *any amount* of its current allocation or has free quota. That is, if j would prefer to be assigned to m over its current partner, than m must be saturated with jobs that m prefers to j. Aside of the integrity constraint, this definition is fully aligned with the classical definition of a stable allocation. As in the splittable case, popular machines must therefore be saturated. Practice shows (Roth 1996) that if a hospital is willing to hire one person in a couple, but it has no free job opening for the partner, it is most likely amenable to make room for both applicants. Therefore, our definition of a blocking pair serves practical purposes.

Relaxed unsplit allocations The downside of our alternative definition of stability is that it is unlikely to allow feasible unsplit stable allocations to exist in most large instances. Therefore, we consider allowing mildly-infeasible solutions where each machine can be over-capacitated by a single job—a model popularized by previous results in the approximation algorithm literature for standard unsplittable assignment problems (Dinitz et al. 1999; Skutella 2000; Shmoys and Tardos 1993), and introduced in the context of unsplittable stable allocation by Dean et al. (2006). Specifically, we say that $x : E \to \mathbb{R}_{\geq 0}$ is a *relaxed unsplit allocation* if $x(jm) \in \{0, q(j)\}$ for every edge $jm \in E$, $x(j) \leq q(j)$ for every job $j \in J$, and for each machine m, the removal of the least-preferred job assigned to m would cause $x(m) < q(m)$.[1] Our definition of stability extends readily to the relaxed setting, and we would argue that it is perhaps

[1] The model introduced in Dean et al. (2006) allows $x(m) \leq q(m)$, but we believe strict inequality is actually a better choice—mathematically and from a modeling perspective. For example, the old definition applied to a hospital–resident matching scenario with married couples might cause a hospital to accept two more residents than its quota, while the new definition would only require accepting one more resident. The results in Dean et al. (2006) hold with either definition.

the most natural mathematical notion of stability to consider in this setting (whereas the form of stability in McDermid and Manlove (2010) is probably the most natural for the hard capacity setting). We say a relaxed unsplit allocation x is *stable* if for every edge jm with $x(jm) = 0$, either j is assigned to a machine that j prefers to m, or m's quota is filled or exceeded with jobs that m prefers to j. Otherwise, if edge jm with $x(jm) = 0$ is preferred by j to its allocated machine and m's quota is not filled up with better edges than jm, then jm *blocks* x.

Note that the relaxed unsplit model differs from the non-relaxed unsplit model with capacities inflated by $\max q(j)$, since stability is still defined with respect to original capacities. It may be best to regard "capacities" here as constraints governing start time, rather than completion time of jobs. A machine below its capacity is always willing to launch a new job, irrespective of job size.

3 Machine-optimal relaxed unsplit allocations

In Dean et al. (2006), a version of the Gale–Shapley algorithm is described to find the job-optimal relaxed unsplit stable allocation x_{jopt}. In this context, job-optimal means that there is no relaxed unsplit stable allocation x' such that any job is assigned to a better machine in x' than in x_{jopt}. The implementation described in Dean et al. (2006) runs in $O(|E||V|\log|V|)$ time, but $O(|E|)$ is also easy to achieve. In this section, we show how to define and compute a *machine-optimal* relaxed unsplit stable allocation x_{mopt} also in $O(|E|)$ time, and we prove the following:

Theorem 1 *Among all relaxed unsplit stable allocations x, $|x|$ is maximized at $x = x_{jopt}$ and minimized at $x = x_{mopt}$.*

One of the main challenges with computing a machine-optimal allocation is defining machine-optimality. In the stable allocation problem, existence of a machine-optimal allocation follows from the fact that all stable allocations form a distributive lattice under the standard min–min ordering relationship introduced in Baïou and Balinski 2002. However, this ordering seems to depend crucially on the existence of a rural hospital theorem, which no longer holds in the relaxed unsplit case, since relaxed unsplit stable allocations may differ in cardinality (Fig. 1). Even an appropriately relaxed version of the rural hospital theorem seems difficult to formulate over relaxed instances: machines can be saturated or even over-capacitated in one relaxed unsplit stable allocation, while being empty in another one (Fig. 1). Nonetheless, we can still prove a result in the spirit of the rural hospital theorem, which we discuss further in Sect. 3.3.

Without an "exact" rural hospital theorem, comparing two allocations using the original min–min ordering seems problematic, and indeed one can construct instances where two relaxed unsplit stable allocations are incomparable according to this criterion (Fig. 1). We therefore adopt a different but nonetheless natural ordering relation: *lexicographical order*. We say that machine m prefers unsplit allocation x_1 to allocation x_2 if the best edge in $x_1 \triangle x_2$ belongs to x_1, where $\triangle$ denotes the symmetric difference operation. The opposite ordering relation is based on the position of jobs, and since jobs are always assigned to machines in an unsplit fashion, the lexicographic

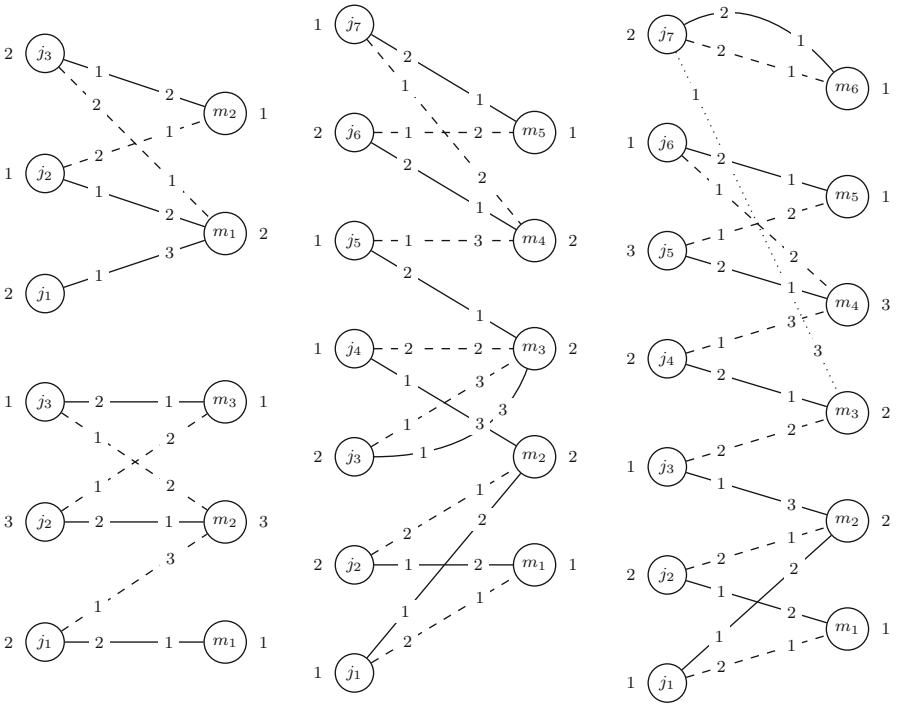

Fig. 1 The *upper-left* instance admits two relaxed unsplit allocations differing in cardinality: The *dashed* edges form a stable allocation of size 3, while the remaining edges build another stable allocation of size 5. The *lower-left* example is evidence against an exact rural hospital theorem, where m_1 is empty in one relaxed unsplit stable allocation (given by the *dashed* edges) but filled beyond its capacity in another (given by the *solid edges*). The graph in the *middle* shows two relaxed unsplit allocations that are incomparable from the perspective of m_3. The last instance is a counterexample showing the difficulty of formulating join and meet operations

and min–min relations are actually the same from the job's perspectives; hence, "job optimal" means the same thing under both. The lexicographical position of the same agent in different allocations can always be compared, and we say a relaxed stable allocation x is *machine-optimal* if it is at least as good for all machines as any other relaxed stable allocation (although we still need to show that such a allocation always exists).

3.1 The reversed Gale–Shapley algorithm

For the classical stable marriage problem, the Gale–Shapley algorithm can be reversed easily, with women proposing instead of men, to obtain a woman-optimal solution. We show that this idea can be generalized (carefully accounting for multiple assignment and congestion among machines) to compute a machine-optimal relaxed unsplit stable allocation. Pseudocode for the algorithm appears in Fig. 2.

Claim The algorithm terminates in $O(|E|)$ time.

1: $x(jm_d) := q(j)$ for all $j \in J$, $x(jm) := 0$ for every other $jm \in E$
2: **while** $\exists m : x(m) < q(m)$ with a non-empty preference list **do**
3: m proposes to its best job j with value $q(j)$
4: **if** j prefers m to its current partner **then**
5: $x(jm) := q(j)$
6: $x(jm') := 0$ for $\forall m' \neq m$
7: **end if**
8: delete j from m's preference list
9: **end while**

Fig. 2 Reversed relaxed unsplit Gale–Shapley algorithm

Proof In each step, a job is deleted from a machine's preference list.

Claim The algorithm produces an allocation x that is a relaxed unsplit stable allocation.

Proof First, we check the three feasibility constraints for x. Since proposals are always made with $q(j)$ and refusals are always full rejections, the quota constraints of the jobs may not be violated. Moreover, each job is assigned to exactly one machine. Machines can be over-capacitated, but deleting the worst job from their preference list results in an allocation under their quota. Otherwise the machine would not have proposed along the last edge. If x is unstable, then there is an empty edge jm blocking x. During the execution, m must have proposed to j. This offer was rejected, because j already had a better partner in the current allocation. Since jobs monotonically improve their position in the allocation, this leads to a contradiction.

Claim The output x is the machine-optimal relaxed unsplit stable allocation (i.e., no machine has a better lexicographical position in any other relaxed unsplit stable allocation).

Proof Assume that there is a relaxed unsplit stable allocation x', where some machines come better off than in x. To be more precise, in the symmetric difference $x \triangle x'$, the best edge incident to these machines belongs to x'. When running the reversed relaxed unsplit Gale–Shapley algorithm, there is a step when the first such edge jm_1 carries a proposal from m_1 but gets rejected. Otherwise, m_1 filled up or exceeded its quota in x with only better edges than jm_1. Let us consider only this edge first and denote the feasible, but possibly unstable relaxed allocation produced by the algorithm so far by x_0.

When j refused jm_1, it already had a partner m_0 in x_0, better than m_1. Even if there is no guarantee that $jm_0 \in x$, it is sure that $jm_0 \notin x'$ and jm_0 does not block x', though $\text{rank}_j(jm_0) > \text{rank}_j(jm_1)$ for $jm_1 \in x'$. It is only possible if m_0 is saturated or over-capacitated in x' with edges better than jm_0. Since $jm_0 \in x_0$, x_0 may not contain all of these edges, otherwise m_0 is congested in x_0 beyond the level required for a relaxed unsplit allocation. During the execution of the reversed relaxed unsplit Gale–Shapley algorithm, m_0 proposed along all of these edges and got rejected by at least one of them. This edge is never considered again, it may not enter x later. Thus, jm_1 is not the first edge in $x' \setminus x$ that was rejected in the algorithm.

With this, we completed the constructive proof of the following theorem:

Theorem 2 *The machine-optimal relaxed unsplit stable allocation x_{mopt} can be computed in $O(|E|)$ time.*

3.2 Properties of the job- and machine-optimal solutions

Theorem 3 *The job-optimal relaxed unsplit stable allocation x_{jopt} is the machine-pessimal relaxed unsplit stable allocation and vice versa, the machine-optimal relaxed unsplit stable allocation x_{mopt} is the job-pessimal relaxed unsplit stable allocation.*

Proof We start with the first statement. Suppose that there is a relaxed unsplit stable allocation x' that is worse for some machine m than x_{jopt}. This is only possible if m's best edge jm in $x_{\text{jopt}} \triangle x'$ belongs to x_{jopt}. Since x_{jopt} is the job-optimal solution, jm', j's edge in x' is worse than jm. But then, m is saturated or over-capacitated in x' with better edges than jm. We assumed that all edges in x' that are better than jm are also in x_{jopt}. Thus, omitting m's worst job from x_{jopt} leaves m at or over its quota.

The second half of the theorem can be proved similarly, using the reversed Gale–Shapley algorithm. Assume that there is a relaxed unsplit stable allocation x' that assigns some jobs to worse machines than x_{mopt} does. Let us denote the set of edges preferred by any job to its allocated machine in x' by E'. Due to our indirect assumption, E' contains some edges of x_{mopt}. When running the reversed Gale–Shapley algorithm on the instance, there is an edge $jm \in E'$ that is the first edge in E' carrying a proposal. Since j is not yet matched to a better machine, it also accepts this offer. Even if $jm \notin x_{\text{mopt}}$, j's edge in x_{mopt} is at least as good as m, because jobs always improve their position during the course of the reversed Gale–Shapley algorithm. On the other hand, m cannot fulfill its quota in x_{mopt} with better edges than jm, simply because the proposal step along jm took place.

Since $jm \notin x'$, but j prefers jm to its edge in x', m is saturated or over-capacitated with better edges than jm in x'. As observed above, not all of these edges belong to x_{mopt}. Let us denote one of them in $x' \setminus x_{\text{mopt}}$ by $j'm$. Before proposing along jm, m submitted an offer to j' that has been refused. The only reason for such a refusal is that j' has already been matched to a better machine m'. But since $j'm \in x'$, $j'm' \in E'$. This contradicts to our indirect assumption that jm is the first edge in E' that carries a proposal.

Theorem 1 also follows from the proof above.

We note that although we can compute the job-optimal and machine-optimal relaxed unsplit stable allocations, there in general does not appear to be an obvious underlying lattice structure behind relaxed unsplit solutions. For stable matching or fractional stable allocation, computing the meet or join of two solutions is fairly easy. In order to reach the join (meet) of x_1 and x_2, all machines (jobs) choose the better edge set out of those two allocations (Fleiner 2010). The example in Fig. 1 illustrates that this property does not carry over to relaxed unsplit allocations. If all jobs chose the better allocation, m_3 remains empty and j_7m_3 becomes blocking. Similar examples can easily be constructed to show that choosing the worse allocation also can lead to instability.

Our ability to compute x_{mopt} in $O(|E|)$ time now gives us a linear-time method for solving the (non-relaxed) unsplittable stable allocation problem (according to our, stricter notion of stability).

Lemma 1 *If an instance $\mathcal{I}$ admits an unsplit stable assignment x, then the machine-optimal relaxed unsplit stable assignment x_{mopt} on the corresponding relaxed instance $\mathcal{I}'$ is also an unsplit stable assignment on $\mathcal{I}$.*

Proof Suppose the statement is false, e.g. although there is an unsplit stable assignment x, x_{mopt} is no unsplit stable assignment on $\mathcal{I}$. This can be due to two reasons: either the feasibility or the stability of x_{mopt} is harmed on $\mathcal{I}$. The latter case is easier to handle. An allocation that is feasible on both instances and stable on $\mathcal{I}'$ may not be blocked by any edge on $\mathcal{I}$, since the set of unsaturated edges is identical on both instances. The second case, namely if x_{mopt} violates some feasibility constraint on $\mathcal{I}$, needs more care.

$\mathcal{I}$ and $\mathcal{I}'$ differ only in the constraints on the quota of machines. If x_{mopt} is infeasible on $\mathcal{I}$, then there is a machine m for which $x_{\text{mopt}}(m_1) > q(m_1)$. Regarding the unsplit stable assignment x, the inequality $x(m_1) \leq q(m_1)$ trivially holds. Now we use Theorem 1 for x and x_{mopt} that are both relaxed unsplit stable assignments on $\mathcal{I}'$. This corollary implies that if there is a machine m_1 with $x_{\text{mopt}}(m_1) > x(m_1)$, then another machine m_2 exists for which $x_{\text{mopt}}(m_2) < x(m_2)$ holds.

This machine m_2 plays a crucial role in our proof. It has a lower allocation value in the machine-optimal relaxed solution x_{mopt} than in another relaxed stable solution x on $\mathcal{I}$. Its lexicographical position can only be better in x_{mopt} than in x if the best edge j_2m_2 in $x \triangle x_{\text{mopt}}$ belongs to x_{mopt}. Moreover, $x \triangle x_{\text{mopt}}$ also contains an edge $j_3m_2 \in x$, otherwise $x_{\text{mopt}}(m_2) > x(m_2)$. Naturally, $\text{rank}_m(j_2m_2) < \text{rank}_m(j_3m_2)$. At this point, we use the property that $x_{\text{mopt}}(m_2) < q(m_2)$. Since m_2 has free quota in x_{mopt} and j_3m_2 is not a blocking edge, j_3 must be matched to a machine better than m_2 in x_{mopt}. Thus, there is a job that comes better off in the machine-optimal (and job-pessimal) relaxed solution than in another relaxed stable solution. This contradiction to Theorem 3 finishes our proof.

Lemma 1 shows that if there is an unsplit solution, it can be found in linear time by computing the machine-optimal relaxed solution. Unfortunately, the existence of such an unsplit assignment is not guaranteed. Our next result applies to the case when no feasible unsplit solution can be found. In terms of congestion, with the machine-optimal solution we come as close as possible to feasibility.

Theorem 4 *Amongst all relaxed unsplit stable solutions, x_{mopt} has the lowest total congestion.*

Proof Let M_u denote the set of machines that remain under their quota in x_{mopt}. Note that $\sum_{m \notin M_u} x_{\text{mopt}}(m)$, the total allocation value on the remaining machines clearly determines the total congestion of x_{mopt}, given by $\sum_{m \notin M_u} x_{\text{mopt}}(m) - q(m)$. Let x be an arbitrary relaxed solution. Due to Theorem 1, the total allocation value is minimized at x_{mopt}. Therefore, for any relaxed unsplit stable allocation x, the following inequalities hold:

$$\sum_{m\in M} x(m) \geq \sum_{m\in M} x_{\text{mopt}}(m)$$

$$\sum_{m\notin M_u} x(m) + \sum_{m\in M_u} x(m) \geq \sum_{m\notin M_u} x_{\text{mopt}}(m) + \sum_{m\in M_u} x_{\text{mopt}}(m)$$

$$\sum_{m\notin M_u} x(m) - \sum_{m\notin M_u} x_{\text{mopt}}(m) \geq \sum_{m\in M_u} x_{\text{mopt}}(m) - \sum_{m\in M_u} x(m)$$

$$\sum_{m\notin M_u} (x(m) - q(m)) - \sum_{m\notin M_u} (x_{\text{mopt}}(m) - q(m)) \geq \sum_{m\in M_u} x_{\text{mopt}}(m) - \sum_{m\in M_u} x(m)$$

At this point, we investigate the sign of both sides of the last inequality. The core of our proof is to show that for each $m \in M_u$ and relaxed stable solution x, $x_{\text{mopt}}(m) \geq x(m)$. This result, proved below, has two benefits. On one hand, the term on the right hand-side of the last inequality is non-negative. Therefore, the inequality implies that the total congestion on machines in $M \setminus M_u$ is minimized at x_{mopt}. On the other hand, no machine in M_u is over-capacitated in any relaxed solution. Thus, the total congestion is minimized at x_{mopt}.

Our last observation in this subsection refers to the unsaturated machines.

Lemma 2 *For every $m \in M_u$ and relaxed solution x, the inequality $x_{mopt}(m) \geq x(m)$ holds.*

Proof Suppose that there is a machine $m \in M_u$ for which $x_{\text{mopt}}(m) < x(m)$ for some relaxed solution x. Since m is unsaturated in x_{mopt}, it is unpopular. On the other hand, there is at least one job j for which $jm \in x \setminus x_{\text{mopt}}$. As m is unpopular in x_{mopt}, j is allocated to a better machine in x_{mopt} than in x. Since x_{mopt} is the job-pessimal solution, we derived a contradiction.

3.3 A variant of the "Rural Hospital" theorem

In the relaxed unsplit case, one can find counterexamples against an exact rural hospital theorem (e.g., where all machines have the same amount of allocation in all relaxed unsplit allocations) or even a weakened theorem stating that all unsaturated / congested machines have the same status in all relaxed unsplit allocations. Lemma 2 above however suggests an alternative variant of "rural hospital" theorem that does hold.

Theorem 5 *A machine m that is not saturated in x_{mopt} will not be saturated in every relaxed unsplit stable solution, and a machine m that is over-capacitated in x_{jopt} must at least be saturated in every relaxed unsplit stable solution.*

Proof The first part is shown by Lemma 2. For the second part, consider a machine m that is over-capacitated in x_{jopt} but has $x(m) < q(m)$ in some relaxed unsplit allocation x. Consider any job j in $x_{\text{jopt}} \setminus x$, and note that since x_{jopt} is job-optimal, j prefers m to its partner in x. Hence, jm blocks x.

As of the jobs' side, Theorem 3 already guarantees that if a job is unmatched in x_{jopt}, then it is unmatched in all relaxed stable solutions and similarly, if it is matched in x_{mopt}, then it is matched in all relaxed stable solutions.

4 Rounding algorithms

We have seen now how to compute x_{jopt} and x_{mopt} in linear time. We now describe how to find potentially other relaxed unsplit solutions by "rounding" solutions to the (fractional) stable allocation problem. For example, this could provide a heuristic for generating relaxed unsplit solutions that are more balanced in terms of fairness between the jobs and machines. Our approach is based on augmentation around *rotations*, alternating cycles that are commonly used in stable matching and allocation problems to move between different stable solutions (see, e.g., Dean and Munshi 2010; Gusfield and Irving 1989).

We begin with a stable allocation x with $x(j) = q(j)$ for every job j, thanks to the existence of a dummy machine. For each job j that is not fully assigned to its first-choice machine, we define its *refusal edge* $r(j)$ to be the worst edge jm incident to j with $x(jm) > 0$. Jobs with refusal edges also have *proposal edges*—namely all their edges ranked better than $r(j)$. Recall that a machine with incoming proposal edges is said to be *popular*. We call a machine *dangerous* if it is over-capacitated and has zero assignment on all its incoming proposal edges.

Claim Consider a popular machine m in some fractional stable allocation x. Amongst all proposal edges incoming to m, at most one has positive allocation value in x, and this positive proposal edge is ranked lower on m's preference list than any other edge into m with positive allocation.

Proof Let $\text{rank}_m(j_1m) > \text{rank}_m(j_2m)$ be proposal edges such that $x(j_1m)$ and $x(j_2m)$ are both positive. Note that j_1m blocks x, since j_1 and m have worse allocated edges in x. A similar argument implies the last part of the claim.

Our algorithm proceeds by a series of augmentations around rotations, defined as follows. We start from a popular, non-dangerous machine m (if no such machine exists, the algorithm terminates, having reached an unsplit solution). Since m is popular and non-dangerous, it has incoming proposal edges with positive allocation, and due to the preceding claim, it must have exactly one such edge jm. We include jm as well as j's refusal edge jm' in our partial rotation, then continue building the rotation from m' (again finding an incoming proposal edge, etc.). We continue until we close a cycle, visiting some machine m visited earlier (in which case we keep just the cycle as our rotation, not the edges leading up to the cycle), or until we reach a machine m that is unpopular or dangerous, where our rotation ends.

To enact a rotation, we increase the allocation on its proposal edges by ε and decrease along the refusal edges by ε, where ε is chosen to be as large as possible until either (i) a refusal edge along the rotation reaches zero allocation, or (ii) a dangerous machine at the end of the rotation drops down to being exactly saturated from being over-capacitated, and hence ceases to be dangerous. We call case (i) a "regular" augmentation. This concludes the algorithm description.

Claim The algorithm terminates after $O(|E|)$ augmentations.

Proof Jobs remain fully allocated during the whole procedure, and their lexicographical positions never worsen. With every regular augmentation, some edge stops being

a refusal edge, and will never again be increased or serve as a proposal or refusal edge. We can therefore have at most $O(|E|)$ regular augmentations. Furthermore, a machine can only become dangerous if one of its incoming refusal pointers reaches zero allocation, so the number of newly-created dangerous machines over the entire algorithm is bounded by $|E|$. Hence, the number of non-regular augmentations is at most $O(|M| + |E|) = O(|E|)$.

Claim The final allocation x is a feasible relaxed unsplit assignment.

Proof Since we start with a feasible assignment and jobs never lose or gain allocation, the quota condition on jobs cannot be harmed. If there is any edge jm with $0 < x(jm) < q(j)$, then j has at least two positive edges, the better one must be a positive proposal edge. This contradicts the termination condition, and hence x is unsplit.

We now show that deleting the worst job from each machine results in an allocation strictly below the machine's quota. It is clearly true at the beginning, where no machine is over-capacitated (since x starts out as a feasible stable allocation). The only case when $x(m)$ increases is when m is the first machine on a rotation. As such, m has a positive proposal edge jm, which is also its worst allocated edge, due to our earlier claim. If m is not over-capacitated when choosing the rotation, then even if $x(jm)$ rises as high as $q(j)$, this increases $x(m)$ by strictly less than $q(j)$. Thus, deleting jm, the worst allocated edge of m, guarantees that $x(m)$ sinks under $q(m)$. If m is saturated or over-capacitated when choosing the rotation, then jm would have been the best proposal edge of m earlier, when $x(m)$ was not greater than $q(m)$. Thus, assigning j entirely to m does not harm the relaxed quota condition. Let us consider the last step as $x(m)$ exceeded $q(m)$. Again, m was the starting vertex of an augmenting path, having a positive proposal edge. If it was jm, our claim is proved. Otherwise m became over-capacitated while $x(jm)$ was zero, and then increased the allocation on jm. But between those two operations, m had to become dangerous, because it switched its best proposal edge to jm. Dangerous machines never start alternating paths. Thus, we have a contradiction to the fact that we considered the last step when $x(m)$ exceeded $q(m)$.

Claim The final allocation x is stable.

Proof Suppose some edges block x. Since we started with a stable allocation, there was a step during the execution of the algorithm when the first edge jm became blocking. Before this step, either j or m was saturated or over-capacitated with better edges than jm. The change can be due to two reasons: either j gained allocation on an edge worse than jm, or m gained allocation on an edge worse than jm. As already mentioned, j's lexicographical position never worsens: $\text{rank}_j(p) > \text{rank}_j(r(j))$ always holds. The second event also may not occur, because machines always play their best response strategy. An edge jm that becomes blocking when allocation is increased on an edge worse than it, was already a proposal edge before. Thus, m would have chosen jm, or an edge better than jm to add it to the augmenting path.

Since each augmentation requires $O(|V|)$ time and there are $O(|E|)$ augmentations, our rounding algorithm runs in $O(|E||V|)$ total time. If desired, dynamic tree data structures can be used (much like in Dean and Munshi 2010) to augment in $O(\log |V|)$ time, bringing the total time down to just $O(|E| \log |V|)$.

Although jobs improve their lexicographical position in each rotation, the output of the algorithm is not necessarily x_{jopt}. In fact, even x_{mopt} can be reached via this approach. Ideally, this approach can serve as a heuristic to generate many other relaxed unsplit stable allocations, if run from a variety of different initial stable solutions x.

5 Conclusion and open questions

In our present work, we reformulated the definition of stable unsplit allocations. Several basic properties of stable matchings and allocations are discussed on the relaxed setting. Most of them carry over to unsplit allocations, but we also showed examples for certain structural properties that do not hold in the unsplittable case.

We proved that the reversed relaxed unsplit Gale–Shapley algorithm can be used to decide in polynomial time whether a regular instance admits an unsplit stable assignment. If not, relaxed solutions can be searched for. Besides constructing the job-optimal and the machine-optimal solutions, we also showed a method that rounds any fractional stable solution to a relaxed unsplit stable allocation.

On the other hand, we mentioned that the well-known rural hospital theorem has no generalization for unsplit assignments: relaxed stable solutions can have different cardinality and the same vertex can have various lexicographical position in them. The set of relaxed solutions also has been investigated: the distributive lattice structure known for matchings and allocations cannot be observed here. Although rotations can be used to derive an unsplit solution from a fractional one, moving form one unsplit solution to another one is impossible just by rotations along cycles.

This latter obstacle raises a problem about optimizing over the set of solutions. If the instance contains cost on edges, how to find a minimum-value stable solution? Rounding the optimal fractional assignment does not necessarily lead to an optimal unsplit allocation. A similar question can be addressed about fair allocations. Aside from these, any combination with well-known notions in stability problems can be studied: ties, restricted edges, etc. Another straightforward generalization would be to define k-splittable allocations and investigate their properties.

Acknowledgments This work was partially supported by the Deutsche Telekom Stiftung, the Deutsche Forschungsgemeinschaft within the research training group Methods for Discrete Structures (GRK 1408), and by the USA National Science Foundation CAREER award CCF-0845593.

References

Baïou M, Balinski M (2002) The stable allocation (or ordinal transportation) problem. Math Oper Res 27(3):485–503

Biró P, Klijn F (2013) Matching with couples: a multidisciplinary survey. Int Game Theory Rev (IGTR) 15(02):1340008-1–1340008-18

Caprara A, Kellerer H, Pferschy U (2000) A PTAS for the multiple subset sum problem with different knapsack capacities. Inf Process Lett 73(3–4):111–118

Chekuri C, Khanna S (2005) A polynomial time approximation scheme for the multiple knapsack problem. SIAM J Comput 35(3):713–728

Dean B, Goemans M, and Immorlica N (2006) The unsplittable stable marriage problem. In: IFIP TCS, volume 209 of IFIP, pp 65–75. Springer:Berlin

Dean B, Munshi S (2010) Faster algorithms for stable allocation problems. Algorithmica 58(1):59–81
Dinitz Y, Garg N, Goemans M (1999) On the single-source unsplittable flow problem. Combinatorica 19:17–41
Fleiner T (2010) On stable matchings and flows. In: Proceedings of the 36th International Workshop on Graph-Theoretic Concepts in Computer Science, WG'10, pp 51–62
Gale D, Shapley L (1962) College admissions and the stability of marriage. Am Math Mon 1:9–14
Gale D, Sotomayor M (1985) Some remarks on the stable matching problem. Discrete Appl Math 11:223–232
Gusfield D, Irving R (1989) The stable marriage problem: structure and algorithms. MIT Press, Cambridge
McDermid E, Manlove D (2010) Keeping partners together: algorithmic results for the hospitals/residents problem with couples. J Comb Optim 19(3):279–303
Roth A (1996) The national residency matching program as a labor market. J Am Med Assoc 275(13):1054–1056
Roth A (2008) Deferred acceptance algorithms: history, theory, practice, and open questions. Int J Game Theory 36(3–4):537–569
Shmoys D, and Tardos E (1993) Scheduling unrelated machines with costs. In: Proceedings of SODA, pp 448–454
Skutella M (2000) Approximating the single source unsplittable min-cost flow problem. In: Proceedings of FOCS, pp 136–145