Applying Metaheuristic Techniques to Search the Space of Bidding Strategies in Combinatorial Auctions

Ashish Sureka and Peter R. Wurman
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-7535
asureka@unity.ncsu.edu, wurman@ncsu.edu

ABSTRACT

Many non-cooperative settings that could potentially be studied using game theory are characterized by having very large strategy spaces and payoffs that are costly to compute. Best response dynamics is a method of searching for pure-strategy equilibria in games that is attractive for its simplicity and scalability (relative to more analytical approaches). However, when the cost of determining the outcome of a particular set of joint strategies is high, it is impractical to compute the payoffs of all possible responses to the other players actions. Thus, we study metaheuristic approaches–genetic algorithms and tabu search in particular–to explore the strategy space. We configure the parameters of metaheuristics to adapt to the problem of finding the best response strategy and present how it can be helpful in finding Nash equilibria of combinatorial auctions which is an important solution concept in game theory.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and problem complexity; J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*

General Terms

Algorithms, Economics

Keywords

Game Theory, Combinatorial Auctions, Genetic Algorithms, Tabu Search

1. INTRODUCTION

Metaheuristic search algorithms have been applied successfully to a number of optimization problems in many engineering disciplines [7, 18]. In this paper, we apply metaheuristics to a complex combinatorial optimization problem in the field of game theory and combinatorial auctions. We provide only a short introduction to the application domain; interested readers can consult the literature cited in this paper for further information on game theory and combinatorial auctions. We present a formal description of the optimization problem and explain the need for applying metaheuristics. We apply tabu search and genetic algorithms to search the very large space of bidding strategies in combinatorial auctions to find a *good* strategy in a *reasonable* amount of time. We perform simulations on a variety of problem types and tune various tabu search and genetic algorithm parameters. In the following sections, we present the design and experimental results of the algorithm's performance.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'05, June 25–29, 2005, Washington, DC, USA.
Copyright 2005 ACM 1-59593-010-8/05/0006 ...$5.00.

	c_1	c_2	c_3	c_4	c_5
r_1	6, 8	4, 12	4, 18	4, 13	8, 4
r_2	6, 12	6, 18	6, 4	8, 6	6, 6
r_3	4, 8	8, 6	21, 14	12, 20	4, 12
r_4	2, 2	14, 5	5, 6	12, 8	18, 20
r_5	8, 4	12, 6	8, 8	16, 18	14, 14

Table 1: An example game matrix.

1.1 Game Theory Background

Game theory is the study of decision making in situations in which the outcome depends on the interaction of two or more persons who have opposed, or at best mixed, incentives [5, 6, 15]. A game in *normal form* describes the actions (strategies) available to each player and the payoffs each player will receive from each possible joint action set. For small, two agent games, we usually draw the normal form game as a matrix. An example game is shown in Table 1.

The objective of a game theoretic analysis of an interaction is to predict the behavior of the participants. The most widely-used solution concept in game theory is attributed to John Nash [14] who proved the existence of equilibrium in all finite, normal-form games. A Nash equilibrium describes a steady state of the play of a strategic game in which each player's chosen action is a best response to the other players' chosen actions. That is, a joint set of strategies is a Nash equilibrium if it has the property that no player can benefit by changing her strategy while the other players keep their strategies unchanged. Nash's result holds for the general case only if we allow for *mixed strategies*, which allow participants to play strategies with random elements. In this paper, we focus on only pure strategies, that is, ones in which the participant plays a particular strategy with probability 1.0. The game shown in Table 1 has two pure-strategy Nash equilibria: $\{r_5, c_4\}$ and $\{r_4, c_5\}$.

Computing Nash equilibria of a game is a hard problem and several algorithms have been proposed over the years to solve it [12]. The current state-of-the-art algorithms for computing Nash equilibria are the Lemke-Howson algorithm [20] for two-player games, the Govindan-Wilson algorithm [9] and an algorithm based on simplical subdivision [21] for n-player, finite games. Several other algorithms for solving finite games are implemented in Gambit [11], which is a library of game theory software and tools for the construction and analysis of finite extensive and normal form games. The appropriate algorithm for computing the Nash equilibria for a game depends on a number of factors, such as whether you want to find pure strategy equilibria or mixed strategy equilibria, or whether you want to find just one equilibrium or find all the equilibria. But the underlying assumption in current algorithms is the availability of a complete payoff matrix.

In the problem domains we address in this paper, computing the payoff matrix is costly, and it is undesirable to completely enumerate the normal form matrix. Therefore we adopt a simple approach to finding equilibria in normal form games that allows us to postpone the computation of the payoffs for a majority of cells. *Best response dynamics* searches for pure-strategy Nash equilibria by iteratively examining each agent's best response to the current actions of the other agents until a joint strategy is reached from which no agent can find a better action. It is well known that best response dynamics may cycle–an issue we attempt to address in another paper [20]–and that the process may not find all equilibria unless the search is begun from every possible starting location. Despite these flaws, best response dynamics is useful when other algorithms become intractable, and easily applied to larger games when our heuristic methods of searching the strategy space are used.

To illustrate the process of using best responses to find equilibria, we again examine Table 1. Assume the search starts from the initial action profile $\{r_1, c_1\}$, which is clearly not a Nash equilibrium because each player can benefit by unilaterally deviating. Assume we analyze the column player first. The column player's best respsonse to r_1 is c_3, thus the current solution candidate becomes $\{r_1, c_3\}$. Now it is the row player's turn, and he finds r_3 is better to play when the column player is playing c_3. From $\{r_3, c_3\}$, the column player switches to $\{r_3, c_4\}$, and from there the row player switches to $\{r_5, c_4\}$. $\{r_5, c_4\}$ is a point in the space of outcomes in which both player's actions form best responses to one another, and thus is a Nash equilibrium. Note that different choices of starting locations or first agents may lead to the other Nash equilibrium.

1.2 Combinatorial Auctions

Auctions are used to allocated resources and enable dynamic pricing. Many different types of auctions are in widespread use, including the English, Dutch, Vickrey and First-Price Sealed-Bid auctions [2]. Most of these auctions involve trading one or more units of a single type of item. Many trading scenarios, such as the allocation of airport time slots [16], delivery routes, distributed scheduling, task assignment, and the allocation of radio frequencies [3, 13] involve bidders that have non-additive values for the items being allocated. For example, the value of a property to a bidder may be significantly increased if an adjacent property is won by that bidder [10]. If the items are auctioned individually, bidders who have synergies for combinations of items risk either not bidding enough to win the combination, or bidding too much on a portion of the combination only to fail to win the rest–a problem referred to in the literature as the *exposure problem* [19]. Auctions that allow participants to bid directly on combinations are called *combinatorial auctions* (CAs) and are designed to eliminate the exposure problem.

There are many variations of combinatorial auctions, including several sealed-bid and iterative designs [4]. In this paper, we restrict our attention to *proxied* CAs, that is, auctions in which bidders give some information to a proxy agent which then executes a fixed bidding policy based on the input and the trajectory of the auction. The participants strategic choice is what statement to give to her proxy agent. The bidder's statement of values is referred to as her proxy bid. Given a possible set of proxy bids, the bidders' choices can be cast as a normal form game in which the payoffs are the outcome of the proxied CA. Unfortunately, computing the payoffs for each possible joint strategy is NP-hard, and doing so for the complete matrix is exponentially worse. In an attempt to tame this complexity, we examine methods of partially searching the space of best-responses using either genetic algorithms or tabu search.

2. PROBLEM FORMULATION

Denote the set of bidders by $I = \{1, ..., m\}$ and the set of items $J = \{1, ..., n\}$. There are $2^n - 1$ possible distinct bundles of the items in J. Each bidder has a true value for each bundle, denoted $v_i(x)$. Bidder i will submit a statement, b_i, to her proxy agent that specifies a value for every bundle under consideration. Let $b_i(x)$ be i's stated value for bundle x. The strategic decision faced by each bidder is what statement to issue to her proxy bidder.[1]

Finding a best response strategy can be formulated as an optimization problem. Let agent i's surplus when playing strategy b_i be determined by the function $f_i(b_i, b_{-i})$, where b_{-i} represents the proxy bid vectors of all agents other than i. Agent i's task is to find the surplus maximizing strategy, b_i^*, given the strategies of all other agents. That is, agent i must find a bidding strategy b_i^* such that:

$$f_i(b_i^*, b_{-i}) \geq f_i(b_i, b_{-i}), \ \ \forall b_i. \tag{1}$$

In a Nash equilibrium, equation (1) is simultaneously satisfied for all agents.

We assume each bidder has a discrete, finite strategy space, S_i, that maps into the space of bids. Let $s_i^k \in S_i$ denote the kth strategy for agent i. To apply the heuristic search techniques, we encode the strategy space in a binary string using the encoding function $E : s_i^k \rightarrow b_i$. There are many potential mappings between a binary encoding and the proxy bid value. For instance, the binary string could be used to represent the actual proxy values given to the proxy agent. That is, the xth word in s_i^k is an encoding of the value of $b_i(x)$. We determined that such a direct encoding would not provide adequate resolution across the range of bundle values. Instead, we opted for an encoding that would provide the same number of expressable values regardless of the domain of the bids.

In our scheme, each word of the strategy encodes a factor that is multiplied by the bidder's true value for a bundle to get the bid value. That is,

$$b_i(x) = s_i^k(x) * v_i(x).$$

Typically, we defined $s_i^k(x) \leq 1$. Permitting $s_i^k(x)$ to be greater than one allows bidders to encode strategies in which they offer more than their true values for an object; although we have some preliminary explorations with this feature, the results are beyond the scope of this paper. The number of bits used to encode $s_i^k(x)$ is

[1]In practice, a statement does not need to specify a value for every combination; combinations for which the bidder does not represent a value are assumed to be not of interest and have value of zero.

the *resolution*, r. Because we are limiting the factor to a maximum value of one, the fraction represented by the r bits encoding the word is simply the binary value of the r bits divided by 2^r. For example if $s_i^k(x) = [10]$ then the factor is $= ([10]/[11]) = 2/3 = 0.667$.

The length of a strategy is a function of the number of items and the resolution used to encode each bundle factor:

$$l = r * 2^n.$$

All 2^l permutations of the l binary variables constitute unique strategies. The size of the strategy space increases exponentially with increases in resolution, and super-exponentially with increases in the number of items. Thus, the number of candidate solutions is very large and grows exponentially with the problem size so that simple enumeration schemes are rendered impractical.

3. EXPERIMENTAL DESIGN

Our experiments involve evaluating the applicability of tabu search and genetic algorithms as search methods over the space of possible best responses. In this section, we describe the application of each of these techniques to the combinatorial auction problem domain.

3.1 Tabu Search

Tabu search is a variation of hill-climbing search enhanced with a memory that keeps the search from backtracking into space it has already examined. Tabu search begins at a seed solution and iteratively moves from one solution to its best neighbor until a termination condition is satisfied. The algorithm keeps track of the best solution found so far, and incorporates some elements of simulated annealing that allow it to escape from local optima. These features are particularly useful in search problems with many plateaus or rugged topology. The important components of tabu search are the definition of the neighborhood function, memory, aspiration criteria, and termination criteria.

3.1.1 Neighborhood function

The neighborhood function determines which solutions are considered "next" to each other and governs which direction the search is allowed to proceed. The definition of a neighborhood function is a crucial factor in tabu search and has a strong influence on the search procedure [1, 8]. We examined two neighborhood functions.

The *toggle-bit* neighborhood is the set of solutions that differ from solution s_i^k by the value of a single bit. The number of neighbors equals to the number of bits in the solution. The *value-shift* neighborhood is the set of solutions that can be reached from s_i^k by changing the value of one of the words of s_i^k by one increment. To see the difference, consider the six bit strategy $[01, 00, 10]$. The six neighbors are $[11, 00, 10]$, $[00, 00, 10]$, $[01, 10, 10]$, $[01, 01, 10]$, $[01, 00, 00]$, $[01, 00, 11]$. In comparison, the six neighbors when using the value-shift neighborhood function are $[00, 00, 10]$, $[10, 00, 10]$, $[01, 01, 10]$, $[01, 11, 10]$, $[01, 00, 01]$, $[01, 00, 11]$. The latter method is searching the strategy space by incrementing and decrementing the words, and thus is "aware" of the context in which it is searching. In contrast, the toggle-bit method is independent of the context and is simply searching the representation space.

3.1.2 Memory mechanism

The notion of exploiting flexible memory to control the search process is the central theme underlying tabu search [17]. When designing the memory scheme for tabu search we must first decide what to remember, and then how long to remember it. The two types of memory commonly used in tabu search are *explicit memory* and *attribute-based memory*. Explicit memory records complete solutions in the tabu list. Attribute-based memory records information about solution attributes that change in moving from one solution to another. In general, attribute-based memory has a smaller memory footprint, which may allow us to keep larger tabu lists. On the other hand, by making entire attributes off limits, we may prevent searching in a promising direction. In our experiments we test both explicit memory and attributive memory.

For example, in moving from $[01, 00, 10]$ to $[01, 01, 10]$, explicit memory will record the former solution in the tabu list, thus preventing the search from immediately returning to $[01, 00, 10]$. However, it would permit a next move to $[01, 11, 10]$. Attribute-based memory might, instead, record that word two of the encoding was manipulated, and thus prevent word two from being manipulated again for the duration of the tabu tenure. This would prevent a subsequent move to $[01, 11, 10]$ until the attribute had expired from the tabu list. In our experiments, we employ a more forgiving form of attribute based memory that records both the word that was manipulated and the direction opposite the change. Thus, it would remember $\{$*word 2, decrement*$\}$ in the tabu list, and would prevent a move that involved decrementing word 2 until the attribute had expired from the tabu list.

3.1.3 Aspiration criterion

When using attribute-based memory, the aspiration criteria allows us to override the tabu restrictions when a sufficiently better neighbor is found. For example, if we find that a solution better than the best solution found so far can be reached from the current node by $\{$*word 2, decrement*$\}$, we may move in that direction even if $\{$*word 2, decrement*$\}$ is in the tabu list. In our experiments, we use aspiration criteria ranging from 80% to 100%, relative to the best solution found so far.

3.1.4 Termination criteria

Finally, we specify the termination criterion used in our experiments.

- The number of iterations is fixed to some value c_{total}.
- The number of iterations since the last improvement of the best solution is larger than a specified number $c_{\text{unimproving}}$.

3.2 Genetic Algorithm

We also employed genetic algorithms (GAs) to explore the space of best responses. The encoding for the GA was the same as in the tabu search. That is, the binary expression of a strategy is an individual in population maintained by the genetic algorithm. We used the standard GA operators, crossover and mutation, to derive a new population from the previous generation. Each member of the population was run against the current strategies of the other bidders and given a score. The population was then ranked by score, and the top third was selected to produce children in the next iteration. Two parents were randomly selected from the top third of the population to produce two complementary children, using a single crossover point. Once the subsequent generation was created, the mutation operator was applied to each bit of each individual, and with a probability of .05, switched the value of the bit.

4. RESULTS

To test the metaheuristic algorithms, we designed a set of problems and classified them by the size of the strategy space and the size of the outcome space. To illustrate the considerations that go into generating interesting problems, we present a small example in Table 2. This example has three agents and two items with the optimal allocation being to give Agent 1 item A, and Agent 3 item B.

	A	B	AB
Agent 1	25	15	45
Agent 2	20	20	45
Agent 3	10	25	40

Table 2: An example combinatorial auction problem with three agents and two items

All three agents have *complementary preferences*, that is, the valuation of a particular bundle of items is greater than the the sum of valuations of the individual items. For a fixed number of items and agents, there are several possible variations in constructing a problem by changing the valuation profile of the agents. For example, we can construct problems where the agents have sub-additive preferences and every item is allocated to a different agent. We designed the test problems by varying the number of agents, the number of items, the resolution and the valuation profiles of the agents to make sure that the results are not an artifact of a particular problem type.

The problem size grows exponentially with the number of agents, the number of items and the resolution. Table 3 is a list of 13 problem types in increasing order of the size of the outcome space. Recall that the size of the strategy space is 2^{r2^n}, and the size of the joint strategy space is $(2^{r2^n})^m$. In our model, we assume that the size of strategy space for each agent is the same.

We applied the algorithms to several problems of each problem type, beginning each run with random starting solutions, and averaged the results. The experiments were conducted on a single Apple Macintosh machine. The processor was a 1 GHz PowerPC G4 with 1024MB memory. The development environment was Macintosh Common Lisp version 5.0 on a Mac OS X version 10.2.8. Because the task of solving computing the results of the CA for a given joint strategy set was costly–for the larger problems in our test set evaluating each joint strategy took up to 10 minutes–for games larger than size 13 it was not possible to verify the quality of the solutions found by the metaheuristic search algorithms through exhaustive search of the normal form game. Thus, for this study, we limited our test set to problems we for which we could verify solutions.

Figure 1 shows the number of seconds required by both tabu search and the genetic algorithm implementations for each of the problem sizes. For the graph in Figure 1, we used an explicit-memory version of the algorithm with the toggle-bit neighborhood function. The results suggest that the absolute running time required by the genetic algorithm is significantly more than that required by tabu search. Several factors affect the total amount of time required by the program, not the least of which is the level to which each program has been optimized. However, we believe several useful observations can be made from these results. First, the search space, at least for these smallish problems, probably has structure that tabu search is able to take advantage of. With few bundles and agents, the landscape probably has few local maxima. We expect this will change with larger problems. Second, our implementation of the genetic algorithm does not reuse computations from previous generations, so if the same individual appears in a subsequent generation, it will recalculate the results of the proxy CA. We intend to measure the frequency with which this occurs, and implement code to cache pre-computed results. Interestingly, the tabu list in tabu search prevents a lot of this re-computation of previous results.

It is important to note that the actual time taken also depends on other factors, including the bid increment and the rules of the

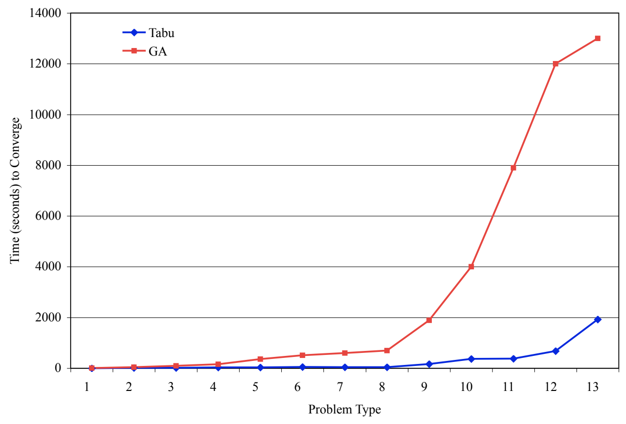

Figure 1: The time to converge to best responses as a function of problem size.

auction. Increasing the bid increment results in faster convergence, but may reduce the accuracy of the outcome.

The search for a best response is only a step in the process of using best-response dynamics to look for equilibrium. Each *round* of the best-response dynamics involves searching the space of bidding strategies for each agent while keeping the strategies of all other agents fixed. Thus, in a three agent problem, the round is complete once all three agents have had a chance to search for a best response. Best response dynamics may take several such rounds to converge to a Nash equilibrium, if it converges at all. The actual number of rounds taken to converge varies with the type of the problem, the starting solution and the auction type. Figure 2 shows the average and maximum number of rounds taken to converge to an equilibrium by tabu search using explicit memory and the toggle-bit neighborhood function.

We observe that the average and maximum number of rounds to converge to an equilibrium remains the same until problem type 8, and then increases. The largest problem to which we applied the algorithm was of type 13 (refer to Figure 3) and it takes on an average of 8.3 rounds to converge to a solution. For the results in Figure 2, we used an explicit memory version of the algorithm with a toggle-bit neighborhood function. We increased the number of rounds required to elapse before the termination was invoked as the problem size grew. The termination threshold is shown in the Figure 2. We did not observe any noticeable difference between the value-shift and toggle-bit neighborhood functions. Nor did we notice a significant difference between the explicit or the attribute-based memory variations of the algorithm. The path taken to converge to a equilibrium depends on the type of memory and neighborhood function used by the algorithm but we did not observe any consistant evidence of one type of adaptive memory or neighborhood function being better than the other. We plan to do further tests on the effect of the tabu search parameters on the search trajectory and convergence time.

The first termination criteria mentioned in Section 3.1.4 was the limit on the total number of moves that the tabu search could take, c_{total}. We carried out experiments to measure the effect of the limit on the number of iterations on the number of rounds required by

Problem Size	**Num Agents**	**Resolution**	**Num Items**	**Encoding Length**	**Strategy Space**	**Outcome Space**
1	2	1	2	3	8	10^2
2	2	2	2	6	64	$4 * 10^3$
3	2	1	3	7	128	$2 * 10^4$
4	2	3	2	9	512	$2 * 10^5$
5	3	2	2	6	64	$2 * 10^5$
6	3	3	2	9	512	$2 * 10^8$
7	2	2	3	14	$1.6 * 10^4$	$2 * 10^8$
8	2	3	3	21	$2 * 10^6$	$4 * 10^{12}$
9	3	2	3	14	$1.6 * 10^4$	$4 * 10^{12}$
10	4	2	3	14	$1.6 * 10^4$	$4 * 10^{16}$
11	3	3	3	21	$2 * 10^6$	$8 * 10^{18}$
12	4	3	3	21	$2 * 10^6$	$2 * 10^{25}$
13	4	2	4	30	10^9	10^{36}

Table 3: Classification of problem types based on the number of agents, items and resolution

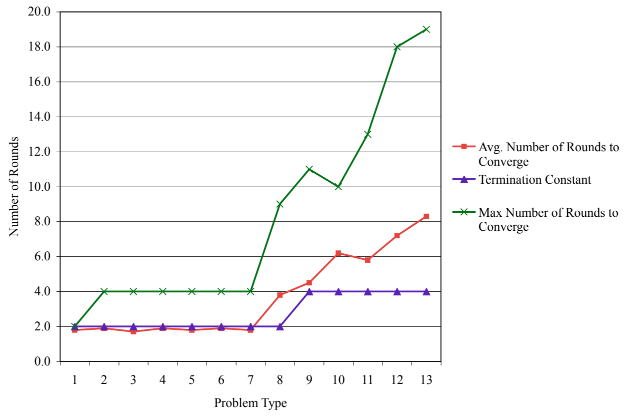

Figure 2: The number of rounds for the best-response dynamics to converge to equilibrium using tabu search to look for best responses.

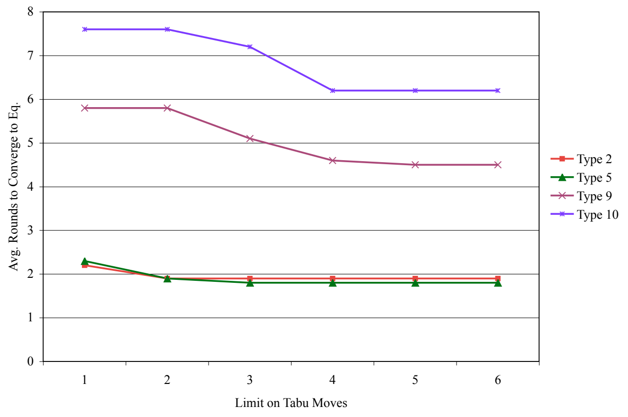

Figure 3: The time to converge to best responses as a function of problem size.

the best-response dynamics. An increase in the number of moves results in an increased coverage of the search space, but may not improve the quality of the solution found. Figure 3 shows the experimental results of changing c_{total} parameter on the number of rounds require to converge to a Nash equilibrium. The experiment was performed on problems of various sizes. We observe that for problems of type 9 and 10, setting the limit to more than 4 has no additional benefit and does not make the algorithm converge faster, whereas a value less than 4 leaves some part of the strategy space uncovered and requires more iterations to find the Nash equilibrium.

We found similar relationships among the parameters that govern the genetic algorithm. In particular, we found that a population size of 20 works well for problems of type 9 or less. For problems of size 9 to 13, we needed to increase the population size as the size of the strategy space increases. We tried a range of population sizes in multiples of 10 before settling on a size that works best with a problem of a particular size.

We verified the results obtained from tabu search using enumeration. Based on our simulation results we conclude that the quality of the solution found depends on search parameters. We performed experments to find good values of these parameters for a specific problem type. Once good parameter values were determined, we ran the tabu and genetic algorithms to find candidate equilibria. We then enumerated all strategic deviations and observed that the candidate equilibrium found were all Nash equilibria. We used this method of verification through enumeration for problems up to size 13, which as the pratical limit given our current hardware and software resources. Figures 4 and 5 show the amount of the strategy space explored by tabu search to find the best response strategy for problems of various sizes. Similar results were obtained for the genetic algorithms, though we performed less thorough testing with that algorithm because it took considerably longer to run.

The single most important observation to draw from Figures 4 and 5 is that the metaheuristic search algorithms allowed us to search a small portion of the combined strategy space to find equilibria. In the case of problem types 9 and 10, the algorithm explored

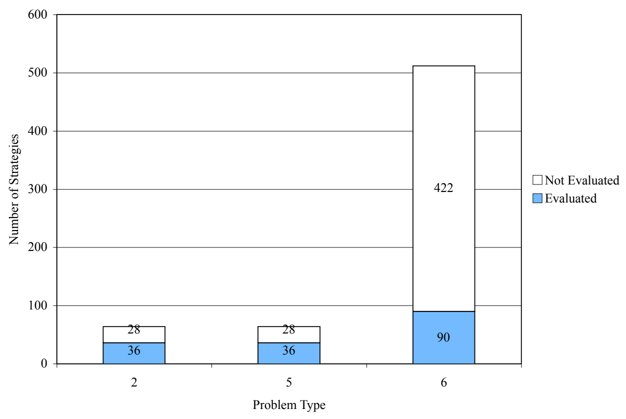

Figure 4: The number of strategies examined by tabu search for problem types 2, 3, and 5.

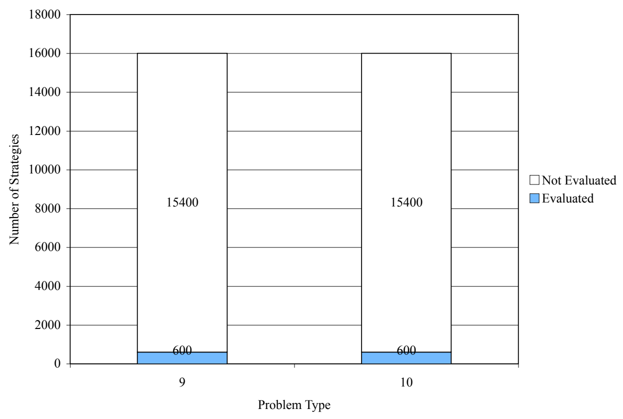

Figure 5: The number of strategies examined by tabu search for problem types 9 and 10.

only 3.75% of the strategy before finding an equilibrium. In the process of verifying that the solution found by tabu best-response dynamics with tabu search was an equilibrium, we confirmed that there are several equilibrium solutions. Thus, part of the success of the approach is due to the fact that, when more than one equilibrium is present, it takes less search to find one. However, we believe that a significant portion of the reduction in search comes from the application of the metaheuristic algorithms.

5. CONCLUSION

Metaheuristic search techniques show promise as an element of searching for equilibrium using best-response dynamics to problems in combinatorial auctions and game theory. In our experiments, tabu search and genetic algorithms proved to be good approximation techniques to find best response strategies. Our experiments show a significant reduction in the amount of the search space that needs to be explored. Naturally, the limitation of this approach is that neither tabu search nor genetic algorithms are guarantee to find the optimal solution. While we cannot prove convergence to equilibria with this technique, our experiments suggest that the technique can be used to find good bidding strategies in cases where it is computationally expensive to find an optimal bidding strategy.

Acknowledgments

This research was funded by NSF CAREER award 0092591 and the E-Commerce program NC State. The views and conclusions contained herein are those of the authors, as are any errors. We thank Gangshu Cai and Tiejun Li, and other members of the Intelligent Commerce Research Group at NCSU.

6. REFERENCES

[1] E. Aarts and J. Lenstra, editors. *Local Search in Combinatorial Optimization*. John Wiley and Sons, Chichester, UK, 1997.

[2] R. Cassady. *Auctions and Auctioneering*. Berkely University of California Press, 1967.

[3] P. Cramton. The fcc spectrum auctions: An early assessment. *Journal of Economics and Management Strategy*, 6(3):431–495, 1997.

[4] S. de Vries and R. Vohra. Combinatorial auctions: A survey. *INFORMS Journal on Computing*, June 15 to appear.

[5] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1996.

[6] R. Gibbons. *Game Theory for Applied Economists*. Princeton University Press, 1992.

[7] F. Glover, W. Kochenberger, and A. Gary, editors. *Handbook of Metaheuristics*, volume 57. 2003.

[8] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, 1997.

[9] S. Govindan and R. Wilson. A global newton method to compute nash equilibria. *Journal of Economic Theory*, 110(1), 2003.

[10] F. Kelly and R. Steinberg. A combinatorial auction with multiple winners for universal services. *Management Science*, 46(4):586–596, 2000.

[11] R. McKelvey, D. Richard, A. McLennan, M. Andrew, and T. Theodore. Gambit: Software tools for game theory. 2004.

[12] R. D. McKelvey and A. McLennan. Computation of equilibria in finite games. In H. Amman, D. A. Kendrick, and J. Rust, editors, *The Handbook of Computational Economics*, volume 1, pages 87–142. Elsevier Science, B.V., Amsterdam, 1996.

[13] J. McMillan. Selling spectrum rights. *Journal of Economic Perspectives*, 8(3):145–162, 1994.

[14] J. Nash. Two-person cooperative games. *Proceedings of the National Academy of Sciences*, 21:128–140, 1950.

[15] M. J. Osborne. *An Introduction to Game Theory*. Oxford University Press, 2004.

[16] S. J. Rassenti, V. L. Smith, and R. L. Bulfin. A combinational auction mechanism for airport time slot allocation. *Bell Journal of Economics*, 13:402–417, 1982.

[17] C. Reeves, editor. *Modern Heuristic Techniques for Combinatorial Problems*. John Wiley and Sons, Norwell, MA, 1997.

[18] M. Resende and P. deSousa, editors. *Metaheuristics Computer Decision-Making*, volume 86. 2004.

[19] M. H. Rothkopf, A. Pekeč, and R. M. Harstad. Computationally manageable combinational auctions. *Management Science*, 44(8):1131–1147, 1998.

[20] A. Sureka and P. R. Wurman. Applying tabu search for finding pure strategy nash equilibria in very large normal form games. In *Fourth International Joint Conference on Autonomous Agents and Multi Agent Systems*, Utrecht, Netherlands, 2005.

[21] G. van der Laan, A. J. J. Talman, and L. van Der Heyden. Simplicial variable dimension algorithms for solving the nonlinear complementarity problem on a product of unit simplices using a general labelling. *Mathematics of Operations Research*, pages 377–397, 1987.