

Towards Fast Vickrey-Pricing using Constraint Programming

Alan Holland and Barry O’Sullivan
(`{a.holland|b.osullivan}@cs.ucc.ie`)
*Cork Constraint Computation Centre,
Department of Computer Science, University College Cork, Ireland.*

Abstract. Ensuring truthfulness amongst self-interested agents bidding against one another in an auction can be computationally expensive when prices are determined using the Vickrey-Clarke-Groves (VCG) mechanism. VCG guarantees that each agent’s dominant strategy is to tell the truth, but it requires solving $n + 1$ optimization problems where the overall optimal solution involves n agents. This paper first examines a case-study example demonstrating how Operations Research techniques can be used to compute Vickrey prices efficiently. In particular, the case-study focuses on the Assignment Problem. We show how Vickrey prices can be computed in the same asymptotic time complexity as that of the original optimization problem. This case-study can be seen as serving a pedagogical role in the paper illustrating how Operations Research techniques can be used for fast Vickrey pricing. We then propose a Constraint Programming (CP) approach which can be used in a more general context, where nothing is assumed about the nature of the constraints that must be satisfied or the structure of the underlying problem. In particular, we demonstrate how no-good learning can be used to improve the efficiency of Vickrey pricing in Combinatorial Auctions.

1. Introduction

Eliciting truthful responses from self-interested agents has been previously studied in game theory and economics. A class of Vickrey-Clarke-Groves (VCG) mechanisms have been developed whereby the dominant strategy for any agent is to tell the truth, meaning that rational agents maximize their utility by truthfully revealing their preferences [26]. Truthfulness is a dominant strategy when there is no circumstance under which an agent can benefit from lying. VCG is also commonly referred to as the Generalized Vickrey Auction (GVA).

In a GVA bids are solicited from the agents for a range of items being auctioned. An optimal solution is found, and subsequently the payment to each agent, or Vickrey price, is determined by finding the cost of an optimal solution without that agent present. Each agent cannot control the price paid to it because it is a function of the other agents’ preferences. This removes any potential profiteering by insidious providers. For an overview of Mechanism Design in general, and the

VCG mechanism in particular, the reader is referred to Mas-Collel *et al* [18].

The research that we report in this paper has many practical applications. For example, the Computational Grid has most recently been defined as “*coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations*” [9]. Consequently the Grid presents difficulties in terms of scheduling and brokering of distributed applications. We can regard the service providers in the Grid as self-interested agents whose principle motivation is to maximize their own profit when bidding for tasks. Truthfulness in such a scenario is desirable but implementing it within a mechanism can become computationally expensive.

The vision for the future of the Grid is analogous to that of the electricity grid. Computing power, storage facilities, specialized software and other resources should be provided on-demand and the source of the desired resource should be transparent to the end-user. These resources are ultimately provided by a wide range of agents, who can be regarded as self-interested and motivated by profit. They may act dishonestly if they deem it beneficial. These agents can pool their resources in a co-operative in which a broker disseminates tasks based upon solicited bids for them.

The main contributions of this paper are as follows:

- We consider a case-study example, where n agents are bidding for m tasks where $m < n$. This corresponds to a weighted bipartite matching problem, otherwise known as the Assignment Problem. We show how Vickrey prices can be computed in the same asymptotic time complexity as that of the original optimization problem, thus tackling a problem posed by Hershberger and Suri [12]. This case-study can be seen as serving a pedagogical role in the paper illustrating how Operations Research techniques can be used for fast Vickrey pricing.
- We then propose a Constraint Programming (CP) approach which can be used in a more general context, where nothing is assumed about the nature of the constraints that must be satisfied or the structure of the underlying problem. In particular, we demonstrate how no-good learning [25] can be used to improve the efficiency of Vickrey pricing in Combinatorial Auctions, which are NP-complete in general.

This paper is organized as follows. Section 2 presents a brief overview of Mechanism Design. Section 3 presents a detailed case-study of fast Vickrey pricing for the Assignment Problem, which is equivalent to

scheduling m tasks on n machines where $m < n$. It gives a general description of the problem and shows how the Hungarian Method can be used for fast Vickrey pricing. Section 4 presents an approach to fast Vickrey pricing based on Constraint Programming for the more general case of Combinatorial Auctions. Some concluding remarks are made in Section 5.

2. A Brief Overview of Mechanism Design

Mechanism Design is concerned with specifying the rules of a game where a collection of agents, each holding private information about their preferences over a set of outcomes, interact with each other in their own self-interest. An example scenario is an auction where the agents are the bidders and the set of outcomes describes the possible assignments of goods to bidders. A mechanism is then a tuple that comprises the set of possible actions that an agent may take and an outcome rule. A strategy defines the actions an agent will take over all its preferences. An example strategy may be to always exaggerate your true valuation for an item by a given amount. The outcome rule then assimilates the actions of the agents and determines an outcome. A crucial assumption in Mechanism Design is that all the agents are rational and selfish, and therefore seek to maximize their own utility by choosing an appropriate strategy.

2.1. MECHANISM DESIGN: THE GOALS

The traditional goal of Mechanism Design is to design a game in which an overall equilibrium (or equilibria) is reached according to some desirable system-wide properties, given that all participating agents act in their own self-interest. A *social choice function* (SCF) describes the properties that the outcome should possess.

Some typical desirable properties include the following:

- *Individual Rationality*: The SCF makes guarantees about an agent's utility with respect to their participation in an auction. This is an important property if agent participation is voluntary.
- *Efficiency*: The outcome must maximize overall agent utility, thereby maximizing social welfare.
- *Revenue Maximizing*: A single agent, an auctioneer for example, maximizes her revenue (utility).

- *Budget Balance*: The sum of all agent payments equals zero, therefore no money is extracted or injected into the system. This is particularly important for any self-sustaining mechanism where no external benefactor exists to subsidize the system.

These desirable properties may directly conflict with each another. Budget Balance and Efficiency conflict in the VCG mechanism, which achieves only the latter property. The goal of the mechanism, for example, in the context of task allocation, is to try to find the most efficient means of getting a set of given tasks completed, satisfying all of their timing and quality of service constraints. However, the VCG mechanism is not revenue-maximizing and is dependent on a benevolent mediator to subsidize the payments in the outcome that are necessary to incentivize truthful bidding.

2.2. THE GENERALIZED VICKREY AUCTION

For a given mechanism, a *solution concept* is used to predict the strategies agents will choose in order to maximize their utility, thus determining an equilibrium position for the game. The GVA has a *dominant strategy* equilibrium. This means that the best response strategy for each agent remains the same, irrespective of its knowledge about other agents or their actions. Agents bid their truthful valuation for an item and do not profit from under-estimating or exaggerating their valuation even if they know all other bids. This is a powerful solution concept and makes it unprofitable for agents to concern themselves with other agents bids.

The GVA also ensures optimal efficiency, thereby maximizing social welfare according to some objective such as fairness. However, it is not budget balanced, and a benevolent external party, such as a government, may be required to supplement the budget.

To determine payments to each agent participating in the overall optimal solution, the overall cost is determined without each agent present in turn. This involves $m + 1$ optimization problems if m agents participate in the optimal solution to the allocation problem. For example, in an auction for task assignment where any one of n agents can receive only one task out of the m being auctioned, $n - m$ agents will not participate in the overall optimal solution so there is no need to determine Vickrey payments for these agents.

2.3. DISADVANTAGES OF GENERALIZED VICKREY AUCTIONS

However, there are several disadvantages associated with the Vickrey mechanism which are worth highlighting. If non-optimal solutions are

found to the optimization problems that determine the prices paid in a GVA (based on the Vickrey-Clarkes-Groves mechanism) the mechanism is no longer guaranteed to be truthful. This is a major disadvantage of the GVA. Various polynomial-time heuristics and approximation algorithms can provide good or near optimal solutions very quickly. However, Nisan and Ronen [20] showed, constructively, that a non-optimal solution can in fact result in payments arbitrarily far from optimum. If an auctioneer seeks to approximate optimal solutions in a GVA using polynomial-time algorithms the results may not be reliable and agents may have an incentive to lie.

Other limitations include reduced revenue compared with other auctions and susceptibility to a fraudulent auctioneer. It is possible for an auctioneer to introduce fake bids just below the value of the winning bids to increase revenue. For this reason a trustworthy auctioneer is imperative in a GVA.

A GVA is also sensitive to bidder collusion. Bidders may coordinate their bid prices so that the bids remain artificially low. In this manner, the bidders get the item at a lower price than they normally would.

3. Case-Study: An OR-Based Approach

In order to demonstrate a classical approach to reducing the complexity of computing Vickrey payments, we will focus on the well-known Assignment Problem and an Operations Research technique to solving it. This case-study can be regarded as serving a pedagogical role demonstrating how Operations Research techniques can be used for fast Vickrey pricing. The approach presented in this section relies on the characteristics of the underlying problem. Later in this paper we will present a more general approach to the more general case of Combinatorial Auctions based on Constraint Programming which makes no assumptions about the structure of the underlying problem, while still achieving improvements in the efficiency of Vickrey pricing.

We chose to focus on the Assignment Problem for the purposes of our case-study for the following reasons. Hershberger and Suri solved an open problem presented by Nisan and Ronen [19], when they presented an algorithm for reducing the computational complexity of Vickrey payments for the specific case of shortest path routing in networks [12]. They reduced the overall time complexity to the same asymptotic complexity of the original problem. They also posed the following question: “*Can one achieve similar improvements in other network settings?*”. Hershberger and Suri also identified the problem of auctioning m tasks

amongst n self-interested agents, where $m < n$, and how an optimal task allocation requires solving the Assignment Problem.

An algorithm for solving it is the *Hungarian Method*, upon which we base our case-study. This is a well-known primal-dual algorithm. There are a number of approaches in the literature which are similar to this approach. Bikchandani et al. [1] investigated how payments that bidders receive under the Vickrey auction correspond to duality in linear programs. Recent work by David Parkes [22] has also developed a primal-dual method to determine the Vickrey outcome efficiently. The approach we demonstrate here uses techniques similar to those used by Parkes.

In the Assignment Problem, otherwise known as the (Minimum Cost or Maximum Weight) Bipartite Matching Problem, the objective is to find the matching with the greatest total weight, or alternatively with the lowest total cost. It has a diverse range of applications including aircraft pilot rostering and dormitory room assignment.

In the case of task auctioning each agent owns some set of resources and submits her bid to complete a given task using these resources. The task description contains the due-date and Quality of Service requirements. If an agent cannot meet these demands it will not bid for the task. The optimal solution, having minimal cost, is computed by the auctioneer. The computation involves assigning all the tasks to separate agents. The Vickrey payments to each of the successful bidders is computed by removing each of them in turn from the problem and resolving the matching problem.

3.1. THE HUNGARIAN METHOD

The Hungarian Method is an algorithm that finds a maximum cardinality match in a graph. In our scenario this would equate to solving a matching problem in which all the edges have the same cost. The Kuhn-Munkres algorithm [15] is an extension of the Hungarian Method in which an optimal assignment to a problem with weighted edges is solved, and this algorithm is often referred to as the Hungarian Method itself [14, 21]. It is no longer the state-of-the-art OR algorithm for solving the Assignment Problem but it does offer a straightforward implementation and an incremental architecture for finding solutions to slightly differing problems.

The Hungarian Method has $O(mn^2)$ time complexity, therefore solving m problems is $O(m^2n^2)$, when there are m tasks and n bidders. However, Papadimitriou and Steiglitz [21] have proposed a data structure for the Hungarian method that reduces the time complexity to $O(mn^2)$.

However, there are other approaches that could be taken. Floyd's algorithm can be applied to find the shortest paths that takes $O(mn^2)$. Hung et al. describe in [13] an algorithm for finding what they term as “the most vital edges of matching in a bipartite graph”. An edge is called a most vital edge (MVE), with respect to a weighted matching, if its removal from the graph results in the largest decrease in the total weight of the maximum matching, or alternatively, the greatest increase in cost in a minimal cost matching. The MVE algorithm finds a maximal weighted matching with each edge removed in turn, in order to find the most important edge. This is precisely what is required to calculate Vickrey prices for each provider.

3.2. BIPARTITE MATCHING PRELIMINARIES

A graph G is bipartite if the vertex set $V(G)$ can be partitioned into two sets X and Y such that no two vertices in the same set are adjacent. In the case of the task allocation problem, the set X may represent the set of agents and set the Y represent the tasks. The edges between the nodes in these two sets are the bids, where $|X| = n$ and $|Y| = m$ ¹. See Figure 1 for an example of a bipartite graph.

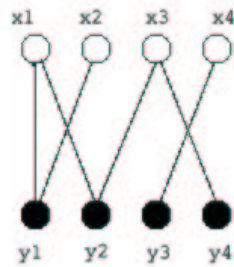


Figure 1. An example of a bipartite graph.

A *matching* in a graph G occurs if no two edges have a common end-vertex. A matching with the largest possible number of edges is called a *maximum matching*. If M is considered to be a matching of a graph G , a vertex v is said to be *saturated* or *covered* by M if any edge of M is incident with v . A vertex that is *unsaturated* is also called *free*. A path, tree or cycle is *alternating*, relative to M , if its edges are alternately in $E(G) \setminus M$ and M , where $E(G)$ denotes the set of edges in G . A path is an *augmenting path* if it is an alternating path with free

¹ Note that this notation for bipartite matchings differs from general graph literature where m denotes the number of edges in a graph and n for the number of vertices. Our notation is derived from [21].

origin and terminus. A graph H is a *subgraph* of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$ denoted by $H \subseteq G$. H is a spanning subgraph of G when $V(G) = V(H)$.

The following property of matchings illustrates the importance of augmenting paths [23]:

Property 1. Let M be a matching of G and P is an augmenting path relative to M . The set difference of M and P , $M \Delta P$, is also a matching for G and $|M \Delta P| = |M| + 1$

The exploitation of this property is integral to most maximum matching algorithms because it permits them to incrementally solve Assignment Problems. Thus, this property can be exploited for enabling fast computation of Vickrey payments.

3.3. FAST VICKREY PAYMENTS FOR THE ASSIGNMENT PROBLEM

To compute Vickrey payments quickly we exploit the incrementality of the Hungarian Method. In particular, we compute the best matchings $M_i = M(G \setminus x_i)$ for $i = 1, \dots, m$ where agent x_i participates in the overall optimal solution, $M(G)$. Recall that optimal solutions are required for each agent absent in turn in order to determine Vickrey payments for each agent.

The Hungarian Method can solve the problem incrementally, recomputing the optimal matching in a single iteration when a single edge is discarded at a node of the search tree. After computing the solution to the original minimum cost matching problem, we can discard all the edges incident with the vertex representing the agent whose payment is being determined. An extra stage of the Hungarian Method will then lead to an optimal solution without that agent present.

We can, in fact, say that Vickrey pricing for n providers bidding for m tasks, where $m < n$, in the VCG mechanism, can be computed in $O(mn^2)$ time complexity using the Hungarian Method. Each stage of the Hungarian Method augments the matching by one edge, therefore there is a maximum of $m + 1$ stages to be considered. To examine the complexity of each stage, the complexities for the search and dual variable modifications must be considered separately. Search requires $O(n^2)$ operations, because each vertex is examined sequentially and never re-examined. The removal of a vertex from X requires $O(n)$ operations. However, for dual variable modifications we either have a new vertex that is labeled or we have an augmentation. We can therefore have a maximum of n dual variable modifications at each stage. Each modification takes $O(n)$ time so each stage requires $O(n^2)$ operations.

Solving m problems from scratch incurs $m(m+1)$ stages in total, therefore the overall time complexity becomes $O((m^2+m)(n^2)) = O(m^2n^2)$. However, we can use the optimal solution to the original optimization problem to our advantage. One of the vertices in X is altered so that one agent's bids all become zero, effectively removing it from the problem. The subsequent problem therefore only requires one further stage to modify the dual variables and search for an augmenting path. The time complexity to solve m subproblems in this manner becomes $O(mn^2)$.

Note that we only need to solve m subproblems, even though there are n providers because only m providers were assigned tasks in the original solution so each one of these providers is omitted from each subproblem in turn.

3.4. EMPIRICAL VERIFICATION

We have used an implementation of the Hungarian Method based upon Donald Knuth's Stanford GraphBase [14] that uses data structures proposed in [21] and gives a time complexity of $O(mn^2)$. When calculating prices we have $m+1$ problems to solve so we would naturally expect the running time to become $O(m^2n^2)$, but the incremental nature of the Hungarian Method allows us to change one row of the matrix and recompute a solution in $O(n^2)$.

We present the results of our experiments in Figure 2. Our main interest in this graph are the three lines representing the results for the incremental approach that solves the Vickrey payment problems incrementally. The slopes of all three lines ≈ 1 . This empirical evidence confirms the time complexity is $O(mn^2)$. Also notice that the naive approach of solving all pricing problems from scratch has slope ≈ 2 . This confirms its complexity is $O(m^2n^2)$.

This result demonstrates a positive answer to the question posed by Hershberger and Suri [12]: whether a lower bound for the auctioning of m tasks amongst n providers where $m < n$, can be reduced to less than that of computing $m+1$ optimization problems.

4. A Constraint Programming Approach

The approach presented in the previous section can be regarded as a classical approach to fast Vickrey pricing: we match an algorithm to the characteristics of the problem we wish to solve. In the case of the task allocation problem we have presented in the previous section, the Assignment Problem provides a convenient model for encoding the problem, while the Hungarian Method is one technique which can be

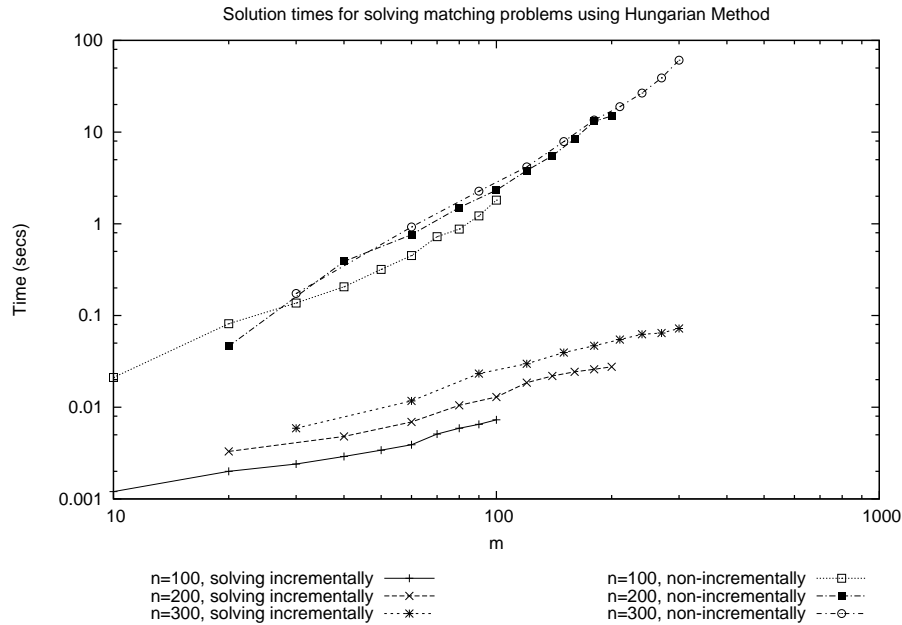


Figure 2. Comparative results for solving Vickrey pricing problems incrementally versus solving all pricing problems from scratch.

used to compute an optimal solution and then compute Vickrey prices efficiently. In this section we present a generic approach to computing Vickrey prices efficiently in the context of Combinatorial Auctions, which are known to be NP-complete in general. The approach is based on Constraint Programming [5, 16]. Taking inspiration from our case-study, we will present a technique which learns as it searches so that we can compute Vickrey prices more efficiently than if we did not learn. Furthermore, the approach we present can be regarded as more robust than classical methods since additional arbitrary constraints, which we will refer to as *side* constraints, on the auction can be naturally assimilated into the constraint model.

The Constraint Satisfaction Problem (CSP) [16] is modeled as a set of n variables $X = \{x_1, \dots, x_n\}$, a set of domains $D = \{D(x_1), \dots, D(x_n)\}$, where $D(x_i)$ is the set of finite possible values for variable x_i and a set $C = \{C_1, \dots, C_m\}$ of constraints, each restricting the assignments of some subset of the variables in X . Constraint satisfaction involves finding values for each of the problem variables such that all constraints are satisfied. Its main advantages are its declarative nature and flexibility in tackling problems with arbitrary side constraints.

Constraint Optimization seeks to find a solution to a CSP that optimizes some objective function. A common technique for solving

constraint optimization problems is to use branch-and-bound techniques that avoid exploring sub-trees that are known not to contain a better solution than the best found so far. An initial bound can be determined by finding a solution that satisfies all constraints in C or by using some heuristic methods.

4.1. DYNAMIC PROBLEMS AND NO-GOOD LEARNING

Schiex and Verfaillie introduced the concept of *no-good learning* for Constraint Satisfaction Problems [25]. This technique was designed to tackle the solution maintenance problem in a Dynamic CSP where the problem is subject to change. A Dynamic CSP is defined as a sequence of static CSPs, each one resulting from a modification to the preceding one. This change may be a restriction (addition of a constraint) or a relaxation (removal of a constraint). The main motivation for using no-goods for Dynamic CSPs is to improve the search for new solutions to the problem as efficiently as possible as changes occur over time.

No-goods also provide a basis for learning during search by helping to avoid using assignments that cannot lead to a solution. For example, consider the following no-good: $\{(x_1 = 1), x_1 \geq x_2\}$. This no-good states that x_1 cannot be assigned the value 1, because of the constraint $x_1 \geq x_2$.

DEFINITION 1. *A no-good is a pair (A, J) , where A is an assignment of values to a subset of variables in the CSP and $J \subset C$ is the set of constraints precluding A from any solution to the CSP (X, J) . The set J is termed the no-good justification [25].*

These techniques have been applied to Constraint Optimization Problems, where the notion of *valued no-goods* has been introduced [3]. A valued no-good incorporates a value v such that every complete extension of the assignment A has a valuation greater than or equal to v . An example of a valued no-good may be $\{(x_1 = 7, x_2 = 7), 100\}$. This no-good implies that if $x_1 = 7$ and $x_2 = 7$, we will not be able to extend to a solution with an objective function value greater than 100 (when the objective is to maximize the overall solution value).

DEFINITION 2. *A valued no-good is a pair (A, v) , where A is a partial assignment of values to a subset of variables in the CSP and v is a valuation such that every complete extension of A has a valuation less than or equal to v [3].*

Note that it is possible to have a valued no-good at every node of the search tree. A valued no-good may also become redundant when

another valued no-good is placed above it in the search tree, so pruning redundant no-goods may become necessary depending on the size of the search space.

4.2. CONSTRAINT PROGRAMMING AND COMBINATORIAL AUCTIONS

We have studied how Constraint Programming (CP) techniques can be used to improve the efficiency with which Vickrey payments can be computed in Combinatorial Auctions, where bidders may show preferences for bundles of items. Combinatorial Auctions enhance economic efficiency because bidders can specify preferences over combinations of items that suit their needs rather than having the auctioneer impose restrictions on sets of items that can be sold together.

In a Combinatorial Auction the auctioneer is faced with determining the winning bids that maximize revenue. This is a Weighted Set-Packing Problem, otherwise known as the *Winner Determination Problem* (WDP). The WDP is NP-complete and inapproximable but there has been considerable progress in tackling this problem recently [24].

Dynamic CSP offers the ability to accommodate a change to the problem specification and re-solving it quickly. Subsequent pricing problems in a GVA may be viewed as perturbations to the initial problem formulation, representing the auction, and no-goods learned while solving the initial problem are still useful.

While CP may be unable to compete with specially tailored algorithms for specific problem classes, such as the Hungarian Method for the Assignment Problem, in terms of performance, it has considerable potential when applied to auctions with arbitrary side constraints that preclude the use of tailored approaches.

4.3. CONSTRAINT-BASED VICKREY PRICING

Calculating Vickrey payments may be viewed in terms of a Dynamic CSP. However, we have an advantage in that we can restrict the possible changes to the CSP to the removal of a single agent and her bids from the original problem. Each Vickrey pricing problem is a restriction of the original problem in which a constraint is added stipulating that all the bids of a particular bidder lose. No-goods learned in the original optimization problem may be used in all subsequent Vickrey pricing problems to improve pricing efficiency.

There are no new no-goods recorded after the original problem because such no-goods learned on a subsequent Vickrey problem are not necessarily applicable to other Vickrey problems that may be considered a relaxation, including parts of the search space not explored in another pricing problem.

One drawback of no-good learning is its worst-case exponential space complexity. To establish a polynomial bound on the space complexity required by no-good learning, the cardinality of any assignment A in a no-good may be restricted to being no greater than a given constant i , that we refer to as the *order* of the no-good [4, 25]. Fortunately, smaller assignments, shorter no-goods, are more useful because they eliminate larger branches of the search tree. We use the term “0th-order no-good” to refer to the situation where we do not learn no-goods.

4.4. EXPERIMENTAL ANALYSIS

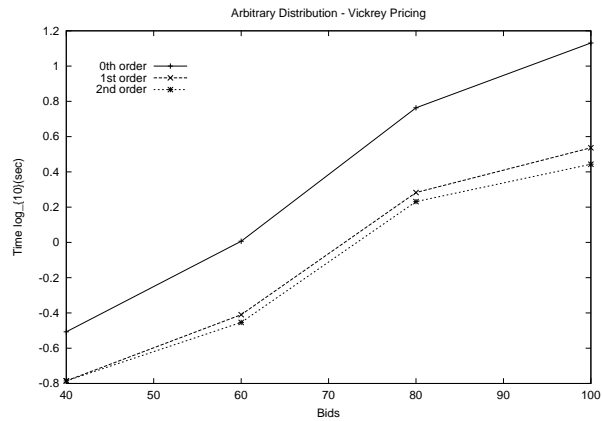
We used the Combinatorial Auction Test Suite (CATS) [17] to generate sample auction data in order to test the performance of no-good learning in improving the efficiency of Vickrey pricing in a Combinatorial Auction. We generated sample problems ranging from 40-100 bids on 20 items. We used both the *arbitrary-npv* and *regions-npv* distribution types. Distribution types in CATS are analogous to real-life scenarios. The *regions-npv* simulates an auction for items that complement each other because of their adjacency, such as a radio spectrum auction. The *arbitrary-npv* distribution indicates agents preferences exhibit arbitrary complementarities².

We generated Constraint Optimization Problems from the CATS output files by representing bids as 0-1 variables in lexicographical order. An item shared between bids induces a binary constraint precluding the instantiation of both relevant variables to 1. We adopted a conservative approach to grouping bids to agents, whereby any bids not containing the same dummy item were regarded as being from different bidders. CATS uses dummy items in bids to simulate XOR-bids or substitutability³. Ten instances of each problem were used to generate average results. Our solver is implemented in Java (jdk 1.4.2) and all experiments were conducted on a 1.8GHz Pentium 4 machine running Windows 2000.

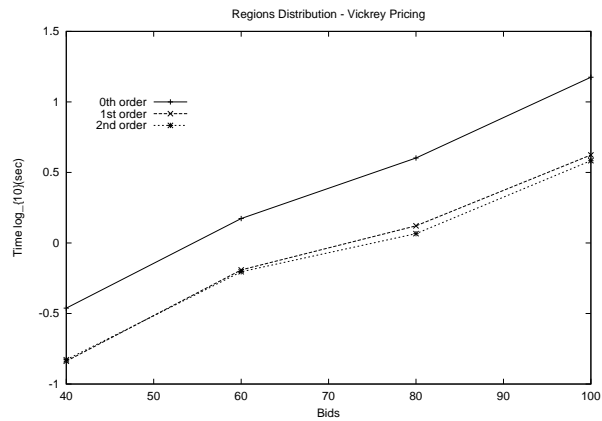
No-good learning leads to major improvements over basic branch-and-bound search (using 0th order no-goods, i.e. not learning during search at all) in the Vickrey Pricing problems, as can be clearly seen from Figure 3. The overhead incurred by recording no-goods is negated by the performance improvement in the original problems. In fact, all

² The CATS flags included `int_prices` with the `bid_alpha` parameter set to 1000.

³ Dominated bids are excluded by CATS, but such bids may influence Vickrey payments. Bids that are *only* dominated by the relevant agents bids in a pricing problem *should* be included, so as the problem size grows the relative effect of this omission is reduced.



(a) Arbitrary



(b) Regions

Figure 3. Effect of No-good Learning for Vickrey Pricing.

initial problems (the winner determination phase) are also solved more quickly whilst they are learning and exploiting no-goods.

In the WDP, no classical no-goods are learned because there is a solution under every valid node of the search tree, (all unset bid-variables become zero). However, the valued no-goods alone significantly improve the performance. We found that 2nd-order no-good recording outperforms 1st-order and thus offers significant potential in tackling larger problems. However, there is a limit to the order of no-good that is worthwhile given the worst-case complexity of no-good recording.

4.5. ADVANTAGES OF CP APPROACH

One major benefit of CP over OR algorithms is its simplicity and straightforward implementation. Its compact program structure permits more flexibility when different input constraints are required and offers easier problem maintenance. OR algorithms are often difficult to implement efficiently and rely on specialized data structures for good performance; [21] describes a specialized data structure for the Hungarian Method, for example.

Solutions to all the pricing sub-problems in the GVA must be optimal, thus precluding non-deterministic search algorithms. Simulated-annealing and genetic algorithms are both hill-climbing algorithms with random components that allow them to escape from local maxima, but they are not guaranteed to find the optimal solution. A systematic search is necessary in order to guarantee optimality. [2] and [8] provide evidence that constraints offer a promising alternative when more than one solution to a matching problem is required.

CP can offer other benefits to auctions such as incremental solving in cases where perturbations occur that render a solution infeasible. Perturbations for combinatorial auctions may include

- Bid Retraction;
- Bid Addition;
- Change of valuation on a bid;
- Change in set of desired items in a bid.

These perturbations can be represented as variable and constraint removals or additions, and are relevant to areas such as Supply Chain Management. Walsh, Wellman *et al.* have proposed using combinatorial auctions for the formation of efficient supply chains [27]. No-good learning can offer performance enhancements when such unpredictable changes to the problem definition are envisaged. The problem may become relaxed rather than restricted however, so it is necessary to record the justifications also (see Section 4.1) so that when a constraint is removed irrelevant no-goods are discarded. Recall that a Vickrey pricing problem is always a restricted version of the original problem so justifications are not necessary.

5. Conclusions

Ensuring truthfulness amongst self-interested agents bidding against one another in an auction can be computationally expensive when payments are determined using the Vickrey-Clarke-Groves (VCG) mechanism. VCG guarantees that each agent's dominant strategy is to tell the truth, but it requires solving $n + 1$ optimization problems where the overall optimal solution comprises n agents.

This paper first examined a case-study example demonstrating how Operations Research techniques can be used to compute Vickrey prices efficiently. In particular, the case-study focuses on the Assignment Problem. We showed how Vickrey prices can be computed in the same asymptotic time complexity as that of the original optimization problem. This case-study can be seen as serving a pedagogical role in the paper illustrating how Operations Research techniques can be used for fast Vickrey pricing.

We have proposed a Constraint Programming (CP) approach that can be used in a more general context, where nothing is assumed about the nature of the constraints that must be satisfied or the structure of the underlying problem. In an auction scenario, no assumptions are made about the structure of the auction. In particular, we have demonstrated how no-good learning can be used to improve the efficiency of Vickrey pricing in Combinatorial Auctions. In contrast, specialized state-of-the-art OR techniques are ideal for solving specific types of problems quickly, but are not robust to changes in the structure of the problem.

Finally, while not considered here, we believe that CP algorithms can work in harmony with OR algorithms to minimize the punitive computational burden of ensuring truthfulness. Using propagation and inference techniques, CP can be combined with OR to aid search when the conditions for solutions are complicated by additional constraints. This hybridization of techniques for fast Vickrey pricing is just one aspect of the very exciting research agenda that this area has to offer.

Acknowledgments

The research has received support from Enterprise Ireland through their Research Innovation Fund (Grant Number RIF-2001-317).

References

1. Bikhchandani, S. and de Vries, S. and Schummer, J. and Vohra, R. and : 2002, 'Linear Programming and Vickrey Auctions'. In: *Mathematics of the Internet: E-Auction and Markets*. pp. 75–115.
2. Caseau, Y. and F. Laburthe: 1997, 'Solving Various Weighted Matching Problems with Constraints'. In: *Principles and Practice of Constraint Programming*. pp. 17–31.
3. P. Dago and G. Verfaillie. Nogood Recording for Valued Constraint Satisfaction Problems. In *Proceedings of the 8th International Conference on Tools with Artificial Intelligence (ICTAI '96)*, pages 132–139, Toulouse, 1996.
4. R. Dechter. Enhancement schemes for constraint processing : Backjumping, learning and cutset decomposition. *Artificial Intelligence*, 41(2):273–312, 1990.
5. R. Dechter. 2003. *Constraint Processing*. Morgan Kaufmann.
6. Dijkstra, E. W.: 1959, 'A note on two problems in connexion with graphs'. *Numerische Mathematik* **1**, 269–271.
7. Edmonds, J. and R. M. Karp: 1972, 'Theoretical Improvements in algorithmic efficiency for network flow problems'. *Journal of the Association of Computing Machinery* **19**(2), 248–264.
8. Focacci, F., A. Lodi, and M. Milano: 1999, 'Integration of CP and OR methods for matching problems'. In: *CP-AI-OR'99 Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimization Problems*.
9. Foster, I.: 2001, 'The Anatomy of the Grid: Enabling Scalable Virtual Organizations'. *Lecture Notes in Computer Science* **2150**.
10. Fredman, M. L. and R. E. Tarjan: 1984, 'Network flow and testing graph connectivity'. In: *25th FOCS*. pp. 338–346.
11. Gondran, M. and M. Minoux: 1984, *Graphs et Algorithmes*. Wiley, English edition. Translated by Steven Vajda.
12. Hershberger, J. and S. Suri: 2001, 'Vickrey Pricing in Network Routing: Fast Payment Computation'. In: *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*. pp. 252–259.
13. Hung, C.-N., L.-H. Hsu, and T.-Y. Sung: 1993, 'The Most Vital Edges in a bipartite graph'. *Networks* **23**(4), 309–313.
14. Knuth, D. E.: 1993, *The Stanford GraphBase*. Addison Wesley.
15. Kuhn, H. W. 1955, The Hungarian Method for the Assignment Problem. *Naval Research Quarterly* **2**, 83.
16. Kumar, V. 1992. Algorithms for constraint-satisfaction problems : A survey. *AI Magazine* 13(1):32–44.
17. Leyton-Brown, K. and Pearson, M. and Shoham, M.: 2000, 'Towards a universal test suite for combinatorial auction algorithms', In *ACM Conference on Electronic Commerce*, pp. 66–76.
18. Mas-Collel, A., M. D. Whinston, and J. R. Green: 1995, *Microeconomic Theory*. Oxford University Press.
19. Nisan, N. and A. Ronen: 1999, 'Algorithmic Mechanism Design'. In: *Proceedings of the 31st Annual ACM Symposium on Theory Computing*.
20. Nisan, N. and A. Ronen: 2000, 'Computationally feasible VCG mechanisms'. In: *ACM Conference on Electronic Commerce*. pp. 242–252.
21. Papdimitriou, C. H. and K. Steiglitz: 1982, *Combinatorial Optimisation*. Prentice-Hall.

22. Parkes, D. C.: 2001, *Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency*. PhD Thesis, Department of Computer and Information Science, University of Pennsylvania, May 2001.
23. Radian, A. S., T. M. Denley, and R. Haggkvist: 1998, *Bipartite Graphs and their Applications*. Cambridge University Press.
24. Sandholm, T.: 2002, 'Algorithm for optimal winner determination in combinatorial auctions', *Artificial Intelligence* **135**(1-2), 1–54.
25. Schiex, T. and G. Verfaillie: 1994, 'Nogood Recording for Static and Dynamic Constraint Satisfaction Problem'. *International Journal on Artificial Intelligence Tools (IJAIT)* **3**(2), 187–207.
26. Vickrey, W.: 1961, 'Counterspeculation, auctions, and competitive sealed tenders'. *Journal of Finance* pp. 8–37.
27. Walsh, W. and Wellman, M. and Ygge, F. : 2000, 'Combinatorial Auctions for Supply Chain Formulation'. *ACM Conference on Electronic Commerce*.