Finding Approximate Competitive Equilibria: Efficient and Fair Course Allocation

Abraham Othman, Tuomas Sandholm
Computer Science Department
Carnegie Mellon University
{aothman,sandholm}@cs.cmu.edu

Eric Budish
University of Chicago
Booth School of Business
eric.budish@chicagobooth.edu

ABSTRACT

In the course allocation problem, a university administrator seeks to efficiently and fairly allocate schedules of over-demanded courses to students with heterogeneous preferences. We investigate how to computationally implement a recently-proposed theoretical solution to this problem (Budish, 2009) which uses approximate competitive equilibria to balance notions of efficiency, fairness, and incentives. Despite the apparent similarity to the well-known combinatorial auction problem we show that no polynomial-size mixed-integer program (MIP) can solve our problem. Instead, we develop a two-level search process: at the master level, the center uses tabu search over the union of two distinct neighborhoods to suggest prices; at the agent level, we use MIPs to solve for student demands in parallel at the current prices. Our method scales near-optimally in the number of processors used and is able to solve realistic-size problems fast enough to be usable in practice.

Categories and Subject Descriptors

J.4 [**Social and Behavioral Sciences**]: Economics

General Terms

Economics, Algorithms, Experimentation

Keywords

Mechanism Design, Combinatorial Allocation, Winner Determination, Local Search

1. INTRODUCTION

At many educational institutions, students' learning experience requires limiting the number of students in a class. This creates a challenging allocation problem for school administrators: how should scarce course seats be allocated amongst students with heterogeneous preferences?

Course allocation is a well known open problem in market design (c.f. Roth [2002], Milgrom [2007]). Most of what is known theoretically about this problem are impossibility theorems which indicate that there is no perfect solution: there is no mechanism that is efficient, strategyproof, and fair [Pápai, 2001, Ehlers and Klaus, 2003]. Practitioners have not solved the problem either: the most widely-used course-allocation mechanism in practice is inefficient, easy to manipulate, and risks outcomes that are highly unfair (some students may obtain zero courses) [Sönmez and Unver, Forthcoming].

In recent work, Budish [2009] proposes a new theoretical solution to the course-allocation problem. His mechanism—which adapts the old idea of Competitive Equilibrium from Equal Incomes [Foley, 1967, Varian, 1974, Thomson and Varian, 1985] to the case of indivisible goods—satisfies tight approximations of the efficiency, incentive compatibility and fairness properties a mechanism ideally should satisfy. However, Budish's theoretical solution is not yet a practical solution for two reasons. First, the mechanism is non-constructive: it relies on finding a Kakutani fixed point of a highly discontinuous function. Second, the mechanism requires that students report their ordinal preferences over all possible schedules of courses: this is impractical because the number of schedules can be exponentially large.

This paper proposes a computational method for implementing Budish's Approximate CEEI Mechanism (A-CEEI). First, students report their ordinal preferences over schedules using a reporting language that is concise but expressive. Second, we use a decentralized, two-level search process to solve for A-CEEI prices and allocations. At one level of the search, the master level, the center searches for prices for the courses, denominated in an artificial currency. At the other level, the agent level, each agent searches for their most preferred affordable bundle given the prices. We use tabu search over price space; a unique feature of our search is a hybrid neighborhood. We use a mixed-integer program (MIP) solver for the agents' individual demand problems. Traditional economic theory tells us that one of the advantages of prices is that they allow disjoint agents to make decisions disjointly [Hayek, 1945]; we exploit Hayek's insight by fully parallelizing the solving of agent demand problems. Since nearly all of our computation time is spent on the agent level, we achieve excellent scaling on multicore systems. Our method is able to solve realistic-size problems fast enough to be usable for practice.

The course-allocation problem belongs to a broader class of problems called *combinatorial assignment*, in which a set of indivisible integer-supply objects is to be allocated amongst a set of agents with preferences over bundles, without the use of monetary transfers[1]. (A-CEEI uses artificial currency with no value outside the problem at hand). This is similar to the well-known *combinatorial auction* problem: the only difference is the prohibition against monetary transfers. This seemingly small difference has important

[1]Other potential examples of combinatorial assignment problems include the assignment of fixed-wage shifts (or tasks) to interchangeable workers, leads to salespeople, players to amateur sports teams, airport takeoff-and-landing slots to airlines, and shared scientific resources to scientists.

Cite as: Finding Approximate Competitive Equilibria: Efficient and Fair Course Allocation, Abraham Othman, Eric Budish, and Tuomas Sandholm, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 873-880

implications for what allocation rules will have attractive efficiency, fairness and incentive properties, and ultimately for what computational procedures will be attractive. Whereas MIPs are highly effective at solving combinatorial auction problems, our analysis suggests that MIPs alone cannot be used to solve the combinatorial assignment problem. Specifically, we prove that no polynomial-size MIP implements A-CEEI, even for the simplest class of student preferences over courses (additive separable).

We also briefly review why the Eisenberg-Gale paradigm cannot be used in our context, and discuss the subtle but critical differences between A-CEEI and the course-bidding procedures currently in widespread use at professional schools around the world.

The remainder of this paper is organized as follows. Section 2 describes the theoretical mechanism. Section 3 shows that MIPs alone cannot be used to implement A-CEEI. Section 4 describes our two-level search process. Section 5 reports computational experiments that assess our method's speed. Section 6 concludes.

2. A-CEEI: THEORY

The ingredients of a course-allocation problem are as follows. There is a set of M courses with integer capacities (supply) $(q_j)_{j=1}^M$. There is a set of N students. Each student i has a set $\Psi_i \subseteq \{0,1\}^M$ of permissible schedules. A schedule consists of 0 or 1 seats of each course, and at most $k \leq M$ courses altogether. The set Ψ_i encodes any universal scheduling constraints (e.g. courses that meet at the same time) and any constraints specific to i (e.g. prerequisites). Each student i has a utility function over her permissible schedules, $u_i : \Psi_i \rightarrow \mathbb{R}_+$. We assume that preferences are strict, i.e., for $x_i \neq x_i' \in \Psi_i, u_i(x_i) \neq u_i(x_i')$. We do not place any further restrictions on preferences. In particular, students are free to regard courses as substitutes or complements.

Budish [2009] proposes the following course-allocation mechanism.

MECHANISM 1. ***A-CEEI**:*

1. *Students report their preferences. Denote the reports $(\hat{u}_i)_{i=1}^N$*
2. *Assign each student i a budget b_i^* that is a uniform random draw from $[1, 1+\beta]$, with $0 < \beta \ll \min(\frac{1}{N}, \frac{1}{k-1})$.*
3. *Compute a set of prices $(p_j^*)_{j=1}^M$ and allocations $(x_i^*)_{i=1}^N$ such that*
   (a) *Students maximize utility, based on their reported preferences, subject to their budget constraints. Formally,*

$$\forall i : x_i^* = \arg\max_{x' \in \Psi_i} \left[\hat{u}_i(x_i) : \sum_j x_{ij} p_j^* \leq b_i^* \right]$$

   (b) *The magnitude of market-clearing error is as small as possible. Formally, minimize*

$$\alpha \equiv \sqrt{\sum_j \xi_j^2}$$

   Where ξ_j is given by

$$\xi_j = \begin{cases} \sum_i x_{ij}^* - q_j & \text{if } p_j^* > 0; \\ \max\left[\left(\sum_i x_{ij}^* - q_j\right), 0\right] & \text{if } p_j^* = 0. \end{cases}$$

If $\alpha = \beta = 0$ this mechanism would be equivalent to the Competitive Equilibrium from Equal Incomes proposed by Foley [1967], Varian [1974], and numerous others since. The only difference is that we use a Fisher economy rather than an endowment economy (c.f., [Vazirani, 2007]) because equal endowments are not well defined with indivisibilities. Unfortunately exact CEEI need not exist. Budish [2009, Theorem 1] shows that, as long as budget-inequality β is strictly positive, there exist prices that clear the market to within error of at most $\alpha = \sqrt{kM/2}$. This worst-case bound is small for realistic problems, and, as $N, q \rightarrow \infty$, market-clearing error converges to zero as a fraction of the endowment.

But for the small amount of market-clearing error, the allocation is Pareto efficient. Budish [2009] also shows that the mechanism satisfies attractive criteria of fairness and satisfies an intuitive relaxation of strategyproofness; the reader is encouraged to consult that paper for additional details.

2.1 An Example

We present a simple example to illustrate A-CEEI and its efficiency and fairness properties.

EXAMPLE 1. *There are two students $\{1,2\}$ and four courses $\{a,b,c,d\}$, each in unit supply. Students consume at most two courses each; there are no other scheduling constraints. Students' utility functions are additive separable over objects. Suppose they report the following preferences (normalized to sum to 100):*

	a	b	c	d
1	*60*	*30*	*6*	*4*
2	*62*	*32*	*4*	*2*

We can think of a and b as courses taught by star professors and c and d as unpopular courses. Suppose A-CEEI draws random budgets of $b_1^* = 1.1$ and $b_2^* = 1.0$. Then at prices of $p_1^* = 1.1, p_2^* = 0.9, p_3^* = 0.1,$ and $p_4^* = 0.0$, student 1's utility maximizing affordable bundle is $\{a,d\}$, while 2's is $\{b,c\}$. This allocation exactly clears the market.

Observe that the allocation in Example 1 is Pareto efficient; further, it is intuitively as fair as possible given the constraints of the problem. (Fairness criteria are formalized in Budish [2009]).

3. WHY TRADITIONAL SOLUTIONS FAIL

Consider the well-known winner determination problem from combinatorial auctions [Lehmann et al., 2006], applied to our setting. We use x_i to denote the bundle assigned to student i and $x_{ij} \in \{0,1\}$ as an indicator of whether bundle x_i contains course j:

$$\max_x \sum_i u_i(x_i) \text{ subject to} \tag{1}$$

$$\sum_j x_{ij} \leq q_j, \forall j$$

$$x_i \in \Psi_i, \forall i$$

Program (1) is well defined in our context, and, though NP-hard, will often be easy to solve in practice [Hoos and Boutilier, 2001, Sandholm, 2007]. But for course allocation this is the wrong program to solve! First, it will lead to highly unfair allocations. For instance, in Example 1 it will allocate both of the good courses to student 2 and both of the bad courses to student 1. Second, it will create incentives to misreport. If student 1 misreports her utility vector as $(63, 33, 3, 1)$ then *she* will get both of the good courses.

The basic reason why Program (1) works well in combinatorial auction contexts but not in course allocation is that, in a combinatorial auction, the solution to (1) can be supported by transfers of real money. The prices that define these transfers can be calculated from the dual to (1). Given these supporting prices, each

bidder is assigned the bundle that maximizes their utility *less their expenditure*. Consider Example 1, supposing that the object values represent students' willingness-to-pay in dollars. Then the outcome in which student 2 gets both a and b but *has to pay for them with real money* (e.g., pays \$60 for a and \$30 for b) will be envy free. Student 1 does not envy the right to pay \$90 for $\{a, b\}$. (See Alkan et al. [1991]).

In our problem, money is artificial and has no outside use. As a result, prices play a conceptually different role: prices define students' *choice sets*. At price vector $\mathbf{p}^*$, each student is assigned the bundle that maximizes their utility *out of all schedules in their choice set*. That is, student i with budget b_i is assigned

$$\arg\max_{x \in \Psi_i} [u_i(x) : \mathbf{p}^* \cdot x \leq b_i]$$

whereas, in a combinatorial auction, student i is assigned

$$\arg\max_{x \in \Psi_i} [u_i(x) - \mathbf{p}^* \cdot x]$$

The reason Mechanism 1 has attractive fairness properties is that students' choice sets are roughly similar in quality (because budget inequality is small). The reason Mechanism 1 has attractive incentive properties is that each student is allocated her most-preferred bundle in the choice set defined by the prices and her budget.

A natural idea is to seek a way to augment Program (1) so as to implement Mechanism 1. Suppose, to conserve notation, that there exists a solution with zero market clearing error. Consider the following modified program:

$$\max_{x,p} \sum_i u_i(x_i) \text{ subject to}$$

$$\sum_j x_{ij} \leq q_j, \forall j$$

$$x_i = \arg\max_{x' \in \Psi_i} [u_i(x) : \mathbf{p} \cdot x \leq b_i], \forall i \qquad (2)$$

In order to use a MIP, one would have to find a way to re-express the optimization constraints (2) as linear inequalities. There are various ways to do this, but none that is concise. We formalize this claim in Theorem 1 below. To state Theorem 1, we need the following two definitions.

DEFINITION 1. *An agent has* additive separable utility *if their utility for a bundle is the sum of their utilities for the individual courses within that bundle. Formally, for each student i, there exists a vector of course values $v_i = (v_{i1}, \ldots, v_{iM})$ such that $u_i(x_i) = \sum_j v_{ij} x_{ij}$ for all $x_i \in \Psi_i$.*

This is the simplest and most immediate way to think about students' values for a bundle of courses—simply add up the independent values for the courses within the bundle. This way of thinking about valuations is also used (implicitly or explicitly) by many current course-selection mechanisms, which assume additive separability when they allot each student their most preferred available course in turn.

DEFINITION 2. *In the KNAPSACK OPTIMIZATION problem, we are given a budget and a set of items. Each item has a value and a price. We are asked which set of items maximizes the sum of values without exceeding the budget. KNAPSACK OPTIMIZATION is NP-hard [Cormen et al., 2001].*

THEOREM 1. *If an agent can express additively separable utility, their maximization objective cannot be represented in a polynomial number of constraints (unless P=NP).*

PROOF. Solving the maximization objective for an agent with additively separable utility is equivalent to solving KNAPSACK OPTIMIZATION. Now imagine we could express an agent's maximization objective in a polynomial number of constraints. Then we would have a polynomial-time verifier for KNAPSACK OPTIMIZATION: simply test a prospective optimization solution on each constraint in turn. Having a polynomial number of constraints would therefore imply that the verification version KNAPSACK OPTIMIZATION (i.e., "Is this the optimal solution?") is in NP, because it could be verified in polynomial time.

Now imagine that we could verify any ϵ-optimal solution with our (unconditional in ϵ) polynomial verifier. Put another way, our verifier could distinguish between an ϵ-optimal solution and an optimal solution for all ϵ, with no size dependence on ϵ. Since if P$\neq$ NP, any polynomial verifier for KNAPSACK OPTIMIZATION must have a dependence on $1/\epsilon$, by shrinking ϵ small enough we will either break our verifier or have P=NP. ■

3.1 Eisenberg-Gale and Bidding Points Mechanisms

In this section we briefly discuss two other natural ideas for how to solve the course-allocation problem. The first is to use a convex program in the Eisenberg-Gale paradigm [Eisenberg and Gale, 1959]. As Vazirani [2007] and others have shown, this approach can often yield competitive equilibrium prices in polynomial time for problems in which (i) agents' utilities are linear in their consumptions; and (ii) agents face budget constraints. This sounds promising for implementing A-CEEI for the case of additive separable preferences.

The basic difficulty is that the Eisenberg-Gale paradigm assumes that goods are perfectly divisible. In many indivisible goods allocation problems, it actually works quite well to first solve for an optimal allocation *as if the goods were perfectly divisible*, and then "round" the resulting fractional allocation to a "nearby" integer allocation. This approach is taken e.g., in Jain and Vazirani [2007], and is discussed more generally in Budish et al. [2009]. However this approach is dangerous in contexts like course allocation where we care about ex-post fairness. It is possible, for instance, that some student will use his entire budget to obtain a high-probability chance of taking some star professor's class, only then not to get it (nor anything else he likes) [see Budish, 2009, Section 7.3].

A second natural idea is to allocate students equal budgets of artificial currency, and then have them bid their points for different courses. For each course j, the q_j highest bidders get a seat. Variants on this idea are widely used to allocate students to courses around the world, and especially at professional schools in the US. However, this mechanism is making the same mistake as Program (1): it implicitly treats fake money as real money that enters the utility function. If, in the environment of Example 1, students submit bids in *real money* according to the values table (e.g., student 1 bids 60 *dollars* for a), then student 1 will get stuck with a bad course bundle $\{c, d\}$ but at least he can spend his money on some outside good. If, as is the case in practice, students are bidding *fake money*, then student 1 gets stuck with $\{c, d\}$ and is implicitly expected to take consolation in a large bank account of unspent fake money [see Budish, 2009, Section 7.3].

4. SOLVING A-CEEI

There are two basic computational challenges for implementing A-CEEI. First, calculating a student's demand at a particular price

vector is NP-hard. Second, even if we had an oracle for students' demands, we still need to find an approximate zero of a highly discontinuous function—the market-clearing error α of Mechanism 1, Step 3(b)—in a large-dimensional price space.

There is also an important practical issue to keep in mind: A-CEEI requires as input students' ordinal preferences over schedules of courses, but the number of possible schedules can be exponentially large.

This section proposes a computational method for implementing A-CEEI. We use a decentralized, two-level search process to solve for A-CEEI prices and allocations. At the master level, the center searches through price space to determine what prices to next propose to the agents. At the agent level, each agent searches through bundle space to find their most-preferred affordable bundle at the current prices. We use a novel implementation of tabu search on the master level, and MIPs on the agent level.

We also develop a novel language with which students report their demands. The language is motivated by institutional features of the course-allocation problem, and is tailored to our use of MIPs on the agent level. Section 4.1 describes the agent level of our search, including both the reporting language and its MIP representation. Sections 4.2 and 4.3 describe the master level of our search.

The following subsections describe in more detail the search at the agent level and at the master level (price-setting level), respectively.

4.1 The Agent Level

4.1.1 Preference Language

We propose a language in which each agent's valuation for a bundle is defined by three components:

- An additive component. For each course j that agent i receives, they receive utility of v_{ij}.
- Substitute/complement effect variables defined on sets of courses. If agent i receives all courses in set s she receives additional utility η_{is}. A positive value indicates complementarity and a negative value substitutability.
- A list, c, of constraints of the form, "No more than x of the following y courses", where any bundle violating the constraint is never selected.

Each student reports v and η information to the center. For any subset of courses s for which student i does not specifically report η_{is} the center interprets that $\eta_{is} = 0$. The c constraints are partly set by the market administrator (e.g., "no more than 1 course that meets Thursday at 8:30am") and are partly reported by the student (e.g., "no more than 2 courses with a lot of programming"). The basic challenge in any preference-reporting language is to balance expressiveness with conciseness. A student with particularly simple preferences—additive separable—need only report M individual course values. In most course-allocation mechanisms currently used in practice, students are restricted to reporting only this kind of information. Our language allows students with non-additivities in their preferences to report these; in principle a student can report arbitrarily complex preferences over schedules, via the η terms. Hence our language is expressive in the sense of Nisan [2006]. If the non-additivities in a student's preferences are due only to substitutabilities and complementarities between pairs of courses, then she can fully express her preferences using M course values and $\Theta(M^2)$ of the η variables. The c constraints are inspired by the recently proposed bidding language of Milgrom [2008] in the context of auctions. Milgrom [2008] restricts the constraints that agents report in his language to those that satisfy a technical condition known as hierarchy; this has both an economic and computational payoff in his context. In our context there is no similar reason to make such a restriction.

4.1.2 Agent-level Demand Computation

Given a set of prices, the optimization problem faced by an agent with this utility model is NP-hard. Let each constraint in the constraint list, $c_i = (c_{i1}, c_{i2}, \ldots)$, take the following form: c_{ik} consists of courses $x_{ic^1_{ik}}, x_{ic^2_{ik}}, \ldots$ and has capacity constraint $\bar{c}_k$. To solve this problem, we use the following MIP. The input to this program is the set of prices $\mathbf{p}$, and the output is an assignment to the set of binary decision variables x_i corresponding to the courses demanded by the agent. In our simulations, we considered only *pairwise* substitutability/complementarity effects, which are given in the program by the $\eta_{ij,j'}$.

$$\begin{aligned} \max \sum_j v_{ij} x_{ij} \quad &+ \quad \sum_{j<j'} \eta_{ij,j'} z_{ij,j'} \\ \text{subject to} \quad \sum_j p_{ij} x_{ij} \quad &\leq \quad b_i \\ z_{ij,j'} \quad &= \quad x_{ij} \wedge x_{ij'} \\ \sum_l x_{ic^l_{ik}} \quad &\leq \quad \bar{c}_{ik} \end{aligned}$$

Though the individual demand problem at any point is NP-hard, MIPs are an effective way to solve this problem in practice. Examples corresponding to an academic year—picking 10 courses out of 100—solved in milliseconds. As we will discuss, however, this program must be solved millions of times for different agents and candidate price vectors. As a result, profiling revealed that nearly all of our computation time is spent solving these MIPs.

Traditional economic theory (c.f. Hayek [1945]) tells us that one of the advantages of prices is that they allow disjoint agents to make decisions disjointly, so no individual need factor the decision of any other directly into her own decision-making process. This has an analogue in computational parallelism. Given a set of prices, each agent's demand can be computed separately, on a separate processor, at the same time. Since virtually all of our computation time is spent solving agents' demand problems, we achieved extremely good scaling from parallelizing our code in this manner.

4.2 Master Level: Tabu Search

The local search technique that we employ is tabu search, because it yielded good performance during our exploratory data analysis. On the surface, tabu search is a straightforward modification of hillclimbing. It remembers the last t nodes the search has expanded, and does not return to them. In practice, this is remarkably effective, mitigating the weaknesses of hillclimbing (getting stuck in local minima or looping endlessly) while retaining its ability to efficiently pursue good solutions [Glover, 1990]. The pseudocode for tabu search is below.

Our tabu search operates through price space, so a node represents a set of prices (and the demands they induce). We define two search nodes as *equal*, for the purposes of tabu list checking, if each induces the same aggregate demand (i.e., $\sum_i x^*_{ij}$). As the heuristic score, we use the clearing error (squared Euclidean distance between our solution and a true competitive equilibrium in demand space) in order to stay in line with Budish's theoretical result discussed earlier in this paper. We stop if we have not improved on our

$tabu \leftarrow$ A queue of fixed length t
$curnode \leftarrow$ Random start point
$bestnode \leftarrow curnode$
while $bestnode.score() >$ *some bound* **do**
  $tabu.push(curnode)$
  $n \leftarrow$ neighbors ($curnode$), by score ascending
  while $n.front() \in tabu$ **do** $n.pop()$
  $curnode \leftarrow n.front()$
  if $curnode.score() < bestnode.score()$ **then**
    $bestnode \leftarrow curnode$
end

Algorithm 1: *Tabu search algorithm.*

best-found solution in 100 steps, provided that the solution satisfies the theoretical bound.

4.3 Master Level: Tabu Neighborhood Selection

We developed a hybrid neighborhood approach that combines two different, independent ways of generating neighbors. These are discussed in the next two subsections, respectively.

4.3.1 Gradient Descent

Given our heuristic function of squared error, a simple way to generate new neighbors is to move along the gradient of error, raising prices of classes that are oversubscribed and lowering the prices of classes that are undersubscribed. Formally, let class j have current price p_j, demand $d_j \equiv \sum_i x_{ij}$, and supply q_j. The gradient is

$$\nabla_j = \begin{cases} d_j - q_j & \text{if } p_j > 0 \\ \max(d_j - q_j, 0) & \text{if } p_j = 0 \end{cases}$$

Because the error function is non-linear and moves in discrete jumps, finding the best step along the gradient is non-trivial. As an alternative to exhaustive search for the best step, we generate neighbors at different magnitudes along the gradient vector, taking care so that prices were truncated below at zero to avoid negative prices. (In our experiments, the average budget was about 100. We generated neighbors such that the largest change in price was 10, 5, 1, 0.5, and 0.1.)

Gradient descent finds its analogue in economic theory with the idea of tâtonnement. In tâtonnement, a central auctioneer adjusts prices of goods, raising those that are overdemanded and lowering those that are underdemanded. It is well-known that the tâtonnement process might not establish efficient prices, even in the presence of divisible goods and convex preferences [Scarf, 1967].

4.3.2 Individual Price Adjustments

Our second neighborhood is formed by adjusting the prices of individual courses one at a time. That is, each neighbor from this scheme has the same set of prices as its parent, with a change applied to only a single course. This is in contrast to the gradient method, where most (if not all) courses have their prices changed.

For a course that is oversubscribed, we raise its price to reduce its demand by exactly one student. For a course that is undersubscribed, we lower its price to zero.

An alternative to this that could be considered symmetric would be lowering the price of an undersubscribed course until exactly one more agent demands it. However, finding this price reduction would require significantly more computation time because the set of courses an agent demands at a price vector is dramatically smaller than the set of courses an agent *does not* demand.

The key difficulty here is determining the minimum price increase to lower demand on an oversubscribed course by exactly one student. We give a solution first for the general case, and then use this intuition to derive a tractable formulation for our specific bidding language.

General Case

Imagine the list of bundles ordered by the preference of agent i. Given current prices $\mathbf{p}$, the agent's current selection is their k-th preferred bundle, which we will denote as x_i^k. (This is the most preferred bundle that i can afford). Put another way, for every other bundle $x_i^{k'}$, $k' < k$, $\mathbf{p} \cdot x_i^{k'} > b_i$.

Considering a course κ from bundle x_i^k, we seek to find the minimum amount that price p_κ must be raised to have the agent select a bundle that does not include $x_{i\kappa}$.

This amount is not simply $b_i - \mathbf{p} \cdot x_i^k + \epsilon$, the agent's leftover budget from selecting their current choice x_i^k. As the price of course $x_{i\kappa}$ is raised, the agent could substitute another course in the bundle for a less-expensive course. We must find the lowest-cost bundle that still includes $x_{i\kappa}$ that would be selected before picking a bundle not including $x_{i\kappa}$.

Going down the list of affordable bundles from x_i^k, eventually we will hit bundle $x_i^{-\kappa}$, which does not include course $x_{i\kappa}$. Let X^κ denote the sequence of bundles from x_i^k up to (but not including) $x_i^{-\kappa}$. The minimum amount we would need to raise price p_j to get agent i to no longer demand course $x_{i\kappa}$ is given by $b_i - \min_{x \in X^\kappa} \mathbf{p} \cdot x + \epsilon$. For every course, this is calculated over each agent demanding the course, and then taking the minimum of these values.

Using Our Bidding Language

The above description relies on having an ordered list of bundles, which has exponential size. With our bidding language, we can avoid this explicit enumeration with a MIP.

As in the section above, we need to find the minimum-cost bundle that still involves course $x_{i\kappa}$ before the first bundle that does not include $x_{i\kappa}$. This can be solved using two MIPs.

The first finds agent i's most preferred bundle that does not include $x_{i\kappa}$:

$$\begin{aligned} \max \quad & \sum_j v_{ij} x_{ij} + \sum_{j<j'} \eta_{ij,j'} z_{ij,j'} \\ \text{subject to} \quad & \sum_j p_{ij} x_{ij} \leq b_i \\ & z_{ij,j'} = x_{ij} \wedge x_{ij'} \\ & \sum_l x_{ic_{ik}^l} \leq \bar{c}_{ik} \\ & x_{i\kappa} = 0 \end{aligned}$$

Let the objective value of this solution be o. It follows that any feasible solution with an objective value greater than that of o must include course $x_{i\kappa}$. We must find the minimum cost bundle among this set. We accomplish this with the following MIP:

$$\begin{aligned} \min \quad & \sum_j p_j x_{ij} \\ \text{subject to} \quad & \sum_j v_{ij} x_{ij} + \sum_{j<j'} \eta_{ij,j'} z_{ij,j'} > o \\ & z_{ij,j'} = x_{ij} \wedge x_{ij'} \\ & \sum_l x_{ic_{ik}^l} \leq \bar{c}_{ik} \\ & x_{i\kappa} = 1 \end{aligned}$$

Letting the objective value of this solution be π, the minimum

amount p_s must be raised by is $b_i - \pi + \epsilon$.

5. EXPERIMENTS

In this section, we empirically investigate the properties of our algorithm. We study in detail the convergence properties of a realistic-sized test case corresponding roughly to assigning courses for a business school semester. We then examine how our algorithm performs as we alter the pertinent properties of the example cases.

5.1 Solving the Problem

Our baseline problem is as follows. There are 250 students and 50 courses. Each student requires 5 courses, hence total demand is for 1250 course seats. Each course has a capacity of 27 seats, so that courses are slightly oversupplied on average. The problem size and level of excess capacity correspond roughly to a single-semester course-allocation problem at an average-sized professional school.

Student i's additive utility for course j, v_{ij}, is generated as:

$$v_{ij} = \bar{v}_j + \epsilon_{ij} \tag{3}$$

where $\bar{v}_j = j$, so that mean course utilities ranged between 1 and M, and each ϵ_{ij} is an independent draw from the normal distribution with mean 0 and standard deviation 10. In order to simulate non-additive complementarity/substitutability utility effects, Each student also had 10 effects η drawn for random pairs of courses, with values drawn uniform on $[-10, 10]$. Our utility model generates imperfect correlation in students' preferences for courses, which is important for realism because courses often vary in their popularity. Models like (3) are widely used in econometrics, dating from the work of McFadden [1976].

5.1.1 Computational Setup

We ran our code on a machine running Fedora Core 7, with four quad-core Intel Xeon 3.40GHz CPUs (for a total of 16 cores). The MIPs in the agent-level search were solved with CPLEX 10.0.

5.1.2 Finding a Solution

We ran 100 instances of the baseline problem. With 16 cores, each instance took about two hours to complete 100 iterations.

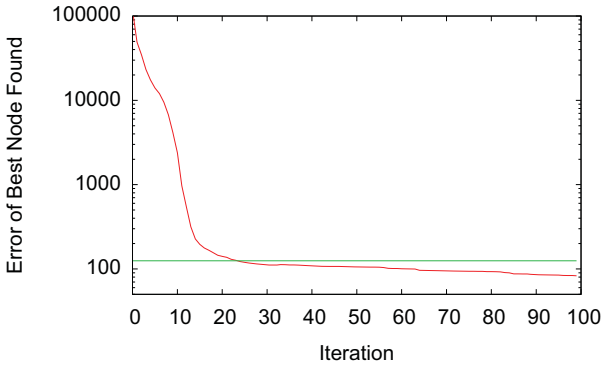

Figure 1: *The (squared) error of the best node found at each iteration, averaged over 100 trials. Budish's theoretical worst-case bound for market-clearing error is given by the flat line. The y-axis has logarithmic scale.*

Figure 1 shows the average clearing error over the 100 instances. In a problem of this size, Budish's theoretical worst-case bound for market-clearing error is $\alpha = \sqrt{kM/2} = \sqrt{125}$. The average instance was under the theoretical bound after 23 iterations, and all 100 of the instances were under the theoretical bound after 100 iterations[2].

Our algorithm's running time can be decomposed into the product of the time per iteration and the number of iterations to achieve the bound. We proceed to discuss both of these components.

5.2 Time per Iteration

We consider adding additional students, and then adding additional courses. As we change the parameters, we are careful to ensure that total course seats are oversupplied by about 10%, as in our semester-sized examples.

5.2.1 Number of Students

Adding students linearly increased the time per iteration. Figure 2 displays our results.

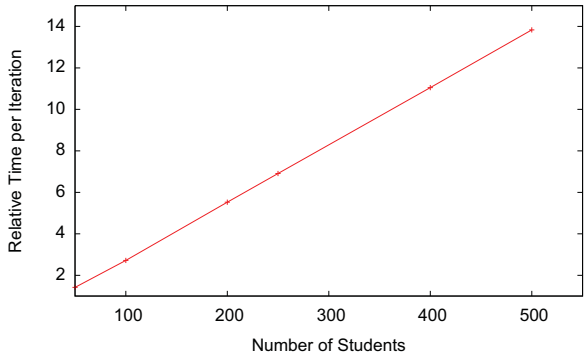

Figure 2: *Time per iteration scales linearly in the number of students.*

This result was expected. Adding an additional student means that there is another MIP to solve in the agent level of the search, but has no affect on the number of neighbors generated in an iteration of the master level of the search.

5.2.2 Number of Courses

We next consider varying the number of courses: both the overall number of courses (M) and the number of courses per student (k). One way to think about our method is that of determining a student's course allocation over a progressively larger number of academic units. Specifically, in each academic unit (quarter) of the school year, students select two courses from a possible 20, and we vary the number of quarters simultaneously considered. We tested on sizes ranging from selecting two courses out of 20, to 20 courses out of 200. We also include our baseline of five courses out of 50.

Figure 3 depicts the increase in time per iteration as M and k increase. Though the total number of bundles available to students increases exponentially in the number of courses, MIPs were still able to calculate individual demands effectively, and the time per iteration does not blow up. This is broadly consistent with findings in the context of the knapsack problem [see Kellerer et al., 2004].

[2]Budish [2009] uses the technique developed in this paper to solve problems based on the actual preferences of students at Harvard Business School. These computations on real data take less time than similarly-sized instances here. This suggests that if anything our utility model and testing setup are more challenging than what is encountered in practice.

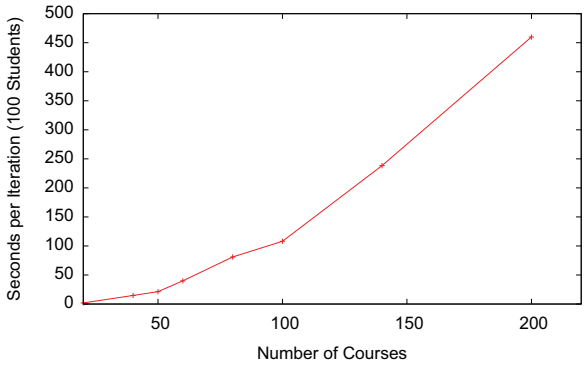

Figure 3: *Time per iteration increases in the number of courses cleared simultaneously.*

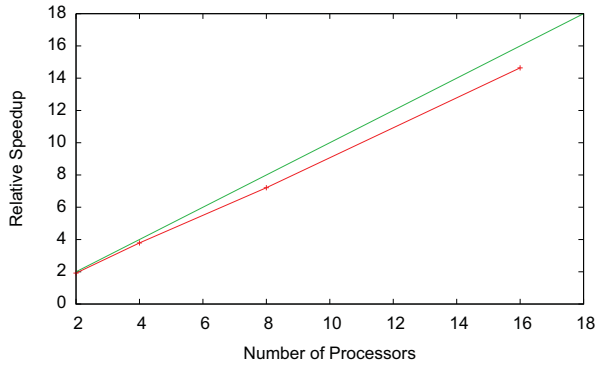

Figure 4: *Our search method parallelized with more than 90% efficiency. The upper line shows perfect scaling.*

5.3 Iterations to Clear Bound

It took between 10 and 86 iterations in our 100 baseline trials to find prices that clear the market to within Budish's theoretical bound. Because examining iterations until convergence requires running large numbers of trials for large numbers of iterations, we were limited to testing for convergence over these semester-sized examples. However, we can discuss the sensitivity to these parameters qualitatively.

Within the natural bound that overall course demand does not exceed overall course supply (i.e., that some students would be forced to take reduced course loads because of lack of space), adding students without increasing the number of course seats makes convergence harder. A smaller difference between course supply and course demand means that fewer courses will be undersubscribed, and so fewer courses will have prices of zero. In most cases, setting an unpopular course's price to zero allows that course to be ignored for the rest of the search process, and so effectively reduces the dimension of the problem.

Adding additional courses to clear has a mixed impact. If courses are added but the number of students is not increased, then more courses will have a price of zero. Moreover, adding more courses makes the theoretical bound looser. On the other hand, adding courses increases the dimension of the search space. Preliminary testing suggests that our method scales effectively to larger examples. In particular, our method works in an example that is the size of a full-year's course-allocation problem at Harvard Business School , with ~1000 students selecting 10 courses out of ~100. (HBS is one of the world's largest professional schools). We find prices that clear the market to within the theoretical bound in around 30 iterations, taking around 11 hours.

5.4 Useful Tricks

Our method relies on two useful tricks: Parallelizing demand computation through prices at the agent level, and combining two different neighborhoods at the master level. As we discuss, both of these features helped to make A-CEEI tractable.

5.4.1 Parallelization

One of our hypotheses was that the prices could be used as an effective tool to parallelize the computation of individual demands. By varying the number of simultaneous threads generated, we found that our method parallelized with about 90% efficiency, so that using n cores was about $.9n$ times faster than using a single core. Figure 4 shows our results, comparing them with the optimal speedup.

5.4.2 Hybrid Neighborhoods

Our method uses two different neighborhoods in local search, one which moves prices along the gradient, and another which adjusts prices of individual courses.

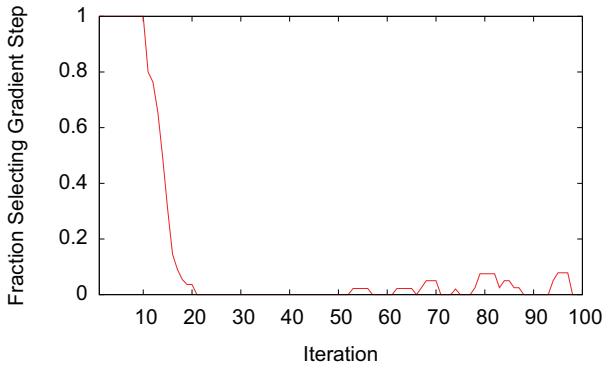

Figure 5: *In our hybrid neighborhood scheme, nodes are selected either from moving along the gradient or by adjusting the price of a single course. The fraction of nodes selected along the gradient in our trials is given on the y-axis.*

Figure 5 shows which neighborhood method provided the node that was selected in each iteration in the semester-sized data we experimented with. There is a transition at about twenty steps into the algorithm where it selects gradient steps before and individual price adjustments after. Beyond 50 iterations, the steps chosen were mostly, but not exclusively, individual price adjustments. One possible interpretation is that traditional tâtonnement works well when we are far from a solution (the error is large), but when we are close to a solution it is very difficult to predict what kinds of price adjustments will reduce error.

Neighborhood	Additively Separable	Arbitrary
Gradient only	45	0
Individual only	0	66
Hybrid	**99**	**98**

Table 1: *Number of instances (of 100) that found nodes with clearing error under the theoretical bound in 100 iterations for two different demand structures.*

Another advantage of hybrid neighborhoods comes when using different utility models. As an extreme example, imagine that students' ordinal preferences over bundles are entirely unstructured: i's preference order is just some random permutation of i's permissible bundles. In such a model, gradient steps are much less informative, because once an agent can no longer afford their current bundle their new demand will bear no relation to their previous selection. Table 1 shows the strength of the hybrid neighborhood method. By combining the gradient neighborhood with the individual price adjustment neighborhood, virtually every instance in both a standard additively-separable utility model as well as an arbitrary utility model cleared to within the theoretical error bound in 100 iterations. Interestingly, the hybrid neighborhood resulted in more test instances being under the theoretical bound in 100 iterations than either neighborhood on its own. This result suggests that the hybrid neighborhood technique is considerably more powerful than traditional tâtonnement, even over demand structures that are particularly amiable to tâtonnement.

6. CONCLUSIONS AND FUTURE WORK

We explored a new interface of computer science and economics, the combinatorial assignment problem. This problem is distinct from the combinatorial auction problem because of the use of "funny money", that is, the lack of numeraire with which to compare individual utilities. This distinction carries over into solution methods: we showed that the problem cannot be modeled with any polynomial-sized MIP.

Instead, we show that a two-level search process can solve A-CEEI effectively. At the master level, the market searches for prices for items. At the agent level, each agent searches for their most preferred affordable bundle given the prices. The search at both levels leverages a structured utility model (i.e., bidding language) that we developed for representational conciseness, An interesting aspect of our master search is the use of a hybrid neighborhood, which effectively combines two independent ways of generating neighbors. The hybrid neighborhood outperformed the union of each neighborhood on its own. Overall, our method is the first to be able to clear realistic-size course-allocation problems fast enough to be usable in practice. Because our method parallelized so effectively, it will only run faster on the large-scale multicore systems of the future. Prices, which in economics serve as a decentralized messaging system, allow us to fully parallelize the most processor-intensive part of our code. Price-based algorithms are promising for large-scale agent-based systems, especially with the proliferation of multi-core architectures.

We considered only the traditional course-allocation setting but, as we noted, combinatorial allocation has a variety of potential applications, including workers to tasks and scientists to shared resources. Since the fundamental structure of all these problems is the same, we anticipate that our search technique would be able to effectively solve these related problems as well.

The combinatorial assignment problem also opens up a line of theoretical work. Some interesting questions are analogous to similar questions that have been studied in the context of combinatorial auctions. One specific example relates to preference elicitation. In the auction context, theorists have studied the number of queries about agents' valuations required to find an optimal or near-optimal allocation. In the auction context, a query is returned with the bundle that maximizes an agent's value less expenditure; in our context, one could ask similar questions, but queries would be returned with the agent's most preferred affordable bundle. Other interesting questions relate to how best to implement A-CEEI. For instance, if supplied with a demand oracle, how many iterations (of any price-changing algorithm) would it take to find a solution that minimizes error (or at least satisfies Budish's bound), and how does this vary with the structure of agents' preferences.

Acknowledgements

We thank Intel for donating the hardware used to run our experiments. Othman and Sandholm are supported by NSF IIS-0905390.

References

A. Alkan, G. Demange, and D. Gale. Fair allocation of indivisible goods and criteria of justice. *Econometrica*, 59(4):1023–1039, 1991.

E. Budish. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. Technical report, University of Chicago Booth School of Business, 2009.

E. Budish, Y.-K. Che, F. Kojima, and P. Milgrom. Implementing random assignments: A generalization of the birkhoff-von neumann theorem. Technical report, 2009.

T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, second edition, 2001.

L. Ehlers and B. Klaus. Coalitional strategy-proof and resource-monotonic solutions for multiple assignment problems. *Social Choice and Welfare*, 21(2):265–280, 2003.

E. Eisenberg and D. Gale. Consensus of Subjective Probabilities: The Pari-Mutuel Method. *The Annals of Mathematical Statistics*, 30(1):165–168, 1959.

D. Foley. Resource Allocation and the Public Sector. *Yale Economic Essays*, 7(1):45–98, 1967.

F. Glover. Tabu search: a tutorial. *Interfaces*, 20(4):74–94, 1990.

F. Hayek. The use of knowledge in society. *American Economic Review*, 35:519–530, 1945.

H. Hoos and C. Boutilier. Bidding languages for combinatorial auctions. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1211–1217, Seattle, WA, 2001.

K. Jain and V. V. Vazirani. Eisenberg-gale markets: algorithms and structural properties. In *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 364–373, 2007.

H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack problems*. Springer Verlag, 2004.

D. Lehmann, R. Müller, and T. Sandholm. The winner determination problem. In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*, pages 297–317. MIT Press, 2006. Chapter 12.

D. McFadden. Quantal choice analysis: a survey. *Annals of Economic and Social Measurement*, 5(4):363–390, 1976.

P. Milgrom. Package auctions and exchanges. *Econometrica*, 75 (4):935–965, 2007.

P. Milgrom. Assignment exchanges. In *WINE*, 2008.

N. Nisan. Bidding languages for combinatorial auctions. In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*, chapter 9. MIT Press, 2006.

S. Pápai. Strategyproof and nonbossy multiple assignments. *Journal of Public Economic Theory*, 3(3):257–271, 2001.

A. Roth. The economist as engineer: Game theory, experimentation, and computation as tools for design economics. *Econometrica*, pages 1341–1378, 2002.

T. Sandholm. Expressive commerce and its application to sourcing: How we conducted $35 billion of generalized combinatorial auctions. *AI Magazine*, 28(3):45–58, 2007.

H. Scarf. *On the computation of equilibrium prices*. Cowles Foundation for Research in Economics at Yale University, 1967.

T. Sönmez and U. Unver. Course Bidding at Business Schools. *International Economic Review*, Forthcoming.

W. Thomson and H. Varian. Theories of justice based on symmetry. *Social Goals and Social Organizations: Essays in Memory of Elisha Pazner*, 1985.

H. Varian. Equity, envy, and efficiency. *Journal of Economic Theory*, 9(1):63–91, 1974.

V. Vazirani. Combinatorial algorithms for market equilibria. In *Algorithmic Game Theory*, chapter 5, pages 103–134. Cambridge University Press, 2007.