Combinatorial Auction Based Mechanism Design for Course Offering Determination

Anton Vassiliev, Fuhua Lin(✉), and M. Ali Akber Dewan

School of Computing and Information Systems,
Athabasca University, Edmonton, Canada
anton.a.vassiliev@gmail.com,
{oscarl,adewan}@athabascau.ca

Abstract. Course Offering Determination (COD) is a strategy of an educational institution to maximize the satisfaction of the students and the enrollment of the courses within budget and other resource constraint. COD is a resource allocation problem which is difficult to solve due to the complexity in students' preferences and resource constraints. In this paper, a mechanism for interactive and dynamic decision making is proposed to solve the problem. In this mechanism, the agents negotiate using a protocol which is based on a multi-unit Combinatorial Auctions (CA). To solve the Winner Determination Problem (WDP) in CA, we modified the Branch On Bids (BOB) algorithm to account for multi-unit nature of courses as well as the other constraints from students and administrators. A case study demonstrates the ability and effectiveness of the proposed mechanism in COD for the graduate and undergraduate level studies.

Keywords: Course Offering Determination · Combinatorial auctions · Winner Determination Problem · Branch On Bids · Multi-agent systems

1 Introduction

Every semester or every year administrators of academic programs in colleges and universities must determine the courses that their institution will be offering for the upcoming semester or year. We refer to the task of determining the set of courses to be offered as Course Offering Determination (COD). The main question COD attempts to answer is: *What course offering determination strategy for a program in an institution maximizes the satisfaction of the students and maximizes the enrollment of the courses within the budget and the other resource constraints* [1]*?*

The importance of COD cannot be overestimated. For students, the courses offered drive their ability to graduate, timelines for achieving selected majors, prospects regarding the continuation of academic career and overall success in the job market. For institutions and program administrators, student's satisfaction with the courses offered leads to higher the enrollment. Career and academic achievements upon graduation of the students shape the reputations of the institutions. The courses offered also affect the institutions' budget and consume other resources such as the availability of instructors,

V.G. Duffy (Ed.): DHM 2017, Part I, LNCS 10286, pp. 376–392, 2017.
DOI: 10.1007/978-3-319-58463-8_32

lecture halls, experiment laboratories etc. Today in practice, to determine the course offerings, a program administrator uses the history of offered courses and resource availability. The best judgment and experience of the program administrator plays the vital role among the various deciding factors. This process does not directly consider students' preferences and lacks precision in budget and resource allocation.

In this paper, we model the COD problem as a multiagent dynamic constraint resource allocation problem and design a mechanism for interactive and dynamic decision-making to solve the problem. In the proposed mechanism, the agents negotiate using a protocol based on a multi-unit combinatorial auctions (CA) [2]. To solve the Winner Determination Problem (WDP) in CA, we extend the Branch On Bids (BOB) algorithm to account for multi-unit nature of courses as well as other constraints from students and administrators. The application of CA to COD aims to improve process transparency, lower transaction costs, and introduce a robust method to account for the complementary nature of values for the offered courses.

The rest of the paper is organized as follows: Sect. 2 reviews the related work. Section 3 presents the proposed multiagent system for COD. Section 4 discusses the modified BOD algorithm for COD. Section 5 discusses the performance of the algorithm. Section 6 concludes the paper and discusses the future work.

2 Literature Review

There have been considerable researches in multiagent course scheduling [3], collaborative learning environment [4], student advising application [5], time-table scheduling system for educational institutions [6] and collaborative personal study planning system [7]. Recently, some researches have also been done on multiagent-based solutions for COD that addresses the issues of balancing student preferences with institution's limited resources [1]. The Single Transferable Vote (STV) protocol [8] is used to aggregate student preferences, and the COD negotiation protocol is modeled via Petri Nets [9]. The COD system and the negotiation protocol are modeled via Contract-Net Protocol (CNP) [10] and Monotonic Concession Protocol (MCP) [11].

The COD system proposed in this paper is inspired by the system in [1] and expands upon it with the use of CAs. It introduces bids on course bundles and offers simplicity and transparency to the end users while retaining all the efficiency associated with the previous approach of [1]. CAs allows several functional flexibilities and advantages over conventional auction design [2]. It allows bidders to bid on bundle of items instead of limiting to the bid on a single item at a time. Bidding on bundles allows accounting for the complimentary values of the items composing the bundles, which may result in increased economic efficiency and higher utility for both buyers and sellers.

CAs have been used in many large-scale real-life applications with a great success. To name a few, Sears used CAs to design the auction of eight hundred and fifty-four delivery routes and reducing its logistic costs by thirteen percent [12]; the Federal Communications Commission (FCC) used CAs for selling spectrum licenses for wireless services [13, 14]; the London Transport used CAs to distribute city bus routes between bidding contractor companies [15]; and Bell Laboratories proposed a CA based system for allocating airport time slots in USA [16].

An imperative topic for CAs is the bidding language. A bidding language assigns semantic meaning to syntactic constructs that are used for definition and representation of combinatorial bids. Certainly, the way we represent bids has a great influence on the CA protocol design and we must approach it with caution, keeping the specifics and the goals of our COD application in mind. Moreover, the cardinality of the bidding space in CAs is quite large (2^m) with m number of auctioned items, so a succinct language that does not require excessive bidding is of the great importance. Logical languages and generalized logical languages provide very powerful syntax with flexible semantics [17]. They offer the ability to express complex preferences to a considerable depth with ease. Bidding languages for mixed multi-unit CAs expand that ability to include bids with quantity ranges [18].

While this expressive power injects the possibility for a wide range of bidding combinations, they introduce increased complexity to winner determination. In the proposed mechanism, we rely on a simpler version of a bidding language to streamline the initial design of the CA-based COD protocol, with little or no loss to COD functionality. To determine the most efficient assignment of the bundles to the winning bidders, the CA auctioneer must solve the Winner Determination Problem (WDP) [19]. The main challenge of WDP is that it is an intractable NP-complete problem and it presents significant difficulties for time-efficient and space-efficient solutions. There exist many algorithms that offer both exact and approximate solutions for WDP [2]. The most obvious solution is the explicit, full enumeration of all possible combinations. However, this approach is impractical and the computational effort quickly grows intractable.

The more sophisticated solutions to this problem are the Integer Programming [20] that searches the extreme of the objective function that represents the CA, the Dynamic Programming [21] that splits the CA into smaller problems using the bottom-up principle and the Branch On Bids (BOB) [2] that minimizes the search spaces by pruning branches that would not provide a satisfactory solution. The BOB stands outstanding than the others because it offers an efficient method to model multi-unit auctions [22] and a way to utilize problem-specific heuristics to prune search branches and improve efficiency [23]. For this research, the multi-unit auctions and problem-specific heuristics serve as compelling reasons for selecting the BOB as the way to solve the WDP.

3 Multi-agent System for COD

The proposed multi-agent system consists of three types of agents: Auctioneer Agent (AA), Student Agent (SA) and Administrator Agent (AD). The Auctioneer Agent (AA) plays the role of mediation between AA and SAs by using a protocol based on CA. It enables a negotiation between multiple SAs and the AD, which results in a set of courses.

A graphical outline of a COD process is shown in Fig. 1, where AA, SAs and AD are involved in negotiation for an optimal course offering using CA. In this architecture, SA, AD, and AA represent a student, a program administrator and a CA auctioneer, respectively. The SA and AD work in the best interest of the human actors they represent and the AA agent coordinates their effort via a CA-based protocol to achieve a mutually agreeable and efficient solution. The COD negotiation is triggered by the AD agent which prepares a set of must offer courses and a set of negotiable courses and forwards them to the AA and SA agents. The SA agent generates bids for the CA protocol and the AA agent manages the auction.

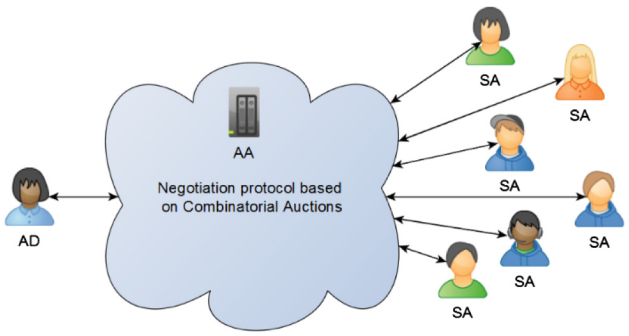

Fig. 1. Multi-agent system for COD.

3.1 Combinatorial Auction (CA) Based Negotiation Protocol

Two important concepts that are needed to be clarified first to explain the CA-based negotiation protocol: *negotiable courses* and *course bidding points*. The CA-based negotiations for COD begins before course registration for a school year. Program administrators notify their respective AD to begin building a set of must-offer courses, a set of must-not-offer courses, and a set of *negotiable courses*. The list of must-offer courses and must-not-offer courses depends on previous faculty commitments and does not depend on student preferences. Program courses that are not included in the must-offer and must-not-offer sets comprise the set of *negotiable courses*. *Course bidding points* are used for bidding that are allotted to each SA at the beginning of a program and are never refilled. A winning bid will withdraw the spent points from the student's account, thus emulating the use of money in conventional auctions.

CA-based negotiation protocol works in the following steps. The set of negotiable courses and the set of must-offer courses are forwarded to AA. The AA agent forwards these courses to all SAs and initiates a CA-based negotiation. The SAs begin to negotiate by working with the students to update the study plan based on the students' preferences, must-offer courses, and negotiable courses, and thus determine the set of available courses for bidding. The students select the bundles of available courses that she/he will prefer to take over the school year and places their bids on these bundles. Each student can place multiple bids at each round, and once the set of bids is determined, it is forwarded to AA. The AA aggregates the bids from all students and uses BOB algorithm to solve the WDP for the current bidding round. A complete flow diagram for CA-based negotiation protocol is shown in Fig. 2.

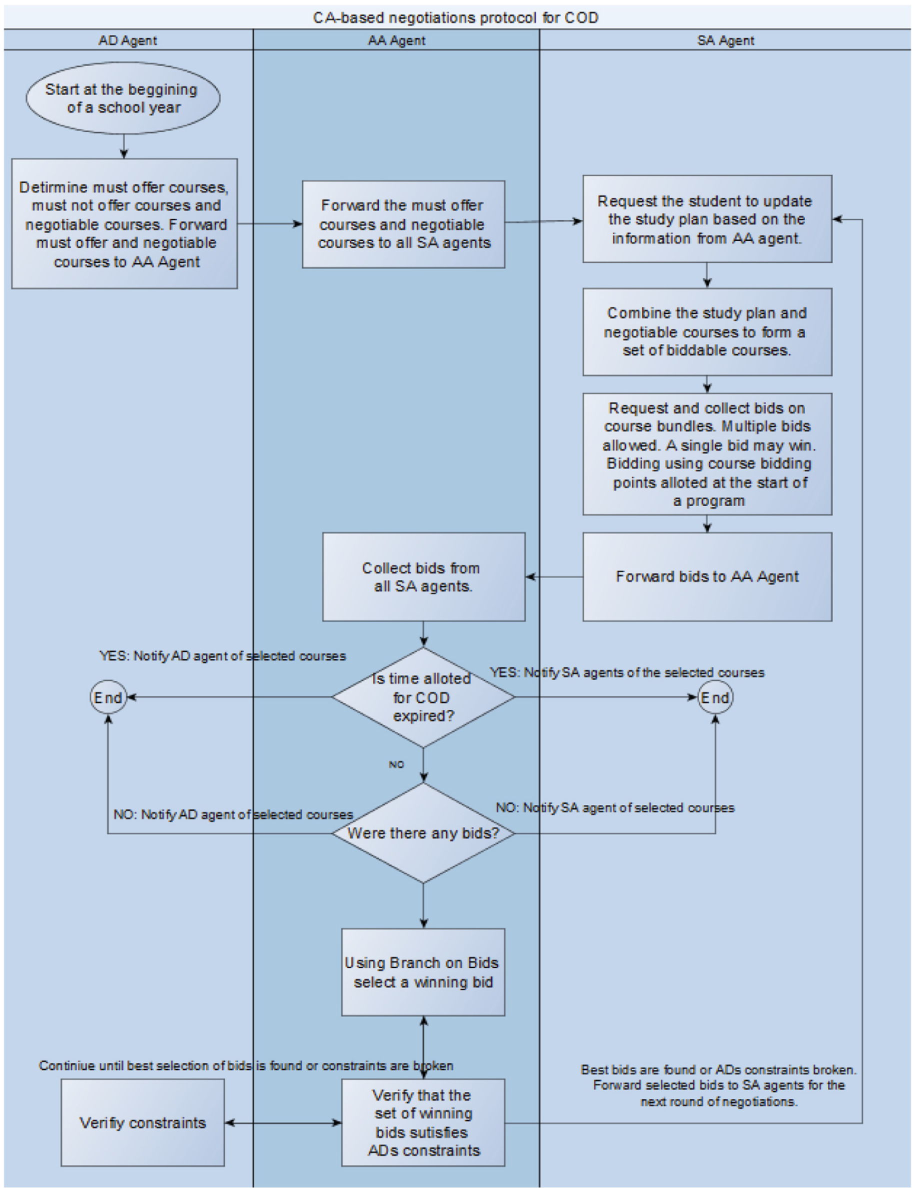
CA-based negotiations protocol for COD
AD Agent
AA Agent
SA Agent
Start at the beggining of a school year
Detirmine must offer courses, must not offer courses and negotiable courses. Forward must offer and negotiable courses to AA Agent
Forward the must offer courses and negotiable courses to all SA agents
Request the student to update the study plan based on the information from AA agent.
Combine the study plan and negotiable courses to form a set of biddable courses.
Request and collect bids on course bundles. Multiple bids allowed. A single bid may win. Bidding using course bidding points alloted at the start of a program
Forward bids to AA Agent
Collect bids from all SA agents.
YES: Notify AD agent of selected courses
Is time alloted for COD expired?
YES: Notify SA agents of the selected courses
End
End
NO
NO: Notify AD agent of selected courses
Were there any bids?
NO: Notify SA agent of selected courses
Using Branch on Bids select a winning bid
Continiue until best selection of bids is found or constraints are broken
Best bids are found or ADs constraints broken. Forward selected bids to SA agents for the next round of negotiations.
Verify constraints
Verify that the set of winning bids sutisfies ADs constraints

Fig. 2. CA-based negotiation protocol for COD.

3.2 Branch on Bids (BOB) Algorithm

A modified version of BOB algorithm is used in the proposed architecture considering the multi-unit nature of courses and the other constraints from SA and AD. In each round, the winning set of bids is verified with AD to ensure that the winning solution complies with the AD's constraints. The BOB algorithm is executed until registration is completed or until the AD's constraints are satisfied. At this point, the winning bids are forwarded to SAs and the next cycle of the protocol begins. The protocol terminates when the time allotted for the auction is expired or if no more bids were placed. The last selected set of winning bids combined with must offer courses form the course offering for the year. As the proposed mechanism of COD relies on the BOB algorithm to find the best bids, we reviewed a generic BOB algorithm [22, 23] to solve WDP in CA below this section. The modified version of BOB algorithm used for the proposed mechanism is explained in Sect. 4.

Combinatorial Auction (CA) can be defined by a set of items $M = \{i_{1,}\ldots, i_m\}$ available for bidding, where $|M| = m$; and a set of bids $B = \{b_1, \ldots, b_n\}$ generated by the bidders, where $|B| = n$. A bid b_j is defined as a two-tuple, $b_j = <S_j, p_j>$, where S_j is the set of bidding items and p_j is the bidding price such that $S_j \subseteq M$, $p_j \geq 0$. To identify the winner in CA, the WDP is solved with the following function:

$$\max \sum\nolimits_{j=1}^{n} p_j x_j \tag{1}$$

where $\sum_{j|i \in S_j} x_j \leq 1$, $i = 1, \ldots, m$, and $x_j \in \{0, 1\}$. Here, $x_j = 1$ signifies a winning bid b_j and $x_j = 0$ signifies a losing bid b_j. Here, we must maximize the sum of winning bids, while ensuring that the sets of items in winning bids do not intersect. The last condition is imposed by the assumption that each offered item is unique. The unique item requirement must be relaxed for solving COD. The BOB algorithm to solve a

Table 1. Terms and symbols used in BOB (see Algorithm 1)

Terms and symbols	Description
IN	Set of bids that are labeled winning on the path to the current search node
*IN**	Set of bids that are winning in the best allocation found so far
g	Revenue from the bids in *IN*
*f**	Revenue from the bids in *IN**
e_j	Exclusion count for bid b_j
M'	Set of unallocated item in the current search path
h	Upper bound on the revenue from the unallocated items, M'
c(i)	This is the admissible heuristic for estimating revenue for item $i \in M'$. The *c(i)* is calculated using the formula $max_{j\|i \in S_j, e_j=0} p_j / \|S_j\|$. This can be interpreted as the maximum per-item bidding price among all bids containing the item *i* and that are not part of IN or IN*
ChooseBranch	Selecting a bid b_k to branch on. Let B_0 be a set of bids, where $B_0 \subseteq B$ and $\forall$ $b_j \in B_0$, $e_j = 0$, then $b_k \in B_0$ and $\forall$ $b_j \in B_0 \frac{p_k}{\sqrt[2]{\|S_k\|}} \geq \frac{p_j}{\sqrt[2]{\|S_j\|}}$

typical WDP in CA is presented in Algorithm 1, where all the terms and symbols used in the algorithm is listed in Table 1.

Algorithm 1. Branch On Bids (BOB) Pseudo code.

```
1:  begin
2:      if $g>f^*$ then        //if current search is better, remember it as all-time best
3:          $IN \rightarrow IN^*$
4:          $g \rightarrow f^*$
5:      end if
6:      $h = \sum_{i \in M`} c(i)$      //Calculate the upper bound
7:      if $g+h \leq f^*$ then     // Check if this branch can produce better results than
                                   we already have. // If not – bound
8:          return
9:      end if
10:     Select a bid $b_k$ using ChooseBranch algorithm
11:     if $b_k = NULL$ then
12:         return
13:     end if
14:     $IN \cup \{b_k\} \rightarrow IN$     // Prepare to branch on $b_k$
15:     $e_k=1$
16:     $\forall \; b_j \in B |\; b_j \neq b_k$ and $S_j \cap S_k \neq \emptyset$   // Update the exclusion count for all
                                                     the bids that share items with $b_k$
17:     $e_j = e_j + 1$
18:     BOB (M`- $S_k$, g + $p_k$)                   // Branch in
19:     $IN - \{b_k\} \rightarrow IN$                   // Prepare to branch out
20:     $\forall \; b_j \in B |\; b_j \neq b_k$ and $S_j \cap S_k \neq \emptyset$   // Update the exclusion count for all
                                                     the bids that share items with $b_k$
21:     $e_j = e_j - 1$
22:     BOB(M`,g)                                  //Branch $b_k$ out
23:     $e_k = 0$                                    // Done with this branch
24:     return
25: end
```

In *ChooseBranch* function, the square root of cardinality of item subsets has been shown to be effective at selecting bids with balance between high price bids, but with large number of items, verses bids with low item count, but also with low bidding value [22, 24]. This is an important feature that under proper circumstances may result in faster convergence to the winning bid allocation. The BOB (see Algorithm 1) takes M′ and *g* as its input. Upon completion of BOB execution, *IN** contains the set of winning bids. The first call is BOB(M, 0).

3.3 Interaction Model and Bidding Language

The visual interactions model is presented in Fig. 3. All COD interactions that happen during CA negotiations are listed in Table 2 with their descriptions.

Bidding language used for auction is introduced below, where we assign semantic meaning to syntactic constructs that are later used to define student's course bids.

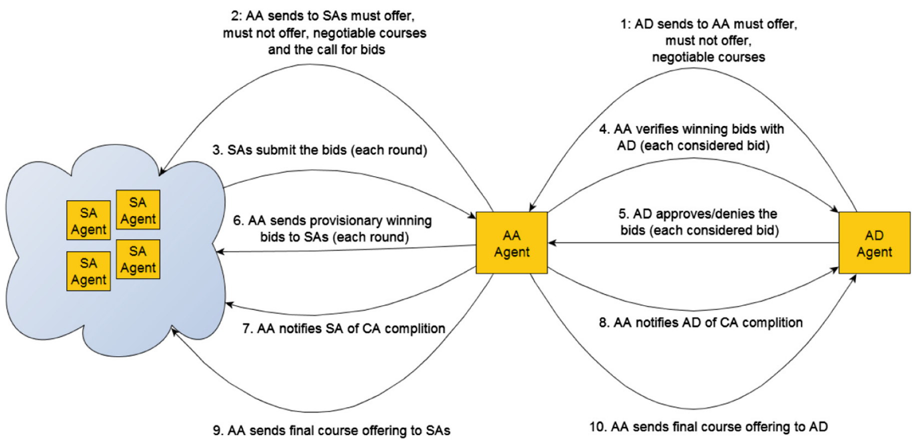
2: AA sends to SAs must offer, must not offer, negotiable courses and the call for bids
1: AD sends to AA must offer, must not offer, negotiable courses
3. SAs submit the bids (each round)
4. AA verifies winning bids with AD (each considered bid)
SA Agent
SA Agent
SA Agent
SA Agent
6. AA sends provisionary winning bids to SAs (each round)
AA Agent
5. AD approves/denies the bids (each considered bid)
AD Agent
7. AA notifies SA of CA complition
8. AA notifies AD of CA complition
9. AA sends final course offering to SAs
10. AA sends final course offering to AD

Fig. 3. COD interaction model

Table 2. COD interaction model

Step	From	To	Description
1	AD	AA	AD sends to AA the sets of must offer, must not offer and negotiable courses
2	AA	SA	AA sends to SAs the sets of must offer, must not offer, negotiable courses and the call for bids
3	SA	AA	SAs submit the bids (each CA round)
4	AA	AD	Each time a new bid is considered as a provisional winning bid, AA sends the set of provisional winning bids to AD for verification against resource constraints
5	AD	AA	AD approves or denies the provisional winning bids proposed in step 4. The decision is based upon available school resources
6	AA	SA	Upon completion of each round, AA sends all provisionary winning bids to SAs. New round begins by cycling to step 3. Students place new bids against provisionary winning bids in step 6
7	AA	SA	AA notifies SA of CA completion
8	AA	AD	AA notifies AD of CA completion
9	AA	SA	AA sends the final course offering to SA
10	AA	AD	AA sends the final course offering to AD

Definition 1: A *submission* is consisted of three-tuple <*B, sa, po, co*>, where $\boldsymbol{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n\}$ is a set of bids with $|B| = n$, *sa*is the name of the SA agent submitting the bid, *po*is the total amount of course bidding points available for the SA agent, and *co* is the maximum number of courses the student can take. An example for a submission is given in Table 3.

Table 3. An example for a submission

Variable	Value for example 1
Submission	<B, “sa1”, 10, 6>
B	$\{b_1, b_2, b_3\}$
b_1	$<S_1, 6>$
S_1	{“comp501”, “comp503”, “comp504”}
b_2	$<S_2, 6>$
S_2	{“comp504”, “comp505”, “comp506”}
b_3	$<S_3, 2>$
S_3	{“comp601”}

Definition 2: A *bid* $\boldsymbol{b_j}$ is a two-tuple $<\boldsymbol{S_j}, \boldsymbol{p_j}>$, where $\boldsymbol{S_j} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s_k}\}$ with $\boldsymbol{S_j} \subseteq \boldsymbol{M}$, $|\boldsymbol{S_j}| = \boldsymbol{k_j}$, $\boldsymbol{k_j} \leq co \leq m + l + t$, and $\boldsymbol{p_j}$ is the amount of course bidding points assigned to this bid by the student.

A set of courses available for bidding is defined as $M = \{i_1, \ldots, i_m\}$ with $|M| = m$. This is a set of courses the students can bid on. A set of must-offer and negotiable courses are defined as $O = \{o_1, \ldots, o_n\}$ with $|O| = l$ and $N = \{n_1, \ldots, n_t\}$ with $|N| = t$, respectively.

A set of rules are while selecting the winning bid using the CA protocol. The set of bids B is interpreted by the CA protocol as a non-exclusive OR concatenation of the bids, baring the rules follows:

- Winning bids from the same student may not contain intersecting sets of courses.
- Winning bids may not add up to a total exceeding the amount of course bidding point available to the student
- The number of courses comprising winning bids may not exceed maximum number of courses the student can take
- The courses comprising winning bids may not contradict course interrelationships (for example there may not be any anti-requisites)

4 Modified Branch on Bids (BOB) Algorithm for COD

In contrast to the traditional bidding strategy, a notable change is needed in the proposed COD to adjust the fact that multiple students can bid on the same course or on the same combination of courses and can win. This is allowed because multiple students can take the same course or the same combination of courses at the same time in educational institutions. The only restriction is that the total number students is not allowed to exceed a pre-specified limit which is the maximum enrollment for that course. The above constrained are enforced by the AD agent. We address this challenge through using the notion of multi-unit auctions [22], with the number of available course units equal to the enrollment limit for the course. Using this approach, we can apply the Branch-on-Bids algorithm with only minor modifications.

Algorithm 2. Modified Branch On Bids (BOB) for COD

```
1:  If $g>f^*$, then $IN \rightarrow IN^*$ , $g \rightarrow f^*$  //if current search is better, remember it as the all-time
    best
2:  h=$\sum_{i \in M`} c(i)$ //Calculate the upper bound
3:  // Check if this branch can produce better results than we already have.
    If g+h≤f*, return () // If not – bound
4:  Use the ChooseBranch algorithm to select a bid $b_k$ to branch on
    If no such bid exists, return
5:  IN U $\{b_k\} \rightarrow IN$ // Prepare to branch on $b_k$
    $e_k$=1
6:  Verify that IN set satisfies the SA_Constraints and the AD_Constraints
    If not then IN - $\{b_k\} \rightarrow IN$; $e_k$=0; EX U $\{< d, b_k >\} \rightarrow EX$;
    Return to step 4
7:  // Update the set of available course units U`.
    For each course in $S_k$ subtract 1 from the matching course unit in U`
8:  // Update the set of available courses M`.
    If an item $i_j \in$ M` and $u_j = 0$ then M`=M`-$\{i_j\}$
9:  // Update the exclusion count for all the bids that share courses with $b_k$
    // and have no enrollment places available
    $\forall\ b_j \in B |\ b_j \neq b_k$ and $S_j \cap S_k \neq \emptyset$ and $\exists$ an item $i_e \in S_j \cap S_k$ s.t. $u_e = 0$
    then $e_j = e_j + 1$
10: BOB(M`, U`, g + $p_k$, d+1) //Branch in
11: IN - $\{b_k\} \rightarrow IN$ //Prepare to branch out
12: / Update the exclusion count for all the bids that share courses with $b_k$
    // and have no enrollment places available
    $\forall\ b_j \in B |\ b_j \neq b_k$ and $S_j \cap S_k \neq \emptyset$ and $\exists$ an item $i_e \in S_j \cap S_k$ s.t. $u_e = 0$
    then $e_j = e_j - 1$
13: // Update the set of available course units U`.
    For each course in $S_k$ add 1 to the matching course unit in U`
14: // Update the set of available courses M`.
    If an item $i_j \notin$ M` and $u_j \geq 0$ then
    M`=M`+$\{i_j\}$
15: //Update the exclusions set EX
    Remove all exclusions for branch d
    $\forall\ ex_j \in EX$ s.t. $d_j = d$
    EX = EX – $\{ex_j\}$
16: //Branch $b_k$ out
    BOB(M`, U`, g, d)
17: //Done with this branch
    $e_k$=0;
    return
```

Reusing and adjusting the formal definition of CAs from Sect. 3, we define CA in the context of COD as a three-tuple <*M*, *U*, *B*> , where $M = \{i_1, \ldots, i_m\}$ with $|M| = m$ is a set of courses available for bidding; $U = \{u_1, \ldots, u_m\}$ with $|U| = m$ specifying the maximum enrollment for the matching course; and $B = \{b_1, \ldots, b_n\}$ is a set of bids by students. Finally, the WDP can be defined same as Eq. (1). It is to be mentioned that, in the context of COD an item (a course or a course package) within a bid will always have quantity of exactly one because a student can only enroll in a course once per semester. Therefore, we use a modified version of multiunit WDP [22]. We also add to the set of constraints all limits induced by the limitations of SA agents (see submission interpretations in subsection 3.3) and the resource limitations enforced by the AD agent.

The modified BOB for WDP in the proposed COD relies on the adjusted variables outlined in Table 4 and the proposed BOB algorithm is presented in Algorithm 2. BOB takes on input the set of available courses, their associated available enrollment limits, current search revenue and branch depth. The first call of this algorithm is BOB (M, U, 0, 1). Upon the completion of the algorithm execution the *IN** set will contain the set of winning bids.

Table 4. Branch on Bids for Course Offering Determination variables

Variable	Definition
M′	The set of available courses (courses that have available enrollment places). Can be thought of as a set of auction items that still have available inventory
U′	The set of course units matching M′. When branching in on a bid, the course units for the courses in the bids get subtracted a unit. When branching out on a bid, the course units for the courses in the bid get an extra unit. In other words, this variable tracks available enrollment places in a course
c(i)	The admissible heuristic for estimating revenue for item i∈M′ $c(i) = max_{j \mid i \in S_j, e_j = 0} u_i * p_j / \lvert S_j \rvert$ (this formula can be interpreted as the number of available course units multiplied by the maximum per-item bidding price among all bids containing the item *i* and that are not part of *IN* or *IN**)
d	The depth of the current branch
EX	A set of "exclusion" two-tuples <*d*, *b*> A tuple represents a bid *b* excluded at branch depth *d* due to violation of Administrator or Student constraints
ChooseBranch	The heuristic algorithm for selecting a bid b_k to branch on Let B_0 be a set of bids s.t. $b_j \in B_0$ iff $b_j \in B$ *and* $e_j = 0$ and <d, b_j> $\notin$ EX for any integer d Then $b_k \in B_0$ and $\forall\ b_j \in B_0 \frac{p_k}{\sqrt[2]{\lvert S_k \rvert}} \geq \frac{p_j}{\sqrt[2]{\lvert S_j \rvert}}$
SA_Constraints	Winning bids from the same student may not contain intersecting sets of courses Winning bids may not add up to a total exceeding the amount of course bidding point available to the student The number of courses comprising winning bids may not exceed maximum number of courses the student is allowed to take The courses comprising winning bids may not contradict course interrelationships (for example there may not be any antirequisites)
AD_Constraints	For the purpose of this paper, the AD agent will enforce the maximum number of courses that the school can offer

5 Result and Discussion

5.1 Solution Quality

Distributed multi-agent system based solution quality is a complex, multifaceted topic that involves, amongst others, those of social welfare of agent societies, individual rationality, stability and efficiency [25]. The practical side of COD solution quality should be measured through a pilot project that would follow a group of participating students through their academic progress for a period of one or more semesters. The student's subjective utility with the program and academic performance may be compared to those of non-participating students, drawing the conclusions on the quality of proposed solution. Because of technical complexity of the solution quality measurement and time constrained involved with a pilot program, we followed up with a discussion on a case study based on the quality analysis principles as outlined in [25].

The CA-based COD algorithm is a form of a negotiation mechanism that uses CAs to provide a transparent and economical solution for COD. The mechanism aims to benefit the crucial players of the system: the students and the administrators, increasing social welfare through the mutually satisfactory solution. We measure social benefit of the students through the value they assign to the courses in their bids, and the administrator's benefit through student enrollment (subject to the resource constraints). It follows that since CAs maximize the economical payoff through increasing the course bidding points revenue, the CA-based COD algorithm should lead to the course selection most beneficial within the constructed environment.

Moreover, the proposed solution encourages student's participation in the bidding process through the potential to improve the course offerings for the bidding student. Participation in the bidding process will always lead to the results that are at least as good as for non-participating students thus providing the individual rationality. The bidding process and the limited course bidding points inject stability into the protocol, promoting truthful bidding for the courses the students are interested in.

5.2 Performance

An unsatisfactory performance may render even the best designs unusable in practice. In the case of CA-based COD algorithm, the WDP is solved off-line and therefore the expectation is that the problem is solved in a reasonable time and the solution process does not consume much computational resources in a manner that significantly detrimental to the rest of the services in an academic institution. The looseness of this performance requirement is possible because the COD needs to be addressed only once a semester or once a year, and the bidding rounds are executed once a day (or possibly on some other prolonged schedule) during the COD process and do not require immediate response.

The version of BOB utilized in this paper is a search algorithm that builds and traverses a binary bid-search tree by branching on bids [22]. The performance of BOB is proportional to the number of leaves in our search tree which is not greater than $(nk/m+1)^{\lfloor m/k \rfloor}$, where n is the number of bids, m is the number of items, and k is the smallest number of items among all bids.

The number of leaves is exponential on the number of items (courses available for bids), but polynomial on the number of bids. This is an important and positive observation because program administrators can control the number of biddable items (courses) in case of unsatisfactory algorithm performance. The number of bids, however, depends on the number of students and their personal preferences and involvement, and therefore is harder to control.

The performance of original BOB algorithm is analyzed in [23]. Bid sets containing from five hundred to two thousand bids for sets of ten items were solved in less than six seconds. While bid sets containing four hundred and fifty bids for sets of forty five items were solved in less than twenty seconds. In the context of COD these results appear to be very promising. We can estimate that the number of items (courses) in most COD auctions is unlikely to exceed a few dozen. The number of bids will vary between programs and schools and will greatly depend on student enrollment, however

Table 5. Partial flow of BOB for COD example

Execution Step	BOB Step	Comments
1:	n/a:	BOB ({c1, c2, c3, c4, c5}, {u1 = 1, u2 = 2, u3 = 2, u4 = 1, u5 = 1}, 0, 1)
2:	1:	n/a
3:	2:	h = c(c1) + c(c2) + c(c3) + c(c4) + c(c5) = 2 + 4 + 3 + 2 + 2 = 13
4:	3:	13 > 0 – proceed
5:	4:	Following the ChooseBranch algorithm we select bid <{c1, c2}, 4>
6:	5:	IN = {<{c1, c2}, 4>} $e_1 = 1$
7:	6:	No constraint violations – proceed
8:	7:	U′ = {u1 = 0, u2 = 1, u3 = 2, u4 = 1, u5 = 1}
9:	8:	M′ = {c2, c3, c4, c5}
10:	9:	No bids intersect with b_1 – proceed
11:	10:	BOB({c2, c3, c4, c5}, {u1 = 0, u2 = 1, u3 = 2, u4 = 1, u5 = 1}, 4, 2)
12:	1:	IN* = {<{c1, c2}, 4 >}, f* = 4
13:	2:	h = c(c2) + c(c3) + c(c4) + c(c5) = 0 + 3 + 2 + 2 = 7
14:	3:	4 + 7 = 11 > 4 – proceed
15:	4:	Following ChooseBranch algorithm we select bid <{c4, c5}, 4>
16:	5:	IN = {<{c1, c2}, 4>, <{c4, c5}, 4>} $e_3 = 1$
17:	6:	No constraint violations – proceed
18:	7:	U′ = {u1 = 0, u2 = 1, u3 = 2, u4 = 0, u5 = 0}
19:	8:	M′ = {c2, c3}
20:	9:	$e_2 = 1$
21:	10:	BOB({c2, c3}, {u1 = 0, u2 = 1, u3 = 2, u4 = 0, u5 = 0}, 8, 3)
22:	1:	IN* = {<{c1, c2}, 4>, <{c4, c5}, 4>}, f* = 8
23:	2:	h = c(c2) + c(c3) = 0 + 3 = 3
24:	3:	8 + 3 = 11 > 8 – proceed
25:	4:	At this point there are not more bids available – return

for most the programs it will remain under a few thousand. It appears likely that WDP for COD may be solved within a reasonable time-frame, measuring seconds or minutes.

5.3 A Case Study

In this section, we provide a simple case study that illustrates the partial listing of the BOB for COD algorithm flow that terminates the listing at the first leaf. Please note the simple form of the student preference submissions, which represents a clear improvement over precedence, grouping and progression model used in [1, 26]. Rationality for the winning bids is also easy to understand, which should lead to students expressing their preferences through bidding rounds with higher precision.

In the case study, we assume that due to resource constraints (*AD_Constraints*) we can offer at most five courses, where

M = {c1, c2, c3, c4, c5}
U = {u1 = 1, u2 = 2, u3 = 2, u4 = 1, u5 = 1}
Submission1: <{<{c1, c2}, 4>, <{c3, c4}, 3>}, sa1, 10, 4>
Submission2: <{<{c4, c5}, 4>}, sa3, 10, 4>

The execution flow is provided in Table 5.

We terminated our example at step 25, however the algorithm will continue to explore remaining branches, determining {<{c1,_c2}, 4>, <{c4,_c5}, 4>} as the winning bids. Figure 4 shows the complete progress of the algorithm in a graphical form. The bright orange rectangles represent bids in the IN set, the grey rectangles represent the excluded bids.

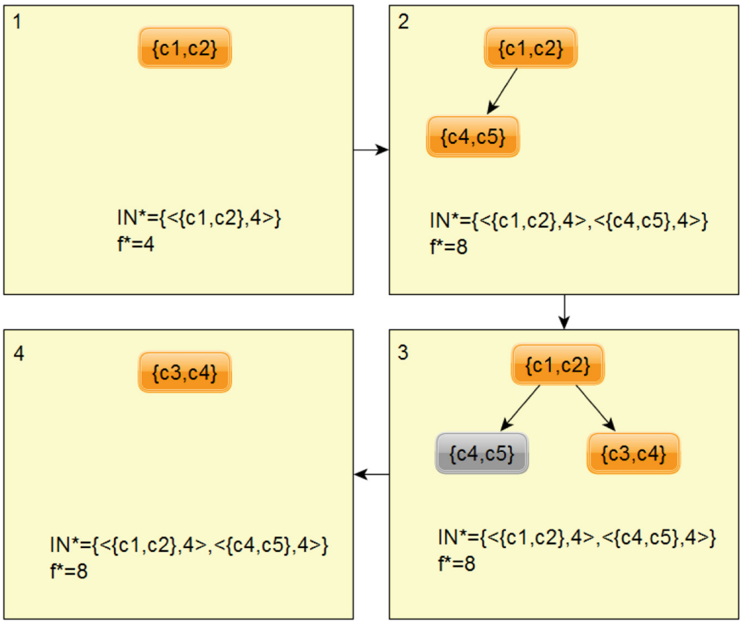
1
{c1,c2}
IN*={<{c1,c2},4>}
f*=4
2
{c1,c2}
{c4,c5}
IN*={<{c1,c2},4>,<{c4,c5},4>}
f*=8
3
{c1,c2}
{c4,c5}
{c3,c4}
IN*={<{c1,c2},4>,<{c4,c5},4>}
f*=8
4
{c3,c4}
IN*={<{c1,c2},4>,<{c4,c5},4>}
f*=8

Fig. 4. BOB for COD example.

6 Conclusions

The main contribution of this research is the introduction of CAs to the process of the COD negotiations. CAs bring a few benefits to distributed negotiation protocols, for instance: accounting for the complimentary values of the course bundles, increased economic efficiency, expressive negotiations format that lowers transaction costs and high level of transparency that ensures fairness.

The challenge of efficient negotiation was met with introduction of multi-unit auctions, and the ability for multiple students to win bids on the same courses was handled through multi-unit auctions. The intractable WDP was tackled using the BOB algorithm that offers polynomial performance on bids and generalization that includes the multi-unit solutions.

The expected performance of the CA-based COD solution promises to be within reasonable and practical limits. The most computation-costly part of the algorithm, the WDP, is solved off-line and is expected to complete execution within seconds or minutes for the most extreme cases.

A potential area of interest for CA-based COD research may be the application of it to the Massive Open Online Courses (MOOC). The scale and popularity of online programs serve as a powerful argument in favor of application of CA-based COD, which aspires to improve economic efficiencies. Further experiment and elaboration of the CA-based COD solution will be done to improve its efficiency and flexibility:

- The bidding language may be extended to include complex logical combinations of bids, such as “AND” and “EXCLUSIVE OR”;
- Student and Administrator constraints may be improved to provide more realistic rules. For example, Administrators may not only limit the total number of courses, but also courses that may not be offered together;
- The *ChooseBranch* and c(i) heuristics may be improved from their general form to more applicable algorithms that consider the specifics of COD;
- The SA and AD agents may be extended with the use of Machine Learning techniques to self-customization for providing a more personalized service to the students and administrators they represent.
- A data mining system may be designed to learn the effects of the CA-based COD on the welfare of the system, and empower research for further improvements.
- The proposed mechanism can be extended to have multiple-round bidding that may promotes a fair allocation of coveted courses. When a student is unable to obtain a desired course in an early round, she or he will have more points to bid for courses that might be offered in a later round.

References

1. Lin, F., Chen, W.: Designing a multiagent system for course-offering determination. In: Boella, G., Elkind, E., Savarimuthu, B.T.R., Dignum, F., Purvis, M.K. (eds.) PRIMA 2013. LNCS (LNAI), vol. 8291, pp. 165–180. Springer, Heidelberg (2013). doi:10.1007/978-3-642-44927-7_12

2. Schwind, M.: Combinatorial auctions for resource allocation. In: Schwind, M. (ed.) Dynamic Pricing and Automated Resource Allocation for Complex Information Services. LNEMS, vol. 589, pp. 137–190. Springer, Heidelberg (2007). doi:10.1007/978-3-540-68003-1_5
3. Oprea, M.: MAS-UP-UCT: a multi-agent system for university course timetable scheduling. Int. J. Comput. Commun. Control **2**(1), 94–102 (2007)
4. Vassileva, J., McCalla, G., Greer, J.: Multi-agent multi-user modeling in I-Help. User Model. User-Adap. Inter. **13**(1), 179–210 (2003)
5. Hamdi, M.S.: MASACAD: a multiagent-based approach to information customization. IEEE Intell. Syst. **21**(1), 60–67 (2006)
6. Tariq, M., Mirza, M., Akbar, R.: Multi-agent based university time table scheduling system. Int. J. Multidiscip. Sci. Eng. **1**(1), 33–39 (2010)
7. Vainio, A., Salmenjoki, K.: Improving study planning with an agent-based system. Informatica **29**, 453–459 (2005)
8. Bartholdi, J.J., Orlin, J.B.: Single transferable vote resists strategic voting. Soc. Choice Welfare **8**(4), 341–354 (1991)
9. Fehling, R.: A concept of hierarchical Petri nets with building blocks. In: Rozenberg, G. (ed.) ICATPN 1991. LNCS, vol. 674, pp. 148–168. Springer, Heidelberg (1993). doi:10.1007/3-540-56689-9_43
10. Smith, R.G.: The contract net protocol: high-level communication and control in a distributed problem solver. IEEE Trans. Comput. **C-29**(12), 1104–1113 (1980)
11. Rosenschein, J.S., Zlotkin, G.: Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers. MIT Press, Cambridge (1994)
12. Ledyard, J., Olson, M., Porter, D., Swanson, J., Torma, D.: The first use of a combined-value auction for transportation services. Interfaces **32**(5), 4–12 (2002)
13. Crampton, P., Kwerel, E., Rosston, G., Skrzypacz, A.: Using spectrum auctions to enhance competition in wireless services. J. Law Econ. **54**(4), 167–188 (2011)
14. Porter, D., Rassenti, S., Roopnarine, A., Smith, V.: Combinatorial auction design. Natl. Acad. Sci. **100**(19), 11153–11157 (2003)
15. Vries, S., Vohra, R.: Combinatorial auctions: a survey. Informs J. Comput. **15**(3), 284–309 (2003)
16. Rassenti, S., Smith, V., Bulfin, R.: A combinatorial auction mechanism for airport time slot allocation. Bell J. Econ. **13**(2), 402–417 (1982)
17. Boutilier, C., Hoos, H.: Bidding languages for combinatorial auctions. In: International Joint Conference on Artificial Intelligence, Seattle, USA, August 2001
18. Cerquides, J., Endriss, U., Giovannucci, A., Rodriguez-Aguilar, J.: Bidding languages and winner determination for mixed multi-unit combinatorial auctions. In: International Joint Conference on Artificial Intelligence, Hyderabad, India, January 2007
19. Lehmann, D., Müller, R., Sandholm, T.: The winner determination problem. In: Cramton, P., Shoham, Y., Steinberg, R. (eds.) Combinatorial Auctions, pp. 297–318. MIT Press, Cambridge (2006)
20. Anderson, A., Tenhunen, M., Ygge, F.: Integer programming for combinatorial auction winner determination. In: International Conference on Multiagent Systems, Boston, MA, July 2000
21. Rothkopf, M., Pekec, A., Harstad, R.: Computationally manageable combinatorial auctions. Manag. Sci. **44**(8), 1131–1147 (1998)
22. Sandholm, T., Suri, S.: BOB: improved winner determination in combinatorial auctions and generalizations. Artif. Intell. **145**, 33–58 (2003)
23. Sandholm, T., Subhash, S., Gilpin, A., Levine, D.: CABOB: a fast optimal algorithm for combinatorial auctions. In: International Joint Conference on Artificial Intelligence, Seattle, USA, August 2001

24. Lehmann, D., O'Callagham, L., Shoham, Y.: Truth revelation in rapid, approximately efficient combinatorial auctions. J. ACM **49**(5), 96–102 (2002)
25. Sandholm, W.: Distributed rational decision making. In: Multi-agent Systems, pp. 201–258. MIT Press, Cambridge (1999)
26. Armstrong, A.J.: Optimizing course-offerings with MAS. Master's Essay, Athabasca University, Athabasca, Alberta, March 2012
27. Weiss, G.: Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence. MIT Press, London (1999)
28. Conitzer, V.: Making decisions based on the preferences of multiple agents. Commun. ACM **53**(3), 84–94 (2010)
29. Stone, P., Veloso, M.: Multiagent systems: a survey from a machine learning perspective. Auton. Robot. **8**(3), 345–383 (2000)
30. Wooldridge, M., Jennings, N.R., Kinny, D.: The gaia methodology for agent-oriented analysis and design. J. Auton. Agents Multi-agent Syst. **3**(3), 285–312 (2000)
31. Winikoff, M., Padgham, L.: Agent-oriented software engineering. In: Multiagent Systems, pp. 695–758. MIT Press, Heidelberg (2014)
32. Dorca, F.A., Lopes, C.R., Fernandes, M.A.: A multiagent architecture for distance education systems. In: IEEE International Conference on Advanced Learning Technologies, Athens, Greece, July 2003
33. Graesser, A., Chipman, P., Haynes, B., Olney, A.: AutoTutor: an intelligent tutoring system with mixed-initiative dialogue. IEEE Trans. Educ. **48**(4), 612–618 (2005)
34. Mitrovic, A., Ohlsson, S.: Evaluation of a constraint-based tutor for a database language. Int. J. Artif. Intell. **10**, 238–256 (1999)
35. Wilson, D.C., Leland, S., Godwin, K., Baxter, A., Levy, A., Smart, J., Najjar, N., Andaparambil, J.: SmartChoice: an online recommender system to support low-income families in public school choice. AI Mag. **30**(2), 46–58 (2009)
36. Lin, F., Leung, S., Wen, D., Zhang, F., Kinshuk, McGreal, R.: E-advisor: a multi-agent system for academic advising. Int. Trans. Syst. Sci. Appl. **4**(2), 89–98 (2008)
37. Cernuzzi, L., Molensini, A., Omicini, A., Zambonelli, F.: Adaptable multi-agent systems: the case of the gaia methodology. Int. J. Softw. Eng. Knowl. Eng. **21**(4), 491–521 (2011)
38. Moraïtis, P., Petraki, E., Spanoudakis, N.I.: Engineering JADE agents with the gaia methodology. In: Carbonell, Jaime G., Siekmann, J., Kowalczyk, R., Müller, Jörg P., Tianfield, H., Unland, R. (eds.) NODe 2002. LNCS (LNAI), vol. 2592, pp. 77–91. Springer, Heidelberg (2003). doi:10.1007/3-540-36559-1_8
39. Wooldridge, M.: A modern approach to distributed artificial intelligence. In: Multiagent Systems, pp. 27–77. MIT Press, Cambridge (1999)
40. Symeonidis, A., Mikas, P.: Agent Intelligence Through Data Mining. Springer, Heidelberg (2005)
41. Russel, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, Upper Saddle River (2010)
42. Sturm, A., Shehory, O.: Agent-oriented software engineering: revisiting the state of the art. In: Shehory, O., Sturm, A. (eds.) Agent-Oriented Software Engineering, pp. 13–26. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54432-3_2
43. JASON. http://jason.sourceforge.net/wp/. Accessed Nov 2015