

Finding Skyline and Top- k Bargaining Solutions

Mohamed A. Soliman
School of Computer Science
University of Waterloo
m2ali@cs.uwaterloo.ca

Ihab F. Ilyas
School of Computer Science
University of Waterloo
ilyas@uwaterloo.ca

Nick Koudas
Department of Computer Science
University of Toronto
koudas@cs.toronto.edu

Abstract

We address skyline and top- k processing in web interaction scenarios. We model the problem space based on game theory principles and present new algorithms and heuristics to realize solutions efficiently.

1 Introduction

Skyline (pareto-optimal) objects over d dimensions is the set of data objects that are not dominated by any other objects restricted to those dimensions. On the other hand, top- k objects are the k objects with the highest scores according to some scoring function. Skyline and top- k queries are gaining increasing importance in several domains, such as OLAP, and Web databases. Current techniques assume deterministic settings where scores are exact. However, in real world we often deal with various sources of uncertainty. Factoring uncertainty in skyline and top- k processing is lacking in current approaches.

In this paper we initiate work in the direction of skyline and top- k computation in web bargaining scenarios, where interactions between involved parties have uncertain outcomes (scores). Interaction is based on the existence of an underlying platform to manage preferences and privacy settings. For example in Platform for Privacy Preferences (P3P) protocol [2], one could declare information one is willing to reveal, e.g., an email address. In a similar fashion a business can declare types of acceptable contacts, payments, etc. Custom pieces of software, e.g., Privacy Bird [2], can check such specifications. When preference settings have conflicts, *negotiation* could be used to resolve such conflicts and enable transaction to proceed.

Negotiation is based on an underlying *utility function* that interacting parties aim to increase. Figure 1(a) is a possible *negotiation profile* of an e-shopping site showing negotiable entities (Bonus and Delivery Schedule), and utilities against different user actions. Figure 1(b) is the corresponding user profile. The numeric utility values could express monetary profit, gained popularity, satisfaction level, etc. The symbol ('-') refers to a pair of non-negotiable en-

tities. Profiles are *matched* before interaction commences to form the bi-matrices in Figure 1(c) which represent two potential interactions. The utility pair (5, 30), for example, means that site gains a utility of 5 if it gives a discount in exchange of user's email, while a gains of 30 in utility is obtained if one reveals an email address as a result of obtaining a discount. An *arbitration procedure* is needed to resolve each interaction (bi-matrix) into some *mutually beneficial* outcome.

In large scale web transactions, it is infeasible to resolve every potential interaction due to complex negotiation mechanisms and time or budget constraints. Moreover, the interaction outcomes might redefine the interaction space. For example, resolving some interaction might yield an outcome that makes other interactions non-profitable or restricts their beneficial outcome space.

We envision this problem as a *data management problem* of objects with uncertain scores. Arbitration procedures that resolve interaction scores, i.e., reduce the uncertainty, are considered as *expensive predicates* whose invocation needs to be minimized. We study how to assign "importance" scores to interactions, and how to realize skyline or top- k bargaining solutions while evaluating the minimum number of interactions. We summarize our contributions as follows:

- We define *scored join results* in the context of web bargaining. We use game theory principles to model join results as *cooperative games*.
- We address skyline and top- k computation in web bargaining context, and show that interaction makes traditional algorithms prohibitively expensive.
- We conduct an extensive experimental study to show the effectiveness of our techniques in different settings.

2 Problem Definition

We model interactions, e.g., the bi-matrices in Figure 1(c), as 2-player *cooperative games* [4, 3], where *solving* a game corresponds to obtaining an interaction outcome. Figure 2 depicts an example game. A game is approximated

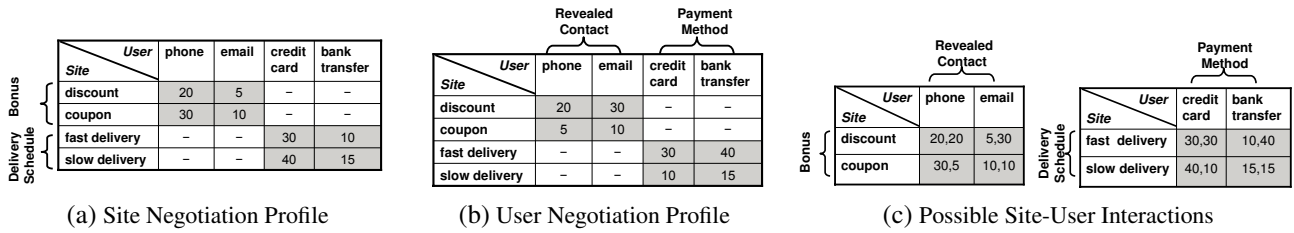


Figure 1. Two Negotiation Profiles and Possible Generated Interactions.

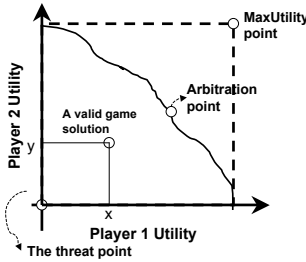


Figure 2. Example game.

as a rectangle containing an *arbitration point* (game solution), and is enclosed between the *MaxUtility point* and the *threat point*. We omit theoretical discussion of how these points are derived for space constraints, and refer the reader to our extended version of this paper [6] for details.

In the remainder of this paper, we consider games as *rectangles*, where the *threat point* and *MaxUtility point* are the lower-left and upper-right corners of the rectangle, respectively. Game rectangle encloses game *cooperative payoff region*, which is the region of interesting utility pairs. Arbitration pair always belongs to the pareto-frontier of the cooperative payoff region [4]. We emphasize that obtaining the arbitration pair of each game is an expensive computational task with complexity proportional to the game area.

We view the interaction between two negotiation profiles (one from each party) as a “join” process that results in a set of “cooperative games”. We assume an intermediate entity that matches negotiation profiles and forms a set of corresponding games as in Figure 1. Game solutions define the scores of join results. Our main goal is to find the set of games with maximum or non-dominated scores. More formally, given a collection of N games let S be a set of rectangles obtained by deriving the *threat point* and the *MaxUtility point* of each game, the goal is to find the pareto-optimal set (skyline) of arbitration pairs in S . For each arbitration pair, (u_i^*, v_i^*) , in the pareto-optimal set, there exists no arbitration pair (u_j^*, v_j^*) with $u_j^* > u_i^*$ and $v_j^* > v_i^*$. We aim to minimize the *total number of solved games* to identify the pareto-optimal set. There are two considerations affecting this approach:

- *Game Dominance*: Solving a game might make other games uninteresting. Let (u_i^*, v_i^*) denote the arbitra-

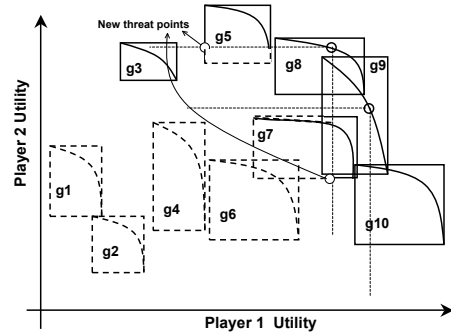


Figure 3. Example game set.

tion pair of game i . Game i dominates game j if (u_i^*, v_i^*) dominates the *MaxUtility point* of game j . Dominated games are uninteresting since their arbitration point is not pareto-optimal. Consider Figure 3. Games $g1$, $g2$, and $g4$ can be pruned before processing any game, since their *MaxUtility points* are dominated by the *threat point* of other games. Games $g6$ and $g7$ can be pruned after solving game $g9$; since $g9$ solution dominates the *MaxUtility point* of $g6$ and $g7$, making them uninteresting.

- *Game Clipping*: The knowledge gained by the solution of some games may force us to reconsider previous solutions. Consider Figure 3 again. After solving $g8$, we know that we can gain a utility of u_{g8}^* , and therefore the *threat point* of $g5$ needs to be redefined. This essentially redefines the entire game and may result in a new arbitration point for $g5$. The solution of $g8$ has a similar effect on $g7$. As a result, rectangles corresponding to solved or yet unsolved games are *clipped*.

It is clear from Figure 3 that game solving order affects the total number of games need to be played; solving $g7$ before $g9$ causes redundant playing of an extra game ($g7$).

We distinguish two problem settings: (1) *no-interaction*, where only game dominance relationships are considered; and (2) *interaction*, where both game dominance and clipping are considered. The latter case is more challenging since solved games might lead to clipping other games necessitating “replaying” them. We restate our main problem in the context of cooperative games as: *Find a complete*

version of g_2 . Solving g_2 redefines g_1 , and hence g_1 needs to be played again. The resulting sequence (g_1, g_2, g_1) is a complete one since no other game needs to be played. Now consider the other shaded path that starts by g_3 . The solution of g_3 does not fully dominate any other game, hence g_1, g_2 and g_4 are all possible children of g_3 . Solving g_4 next, redefines g_3 , making it a possible child of g_4 . The path continues by solving g_2 , then g_1 . The path is not complete since g_1 affects back g_2 necessitating solving g_2 again.

Search Algorithms. We designed a *branch-and-bound* depth-first search (DFS) algorithm that explores the search tree to find the shortest path from tree root to a leaf node. Each node is expanded by generating all remaining games yet to be played. When a node is reached such that its solution does not modify the game set and no other games need to be played, a complete sequence is obtained. The algorithm prunes all search paths of length larger than the current shortest complete sequence. Each node saves game configuration that resulted by playing the node’s game. The algorithm does not generate the complete search space by pruning paths that will not contribute to the shortest sequence. Branch-and-bound algorithm is guaranteed to find a (or all) shortest complete sequences. However, the number of solved games is much larger than the complete sequence length. We next describe heuristics to find short sequences quickly in the search tree.

Heuristics. In order to take into account clipped games, we modify the second estimate of the pruning power of a game g described in Section 3.1 as follows: *Number of affected games*; the pruning power of g is represented by the number of games that are either fully or “partially” dominated by $(\hat{u}_g^*, \hat{v}_g^*)$. The *Pruning Area* estimate remains unchanged since it is context-independent (referred to here as *Clipping Area*). Our heuristics evaluate pruning power after solving each game, and use it to score games. Next game to play is the game with the current highest score.

Algorithm 1 modifies the general procedure in Section 3.1 to consider continuously changing game set. A priority queue, Q , of all unsolved games, and a set S of solved games are maintained. Initially all games are inserted in Q in descending pruning power, which is estimated according to one of two approximations: number of affected games, or clipping area. After solving game g , retrieved from Q top, dominated games are eliminated and affected games are redefined. Then, g is inserted in S , and all affected games in S are removed and reinserted in Q again. The algorithm terminates when Q is empty, where S contains the pareto-optimal solutions.

3.3 One-to-Many Interactions

In this setting, two parties P_1, P_2 are involved, one defining a single negotiation profile and the other defining multiple profiles, e.g., a user against multiple web sites/services. Our interest is to find the game that maximizes the util-

Algorithm 1 Finding a Good Sequence

- 1: initialize Q a priority queue of all unsolved games
 - 2: initialize $S \leftarrow \{\}$ a set of all solved games
 - 3: **for** each game g **do**
 - 4: compute $(\hat{u}_g^*, \hat{v}_g^*)$
 - 5: compute $score_g$ according to one the two estimates of the pruning power of g
 - 6: insert g in Q with $score_g$
 - 7: **end for**
 - 8: **while** Q is not empty **do**
 - 9: $g \leftarrow Q.top$ and remove g from Q
 - 10: solve g to get (u_g^*, v_g^*)
 - 11: add g to S
 - 12: remove all fully dominated games from S and Q
 - 13: **for** each game l affected by (u_g^*, v_g^*) **do**
 - 14: redefine l
 - 15: **if** $l \in S$ **then**
 - 16: remove l from S
 - 17: compute $(\hat{u}_l^*, \hat{v}_l^*)$ and $score_l$
 - 18: insert l in Q with $score_l$
 - 19: **end if**
 - 20: **end for**
 - 21: **end while**
-

ity/payoff of P_1 among all games that can be formed with the profiles of P_2 . The problem can be viewed as a projection of all games on the P_1 utility axis. Each unplayed game is represented as an interval from $(min_g^{(x)}, max_g^{(x)})$.

The No-interaction Scenario A naïve approach is to solve all games and sort them by P_1 utility in the arbitration point. Pruning can be also conducted on-line; pruning all fully dominated games after solving each game. Straightforward modifications to heuristics in Section 3.1 can be applied for more aggressive pruning.

Allowing Interaction In this scenario, a game solution can dominate part of another game solution space. Figure 5 shows game configuration projected on P_1 ’s utility axis. Game g_1 was pruned before playing any game since the minimum utility of g_2 dominates g_1 ’s maximum profit. After solving g_2 further pruning can be achieved by eliminating g_5 and clipping g_4 and g_6 .

4 Identifying Top- k Games

We discuss in this section finding the top- k game solutions that maximize a function \mathcal{F} on players payoffs, e.g., the sum of the payoffs. Pareto-optimal solutions are not guaranteed to contain the top- k solutions (only the top-1). Top- k as defined is not applicable when game interaction is allowed, since pruning games is part of the problem definition, and the game set changes by redefining games.

We now describe how to compute top- k game solutions with no interaction. Consider a function $\mathcal{F}(u_x, u_y)$ on players utility and a set of unsolved games. Let score of game g , $score_g$, be the result of evaluating \mathcal{F} on (u_g^*, v_g^*) . Hence,

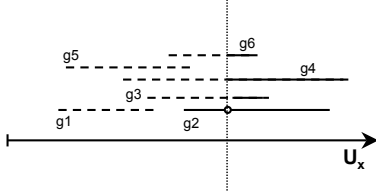


Figure 5. On-to-many game interaction.

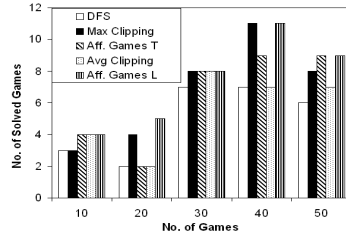


Figure 6. No Interaction

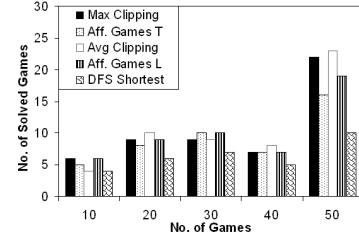


Figure 7. Allowing Interaction

Algorithm 2 Get Next Top- k Game

```

1: while  $Q$  is not empty do
2:    $g \leftarrow Q.top$  and remove  $g$  from  $Q$ 
3:    $T \leftarrow \overline{\mathcal{F}}_g$  of  $Q.top$ 
4:   if  $g$  is solved OR  $\mathcal{F}_g \geq T$  then
5:     return  $g$  and break;
6:   else
7:     solve  $g$  to get  $(u_g^*, v_g^*)$ 
8:      $\mathcal{F}_g \leftarrow \mathcal{F}(u_g^*, v_g^*)$ 
9:     insert  $g$  in  $Q$  with  $\mathcal{F}_g$ 
10:  end if
11: end while

```

$score_g = \mathcal{F}(u_g^*, v_g^*)$. For a game g , let $\overline{\mathcal{F}}_g$ and $\underline{\mathcal{F}}_g$ be an upper-bound and a lower-bound of $score_g$, respectively. Algorithm 2 incrementally retrieves the top- k games with respect to \mathcal{F} . The algorithm assumes a priority queue, Q , of all games ranked on their upper-bound score $\overline{\mathcal{F}}$. When a game is solved, its score can be accurately calculated. Algorithm 2 reports (on each invocation) the next top- k game according to \mathcal{F} . Note that no pruning of games is performed, instead all games are kept ranked on their score.

It can be shown that if \mathcal{F} is monotone, then applying \mathcal{F} on *MaxUtility point*, gives a tight upper-bound and makes the aforementioned algorithms optimal in the number of solved games. Since $\overline{\mathcal{F}}_g$ is the tightest upper-bound that can be obtained without solving a game or making a random guess; the optimality follows from the \mathcal{A}^* search algorithm. We omit the proof details due to space constraints.

Algorithm 2 introduces further optimization, by allowing the top- k game to be reported without solving if it is guaranteed to score higher than all other games. The idea is to keep a lower bound on the game score (computed on the *threat point*) and report a game when the lower-bound score is higher than the upper-bound score of all games.

5 Experiments

We describe here an experiment that evaluates complete sequence length obtained using different heuristics. We refer to [6] for our full experimental study. In this experiment we evaluated four different heuristics that estimate pruning power of a game g :

- *Max (Avg) Clipping* : Pruning power is the area between the origin and g 's *MaxUtility* (center) point.
- *Loosely (Tightly) Affected Games* : Pruning power is the number of games partially or fully inside the area between the origin and g 's *MaxUtility* (center) point.

Figures 6,7 compare heuristics to optimal path length (obtained from depth-first exhaustive search algorithm) in *no interaction* and *allowing interaction* scenarios, respectively. We used a synthesized datasets of games that are uniformly distributed in the space. In general, the *Tightly Affected Games* heuristic generates the shortest sequences compared to other ordering heuristics. The saving in sequence length is due to the context-awareness of the *Affected Games* heuristic. The *Loosely Affected Games* heuristic shows a relatively worse behavior compared to the *Tightly Affected Games*. This is explained by the large variation in game size which leads the *Loosely Affected Games* heuristic to false estimates whenever the solution point of a game lies far from its *MaxUtility point*.

6 Conclusions

We addressed the problem of identifying pareto-optimal and top- k bargaining solutions in large-scale web interaction. We proposed algorithms and heuristics to efficiently solve different problem variants. Our work raises several interesting questions for further study; identifying and analyzing the complexity of the problem, providing hardness results or polynomial time algorithms, and studying the applicability of more elaborate models are important future directions.

References

- [1] S. Borzsonyi, D. Kossmann, , and K. Stocker. The skyline operator. In *ICDE*, 2001.
- [2] L. F. Cranor. *Web Privacy with P3P*. O'Reily, 2001.
- [3] P. Morris. *Introduction to Game Theory*. Spinger, 1991.
- [4] J. F. Nash. The Bargaining Problem. *Econometrica*, 1950.
- [5] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. In *SIGMOD*, 2003.
- [6] M. A. Soliman, I. F. Ilyas, and N. Koudas. *Skyline and Top-k Processing in Web Bargaining*. Technical Report CS-2006-45, University of Waterloo, 2006.