Practical algorithms and experimentally validated incentives for equilibrium-based fair division (A-CEEI)

ERIC BUDISH*, University of Chicago , USA
RUIQUAN GAO, Stanford University, USA
ABRAHAM OTHMAN*, University of Pennsylvania, USA
AVIAD RUBINSTEIN†, Stanford University, USA
QIANFAN ZHANG, Princeton University, USA

Approximate Competitive Equilibrium from Equal Incomes (A-CEEI) is an equilibrium-based solution concept for fair division of discrete items to agents with combinatorial demands. In theory, it is known that in asymptotically large markets:

- For incentives, the A-CEEI mechanism is Envy-Free-but-for-Tie-Breaking (EF-TB), which implies that it is Strategyproof-in-the-Large (SP-L).
- From a computational perspective, computing the equilibrium solution is unfortunately a computationally intractable problem (in the worst-case, assuming PPAD ≠ FP).

We develop a new heuristic algorithm that outperforms the previous state-of-the-art by multiple orders of magnitude. This new, faster algorithm lets us perform experiments on real-world inputs for the first time. We discover that with real-world preferences, even in a realistic implementation that satisfies the EF-TB and SP-L properties, agents may have surprisingly simple and plausible deviations from truthful reporting of preferences. To this end, we propose a novel strengthening of EF-TB, which dramatically reduces the potential for strategic deviations from truthful reporting in our experiments.

A (variant of) our algorithm is now in production: on real course allocation problems it is much faster, has *zero* clearing error, and has stronger incentive properties than the prior state-of-the-art implementation.

CCS Concepts: • **Theory of computation → Algorithmic game theory and mechanism design**.

Additional Key Words and Phrases: A-CEEI, incentive compatibility, equilibrium computation in practice

ACM Reference Format:
Eric Budish, Ruiquan Gao, Abraham Othman, Aviad Rubinstein, and Qianfan Zhang. 2023. Practical algorithms and experimentally validated incentives for equilibrium-based fair division (A-CEEI). In *Proceedings of the 24th ACM Conference on Economics and Computation (EC '23), July 9–12, 2023, London, United Kingdom.* ACM, New York, NY, USA, 32 pages. https://doi.org/10.1145/3580507.3597809

1 INTRODUCTION

Competitive Equilibrium from Equal Incomes (CEEI) [Foley 1967; Thomson and Varian 1985; Varian 1974] is an attempt to leverage the economic efficiency of market equilibria while preserving the *ex*

*Disclosure: Cognomos Inc. implements the technology described in this paper as part of its commercial course allocation product, Schedule Scout. Budish and Othman are founding directors of Cognomos Inc.
†Supported by NSF CCF-2112824, and a David and Lucile Packard Fellowship.

EC '23, July 9–12, 2023, London, United Kingdom

ACM ISBN 979-8-4007-0104-7/23/07…$15.00
https://doi.org/10.1145/3580507.3597809

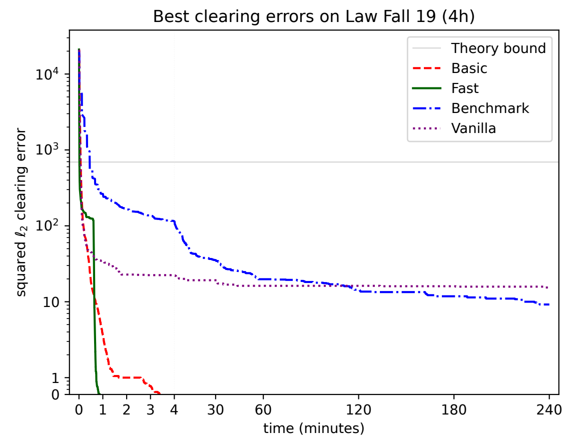

Fig. 1. For each of the algorithms tested and time t (up to 4-hours), we plot the best clearing error obtained by the algorithm up to time t. Note that our final algorithm (Fast) finds a zero-clearing-error solution after less than 1 minutes, while previous commercial state-of-the-art (Benchmark) does not converge after 4 hours. See Section 4 for details.

post fairness properties that come from equal incomes. For complex preferences, however, CEEI does not necessarily exist.

Budish [2011] developed a relaxation of CEEI, *Approximate CEEI (A-CEEI)*. He showed that an approximate equilibrium from approximately equal incomes always exists. Budish [2011] also showed if the perturbations to agents' incomes (henceforth *budgets*) are chosen at random, the mechanism satisfies an Envy-Free-but-for-Tie-Breaking (EF-TB) property, which by Azevedo and Budish [2019] implies that it is Strategyproof in the Large (SP-L); i.e. as the number of agents tends to infinity, the mechanism becomes approximately strategyproof.

Budish's original theoretical work invoked a fixed-point theorem to prove existence. That makes it inherently non-constructive. It was shown by Othman et al. [2016] that finding an A-CEEI is PPAD-complete, even if we allow constant budget inequality [Rubinstein 2018]. A-CEEI is therefore similar to many other economic equilibria whose existence rely on non-algorithmic proofs, the most famous of which is the Nash equilibrium [Chen et al. 2009; Daskalakis et al. 2009].

Despite this theoretical infeasibility, a heuristic algorithm exists that solves the problem adequately in practice [Budish et al. 2017]. The existing heuristic algorithm makes A-CEEI a practical solution concept for settings that require efficiency and fairness but for which the use of real money is impractical or repugnant [Roth 2007]. The setting where A-CEEI has seen the most practical application is in the allocation of courses to students. Student preferences are often quite complex in course allocation. Students typically demand many courses and individual courses could be complements or substitutes depending on the bundle of other courses in a student's schedule. Course allocation—particularly in professional schools—also tends to be a challenging allocation problem, as the most popular "star courses" tend to have far more demand than supply.

In this paper we explore the course allocation problem experimentally using real data from properly motivated student preferences over schedules of professional school courses, using data from the commercial implementation of A-CEEI fielded by Cognomos.

1.1 Contribution I: A (much) faster heuristic algorithm for computing A-CEEI

Our first contribution is an improved heuristic algorithm for computing A-CEEI. Our algorithm outperforms the commercial state-of-the-art by several orders of magnitude in both the quality of produced solution as well as runtime on real course allocation problems, see e.g. Figure 1.

We highlight some of our algorithmic findings here, with details in Sections 3 and 4. Further details about the previous state-of-the-art algorithm (henceforth *benchmark*), can also be found in academic publications [Budish et al. 2017; Othman et al. 2010]. At a high level, both our algorithm and the benchmark, perform variants of the classic tatonnement algorithm: increase the price of over-demanded courses, and decrease the prices of under-demanded courses.

One of the main technical innovations introduced in the benchmark algorithm was the use of individual price adjustments: on some iterations, the algorithm can make a larger improvement on the clearing error by adjusting the price of a single course rather than all the courses simultaneously. Based on extensive experiments on randomly generated data (real data from students bidding was not yet available at the time), Othman et al. [2010] reported that the algorithm was much more likely to find a solution with acceptable clearing error when mixing individual price adjustments and full tatonnement updates. Our first algorithmic insight is to largely *reverse this finding of Othman et al. [2010]*. We observe that on real world instances, it is in fact much more efficient to use only full tatonnement updates, even when they locally increase the clearing error. We discuss evidence, explanations, and caveats of this finding in Section 4.

Our second (and perhaps more interesting) algorithmic insight focuses on the "end game", when the algorithm is already close to a reasonable clearing error. Here, we show that making tiny-but-cleverly-optimized perturbations to the *budgets* (rather than prices) can quickly lead to the holy grail of zero clearing error. Before going into the details, we remark that (i) in the practical benchmark implementation, students budgets are already perturbed at random (and with larger perturbations); (ii) our insight is inspired directly by the non-algorithmic existence proof in Budish [2011], which perturbs the students' budgets twice: once to guarantee a desirable fixed point of the tatonnement correspondence, and a second time to break ties between marginal students at the fixed point.

To describe the budget perturbations, let's start from the end: we would like to return a vector of course prices and almost-equal student budgets and (ideally) zero clearing error. Towards this, at each iteration of the tatonnement the algorithm solves an (NP-hard but fast in practice) integer program to look for a small budgets perturbation that will nudge students' demand bundles to zero the overall market clearing error. More generally, at each iteration we can find the *optimal budgets perturbation*, aka the one that minimizes the clearing error. Since our ultimate goal is to find a price vector that works well with an optimal budget perturbation, we use the clearing error with respect to the optimal budget perturbation to perform the next iteration of tatonnement.

While the idea of using optimal budget perturbations is extremely effective algorithmically, we have to be careful not to open opportunities for manipulability: although budget increases are very small, doing it based on student's reported course preferences could open an opportunity for strategic reporting. To this end, we encode in our integer program the same EF-TB constraints that guarantee the SP-L (Strategyproof in the Large) condition for the original A-CEEI mechanism. Although this makes the integer program larger, we can solve it very efficiently (see Section 3.1.1 for some optimizations).

1.2 Contribution II: Empirically evaluating the incentives of A-CEEI

In theory, the A-CEEI mechanism has a very desirable property: it is *Strategyproof in the Large (SP-L)*; i.e. the expected utility any student can gain by misreporting their true preferences diminishes as the number of students goes to infinity [Azevedo and Budish 2019][1]. What does this theory guarantee for realistic schools with a finite number of students? Unfortunately, the answer is not much: the formal SP-L convergence guarantee ([Azevedo and Budish 2019, Theorem 1]) requires the number of students to be much larger than the number of possible types. For the general A-CEEI mechanism[2], a type consists of a ranking of all possible schedules; the number of possible schedules is exponential in the number of courses, and the number of ways they can be ranked adds another layer of exponentiation. In other words, until the number of students is *doubly-exponential* in the number of courses, the formal convergence results for SP-L mechanisms are meaningless.

The situation is further complicated by the computational intractability of the A-CEEI problem. A natural approach for dealing with the large number of possible deviations is to directly analyze the way students could manipulate the algorithm's choice of equilibrium. This has been fruitful in analyzing simple algorithms [Akbarpour et al. 2020; Vitercik 2021]. But because of the PPAD-hardness of the A-CEEI problem we resort to a highly nontrivial heuristic algorithm (see Contribution I). Theoretically analyzing how a possible misreport of preferences would affect the trajectory in price space taken by this algorithm seems far from tractable.

Going beyond intractable theoretical guarantees on students' incentives, our approach is to empirically evaluate them. Our main question in this part is:

> In practice, can students gain in the A-CEEI mechanism by manipulating their reported preferences?

Here, it is paramount that we have our novel fast heuristic algorithm: Previously, it was barely possible to compute one allocation, so re-computing allocations for each possible deviation was completely out of the question. Anecdotally, this is the reason why the computational exploration of the same issue in Budish's original paper was limited to tiny examples with 2 students and 4 courses [Budish 2010, Footnote 31].

Still, no matter how fast our equilibrium computation algorithm, we cannot possibly enumerate all possible deviations (again, in theory their number is doubly exponential in the number of courses). However, it is reasonable to expect that computationally- and informationally-bounded students also cannot test every possible deviation. We thus model a strategic student using a simple hill-climbing algorithm that adjusts the single course weights starting from the original truthful report. We also consider different restrictions on the student's information about the market, as detailed in Definition 7.

In Section 6 we use our manipulation-finding algorithm in combination with our fast A-CEEI finding algorithm to explore the plausibility of effective manipulations for students bidding in A-CEEI. Originally, we had expected that since our mechanism satisfies the EF-TB and SP-L properties, it would at least be practically strategyproof — if even we don't really understand the way our algorithm chooses among the many possible equilibria, how can a student with limited information learn to strategically bid in such a complex environment?

Indeed, in 2 out of 3 schools that we tested, our manipulation-finding algorithms finds very few or no statistically significant manipulations at all. However, when analyzing the 3rd school, we stumbled upon a simple and effective manipulation for (the first iteration of) our mechanism. We emphasize that although the manipulation is simple in hindsight, in over a year of working on this project we failed to predict it by analyzing the algorithm — *the manipulation was discovered by the algorithm.*

[1] The informal intuition is that students who are "price-takers", i.e. they regard the prices of courses as set "by the market", receive the optimal schedule they can afford and thus have no incentive to misreport their preferences.

[2] In practice, students' reporting language is often more restricted, but still the number of students is always much smaller than the number of possible types.

Inspired by this manipulation, we propose a natural strengthening of envy-free (discussed below), which we call *contested-envy free*. We encode the analogous *contested EF-TB* as a new constraint in our algorithm (specifically, the integer program for finding optimal budget perturbations). Fortunately, our algorithm is still very fast even with this more elaborate constraint. And, when we re-run our manipulation-finding experiments, we observe that contested EF-TB significantly reduces the potential for manipulations in practice.

1.3 Contribution III: Contested Envy Free (but for Tie Breaking)

In this section we gradually build towards our new notions of contested-envy free, and contested EF-TB. We begin with the basic notion of *Envy-Free (EF)*: an allocation is said to be envy free if no student i prefers the schedule allocated to another student i'.

In the course allocation problem, due to the challenges of integrality constraint and combinatorial demand, EF allocations rarely exist, but A-CEEI allocations are guaranteed to satisfy important relaxations of EF such as EF-TB (discussed below). In contrast, contested-EF is a *strengthening* of EF. To motivate the distinction between EF and contested-EF, consider the following anecdote. (This anecdote is for illustration purposes only; in Section 5 we discuss examples of courses and students derived from manipulations found by our algorithm on instances with real preferences.)

Example 1 (Contested envy free).
Eric drives a Honda and Mohammad drives a Porsche. Eric would rather have Mohammad's Porsche than his Honda. However, Eric doesn't envy *Mohammad, because his kids are in his Honda, and he wouldn't trade the bundle of* {*the Honda and his kids*} *for Mohammad's Porsche (with no kids).*

Eric loves his own kids, but nobody else would want them in their car; we thus say that they're *uncontested* for understanding the envy between Eric and Mohammad. Formally, in the the specific context of A-CEEI[3] we say that contested are the goods with strictly positive price, and goods with zero price are uncontested. Considering uncontested goods for the purposes of determining envy is an obvious source of incentive issues: Eric can always report a low value for his kids to claim to envy Mohammad's allocation. Because they're uncontested, Eric is still guaranteed to have them in any Pareto optimal allocation.

If we restrict our attention to *contested* goods, Eric would indeed rather trade his Honda for Mohammad's Porsche. More generally, we say that Eric *contested-envies* Mohammad if Eric prefers any subset of {Mohammad's allocation} ∪ {uncontested goods} over his own allocation. Notice that contested EF is a strengthening of EF.

As mentioned before, EF allocations rarely exist in the course allocation problem. Azevedo and Budish [2019] relax the notion of EF to allow *tie-breaking* (EF-TB): The students are ranked at random[4], and the allocation is said to satisfy EF-TB if no student i envies any lower-ranked students. (However, i may envy higher-ranked students.) EF-TB is not a very satisfying fairness criterion[5], but it does imply strategyproof-in-the-large (SP-L) [Azevedo and Budish 2019].

A-CEEI allocations satisfy EF-TB when the budgets are assigned at random because no student can envy another student with a lower budget (if i''s budget is lower, then i''s schedule must also be affordable for i). When we introduce optimal budget perturbations, we simply encode EF-TB as a constraint in our perturbation-finding integer program: if i''s initial tie breaker is lower than i's, then the EF-TB constraint is that i cannot envy i' in the perturbed economy. While in theory this

[3] More generally when prices aren't available it is natural to extend the notion of "uncontested" to capture under-demanded goods.
[4] In practice a combination of seniority and random ranking may be used.
[5] For fairness, Budish [2011] introduced the notion of Envy-Free-up-to-1-good (EF1) and proved that it is satisfied by A-CEEI.

approach satisfies SP-L, in practice it can open the door to manipulations; see Sections 5 and 6 for discussion and evidence from experiments.

We can generalize EF-TB to *contested EF-TB* in the natural way: no student can contested-envy a lower ranked student. We henceforth refer to the EF-TB criterion as *classic EF-TB* to distinguish it from contested EF-TB.

Note that contested EF-TB is a strengthening of classic EF-TB, so it also implies SP-L. Observe also that if all the budgets respect the random TB rule, then any A-CEEI allocation is contested EF-TB. In our new algorithm, because of the optimal budget perturbations, the budgets may not respect the original TB order; instead, we encode the contested EF-TB constraint in the integer program that optimizes over budget perturbations.

In Section 6 we bring a quantitative analysis of our manipulation-finding algorithm: we give evidence from experiments on real students bids, suggesting that when we enforce contested EF-TB, it is fairly hard to find successful manipulations.

1.4 Limitations of our approach

Modeling students' valuations. Throughout the paper we take students' reported preferences as their true valuations. In practice, students do not perfectly report their full preferences [Budish and Kessler 2022]. In fact, students usually have a limited interface, e.g. they may rate courses as "favorite, great, good, fair", and there is a hand-tuned formula converting those ratings to utilities. Concurrent work by Soumalias et al. [2022] focuses on the orthogonal direction of better eliciting and modeling of students' preferences using neural networks. Equilibrium computation is a major bottleneck of their approach, and we leave it to future work to see if our respective algorithms can be combined effectively.

Variance across markets. We are very fortunate to be able to test our algorithms on real data from a few different programs shared with us by Cognomos. There seem to be large variance between instances from different programs. In particular, in one school we find significant manipulations for the no-EF-TB and classic-EF-TB constraints, whereas in others those variants were also hard to manipulate. Running times also vary greatly between instances. However, two trends are consistent between all the instances we tested: (i) our algorithm is much faster than the previous state of the art, and (ii) contested EF-TB seems to have desirable incentive properties. We plan to test on more datasets as they become available.

Limitations of the manipulation finding algorithm. Our algorithm for discovering profitable manipulations is a highly imperfect surrogate to the real question of *can real students find robust manipulations?* On one hand, students can come up with more complicated manipulations than the ones considered by our algorithm; on the other hand, the algorithm has more information than any single student would normally have. A particular issue is that we run our experiment for exploring profitable deviations with very small sample sizes: even with a very fast algorithm and restriction to very simple manipulations, we have to restrict our experiment to a as few as five samples, which is quite noisy. (We later validate every candidate manipulation with at least 100 iterations.) To make sure that our manipulation-finding algorithm still makes sense, we benchmark it on the HBS mechanism which is known to be manipulable [Budish and Cantillon 2012]; indeed we find significant manipulations are possible with HBS on all instances.

1.5 Conclusion

In this work, we give a significantly faster algorithm for computing A-CEEI. Kamal Jain's famous formulation "if your laptop cannot find it then neither can the market" [Papadimitriou 2007] was originally intended as a negative result, casting doubt on the practical implications of many famous economic concepts because of their worst-case computational complexity results. Even for course allocation, where a heuristic algorithm existed and worked in practice, Jain's formulation seemed to still bind, as solving A-CEEI involved an intense day-long process with a fleet of high-powered cloud servers operating in parallel. The work detailed in this paper has significantly progressed what laptops can find: even the largest and most challenging real course allocation problems we have access to can now be solved in under an hour on a commodity laptop.

This significant practical improvement suggests that the relationship between prices and demand for the course allocation problem—and potentially other problems of economic interest with complex agent preferences and heterogeneous goods—may be much simpler than has been previously believed and may be far more tractable in practice than the worst-case theoretical bounds. Recalling Jain's dictum, perhaps many more market equilibria can be found by laptops—or, perhaps, Walras's original and seemingly naive description of how prices iterate in the real world may in fact typically produce approximate equilibria.

Our fast algorithm also opens the door for empirical research on A-CEEI, because we can now solve many instances and see how the solution changes for different inputs. We took it in one direction: empirically investigating the incentives properties of A-CEEI for the first time. For course allocation specifically, this faster algorithm opens up new avenues for improving student outcomes through experimentation. For instance, university administrators often want to subsidize some group of students (e.g., second-year MBA students over first-year MBA students), but are unsure how large of a budget subsidy to grant those students to balance equity against their expectations. Being able to run more assignments with different subsidies can help to resolve this issue.

Remark 1 (Zero vs small clearing error)**.** We highlight that our algorithm is not only fast - it also finds allocations with *zero* clearing error. Even the non-algorithmic existence proof of Budish [2011] only guarantees a *small* clearing error.

While the previous heuristic algorithm was able to find adequate allocations in practice, it introduces some additional potential manipulability. That algorithm was a three-stage process. The first stage finds an approximate equilibrium, which equilibrium will tend to have both undersubscription in positive-price courses as well as oversubscription. This is the approximate equilibrium guaranteed to exist by Budish [2011]. The second stage progressively increases course prices to eliminate all oversubscription. This tends to increase total clearing error but makes the solution implementable, since all of the error comes from underallocating seats in valuable courses. The final stage is a "backfill" process, where students are sequentially allocated extra budget and allowed to spend that budget on courses that are undersubscribed.

While the backfill process substantially reduces the deadweight loss of computed assignments it may have problematic incentive and fairness properties. Students who get first shot at spending that extra budget in the backfill may be able to add an excellent course to their schedule. In particular, the backfill process is not known to satisfy properties like (contested) EF-TB or SP-L. Observe, however, that the backfill process is only necessary because of the clearing error found in the original approximate equilibrium. In contrast, so far our new algorithm has found A-CEEI with *zero* clearing error on every instance it has encountered. This completely obviates the need for the second and third stages of Budish et al. [2017]: with no clearing error, there are no seats that need to be backfilled.

Remark 2 (Social welfare)**.** Although our main focus is on improving the algorithmic efficiency and incentives guarantees of the A-CEEI mechanisms, in Appendix C we compare the social welfare of our algorithm and the previous approach. Although the results aren't as decisive as on other metrics, we observe that our algorithm tends to give better allocations in terms of (utilitarian and Nash) social welfare.

Discussion of additional related work can be found in Appendix A.

2 PRELIMINARIES

Definition 1 (The course allocation market)**.** A *course allocation market* $\left(\boldsymbol{u} = (u_i)_{i=1}^n, \boldsymbol{c} = (c_j)_{j=1}^m\right)$ consists of:

- m courses, where each course $j \in [m]$ has an integral amount of capacity c_j;
- n students, where each student $i \in [n]$ has a utility function u_i over each course bundle.

Definition 2 (Allocation, excess demand, and market-clearing error)**.** Fix a market $(\boldsymbol{u}, \boldsymbol{c})$, course prices $\boldsymbol{p} = (p_j)_{j=1}^m$, and student budgets $\boldsymbol{b} = (b_i)_{i=1}^n$, the allocation function $\boldsymbol{a} = (\boldsymbol{a}_i)_{i=1}^n$ is defined as

$$\boldsymbol{a}_i(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{b}) = \operatorname*{arg\,max}_{\boldsymbol{x} \in 2^{[m]}, \boldsymbol{p} \cdot \boldsymbol{x} \le b_i} u_i(\boldsymbol{x}).$$

We further define the *excess demand* function $\boldsymbol{z} = (z_j)_{j=1}^m$ as

$$z_j(\boldsymbol{u}, \boldsymbol{c}, \boldsymbol{p}, \boldsymbol{b}) = \sum_{i=1}^n a_{ij}(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{b}) - c_j,$$

and the *clipped excess demand* function $\tilde{\boldsymbol{z}} = (\tilde{z}_j)_{j=1}^m$ as

$$\tilde{z}_j(\boldsymbol{u}, \boldsymbol{c}, \boldsymbol{p}, \boldsymbol{b}) = \begin{cases} z_j(\boldsymbol{u}, \boldsymbol{c}, \boldsymbol{p}, \boldsymbol{b}) & \text{if } p_j > 0, \\ \max\{0, z_j(\boldsymbol{u}, \boldsymbol{c}, \boldsymbol{p}, \boldsymbol{b})\} & \text{if } p_j = 0. \end{cases}$$

And we define the *market-clearing error* as $\|\tilde{\boldsymbol{z}}(\boldsymbol{u}, \boldsymbol{c}, \boldsymbol{p}, \boldsymbol{b})\|_2$.

Definition 3 (Approximate competitive equilibrium from equal incomes (A-CEEI))**.** For constant $\alpha, \beta > 0$ and market $(\boldsymbol{u}, \boldsymbol{c})$, we say a pair of prices and budgets $(\boldsymbol{p}, \boldsymbol{b})$ forms an (α, β)*-CEEI* if there is

- small market-clearing error: $\|\tilde{\boldsymbol{z}}(\boldsymbol{u}, \boldsymbol{c}, \boldsymbol{p}, \boldsymbol{b})\|_2 \le \alpha$;
- small budget perturbation: $b_i \in [1, 1 + \beta] \quad \forall i$.

2.1 (Contested) Envy-Free-but-for-Tie-Breaking

We formally define the notions of EF-TB and contested EF-TB. For both, it is important to make the distinction between the initial budgets $\boldsymbol{b_0}$ that are determined exogenously (e.g. at random), and the final budgets $\boldsymbol{b}$ which may also depend on the reported preferences. In both cases, students' initial budgets play a second role in determining the direction in which envy is allowed; in particular the initial budgets are assumed to be distinct (hence "tie-breaking").

Definition 4 (Envy-Free-but-for-Tie-Breaking (EF-TB))**.** Given an initial budget $\boldsymbol{b_0}$, for a market $(\boldsymbol{u}, \boldsymbol{c})$, price $\boldsymbol{p}$, and budget $\boldsymbol{b}$, the allocation $\boldsymbol{a}(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{b})$ is called *EF-TB with respect to budget* $\boldsymbol{b_0}$, if for all student $i, j \in [n]$ such that $b_{0,i} > b_{0,j}$, we have $u_i(a_i(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{b})) > u_i(S)$ for all bundle $S \subseteq a_j(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{b})$.

Furthermore, we say that an A-CEEI algorithm $A(\boldsymbol{u}, \boldsymbol{c}, \boldsymbol{b_0})$ is EF-TB, if for any market $(\boldsymbol{u}, \boldsymbol{c})$ and any initial budget $\boldsymbol{b}_0$, the final allocation $\boldsymbol{A}(\boldsymbol{u}, \boldsymbol{c}, \boldsymbol{b_0})$ is always EF-TB with respect to $\boldsymbol{b_0}$.

Definition 5 (Contested Envy-Free-but-for-Tie-Breaking (Contested EF-TB))**.** Given an initial budget $\boldsymbol{b_0}$, for a market $(\boldsymbol{u}, \boldsymbol{c})$, price $\boldsymbol{p}$, and budget $\boldsymbol{b}$, the allocation $\boldsymbol{a}(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{b})$ is called *contested EF-TB with respect to budget* $\boldsymbol{b_0}$ *and price* $\boldsymbol{p}$, if for all student $i, j \in [n]$ such that $b_{0,i} > b_{0,j}$, we have $u_i(a_i(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{b})) > u_i(S)$ for all bundle S such that $S \subseteq a_j(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{b}) \cup \{k \in [m] : p_k = 0\}$.

Furthermore, we say that an A-CEEI algorithm $A(\boldsymbol{u}, \boldsymbol{c}, \boldsymbol{b_0})$ is contested EF-TB, if for any market $(\boldsymbol{u}, \boldsymbol{c})$ and any initial budget $\boldsymbol{b}_0$, the final allocation $\boldsymbol{A}(\boldsymbol{u}, \boldsymbol{c}, \boldsymbol{b_0})$ is always contested EF-TB with respect to $\boldsymbol{b_0}$ and $\boldsymbol{p}$.

2.2 Utility functions

While the A-CEEI existence works for general ordinal preferences, we focus on the following restricted class of utility functions. This class is consistent with most utilities reported in practice, which are typically taken to be additively-separable utilities, (i) with a preference for schedules satisfying a minimum-number-of-course-units requirements, and (ii) subject to satisfying simple constraints, e.g. timing and curriculum conflicts. Using the language of Operations Research, schedule validity is a hard constraint, while having a schedule meet a student's requirements is a soft constraint. (The problem remains PPAD-complete in the worst-case when restricted to this class; see Appendix D.)

Formally, every utility function u can be described by a tuple ($\boldsymbol{w} \in \mathbb{R}^m, valid, req : 2^{[m]} \to \mathbb{N}$), such that for every possible bundle $\boldsymbol{x} \in 2^{[m]}$,

$$u(\boldsymbol{x}) = \begin{cases} \boldsymbol{w} \cdot \boldsymbol{x} + B & valid(\boldsymbol{x}) = 1, req(\boldsymbol{x}) = 1 \\ \boldsymbol{w} \cdot \boldsymbol{x} & valid(\boldsymbol{x}) = 1, req(\boldsymbol{x}) = 0 \\ -\infty & valid(\boldsymbol{x}) = 0 \end{cases} \tag{1}$$

where B is some large number such that $B > \|\boldsymbol{w}\|_1$. The function $valid$ and req also follow some structures so that they can be efficiently represented (e.g., in a mixed-integer program).

3 OUR ALGORITHM

In this section we describe our fast heuristic algorithm for computing A-CEEI. We begin with a description of the basic algorithm (Subsection 3.1). We then move to describe some further optimizations that we found helpful when solving larger instances (Subsection 3.2).

3.1 Our Basic Algorithm

All our algorithms take as inputs the students' reported preferences $\boldsymbol{u}$, the course capacities $\boldsymbol{c}$, and initial budgets $\boldsymbol{b_0}$ that are determined at random, sometimes with a bonus for seniority. Our basic algorithm proceeds as follows: At each iteration, the algorithm looks for the optimal budget perturbation given current prices; then it computes the market clearing error for this optimal budget perturbation; finally, it updates the prices according to tatonnement rule. The algorithm terminates when it reaches zero clearing error[6]. The pseudocode is given in Algorithm 1.

Computing the ϵ-budget perturbation. To compute the ϵ-budget perturbation in the second step, we should optimize among all possible budget perturbations so that the resulting clearing error is minimized. Observe that for any fixed price vector, the demand of any student can only change on some budgets that are the sum of some prices; and the sum of prices is always a multiple of the fixed step size δ.

Therefore, we can always partition the interval $[b_{0i} \pm \epsilon]$ of Student i's possible budgets into $k_i \le \frac{2\epsilon}{\delta} + 1$ sub-intervals $(\underline{b_{i\ell}}, \overline{b_{i\ell}}]$, such that i's demand bundle $\boldsymbol{a_{i\ell}}$ is constant on each sub-interval:

$$\left\{(\boldsymbol{a_{i\ell}}, \underline{b_{i\ell}}, \overline{b_{i\ell}}) : \text{student } i\text{'s demand is } \boldsymbol{a_{i\ell}} \text{ for every budget in } [\underline{b_{i\ell}}, \overline{b_{i\ell}}]\right\}_{\ell=1}^{k_i}, \tag{2}$$

where $\underline{b_{i\ell}}, \overline{b_{i\ell}}$s are multiples of δ in $b_{0i} \pm \epsilon$.

[6]It is also a good idea to enforce a time limit as a solution with zero clearing error is not even guaranteed to exist, but in practice we managed to find solutions with zero clearing error on all the instances we encountered.

ALGORITHM 1: Find an A-CEEI with (contested) EF-TB property

Inputs: students' utility functions $\boldsymbol{u}$, course capacities $\boldsymbol{c}$, initial budgets $\boldsymbol{b_0}$
Outputs: final prices $\boldsymbol{p}^*$ and budgets $\boldsymbol{b}^*$
Parameters: step size δ, maximum budget perturbation ϵ, type t of the EF-TB constraint used (0 for no EF-TB constraint, 1 for EF-TB constraint, and 2 for contested EF-TB)
Algorithm:

(1) Let $\boldsymbol{p} \leftarrow \boldsymbol{0}$.
(2) ϵ-**budget perturbation**: find budgets $\boldsymbol{b}$ such that the market-clearing error $\|\tilde{z}(\boldsymbol{u}, \boldsymbol{c}, \boldsymbol{p}, \boldsymbol{b})\|_1$ is minimized under the following constraints:
  (a) The maximum perturbation $\|\boldsymbol{b} - \boldsymbol{b}_0\|_\infty \leq \epsilon$;
  (b) Allocation $\boldsymbol{a}(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{b})$ is EF-TB with respect to $\boldsymbol{b}_0$ if $t = 1$, or contested EF-TB with respect to $\boldsymbol{b}_0$ and $\boldsymbol{p}$ if $t = 2$.

  Furthermore, we shall use $\|\boldsymbol{b}\|_1$ as the tie-breaker to guarantee the uniqueness of the solution, i.e., always picking $\boldsymbol{b}$ with minimum $\|\boldsymbol{b}\|_1$ among all optimal solutions.
(3) If $\|\tilde{z}(\boldsymbol{u}, \boldsymbol{c}, \boldsymbol{p}, \boldsymbol{b})\|_2 = 0$, terminate with $\boldsymbol{p}^* = \boldsymbol{p}, \boldsymbol{b}^* = \boldsymbol{b}$.
(4) Otherwise, update $\boldsymbol{p} \leftarrow \boldsymbol{p} + \delta\tilde{z}(\boldsymbol{u}, \boldsymbol{c}, \boldsymbol{p}, \boldsymbol{b})$, then go back to step 2.

Once we compute these arrays, we can solve for the optimal ϵ-budget perturbation using the following integer linear program:

(Budget-Perturb-ILP)

$$
\begin{array}{llrr}
\min & \|z\|_1 & & \text{(Minimize clearing error)} \\
\text{s.t.} & \displaystyle\sum_{i\in[n]}\sum_{\ell\in[k_i]} x_{i\ell}\cdot a_{i\ell j} = c_j + z_j & \forall j \in [m], p_j > 0 & \text{(Clearing error: } p_j > 0) \\
& \displaystyle\sum_{i\in[n]}\sum_{\ell\in[k_i]} x_{i\ell}\cdot a_{i\ell j} \leq c_j + z_j & \forall j \in [m], p_j = 0 & \text{(Clearing error: } p_j = 0) \\
& \displaystyle\sum_{\ell\in[k_i]} x_{i\ell} = 1 & \forall i \in [n] & \text{(1 schedule per student)} \\
& x_{i\ell} \in \{0,1\} & \forall i \in [n], \ell \in [k_i] & \text{(Integral allocations)}
\end{array}
$$

We add the following constraint to ensure (contested) EF-TB: For any student $i, i' \in [n]$ such that i's priority is higher than i' (i.e. $b_{0i} > b_{0i'}$), and any $\ell \in [k_i], \ell' \in [k_{i'}]$, if the (contested) EF-TB is violated when student i is allocated $\boldsymbol{a_{i\ell}}$ and student i' is allocated $\boldsymbol{a_{i'\ell'}}$, i.e. according to Definition 4 and 5, then we prevent simultaneously allocating $\boldsymbol{a_{i\ell}}$ to student i and allocating $\boldsymbol{a_{i'\ell'}}$ to student i'.

$$
x_{i\ell} + x_{i'\ell'} \leq 1 \text{ if } \begin{cases} \exists S \subseteq \boldsymbol{a_{i'\ell'}}, u_i(\boldsymbol{a_{i\ell}}) < u_i(S) & \text{(for classic EF-TB)} \\ \exists S \subseteq \boldsymbol{a_{i'\ell'}} \cup \{k : p_k = 0\}, u_i(\boldsymbol{a_{i\ell}}) < u_i(S) & \text{(for contested EF-TB)} \end{cases} . \tag{3}
$$

Remark 3. Solving integer programs is NP-hard in general, but we can solve (Budget-Perturb-ILP) quite fast in practice with modern SAT solvers.

Fact 1. *Fix parameters $\delta, \epsilon > 0$ and $t \in \{0,1,2\}$. For a market $(\boldsymbol{u}, \boldsymbol{c})$ and initial budgets $\boldsymbol{b_0}$, if Algorithm 1 terminates on input $\boldsymbol{u}, \boldsymbol{c}, \boldsymbol{b_0}$, its output $(\boldsymbol{p}^*, \boldsymbol{b}^*)$ forms a $\left(0, \frac{\max_{i\in[n]}(b_0)_i+\epsilon}{\min_{i\in[n]}(b_0)_i-\epsilon}\right)$-CEEI and the final allocation $\boldsymbol{a}(\boldsymbol{u}, \boldsymbol{p}^*, \boldsymbol{b}^*)$ is EF-TB with respect to $\boldsymbol{b_0}$ if $t = 1$, or contested EF-TB with respect to $\boldsymbol{b_0}$ and $\boldsymbol{p}$ if $t = 2$.*

3.1.1 Speeding up the search of budget perturbations satisfying EF-TB constraints. The simple implementation of searching EF-TB constraints is that we enumerate all pairs of students i, i' with their possible demands $\boldsymbol{a}_{i\ell}, \boldsymbol{a}_{i'\ell'}$ and check (3) for all the EF-TB constraints. However, on larger instances this process is very slow because we need to consider $\binom{n}{2}$ pairs of students, each student may receive one of k_i bundles, and finally for each pair of possible bundles checking (3) requires to solve student i's optimal bundle out of a subset of courses $\tilde{\boldsymbol{a}}_{i'\ell'}$ ($\boldsymbol{a}_{i'\ell'}$ for classic EF-TB and $\boldsymbol{a}_{i'\ell'} \cup \{k : p_k = 0\}$ for contested EF-TB). For an economy with more than 3000 students, it requires to solve for more than 4.5 million such optimal bundles. Repeating that on each iteration is quite slow! For this issue, we consider two optimizations:

Two simple sufficient conditions for no-envy: Fixing i, i', ℓ, ℓ', we shall use two simple sufficient conditions for proving that allocating $\boldsymbol{a}_{i\ell}$ and $\boldsymbol{a}_{i'\ell'}$ to i, i' does not violate the (contested) EF-TB constraints. Because these two conditions are easy to verify, we can reduce the number of times we check (3) a lot and thus save a lot of time. The first condition comes from the reporting language: even if the entire super-bundle $\tilde{\boldsymbol{a}}_{i'\ell'}$ is invalid for student i, its utility from i's perspective is no greater than that for $\boldsymbol{a}_{i\ell}$, i.e.,

$$B \cdot req_i(\tilde{\boldsymbol{a}}_{i'\ell'}) + \boldsymbol{w_i} \cdot \tilde{\boldsymbol{a}}_{i'\ell'} \leq B \cdot req_i(\boldsymbol{a}_{i\ell}) + \boldsymbol{w_i} \cdot \boldsymbol{a}_{i\ell} \, .$$

The second condition is from the fact that $\boldsymbol{a}_{i\ell}$ is the i's optimal allocation under budget $\overline{b_{i\ell}}$. If the total price of bundle $\tilde{\boldsymbol{a}}_{i'\ell'}$ is upper bounded by $\overline{b_{i\ell}}$, $\forall S \subseteq \tilde{\boldsymbol{a}}_{i'\ell'}, u_i(S) \leq u_i(\boldsymbol{a}_{i\ell})$.

Memorize envious pairs of students: Students with very different preferences are likely to never envy each other throughout the run of the algorithm. We take advantage of this idea by only enumerating all pairs of students every 10 iterations, and in the other 9 iterations we only consider pairs of students whose envy constraint was tight in a past iteration. In particular, when the optimal budget perturbation results in a zero-error solution (under a partial enumeration of possible envies), we force the algorithm to recompute the iteration by enumerating all pairs of students. Note that this implementation cannot guarantee that the allocation computed in each iteration satisfies (contested) EF-TB. However, it guarantee the final allocation satisfies (contested) EF-TB.

On an instance with approximately 3000 students, these two optimizations speed up our time-per-iteration (amortized including iterations where we check all pairs) by a factor of about 300.

3.2 Shortcuts in price space

Warm starts. In the preliminary experiments, we observe the following phenomena when using different step sizes and proper budget perturbation.

- When the step size δ is small compared to ϵ, the algorithm can converge to a zero-error solution. However, for courses that are consistently slightly over-demanded, their prices increase slowly from 0 to their final prices. With these courses, the algorithm needs almost $1/\delta$ or even more steps to converge.
- On the other hand, when the step size δ is large, the number of possible budget-demand pairs for each student may not be enough to help the budget perturbation significantly improve the clearing error. However, even if we set ϵ to 0 (i.e. we only use discrete tatonnement), the prices found by the algorithm can be quite close to good regions, where prices found with smaller δ lie, and the algorithm only takes much less time to reach such regions because of the larger step size.

Motivated by these observations, we shall combine discrete tatonnement and our algorithm together to improve the speed. We shall first run discrete tatonnement with a larger step size δ_0 and with $(1 + \beta)/\delta_0$ steps, and then turn to our algorithm with a smaller step. In the first warm-start phase we also save time by not computing optimal budget perturbations.

Merge Equivalent Steps. As discussed above, when we use smaller step sizes, it may take the algorithm many iterations to update some prices. Fortunately, it turns out that for many of those iterations the set of possible demands remains constant across all students for many consecutive iterations. Whenever this is the case, we can save time by binary searching for the next iteration where the excess demand changes. A further more clever optimization considers stretches where the demand sets alternate between only a few possible vectors; again we can binary search for the number of iterations that we need to take to reach a new demand vector.

4 COMPUTATIONAL PERFORMANCE OF OUR ALGORITHM

In this section, we discuss the computational performance of our algorithm, in particular in comparison to the previous state of the art. In Subsection 4.1, we describe our experiments comparing the algorithms. Then, in Subsection 4.2 we focus on how the improvements described in Section 3.2 compare to our basic algorithm.

4.1 Comparing with the benchmark

In this subsection we compare our algorithm with the benchmark algorithm. As we will soon see (Figure 2), our algorithm is much faster. To understand why, we also consider two intermediate algorithms. Overall, the algorithms we compare are:

Benchmark The (previous) state-of-the-art commercial algorithm; see description in Appendix B.

Vanilla Vanilla tatonnement, aka without optimizations such as tabu search, individual price adjustments, or optimal budget perturbations, that are used in other variants that we consider (for pseudocode, see Algorithm 1 with $\epsilon = 0$).

Basic Our basic algorithm, aka the algorithm descirbed in Subsection 3.1, which adds optimal budget perturbations to tatonnement, but without further optimizations described in Subsection 3.2.

Fast Our final algorithm, including optimizations from Subsection 3.2.

Choice of parameters. All the algorithms perturb the students' budgets: Benchmark and Vanilla use only random perturbations, while Basic and Fast use a mix of random and optimal budget perturbations. Larger budget perturbations tend to make the algorithmic task of finding an (approximate) equilibrium easier: At one extreme, the existence proof holds even with infinitesimal budget perturbations; at the other extreme, with infinite budget perturbations the mechanism reduces to Random Serial Dictatorship which is computationally trivial.

For the sake of a fair comparison of the algorithms, we ensure the algorithms have the same magnitude of total budget perturbations:

- For Benchmark and Vanilla we draw the budgets uniformly i.i.d. from $[1, 1 + \beta]$.
- For Basic and Fast, we draw the *base budgets* uniformly i.i.d. from $[1 + \beta/4, 1 + 3\beta/4]$, and allow further optimal budget perturbation of magnitude $\epsilon = \beta/4$.

In particular, we consider $\beta = 0.04$ here. We use a large step size of 0.005 for the warm start of Fast. For the remaining parameters, we replicate that of Budish et al. [2017] for Benchmark and use the same step size of $\delta = 0.002$ for all the algorithms (except in the warm start).

Instances. For the computational experiments, we use the largest instances available to us:

- **Law** - A law school with about 500 students and 125 classes/sections
- **Ivy Large** - A large Ivy-league business school with several thousand students and several hundred classes/sections
- **Ivy Huge** - A large Ivy-league business school with several thousand students and around a thousand classes/sections and lots of challenging constraints

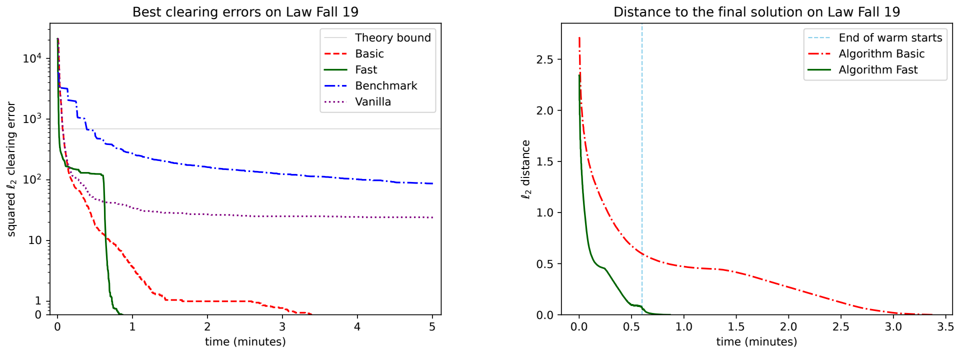

Fig. 2. For each of the algorithms tested and time t, we plot the best clearing error obtained by the algorithm up to time t (left). For Basic and Fast, we plot the distance between the found solution and the final solution with respect to the running time (right).

A typical student's schedule in those instances has around 5 courses, although some schools bid in the fall for the entire year (so around 10 courses total).

Findings. Figure 2 presents the average[7] best clearing error found by the four algorithms with respect to time.

Observation 1: optimal budget perturbations help, significantly. From the plot, we can see that the two algorithms with budget perturbation (i.e. Basic and Fast) significantly outperform those without budget perturbation (i.e. Benchmark and Vanilla) on clearing errors — when our basic algorithm terminates with a $(0, \beta)$-CEEI, the average best clearing errors found by Benchmark and Vanilla are still larger than 30. Therefore, we believe budget perturbation is the most important ingredient introduced in our algorithms for clearing the market.

Observation 2: individual price adjustments do not help. In Figure 2 and 3 it can be observed that Vanilla obtains lower clearing error than Benchmark on all the instances we tested. More precisely if we run both algorithms long enough then Benchmark does eventually catch up (Figure 1), but only at a time scale much larger than our algorithm needs to converge to zero clearing error.

The observation that individual price adjustments slow the algorithm is very surprising since it stands in contrast to the findings of Othman et al. [2010]. It is even more puzzling because in each iteration Algorithm 3 chooses the (myopically) better of updating all the prices or one, so it seems intuitive that individual price adjustments can only help. One simple reason is that the time-per-iteration of Benchmark is significantly slower compared to Vanilla[8]. (In Othman et al. [2010] this issue did not come up because both algorithms were tested in terms of number of iterations.)

Another interesting piece of this puzzle is that Othman et al. [2010] thought of tatonnement as computer scientists often do: a direction for a local search algorithm with the objective of

[7]This is an average over different draws of students' initial budgets; we use 20 runs for Basic and Fast, and 10 runs for the slower Benchmark and Vanilla.

[8]We note that in the experiments we actually measure a proprietary variant of the benchmark algorithm whose time-per-iteration has been heavily optimized. We also note that when we tried to combine individual price adjustments with optimized budget perturbations, the computational overhead of individual price adjustments was even worse because we had to resolve for optimal budget perturbation for each individual price adjustment.

minimizing the clearing error (indeed, in some utility models it exactly corresponds to a (sub)-gradient of the clearing error, e.g. Cheung et al. [2020]; Kelso and Crawford [1982]; Leme and Wong [2020]). When viewed in this way, it may sometimes get stuck in local minima. However, inspired by economics, we view tatonnement as a distributed process where we modify the price of each course simultaneously without worrying about the global clearing error. When used in this way, it may sometimes locally *increase* the clearing error; but our experiments suggest that it can effectively escape those local minima, leading to better solutions faster. Indeed the clearing error does not monotonically decrease throughout the run of our algorithm.

Observation 3: Our optimized algorithm initially has larger clearing errors than our basic algorithm. In the plot, it is easy to spot the sharp drop of clearing error for Fast — this is exactly the time when Fast switches from the warm start to the second phase. It may appear that the time we spend on the warm start is too long, but this is in fact not the case, as we discuss in Subsection 4.2.

4.2 Our optimized algorithm

As mentioned in Observation 3 above, Fast initially makes slow progress in terms of market clearing error during the warm start compared to Basic. We argue that it still makes important progress towards the eventual equilibrium even if we don't see that in the clearing error. First, we note that although we did not carefully optimize the cutoff of the warm start (we heuristically set it to approximately $1/\delta$, where δ is the step size), Fast seems to work really well in practice!

More interestingly, we can measure the progress towards the eventual equilibrium in terms of distance in price space (instead of current clearing error). In Figure 2 and 4, we show that the optimized algorithm approaches an ultimate equilibrium much faster when measured in price space distance. We also observe that at the time that the algorithm switches phases, we're already quite close in price space, so a more refined (small step size) second phase is appropriate. (Of course we unfortunately only know the distance to eventual equilibrium in hindsight, otherwise this could have made for a great heuristic approach to knowing when to switch phases!)

5 EXAMPLES OF SUCCESSFUL MANIPULATIONS

How can strategic students manipulate SP-L mechanisms in realistic instances? To really understand the nature of manipulations found by the algorithm, in this section we report insights from our qualitative analysis that zooms in on specific manipulations, one for each variant of our algorithm. Our representation is over simplified with made up student names and courses — but they're all based on manipulations discovered by our manipulation-finding algorithm on almost-real data (see Remark 4). Of course, each case study may not be representative of all possible manipulations. However, we find this methodology quite helpful for gaining intuition. In particular, we were able to use the case study for classic EF-TB to extract the simple manipulation described in the introduction, and propose the contested EF-TB criterion in response.

All the examples are described in what can be informally thought of as an "almost-large-market": from the perspective of an individual student, course prices are approximately set by other students, but a student's reported preferences can nudge them infinitesimally towards a market clearing equilibrium.

Remark 4 (5-additive utilities)**.** Unfortunately, the original preferences in the true instances are incredibly complicated, mostly due to various constraints imposed by the schools (e.g. avoiding courses with conflicting meeting times, meeting minimum unit requirements, etc). To keep the case study analyses tractable, we repeat the manipulation finding experiments, but on modified preferences that ignore all those constraints and simply assume that the students utility is *5-additive*, i.e. each student wants the schedule of 5 courses that maximizes the total weight; here we use the original course weights reported by the students, but ignore all conflicts and requirements. Formally, for every student $i \in [n]$, the new utility function can be described by tuple $(\boldsymbol{w}', valid', req')$ where $\boldsymbol{w}' = (w_j)_{j\in[m]}$, and for every bundle $\boldsymbol{x} \in 2^{[m]}$,

$$valid'(\boldsymbol{x}) = \begin{cases} 1 & \|\boldsymbol{x}\|_1 \le 5 \\ 0 & \|\boldsymbol{x}\|_1 > 5 \end{cases}$$
$$req'(\boldsymbol{x}) = 0$$

5.1 A simple manipulation without EF-TB

Example 2 (Manipulation without EF-TB constraints).
Alice and Bob both want only the last seat of CS161. Because of other students' demand, ECON101 is full but has a low price. With true reporting CS161 could go to either Alice or Bob, depending on the random initial budgets.

Bob can manipulate his preferences to report that he wants ECON101 as his second course (aka Bob's manipulated preferences are:

$$\{CS161, ECON101\} \succcurlyeq \{CS161\} \succcurlyeq \{ECON101\}.$$

Under the manipulated preferences, since ECON101 is cheap, the only way it will not be allocated to Bob is if Bob exhausts his budget on CS161. Even if Alice's initial budget is higher, the optimal unconstrained budget perturbation will make sure that Bob's final budget is higher than Alice's, in which case he gets CS161 and Alice gets nothing — an equilibrium.

For this manipulation to work, Bob had to know that at equilibrium prices ECON101 is already exactly filled by other students — a knowledge he is unlikely to have in realistic bidding. Indeed, suppose that demand for ECON101 was higher this semester, the algorithm raised its price, and now it is missing exactly one student: in this case the optimal budget perturbation would have to ensure that Bob does get into ECON101, which can be achieved by perturbing budgets *against* Bob and letting Alice grab the last seat in CS161.

However, this manipulation is fairly robust if the price of ECON101 is always very low ("ECON101 tends to have a low price" is a general statistic that a student could plausibly learn from historical bids). In this case, even if ECON101 is missing a student, Bob's budget may be larger than Alice's by a sufficient margin to afford both CS161 and ECON101. So when ECON101 is undersubscribed, the budget perturbation could go either way[9], but it *always* goes in favor of the strategic Bob when ECON101 is oversubscribed.

5.2 A simple manipulation with classic EF-TB

Example 3 (Manipulation with classic EF-TB constraints).
Alice and Bob both want the last seat in the popular CS161 course. But Alice is even more excited about taking independent research units with her advisor, of which there is unlimited supply, so the price is always zero (aka this is an uncontested *course); she would like to take both. Bob's second choice is ECON101; because of other students' demand, ECON101 is full but has a low price.*

With true preferences, since ECON101 is cheap, the only way it will not be allocated to Bob is if Bob exhausts his budget on CS161. Thus even if Alice ranks higher, the optimal budget perturbation sets her budget lower than Bob. In this case Alice always gets independent study (only), and Bob always gets CS161 (only) — an equilibrium.

If Alice misreports her preferences to rank independent research units lower than CS161, she would envy Bob whenever he gets the last seat to CS161. Whenever her initial budget is higher, this prevents

[9] Alice would have a slight advantage due to our particular tie-breaking rule.

the optimal budget perturbation from driving it below Bob's, increasing her chances of getting the last seat in CS161.

Note that ranking the uncontested course (independent research units) lower *never hurts* Alice. So this manipulation is profitable in expectation for Alice even if she only has very noisy information about her rank and other students' demand.

Interestingly, this manipulation works *because* of the EF-TB constraints that we introduce to prevent manipulations!

5.3 A simple manipulation with contested EF-TB

We now discuss a simple manipulation that our algorithm discovered even with the contested EF-TB; while some profitable manipulations may exist in practice, as we show in Section 6 they're extremely rare.

Example 4 (Manipulation with contested EF-TB constraints).

Many students, including Bob, like to take CS161 and ECON101 together, but they rank ECON101 over CS161. Alice already took ECON101 last semester, so she only wants the last seat in CS161. Because of other students' demand, ECON101 is full but has a very low price.

With true preferences, if Bob's budget is higher than Alice's by a margin greater than the price of ECON101, he could afford both CS161 and ECON101, leaving Alice with nothing.

Alice could manipulate her preferences to report that she wants ECON101 as her second course. Bob always gets ECON101, because it's cheap and it's his top priority. Whenever Alice doesn't get CS161 she has to get ECON101 (because its price is cheap); if both Bob and Alice get ECON101, the course becomes oversubscribed, which would cause a clearing error. Therefore the optimal budget perturbation would make sure Bob's budget is low enough compared to Alice's that he can't afford both courses: he will get ECON101, and Alice will get CS161 - an equilibrium.

As with Example 2, this manipulation *does* pose some risk — if ECON101 is undersubscribed, the optimal budget perturbation may reduce Alice's budget below the price of CS161 so that she has to take ECON101. However, if ECON101 is very cheap, there's always also the small perturbation that increases Alice's budget so that she can afford both ECON101 and CS161 (while Bob only affords ECON101). So, because of the asymmetry in the prices of CS161 and ECON101, if ECON101 is oversubscribed, this manipulation can increase Alice's chances of getting into CS161, but if ECON101 is undersubscribed, Alice's chances aren't hurt by much.

5.4 Can A-CEEI with random budget perturbation be manipulated?

All the manipulations that we found seemed tied to our optimal budget perturbations procedure. So it is natural to ask whether the original A-CEEI[10] without optimal budget perturbations can also be manipulated

We speculatively conjecture in practice that manipulations similar to those described in Example 4 can also be profitable without optimal budget perturbations: the simplest way to think about this example is that Alice adding ECON101 to her demand should (slightly) increase the price of ECON101; this makes it less likely for Bob to be able to afford both ECON101 and CS161; this in turn makes it more likely that Alice can get a seat in CS161.

While it seems plausible that such manipulations are profitable in practice, note that for our algorithm with contested EF-TB, our numerical analysis in Section 6 suggests that they are extremely rare. Either way, at this point for A-CEEI without optimal budget perturbations we can only

[10]For fair comparison, note that the previous state-of-the-art practical implementation augmented the original A-CEEI mechanism with two stages that had other incentive issues (see Remark 1).

speculate: we could not run our manipulation-finding algorithm with the original A-CEEI algorithm that only uses random budget perturbation because this algorithm is too slow. (The manipulation-finding algorithm needs to make many calls to the A-CEEI algorithm to evaluate different possible deviations for every student.)

6 QUANTITATIVE ANALYSIS OF MANIPULABILITY

In this section we ask whether in practice strategic students bidding in A-CEEI have an incentive to deviate from truth-telling. To address this question, we model the student's process for choosing her bids with an optimization algorithm that can try different bid manipulations and test whether they improve the student's utility.

6.1 The manipulation-finding algorithm

It is still intractable to consider all possible bid-manipulations. This is true both for a student optimizing their bid, and for our algorithms in our experiments. Instead, we restrict attention to a simple hill-climbing algorithm that iteratively looks for a course whose bid-manipulation would increase the utility, in expectation over uncertainty; see Algorithm 2 for details. To validate our approach, we benchmark our manipulation-finding algorithm on a course allocation mechanism that is known to be manipulable [Budish and Cantillon 2012], which is used at Harvard Business School (we henceforth refer to this mechanism as HBS).

To simplify notations, we explicitly define the notation of randomized mechanism for course allocation problem.

Definition 6 (Randomized mechanism). A randomized mechanism $\mathcal{M}$ for course allocation problem can be characterized by a function $f_{\mathcal{M}} : (\boldsymbol{u}, \boldsymbol{c}, \boldsymbol{r}) \mapsto (\boldsymbol{a}_i)_{i=1}^n$, which takes the input of the course allocation market $(\boldsymbol{u}, \boldsymbol{c})$ and randomness $\boldsymbol{r}$, and outputs an allocation $(\boldsymbol{a}_i)_{i=1}^n = (\boldsymbol{M}_i(\boldsymbol{u}, \boldsymbol{c}, \boldsymbol{r}))_{i=1}^n$ where $\boldsymbol{M}_i(\boldsymbol{u}, \boldsymbol{c}, \boldsymbol{r}) \subseteq [m]$ denotes the bundle that the mechanism allocated to student i.

For example, HBS is a randomized mechanism, where the randomness $\boldsymbol{r}$ is used to determine the order of students in the random serial dictatorship process. Our A-CEEI algorithm (Algorithm 1) with fixed parameters can also be seen as a randomized mechanism, since an allocation can be uniquely determined based on its output (prices and budgets), and the base budgets are determined by randomness $\boldsymbol{r}$.

Remark 5. Note that in Definition 6, we do not require the allocation generated by the mechanism to be feasible (i.e., some courses might be oversubscribed). That is because no A-CEEI algorithm can guarantee to always output a price with zero clearing error. Nevertheless, our algorithm was observed to always obtain a feasible solution in all instances we encountered, as mentioned before.

In our experiments on manipulability of a specific mechanism, we iteratively run (a variant of) Algorithm 2 multiple times with respect to different parameters $H = \{\eta_1, \eta_2, ...\}$ for every student, under the different uncertainty models described later in Definition 7.

Practical optimizations. Since computing the exact expected value in the step 4 is generally intractable, we have to use samples to estimate it. For computational efficiency, we only use 5 samples for each estimation in following experiments, which is a fairly small number. As a result, some profitable manipulations might be missed, and some non-profitable manipulations can be reported incorrectly. Nevertheless, we will use a larger number of samples to test the statistical significance of those manipulations found by Algorithm 2.

We also use another optimization to reduce the number of times we have to solve for an A-CEEI, we search for manipulations in parallel: in each run of Algorithm 2 we actually consider a subset of several students and who are trying out their deviation on the same instance (students subsets are

ALGORITHM 2: Find a profitable manipulation for a student

Inputs: a randomized mechanism $\mathcal{M}$, student i and its utility function u_i, (previous best manipulation v_0), the criteria for profitable manipulation (resampled randomness $(\boldsymbol{u}_{-i}, \boldsymbol{c}, \mathcal{R})$ or population $(\mathcal{U}_{-i}, \boldsymbol{c}, \mathcal{R})$)
Outputs: a profitable manipulation u'_i
Parameters: a local update coefficient η
Algorithm:

(1) Let $v_0 \leftarrow u$ (or the best manipulation found in previous iterations with different η).
(2) Denote the description for v_0 by $(\boldsymbol{w}, valid, req)$.
(3) Try to increase or decrease the weight w_j for each course j in v_0 to obtain new misreports $V = \{v_{j,\pm 1}\}_{j\in[m]}$. Each $v_{j,k}$ can be described by $(\boldsymbol{w}', valid, req)$ where $w'_{j'} = \begin{cases} w_j & j' \neq j \\ \eta^k w_j & j' = j \end{cases}$ for $j' \in [m]$, $k \in \{\pm 1\}$.
(4) Let $v^* = \begin{cases} \arg\max_{v\in V\cup\{v_0\}} \mathsf{E}_{\boldsymbol{r}\sim\mathcal{R}}[u_i(\boldsymbol{M}_i([v_j, \boldsymbol{u}_{-i}], \boldsymbol{c}, \boldsymbol{r}))] & \text{resampled randomness,} \\ \arg\max_{v\in V\cup\{v_0\}} \mathsf{E}_{\boldsymbol{u}_{-i}\sim\mathcal{U}_{-i},\boldsymbol{r}\sim\mathcal{R}}[u_i(\boldsymbol{M}_i([v_j, \boldsymbol{u}_{-i}], \boldsymbol{c}, \boldsymbol{r}))] & \text{resampled population.} \end{cases}$
(5) If $v^* = v_0$, terminate with v_0 as the best manipulation found when $v_0 \neq u$, otherwise return failed.
(6) Otherwise, update $v_0 \leftarrow v^*$ and go back to step 2.

shuffled in each iteration to ensure that the signal corresponds to the deviation by the student and not to particular deviations by their peers).

Handling false positive manipulations. For any manipulation that is profitable on average in the exploration phase (which, as mentioned above is very noisy), we run it for 33 iterations, and then 67 more if it is still profitable. At this point we eliminated manipulations whose profitability did not meet .05 p-value statistical significance.

The original experiments had many false positives, and we expect as many as 5% of them would to survive the p-value test. So for all manipulations that survived the first p-value test, we run 100 additional iterations, and report manipulations that still have p-value .05.

6.2 Modeling students' uncertainty

In practice, while students often know the course capacities, they tend to have limited, aggregate information about historical bids; furthermore it is impossible for the students to know the algorithm's randomness at the time of bidding. We therefore study the students' decision process under two different models of uncertainty: *Resampled randomness* is the most conservative model of students' uncertainty — even if they have perfect knowledge of their peers' preferences, they never know the randomness of the algorithm (in particular, the random budget perturbation) at the time of bidding. *Resampled population* adds the students' uncertainty about their peers' preferences. See details of both models in Definition 7.

Definition 7 (Profitable manipulation)**.** For a randomized mechanism $\mathcal{M}$, student i's original utility function u_i and some manipulation u'_i, we say the manipulation from u_i to u'_i is *profitable*

- under *resampled randomness* $(\boldsymbol{u}_{-i}, \boldsymbol{c}, \mathcal{R})$: profitable in expectation under the market $([u_i, \boldsymbol{u}_{-i}], \boldsymbol{c})$ and randomness $\boldsymbol{r}$ where $r \sim \mathcal{R}$, i.e.,

$$\mathsf{E}_{\boldsymbol{r}\sim\mathcal{R}}[u_i(\boldsymbol{M}_i([u'_i, \boldsymbol{u}_{-i}], \boldsymbol{c}, \boldsymbol{r}))] > \mathsf{E}_{\boldsymbol{r}\sim\mathcal{R}}[u_i(\boldsymbol{M}_i([u_i, \boldsymbol{u}_{-i}], \boldsymbol{c}, \boldsymbol{r}))].$$

- under *resampled population* $(\mathcal{U}_{-i}, \boldsymbol{c}, \mathcal{R})$: profitable in expectation under the market $([u_i, \boldsymbol{u}_{-i}], \boldsymbol{c})$ and randomness $\boldsymbol{r}$ where $\boldsymbol{u}_{-i} \sim \mathcal{U}_{-i}$ and $\boldsymbol{r} \sim \mathcal{R}$, i.e.,

$$\mathsf{E}_{\boldsymbol{u}_{-i}\sim\mathcal{U}_{-i},\boldsymbol{r}\sim\mathcal{R}}[u_i(\boldsymbol{M}_i([u'_i, \boldsymbol{u}_{-i}], \boldsymbol{c}, \boldsymbol{r}))] > \mathsf{E}_{\boldsymbol{u}_{-i}\sim\mathcal{U}_{-i},\boldsymbol{r}\sim\mathcal{R}}[u_i(\boldsymbol{M}_i([u_i, \boldsymbol{u}_{-i}], \boldsymbol{c}, \boldsymbol{r}))].$$

Given a known market $(\boldsymbol{u}, \boldsymbol{c})$, for every student i we let $\mathcal{U}_{-i}$ be the distribution that re-samples $n-1$ other students, independently and with replacement, from of the true population $\boldsymbol{u}_{-i}$ of other students.

6.3 Experiment set-up

Choice of parameters. For our A-CEEI algorithm (Algorithm 1), we always seek for a $(0, \beta)$-CEEI, where $\beta = 0.04$. Same as before, we draw base budgets $\boldsymbol{b}_0$ uniformly i.i.d. from $[1+\beta/4, 1+3\beta/4]$ (i.e., $\boldsymbol{b}_0 = \boldsymbol{r} \sim \mathcal{R} = \mathcal{U}^n[1+\beta/4, 1+3\beta/4]$) and allow further optimal budget perturbation of magnitude $\epsilon = \beta/4$. We choose different step sizes for different instances to speed up computation. Specifically, we use $\delta = 0.0005$ for Ivy Small, $\delta = 0.001$ for Biz, and $\delta = 0.002$ for Small.

For the manipulation-finding algorithm (Algorithm 2), we let the set H of local update coefficients be $\{16000, 800, 40, 2\}$.

The instances. Despite various optimizations, our manipulation-finding ultimately requires solving a very large number of A-CEEI instances, which can require significant computational resources even with our efficient algorithm. Therefore we run this experiment on relatively small instances:

- **Small** - A small business school with about 150 students and 50 classes/sections (Fall 2021).
- **Ivy Small** - An Ivy-league business school with about 500 students and 50 classes/sections (Fall 2020).
- **Biz** - A business school with about 500 students and 50 classes/sections (Fall 2020).

6.4 Statistical findings

Our numerical results are summarized in Table 1. Our algorithm successfully finds profitable manipulations for the benchmark manipulable HBS mechanism on all instances. On instances Small and Biz it finds almost no statistically significant profitable manipulations for any variant of our A-CEEI algorithm. For Ivy Small, it finds that about 7% of the students can gain as much as around 10% from misreporting their preferences with no EF-TB constraints, and somewhat less with classic EF-TB constraints; however with contested EF-TB profitable manipulations were extremely rare.

Instance	Uncertainty	Mechanism: No EF-TB #	No EF-TB Gain	Classic EF-TB #	Classic EF-TB Gain	Contested EF-TB #	Contested EF-TB Gain	HBS #	HBS Gain
Small	Randomness	1	0.04%	0	-	1	0.04%	66	8.0 ± 2.1%
	Population	0	-	0	-	0	-	63	9.0 ± 2.1%
Ivy Small	Randomness	21	13.5 ± 3.6%	15	8.5 ± 2.2%	1	0.02%	107	23.5 ± 3.6%
	Population	20	7.4 ± 1.4%	11	4.7 ± 1.0%	0	-	117	20.7 ± 3.0%
Biz	Randomness	0	-	0	-	0	-	87	26.3 ± 4.0%
	Population	1	0.7%	0	-	0	-	64	21.6 ± 2.8%

Table 1. Number of statistically significant manipulations found by our algorithm and their average relative gain (± standard error) in utility.

Remark 6. We are not sure why profitable manipulations for no EF-TB and classical EF-TB manipulations were found only for Ivy Small and not for the other schools. We observe at equilibrium Ivy Small has two courses that are in very high demand — their price exceeds the entire budgets of some students.

REFERENCES

Mohammad Akbarpour, Scott Duke Kominers, Kevin Michael Li, Shengwu Li, and Paul Milgrom. 2020. Investment Incentives in Truthful Approximation Mechanisms. *SSRN* (2020). https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3544100

Georgios Amanatidis, Haris Aziz, Georgios Birmpas, Aris Filos-Ratsikas, Bo Li, Hervé Moulin, Alexandros A. Voudouris, and Xiaowei Wu. 2022. Fair Division of Indivisible Goods: A Survey. *CoRR* abs/2208.08782 (2022). https://doi.org/10.48550/arXiv.2208.08782 arXiv:2208.08782

Georgios Amanatidis, Georgios Birmpas, Aris Filos-Ratsikas, Alexandros Hollender, and Alexandros A. Voudouris. 2021. Maximum Nash welfare and other stories about EFX. *Theor. Comput. Sci.* 863 (2021), 69–85. https://doi.org/10.1016/j.tcs.2021.02.020

Eshwar Ram Arunachaleswaran, Siddharth Barman, and Nidhi Rathi. 2022. Fully Polynomial-Time Approximation Schemes for Fair Rent Division. *Mathematics of Operations Research* 47, 3 (2022), 1970–1998. https://doi.org/10.1287/moor.2021.1196

Eduardo M Azevedo and Eric Budish. 2019. Strategy-proofness in the Large. *The Review of Economic Studies* 86, 1 (08 2019), 81–116. https://doi.org/10.1093/restud/rdy042 arXiv:https://academic.oup.com/restud/article-pdf/86/1/81/27285295/rdy042.pdf

Haris Aziz. 2015. Competitive equilibrium with equal incomes for allocation of indivisible objects. *Oper. Res. Lett.* 43, 6 (2015), 622–624. https://doi.org/10.1016/j.orl.2015.10.001

Moshe Babaioff, Noam Nisan, and Inbal Talgam-Cohen. 2021. Competitive Equilibrium with Indivisible Goods and Generic Budgets. *Math. Oper. Res.* 46, 1 (2021), 382–403. https://doi.org/10.1287/moor.2020.1062

Siddharth Barman, Sanath Kumar Krishnamurthy, and Rohit Vaish. 2018. Finding Fair and Efficient Allocations. In *Proceedings of the 2018 ACM Conference on Economics and Computation, Ithaca, NY, USA, June 18-22, 2018*, Éva Tardos, Edith Elkind, and Rakesh Vohra (Eds.). ACM, 557–574. https://doi.org/10.1145/3219166.3219176

Shant Boodaghians, Bhaskar Ray Chaudhury, and Ruta Mehta. 2022. Polynomial Time Algorithms to Find an Approximate Competitive Equilibrium for Chores. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, Joseph (Seffi) Naor and Niv Buchbinder (Eds.). SIAM, 2285–2302. https://doi.org/10.1137/1.9781611977073.92

Sylvain Bouveret, Katarína Cechlárová, Edith Elkind, Ayumi Igarashi, and Dominik Peters. 2017. Fair Division of a Graph. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, Carles Sierra (Ed.). ijcai.org, 135–141. https://doi.org/10.24963/ijcai.2017/20

Sylvain Bouveret, Yann Chevaleyre, and Nicolas Maudet. 2016. Fair Allocation of Indivisible Goods. In *Handbook of Computational Social Choice*, Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia (Eds.). Cambridge University Press, 284–310. https://doi.org/10.1017/CBO9781107446984.013

Simina Brânzei, Hadi Hosseini, and Peter Bro Miltersen. 2015. Characterization and Computation of Equilibria for Indivisible Goods. In *Algorithmic Game Theory - 8th International Symposium, SAGT 2015, Saarbrücken, Germany, September 28-30, 2015, Proceedings (Lecture Notes in Computer Science, Vol. 9347)*, Martin Hoefer (Ed.). Springer, 244–255. https://doi.org/10.1007/978-3-662-48433-3_19

Simina Brânzei, Yuezhou Lv, and Ruta Mehta. 2016. To Give or Not to Give: Fair Division for Single Minded Valuations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, Subbarao Kambhampati (Ed.). IJCAI/AAAI Press, 123–129. http://www.ijcai.org/Abstract/16/025

Eric Budish. 2010. The Combinatorial Assignment Problem: Approximate Competitive Equilibrium from Equal Incomes. (2010). https://faculty.chicagobooth.edu/eric.budish/research/budish-approxceei-Aug2010.pdf Working paper version of paper with the same title..

Eric Budish. 2011. The Combinatorial Assignment Problem: Approximate Competitive Equilibrium from Equal Incomes. *Journal of Political Economy* 119, 6 (2011), 1061 – 1103. http://EconPapers.repec.org/RePEc:ucp:jpolec:doi:10.1086/664613

Eric Budish, Gérard P. Cachon, Judd B. Kessler, and Abraham Othman. 2017. Course Match: A Large-Scale Implementation of Approximate Competitive Equilibrium from Equal Incomes for Combinatorial Allocation. *Oper. Res.* 65, 2 (2017), 314–336. https://doi.org/10.1287/opre.2016.1544

Eric Budish and Estelle Cantillon. 2012. The Multi-unit Assignment Problem: Theory and Evidence from Course Allocation at Harvard. *The American Economic Review* 102, 5 (2012), 2237–2271. http://www.jstor.org/stable/41724620

Eric Budish and Judd B. Kessler. 2022. Can Market Participants Report Their Preferences Accurately (Enough)? *Manag. Sci.* 68, 2 (2022), 1107–1130. https://doi.org/10.1287/mnsc.2020.3937

Ioannis Caragiannis, Nick Gravin, and Xin Huang. 2019. Envy-Freeness Up to Any Item with High Nash Welfare: The Virtue of Donating Items. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019*, Anna Karlin, Nicole Immorlica, and Ramesh Johari (Eds.). ACM, 527–545. https://doi.org/10.1145/3328526.3329574

Bhaskar Ray Chaudhury, Jugal Garg, Peter McGlaughlin, and Ruta Mehta. 2022a. Competitive Equilibrium with Chores: Combinatorial Algorithm and Hardness. In *EC '22: The 23rd ACM Conference on Economics and Computation, Boulder, CO, USA, July 11 - 15, 2022*, David M. Pennock, Ilya Segal, and Sven Seuken (Eds.). ACM, 1106–1107. https://doi.org/10.1145/

3490486.3538255

Bhaskar Ray Chaudhury, Jugal Garg, Peter McGlaughlin, and Ruta Mehta. 2022b. On the Existence of Competitive Equilibrium with Chores. In *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA (LIPIcs, Vol. 215)*, Mark Braverman (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 41:1–41:13. https://doi.org/10.4230/LIPIcs.ITCS.2022.41

Bhaskar Ray Chaudhury, Jugal Garg, Kurt Mehlhorn, Ruta Mehta, and Pranabendu Misra. 2021. Improving EFX Guarantees through Rainbow Cycle Number. In *EC ’21: The 22nd ACM Conference on Economics and Computation, Budapest, Hungary, July 18-23, 2021*, Péter Biró, Shuchi Chawla, and Federico Echenique (Eds.). ACM, 310–311. https://doi.org/10.1145/3465456.3467605

Xi Chen, Xiaotie Deng, and Shang-Hua Teng. 2009. Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM (JACM)* 56, 3 (2009), 1–57.

Yun Kuen Cheung, Richard Cole, and Nikhil R. Devanur. 2020. Tatonnement beyond gross substitutes? Gradient descent to the rescue. *Games Econ. Behav.* 123 (2020), 295–326. https://doi.org/10.1016/j.geb.2019.03.014

Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. 2009. The complexity of computing a Nash equilibrium. *SIAM J. Comput.* 39, 1 (2009), 195–259.

Franz Diebold, Haris Aziz, Martin Bichler, Florian Matthes, and Alexander W. Schneider. 2014. Course Allocation via Stable Matching. *Bus. Inf. Syst. Eng.* 6, 2 (2014), 97–110. https://doi.org/10.1007/s12599-014-0316-6

Duncan K. Foley. 1967. Resource allocation and the public sector. *Yale economic essays* 7, 1 (1967), 45–98. mit Bibliogr..

Jugal Garg, Martin Hoefer, Peter McGlaughlin, and Marco Schmalhofer. 2021. When Dividing Mixed Manna Is Easier Than Dividing Goods: Competitive Equilibria with a Constant Number of Chores. In *Algorithmic Game Theory - 14th International Symposium, SAGT 2021, Aarhus, Denmark, September 21-24, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 12885)*, Ioannis Caragiannis and Kristoffer Arnsfelt Hansen (Eds.). Springer, 329–344. https://doi.org/10.1007/978-3-030-85947-3_22

Jonathan R. Goldman and Ariel D. Procaccia. 2014. Spliddit: unleashing fair division algorithms. *SIGecom Exch.* 13, 2 (2014), 41–46. https://doi.org/10.1145/2728732.2728738

Alexander S. Kelso and Vincent P. Crawford. 1982. Job Matching, Coalition Formation, and Gross Substitutes. *Econometrica* 50, 6 (1982), 1483–1504.

Renato Paes Leme and Sam Chiu-wai Wong. 2020. Computing Walrasian equilibria: fast algorithms and structural properties. *Math. Program.* 179, 1 (2020), 343–384. https://doi.org/10.1007/s10107-018-1334-9

Richard J. Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. 2004. On approximately fair allocations of indivisible goods. In *Proceedings 5th ACM Conference on Electronic Commerce (EC-2004), New York, NY, USA, May 17-20, 2004*, Jack S. Breese, Joan Feigenbaum, and Margo I. Seltzer (Eds.). ACM, 125–131. https://doi.org/10.1145/988772.988792

Sören Merting, Martin Bichler, and Aykut Uzunoglu. 2019. Assigning Course Schedules: About Preference Elicitation, Fairness, and Truthfulness. In *Proceedings of the 40th International Conference on Information Systems, ICIS 2019, Munich, Germany, December 15-18, 2019*, Helmut Krcmar, Jane Fedorowicz, Wai Fong Boh, Jan Marco Leimeister, and Sunil Wattal (Eds.). Association for Information Systems. https://aisel.aisnet.org/icis2019/data_science/data_science/15

Abraham Othman, Christos H. Papadimitriou, and Aviad Rubinstein. 2016. The Complexity of Fairness Through Equilibrium. *ACM Trans. Economics and Comput.* 4, 4 (2016), 20:1–20:19. https://doi.org/10.1145/2956583

Abraham Othman, Tuomas Sandholm, and Eric Budish. 2010. Finding approximate competitive equilibria: efficient and fair course allocation. In *9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), Toronto, Canada, May 10-14, 2010, Volume 1-3*, Wiebe van der Hoek, Gal A. Kaminka, Yves Lespérance, Michael Luck, and Sandip Sen (Eds.). IFAAMAS, 873–880. https://dl.acm.org/citation.cfm?id=1838323

C.H. Papadimitriou. 2007. The complexity of finding nash equilibria. In *Algorithmic Game Theory*, Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani (Eds.). Cambridge University Press, 29–51. https://doi.org/10.1017/CBO9780511800481

Dominik Peters, Ariel D. Procaccia, and David Zhu. 2022. Robust Rent Division. In *NeurIPS-22: Proc. 36th Annual Conference on Neural Information Processing Systems, 2022.*

Alexander Peysakhovich and Christian Kroer. 2019. Fair Division Without Disparate Impact. *CoRR* abs/1906.02775 (2019). arXiv:1906.02775 http://arxiv.org/abs/1906.02775

Benjamin Plaut and Tim Roughgarden. 2020. Almost Envy-Freeness with General Valuations. *SIAM J. Discret. Math.* 34, 2 (2020), 1039–1068. https://doi.org/10.1137/19M124397X

Alvin E. Roth. 2002. The Economist as Engineer: Game Theory, Experimentation, and Computation as Tools for Design Economics. *Econometrica* 70, 4 (2002), 1341–1378. https://doi.org/10.1111/1468-0262.00335 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/1468-0262.00335

Alvin E. Roth. 2007. Repugnance as a Constraint on Markets. *Journal of Economic Perspectives* 21, 3 (September 2007), 37–58. https://doi.org/10.1257/jep.21.3.37

Aviad Rubinstein. 2018. Inapproximability of Nash Equilibrium. *SIAM J. Comput.* 47, 3 (2018), 917–959. https://doi.org/10.1137/15M1039274

Ermis Soumalias, Behnoosh Zamanlooy, Jakob Weissteiner, and Sven Seuken. 2022. Machine Learning-powered Course Allocation. *CoRR* abs/2210.00954 (2022). https://doi.org/10.48550/arXiv.2210.00954 arXiv:2210.00954

Warut Suksompong. 2023. A characterization of maximum Nash welfare for indivisible goods. *Economics Letters* 222 (2023), 110956. https://doi.org/10.1016/j.econlet.2022.110956

William Thomson and Hal R. Varian. 1985. Theories of justice based on symmetry. *Social goals and social organization : essays in memory of Elisha Pazner* (1985), 107–129.

Hal R Varian. 1974. Equity, envy, and efficiency. *Journal of Economic Theory* 9, 1 (1974), 63–91. https://doi.org/10.1016/0022-0531(74)90075-1

Ellen Vitercik. 2021. *Automated algorithm and mechanism configuration.* Ph. D. Dissertation. Carnegie Mellon University. http://reports-archive.adm.cs.cmu.edu/anon/2021/abstracts/21-125.html

Zihe Wang, Zhide Wei, and Jie Zhang. 2020. Bounded Incentives in Manipulating the Probabilistic Serial Rule. (2020), 2276–2283. https://ojs.aaai.org/index.php/AAAI/article/view/5605

Mason Wright and Yevgeniy Vorobeychik. 2015. Mechanism Design for Team Formation. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, Blai Bonet and Sven Koenig (Eds.). AAAI Press, 1050–1056. http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9902

A ADDITIONAL RELATED WORK

Wright and Vorobeychik [2015] consider an adaptation of A-CEEI to team formation games (where agents need to be able to afford all their teammates). They empirically evaluate the gain from a manipulation using a different algorithm than the one we describe in Section 6. They report that on an instance based on 17 students ranking each other, the adapted A-CEEI was significantly *more* manipulable than the adapted HBS. In contrast, in our experiments for course allocation HBS is always more manipulable.

Diebold et al. [2014] survey different mechanisms for the course allocation problem. A newer work by Merting et al. [2019] focuses course allocation via a variant of Probabilistic Serial mechanism; incentives properties of Probabilistic Serial were also recently studied by Wang et al. [2020].

Brânzei et al. [2015] study the existence and complexity of exact CEEI with indivisible goods which are either perfect substitutes or complements (unfortunately courses in our instances satisfy neither). Boodaghians et al. [2022]; Chaudhury et al. [2022a,b]; Garg et al. [2021] study the problem of designing (approximate or exact) algorithms for computing a CEEI over divisible chores. Interesting variants of CEEI have also been studied by Aziz [2015]; Babaioff et al. [2021]; Brânzei et al. [2016]; Peysakhovich and Kroer [2019].

The fair allocation of indivisible goods has received a lot of attention recently, see for example the recent surveys of Bouveret et al. [2016] and Amanatidis et al. [2022]. In particular, there is a large body of theoretical work on existence, algorithms, and complexity of allocations satisfying EF1 (as does A-CEEI) or its strengthening EFX (e.g. [Amanatidis et al. 2021; Arunachaleswaran et al. 2022; Barman et al. 2018; Bouveret et al. 2017; Caragiannis et al. 2019; Chaudhury et al. 2021; Lipton et al. 2004; Plaut and Roughgarden 2020; Suksompong 2023]). Most of those works focus on cases where the agents' valuation functions satisfy nice properties ranging from additivity to subadditivity; some of these assumptions are applicable in other settings (see e.g. Spliddit [Goldman and Procaccia 2014]), but for course allocation real-world students utilities are quite complex — they are not sub-additive, and in fact not even monotone.

There has also been a recent body of work on the computation of economic equilibria, which could be considered the most practical application of Alvin Roth's idea of "The Economist as Engineer" [Roth 2002]. Peters et al. [2022], similar to our work, provides theoretical grounding and experimental results based on real data for the fair allocation of indivisible goods in a specific context (rent sharing). Works such as Cheung et al. [2020]; Kelso and Crawford [1982]; Leme and Wong [2020] (and many references therein) provide some theoretical grounding for what we found

ALGORITHM 3: Tabu search

Inputs: courses' capacities $\boldsymbol{c}$, students' preferences $\boldsymbol{u}$, initial budgets $\boldsymbol{b}_0 \in [1, 1+\beta]^n$.
Parameters: neighborhood function $\mathcal{N}(\boldsymbol{p})$, binary relation $\sim_p$ for prices.
Algorithm:

(1) Let $\boldsymbol{p} \leftarrow \text{uniform}(1, 1+\beta)^m, \mathcal{H} \leftarrow \emptyset$.
(2) If $\|z(\boldsymbol{u}, \boldsymbol{c}, \boldsymbol{p}, \boldsymbol{b}_0)\|_2 = 0$, terminate with $\boldsymbol{p}^* = \boldsymbol{p}$.
(3) Otherwise,
- include all equivalent prices of $\boldsymbol{p}$ into the history: $\mathcal{H} \leftarrow \mathcal{H} + \{\boldsymbol{p'} : \boldsymbol{p'} \sim_p \boldsymbol{p}\}$,
- update $\boldsymbol{p} \leftarrow \arg\min_{\boldsymbol{p'} \in \mathcal{N}(\boldsymbol{p}) - \mathcal{H}} \|z(\boldsymbol{u}, \boldsymbol{c}, \boldsymbol{p'}, \boldsymbol{b}_0)\|_2$, and then
- go back to step 2.

practically in developing our new search algorithm: that gradient descent-inspired tatonnement may be surprisingly effective for finding approximate equilibria despite worst-case complexity results.

B THE BENCHMARK ALGORITHM

Our benchmark for comparison is the (previous) commercial state of the art algorithm. We provide an overview of the algorithm here, and more details can be found in academic publications [Budish et al. 2017; Othman et al. 2010][11].

The basic idea for both the benchmark and our algorithm is tatonnement: increase the price of over-demanded courses, and decrease the prices of under-demanded courses (see also Algorithm 1 with $\epsilon = 0$). The benchmark (see 3 for pseudocode) augments the vanilla tatonnement with *tabu search*. That is, in each iteration it explores multiple neighbors in the price space and then updates to the one with minimum clearing error. To avoid repetitive searches, it will not consider price vectors in the neighborhood that are equivalent to some explored price. More specifically, two prices are defined to be equivalent if the student demands are identical under the prices, i.e. for any price vectors $\boldsymbol{p}, \boldsymbol{p'}$, $\boldsymbol{p} \sim_p \boldsymbol{p'}$ if

$$\forall \text{ student } i \in [n],\ \boldsymbol{a}_i(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{b_0}) = \boldsymbol{a}_i(\boldsymbol{u}, \boldsymbol{p'}, \boldsymbol{b_0}) \ .$$

The neighborhood function $\mathcal{N}(\boldsymbol{p})$ consists of the following two types of neighbors.

(1) *Gradient neighbors.* In gradient neighbors, prices are adjusted in proportion to the excess demand, i.e., for a time-varying set $\Delta \subseteq [0, 1]$, the gradient neighbors consist of the following prices:

$$\mathcal{N}^{\text{gradient}}(\boldsymbol{p}, \Delta) = \{\boldsymbol{p} + \delta \cdot \boldsymbol{z}(\boldsymbol{u}, \boldsymbol{c}, \boldsymbol{p}, \boldsymbol{b}) : \delta \in \Delta\} \ .$$

(2) *Individual price adjustment neighbors.* In individual price adjustment neighbors, the price vector is adjusted in only a small number of entries. For over-demanded courses, we consider increasing their prices so that their total demands decrease by one. For under-demanded courses, we consider setting them to zero or decreasing them by a fixed amount. To avoid each iteration exploring too many individual price adjustment neighbors, the algorithm randomly merges these individual price adjustments into 35 price updates.

[11]Our benchmark corresponds to the "first phase" of the algorithm described in Budish et al. [2017]. The remaining two phases are only used to cope with the fact that in practice clearing error is one-sided: undersubscribed courses are undesirable, but oversubscribed courses are absolutely infeasible, due e.g. to safety regulations for room capacities. This is a conservative comparison: we show that our algorithm is already faster than the first phase, and finds assignments with *zero* clearing error.

Instance	Avg. NSW			Avg. USW		
	Fast	Benchmark	Imp.	Fast	Benchmark	Imp.
Ivy Huge Fall 21	470.44	-	$+\infty$	552.92	541.5	2.11%
Law Fall 19	846.66	837.29	1.12%	1045.25	1036.76	0.82%
Law Fall 20	646.16	572.66	12.83%	848.04	764.39	10.94%
Law Fall 21	168.93	150.85	11.99%	194.15	176.24	10.16%
Ivy Small Fall 19	444.32	445.86	−0.35%	567.31	564.43	0.51%
Ivy Small Fall 20	421.54	415.86	1.37%	501.29	496.28	1.01%
Ivy Small Fall 21	437.49	444.79	−1.64%	535.47	540.59	−0.95%
Ivy Small Fall 22	462.49	462.77	−0.06%	521.4	522.94	−0.29%
Biz Fall 18	203.69	202.75	0.46%	232.53	231.77	0.33%
Biz Fall 19	191.53	189.0	1.34%	222.21	219.88	1.06%
Biz Fall 20	209.41	191.97	9.08%	227.47	212.09	7.25%
Biz Fall 21	234.95	234.62	0.14%	240.77	240.51	0.11%

Table 2. Comparing the (nash) social welfare.

C ECONOMIC EFFICIENCY OF OUR ALGORITHM

The focus of our work is improving the computational efficiency and exploring the incentive properties of the A-CEEI problem. However, we were also curious to see how our new algorithm compares to the old benchmark in terms of economic efficiency of the allocations. Specifically, we use weighted variants of social welfare and Nash social welfare. The weights correspond to students' (unperturbed) budgets: while in the original A-CEEI all agents have equal incomes, in practice schools often give students different budgets (mostly based on seniority) that correspond to priorities.

For this section we compare to the first-two-stages-only variant of the old benchmark: The first-stage-only benchmark may have oversubscribed courses, while the full 3-stage benchmark is not known to satisfy fairness or incentives (e.g. SP-L) properties, so those would not be fair benchmarks for comparisons.

The results are summarized in Table 2. At a high level, our algorithm is somewhat better on both metrics on most (although not all) instances.

Utilitarian Social Welfare. We compare the utilitarian social welfare (USW) of our solutions and the solutions found by the benchmark. For any algorithm A, the average USW is defined to be

$$\sum_{i\in[n]} \frac{b_i}{\sum_{i\in[n]} b_i} \cdot u_i(\boldsymbol{A}_i(\boldsymbol{u}, \boldsymbol{c}))\ ,$$

where b_i is the base budget of student i.

Nash Social Welfare. We also compare the nash social welfare (NSW) of our solutions and the solutions found by the whole baseline algorithm. For any algorithm A, the NSW is defined to be

$$\prod_{i\in[n]} u_i(\boldsymbol{A}_i(\boldsymbol{u}, \boldsymbol{c}))^{b_i/\sum_{i\in[n]} b_i}\ ,$$

where b_i is the base budget of student i.

D HARDNESS OF A-CEEI WITH REAL-WORLD UTILITY FUNCTIONS

In this appendix, we show finding an A-CEEI continues to be computationally intractable even in the special case of utilities as described in Eq. (1).

Theorem 2. *There exists a constant $\beta > 0$ such that finding (α, β)-CEEI is* PPAD*-complete for $\alpha = \Omega(n/m)$, even when the utilities are restricted to the form given in Eq.* (1).

Our proof reduces the generalized circuit problem to A-CEEI with utilities as in Eq. (1). The reduction consists of two steps. The first step is to reduce the generalized circuit problem [Rubinstein 2018] to one of its simplifications. In the simplified problem, we need to find an approximate solution of a (possibly cyclic) circuit only involving the SUM and NOT gates (Defintion 11), which are two out of the eight gates for the generalized circuit problem (Definition 9). To show the PPAD-hardness of the simplified problem (Theorem 3), we construct each of the remaining six gates by combinations of SUM and NOT gates. The second step is to reduce the simplified generalized circuit to A-CEEI. Our reduction follows the framework of Othman et al. [2016]. We construct gadgets for the SUM and NOT gates (Lemma 5). The gadgets contain *input courses*, an *output course*, *interior courses* and some students. An *output course* of a gadget can be *input courses* of other gadgets. In each gadget, the prices of the *input courses* and the *output course* respectively correspond to the values of the input variables and the output variable in the circuit. Our construction guarantees that the courses prices approximately satisfy the constraints of the variables of their corresponding variables. However, these gadgets cannot be directly integrated into a cyclic circuit. For each gadget, we require an upper bound for the number of additional students that want the *output course.* In the construction, the upper bound for the output course is strictly smaller than that of the input courses in the gadget. If the circuit is cyclic, there can be contradictions on magnitude of the upper bounds. To resolve this issue, Othman et al. [2016] introduces *course-size amplification gadget*, a COPY gadget in which the price of the output course approximately equals the price of the input course and the upper bound of the output course is twice as large as that of the input course. And finally, we finish the proof by constructing the *course-size amplification gadget* under utilies as in Eq. (1) (Lemma 6).

D.1 Simplified Generalized Circuit

In this subsection, we present our simplified generalized circuit and the PPAD-hardness solving it. We start from the definition of the generalized circuit problem as in Othman et al. [2016].

Definition 8 (Generalized circuits)**.** A generalized circuit $\mathcal{S} = (V, \mathcal{T})$ is a collection of nodes V and gates $\mathcal{T}$. Each gate $T \in \mathcal{T}$ is a 4-tuple (G, v_1, v_2, v), consisting of a type of gate $G \in \{G_{/2}, G_{\frac{1}{2}}, G_+, G_-, G_<, G_\wedge, G_\vee, G_\neg\}$, two input nodes $v_1, v_2 \in V \cup \{nil\}$ and an output node v. Each node $v \in V$ may be the output node for at most one gate, i.e., for every two gates $T = (G, v_1, v_2, v)$ and $T' = (G', v'_1, v'_2, v')$ in $\mathcal{T}$, $v \neq v'$.

Definition 9 (ϵ-Gcircuit problem)**.** Given a generalized circuit $\mathcal{S} = (V, \mathcal{T})$ and an assignment $\boldsymbol{x} : V \rightarrow \mathbb{R}$, we say $\boldsymbol{x}$ *ϵ-approximately satisfies $\mathcal{S}$* if $\boldsymbol{x}(v) \in [0, 1+\epsilon]$ for any $v \in V$, and for each gate $T = (G, v_1, v_2, v) \in \mathcal{T}$, we have that $|\boldsymbol{x}(v) - f_G(\boldsymbol{x}(v_1), \boldsymbol{x}(v_2))| \leq \epsilon$, where f is defined as follows:

(1) HALF: $f_{G_{/2}}(x) = x/2$
(2) VALUE: $f_{G_{\frac{1}{2}}} \equiv \frac{1}{2}$
(3) SUM: $f_{G_+}(x, y) = \min(x + y, 1)$
(4) DIFF: $f_{G_-}(x, y) = \max(x - y, 0)$
(5) LESS: $f_{G_<}(x, y) = \begin{cases} 1 & x > y + \epsilon \\ 0 & y > x + \epsilon \end{cases}$

(6) AND: $f_{G_\wedge}(x,y) = \begin{cases} 1 & (x > 0.9+\epsilon) \wedge (y > 0.9+\epsilon) \\ 0 & (x < 0.1-\epsilon) \vee (y < 0.1-\epsilon) \end{cases}$

(7) OR: $f_{G_\vee}(x,y) = \begin{cases} 1 & (x > 0.9+\epsilon) \vee (y > 0.9+\epsilon) \\ 0 & (x < 0.1-\epsilon) \wedge (y < 0.1-\epsilon) \end{cases}$

(8) NOT: $f_{G_\neg}(x) = 1-x$.

Remark 7. Definition 9 is slightly different from those in Othman et al. [2016] and Rubinstein [2018] on AND and OR gates. It can be observed that this problem (with fan-out 2) is harder than that in Rubinstein [2018] for sufficiently small constant $\epsilon > 0$ and is thus PPAD-complete for some constant $\epsilon > 0$.

Our simplification is that we restrict the set of types of gates to only consist of SUM and NOT. Formally, the definition of the circuit and the problem is as follows.

Definition 10 (Simplified Generalized Circuit)**.** A simplified generalized circuit $\mathcal{S} = (V, \mathcal{T})$ is a collection of nodes V and gates $\mathcal{T}$. Each gate $T \in \mathcal{T}$ is a 4-tuple (G, v_1, v_2, v), consisting of a type of gate $G \in \{G_+, G_\neg\}$, two input nodes $v_1, v_2 \in V \cup \{nil\}$ and an output node v. Each node $v \in V$ may be the output node for at most one gate, i.e., for every two gates $T = (G, v_1, v_2, v)$ and $T' = (G', v_1', v_2', v')$ in $\mathcal{T}$, $v \neq v'$.

Definition 11 (Simplified ϵ-Gcircuit problem)**.** Given a simplified generalized circuit $\mathcal{S} = (V, \mathcal{T})$ and an assignment $\boldsymbol{x} : V \longrightarrow \mathbb{R}$, we say $\boldsymbol{x}$ *ϵ-approximately satisfies* $\mathcal{S}$ if $\boldsymbol{x}(v) \in [0, 1+\epsilon]$ for any $v \in V$, and for each gate $T = (G, v_1, v_2, v) \in \mathcal{T}$, we have that $|\boldsymbol{x}(v) - f_G(\boldsymbol{x}(v_1), \boldsymbol{x}(v_2))| \leq \epsilon$, where f is defined as follows:

(1) SUM: $f_{G_+}(x,y) = \min(x+y, 1)$
(2) NOT: $f_{G_\neg}(x) = 1-x$.

In the remaining part of this subsection, we will prove the following hardness result:

Theorem 3. *There exists a constant $\epsilon > 0$ such that simplified ϵ-Gcircuit problem with fan-out 2 is* PPAD*-complete.*

More specifically, we will prove the hardness result by showing that a ϵ-Gcircuit problem can be reduced to a simplified $\epsilon_0 = \epsilon/11.5$-Gcircuit problem. Next, we present how to approximately construct the other 6 gates in generalized circuits by NOT and SUM gates that ϵ_0-approximately satisfy SUM and NOT. For convenience, we define a NOT-SUM gadget that applies SUM on the two input variables and then applies NOT on the resulting variable. It is clear that we can approximate such gadgets with errors less than $2\epsilon_0$.

DIFF gates. We implement

$$v_1 - v_2 = \neg\Big((\neg v_1) + v_2\Big).$$

In more detail, notice that with the NOT gate, we can construct a variable $\overline{v_1}$ with value $\boldsymbol{x}(\overline{v_1}) \in [1 - \boldsymbol{x}(v_1) - \epsilon_0, 1 - \boldsymbol{x}(v_1) + \epsilon_0]$. By applying NOT-SUM on $\overline{v_1}$ and v_2, we can construct a variable v such that

$$\begin{aligned} \boldsymbol{x}(v) &\in [1 - \boldsymbol{x}(\overline{v_1}) - \boldsymbol{x}(v_2) - 2\epsilon_0, 1 - \boldsymbol{x}(\overline{v_1}) - \boldsymbol{x}(v_2) + 2\epsilon_0] \\ &\subseteq [\boldsymbol{x}(v_1) - \boldsymbol{x}(v_2) - 3\epsilon_0, \boldsymbol{x}(v_1) - \boldsymbol{x}(v_2) + 3\epsilon_0], \end{aligned}$$

which approximates the output with an error less than $3\epsilon_0$.

HALF and VALUE gates. Taking advantage of the fact that we have a *generalized* circuit, we observe that $v = v_1/2$ is the unique solution

$$v = v_1 - v.$$

In more detail, notice that with the previously constructed DIFF gates, we can construct a variable v such that

$$\boldsymbol{x}(v) \in \boldsymbol{x}(v_1) - \boldsymbol{x}(v) \pm 3\epsilon_0 \Rightarrow \boldsymbol{x}(v) \in \frac{1}{2}\boldsymbol{x}(v_1) \pm 1.5\epsilon_0 ,$$

which approximates the output of the HALF gate with an error less than $1.5\epsilon_0$.

Similarly, we observe that $v = \frac{1}{2}$ is the unique solution of

$$v = 1 - v .$$

In more detail, notice that with NOT gates, we can construct a variable v such that $\boldsymbol{x}(v) \in 1-\boldsymbol{x}(v)\pm\epsilon_0$ and approximates the output of the VALUE gate with error less than $0.5\epsilon_0$.

ROUND gadgets. We shall introduce the ROUND gadget, which will be used for the construction of the remaining AND, OR, LESS gates. The ROUND gadget has one input variable and one output variable, and approximates the following function:

$$f_{G_{[]}}(x) = \begin{cases} 0 & x \leq 0.25 , \\ 1 & x \geq 0.75 . \end{cases}$$

With a VALUE and a HALF gate, there can be a variable $v_{1/4}$ with value $0.25 \pm 2\epsilon_0$. By applying a DIFF gate on $v_1, v_{1/4}$, we can get a variable v_1^- such that

$$\boldsymbol{x}(v_1^-) \in \begin{cases} 0 \pm 3.5\epsilon_0 & \boldsymbol{x}(v_1) \leq 0.25 \\ 0.5 \pm 3.5\epsilon_0 & \boldsymbol{x}(v_1) \geq 0.75 \end{cases}$$

Therefore, using a SUM gate with both inputs v_1^-, we can approximate the ROUND gadget with error less than $8\epsilon_0$.

AND and OR gates. We shall first show the construction of NOR gadgets, which approximates the following function

$$f_{G_\downarrow}(x, y) = \begin{cases} 0 & (x > 0.9) \vee (y > 0.9) \\ 1 & (x < 0.1) \wedge (y < 0.1) \end{cases}$$

We can construct them with errors less than $8\epsilon_0$ by applying a NOT-SUM gadget on v_1, v_2 and then applying a ROUND gadget on the resulting variable. To construct OR gates, we can first apply a NOR gadget and then apply a NOT gate. Then, the resulting variable v satisfies

$$\boldsymbol{x}(v) \in \begin{cases} 1 \pm 9\epsilon_0 & (\boldsymbol{x}(v_1) > 0.9) \vee (\boldsymbol{x}(v_2) > 0.9) \\ 0 \pm 9\epsilon_0 & (\boldsymbol{x}(v_1) < 0.1) \wedge (\boldsymbol{x}(v_2) < 0.1) \end{cases}$$

which approximates OR gates with errors less than $9\epsilon_0$. To construct AND gates, we can first apply two NOT gates respectively on v_1, v_2 and then apply the NOR gadget. Then, the resulting variable v satisfies

$$\boldsymbol{x}(v) \in \begin{cases} 1 \pm 8\epsilon_0 & (\boldsymbol{x}(v_1) > 0.9 + \epsilon_0) \wedge (\boldsymbol{x}(v_2) > 0.9 + \epsilon_0) \\ 0 \pm 8\epsilon_0 & (\boldsymbol{x}(v_1) < 0.1 - \epsilon_0) \vee (\boldsymbol{x}(v_2) < 0.1 - \epsilon_0) \end{cases}$$

which approximates AND gates with errors less than $8\epsilon_0$.

LESS gates. With a DIFF gate on v_1 and v_2, we can get a variable $\tilde{v}$ such that

$$\boldsymbol{x}(\tilde{v}) \in \begin{cases} (10\epsilon_0, +\infty) & \boldsymbol{x}(v_1) > \boldsymbol{x}(v_2) + 11.5\epsilon_0 \\ (-\infty, 1.5\epsilon_0) & \boldsymbol{x}(v_1) < \boldsymbol{x}(v_2) - 11.5\epsilon_0 \end{cases}$$

Next, we consider a DOUBLE gadget, which is implemented by a SUM gate with both input variables. Because the error of SUM is ϵ_0, the error of DOUBLE is also ϵ_0. Then, we can derive the following lemma.

Lemma 4. *After applying DOUBLE gadgets on a variable v for k times, the resulting variable v' satisfies:*

$$\boldsymbol{x}(v') \in [2^k \boldsymbol{x}(v) - (2^k - 1)\epsilon_0, 2^k \boldsymbol{x}(v) + (2^k - 1)\epsilon_0] \ .$$

Proof. We shall prove it by induction. The base case is trivial when $k = 0$. Suppose the statement holds for k. After applying the $(k+1)$th DOUBLE gadget, the resulting value lies in the interval:

$$\begin{aligned} &[2(2^k \boldsymbol{x}(v) - (2^k - 1)\epsilon_0) - \epsilon_0, 2(2^k \boldsymbol{x}(v) + (2^k - 1)\epsilon_0) + \epsilon_0] \\ &\qquad = [2^{k+1} \boldsymbol{x}(v) - (2^{k+1} - 1)\epsilon_0, 2(2^{k+1} \boldsymbol{x}(v) + (2^{k+1} - 1)\epsilon_0)] \ . \end{aligned}$$

□

Therefore, by applying DOUBLE gadget $-\log(10\epsilon_0)$ times on $\tilde{v}$, we can get a variable $\tilde{v}'$ such that:

$$\boldsymbol{x}(\tilde{v}'') \in \begin{cases} (0.9 + \epsilon_0, +\infty) & \boldsymbol{x}(v_1) > \boldsymbol{x}(v_2) + 11.5\epsilon_0 \\ (-\infty, 0.25 - \epsilon_0) & \boldsymbol{x}(v_1) < \boldsymbol{x}(v_2) - 11.5\epsilon_0 \end{cases}$$

Further, with a ROUND gadget on it, we can construct LESS gates with errors less than $8\epsilon_0$.

D.2 Reduce Simpified Gcircuit to A-CEEI

In the reduction, we shall use a two-credit auxiliary course c_0 with infinite capacity. According to the definition of A-CEEI, in any (α, β)-CEEI, its price p_0 must be zero. For each course i, let n_i be the upper bound for the number of students wanting course i in gadgets where i is not the *output course.*

We first present the construction of the following NOT-SUM gadget, which can be easily used to construct NOT and SUM gates.

Lemma 5 (NOT-SUM gadget)**.** *Let $n_j \geq n_i > 3\alpha$ and suppose that the economy contains the following courses:*

- *input course c_i, c_j;*
- *output course c_k with capacity $q_k = 2n_i/3$ (at most $n_k = n_i/3$ other students want c_k);*
- *interior course c_0;*

and n_i following students:

- *1 conflict: course c_k and c_0 can not be simultaneously allocated to a student, i.e. $valid(\boldsymbol{x}) = 1$ if and only if $x_0 + x_k \leq 1$;*
- *1 requirement: the total allocated credits among courses c_i, c_j, c_0 is at least 2, i.e. $req(\boldsymbol{x}) = 1$ if and only if $x_i + x_j + 2x_0 \geq 2$;*
- *the utility function is:*

$$u(\boldsymbol{x}) = \begin{cases} 8x_i + 4x_j + 2x_k + x_0 + 16 & valid(\boldsymbol{x}) = 1, req(\boldsymbol{x}) = 1 \ , \\ 8x_i + 4x_j + 2x_k + x_0 & valid(\boldsymbol{x}) = 1, req(\boldsymbol{x}) = 0 \ , \\ -\infty & valid(\boldsymbol{x}) = 0 \ . \end{cases}$$

Then in any (α, β)-CEEI,

$$p_k^* \in [1 - p_i^* - p_j^*, 1 - p_i^* - p_j^* + \beta] \ .$$

Proof. It is easy to observe that the students' preferences are as follows:

$$\{c_i, c_j, c_k\} \succcurlyeq \{c_i, c_j, c_0\} \succcurlyeq \{c_i, c_j\} \succcurlyeq \{c_i, c_0\} \succcurlyeq \{c_j, c_0\} \succcurlyeq \{c_0\}$$
$$\succcurlyeq \{c_i, c_k\} \succcurlyeq \{c_i\} \succcurlyeq \{c_j, c_k\} \succcurlyeq \{c_j\} \succcurlyeq \{c_k\} \succcurlyeq \emptyset$$

Because course c_0 is always affordable, the students must get a bundle that is worse than $\{c_0\}$. Next, we discuss all possibilities of prices p_i^*, p_j^*, p_k^* of c_i, c_j, c_k in any (α, β)-CEEI.

- If $p_i^* + p_j^* + p_k^* < 1$, the students will be all allocated the bundle $\{c_i, c_j, c_k\}$. Therefore, there are at least n_i students allocated c_k, and thus the excess demand of c_k is at least $n_i/3 > \alpha$, which is impossible in any (α, β)-CEEI.
- If $p_i^* + p_j^* + p_k^* > 1 + \beta$, because the bundle $\{c_0\}$ is always affordable and better than any bundle involving c_k, no students will be allocated c_k. Therefore, there are at most $n_i/3$ students allocated c_k, and thus the excess demand of c_k is at most $-n_i/3 < -\alpha$, which is only possible in any (α, β)-CEEI when $p_k^* = 0$.

Hence, in any (α, β)-CEEI,

$$p_k^* \in [1 - p_i^* - p_j^*, 1 - p_i^* - p_j^* + \beta] \ .$$

□

Note that this gadget cannot work when the output course is identical to any of the input courses. To fix this issue, for any (possibly identical) input variables i, j and output variable k, we can use two additional auxiliary courses c_x, c_y and three above gadgets as follows:

- input c_i, c_j and output c_x, after which $p_x^* \in [1 - p_i^* - p_j^*, 1 - p_i^* - p_j^* + \beta], n_x = n_i/3$;
- input c_x, c_0 and output c_y, after which $p_y^* \in [1 - p_x^*, 1 - p_x^* + \beta] \subseteq [p_i^* + p_j^* - \beta, p_i^* + p_j^* + \beta], n_y = n_x/3 = n_i/9$;
- input c_y, c_0 and output c_k, after which $p_k^* \in [1 - p_y^*, 1 - p_y^* + \beta] \subseteq [1 - p_i^* - p_j^* - \beta, 1 - p_i^* - p_j^* + 2\beta], n_k = n_y/3 = n_i/27$.

However, because $n_k < n_i$ in the above construction, the gadgets cannot be directly integrated into cyclic circuits. To integrate the gadgets, we can use the following *course-size amplification gadget*, which can approximately preserve the price of the input course and allow twice as many additional students to want the output course.

Lemma 6 (Course-size amplification gadget)**.** *Let $n_i > 2\alpha$ and suppose that the economy contains the following courses:*

- *input course c_i;*
- *output course c_j with capacity $q_j = 2.5n_i$ (at most $n_j = 2n_i$ other students want c_j);*
- *interior course c_0, and course c_1 with capacity $q_1 = 3.5n_i$;*

and the following sets of students:

- *n_i students with:*
  - *1 conflict: course c_0 and c_1 can not be simultaneously allocated to a student, i.e. $valid(\boldsymbol{x}) = 1$ if and only if $x_0 + x_1 \leq 1$;*
  - *1 requirement: the total number of allocated courses among c_i, c_0 is at least 1, i.e. $req(\boldsymbol{x}) = 1$ if and only if $x_i + x_0 \geq 1$;*

  – *the utility function is:*

$$u(\boldsymbol{x}) = \begin{cases} 4x_i + 2x_1 + x_0 + 8 & valid(\boldsymbol{x}) = 1, req(\boldsymbol{x}) = 1 \,, \\ 4x_i + 2x_1 + x_0 & valid(\boldsymbol{x}) = 1, req(\boldsymbol{x}) = 0 \,, \\ -\infty & valid(\boldsymbol{x}) = 0 \,. \end{cases}$$

- $n_1 = 3n_i$ *students with:*
  – *1 conflict: course* c_0 *and* c_j *can not be simultaneously allocated to a student, i.e.* $valid(\boldsymbol{x}) = 1$ *if and only if* $x_0 + x_j \leq 1$;
  – *1 requirement: the total number of allocated courses among* c_0, c_1 *is at least 1, i.e.* $req(\boldsymbol{x}) = 1$ *if and only if* $x_0 + x_1 \geq 1$;
  – *the utility function is:*

$$u(\boldsymbol{x}) = \begin{cases} 4x_1 + 2x_j + x_0 + 8 & valid(\boldsymbol{x}) = 1, req(\boldsymbol{x}) = 1 \,, \\ 4x_1 + 2x_j + x_0 & valid(\boldsymbol{x}) = 1, req(\boldsymbol{x}) = 0 \,, \\ -\infty & valid(\boldsymbol{x}) = 0 \,. \end{cases}$$

Then in any (α, β)*-CEEI,*

$$p_j^* \in [p_i^* - \beta, p_i^* + \beta] \,.$$

Proof. It is easy to observe that the preferences of the first n_i students and the last n_1 students are respectively as follows:

$$\{c_i, c_1\} \succcurlyeq \{c_i, c_0\} \succcurlyeq \{c_i\} \succcurlyeq \{c_0\} \succcurlyeq \{c_1\} \succcurlyeq \emptyset \,,$$
$$\{c_1, c_j\} \succcurlyeq \{c_1, c_0\} \succcurlyeq \{c_1\} \succcurlyeq \{c_0\} \succcurlyeq \{c_j\} \succcurlyeq \emptyset \,.$$

Because course c_0 is always affordable, the students must get a bundle that is no worse than $\{c_0\}$. Next, we start by discussing the prices p_i^*, p_1^* of c_i, c_1 in any (α, β)-CEEI.

- If $p_i^* + p_1^* < 1$, the first n_i students will be all allocated $\{c_i, c_1\}$. Further, because $p_1^* < 1$, the other $3n_i$ students will all be allocated course c_1. Therefore, there are $4n_i$ students allocated c_1, and thus the excess demand of c_1 is $n_i/2 > \alpha$, which is impossible in any (α, β)-CEEI.
- If $p_i^* + p_1 > 1 + \beta$, none of the first n_i students will be allocated course c_1. Therefore, there are at most $3n_i$ students allocated c_1, and thus the excess demand of c_1 is at most $-n_i/2 < -\alpha$, which is impossible in any (α, β)-CEEI.

Hence, $p_1^* \in [1 - p_i^*, 1 - p_i^* + \beta]$. Finally, we shall show $p_j^* \in [1 - p_1^*, 1 - p_1^* + \beta]$ to finish the proof.

- If $p_j^* + p_1^* > 1 + \beta$, none of the last n_1 students will be allocated course c_j. Therefore, there are at most n_j students allocated c_j, and thus the excess demand of c_j is at most $-n_i/2 < -\alpha$. Because $p_1^* \leq 1 + \beta$ in any (α, β)-CEEI, $p_j^* > 0$ and this is impossible in any (α, β)-CEEI.
- If $p_j^* + p_1^* < 1$, all the last n_1 students will be allocated the bundle $\{c_1, c_j\}$. Therefore, there are at least $3n_i$ students allocated c_j, and thus the excess demand of c_j is at least $n_i/2 > \alpha$, which is impossible in any (α, β)-CEEI. □

E SUPPLEMENTARY FIGURES

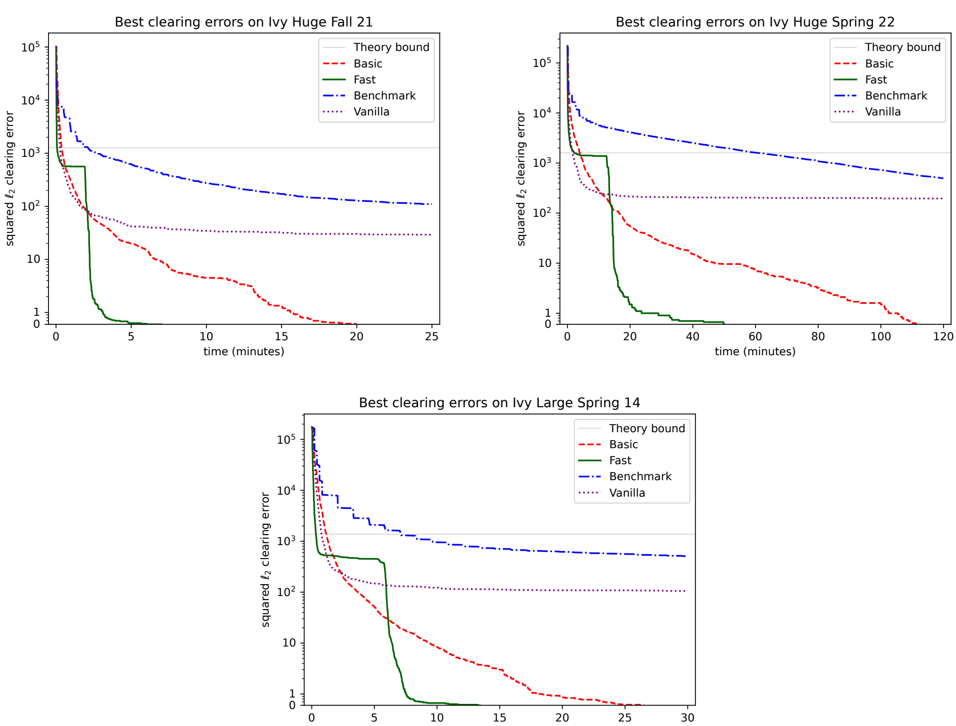

Fig. 3. For each of the algorithms tested and time t, we plot the best clearing error obtained by the algorithm up to time t.

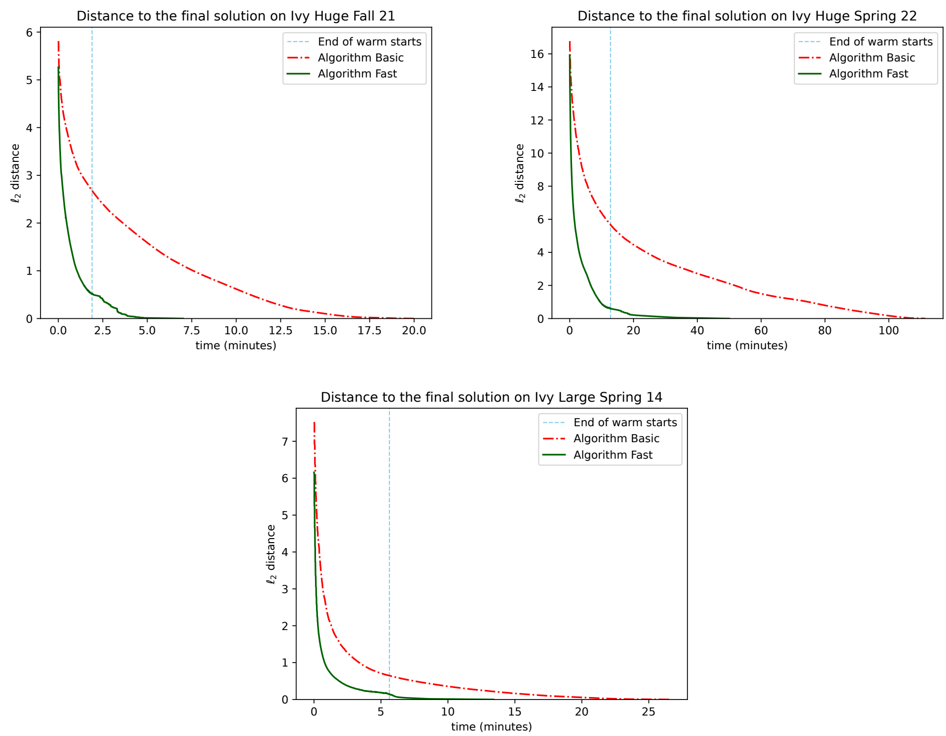

Fig. 4. The distance between the found solution and the final solution with respect to the running time.