Using Tabu Best-Response Search to Find Pure Strategy Nash Equilibria in Normal Form Games

Ashish Sureka   Peter R. Wurman

Department of Computer Science
North Carolina State University
Raleigh, NC 27695-7535

asureka@unity.ncsu.edu   *wurman@csc.ncsu.edu*

ABSTRACT

We present a new method for computing pure strategy Nash equilibria for a class of n-person games where it is computationally expensive to compute the payoffs of the players as a result of the joint actions. Previous algorithms to compute Nash equilibria are based on mathematical programming and analytical derivation, and require a complete payoff matrix as input. However, determining a payoff matrix can itself be computationally intensive, as is the case with combinatorial auctions. This paper proposes an approach, based on best-response dynamics and tabu search, that avoids the requirement that we have a complete payoff matrix upfront, and instead computes the payoffs only as they become relevant to the search. The tabu features help break best-response cycles, and allow the algorithm to find pure strategy Nash equilibria in multiplayer games where best-response would typically fail. We test the algorithm on several classes of standard and random games, and present empirical results that show the algorithm performs well and gives the designer control over the tradeoffs between search time and completeness.

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, Theory

Keywords

Game Theory, Tabu Search, Algorithms for computing Nash equilibria, Multi-Agent Systems

1. INTRODUCTION

Game theory is an important analytical tool used to study a wide variety of human interactions [12, 15] in which the outcomes depend on the joint strategies of two or more persons. An *equilibrium* is a stable outcome of the game. Although there are a variety of equilibrium concepts in the literature, Nash equilibrium is arguably the most important and overwhelmingly the most studied. A Nash equilibrium is a profile of strategies such that each player's strategy is an optimal response to the other players strategies [13].

Computing Nash equilibria of general normal form games is a hard problem. Several algorithms have been proposed over the years to compute equilibria (see [11] for an overview), including the Lemke-Howson algorithm [9] for two-player games, and the Govindan-Wilson algorithm [7] and an algorithm based on simplical subdivision [23] for n-player finite games. Several other algorithms for solving finite games are implemented in Gambit [10], 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, including whether you want to find pure strategy equilibria or mixed strategy equilibria, and whether you want to find just one equilibrium or all the equilibria.

One critical assumption underlying the majority of the current algorithms is that a complete payoff matrix is available as an input to the algorithm. However, determining a payoff matrix can itself be computationally intensive. For example, if it requires an hour to determine the values of each cell of the payoff matrix in a normal form game, then it will take many days to fill out the game matrix for a 2 player game in which each player has 10 actions. Despite the small size of the game, determining the Nash equilibria becomes time consuming because the cost of computing the payoffs dominates the cost of find the equilibria. Similarly, when there are large number of players and actions, the size of the file input to the algorithm can be unmanageable. For example, in a game with 6 players each with 10 actions to chose from, there are one million entries in the normal form game. Computing the payoffs and storing this type of game as a text file can be very time consuming and can easily consume many mega-bytes of disk space.

One example of the problem arises from *combinatorial auctions* (CAs), a class of auctions that allow participants to bid on combinations of items [1, 2, 16, 20, 26]. When the bidders' preferences exhibit complementarity and substitutability, combinatorial auctions can reach an economically efficient allocation of goods and services where auctioning individual items sequentially or in parallel may not. There has been a recent surge of research addressing the

	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.

	c_1	c_2	c_3	c_4	c_5
r_1	5, 10	7, 6	8, 9	9, 8	4, 12
r_2	0, 7				0, 3
r_3	1, 5		Nash Eq.		3, 6
r_4	3, 7				4, 8
r_5	3, 12	6, 7	0, 0	3, 3	9, 5

Table 2: A game with a best-response cycle.

strategic and computational aspects of CAs. Although the equilibria for some CAs, such as the Generalized Vickrey Auction, can be solved directly, others have no known closed form solution. For the latter class, we can discretize the strategy spaces and define the corresponding normal form game, but we are soon faced with the computational cost of computing the outcome of a combination of bidding strategies. It is well known that, in general, solving the *winner-determination problem* in a CA is NP-hard [2, 4, 21], and for iterative CAs, this must be done many times.

Similar issues arise in many other real-world markets and multi-agent decision making situations. One such example is the Trading Agent Competition (TAC) [25]. TAC is a test-bed developed to challenge the research community to study strategic decision making in multi-agent systems. The TAC family of games includes a supply-chain management scenario and a travel agent scenario, both of which pit several software agents against each other while trying to maximize their individual utilities. While the strategies employed in TAC are heuristic in nature, abstractions of these strategies could be formulated as a normal form game and reasoned about. However, in the TAC games, like the combinatorial auctions, computing the outcome of a a single cell in the matrix is costly: the supply chain game takes 55 minutes to run, and the travel game takes 15 minutes. In general, the complexity of specifying and solving games has limited the application of game theory to small models.

This paper proposes a novel approach to searching for pure strategy equilibria in games that marries tabu search [5, 6] with best-response dynamics. The approach sacrifices completeness to gain the ability to work with partially specified payoff matrices. Because we use best-response dynamics, the algorithm is capable of finding only *pure strategy* equilibria. Concepts from tabu search are used to combat the cycling problem that often plagues the best-response analysis. In addition, the tabu framework parameters give the user control over some of the tradeoffs that can be made in the search. Note that Nash's original proof that equilibrium in finite, normal form games always exists [13] holds only by including mixed-strategies. Thus, by limiting ourselves to looking at only pure strategies, we may not be able to solve games in which the only equilibria are mixed.

2. BACKGROUND

2.1 Best-Response Dynamics

Best-response dynamics is a means of looking for a Nash equilibrium by iteratively letting each player select the best response to their opponent's current best strategies. To see how the process works, consider the example game in 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. If we had started the search at $\{r_2, c_1\}$ and let the column player search first, she would choose $\{r_2, c_2\}$. From there, the row player would switch to $\{r_4, c_2\}$ to which the column player would respond by selecting $\{r_4, c_5\}$, thus finding the second (and dominant) equilibrium to this game.

It is important to note that standard best-response dynamics may not converge to a Nash equilibrium even if one exists. Table 2 is an example of a normal-form game where starting the best-response analysis from any of the corners creates a clockwise cycle, and fails to find the Nash equilibrium in the middle of the matrix.

2.2 Tabu Search

Tabu search is one of many gradient search techniques designed for large, combinatorial optimization problems. Tabu search was first presented by Glover [5, 6] with additional formalization reported by Hansen [8] and de Werra and Hertz [3]. Like hill climbing, tabu search is a neighborhood search technique. Beginning from a seed solution, the algorithm iteratively moves from one solution to its best neighbor until a termination condition is satisfied. The key enhancement of tabu search over hill climbing is the addition of a *tabu list*, which records a subset of the recent solutions examined and which is used to prevent the search from backtracking. This feature is particularly useful in search problems with many plateaus or rugged topology.

The memory feature is typically referred to as *adaptive memory* because the tabu list has a fixed size and old moves are removed from the list once their tabu tenure is reached. The adaptive memory typically comes in two forms: explicit memory or attribute-based memory. *Explicit memory* records complete solutions and prevents the search from returning to recently visited states. In the context of our search through game matrices, explicit memory would record the joint strategies that define a particular cell. Explicit memory is shared by all of the players.

Attribute-based memory schemes record a feature of the *operation* that was used to move from one state to another, and which cannot be repeated as long as it remains on the tabu list. For example, in a graph optimization problem, attributes can be operations on particular nodes or arcs, such as addition, deletion, or repositioning. In the context of our problem domain, attribute-based memory becomes interesting when we have three or more players. An attribute-based memory scheme would record that the row player chose r_i and thus when it is the row player's turn to respond, she will not be allowed to select r_i again.

Consider a game with three players: row, column and depth. Suppose the tabu tenure is 2 and the row player responds to $\{r_1, c_1, d_1\}$ by choosing r_2. The action r_2 is placed on the row player's tabu list. Suppose the sequence of best responses that follows is $\{r_1, c_1, d_1\} \rightarrow$

$\{r_2, c_1, d_1\} \rightarrow \{r_2, c_3, d_1\} \rightarrow \{r_2, c_3, d_4\} \rightarrow \{r_3, c_3, d_4\} \rightarrow \{r_3, c_5, d_4\} \rightarrow \{r_3, c_5, d_6\}$. Now it is the row player's turn to respond, and r_2 is still be on row's tabu list, thus the row player is blocked from choosing r_2 even if it is the best response to the choices $\{c_5, d_6\}$ made by the other players.

Attribute-based memory blocks a larger set of possible outcomes than explicit memory, which can be both beneficial and harmful. By blocking an operation instead of just a state, it can eliminate a large number of inferior solutions and force the search into new areas. The harm comes about if a better solution was in the set entail by the blocked operation. To allow the search to select a solution that may be covered by the attributes on the tabu list, an *aspiration criteria* is used. When the value of the blocked solution exceeds the aspiration criteria, the operation is permitted despite its tabu status. Note that with this attribute-based memory scheme each player maintains her own tabu list.

By allowing aspiration values below 1.0, tabu search can capture some of the behavior of simulated annealing that allows it to escape from local maxima. When a local maximum is reached that is not a Nash equilibria (it is a local maximum only because better actions for some players are currently on the tabu list), the algorithm can permit a non-increasing move if the decrease in value to the agent in question is within the tolerance. Note that, in general, when using aspiration levels below 1.0 it is necessary to remember the best solution found so far, in case we later move away from it and don't find one better. In the context of searching games, we do not need to do this because we do not need to return to solutions that are not equilibria.

3. TABU BEST RESPONSE SEARCH

3.1 Formal description

Consider an n-player game in normal form where I is the set of players. For each player $i \in I$ there is a nonempty set A_i that specifies the set of actions available to player i. The number of actions available to each player is denoted by $|A_i|$. Thus, the size of the search space is $\prod_{i \in I} |A_i|$. Let x be an action profile consisting of one action from each player. Let x_i denote the action of player i in the action profile x, and x_{-i} denote the actions of all other players in x.

For x^* to be a Nash equilibrium, it must be that no player i can profitably deviate given the actions of the other players. That is,

$$(x_i^*, x_{-i}^*) \succeq_i (x_i, x_{-i}^*) \quad \forall x_i \in A_i. \tag{1}$$

The tabu best-response search algorithm maintains a tabu list, T, with length L. In the case of attribute-based memory, each player maintains its own tabu list, T_i. We use a neighborhood function the returns all outcomes that can be reached by player i unilaterally changing her action. Player i has a method of evaluating her best response to the actions of the other agents. Let $\beta_i(A_i|x_{-i})$ be player i's best choice among its actions set, A_i, to the actions of the other agents in x. When we are using explicit memory, let $\beta_i(A_i|x_{-i}, T)$ be player i's best response excluding actions that lead to solutions already in T. When using attribute-based memory, the best-response function is $\beta_i(A_i|x_{-i}, T_i)$. For simplicity, we assume that $\beta_i(A_i|x_{-i}, T_i)$ is sensitive to the aspiration criteria and will admit actions in T_i if they improve on the best solution.

Figure 3.1 shows the pseudocode for the algorithm when explicit memory is used. When attribute-based memory is used, simply replace the T's with T_i's. One termination condition is that a Nash equilibrium is found. It is also necessary to be able to terminate without finding an equilibrium if, for instance, an upper limit on the number of iterations is reached.

```
Initialize T to empty
Choose random starting solution, x
Until termination criteria is satisfied {
        i ← next player
        x ← {β_i(A_i|x_{-i}, T), x_{-i}}
        remove oldest item in T if |T| = L
        push x onto T
}
```

Figure 1: Pseudocode for tabu best-response with explicit memory.

	c_1	c_2	c_3	c_4
r_1	9, 5	3, 3	2, 5	4, 8
r_2	6, 4	8, 8	3, 0	5, 3
r_3	2, 2	2, 1	3, 2	4, 6
r_4	4, 4	2, 0	2, 2	9, 3

Table 3: Another example game.

3.2 Example

To illustrate the algorithm, we present the example game in Table 3. A standard best-response approach, when started from any of the corners, would result in a cycle, and would be unable to find the Nash equilibrium at $\{r_2, c_2\}$. Table 4 shows the five steps required by our algorithm to find the Nash equilibrium using explicit memory and a tabu tenure of 3. The critical step occurs at $t = 3$, where the row player choose to deviate to r_2 because the solution $\{r_1, c_1\}$ is in the tabu list. This breaks the best-response cycle and allows the algorithm to find the equilibrium.

4. EXPERIMENTAL RESULTS

To test our algorithm we generated large normal form games using GAMUT, a suite of game generators designated for benchmarking game-theoretic algorithms [14]. The GAMUT library includes thirty-five base games from which a large number of well-known game structures can be generated with variable numbers of players and equilibria. We used different parameter settings in GAMUT to generate both well known games and random games in normal form which were fed into our tabu best-response algorithm.

Our overall objective is to speed up the process of finding Nash equilibria, but rather than measure time directly, we chose two measures that are proportional to run time.

1. **Percent of solution space explored:** Because our approach is motivated by games in which computing the payoffs for a set of joint strategies is expensive, the most important metric for our algorithm is the percentage of the search space that

t	x	Tabu List	Best Response
0	$\{r_1, c_1\}$	$\emptyset$	$c_1 \rightarrow c_4$
1	$\{r_1, c_4\}$	$\{r_1, c_1\}$	$r_1 \rightarrow r_4$
2	$\{r_4, c_4\}$	$\{r_1, c_1\}, \{r_1, c_4\}$	$c_4 \rightarrow c_1$
3	$\{r_4, c_1\}$	$\{r_1, c_1\}, \{r_1, c_4\}, \{r_4, c_4\}$	$r_4 \rightarrow r_2$
4	$\{r_2, c_1\}$	$\{r_1, c_4\}, \{r_4, c_4\}, \{r_2, c_4\}$	$c_1 \rightarrow c_2$
5	$\{r_2, c_2\}$	termination condition satisfied	

Table 4: The five steps of applying the tabu best-response to the example in Table 3.
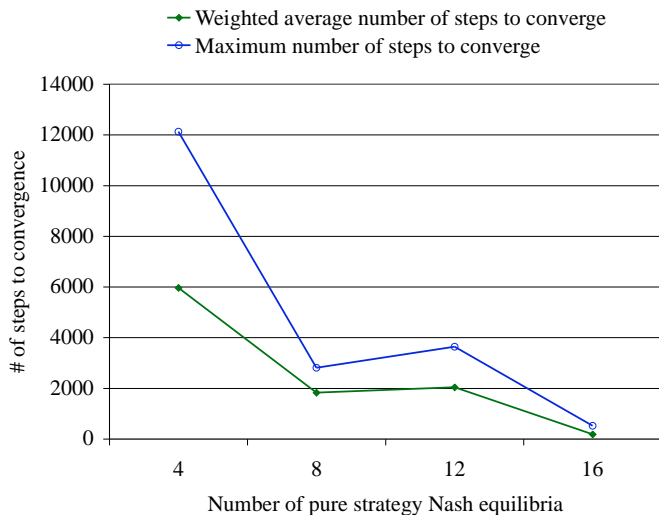
Figure 2: The maximum and average number of steps to converge in random games with explicit memory, as a function of the number of pure strategy Nash equilibria in the game.
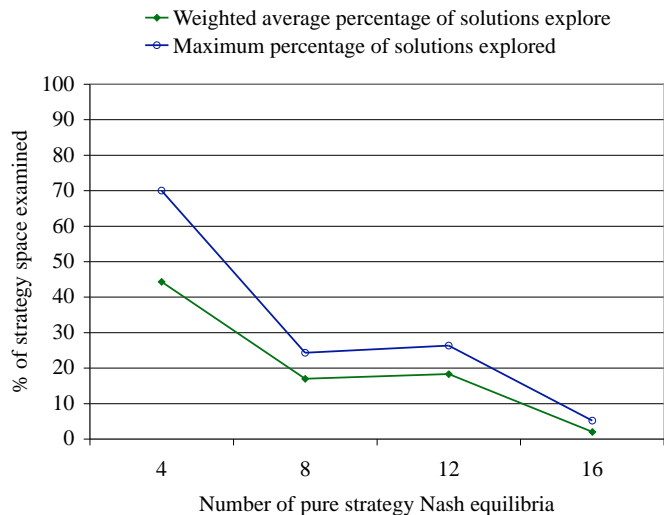

Figure 3: The maximum and average percent of the joint strategy space explored with explicit memory, as a function of the number of pure strategy Nash equilibria in the game.

must be explored in order to converge to a Nash equilibrium. Note that this measure includes not just the candidate solutions, but the solutions accessible by each agent's possible deviations. Thus, even if we seed the algorithm with a Nash equilibrium, it must examine a certain number of cells in order to confirm the equilibrium. If there are n players and each player i has $|A_i|$ possible actions, then the minimum value of the percentage solution space explored is

$$\frac{1 + \sum_{i \in I}(|A_i| - 1)}{\prod_{i \in I} |A_i|}$$

2. **Number of steps to converge:** The number of steps is incremented each time an agent changes its best response during the search process. The minimum number of steps to reach equilibrium is zero, and occurs where the seed solution itself is a Nash equilibrium.

Our first set of experiments involved random games generated by GAMUT. Each game had 5 players and 10 actions each, which means the searchable solution space has 10^5 cells. Keeping the size of the game constant, we generated games with varied numbers of Nash equilibria, creating several games of each type. We ran the algorithm once from every possible initial seed, and tracked the average number of steps to converge and the average amount of the solution space explored for each game. We also include the maximum values recorded.

Figures 2 and 3 show the average and maximum number of steps, and the average and maximum percentage of the space examined to find an equilibrium when using explicit memory as a function of the number of equilibria in the game. Note that, as these are random games–they have no inherent structure–and the number of equilibria is a very small percentage of the searchable space (4 to 16 equilibria in 100,000 cells). Not surprisingly, when there are more equilibria in the game, it takes fewer steps to find them and less of the solution space is explored. It is somewhat surprising how sensitive this relationship is. With four equilibria in the space,

nearly 45% of the space is examined before one is found. However, with 16 equilibria, only 2% of the space needs to be explored.

Tables 4 and 5 show the same types of results when the algorithm uses attribute-based memory. The shapes of the curves are the same as for explicit memory, although attribute-based memory required approximately five times as many iterations to find an equilibrium. On the other hand, the attribute-based approach examined slightly less of the space to be examined, on average. The worst case results, however, are somewhat discouraging for attribute-based memory; for at least one run, when only four equilibria were present, the attribute-based memory had to fill in 98% of the matrix to find an equilibrium.

Our procedure for evaluating the algorithm–starting at every possible seed and running until a solution was found–allowed us to collect statistics on each equilibrium in the game. Table 5 shows statistics for a five-player, ten-action random game with four equilibria. The equilibria are labeled by a concatenation of the strategies (0–9) in each player position. The statistics were recorded for a particular game solved using explicit memory. Solution 64598 was the most likely to be found, accounting for more than 96% of the terminations. In contrast, solution 91904 was found in only 0.16% of the searches. Neither the number of steps required to converge nor the percentage of the space explored seem sensitive to the frequency with which a solution is found; this suggests that the Nash equilibria don't have nicely demarcated basins of attraction, but instead the landscape is much more knotted.

The last two lines of Table 5 show counters kept during the search process that measure the impact of the tabu features on the best response search. The fact that the tabu condition is enforced typically hundreds of times is evidence that the feature is blocking cycles in the best-response dynamics. Further, by running with the aspiration criteria set below 1.0, we allowed the algorithm to take non-improving moves which helped it move into new territory when a best response was on the tabu list.

Table 6 shows similar data collected for a random game with five players, ten actions, and six equilibria. Solution 49660 was by far the most frequently found, and its statistics appear to be about

	Nash Equilibria			
	36473	**64598**	**80337**	**91904**
Number of times found	2550	96240	1050	160
Average number of steps	8484	5915	5764	2790
Maximum number of steps	11477	12137	10116	7726
Average percentage of solutions explored (%)	55.85	44.08	41.81	21.91
Maximum percentage of solutions explored (%)	68.17	69.99	63.38	53.56
Average Number of tabu conditions	817.4	398.4	433	154.2
Average Number of non-improving moves	103.9	54.13	59.66	22.06

Table 5: Simulation results of running the explicit memory version of the algorithm on a 5-Player 10-Action random game consisting of four Nash equilibria.
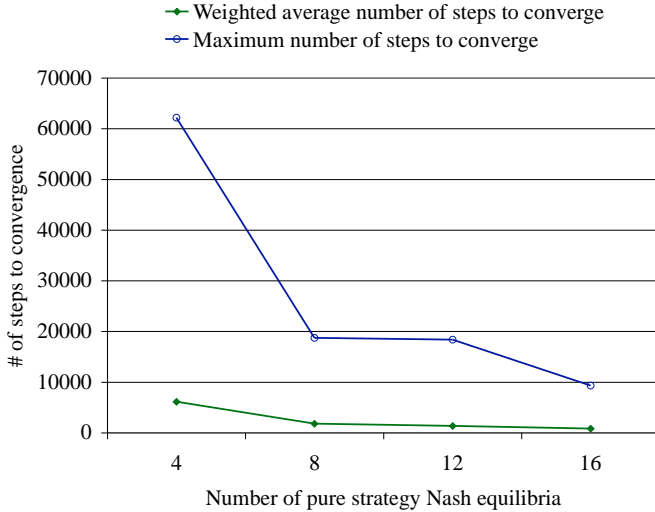

Figure 4: The maximum and average number of steps to converge in random games with attribute-based memory, as a function of the number of pure strategy Nash equilibria in the game.
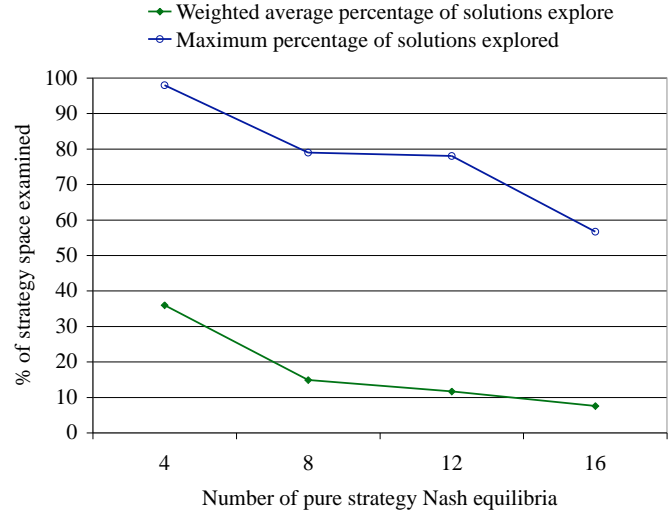

Figure 5: The maximum and average percent of the joint strategy space explored with explicit memory, as a function of the number of pure strategy Nash equilibria in the game.

half that of the popular solution in Table 5. This problem also had two very infrequently found solutions (02835 and 42559) with very small basins of attraction. It is a topic for future study whether the mechanism that GAMUT uses to create random games produces ones with best-response attractors, or whether it is a by-product of our search method.

After studying performance on the random games, we ran our algorithm on several games from the GAMUT library. We tried four different input sizes of the Traveler's Dilemma, Minimum Effort and Covariant games. Note that these games, unlike the random ones used above, have inherent structure that may make finding equilibria easier. Figure 6 shows the average percentage of the strategy space examined by the algorithm. Like the tests above, the experiment involved starting the search from each point in the search space. Overall, the results are very encouraging; the algorithm explored less than 2% of the strategy space in all of the tests. The Traveler's Dilemma and Minimum Effort games, in particular, required very little exploration to solve. The covariance game, which Porter, et al. [17] cite as being particularly difficult to solve, took relatively longer.[1]

---

[1]In their case, they are looking for mixed strategies.

Although this work was inspired by the combinatorial auction applications, we have only preliminary analysis of our algorithm on that application domain, largely because of the computational costs of doing so. We have several other research efforts underway in an attempt to reduce the costs of searching for equilibria in combinatorial auctions. In addition to the applications of tabu search in this paper, we have looked at metaheuristic approaches to searching for an individual player's best response [22] so that not all possible actions are studied. We have also developed algorithms that can more quickly compute exact outcomes of proxied combinatorial auctions [27], which is necessary to fill in the payoffs of the matrix for evaluation that the former two methods choose to inspect.

Our initial simulations have applied the tabu best-response technique to two iterative combinatorial auctions: the Ascending Package Auction[1] and the Ascending k-Bundle Auction [26]. The preliminary results show that the average percentage of the solution space explored varies from 13% to 39%.

5. RELATED WORK

As game theoretic concepts have been adopted by the artificial intelligence community to model multi-agent interactions, there

	Nash Equilibria					
	02835	**19745**	**36200**	**42559**	**46903**	**49660**
Number of times found	30	1210	740	40	830	95270
Average number of steps	1.57	1114	1067	2.15	1033	2085
Maximum number of steps	2	2861	3065	4	2729	3649
Average percentage of solutions explored (%)	0.05	9.86	9.61	0.06	9.8	18.7
Maximum percentage of solutions explored (%)	0.06	25.03	26.34	0.08	23.93	3.07
Average Number of tabu conditions	0	35.32	27.32	0.0	18.73	58.93
Average Number of non-improving moves	0	4.44	3.36	0.0	1.25	5.81

Table 6: Simulation results of running the explicit memory version of the algorithm on a 5-Player 10-Action random game consisting of six Nash equilibria
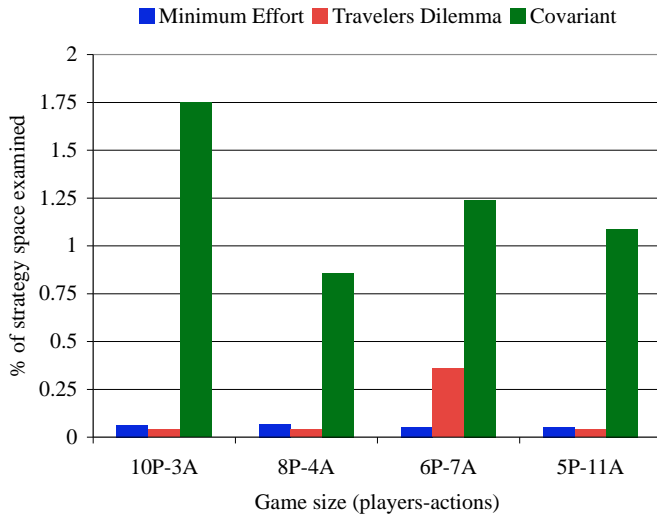

Figure 6: Effects of game type and size on the percentage of the strategy space explored to find a Nash equilibrium.

has been a recent surge of interest in algorithms to compute equilibria. For example, Reeves, et al. [18] explore equilibrium in a complex scheduling scenario using replicator dynamics. By assigning populations to strategies, they are able to run simulations that evolve mixed strategies. In their experiments, however, they are able to apply the technique to problems with only a small numbers of players and strategies before being overwhelmed by the computation. More recently, Reeves and Wellman [19] have developed a method for computing equilibrium in infinite games (games with a continuous strategy space) that has very attractive computational properties.

Porter, et al. [17] propose an algorithm capable of finding mixed strategy equilibria by searching over the *support* for various strategy profiles. They measured runtimes for games of similar size to ours, although a direct comparison is difficult because we measured iterations rather than run times in seconds.[2] More importantly, the critical difference between the two approaches is that our method works on incomplete game matrices, while Porter's method, like most other methods, requires the complete game matrix.

---

[2]Comparing run times between experiments done with different computing hardware and operating systems can easily be misleading. For instance, Porter, et al.'s experiments were run on a cluster of 12 high-end linux machines, while ours were run on a medium-range iMac.

An alternative to computing the Nash equilibrium of a complete, complex game is to to eliminate much of the strategy space and instead restrict attention to a subset of strategies. This approach is adopted by Walsh, et al. [24] to study equilibrium in auctions and other complex negotiation settings. Like us, they are motivated by complex games in which the cost of computing the outcome of a strategy profile is high. Walsh, et al.'s key contribution is the application of information theory to the task of selecting which strategy profile to evaluate next. Their approach differs from ours in several ways, most importantly in that it assumes a symmetric and structured strategy space. It seems likely that aspects of the two techniques may complement each other, which will allow us to come up with more subtle approaches.

6. CONCLUSION

We looked at the relatively unexplored area of finding pure strategy Nash equilibria using search techniques which are applicable to situations where the size of the strategy space is very large and where it is computationally expensive to compute the payoffs. Our method is capable of filling in the strategy space as the search proceeds, and employs features from tabu search to prevent best-response cycling. It is difficult to offer theoretical guarantees about the performance of this type of algorithm as there is a large performance variation depending on the class of game and the number of Nash equilibria that exists in the game. However, our experimental results are encouraging and suggest that the technique significantly reduces the number of cells that need to be evaluated, particularly in games with structure.

7. REFERENCES

[1] L. M. Ausubel and P. R. Milgrom. Ascending auctions with package bidding. *Frontiers of Theoretical Economics*, 1(1):1–42, 2002.

[2] S. de Vries and R. Vohra. Combinatorial auctions: A survey. *INFORMS Journal on Computing*, 15(3):284–309, 2003.

[3] D. de Werra and A. Hertz. Tabu search techniques: A tutorial and an application to neural networks. *OR Spektrum*, 11:131–141, 1989.

[4] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Sixteenth International Joint Conference on Artificial Intelligence*, pages 548–553, 1999.

[5] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13:533–549, 1986.

[6] F. Glover. Tabu search: Part I. *ORSA Journal on Computing*, 1:190–206, 1989.

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

[8] P. Hansen. The steepest ascent mildest descent heuristic for combinatorial programming. In *Talk presented at the Congress on Numerical Methods in Combinatorial Optimization*, 1986.

[9] C. E. Lemke and J. T. Howson. Equilibrium points of bimatrix games. *Journal of the Society of Industrial and Applied Mathematics*, 12:413–423, 1964.

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

[11] 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.

[12] R. B. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, Cambridge, MA, 1991.

[13] J. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 21:128–140, 1950.

[14] E. Nudelman, J. Wortman, K. Leyton-Brown, and Y. Shoham. Run the GAMUT: A comprehensive approach to evaluating game-theoretic algorithms. In *Third International Joint Conference on Autonomous Agents and Multi-Agent Systems 2004*, pages 880–887, 2004.

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

[16] D. C. Parkes and L. H. Ungar. Iterative combinatorial auctions: Theory and practice. In *Seventeenth National Conference on Artificial Intelligence*, pages 74–81, 2000.

[17] R. Porter, E. Nudelman, and Y. Shoham. Simple search methods for finding a Nash equilibrium. In *AAAI-04: Nineteenth National Conference on Artificial Intelligence*, pages 664–669, 2004.

[18] D. Reeves, M. P. Wellman, J. K. MacKie-Mason, and A. Osepayshvili. Exploring bidding strategies for market-based scheduling. *Decision Support Systems*, to appear.

[19] D. M. Reeves and M. P. Wellman. Computing best-response strategies in infinite games of incomplete information. In *Twentieth Conference on Uncertainty in Artificial Intelligence*, page 470478, 2004.

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

[21] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Cabob: A fast optimal algorithm for combinatorial auctions. In *International Joint Conference on Artificial Intelligence*, Seattle, WA, 2001.

[22] A. Sureka and P. R. Wurman. Applying metaheuristic techniques to search the space of bidding strategies in combinatorial auctions. *Genetic and Evolutionary Computation Conference (GECCO-05)*, to appear.

[23] G. van der Laan and A. J. J. Talman. Adjustment processes for finding economic equilibria. In A. J. J. Talman and G. van der Laan, editors, *The Computation and Modeling of Economic Equilibria*, pages 85–123. Elsevier Science Publishers, B. V. North Holland, 1987.

[24] W. E. Walsh, D. C. Parkes, and R. Das. Choosing samples to compute heuristic-strategy Nash equilibrium. In *AAMAS 2003 Workshop on Agent Mediated Electronic Commerce*, Melbourne, Australia, 2003.

[25] M. P. Wellman, P. R. Wurman, K. A. O'Malley, R. Bangera, S.-D. Lin, D. Reeves, and W. E. Walsh. Designing the market game for a trading agent competition. *IEEE Internet Computing*, 5(2):43–51, Mar.-Apr 2001.

[26] P. R. Wurman and M. P. Wellman. A*k*BA: A progressive, anonymous-price combinatorial auction. In *Second ACM Conference on Electronic Commerce*, pages 21–29, 2000.

[27] P. R. Wurman, J. Zhong, and G. Cai. Computing price trajectories in combinatorial auctions with proxy bidding. *Electronic Commerce Research and Applications*, 3(4):329–340, 2004.