Towards a Fair and Efficient Course Allocation System

DASS End Term Paper

Varul Srivastava
Computer Science and Engineering
IIIT Hyderabad
varul.srivastava@research.iiit.ac.in

***Abstract*—In this paper, we attempt to solve the course allocation problem. That is, allocating M courses among N students. We will explore game theoretic approach to make the process fair, and then use mechanism design principles to make the process computationally efficient. Later we will propose a system design to implement the proposed model.**

***Index Terms*—game theory, mechanism design**

I. Introduction

Course allocation scheme that are under implementation have multiple loopholes. It can be exploited and yield unfair results. The schemes which are fair are computationally expensive, NP hard. We aim to design a course allocation scheme that gives fair results(rare unfairness), and whose implementation is computationally efficient to be used in realistic scenario.

The paper addresses the problem of course allocation. The problem states that, for N courses with u_i seats in each of the courses, and M students with some preference order of the courses, what is the optimal strategy for allocating the courses among the students, so that the scheme is strategy-proof, efficient and fair.

So far, we know that course allocation schemes are either fair and NP hard, or computationally efficient but could be strategically exploit to yield unfair results. Budish[4] has suggested an Approximate Competitive Equilibrium from Equal Incomes (abbrv. A-CEEI) algorithm to the case of indivisible goods. This algorithm satisfies the criteria of fairness and efficiency under tight approximations. We aim to provide a system design for the A-CEEI method of Course Allocation.

Note that the course allocation problem has multiple implementations in other domains as well such as,

- First of all, it is of primal implementation in course allocation of Colleges, and in other institutions like allocation of transfer locations to government officials etc.
- In business strategy this problem can be translated (after a minor modification) to allocation of optimal location for shops, advertisement billboards across a city. This is a non-cooperative game with all agents competing to maximize their own advertisement or sales, or allocation of time-slots on Television advertisement schedule to maximize advertisement.
- In politics, this problem can be seen as allocation of candidates of a Political Party to seats in a constituency, in efficient way, to maximize the number of seats won.
- Similarly, this method can be implemented in distributed systems, for allocation of resources such as CPU-time, RAM and network traffic to meet target (either maximize the number of nodes working at optimal efficiency, or maximize the net throughput of the network etc).

Notice that each method requires us to make some change in the algorithm based on our end-goal, or the type of data given/required. For example, in case of business strategy, the winner determination problem from combinatorial auctions strategy can yield useful results, whereas for course allocation system it fails (reason explained in a later section). The generalized version of the Course Allocation Problem is the multi-unit assignment problems for which multiple objects are assigned to agents on the basis of priorities.

Thus, we realise the urgency of the problem at hand and the need for a computationally efficient solution. In this paper we will explore the problem in multiple fragments. First, we will see the problem at hand, and its proposed solution in Introduction. Then, we explore the current state of research in this domain, and possible motivations that we aim to draw from them. Then we explain the theoretical background, to achieve fairness following which, we explain the system architecture to implement these functions.

II. Literature Review

We first discuss the Multi-Unit assignment problem, and reason out why a fair, strategy-proof solution is NP hard. Then we will discuss some alternative approaches, and finally, we will discuss which of these we have tried to implement in our model, and why.

A. Completely fair, strategy proof multi-unit assignment problem is NP hard

The problem of assigning multiple objects to multiple agents based on priorities, has some problems. First, description of the entire preference profile requires listing up to 2^N subsets of courses. Thus processing such requests for M students itself is of O($M \times 2^N$). Thus the problem is NP hard[13].

Other than this, deferred acceptance mechanism can produce unstable matching as Nash Equilibrium's outcome[14].

B. Winner Determination of Combanitorial Auctions

Another approach to solve the Course Allocation Algorithm was using approach of winner determination problem of combanitorial auctions. We shall see the flaw in this method first, by example, and then discuss on it.

	a	b	c	d
1	65	33	1	1
2	60	32	5	3

TABLE I
PREFERENCE SCORES FOR STUDENTS 1,2

Notice that the preference is analogous to the bidding amount for the course. However, due to the fact that this is not actual currency, this is not a zero sum game for each player vs auctioneer, quite opposite to that of Combanitorial Auctions. Thus, course allocations will be, 1 gets courses a,b and 2 gets courses c,d. However, this is not fair allocation. In this case the fairest allocation is 1 gets a,d and 2 gets b,c. Thus we see this is not a strategy proof method and fairness is not guaranteed.

C. Dominant Strategy implementation of Stable Allocations

The Dominant Strategy of Stable allocations[11] is satisfied if and only if the priorities satisfies essential homogeneity$^{\alpha}$. Even so, the mechanism will be dependent on the number of Seats per course (distribution of seats among courses). To gain independence from this, the priorities have to be acyclic. Thus, the essential homogeneity and acyclicity conditions together impose enough restrictions that the model is not general in nature.[3]

D. Approximate Competitive Equilibrium from Equal Incomes (A-CEEI)

The A-CEEI approach[9] is more suited for large markets, or in our case, large values of N and M, as in such cases the mechanism becomes efficient and approximately strategy proof. Further, the problem is the implementation of A-CEEI, which is complex and computationally intensive.

We focus on deriving an implementation for the A-CEEI model, which, except for it's complexity, promises to be the most general, fair and efficient of the lot. Moreover, we will use some optimization techniques to reduce the computation intensiveness enough for real-time implementation.

III. THE MODEL

We now come to understanding of the A-CEEI Model, and extract inferences that can be exploited for computational efficiency.

A. A-CEEI Theory

Parameters We are given a set of M courses, with the i^{th} course having capacity q_i (integer). Each student has a set of permissible schedules (the schedule one can attend some course at). It is shown as a set (or a binary string $\psi \subseteq \{0,1\}^M$, with an added constraint of k $\leq$ M. There is a set of Utility u_i for each course, for a student. More formally $u : Z \longrightarrow R^+$ such that Z is the set of courses and u(x_i)=u_i for some $x_i \in Z$. Further, there is another utility function $\eta : P(Z) \longrightarrow R$ such that η maps the utility of getting a particular set of courses.

Problem Statement : We need to maximize x_i for each student, such that cumulative utility is maximized. Further we need to satisfy the price constraint for each course, and attempt to minimize the market clearing error.

B. Algorithm

1 Students report their preferences in form of the utility mapping $(u_i)_{i=1}^N$

2 Assign each student a budget b_i such that it is a uniform random number from $[1, 1+\beta]$ where, $0 < \beta << min(\frac{1}{N}, \frac{1}{k-1})$

3 Assign a set of prices $(p_j)_{j=1}^M$ to each course, and allocations $(x_i^*)_{i=1}^N$ to each student such that

- Students maximize utility subjected to budget constraint i.e.

$$\forall i : x_i = argmax_{x' \in \psi_i}[u_i(x_i) : \Sigma_j x_{ij} p_j^* \leq b_i^*]$$

- Minimize the magnitude of market clearing error

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

$$\text{where, } \xi_j = \begin{cases} \Sigma_i x_{ij}^* - q_j & if p^* > 0, \\ max[\Sigma_i (x_{ij}^* - q_i), 0] & otherwise \end{cases}$$

C. Advantage over other methods

Now, we discuss what factors made this algorithm a better choice than the others. First, this algorithm provides versatility of expression. It not only allows us to order courses according to weighted preference but also allows us to add utility if a particular bouquet of courses are selected. Notice this is better because in some methods, we will estimate likeliness of some course a is more than that of b. However, in this case we might express likeness as u(a)=51, u(b)=49 or u(a)=99 u(b)=1. Note that implication is same, but the degree of likeness is displayed by assigning integer weights.

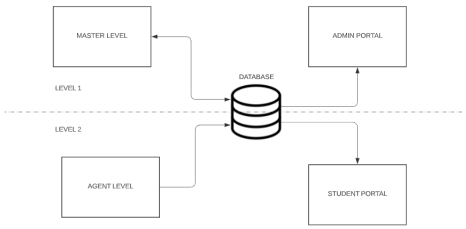
Other than this, this method also adds the option of having more utility for a set of courses, over some discrete courses. It is beneficial in realistic implementations, like for course allocation, it might be more desired to have courses a,b,c together rather than three unrelated but in demand courses d,e,f. Mathematically, $u(a)+u(b)+u(c) < u(d)+u(e)+u(f)$. However it might be that a,b,c together are more desired. In this case, we have added an additional parameter η such that, $u(a)+u(b)+u(c)+\eta(\{a,b,c\}) > u(d)+u(e)+u(f)+\eta(\{d,e,f\})$

IV. SYSTEM ARCHITECTURE

A. Decentralized Multi Layered Architecture

We implement a decentralized multi-layered architecture, to overcome the problem of prompting user for exponential number of entries for η. We propose 2 layers, one the Master

layer, which searches through the price space to decide what price to next propose to the agents. The other layer, i.e. Agent layer searches through the bundle space to find most preferred bundle, at current prices.



Notice in the diagram above, the 2 layers. The processing is done separate from the UI for both layers. A common database is taken for reading-writing.

B. Agent Level Algorithm

1) Input: For N courses, each student has to input their preferences as real values, which sum up to 100 (we can take normalization for this). This function is u_i. We also allow an option to add a set of courses and an additional bouquet utility of η_i associated with it. Note this function η should exist from every subset of courses to real numbers, but we take default value as 0 for non-inputted values. This reduces the number of inputs to non-exponential, (however, user can still input 2^N inputs).

2) Demand Computation: The demand computation is an optimization problem, for an agent given the set of prices, agent's $< u >$ and $< \eta >$. We are given a list of constraints $c_i = (c_{i1}, c_{i2}, ...)$ and courses $x_{ic^1_{ik}}, x_{ic^2_{ik}}, ...$ has capacity constraint $\overline{c_k}$. This problem is algorithmically NP hard. However we can devise a solution using Mixed Integer Programming. The conditions are :

$$max \sum_j v_{ij} x_{ij} + \sum_{j<j'} \eta_{ij,j'} z_{ij,j'}$$

$$subjectedto \sum_j p_{ij} x_{ij} \leq b_i$$

$$z_{ij,j'} = x_{ij} \& x_{ij'}$$

$$\sum_l x_{ic^l_{ik}} \leq \overline{c}_{ik}$$

(& is the bitwise/setwise and operation)

C. Master Level Algorithm

We will have to do local search in Master Level. We will first define the Local Search Model, and then explain the method we use to find neighborhoods for that search model.

1) Local Search: We plan on implementing a variant of Hill-Climbing Search Algorithm. It remembers the last t (distinct) positions it had expanded to. The reason for this modification is :

- It mitigates the weakness of Hill-Climbing (of getting stuck in local minimum).
- It retains the ability of Hill-Climbing of finding optimal solution.

This algorithm is named the **Tabu Search Algorithm[12]**. Our search space is the set of prices and induced demands. We consider equality of 2 nodes when they generate same **aggregate demand** $\sum_i x^*_{ij}$. For minimization of market clearing error, we keep the convergence test as 100 iterations of unimproved results (note this does not minimize the market clearing error, but keeps them within theoretical bounds).

2) Neighborhood Selection: In the hill-climb tabu search[12] function, we need to select neighborhoods. For this, we apply a combination of 2 well known processes. They are **Gradient Descent** and **MIP Price Adjustment**. The first moves price along gradient, and the other does individual price adjustments.

The gradient descent process is similar to economic process of tatonnement, where price of goods in more demand were reduced, and those in less demand were increased, to maintain balance between supply and demand. Mathematically, demand $d_j \equiv \sum_i x_{ij}$, supply is q_j and prices p_j

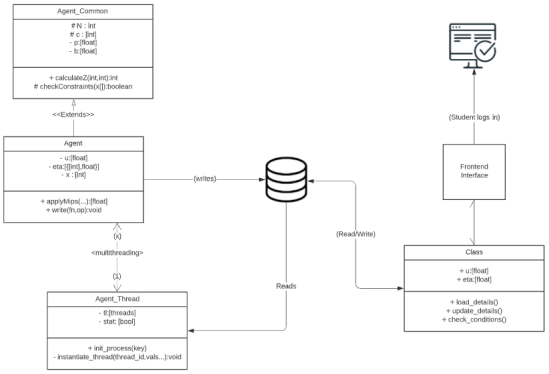
$$\nabla = \begin{cases} d_j - q_j & if p_j > 0, \\ max(d_j - q_j, 0) & otherwise \end{cases}$$

We have another method to adjust price, by each individual course. We change the price of courses in deficit of demand such that demand (no. of students) increases by exactly 1. Notice this, follows conservation, and reduces the demand of courses in over-demand. This process involves using of 2 MIPs[1], and is computationally a little expensive. However, M << N, so it is still not the most expensive procedure computationally.

D. Agent Level Architecture

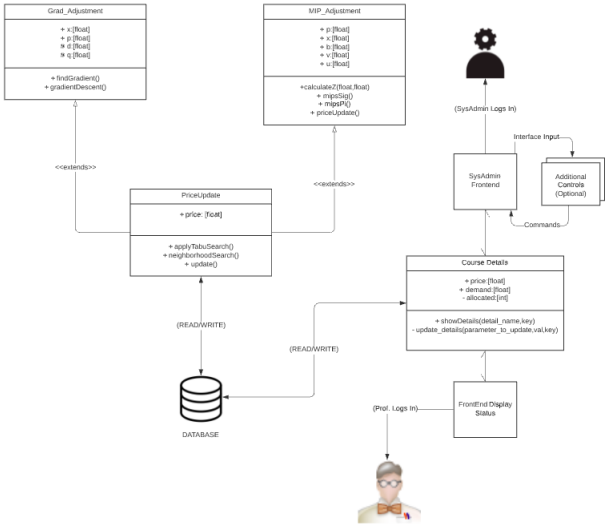
The following shows the UML Class Diagram for the Agent Level. It also shows association and multiplicity. Notice the Database shown will be shared between the 2 levels.

Note the design pattern implemented are Object Pool, Creational Design pattern[8](in the Agent Thread class) as it avoid expensive acquisition and release of resources by recycling threads (objects of agent class) that are no longer in use. Also, it uses the Strategy Behavioural Design[5] Pattern in the Agent class to implement the Demand Computation Algorithm.



E. Master Level Architecture

In master level, we bundle the 2 neighbourhood search algorithm in separate classes, and inherit them in the master class. This allows modularity enough to update the algorithm as and when needed.



In the master level, the design patterns implemented are, Strategy Behavioural Design Pattern[5] to implement Gradient Descent and 2-MIPs algorithms in separate classes.

F. Optimization

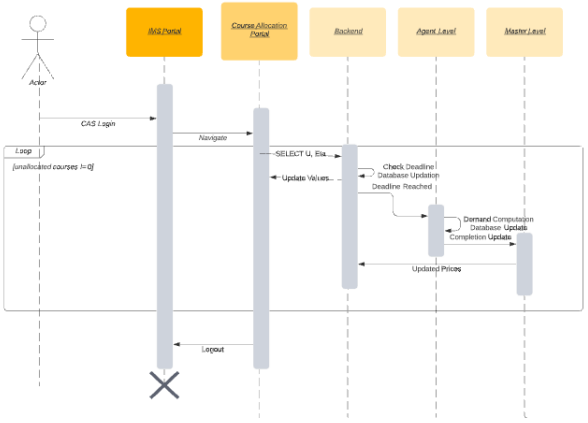
Along with the algorithm, we exploit properties of Economic Theory and Optimization techniques to produce more efficient structures. Notice it is only useful to apply optimization on the slowest/ computationally most expensive portions of the code, which on Agent Level is the Demand Computation, and in Client Level is

1) Parallelism: **Traditional Economic Theory** allows disjoint agents to take decisions dis-jointly. This computational benefit can be exploited by parallelism. Thus for N courses, we can design a multi-threaded model to compute decisions in parallel. We have to experimentally decide optimal balance between system cost and throughput as N can be very large.

2) Hybrid Neighborhoods: We implemented hybrid neighborhoods method, because of its effectiveness over either of the individual methods, i.e. gradient descent, or double-MIPs. It can be inferred from experimental data that in 100 iterations, hybrid neighborhood works for maximum iterations (with changing values). Also, the two different possibilities of user input, i.e. additive separable (realistic inputting) and arbitrary (random input) are both handled by the hybrid variant.

G. Pipeline and Sequence Diagram

The main interaction is between the Student and the Automated system. We will assume multiple rounds of assignment of courses, each with a deadline. The deadline will decide till when editing of preferences is allowed for student, and when execution of program begins.



Notice that Master level interaction with actor (Professor/System-Admin) is trivial and left open for multiple changes according to structure of administration implementing.

V. Conclusion and Future Work

We have thus taken a problem which seemed inefficient and designed a realizable system for it. This describes the power of System Design. We discussed several algorithms, but the versatility and boost in speed by system design achievable in the A-CEEI algorithm was maximum. It is also fair and almost strategy proof.

We notice this system is not generalized but specialized for Course Allocation problem. We can make minor modifications to convert it to other problems discussed in the Introduction section of the paper. I can think of an excellent use of this problem, in consensus algorithms like delegated-proof-of stake[7], where we can select verifying agents based on computational efficiency and fairness. Other than this, modifications can be made to the proposed system, such as cache optimization for Agent-Database connection, for faster computation.

In conclusion, this system is realistic and can be easily implemented due to its modularity. It also allows different segments of development teams (such as back-end, front-end, database, algorithmic) to work without knowing details of algorithm or the types of input from the front-end etc. Further,

due to Strategy Behavioural Design Pattern, modification of the core algorithm does not destroy the entire structure of the system.

ACKNOWLEDGMENT

I would like to acknowledge the clarity gained by taking the course under Prof. R. Loganathan in the process of Optimized System Designs.

REFERENCES

[1] Finding Approximate Competitive Equilibria: Efficient and Fair Course Allocation - E. Budish, A. Othman, T. Sandholm [Paper]

[2] Student Course Allocation with Constraints - Akshay Utture, Vedant Somani, Prem Krishnaa, Meghana Nasre, Meghana Nasre [Paper]

[3] Course Allocation via Stable Matching - Franz Diebold, Haris Aziz, Martin Bichler, Florian Matthes, Alexander Schneider [Paper]

[4] Centralized Course Allocation - Antonio Romero-Medina, Matteo Triossi [Paper]

[5] Design Patterns - sourcemaking(website) [Webpage]

[6] Solving mixed integer programming problems (MIP) - IBM(article/blog) [Link]

[7] DPoS - Bitcoinwiki (article) [Link]

[8] Design Patterns - DASS Lecture Slides - Prof. R. Loganathan

[9] The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. - E. Budish. [Technical Paper]

[10] The use of knowledge in society. American Economic Review, - F. Hayek. [Essay]

[11] Efficient Resource Allocation Under Multi-Demand Games - Kojima, F. [Paper]

[12] Optimization Techniques — Tabu Search - Frank Liang [Blog]

[13] Can Market Participants Report their Preferences Accurately (Enough)? - E. Budish [Article]

[14] Constrained school choice - Guillaume Haeringer,Flip Klijn [Paper]