A Market Clearing Solution for Social Lending

Ning Chen[†] **Arpita Ghosh**[‡]
[†]Division of Mathematical Sciences, Nanyang Technological University, Singapore.
ningc@ntu.edu.sg
[‡]Yahoo! Research, Santa Clara, CA, USA.
arpita@yahoo-inc.com

Abstract

The social lending market, with over a billion dollars in loans, is a two-sided matching market where borrowers specify demands and lenders specify total budgets and their desired interest rates from each acceptable borrower. Because different borrowers correspond to different risk-return profiles, lenders have preferences over acceptable borrowers; a borrower prefers lenders in order of the interest rates they offer to her. We investigate the question of what is a computationally feasible, 'good', allocation to clear this market.

We design a strongly polynomial time algorithm for computing a Pareto-efficient stable outcome in a two-sided many-to-many matching market with indifferences, and use this to compute an allocation for the social lending market that satisfies the properties of stability — a standard notion of fairness in two-sided matching markets — and Pareto efficiency; and additionally addresses envy-freeness amongst similar borrowers and risk diversification for lenders.

1 Introduction

Social lending, or peer-to-peer lending, which allows individuals to lend and borrow money to each other directly without the participation of banks, is an exploding business on the Internet: the total amount of money borrowed using such peer-to-peer loans was approximately $\$650$ million in 2007, and is projected to reach $\$5.8$ *billion* by 2010.

The social lending market consists of borrowers seeking some target loan amount (their demand), and lenders seeking to invest some fixed amount of money in loans (their budget). Lenders usually prefer to invest their budget in multiple borrowers' loans to spread risk from defaulting borrowers, and each borrower's loan is also usually funded by multiple lenders. Borrowers are 'non-homogeneous' — different borrowers have different characteristics such as credit rating and desired loan length, and command different interest rates based on their creditworthiness. That is, different borrowers correspond to investments with different risk-return profiles. As a result, lenders have *preferences* over which borrowers they would like to lend their money to — one lender may prefer high-risk-high-return borrowers, while another may prefer safe borrowers albeit fetching lower interest rates. On the other hand, borrowers also have implicit preferences over lenders since different lenders can offer different interest rates to the same borrower: a borrower simply prefers the lenders who offer him the lowest rate.

We now have a large two-sided matching market where agents on both sides have multiunit capacities, and preferences — lenders have preference rankings (possibly with ties) over the set of acceptable borrowers they're willing to lend to, and borrowers have preferences over lenders based on their offered interest rates. But the preferences of all these agents may be conflicting — many lenders may compete to lend to the same borrower who is their common top choice, while this borrower's preference might be an entirely different lender who offers him a lower interest rate, who in turn has a different top borrower choice. Clearly, it need not be possible to make all agents simultaneously happy, raising the natural question of what constitutes a 'good' assignment. In this paper, we investigate the social lending market from a computational social choice perspective: what is a fair and efficient way to clear this marketplace, and how can it be computed?

Our contributions. We present a model ($\S 2$) for the social lending marketplace based on Zopa (www.zopa.com), which, with over $400,000$ members and $\pounds 100$ million in its markets and over $\pounds 100000$ traded *each day*, is the first and among the largest social lending sites on the Web.

We first address the question of what is a desirable allocation in the social lending market in $\S 3$, and argue that an allocation that is stable, Pareto efficient, fair amongst 'equal' borrowers, and also addresses the need for risk diversification to reduce default risk is a desirable solution concept in this market. We then address the question of finding an algorithm that returns such an allocation in $\S 4$.

When preference lists contain ties, as in our social lending context, not all stable matchings are Pareto efficient. The question of how to find a Pareto-stable matching when preferences contain ties has recently been addressed for the *many-to-one matching* model in [Erdil and Ergin, 2008; 2009]. A naive adaptation of this algorithm to our *many-to-many* market returns a Pareto-stable assignment in time that scales with the total capacity of all nodes in the graph, *i.e.*, the amount of money traded in the market, requiring us to de-

velop a new algorithmic approach. We design a strongly polynomial time algorithm for computing a Pareto-stable outcome in a many-to-many matching market with indifferences, and apply it to a reduced marketplace where 'identical' borrowers are grouped into equivalent classes or 'categories'; we then reallocate amongst categories to achieve fairness amongst borrowers and risk diversification. The overall runtime is polynomial in the number of lenders and the number of borrower categories, which is a small constant (=10) for the Zopa marketplace.

Related work. Social lending is a relatively new application that has only recently begun to be addressed in the research literature, starting with the work in [Freedman and Jin, 2009] on default rates. Most of the research on social lending takes an empirical approach as in [Freedman and Jin, 2009]; [Chen *et al.*, 2009] analyzes the auction held for a single borrower's loan in the Prosper market, but does not address the several coexisting lenders and borrowers in the marketplace. To the best of our knowledge, social lending has not been studied much from a marketplace design or social choice perspective.

There is a very vast literature on two-sided matching markets and stable matching; for a review of the economics literature on the subject, see [Roth and Sotomayor, 1992; Roth, 2008]; for an introduction to the computer science literature addressing algorithmic and computational questions, see, e.g., [Gusfield and Irving, 1989; Iwama and Miyazaki, 2008]. The paper most relevant to our work from the stable matching literature is [Erdil and Ergin, 2008; 2009], who study the algorithmic question of finding Pareto-stable matchings for a *many-to-one* matching market; see §4. The many-to-many setting is far less well studied in the stable matching literature, and focuses largely on structural results in settings without indifferences; see, e.g., [Hatfield and Kominers, 2010; Echenique and Oviedo, 2006].

2 A Lending Market Model

We model the social lending marketplace M as a bipartite graph with nodes (A, B) and edges E. The nodes in A represent the lenders and the nodes in B are the borrowers. Nodes on both sides have multiunit capacities: a lender i's capacity c_i is her budget, the total amount of money she wants to lend. A borrower j's capacity c_j is his demand, the total loan amount he wants to borrow. We will assume that the capacities are integers by expressing them in the smallest unit of currency. The edge set E of M is the set of pairs (i, j) where lender i is willing to lend to borrower j.

Each lender specifies the interest rates at which she is willing to lend money to different acceptable borrowers; as in Zopa, this is the actual interest-rate that she will receive on any loans to that borrower. Note that each lender can offer different interest rates to different borrowers, and the same borrower can be offered different interest rates by different lenders.

Every acceptable borrower, along with the specified interest rate, represents a possible investment for the lender, with a particular risk-return profile. Each lender has a *preference* list P_i ranking the investments corresponding to these borrower-interest rate pairs, *i.e.*, its neighbors $\{j \in B : (i, j) \in E\}$. These preferences need not be strict and the lender can be indifferent between, *i.e.*, equally prefer, two different investments; that is, preference lists can have ties. Since the preference list is restricted to i's neighbors (*i.e.*, acceptable borrowers), it is naturally incomplete. (As an example, lender i's preferences, denoted $P_i = ([j_1, j_2], [j_3, j_4, j_5])$, could be as follows: i is indifferent between j_1 and j_2, and prefers both of them to j_3, j_4, j_5 among all of whom i is indifferent; she finds all other borrowers unacceptable.) In general, a lender can also have preferences over sets of investments; however, here we will restrict ourselves to expressing preferences over individual investments for simplicity[1].

Each borrower j has an implicit *preference* ranking P_j over lenders based on the interest rates they offer him: j prefers lenders in non-increasing order of offered interest rates, and is indifferent amongst lenders who offer him the same interest-rate (so borrowers' preferences P_j can contain ties as well).

We partition the set of borrowers into equivalence classes, or *categories* $C = \{C_1, \ldots, C_m\}$: two borrowers are equivalent, *i.e.*, belong to the same category, if no lender can distinguish between them based on the information available about them in the marketplace. Thus, all lenders are indifferent between the borrowers in a category, and offer them the same interest-rate. This also means that borrowers in a category all have the same preferences over lenders, and that a lender's preference list need only rank categories, not individual borrowers. The number of categories can be as large as the number of borrowers when *personal* information is provided by/about each borrower (as in Prosper), or very small when the only information revealed is the credit-rating and loan length (as in Zopa, which only allows lenders to specify interest rates for 10 borrower categories).

Diversification to decrease default risk is a very important factor in social lending. Instead of modeling this into the preferences of lenders, we deal with it as Zopa— Zopa breaks up each lender's budget into small sums each of which is lent to a different borrower to diversify risk, so we will similarly ensure that each lender's budget is uniformly spread amongst many different borrowers in the final allocation.

We note that we do not model reserve rates, nor the dynamic aspect of the social lending market in this work.

Feasible assignments. The output of the market is a multiunit pairing, or *assignment* $X = (x_{ij})_{(i,j)\in E}$ between A and B, where $x_{ij} \in \mathbb{N} \cup \{0\}$ is the number of units assigned from $i \in A$ to $j \in B$ (when $c_i = c_j = 1$ for all nodes, an assignment reduces to a *matching*). An assignment X is *feasible* if it simply satisfies capacity constraints on both sides, *i.e.*, $\sum_j x_{ij} \leq c_i$ and $\sum_i x_{ij} \leq c_j$. Note that the preferences P_i, P_j do not matter to the feasibility of an assignment.

3 What Is a Good Outcome?

Having defined the set of feasible assignments, how do we choose one from amongst the very large number of possible assignments? An ideal solution concept for the social lending market would be Pareto efficient, fair — both across lenders

[1] This is both for technical tractability and to avoid eliciting complex combinatorial preferences from lenders.

and borrowers, as well as amongst similar borrowers — *exist* for every instance of the input, and be efficiently computable (since social lending markets transact huge amounts of money, it should be implementable in time that depends only on the number of agents, not on the money being traded in the marketplace). What assignment has these properties?

A very widely used solution concept in two-sided matching markets is that of *stability* [Gale and Shapley, 1962]: there is no pair of individuals that *both* strictly prefer each other to some partner they are currently assigned to (such a pair would be called a blocking pair). Stability can be interpreted as a notion of fairness in our context — while it is not possible to guarantee each lender her most preferred allocation, a stable allocation is fair in the sense that if a lender indeed sees a better allocation, that allocation does not also 'prefer' her in return. However, when preference lists contain ties as in our case, it is well-known that stable matchings need not be *Pareto efficient*, even when all nodes have unit capacity [Roth and Sotomayor, 1992], as the following example shows.[2] Therefore, we will need to explicitly require that the solution is both stable and Pareto efficient; such assignments are called *Pareto-stable* assignments [Sotomayor, 2009]. However, as the next example shows, applying the concept of Pareto stability directly to the marketplace M may not produce very desirable solutions: a solution may well be Pareto-stable, but hand out very different interest rates to two identical borrowers, violating our fairness requirement.

Example 3.1. *There are two lenders i_1, i_2 and two borrowers j_1, j_2 with two units of supply/demand each. Both lenders are indifferent between the two borrowers; the first lender i_1 offers 7% to both borrowers, and the second lender i_2 offers 15% to both borrowers. The matching where i_1 lends both units to j_1 (at 7%), and i_2 lends both units to j_2 (at 15%) is stable and also Pareto efficient. However, the matching where both lenders lend one unit to each borrower is also Pareto-stable and 'more fair' since both j_1 and j_2 get equal amounts of the low and high interest rates (note that this matching does not Pareto-improve the previous matching since it makes j_1 strictly worse off). In addition, each lender spreads her loan across more borrowers, so diversity improves as well.*

This example illustrates that we cannot simply apply the solution concept of Pareto stability directly to the social lending marketplace M. Instead, we will consider a modified market, where the lender side is unchanged but borrowers are aggregated by category into 'meta-borrowers', with demand equal to the aggregate demand of that category (recall that a category can consist of a single borrower, when there is plenty of information about borrowers in the marketplace). We will first find a Pareto-stable assignment in this *reduced* marketplace— how to find such an assignment is the key technical problem we need to solve— and then distribute each lender's allocation to a meta-borrower amongst all the borrowers in that category to ensure envy-freeness amongst them.

We note that another natural solution concept, maximum size assignment (*i.e.*, the one with the largest trade volume), is unsuitable here since it ignores all agents' preferences; the maximum-size *stable* assignment is NP-hard to compute [Iwama *et al.*, 1999]. However, it is easy to show that the size of any stable assignment (and therefore also our assignment) is at least half the size of the maximum size assignment.

The assignment we propose to clear the social lending market is the following.

> MARKET CLEARING ASSIGNMENT: Given a lending market $M = (A, B)$ with categories $C = \{C_1, \ldots, C_m\}$:
>
> 1. Create a meta-borrower for each category C_r with demand $\sum_{j \in C_r} c_j$, and the same preferences as those of borrowers in C_r. Denote the resulting market by (A, C).
> 2. Compute a Pareto-stable assignment $X^* = (x^*_{iC_r})$ for (A, C), where $x^*_{iC_r}$ is lender i's total investment in category C_r (§4).
> 3. Assign each lender's investment $x^*_{iC_r}$ across all borrowers in category C_r to ensure diversity and envy-freeness; denote the final assignment by $Y = (y_{ij})$ (§4).

We have the following result about this assignment.

Theorem 3.2 (Main)**.** *The final assignment $Y = (y_{ij})$ can be computed in time $O(|A|^4 + |A||B|)$ and has the following properties:*

1. *Stability: There are no blocking pairs in the original marketplace $M = (A, B)$.*
2. *Pareto efficiency: No agent in M can be made better off without making some other agent in M strictly worse off.*
3. *(Weak) envy-freeness: No borrower envies the allocation of any other borrower in its category.*
4. *Diversity: Given the allocations $X^* = (x^*_{iC_r})$, each lender i spreads her budget amongst the maximum number of distinct borrowers.*

4 Algorithm

We will first address the problem of efficiently finding a Pareto stable assignment in an abstract two-sided many-to-many matching market with separable responsive preferences, and then apply the algorithm we develop to the modified marketplace with lenders and meta-borrowers.

We begin with some formal definitions. Recall that we have a two-sided matching market $M = (A, B)$ with preference lists P_k and multi-unit capacities c_k for all agents k, and k's preference over sets is the natural (partial) order defined by the preferences P_k over individuals as in [Erdil and Ergin, 2008; 2009]. We can assume without loss of generality that $|A| = |B| = n$ by adding dummy isolated nodes with $c_k = 0$ to the market.

Definition 4.1 (Level function)**.** *We use the function $L_i(\cdot)$ to encode the preference list of a node $i \in A$. For each $j \in P_i$, let $L_i(j) \in \{1, \ldots, n\}$ denote the* ranking *of j in i's preference list. Therefore, for any $j, j' \in P_i$, if $L_i(j) < L_i(j')$, then i strictly prefers j to j'; if $L_i(j) \leq L_i(j')$, then i weakly*

[2] A simple example consists of two men and two women, where i_1 strictly prefers j_1 to j_2, but all other nodes are indifferent amongst their possible partners. The matching $(i_1, j_2), (i_2, j_1)$ is stable, but not Pareto efficient since i_1 can be reassigned to j_1 and i_2 to j_2 without making anyone worse off.

prefers j to j'; if $L_i(j) = L_i(j')$, then i is indifferent between j and j'. The definition of the level function $L_j(\cdot)$ for each $j \in B$ is symmetric.

Stability. We say that an assignment $X = (x_{ij})$ is stable if there is no blocking pair (i, j), $i \in A$ and $j \in B$, $(i, j) \in E$, such that both i and j have leftover capacity; *or* i has leftover capacity and there is i', $x_{i'j} > 0$, such that j strictly prefers i to i' (or similarly for some j); *or* there are i' and j', $x_{ij'} > 0$ and $x_{i'j} > 0$, such that i strictly prefers j to j' and j strictly prefers i to i'. Note that *both* members of a blocking pair must strictly prefer to trade with each other. A stable assignment always exists, and can be found (efficiently) using a variant of Gale-Shapley algorithm [Gale and Shapley, 1962] for computing stable matchings.

Pareto efficiency. Given an assignment $X = (x_{ij})$, let $x_i(\alpha) = \sum_{j:\ L_i(j) \le \alpha} x_{ij}$ be the number of units of i's capacity that is assigned at levels no worse than α, and $x_j(\beta) = \sum_{i:\ L_j(i) \le \beta} x_{ij}$ be the number of units of j's capacity that is assigned at levels no worse than β. We say that X is Pareto efficient if there is no other feasible assignment $Y = (y_{ij})$ such that $y_i(\alpha) \ge x_i(\alpha)$ and $y_j(\beta) \ge x_j(\beta)$, for all i, j and α, β, and at least one of the inequalities is strict. That is, X is not Pareto-dominated by any other assignment where at least one agent is strictly better off and no one is worse off.

Pareto stability. A feasible assignment is called *Pareto-stable* if it is both stable and Pareto efficient.

Recall that when preference lists contain ties, a stable matching need not be Pareto efficient. The following definition is critical to our algorithm for Pareto-stable assignment.

Definition 4.2 (Augmenting Path and Cycle)**.** *Given an assignment $X = (x_{ij})$, a sequence $[i_0, j_1, i_1, \ldots, j_\ell, i_\ell, j_{\ell+1}]$ is an* augmenting path *if the following conditions hold:*

- *$x_{i_0} < c_{i_0}$ and $x_{j_{\ell+1}} < c_{j_{\ell+1}}$.*
- *$x_{i_k j_k} > 0$ for $k = 1, \ldots, \ell$.*
- *$L_{i_k}(j_k) \ge L_{i_k}(j_{k+1})$ and $L_{j_k}(i_{k-1}) \le L_{j_k}(i_k)$ for $k = 1, \ldots, \ell$.*

A sequence $[i_1, j_2, i_2, \ldots, j_\ell, i_\ell, j_1, i_1]$ is an augmenting cycle *if the following conditions hold:*

- *$x_{i_k j_k} > 0$ for $k = 1, \ldots, \ell$.*
- *$L_{i_k}(j_k) \ge L_{i_k}(j_{k+1})$ and $L_{j_k}(i_{k-1}) \le L_{j_k}(i_k)$ for $k = 1, \ldots, \ell$, where $i_0 = i_\ell$ and $j_{\ell+1} = j_1$.*
- *At least one of these inequalities is strict. If i_k is such a node, we say the augmenting cycle is associated with i_k at level $L_{i_k}(j_k)$ (and similarly for j_k.)*

Since our nodes have preferences in addition to capacities, augmenting paths and cycles must *improve not just the size of an assignment but also its quality*, as given by node preferences. The first condition in the definition of the augmenting path says that the capacities of i_0 and $j_{\ell+1}$ are not exhausted. The second condition says that there is a positive allocation from i_k to j_k in the current assignment X, and the last condition says that i_k weakly prefers j_{k+1} to j_k and j_k weakly prefers i_{k-1} to i_k. Thus, we can inject (at least) one unit of flow from i_{k-1} to j_k and from i_ℓ to $j_{\ell+1}$ and withdraw the same amount from i_k to j_k for each $k = 1, \ldots, \ell$ in the augmenting path to obtain a Pareto improvement over X. A similar Pareto improvement can be obtained for augmenting cycles.

We have the following easy lemma.

Lemma 4.3. *Any feasible assignment X that has no augmenting paths or cycles is Pareto efficient.*

4.1 Computing a Pareto Stable Assignment

We now give a strongly polynomial time algorithm to compute a Pareto stable assignment. Note that if X is a stable assignment, reassigning according to any augmenting path or cycle of X preserves stability, *i.e.*, any assignment Y that Pareto-dominates a stable assignment X is stable as well [Erdil and Ergin, 2009]. Together with Lemma 4.3, this suggests that starting with a stable assignment, and then making improvements to it using augmenting paths and cycles until no more improvements are possible, will result in a Pareto-stable assignment.

How do we find such augmenting paths and cycles? First consider the simplest case with unit capacity, *i.e.*, $c_i = c_j = 1$ for all i, j, where an assignment degenerates to a matching. Given an existing matching, define a new directed bipartite graph with the same nodes, where all forward edges are "weak improvement" edges with respect to the existing matching, and backward edges correspond to the pairings in current matching. Then we can find augmenting paths by introducing a source s and sink t that link to unmatched nodes on each side and finding s-t paths in the resulting network. Augmenting cycles can be found by a similar construction.

For our general case where $c_i, c_j \ge 1$, however, *even the concept of improvement edges for a node is not well defined*: since a node can have multiple partners in an assignment, a particular edge can be an improvement for some part of that node's capacity and not for some others. For instance, suppose that node i (with $c_i = 2$) is matched to nodes j_1 and j_3, and suppose that i strictly prefers j_1 to j_2 to j_3. Then, (i, j_2) would only represent an improvement relative to (i, j_3), but not with respect to (i, j_1), both of which exist in the current assignment. An obvious way to fix this problem is to make copies of each node, one copy for each unit of its capacity, in which case improvement edges are well-defined — each unit of capacity is associated with a unique neighbor in any assignment. However, this new graph has size $\sum_i c_i + \sum_j c_j$, leading to a runtime that is polynomial in $\sum_i c_i + \sum_j c_j$, which is *exponential* in the size of the input.

Construction of networks. In order to define improvement edges in this setting with multiunit capacities, we will create a new augmented bipartite graph G from the original bipartite market M and the preference lists P_k. The vertex set of G will consist of copies of each node in M, where *each copy represents a level on that node's preference list*. We then define forward and backward edges between the vertices: forward edges are the (weak) improvement edges, while there is one backward edge for every edge $(i, j) \in E$ corresponding to i and j's levels in P_j and P_i. This augmented graph, which is assignment-independent and depends only on the preference lists of the nodes, is then used to define a sequence

of networks with assignment-dependent capacities which we will use to find augmenting paths and cycles.

Definition 4.4. *Given the market M, construct G as follows.*

- *Vertices: For each node $i \in A \cup B$, we introduce n new vertices $T(i) = \{i(1), \ldots, i(n)\}$, where $i(\alpha)$ corresponds to the α-th level of the preference list of i. (If i has $k < n$ levels in his preference list, it suffices to introduce k vertices $i(1), \ldots, i(k)$; here, we use n levels for uniformity.)*
- *Edges: For each pair $(i, j) \in E$, let $\alpha = L_i(j)$ and $\beta = L_j(i)$. We add a* backward *edge between $i(\alpha)$ and $j(\beta)$, i.e., $j(\beta) \to i(\alpha)$. Further, we add a* forward *edge $i(\alpha') \to j(\beta')$ for every pair of vertices $i(\alpha')$ and $j(\beta')$ satisfying $\alpha' \geq \alpha$ and $\beta' \geq \beta$.*

Figure 1 gives an example of the construction of graph G, when M contains two lenders i_1, i_2 and three borrowers j_1, j_2, j_3 (node preferences are specified next to each node in the top figure, e.g., i_2 is indifferent between j_1 and j_2, and prefers both of them to j_3). The figure on the right illustrates the vertices $T(\cdot)$ of G and the backward edges; the figure on the bottom left shows the forward edges between the two groups of vertices $T(i_2)$ and $T(j_3)$ in G.

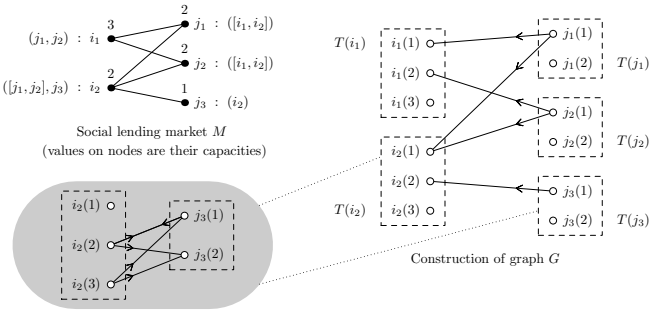

Figure 1: Construction of graph G.

Note that the construction of G is completely independent of any actual assignment X. We next define the networks $H, H_{i,\alpha}, H_{j,\beta}$, whose structure is based on G and is assignment-independent, but whose edge capacities depend on the assignment X.

Definition 4.5 (Network H, $H_{i,\alpha}$ and $H_{j,\beta}$)**.** *Given the graph G and an assignment $X = (x_{ij})$, let $G(X)$ be the network where all forward edges in G are assigned capacity ∞, and all backward edges are assigned capacity x_{ij}. We use $G(X)$ to define the networks $H(X)$, $H_{i,\alpha}(X)$ and $H_{j,\beta}(X)$ for each $i \in A$ and $j \in B$, and $\alpha, \beta = 1, \ldots, n$, as follows.*

For $H(X)$, include a source s and a sink t; further, for each $i \in A$ and $j \in B$, add an extra vertex h_i and h_j, respectively. Connect $s \to h_i$ with capacity $c_i - x_i$, and $h_j \to t$ with capacity $c_j - x_j$, where $x_i = \sum_j x_{ij}$ and $x_j = \sum_i x_{ij}$. Further, connect $h_i \to i(\alpha)$ with capacity ∞ for $\alpha = 1, \ldots, n$, and connect $j(\beta) \to h_j$ with ∞ capacity for $\beta = 1, \ldots, n$.

For $H_{i,\alpha}(X)$, we add a source s and a sink t, and connect $s \to j(\beta)$ with capacity ∞ for each vertex $j(\beta)$ satisfying $\alpha > L_i(j)$ and $\beta \geq L_j(i)$. Further, we connect $j(\beta) \to t$ with capacity x_{ij} for each $j(\beta)$ satisfying $\alpha \leq L_i(j)$ and $\beta = L_j(i)$. The network $H_{j,\beta}(X)$ is defined symmetrically.

We will use the network H to find augmenting paths with respect to an existing stable assignment X. Observe that the only edges from the source with nonzero capacity are those that connect to a node $i \in A$ with leftover capacity; similarly, the only edges to the sink with nonzero capacity are from a node $j \in B$ with leftover capacity. Sending flow from s to t in H therefore involves increasing the total size of the assignment while maintaining quality, exactly as in an augmenting path for X. Similarly, we will use the networks $H_{i,\alpha}$ and $H_{j,\beta}$ to find augmenting cycles associated with i and j at level α and β respectively. Consider any flow from s to t in $H_{i,\alpha}$, say,

$$[s, j_1(\beta_1), i_1(\alpha_1), \ldots, i_2(\alpha_2), j_2(\beta_2), t]$$

We know that $\alpha > L_i(j_1)$ (*i.e.*, i strictly prefers j_1 to all its neighbors at level α) and $L_{j_1}(i_1) = \beta_1 \geq L_{j_1}(i)$ (*i.e.*, j_1 weakly prefers i to i_1). Further, we have $\alpha \leq L_i(j_2)$ (this implies that i strictly prefers j_1 to j_2) and $L_{j_2}(i_2) \leq \beta_2 = L_{j_2}(i)$ (*i.e.*, j_2 weakly prefers i_2 to i). That is, flows from s to t in $H_{i,\alpha}$ correspond to augmenting cycles for node i at levels less than or equal to α in X (and similarly for $H_{j,\beta}$).

Our algorithm, summarized below, finds maximum flows in all the constructed networks H, $H_{i,\alpha}$ and $H_{j,\beta}$ to eliminate augmenting paths and cycles.

PARETO STABLE ASSIGNMENT (ALG-PS)

1. Let X be an arbitrary stable assignment
2. Construct networks $H(X)$, $H_{i,\alpha}(X)$ and $H_{j,\beta}(X)$, for each $i \in A$, $j \in B$, and $\alpha, \beta = 1, \ldots, n$
3. For H, $H_{i,\alpha}$ and $H_{j,\beta}$ constructed above (H to be executed first)
   (a) Compute a maximum flow $F = (f_{uv})$ from s to t (if there is no flow from vertex u to v, set $f_{uv} = 0$)
   (b) For each forward edge $i(\alpha) \to j(\beta)$, let $x_{ij} = x_{ij} + f_{i(\alpha)j(\beta)}$
   (c) For each backward edge $j(\beta) \to i(\alpha)$, let $x_{ij} = x_{ij} - f_{j(\beta)i(\alpha)}$
   (d) If the graph is $H_{i,\alpha}$
   - Let $x_{ij} = x_{ij} + f_{sj(\beta)}$ for each $s \to j(\beta)$
   - Let $x_{ij} = x_{ij} - f_{j(\beta)t}$ for each $j(\beta) \to t$

   (e) If the graph is $H_{j,\beta}$
   - Let $x_{ij} = x_{ij} - f_{sj(\beta)}$ for each $s \to i(\alpha)$
   - Let $x_{ij} = x_{ij} + f_{j(\beta)t}$ for each $i(\alpha) \to t$

   (f) Update capacities for next graph to be executed according to new assignment X
4. Output X (denoted by X^*)

Analysis. To prove that ALG-PS indeed computes a Pareto-stable assignment, we need to show two things— first, that the assignment X^* returned by the algorithm is stable; this follows easily from stability of the original assignment and that reassigning according to augmenting paths and cycles preserves stability.

Second, we need to show that X^* is Pareto efficient, *i.e.*, no further Pareto improvements are possible when the algorithm terminates. The difficulty here is that the assignment X changes through the course of the algorithm, and therefore we need to show that, for instance, no other augmenting paths can be found after the network H has been executed, even though the assignment X that was used to define the network $H(X)$ has been changed (and similarly for all augmenting cycles). That is, while we compute maximum flows in $H(X)$ to find all augmenting paths for a given assignment X, we need to show that no new augmenting paths have showed up in any updated assignments computed by the algorithm. Similarly, finding (i, α) augmenting cycles via $H_{i,\alpha}(X)$ for *some* assignment X does not automatically imply that no further (i, α) augmenting cycles will ever be found in *any* of the (different) assignments computed through the course of the algorithm, since the assignments of all nodes can change each time when a maximum flow is computed, leading to the possibility of new valid s-t paths, and therefore possibly new augmenting cycles. That this does not is due to a careful choice of the construction of the networks H, $H_{i,\alpha}$, $H_{j,\beta}$; in fact, it is possible to construct examples showing that this does not hold for other, perhaps more natural, definitions of the networks.

Our main claim is stated next. The proof uses the assignment-independence of the structure of the networks $H, H_{i,\alpha}, H_{j,\beta}$ to argue that if there is an augmenting path in any assignment produced after H is executed, we could not have found the maximum flow in $H(X)$ to begin with, a contradiction; the argument for augmenting cycles uses a similar idea. All proofs can be found in the full version of the paper[3].

Proposition 4.6. *There is no augmenting path after graph H is executed, and no augmenting cycle associated with i (resp. j) at level α (resp. β) after graph $H_{i,\alpha}$ (resp. $H_{j,\beta}$) is executed in* ALG-PS.

The above claim, together with Lemma 4.3, implies that the outcome returned by ALG-PS is indeed a Pareto-efficient assignment as required.

Running time. Each graph $H, H_{i,\alpha}$ and $H_{j,\beta}$ can be constructed in time $O(m^2n^2)$, where $n = |A|$ and $m = |B|$, and there are $O(mn)$ such graphs in all. Each graph has $O(mn)$ vertices, and is executed exactly once in time $O(m^3n^3)$, which is the running time for maximum flow using any classic network flow algorithm. Therefore, the running time of the algorithm is in $O(m^4n^4)$. We summarize this below.

Theorem 4.7. *Algorithm* ALG-PS *computes a Pareto-stable assignment in strongly polynomial time $O(m^4n^4)$.*

We note that the number of borrower categories in Zopa is a small constant, so this algorithm computes a Pareto-stable assignment in our reduced marketplace (A, C) in time $O(n^4)$ where $n = |A|$ is the number of lenders. In fact, a sharper bound on the running time of the algorithm is $O((\sum_{k\in M} |P_k|)^4)$, where $|P_k|$ is the length of k's preference list. This means that even when each category contains a single borrower as in Prosper (so m is large), the runtime remains practically feasible: since lenders usually place bids on only a small number of borrowers in typical social lending markets, $\sum_{k\in M} |P_k| = O(n)$ leading to runtime $O(n^4)$.

Computing the Market Clearing Assignment. Having computed allocations $X^* = (x^*_{iC_r})$ between lenders and borrower categories using algorithm ALG-PS, we now need to allocate the amount $x^*_{iC_r}$ amongst borrowers in C_r. Note that by feasibility of X^* for (A, C), we have $\sum_{i\in A} x^*_{iC_r} \leq \sum_{j\in C_r} c_j$. We simply divide $x^*_{iC_r}$ amongst borrowers in C_r proportional to their demands:

$$y_{i_0 j_0} = x^*_{i_0 C_r} \cdot \frac{c_{j_0}}{\sum_{j\in C_r} c_j}. \qquad (*)$$

This allocation is feasible since $\sum_{j\in C_r} y_{ij} = x^*_{iC_r}$ and $\sum_{i\in A} y_{ij} \leq c_j$. This assignment $Y = (y_{ij})$ can be proven to satisfy all the properties claimed in Theorem 3.2 for the actual marketplace $M = (A, B)$, and is our desired output.

Acknowledgements. We are very grateful to Gabrielle Demange, Bettina Klaus, Fuhito Kojima, Mohammad Mahdian, Preston McAfee, David Pennock, Michael Schwarz and anonymous referees for helpful discussions and comments.

References

[Chen *et al.*, 2009] N. Chen, A. Ghosh, and N. Lambert. Social lending. In *EC 2009*, pages 335–344, 2009.

[Echenique and Oviedo, 2006] F. Echenique and J. Oviedo. A theory of stability in many-to-many matching markets. *Theoretical Economics*, 1:233–273, 2006.

[Erdil and Ergin, 2008] A. Erdil and H. Ergin. What's the matter with tie-breaking? Improving efficiency in school choice. *American Economic Review*, 98:669–689, 2008.

[Erdil and Ergin, 2009] A. Erdil and H. Ergin. Two-sided matching with indifferences. Working paper. 2009.

[Freedman and Jin, 2009] S. Freedman and G. Jin. Learning by doing with asymmetric information: Evidence from Prosper.com. Working paper. 2009.

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

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

[Hatfield and Kominers, 2010] J. Hatfield and S. Kominers. Matching in networks with bilateral contracts. In *EC 2010*, pages 119–120, 2010.

[Iwama and Miyazaki, 2008] K. Iwama and S. Miyazaki. *Stable Marriage with Ties and Incomplete Lists*. Encyclopedia of Algorithms, 2008.

[Iwama *et al.*, 1999] K. Iwama, D. Manlove, S. Miyazaki, and Y. Morita. Stable marriage with incomplete lists and ties. In *ICALP 1999*, pages 443–452, 1999.

[Roth and Sotomayor, 1992] A. E. Roth and M. Sotomayor. *Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis*. Cambridge University Press, 1992.

[3] http://www.ntu.edu.sg/home/ningc/paper/ijcai11-z.pdf

[Roth, 2008] A. E. Roth. Deferred acceptance algorithms: History, theory, practice, and open questions. *International Journal of Game Theory*, pages 537–569, 2008.

[Sotomayor, 2009] M. Sotomayor. Pareto-stability is a natural solution concept in matching markets with indifferences. Working paper. 2009.

[Yanagisawa, 2007] H. Yanagisawa. *Approximation Algorithms for Stable Marriage Problems*. Ph.D. Thesis, Kyoto University, 2007.

A Other Solution Concepts

Here we briefly discuss other candidate solution concepts, and justify our choice of assignment. Given that one of the challenges of the social lending marketplace is to attract lenders, one possibility is to seek a lender-optimal stable assignment, which is best for every lender amongst the set of all stable assignments. However, it is well known that when preferences contain ties, a lender-optimal stable matching is not well defined, and need not exist. A natural weaker solution concept is to seek a stable lender-Pareto efficient assignment— an assignment that is stable assignment and Pareto efficient for lenders, *i.e.*, there is no other assignment which constitutes a Pareto improvement for lenders. However, as Example 2.31 in [Roth and Sotomayor, 1992] shows, such a lender-Pareto optimal solution again need not exist. Erdil and Ergin [Erdil and Ergin, 2008] consider an even weaker solution concept in the context of school choice: They propose choosing a stable assignment such that there is no other *stable* assignment which is a Pareto improvement for lenders. This solution concept, however, while meaningful in the original context of school choice, unfairly penalizes borrowers at the expense of lenders, as the following example shows.

Example A.1. *There are two lenders i_1, i_2 and two borrowers j_1, j_2 with unit supply/demand each, where j_1 is a low-risk borrower and j_2 is high-risk. Assume that i_1 offers an interest rate of 8% to j_1 and 17% to j_2, and prefers j_2 to j_1 (i.e., the high-risk high-return investment); and i_2 offers 5% to j_1 and 10% to j_2, and is indifferent between these two. Then the matching $\{(i_1, j_1; 8\%), (i_2, j_2; 10\%)\}$ is stable, but is Pareto-dominated, from the perspective of the lenders, by another stable matching $\{(i_1, j_2; 17\%), (i_2, j_1; 5\%)\}$. But note that the second matching gives borrower j_2 a loan with a very high interest-rate (17%), even though there are lenders who are willing to lend to j_2 at a much better interest-rate.*

Another family of solution concepts is based on total size or weight. First off, we note that aggregate solution concepts such as maximum flow or minimum weight flow or maximal flow completely ignore the lender-specified preferences, and indeed, it is not hard to see construct simple examples where the assignments produced by such aggregate solution concepts are disliked by both lenders and borrowers. Next, a maximum size stable assignment does not necessarily guarantee Pareto efficiency (see the example below). Further, it is in general NP-hard to compute [Yanagisawa, 2007] even for the unit node-capacity setting. Therefore, it is not suitable for applications like social lending with massive data input.

Example A.2. (MAX-SIZE STABLE MATCHING NEED NOT BE PARETO EFFICIENT.) *There are two lenders i_1, i_2 and two borrowers j_1, j_2 with unit supply/demand each. Lender i_1 prefers j_2 to j_1 and all other agents are indifferent between the two alternatives. The stable matching $(i_1, j_1), (i_2, j_2)$ has the maximum possible size of 2, but is not Pareto efficient as it is Pareto-dominated by the matching $(i_1, j_2), (i_2, j_1)$, where no one is worse off and i_1 strictly improves her assignment.*

B Missing Proofs: Algorithm ALG-PS

By the flow decomposition theorem, any feasible flow in a network can be decomposed into the sum of flows on paths from s to t, and flows around cycles. Therefore, the reassignments defined by Step 3 in the algorithm can be decomposed according to these individual flows along paths and cycles as well. In the following discussion, for each maximum flow in $H, H_{i,\alpha}$ and $H_{j,\beta}$, we will consider each such component of the decomposed flow separately for clarity of exposition.

Further, consider any s-t flow in $H_{i,\alpha}$, $F = [s, j_1(\beta_1), \ldots, j_2(\beta_2), t]$. The reassignment in Step 3(d) injects flow from i to j_1 and withdraws the same amount of flow from i to j_2. Hence, F is equivalent to a pseudo-flow $F' = [i(\alpha), j_1(\beta_1), \ldots, j_2(\beta_2), i(\alpha)]$, which is around a cycle. The same argument applies to s-t flows in $H_{j,\beta}$. Therefore, to simplify our discussion, we assume (without loss of generality) that all flows in $H_{i,\alpha}$ and $H_{j,\beta}$ are around cycles.

B.1 Feasibility and Stability

We first argue that X^* returned by the algorithm is a feasible assignment, and is a Pareto improvement over X, *i.e.*, every node's assignment during the algorithm will only be enhanced (i.e. never get worse off). We consider flows along paths and cycles respectively.

- The flow is $[i_1(\alpha_1), j_2(\beta_2), \ldots, j_\ell(\beta_\ell), i_\ell(\alpha_\ell), j_1(\beta_1), i_1(\alpha_1)]$, which is around a cycle in either $H, H_{i,\alpha}$ or $H_{j,\beta}$.
  - Feasibility. By the flow conservation constraints and the capacities defined on backward edges, the total number of units for each vertex will not change and still remains feasible.
  - Enhancement. By the construction of the graphs, we know that for each forward edge $i_k(\alpha_k) \to j_{k+1}(\beta_{k+1})$, $k = 1, \ldots, \ell$ (where $j_{\ell+1}(\beta_{\ell+1}) = j_1(\beta_1)$), $L_{i_k}(j_{k+1}) \le \alpha_k$ and $L_{j_{k+1}}(i_k) \le \beta_{k+1}$. Further, since each $j_k(\beta_k) \to i_k(\alpha_k)$ is a backward edge, for $k = 1, \ldots, \ell$, we have $L_{i_k}(j_k) = \alpha_k$ and $L_{j_k}(i_k) = \beta_k$. Hence, after reassigning the assignment (withdraw units from backward edges and inject the same amount on forward edges), each vertex gets an assignment at least as good as its previous assignment.
- The flow $F = [s, h_i, i(\alpha), \ldots, j(\beta), h_j, t]$ is from s to t in H.
  - Feasibility. Since the capacity on edge $s \to h_i$ is $c_i - x_i$ and the capacity on edge $h_j \to t$ is $c_j - x_j$, the new number of units (*i.e.*, flow) injected into i and j is upper bounded by $c_i - x_i$ and $c_j - x_j$, respectively. Hence, the total assignment of i and

j is still feasible. For any vertex between $i(\alpha)$ and $j(\beta)$ in F, similar to the discussion above, the total number of units assigned will not change and it still gives a feasible assignment.

– Enhancement. By the same argument as above, the assignment of all vertices between $i(\alpha)$ and $j(\beta)$ will not get worse off. For the two vertices connecting to h_i and h_j, we will only inject more assignment on i and j, respectively. Hence, no vertex is worse off after reassignment.

Lemma B.1. *The solution X^* returned by* ALG-PS *is a stable assignment.*

Proof. Let $X = (x_{ij})$ be the stable assignment which is used to initialize the algorithm. By the arguments above, we know that $x_i^* \geq x_i$ and $x_j^* \geq x_j$. Assume otherwise that $X^* = (x_{ij}^*)$ is not a stable assignment. Then there is a blocking pair (i, j) for X^*, $(i, j) \in E$. By the definition of stable assignment, we only need to consider the following three cases:

Case 1. $c_i > x_i^*$ and $c_j > x_j^*$, i.e. both i and j have supply and demand left in X^*. Then $c_i > x_i$ and $c_j > x_j$, which implies that (i, j) is a blocking pair for X as well.

Case 2. $c_i = x_i^*$ and $c_j > x_j^*$. Since (i, j) is a blocking pair, there must exist j_1 such that $x_{ij_1}^* > 0$ and i prefers j to j_1. By the enhancement argument above, the levels of assignment of i only improve in to X^* relative to X. Hence, either $c_i > x_i$ or there is a j_2 such that $x_{ij_2} > 0$ and $L_i(j) < L_i(j_1) \leq L_i(j_2)$, i.e. i prefers j to j_2. This implies that (i, j) is a blocking pair for X as well. The case where $c_i > x_i^*$ and $c_j = x_j^*$ is symmetric.

Case 3. $c_i = x_i^*$ and $c_j = x_j^*$. Similar to the above argument, for agent i, either $c_i > x_i$ or there is j' such that $x_{ij'} > 0$ and i prefers j to j'; and for j, either $c_j > x_j$ or there is a i' such that $x_{i'j} > 0$ and j prefers i to i'. These two facts imply that (i, j) is a blocking pair for X.

Therefore, for each case, we get a contradiction. Hence, X^* is a stable assignment. □

B.2 Pareto efficiency

In this subsection, we will prove that algorithm ALG-PS outputs a Pareto efficient assignment.

Lemma B.2. *Any assignment X that has no augmenting path and augmenting cycle is Pareto efficient.*

Proof. For each i and j, we construct a new market by making c_i and c_j identical copies (with unit capacity each and the same preference list), respectively. Note that the new market might contain exponential number of agents (but here we are only concerned with its economic properties rather than algorithmic computations). It can be seen that any Pareto efficient assignment of the original market corresponds to a Pareto efficient matching of the new market. By a claim proved by Sotomayor [Sotomayor, 2009], which says that a matching is Pareto efficient if and only if it contains no augmenting path and cycle, we know that the non-existence of augmenting path and cycle gives a sufficient and necessary condition for Pareto efficiency for our setting as well. □

Proposition B.3. *There is no augmenting path after graph H is executed in step 3 of* ALG-PS.

Proof. Denote by X_0 the assignment used to define H, and let X be a assignment computed during the algorithm at some step after H being executed. Assume otherwise that there is still an augmenting path P in X,

$$P = [i_0, j_1, i_1, \ldots, j_\ell, i_\ell, j_{\ell+1}]$$

Let $\alpha_0 = L_{i_0}(j_1)$, $\alpha_k = L_{i_k}(j_k)$ and $\beta_k = L_{j_k}(i_k)$ for $k = 1, \ldots, \ell$, and $\beta_{\ell+1} = L_{j_{\ell+1}}(i_\ell)$. By definition of augmenting paths, we have $L_{j_k}(i_{k-1}) \leq L_{j_k}(i_k) = \beta_k$ and $\alpha_k = L_{i_k}(j_k) \geq L_{i_k}(j_{k+1})$, for $k = 1, \ldots, \ell$.

Consider the s-t path corresponding to P:

$$F = [s, i_0(\alpha_0), j_1(\beta_1), i_1(\alpha_1), \ldots, j_\ell(\beta_\ell), i_\ell(\alpha_\ell), j_{\ell+1}(\beta_{\ell+1}), t]$$

Since the structure of the networks $H, H_{i,\alpha}, H_{j,\beta}$ are independent of the assignment, all forward edges $i_k(\alpha_k) \to j_{k+1}(\beta_{k+1})$, as well as all backward edges $j_k(\beta_k) \to i_k(\alpha_k)$ are in H, as well as $H_{i,\alpha}$ and $H_{j,\beta}$.

First we will show that there is no augmenting path in the assignment X_1 computed right after $H(X_0)$ is executed by the algorithm (note that we computed the maximum flow in $H(X_0)$, not $H(X_1)$, so this is not obvious). The argument that shows that there is no augmenting path in any assignment X returned after execution of any of the $H_{i,\alpha}$ or $H_{j,\beta}$ in the course of the algorithm is similar and is given next.

- First suppose that X is the new assignment immediately after $H(X_0)$ is executed. If all backward edges in F, $j_k(\beta_k) \to i_k(\alpha_k)$, have non-zero capacities in $H(X_0)$, then the algorithm should be able to inject more flow according to F while computing the maximum flow in $H(X_0)$. Hence, there must be a pair, say i_k and j_k, that has no capacity on $j_k(\beta_k) \to i_k(\alpha_k)$ in $H(X_0)$.

  First suppose for simplicity that there is exactly one such pair i_k, j_k (the argument when there are multiple such pairs is similar, as explained later). Since this backward edge does have capacity in $H(X_1)$, the algorithm must have sent a positive flow, say from $i_k(\alpha')$ to $j_k(\beta')$ while computing the maximum flow in $H(X_0)$. (By the reassignments of assignment in step 3(b) of the algorithm, this will lead to a positive capacity on edge $j_k(\beta_k) \to i_k(\alpha_k)$ after the execution of $H(X_0)$.) Note that $\alpha' \geq \alpha_k$ and $\beta' \geq \beta_k$. Consider the flow F' in $H(X_0)$ that injects flows through $i_k(\alpha') \to j_k(\beta')$.

  – If F' is an s-t flow, then instead of using the edge $i_k(\alpha') \to j_k(\beta')$, F' may follow $i_k(\alpha') \to j_{k+1}(\beta_{k+1})$ and the rest of edges in F; and instead of using the edge $i_{k-1}(\alpha_{k-1}) \to j_k(\beta_k)$, F may follow $i_{k-1}(\alpha_{k-1}) \to j_k(\beta')$ and the rest of edges in F'. Since all other edges in F except the backward edge (j_k, i_k) are feasible in $H(X_0)$, and F' is a feasible flow in $H(X_0)$, the new flow obtained by combining F and F' as described is feasible in $H(X_0)$, and increases the flow from s to t in $H(X_0)$, contradicting the fact that the algorithm computed a maximum flow in $H(X_0)$.

– If F' is a flow around a cycle,

$$F' = [\, i_k(\alpha'), j_k(\beta'), i'_1(\alpha'_1), j'_1(\beta'_1), \ldots, i'_r(\alpha'_r), j'_r(\beta'_r), i_k(\alpha')\,]$$

i.e., through F', we inject assignment from $i_k(\alpha')$ to $j_k(\beta')$, thus increasing the capacity on the backward edge $j_k(\beta_k) \rightarrow i_k(\alpha_k)$. Note that the increment on the capacity of $j_k(\beta_k) \rightarrow i_k(\alpha_k)$ by F' is upper bounded by the capacities of all edges in F'. Observe that an equivalent way to travel through $[i_{k-1}(\beta_{k-1}), j_k(\beta_k), i_k(\alpha_k), j_{k+1}(\beta_{k+1})]$ in F is to take a detour

$$i_{k-1}(\beta_{k-1}), j_k(\beta'), i'_1(\alpha'_1), j'_1(\beta'_1), \cdots, i'_r(\alpha'_r), j'_r(\beta'_r), i_k(\alpha'), j_{k+1}(\beta_{k+1})$$

Note that the first and last edge, $i_{k-1}(\beta_{k-1}) \rightarrow j_k(\beta')$ and $i_k(\alpha') \rightarrow j_{k+1}(\beta_{k+1})$, are guaranteed to exist. Further, the capacity on these two edges is infinite, thus the maximum possible amount of flows on such detour is always at least the capacity of edge $j_k(\beta_k) \rightarrow i_k(\alpha_k)$ increased by F'. So again, we can combine F and F' to obtain a flow which is feasible in $H(X_0)$, and increases the total flow from s to t, contradicting the fact that the algorithm computed a maximum flow in $H(X_0)$.

If there are multiple backward edges in F with zero capacity in $H(X_0)$, the argument proceeds similarly; add to F the flows $F'_{i_k j_k}$ corresponding to each of these backward edges (*i.e.*, the ones that lead to increased assignment on (i_k, j_k)) to obtain a feasible flow in $H(X_0)$ that increases the total flow from s to t giving a contradiction.

Hence, there is no augmenting path after the execution of H.

- Next, we show that there is no augmenting path in any assignment X returned by the algorithm after executing any of the graphs $H_{i,\alpha}, H_{j,\beta}$. The proof proceeds by induction—the base case is to show that there is no augmenting path in the assignment obtained after executing the first $H_{i,\alpha}$ network after H, given that there is no augmenting path after executing H. The induction step shows that if there is no augmenting path after the first $H_{i,\alpha}$ is executed, there can be no augmenting path after any later $H_{i,\alpha}$ or $H_{j,\beta}$ network is executed as well.

  Assume that the first graph executed after H is $H_{i,\alpha}$, and let X and X' be the assignments before and after executing $H_{i,\alpha}$ respectively. We have already shown that is no augmenting path in X after the execution of H. So there must be a backward edge, say $j_k(\beta_k) \rightarrow i_k(\alpha_k)$, which has zero capacity in the assignment X but positive capacity in X'; the capacity of this edge must have been increased while computing the maximum flow in $H_{i,\alpha}(X)$ through either an s-t flow or a flow around a cycle. Similar to the previous arguments, we can combine this flow with the augmenting path in X' to obtain an augmenting path for X. This implies that there is an augmenting path after the execution of H, a contradiction. (The analysis is similar for graph $H_{j,\beta}$.) The induction step uses an almost identical argument to show that if there is an augmenting path after $H_{i,\alpha}$ (or $H_{j,\beta}$), an augmenting path can be shown to exist in the assignment used as input to $H_{i,\alpha}$ as well, violating the induction hypothesis. The claim therefore follows by induction on the execution of all graphs $H_{i,\alpha}$ and $H_{j,\beta}$, for any $i \in A$, $j \in B$, and $\alpha, \beta = 1, \ldots, n$.

□

Next we discuss removing augmenting cycles in Algorithm ALG-PS, and show that once network $H_{i,\alpha}$ is executed by the algorithm, there are no augmenting cycles for node i at level α in any assignment computed by the algorithm thenceforth. Note that the algorithm does not specify any particular order (in particular, on α for a given node i) in which the $H_{i,\alpha}$'s are to be executed. Also, observe that while the structure of the network $H_{i,\alpha}$ allows augmenting cycles of levels *less than or equal to* α to be found (since every j with $L_i(j) \geq \alpha$ is connected to the sink t), it is only guaranteed that augmenting cycles of level *exactly equal* to α will never be found again through the remainder of the algorithm.

A natural question is why it is necessary to link every j, $L_i(j) > \alpha$, to the sink in $H_{i,\alpha}$ when executing $H_{i,\alpha}$ only guarantees removing all augmenting cycles of level *exactly* equal to α. The example below shows what happens if the network $H_{i,\alpha}$ is defined to only include links to the sink for nodes with $L_i(j) = \alpha$, *i.e.*, to only try to find augmenting cycles of level α: In this case, it is possible that there are still augmenting cycles of level α in the assignment computed immediately after $H_{i,\alpha}$ being executed. In our construction when the network is defined to include the remaining links of level strictly less than α as well, this does not happen and $H_{i,\alpha}$ only needs to be executed once.

Example B.4. *Suppose that $A = \{i_1, i_2, i_3, k\}$ and $B = \{j_1, j_2, j_3, j_4, j_5\}$. All nodes except k have unit capacity and are indifferent between all possible partners (i.e., have only one level in their preference list). Node k has capacity 2, and preference list $([j_1, j_5], [j_3, j_4], j_2)$. Suppose we start with the (stable) assignment X_0 where k is matched to j_2, j_3, and the remaining assignments are $(i_1, j_1), (i_2, j_4), (i_3, j_5)$ (note there are no augmenting paths in X_0). Consider finding the maximum flow in network $H_{i,\alpha}$* without *the link $j_2 \rightarrow t$ for $\alpha = 2$. In this network, the total capacity of edges incident to the sink is 1, thus we can send at most one unit flow, for example $k \rightarrow j_1 \rightarrow i_1 \rightarrow j_2 \rightarrow k \rightarrow j_4 \rightarrow i_2 \rightarrow j_3 \rightarrow t$ (note that the two k's here correspond to different vertices in $T(k)$ in the network). After reassigning assignment according to this flow, we obtain the new assignment X' $(i_1, j_2), (i_2, j_3), (i_3, j_5), (k, j_1), (k, j_4)$. But observe that X' still has an augmenting cycle at level 2 for node k: $k \rightarrow j_5 \rightarrow i_3 \rightarrow j_4 \rightarrow k$. However, with the original definition of $H_{i,\alpha}$, which links $j_2 \rightarrow t$, the maximum flow consists of pushing flow along the paths $k \rightarrow j_1 \rightarrow i_1 \rightarrow j_2 \rightarrow t$ and $k \rightarrow j_5 \rightarrow i_3 \rightarrow j_4 \rightarrow i_2 \rightarrow j_3 \rightarrow t$, leading to the new assignment $X'' = (i_1, j_2), (i_2, j_3), (i_3, j_4), (k, j_1), (k, j_5)$ which has no remaining augmenting cycles for k.*

Proposition B.5. *There is no augmenting cycle associated with i (resp. j) at level α (resp. β) after graph $H_{i,\alpha}$ (resp. $H_{j,\beta}$) is executed in step 3 of* ALG-PS.

Proof. We will only prove the claim for $i \in A$ and $H_{i,\alpha}$; the argument for $j \in B$ and $H_{j,\beta}$ is symmetric. Assume otherwise that there is still an augmenting cycle

$$P = [i = i_1, j_2, i_2, \ldots, j_\ell, i_\ell, j_1, i_1]$$

associated with i at level α in some assignment X' computed after executing $H_{i,\alpha}(X)$. Let $\alpha_k = L_{i_k}(j_k)$ and $\beta_k = L_{j_k}(i_k)$ for $k = 1, \ldots, \ell$. Since P is an augmenting cycle associated with i_1 at level α, we have $\alpha = \alpha_1 > L_{i_1}(j_2)$. Consider the cycle corresponding to P in the augmented graph G:

$$F = [i_1(\alpha_1), j_2(\beta_2), i_2(\alpha_2), \ldots, j_\ell(\beta_\ell), i_\ell(\alpha_\ell), j_1(\beta_1), i_1(\alpha_1)]$$

(Since the structure of $H_{i,\alpha}, H_{j,\beta}$ are assignment independent, all forward edges $i_k(\alpha_k) \to j_{k+1}(\beta_{k+1})$ and all backward edges $j_k(\beta_k) \to i_k(\alpha_k)$ are in $H_{i,\alpha}$ and $H_{j,\beta}$.) Note that in $H_{i,\alpha}$, we have $s \to j_2(\beta_2)$ and $j_1(\beta_1) \to t$. Denote $i_{\ell+1}(\alpha_{\ell+1}) = i_1(\alpha_1)$ and $j_{\ell+1}(\beta_{\ell+1}) = j_1(\beta_1)$.

The proof is similar to the previous proposition; the only difference is that we will separately need to consider the case $k = 1$, *i.e.*, the backward edge (j_1, i_1) is the one whose capacity increases.

- First consider the assignment X' returned immediately after executing $H_{i,\alpha}$. If all backward edges $j_k(\beta_k) \to i_k(\alpha_k)$ have non-zero capacities in X, then F is a feasible flow in $H_{i,\alpha}(X)$ which increases the total flow from s to t. Hence, there must be a pair, say i_k and j_k, which has no capacity on $j_k(\beta_k) \to i_k(\alpha_k)$ in $H_{i,\alpha}(X)$, but has positive capacity in X'. Hence, the algorithm must have sent a positive flow, say from $i_k(\alpha')$ to $j_k(\beta')$, while executing $H_{i,\alpha}(X)$. Note that $\alpha' \geq \alpha_k$ and $\beta' \geq \beta_k$. Again, suppose first that there is just one such pair, and consider the flow F' that injects flows through $i_k(\alpha') \to j_k(\beta')$.
  - Suppose F' is an s-t flow, i.e.

    $$F' = [s, \ldots, j''(\beta''), i_k(\alpha'), j_k(\beta'), \ldots, t]$$

    If $k \neq 1$, then instead of using the edge $i_k(\alpha') \to j_k(\beta')$, F' may follow $i_k(\alpha') \to j_{k+1}(\beta_{k+1})$ and the rest of edges in F; and instead of using the edge $i_{k-1}(\alpha_{k-1}) \to j_k(\beta_k)$, F may follow $i_{k-1}(\alpha_{k-1}) \to j_k(\beta')$ and the rest of edges in F'. These flows are both feasible in $H_{i,\alpha}(X)$ and increase the total $s - t$ flow, giving a contradiction.

    If $k = 1$, i.e. $i_k = i_1 = i$, then we have $\alpha = \alpha_1 = \alpha_k \leq \alpha' = L_i(j'')$ and $\beta'' = L_{j''}(i)$. Hence, we have $j''(\beta'') \to t$. Then we can truncate F' at $j''(\beta'')$ and route it to the sink directly through edge $j''(\beta'') \to t$; and re-route F at $i_{k-1}(\alpha_{k-1})$ by following $i_{k-1}(\alpha_{k-1}) \to j_k(\beta')$ and the rest of edges in F'. These new flows are both now feasible in $H_{i,\alpha}(X)$ and increase total flow, a contradiction.
  - If F' is a flow around a cycle (i.e. F' does not effectively contribute to the maximum flow from s to t),

    $$F' = [\, i_k(\alpha'), j_k(\beta'), i'_1(\alpha'_1), j'_1(\beta'_1), \ldots, i'_r(\alpha'_r), j'_r(\beta'_r), i_k(\alpha')\,]$$

    i.e., F' injects assignment from $i_k(\alpha')$ to $j_k(\beta')$, increasing capacity on the backward edge $j_k(\beta_k) \to i_k(\alpha_k)$ in X'. Further, the increment on the capacity of $j_k(\beta_k) \to i_k(\alpha_k)$ by F' is upper bounded by the capacities of all edges in F'.

    If $k \neq 1$, an equivalent way to travel through $[i_{k-1}(\beta_{k-1}), j_k(\beta_k), i_k(\alpha_k), j_{k+1}(\beta_{k+1})]$ in F is to take a detour

    $$i_{k-1}(\beta_{k-1}), j_k(\beta'), i'_1(\alpha'_1), j'_1(\beta'_1), \cdots, i'_r(\alpha'_r), j'_r(\beta'_r), i_k(\alpha'), j_{k+1}(\beta_{k+1})$$

    Note that the first and last edge, $i_{k-1}(\beta_{k-1}) \to j_k(\beta')$ and $i_k(\alpha') \to j_{k+1}(\beta_{k+1})$, are guaranteed to exist. Further, the capacity on these two edges is infinite, thus the maximum possible amount of flows on such a detour is always at least the capacity of edge $j_k(\beta_k) \to i_k(\alpha_k)$ increased by F'. So again, it is possible to construct a feasible flow in $H_{i,\alpha}(X)$ that increases the total flow.

    If $k = 1$, i.e. $i_k = i_1 = i$, then we have $\alpha = \alpha_1 = \alpha_k \leq \alpha' = L_i(j'_r)$ and $\beta'_r = L_{j'_r}(i)$. Hence, we have $j'_r(\beta'_r) \to t$. Then the following flow combining F and F'

    $$i_1(\alpha_1), j_2(\beta_2), i_2(\alpha_2), \ldots, j_\ell(\beta_\ell), i_\ell(\alpha_\ell) \to j_k(\beta'), i'_1(\alpha'_1), j'_1(\beta'_1), \ldots, i'_r(\alpha'_r), j'_r(\beta'_r)$$

    is feasible in $H_{i,\alpha}(X)$, again leading to a contradiction.

  Hence, there is no augmenting cycle associated with i at level α after the execution of graph $H_{i,\alpha}$. The argument for multiple pairs (i_k, j_k) is similar to that sketched for augmenting paths, and proceeds by adding the corresponding flows F'.
- Assume that the graph executed right after $H_{i,\alpha}$ in step 3 is $H_{i'}(\alpha')$. Since there is no augmenting cycle associated with i at level α in the assignment X returned after executing $H_{i,\alpha}$, there must be a backward edge, say $j_k(\beta_k) \to i_k(\alpha_k)$, which has zero capacity after the execution of $H_{i,\alpha}$; the capacity of this edge must be increased by $H_{i'}(\alpha')$ through either an s-t flow or a flow around a cycle. As above, we can construct an augmenting cycle using F that is feasible in X, contradicting that there was no augmenting cycle after $H_{i,\alpha}$.

Hence, there is no augmenting cycle associated with i at level α after $H_{i'}(\alpha')$ is executed as well. The analysis is similar for graph $H_{j,\beta}$. The claim then follows by induction on the execution of all graphs $H_{i,\alpha}$ and $H_{j,\beta}$, for any $i \in A$, $j \in B$, and $\alpha, \beta = 1, \ldots, n$. □

Finally we provide the proof of the main result, Theorem 3.2.

Theorem B.6. *The final assignment $Y = (y_{ij})$ can be computed in time $O(|A|^4 + |A||B|)$ and has the following properties:*

1. *Stability: There are no blocking pairs in the original marketplace $M = (A, B)$.*
2. *Pareto efficiency: No agent in M can be made better off without making some other agent in M strictly worse off.*
3. *(Weak) envy-freeness: No borrower envies the allocation of any other borrower in its category.*
4. *Diversity: Given the allocations $X^* = (x^*_{iC_r})$, each lender i spreads her budget amongst the maximum number of distinct borrowers.*

Proof. Stability of Y in (A, B) follows simply from the fact that $X^* = (x^*_{iC_r})$ is a stable assignment for (A, C) by Theorem 4.7, and the observation that lenders are indifferent between all the borrowers in a category.

For Pareto efficiency, it suffices to show that Y does not contain any augmenting path or cycle in (A, B) by Lemma 4.3. First, since X^* is Pareto efficient in (A, C), i and C_r cannot both have leftover capacity and $x^*_{iC_r}$ is distributed completely amongst borrowers in C_r for each i. Hence, there can be no augmenting path where all borrowers belong only to C_r. Similarly, there is no augmenting cycle with all borrowers only from C_r because if $j_1 \in C_r$ gets a strictly better allocation, there must be $j_2 \in C_r$ who is strictly worse off because all borrowers in C_r have the same preferences over lenders. Therefore, there is no augmenting path or cycle where all borrowers belong to the same category C_r. Now if Y contains an augmenting path $[i_0, j_1, i_1, \ldots, j_\ell, i_\ell, j_{\ell+1}]$ in market (A, B) where j_i belonged to at least two different categories, there is an augmenting path $[i_0, C_{f(j_1)}, i_1, \ldots, C_{f(j_\ell)}, i_\ell, C_{f(j_{\ell+1})}]$ in the virtual market (A, C) as well, a contradiction. The same argument applies for augmenting cycles, so Y is Pareto efficient in (A, B).

(Weak) envy-freeness follows immediately from the proportional distribution of $x^*_{iC_r}$: let lender i offer interest-rate p_i to category C_r, then the net interest-rate of any $j \in C_r$ is

$$\frac{\sum_{i \in A} y_{ij} p_i}{\sum_{i \in A} y_{ij}} = \frac{\sum_{i \in A} x^*_{iC_r} p_i}{\sum_{i \in A} x^*_{iC_r}}$$

which is independent of his capacity c_j. So all borrowers in C_r get the same net interest-rate. The final claim about diversity simply follows from formula $(*)$. □

C Computing a Stable Assignment

Finally, for completeness, we include the algorithm to find a stable assignment, which is a simple modification of the Gale-Shapley algorithm for stable matching; the fairly straightforward proof is omitted.

Theorem C.1. *Algorithm* STABLE-ASSIGNMENT *computes a stable assignment.*

STABLE-ASSIGNMENT

1. For each $i \in A$, set its remaining capacity $c_r(i) = c_i$; and for each $j \in B$, set $c_r(j) = c_j$
2. For each $i \in A$ and $j \in B$, set $x_{ij} = 0$
3. Break ties in all preference lists arbitrarily
4. While there is $i \in A$ with $c_r(i) > 0$ and $P_i \neq \emptyset$
   (a) let j be the neighbor at the top of the preference list P_i
   (b) set i and j to be *engaged* and $P_i \leftarrow P_i \setminus \{j\}$
   (c) let $i_1, \cdots, i_r, i, i_{r+1}, \cdots, i_s$ be the agents in A engaged to j where $L_j(i_1) < \cdots < L_j(i_r) < L_j(i) < L_j(i_{r+1}) < \cdots < L_j(i_s)$
   (d) let $x_{ij} = \min\left\{c_r(i), c_j - \sum_{k=1}^{r} x_{i_k j}\right\}$
   (e) let $c_r(i) \leftarrow c_r(i) - x_{ij}$ and $c_r(j) \leftarrow c_r(j) - x_{ij}$
   (f) if $c_r(j) \leq 0$
   - if $x_{ij} + \sum_{k=1}^{r} x_{i_k j} \geq c_j$, denote i_ℓ to be i
   - else
     - let $i_\ell = \arg\min_\ell \left\{x_{ij} + \sum_{k=1}^{\ell} x_{i_k j} \geq c_j\right\}$
     - let $c_r(i_\ell) \leftarrow c_r(i_\ell) + x_{i_\ell j} - \left(c_j - (x_{ij} + \sum_{k=1}^{\ell-1} x_{i_k j})\right)$
     - let $x_{i_\ell j} = c_j - \left(x_{ij} + \sum_{k=1}^{\ell-1} x_{i_k j}\right)$
   - for each strict successor i' of i_ℓ on j's list P_j
     - let $c_r(i') \leftarrow c_r(i') + x_{i'j}$
     - break engagement between i' and j
     - let $P_{i'} \leftarrow P_{i'} \setminus \{j\}$ and $P_j \leftarrow P_j \setminus \{i'\}$
5. Output assignment $X = (x_{ij})$