OREGON STATE UNIVERSITY

COMPUTATIONAL COMPLEXITY (THEORY OF COMPUTATION) - CS 517

Final Project (Draft)

Professor:
Mike ROSULEK

Student:
Ben MCCAMISH

TA:
Naimisha SAIREDDY

1   Introduction

Most users do not know the structure and content of databases and concepts such as schema or formal query languages sufficiently well to express their information needs precisely in the form of queries [18, 6, 17]. They may convey their intents in easy-to-use but inherently ambiguous forms, such as keyword queries, which are open to numerous interpretations. Thus, it is very challenging for a database management system (DBMS) to understand and satisfy the intents behind these queries. The fundamental challenge in the interaction of these users and DBMS is that the users and DBMS represent intents in different forms.

Many such users may explore a database to find answers for various intents over a rather long period of time. For these users, database querying is an inherently interactive and continuing process. As both the user and DBMS have the same goal of the user receiving her desired information, the user and DBMS would like to gradually improve their understandings of each other and reach a *common language of representing intents* over the course of various queries and interactions. The user may learn more about the structure and content of the database and how to express intents as she submits queries and observes the returned results. Also, the DBMS may learn more about how the user expresses her intents by leveraging user feedback on the returned results. The user feedback may include clicking on the relevant answers [28], the amount of time the user spends on reading the results [12], user's eye movements [16], or the signals sent in touch-based devises [19]. Ideally, the user and DBMS should establish as quickly as possible this common representation of intents in which the DBMS accurately understands all or most user's queries.

Researchers have developed systems that leverage user feedback to help the DBMS understand the intent behind ill-specified and vague queries more precisely [5, 4]. These systems, however, generally assume that a user does *not* modify her method of expressing intents throughout her interaction with the DBMS. For example, they maintain that the user picks queries to express an intent according to a fixed probability distribution. It is known that the learning methods that are useful in a static setting do not deliver desired outcomes in a setting where all agents may modify their strategies [13, 9]. Hence, one may not be able to use current techniques to help the DBMS understand the users' information need in a rather long-term interaction.

2   Framework

We have modeled this interaction between the user and the DBMS as an interactive game. The user will first start by selecting some intent. Then they choose a query to represent that intent and send it to the DBMS. The DBMS, upon receiving this query, will determine what tuple it should return to the user.

2.1   User Strategy

The user begins their interaction by considering what intent to communicate to the DBMS. An intent is the information that the user is seeking from the DBMS. However, the user may not know how to express this intent in such a way that the DBMS can understand what they are seeking. A user unfamiliar with the DBMS structure, content, or query language, may use keyword queries to interact with a DBMS. For example, if the user was searching a database of colleges, they might be searching for "Oregon State University". To express this intent, they may send the keyword query "OSU". Keyword queries can be translated into some structured query language such as SQL. However, for this project we will be referring to queries as keyword queries. For the sake of this project the specific query language does not matter beyond examples.

Thus, the user has a strategy that maps their intents, what they are searching for, to the set of possible queries they can use to express that intent. This mapping is a row stochastic mapping. The user can learn and update their strategy overtime using common reinforcement learning algorithm techniques [25, 8, 2, 3]. Thus, overtime the user can learn to communicate effectively with the DBMS their intents. This may take quite a long amount of time, but it eventually will happen. In practice, a user has finitely many intents and submits finitely many queries in a finite period of time. Hence, we assume that the sets of the user's intents and queries are finite. We index each user's intent and query by $1 \leq i \leq m$ and $1 \leq j \leq n$, respectively. A user strategy, denoted as U, is a $m \times n$ row-stochastic matrix from her intents to her queries.

2.2   DBMS Strategy

The DBMS receives keyword queries from the user and must decide how to respond with tuples. A naive approach would be to simply return all the tuples that have the keyword(s) received ranked using some popular ranking algorithm such as

BM25 [24]. This requires that the user knows some keywords that might possibly return results. This also relies on the ranking algorithm to effectively return the correct tuple high enough in the results such that the user is able to view it.

In our model we take another approach. The DBMS has a strategy that given a keyword query, it responds with some probability with a tuple. This can trivially be extended for the case of returning multiple tuples by either sampling multiple times or returning tuple sets. For now, we will focus on the case where the DBMS only returns a single tuple. The DBMS can also learn and update its strategy overtime using the same reinforcement learning algorithms that the user might use. It can also use other popular algorithms such as UCB-1 [1, 27, 23, 21]. The DBMS strategy is represented for a user as an $n \times o$ row-stochastic matrix from the set of the user's queries to the set of possible tuples to return. We index each query and tuple by $1 \le j \le n$ and $1 \le l \le o$. Each pair of the user and the DBMS strategy, (U, D), is a *strategy profile*. The expected payoff for both players with strategy profile (U, D) is as follows.

$$u_r(U, D) = \sum_{i=1}^{m} \pi_i \sum_{j=1}^{n} U_{ij} \sum_{\ell=1}^{o} D_{jl} \ r, \tag{1}$$

2.3 Overview

The overall view is the user considers an intent to search the DBMS for and picks a query to send, based on their strategy U. The DBMS receives that query and returns some tuple, based on its strategy D. In Equation 1 π is the prior probability that a user will search for some intent i. The reward r then becomes whether the tuple received by the user matches their intent. Of course, more complicated reward metrics may be used such as Precision, Recall, NDCG, or MRR [14, 20]. However, for this project we will simply restrict it to the case of "did the user receive the tuple or not", thus a boolean reward of either 0 or 1. The user may later decide to query for another intent with a different signal. The DBMS might later decide to return a different tuple upon receiving the same query. However, both of the players in this game share the payoff since the goal of both the user and the DBMS is the same, to return a tuple that matches the user's intent.

2.4 Equilibria

Our game model uses one similar to the one presented in Game Theory, called a Signaling Game [7, 15]. This game is founded on the idea that one agent sends signals to another and that agent decides how it should respond to that signal. Each agent can adjust their strategies overtime and eventually reach a point where they are communicating to some degree effectively. In fact, they are communicating to such a degree that any change to the current strategy by either player would reduce the current payoff they are receiving, or a "stable state". Payoff in this case is shown in Equation 1. The range for the payoff values are [0,1].

This "stable state" is called a Nash equilibria. A Nash equilibria is defined as a state in which neither player can unilaterally deviate from their current strategy and receive a higher payoff. Another definition is for a Strict Nash equilibria in which neither player can unilaterally deviate from their current strategy and receive the *same* or higher payoff. There are also instances of either *pure* or *mixed* Nash equilibria. A pure Nash equilibria means that each strategy is deterministic in which actions are taken given an observation of the other players action. A mixed strategy has some probability associated with whether a given action will take place [22].

It has been proved that there is always a mixed Nash equilibria in a game [22]. However, there may not be a pure Strict Nash equilibria. In general, finding a mixed Nash equilibria is not NP-Complete, but instead in a special class called PPAD-Complete [10]. With certain restrictions to the problem, however, it has been shown to be NP-Complete [11, 26]. One such property is when the game is played in such a way that the agents share their payoff. This is the case for our model of the game.

3 Reduction

We will be reducing and creating three main contributions, of which the reductions are showing Sections 3.4, 3.6, and 3.7. The other portions of Section 3 are required to ensure that the solution falls within the bounds of our game. You might say they contain the supporting clauses for the SMT.

3.1 Simplifications

We will be making some simplifications to our model. We are restricting the strategies to only Pure Strategies, thus only probabilities of 0 or 1 in strategies. After our discussion and emailing the student you put me in contact with, we couldn't

find a way to construct a bounded quantifier for the reductions. SMT is also limited in that all variables in the formula must have the same domain. Since we are using a uniform prior over the intents we are left with fractions in Equation 1. Thus all variables in the following reductions are Real numbers.

We will be instantiating some of the reductions for clarity. For these instantiations, we will use the strategy profile and reward matrix in Figure 1, 2, and 3. We will also be using notation such that the variable $U_{apple,x}$ is the variable for that cell in the user strategy. In addition, we will be showing each part of the reduction separately. The final SMT will of course be a conjunction between the individual portions, but it should be clearer by breaking them up into separate parts.

Table 1: User strategy

	x	y
"CS"	0	1
"EE"	1	0

Table 2: DBMS strategy

	"CS517"	"EE533"
x	0	1
y	1	0

Table 3: Reward Matrix

	"CS517"	"EE533"
"CS"	1	0
"EE"	0	1

3.2 Domains

The first clause in the reduction will be that of the domains for each variable. Since the strategies we are looking at are Pure strategies, then the variables can only take on values 0 and 1. Due to some of the limitations of SMT, each of the variables are real numbers. Thus, to enforce this restriction, we have the following SMT:

$$(U_{1,1} = 0 \vee U_{1,1} = 1) \wedge (U_{1,2} = 0 \vee U_{1,2} = 1) \wedge \ldots \wedge (U_{m,n} = 0 \vee U_{m,n} = 1) \tag{2}$$
$$\wedge (D_{1,1} = 0 \vee D_{1,1} = 1) \wedge (D_{1,2} = 0 \vee D_{1,2} = 1) \wedge \ldots \wedge (D_{n,o} = 0 \vee D_{n,o} = 1)$$

As a reminder, n is the number of intents, m is the number of queries, and o is the number of tuples to return. This makes sure that any variables that would be assigned a value in an accepting equilibria will have either a 0 or 1 value. We won't show an example for this, as it takes up quite a bit of space.

3.3 Stochastic

With only the domain reduction, it would be possible to get strategies with 1s in every cell. However, our model uses row stochastic matrices. The SMT to ensure that each strategy is row stochastic is as follows:

$$(U_{1,1} + U_{1,2} + \ldots + U_{1,m} = 1) \wedge (U_{2,1} + U_{2,2} + \ldots + U_{2,m} = 1) \wedge \ldots \wedge (U_{n,1} + U_{n,2} + \ldots + U_{n,m} = 1) \tag{3}$$
$$\wedge (D_{1,1} + D_{1,2} + \ldots + D_{1,o} = 1) \wedge (D_{2,1} + D_{2,2} + \ldots + D_{2,o} = 1) \wedge \ldots \wedge (D_{n,1} + D_{n,2} + \ldots + D_{n,o} = 1)$$

Using the example in Section 3.1, we would have the following:

$$(U_{CS,x} + U_{CS,y} = 1) \wedge (U_{EE,x} + U_{EE,y} = 1) \wedge (D_{x,CS517} + D_{x,EE433} = 1) \wedge (D_{y,CS517} + D_{y,EE433} = 1) \tag{4}$$

Since the sum of all rows now is equal to 1, an assignment of variables to our strategies will be stochastic.

3.4 Payoff

One portion of this project is to see if there is a strategy that can achieve some minimum amount of payoff. The following reduction is how we can ensure that any assignments to the variables must have a minimum payoff using Equation 1. Not all sets of intents, queries, and tuples will be able to achieve all payoffs, thus depending on the setting and the minimum required payoff, no such assignment may exist.

$$(\pi_1 U_{1,1} D_{1,1} R_{1,1} + \pi_1 U_{1,1} D_{1,2} R_{1,2} + \ldots + \pi_2 U_{2,1} D_{1,1} R_{2,1} + \ldots + \pi_m U_{m,n} D_{n,o} R_{m,o}) \geq MinPayoff \tag{5}$$

Of course, this could easily be changed such that we want a payoff of exactly some value, then we could change the inequality to be that of an equals. Using the example in Section 3.1, we would have the following:

$$\begin{aligned}
&\pi_{CS}U_{CS,x}D_{x,CS517}R_{CS,CS517} + \pi_{CS}U_{CS,x}D_{x,EE433}R_{CS,EE433} + \\
&\pi_{CS}U_{CS,y}D_{y,CS517}R_{CS,CS517} + \pi_{CS}U_{CS,y}D_{y,EE433}R_{CS,EE433} + \\
&\pi_{EE}U_{EE,x}D_{x,CS517}R_{EE,CS517} + \pi_{EE}U_{EE,x}D_{x,EE433}R_{EE,EE433} + \\
&\pi_{EE}U_{EE,y}D_{y,CS517}R_{EE,CS517} + \pi_{EE}U_{EE,y}D_{y,EE433}R_{EE,EE433} \\
=&0.5 \cdot 0 \cdot 0 \cdot 1 + 0.5 \cdot 0 \cdot 1 \cdot 0 + 0.5 \cdot 1 \cdot 1 \cdot 1 + 0.5 \cdot 1 \cdot 0 \cdot 0 + \\
&0.5 \cdot 1 \cdot 0 \cdot 0 + 0.5 \cdot 1 \cdot 1 \cdot 1 + 0.5 \cdot 0 \cdot 1 \cdot 0 + 0.5 \cdot 0 \cdot 0 \cdot 1 \\
=&1
\end{aligned} \tag{6}$$

We can see that the current strategy we have in the example is the highest payoff possible in this strategy. However, we by making some changes we can see the payoff change. For example if the DBMS instead returned "CS517" for both queries x and y, the payoff would then be 0.5.

3.5   Nash Equilibria

In order for the assignment of variables to be in either a Strict Nash or Nash equilibria, we need to add to the SMT some inequalities that capture unilateral deviations from the current strategy. A unilateral deviation is where one player changes their strategy while the other player's remains the same. For instance, using the example in Section 3.1, a unilateral deviation would be if the user instead sent query y for "CS" instead of x and the DBMS strategy remained the same.

To perform this reduction, we need a strategy for every single cell in both strategies. For the current example that would be 8 strategies. We will refer to these as *Nash Strategies*. A Nash Strategy is slightly different from the regular strategy. It duplicates the regular strategies variables in every way, including the domain and stochastic restrictions with one exception. It introduces new variables for a single row. The row must remain stochastic and each new variable in that row will have the same domain restrictions. One of these new variables will also have the added restriction that it cannot equal the current strategy. Thus we have the SMT as follows, where $NU_{x,y,j,k}$ is the Nash strategy for the user for the x row and y column. j and k are the cells within that strategy. The same is true for the DBMS side with $ND_{x,y,j,k}$. We denote NU and ND with separate notation indicating which player is unilaterally deviating from their strategy to clarify the reduction. However, it should be noted that all reductions described here apply to either NU or ND.

$$\begin{aligned}
&(NU_{1,1,1,1} \neq U_{1,1}) \wedge (NU_{1,2,1,2} \neq U_{1,2}) \wedge \ldots \wedge (NU_{n,m,n,m} \neq U_{n,m}) \\
&\wedge (ND_{1,1,1,1} \neq D_{1,1}) \wedge (ND_{1,2,1,2} \neq D_{1,2}) \wedge \ldots \wedge (ND_{m,o,m,o} \neq D_{m,o}))
\end{aligned} \tag{7}$$

The notation for this is confusing, so let's look at the example strategy in Section 3.1 again. The current user strategy variables are shown in Table 4. The variables for the Nash User Strategy for the first row and first column are in Table 5. The variables for the Nash User Strategy for the first row and second column are in Table 6.

Table 4: User strategy

$U_{CS,x}$	$U_{CS,y}$
$U_{EE,x}$	$U_{EE,y}$

Table 5: Nash User strategy 1,1

$NU_{1,1,CS,x}$	$NU_{1,1,CS,y}$
$U_{EE,x}$	$U_{EE,y}$

Table 6: Nash User strategy 1,2

$NU_{1,2,CS,x}$	$NU_{1,2,CS,y}$
$U_{EE,x}$	$U_{EE,y}$

Using the SMT above and the two Nash Strategies in Tables 5 and 6, we would have the following:

$$(NU_{1,2,CS,y} \neq U_{CS,y}) \wedge (NU_{1,1,CS,x} \neq U_{CS,x}) \tag{8}$$

Notice that we do not have any restrictions on $NU_{1,2,CS,x}$ and $NU_{1,1,CS,y}$. If we had a strategy that had three queries for example, then if we put restrictions on those variables it would be impossible to find a satisfying assignment as 2 of the variables would have to be 0. At least one of those two would already be 0 in the User strategy, since the strategies are row stochastic. It is also enough to simply say that the Nash User Strategy cell is not equal to the User Strategy cell since the only values they can possess are 0 or 1. Thus, all possible moves from the current strategy will represented by these new variables.

To get an idea of how many new variables we are introducing, consider again the example. We have 8 new Nash Strategies, each of which are introducing new variables for every column. Thus we have $8 \cdot 2 = 16$ new variables, where only 8 of those variables have the above restriction. All 16 have the domain restriction. All 8 of the strategies also have the stochastic restriction. Both of these follow the same reduction as done previously. Consider the Nash Strategy variables in Table 5 again. We would add to the SMT the following:

$$(NU_{1,1,CS,x} + NU_{1,1,CS,y} = 1) \wedge (NU_{1,1,CS,x} = 0 \vee NU_{1,1,CS,x} = 1) \wedge (NU_{1,1,CS,y} = 0 \vee NU_{1,1,CS,y} = 1) \tag{9}$$

Each of these new Nash Strategies also has some payoff, which is calculated the same as the normal one. Consider again the Nash User Strategy in Table 5. The payoff function for this would be as follows:

$$\begin{aligned}
&\pi_{CS} NU_{1,1,CS,x} D_{x,CS517} R_{CS,CS517} + \pi_{CS} NU_{1,1,CS,x} D_{x,EE433} R_{CS,EE433} + \\
&\pi_{CS} NU_{1,1,CS,y} D_{y,CS517} R_{CS,CS517} + \pi_{CS} NU_{1,1,CS,y} D_{y,EE433} R_{CS,EE433} + \\
&\pi_{EE} U_{EE,x} D_{x,CS517} R_{EE,CS517} + \pi_{EE} U_{EE,x} D_{x,EE433} R_{EE,EE433} + \\
&\pi_{EE} U_{EE,y} D_{y,CS517} R_{EE,CS517} + \pi_{EE} U_{EE,y} D_{y,EE433} R_{EE,EE433}
\end{aligned} \tag{10}$$

Note that only the two new variables are introduced into the payoff. This is because we only want to check the payoff after a single move. That is to say, we want to know the payoff after changing a row such that is not equal to the original row.

3.6 Pure Strict Nash Equilibria Reduction

For the reduction to an SMT that will produce a Strict Nash Equilibria if one exists, we take all the Nash Strategies created in Section 3.5 and make sure that their payoff is less than the original strategy payoff. To simplify the notation we will be representing the Nash User Strategy Payoff for the cell x, y, as $NUP_{x,y}$. One example of this formula is in Equation 10. The DBMS movements will be similar with notation $NDP_{x,y}$. The original payoff will be represented as P. The reduction would then be as follows:

$$(NUP_{1,1} < P) \wedge (NUP_{1,2} < P) \wedge \ldots \wedge (NUP_{n,m} < P) \tag{11}$$
$$\wedge (NDP_{1,1} < P) \wedge (NDP_{1,2} < P) \wedge \ldots \wedge (NDP_{m,o} < P) \tag{12}$$

Since this is a Strict Nash equilibria, each of the moves that any player can make must have a strictly smaller payoff then the current one.

3.7 Pure Nash Equilibria Reduction

A Nash equilibria will always exist. However, a Strict Nash equilibria may not. Reduction for a Nash Equilibria is very similar. Instead we adjust the inequality to instead allow for strategies that have the same payoff as follows:

$$(NUP_{1,1} \leq P) \wedge (NUP_{1,2} \leq P) \wedge \ldots \wedge (NUP_{n,m} \leq P) \tag{13}$$
$$\wedge (NDP_{1,1} \leq P) \wedge (NDP_{1,2} \leq P) \wedge \ldots \wedge (NDP_{m,o} \leq P) \tag{14}$$
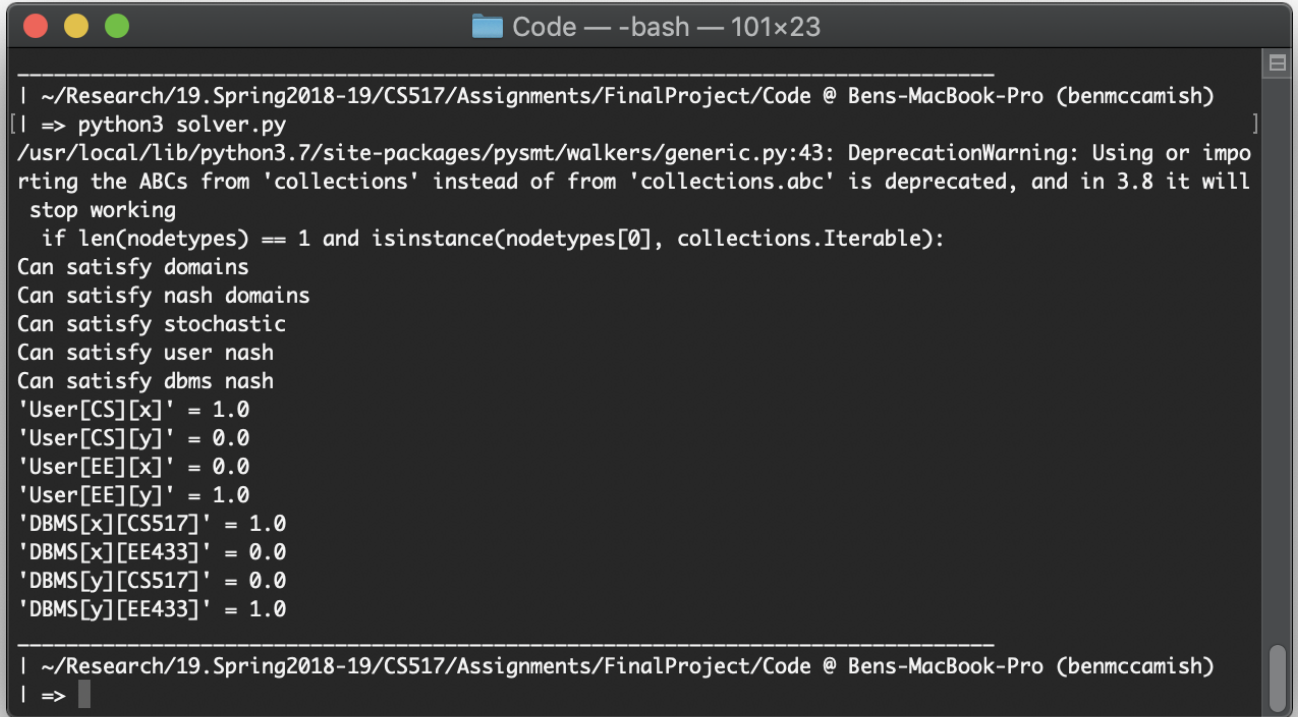
4 Conclusion

This section contains some of my results from running the SMT we encoded using PySMT. We only included two different runs, but there are more setup and ready to be run in the code if you desire. The reason why there are only 2, is because we noticed how long the paper was getting and wanted to keep to shorter. My apologies if it is still quite long. **Note:** The screenshots include some deprecation warnings, since apparently PySMT is using some outdated code. Make sure you run this in Python 3.7.

4.1 Strict Nash Example From Paper

Here we will test the example from the paper. The settings are as follows:

- **Intents:** "CS", "EE"
- **Queries:** "x", "y"
- **Tuples:** "CS517", "EE433"

We can see the output in Figure 4.1. We have materialized this into a strategy for easier viewing in Tables 7,8, and 9.

```
----------------------------------------------------------------------
| ~/Research/19.Spring2018-19/CS517/Assignments/FinalProject/Code @ Bens-MacBook-Pro (benmccamish)
[| => python3 solver.py                                                ]
/usr/local/lib/python3.7/site-packages/pysmt/walkers/generic.py:43: DeprecationWarning: Using or impo
rting the ABCs from 'collections' instead of from 'collections.abc' is deprecated, and in 3.8 it will
 stop working
  if len(nodetypes) == 1 and isinstance(nodetypes[0], collections.Iterable):
Can satisfy domains
Can satisfy nash domains
Can satisfy stochastic
Can satisfy user nash
Can satisfy dbms nash
'User[CS][x]' = 1.0
'User[CS][y]' = 0.0
'User[EE][x]' = 0.0
'User[EE][y]' = 1.0
'DBMS[x][CS517]' = 1.0
'DBMS[x][EE433]' = 0.0
'DBMS[y][CS517]' = 0.0
'DBMS[y][EE433]' = 1.0
----------------------------------------------------------------------
| ~/Research/19.Spring2018-19/CS517/Assignments/FinalProject/Code @ Bens-MacBook-Pro (benmccamish)
| => 
```

Figure 1: Strict Nash Solution

We can see that any deviation from this strategy will lead to a lower payoff. However, notice that the assignments are opposite of the one used in the example in the paper. This of course has the same payoff. The strategies can only reach that one by making two moves, where the first one will receive a lower payoff. This is the main point behind Strict Nash equilibria, as there is no incentive for either player to change their strategy since any move gets them a smaller payoff. There may be other Strict Nash equilibria that exist, however the players will not reach them from this strategy.

Table 7: User strategy

	x	y
"CS"	1	0
"EE"	0	1

Table 8: DBMS strategy

	"CS517"	"EE533"
x	1	0
y	0	1

Table 9: Reward Matrix

	"CS517"	"EE533"
"CS"	1	0
"EE"	0	1

4.2   Nash Example From Paper

Keeping the current settings lets look at the output when we test on just a Nash equilibria. We can see the output in Figure 4.2. We have materialized this into a strategy again for easier viewing in Tables 10,11, and 12.


```
| ~/Research/19.Spring2018-19/CS517/Assignments/FinalProject/Code @ Bens-MacBook-Pro (benmccamish)
[| => python3 solver.py
/usr/local/lib/python3.7/site-packages/pysmt/walkers/generic.py:43: DeprecationWarning: Using or impo
rting the ABCs from 'collections' instead of from 'collections.abc' is deprecated, and in 3.8 it will
 stop working
  if len(nodetypes) == 1 and isinstance(nodetypes[0], collections.Iterable):
Can satisfy domains
Can satisfy nash domains
Can satisfy stochastic
Can satisfy user nash
Can satisfy dbms nash
'User[CS][x]' = 0.0
'User[CS][y]' = 1.0
'User[EE][x]' = 0.0
'User[EE][y]' = 1.0
'DBMS[x][CS517]' = 1.0
'DBMS[x][EE433]' = 0.0
'DBMS[y][CS517]' = 1.0
'DBMS[y][EE433]' = 0.0

| ~/Research/19.Spring2018-19/CS517/Assignments/FinalProject/Code @ Bens-MacBook-Pro (benmccamish)
| => 
```

Figure 2: Strict Nash Solution

This is a good example of a Nash equilibria. It is obviously not the best payoff we know we can receive from this setting. However, any single move that either player makes will result in a lesser or equal payoff. For example, no matter what query the user sends for "EE", it will always receive "CS517". If the DBMS decides to return "EE433" for the query "y", then the EE intent is satisfied, but now the CS intent is not satisfied. Again, a better strategy could be met by making multiple moves, but there is no incentive for the user or the DBMS to do this, as all the payoffs are the same or less.

Table 10: User strategy

	x	y
"CS"	0	1
"EE"	0	1

Table 11: DBMS strategy

	"CS517"	"EE533"
x	1	0
y	1	0

Table 12: Reward Matrix

	"CS517"	"EE533"
"CS"	1	0
"EE"	0	1

Running the encoding of the SMT in Python was fairly fast for small strategies. As soon as I increased the size of intents, queries, and tuples to more than a couple dozen, it takes an incredibly long time. However, PySMT uses Cython so I was able to get that installed and working which I believe improved running times.

References

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002. 2

[2] R. R. Bush and F. Mosteller. A stochastic model with applications to learning. *The Annals of Mathematical Statistics*, pages 559–585, 1953. 1

[3] Y. Cen, L. Gan, and C. Bai. Reinforcement learning in information searching. *Information Research: An International Electronic Journal*, 18(1), 2013. 1

[4] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis. Query recommendations for interactive database exploration. In *Proceedings of the 21st International Conference on Scientific and Statistical Database Management*, SSDBM 2009, pages 3–18, Berlin, Heidelberg, 2009. Springer-Verlag. 1

[5] S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum. Probabilistic information retrieval approach for ranking of database query results. *TODS*, 31(3), 2006. 1

[6] Y. Chen, W. Wang, Z. Liu, and X. Lin. Keyword search on structured and semi-structured data. In *SIGMOD*, 2009. 1

[7] I. Cho and D. Kreps. Signaling games and stable equilibria. *Quarterly Journal of Economics*, 102, 1987. 2

[8] J. G. Cross. A stochastic learning model of economic behavior. *The Quarterly Journal of Economics*, 87(2):239–266, 1973. 1

[9] C. Daskalakis, R. Frongillo, C. H. Papadimitriou, G. Pierrakos, and G. Valiant. On learning algorithms for nash equilibria. In *Proceedings of the Third International Conference on Algorithmic Game Theory*, SAGT'10, pages 114–125, Berlin, Heidelberg, 2010. Springer-Verlag. 1

[10] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009. 2

[11] I. Gilboa and E. Zemel. Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior*, 1(1):80–93, 1989. 2

[12] L. A. Granka, T. Joachims, and G. Gay. Eye-tracking analysis of user behavior in www search. In *SIGIR*, 2004. 1

[13] A. Grotov and M. de Rijke. Online learning to rank for information retrieval: Sigir 2016 tutorial. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 1215–1218, New York, NY, USA, 2016. ACM. 1

[14] V. Hristidis, L. Gravano, and Y. Papakonstantinou. Efficient IR-Style Keyword Search over Relational Databases. In *VLDB 2003*. 2

[15] Y. Hu, B. Skyrms, and P. Tarrès. Reinforcement learning in signaling game. *arXiv preprint arXiv:1103.5818*, 2011. 2

[16] J. Huang, R. White, and G. Buscher. User see, user point: Gaze and cursor alignment in web search. In *CHI*, 2012. 1

[17] S. Idreos, O. Papaemmanouil, and S. Chaudhuri. Overview of data exploration techniques. In *SIGMOD*, 2015. 1

[18] H. V. Jagadish, A. Chapman, A. Elkiss, M. Jayapandian, Y. Li, A. Nandi, and C. Yu. Making database systems usable. In *SIGMOD*, 2007. 1

[19] E. Liarou and S. Idreos. dbtouch in action database kernels for touch-based data exploration. In *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, pages 1262–1265, 2014. 1

[20] C. Manning, P. Raghavan, and H. Schutze. *An Introduction to Information Retrieval*. Cambridge University Press, 2008. 2

[21] T. Moon, W. Chu, L. Li, Z. Zheng, and Y. Chang. An online learning framework for refining recency search results with user click feedback. *ACM Transactions on Information Systems (TOIS)*, 30(4):20, 2012. 2

[22] J. Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951. 2

[23] F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th international conference on Machine learning*, pages 784–791. ACM, 2008. 2

[24] S. Robertson, H. Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009. 2

[25] A. E. Roth and I. Erev. Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term. *Games and economic behavior*, 8(1):164–212, 1995. 1

[26] V. C. T. Sandholm. Complexity results about nash equilibria. 2002. 2

[27] A. Vorobev, D. Lefortier, G. Gusev, and P. Serdyukov. Gathering additional feedback on search results by multi-armed bandits with respect to production ranking. In *WWW*, pages 1177–1187. International World Wide Web Conferences Steering Committee, 2015. 2

[28] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The k-armed dueling bandits problem. *J. Comput. Syst. Sci.*, 78(5), 2012. 1