Stable Marriage Algorithm Using Skylines

Rohit Anurag
ranurag
12585
ranurag@iitk.ac.in
Dept. of CSE

Rahul Nayak
rahulnk
12538
rahulnk@iitk.ac.in
Dept. of CSE

Sudhir
sudkumar
12734
sudkumar@iitk.ac.in
Dept. of CSE

Indian Institute of Technology, Kanpur

Group 14, Skyline Query Project Report

13th November, 2015

Abstract

Job of a good matchmaker can be considered one of the toughest job. His role is to make an arrangement which is stable and does not lead to a breakup in the future. With n number of female clients and male clients our role would be to assign to each of them a good partner according to their preference. Earlier David Gale and Lloyd Shapley tried to solve this problem by making a famous algorithm known as Gale and Shapley Algorithm, but this algorithm on application can prove to be a bit more time and space complex. We in our project tried to solve this issue by introducing skylines in it. It not only makes the algorithm more efficient but also gives it a chance to be more practical and useful for people.

I. Introduction

The problem of stable marriage: Imagine you are a matchmaker, with one hundred female clients, and one hundred male clients. Each of the women has given you a complete list of the hundred men, ordered by her preference: her first choice, second choice, and so on. Each of the men has given you a list of the women, ranked similarly. It is your job to arrange one hundred happy marriages.

It should be immediately apparent that everyone is not guaranteed to get their first choice: if a particular man is the first choice of more than one woman, only one can be matched with him, and the other women will have to make do with less. Rather than guarantee the purest of happiness to everyone — a promise that almost surely would subject you to eventual litigation — your challenge is to make the marriages stable. While finding and keeping a mate is a good deal more complicated then the mathematically simple problem stated, methods for achieving stable marriage have real uses

Then there came two guys Gale and Shapley who took the arduous to solve this problem throgh an interesting mathematical algorithm named as Gale and Shapley algorithm. As an input this algorithm takes the preference order of each man and woman and output the perfect stable match. On making real life application of this algorithm it might be much time complex. So our project tries to solve this issue by introducing skylines.

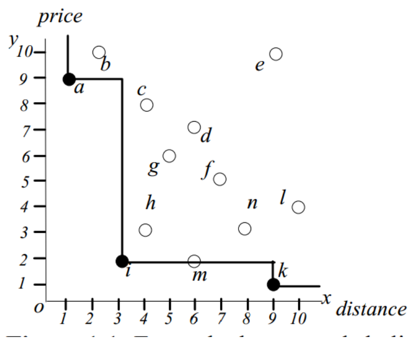

Figure 1: *Example dataset and skyline*

We here take as input two files one for men and other for women. Each line of this input file gives the id no of the man, m number of preference function coefficient and their n number of qualities. The preference coefficient is the . These coefficient makes a preference function for the man or woman. The higher the coefficient the higher the man have the desirability for a particular quality in woman. Let us take an example, let the men files have 4 number of preference coefficient and 3 number of qualities. The qualities of men are height,age and salary. These 4 preference coefficient of men are on the 4 qualities of women. We have four qualities of women as Complexity,Age,Salary and Height and 3 preference coefficient over men on their qualities. The following example linear preference functions for two men are $F = \{f_1, f_2, f_3\}$ which prioritizes on women's qualities

$f_1 = 0.2X + 0.4Y + 0.9Z + 0.6W$
$f_2 = 0.5X + 0.6Y + 0.3Z + 0.4W$

Here the coefficient of X gives the metrics for complexity of women.Second man's coefficient is higher than first man's coefficient, which means second man prefer more fair complex girl than first man. Similarly we have preference function for women.

So as we know that the Gale Shapley algo needs the preference order for each men . We compute the aggregate value for each function over the women's quality. Suppose we take the quality tuple of two woman that is c = (8,5,1,2) and d = (2,4,6,3) . On computing the value of aggregate value of c and d over f_1 we get the following values $f_1(c) = 5.7$ and $f_1(d) = 9.2$. This means f_1 prefers d over c in the preference list. In the same way we make two preference list one for women and other for men and feed them into our developed algorithm.

II. Related Work Used

I. R-Tree Implementation

R-trees are tree data structures used for spatial access methods, i.e., for indexing multi-dimensional information such as geographical coordinates, rectangles or polygons. A common real-world usage for an R-tree might be to store spatial objects such as restaurant locations or the polygons that typical maps are made of: streets, buildings, outlines of lakes, coastlines, etc. and then find answers quickly to queries such as "Find all museums within 2 km of my current location", "retrieve all road segments within 2 km of my location" (to display them in a navigation system) or "find the nearest gas station" (although not taking roads into account). The key idea of the data structure is to group nearby objects and represent them with their minimum bounding rectangle in the next higher level of the tree; the "R" in R-tree is for rectangle. Since all objects lie within this bounding rectangle, a query that does not intersect the bounding rectangle also cannot intersect any of the contained objects. At the leaf level, each rectangle describes a single object; at higher levels the aggregation of an increasing number of objects.

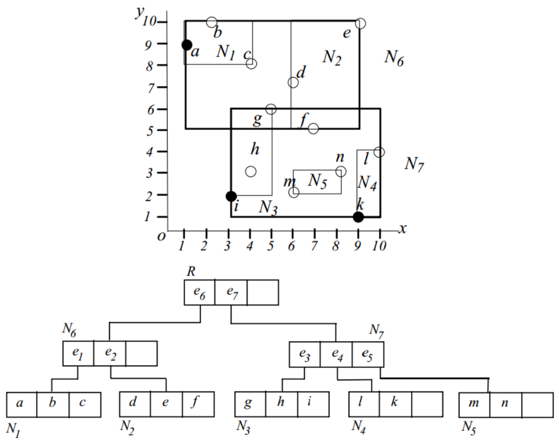

Figure 2: *R-Tree*

We have follwing functions and algorithms used in R- Tree

- **Search:** The input is a search rectangle (Query box). Searching is quite similar to searching in a B+ tree. The search starts from the root node of the tree. Every internal node contains a set of rectangles and pointers to the corresponding child node and every leaf node contains the rectangles of spatial objects (the pointer to some spatial object can be there).
- **Insertion:** To insert an object, the tree is traversed recursively from the root node. At each step, all rectangles in the current directory node are examined, and a candidate is chosen using a heuristic such as choosing the rectangle which requires least enlargement. The search then descends into this page, until reaching a leaf node. If the leaf node is full, it must be split before the insertion is made. Again, since an exhaustive search is too expensive, a heuristic is employed to split the node into two.
- **Splitting:**Since redistributing all objects of a node into two nodes has an exponential number of options, a heuristic needs to be employed to find the best split. In the classic R-tree, Guttman proposed two such heuristics, called QuadraticSplit and LinearSplit. In quadratic split, the algorithm searches for the pair of rectangles that is the worst combination to have in the same node, and puts them as initial objects into the two new groups. It then searches for the entry which has the strongest preference for one of the groups (in terms of area increase) and assigns the object to this group until all objects are assigned (satisfying the minimum fill).
- **Deletion:** Deleting an entry from a page may require updating the bounding rectangles of parent pages. However, when a page is underfull, it will not be balanced with its neighbors. Instead, the page will be dissolved and all its children (which may be subtrees, not only leaf objects) will be reinserted. If during this process the root node has a single element, the tree height can decrease.

II. Branch and Bound Skyline Queries(BBS)

Following we discuss about skylines and algorithm to compute it from the given set. Let us take a set O of D-dimensional points. Point $x \in O$ is said to dominate point $x^{'} \in O$, if for all dimensions i, $1 \leq i \leq D, x_i$ (i.e., the value of x in dimension i), is greater than or equal to $x_i^{'}$.

The skyline of O consists of all points $x \in O$ that are not dominated by any other point in O . Skyline computation gets faster and more efficient if the objects are indexed (here objects is same as points as discussed above) . BBS algorithm assumes that the R Tree must have indexed the set O and computes the skyline of O by accessing the minimum number of R-Tree nodes (i.e it is I/O optimal). Skyline point is the corner of the space with maximum atribute values in all dimensions and hence is considered the most favourable point. BBS gets access to the nodes in ascending distance order from skyline point. Once this point is found by BBS, all the points,nodes and subtree in R-Tree dominated by this point are pruned.

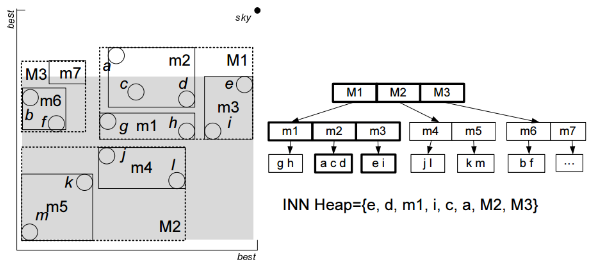

Figure 3: *Example of Branch-and-Bound Skyline*

```
Algorithm BBS (R-Tree R)
S = $\phi$   //list of skyline points
insert all entries of the root $R$ in the heap
while heap not empty do
    remove top entry $e$
    if $e$ is dominated by some point in $S$ then
        discard $e$
    else
        if $e$ is an intermediate entry then
            for each child $e_i$ of $e$ do
                if $e_i$ not dominated by $S$
                then
                    insert $e_i$ into heap
                end
            end
        else
            insert $e_i$ into S
        end
    end
end
```

Algorithm 1: BBS algorithm

The illustration of BBS algorithm is mentioned in given figure below where O contains 13 objects indexed by the shown R-Tree. BBS follows and execute the Incremental Nearest Neighbour Search (INN) from the skypoint. The first nearest object is e. There is also one more below which shows the content of INN heap when e is found. The R-Tree nodes accessed so far are drawn in bold color. Object e is the skyline and is placed n O_{sky}. We see that heap M2 and M3 is dominated by point e and is subsequently pruned. e dominates all the object in the shaded reqion and hence heap elements d,m1,i and c are pruned. The next found is a and is added to O_{sky}. BBS terminates at this moment (INN heap is empty) with O_{sky} = { e,a } .

III. Gale and Shapley Algorithm

An instance of size n of the stable marriage problem involves two disjoint sets of size n, the men and women . Associated with each person is a strictly ordered preference list containing all the members of the opposite sex. Person p prefers q to r, where q and r are the opposite sex to p, if and only if q precedes r on p's preference list.

For such an instance , a matching M is a one-one correspondence between men and women. If man m and woman w are matched in M, then m and w is called partners in M, we write , = $p_M(w)$, $w = p_M(m)$; $p_M(m)$ is the M partner of m and $p_M(w)$ the M partner of w.

A man and woman are said to block the matching M ,or to be a blocking pair for M, if m and w are not partners in M, but m prefers w to $p_M(m)$ and the w prefers m to $p_M(w)$. A matching for there is at least one blocking pair is called unstable and is otherwise unstable.

The stable matching algorithm involves the determination of the stable matching. Here Gale and Shapley both proved that the stable matching is possible and hence GS algorithm is formed by them to compute the stable pairs.

```
Algorithm GSA (M,W)
Inititalize all $m \in$ M and $w \in$ W to *free*
**while** $\exists$ *free man m* **do**
    w = highest ranked woman to whom
    $m$ has not proposed
    **if** *w is free* **then**
        ($m$,$w$) become *engaged*
    **else**
        some pair ($m'$,$w$) already exists
        **if** *w prefers m to m'* **then**
            ($m$,$w$) becomes *engaged*
            $m'$ becomes *free*
        **else**
            ($m'$,$w$) remain *engaged*
        **end**
    **end**
    // all items of M are matched to the items
    of W and someof W may be unmatched
    return (matched,unmatched)
**end**
```

Algorithm 2: Gale-Shapley Algorithm

We have two sets one is M consisting of men and W consisting of women.First the men propose to their highest preferred women that they haven't already proposed to. If she isn't already married, they are paired. If she is already married then if she prefers her current partner our sample suitor proposes to the next woman on his list. If she prefers out sample suitor, they get married and her former partner proposes to the next woman on his list.

III. Problem Statement

Our model includes a set of user preference functions F over a set of multidimensional objects O. Here the preference function belongs to men and object to women or vice versa. Each object $o \in O$ is represented by D feature values $o_1 ... o_D$. Every function $f \in F$ is defined over these D values and maps object $o \in O$ to numeric score $f(o)$. F may contain any monotone function; i.e if for two objects $o, o' \in O, o_i \geq o'_i$, $\forall i \in [1, D]$,then $f(o) \geq f(o')$, $\forall f \in F$. For ease of presentation, however we focus on linear functions; i.e each function specifies D weights $f.\alpha_1 f.\alpha.D$, one for each dimension. The weights are normalized , such that $\sum_{i=1}^{D} f$ equal 1. This assures that no function is favoured over another. Given an object $o \in O$, its score is with respect to an $f \in F$ is.

$$f(o) = \sum_{i=1}^{D} f.\alpha_i \cdot o_i \qquad (1)$$

This score is used to rank the men or women for each individual in a preference order. These preference order is then used to feed as an input in our algorithm. The object with the n dimensionality qualities is also used to find the skylines which is used for efficient query.

Now our goal is to find a stable 1-1 matching between M and W that is between men and women subject to the condition that function f prefers $o$to $o' f(o) > f(o')$ and vice versa applicable to both men and women from two sets.

IV. Gale Shapley Using Skylines

We have now the modified version of Gale Shapley Algorithm which uses skylines. We have two sets M and W which have qualities over D dimensions. At the start of the algorithm R-Tree for both the sets is made. Enter inside the loop untill each of the tree is empty. Now skylines for both of the sets is extracted using the BBS algorithm. Now we got the skyline objects . These skylines is over qualities of both men and women . Now over this skyline sets the preference function for each men and women in skyline set is applied.

This computes the aggregate value which is used to rank men and women from skyline set for each man and woman in that particular set. This two set is then feeded to the GSA algorithm as mentioned above in this report. We get the matched and unmatched portions from them. On getting the matched sets, we delete them from both men and women sets and again move further in calculating the skylines till there is a empty set. The algorithm is given below in a precise manner. In the next section of this report we proof the correctness of our algorithm.

```
Algorithm Stable-Marriage-Skyline (A,B)
// create R-Tree $R_A$ for A and $R_B$ for B
$R_A$ = RTree(A)
$R_A$ = RTree(B)
// now loop until either of the trees is empty
while $R_A$ not empty and $R_B$ not empty do
    // get skylines for $R_A$ and $R_B$ using BBS
    Sky(A) = BBS($R_A$)
    Sky(B) = BBS($R_B$)
    // get stable pairs using Modified-GSA
    (matchedA,matchedB),unmatched =
    M-GSA(Sky(A),Sky(B))
    // delete matched items from their
    corresponding trees
    for each a in matchedA do
        delete $a$ from $R_A$
    end
    for each b in matchedB do
        delete $b$ from $R_B$
    end
end
```

V. Correctness And Stability Proof

Correctness and stability proof of the modified GSA algorithm is done in this particular section. Stability is necessary for the pairs to not have a break in their tie in future. Hence the very correctness is important.

Lemma V.1. *Given $a \in A$ and b_1,$b_2 \in B$ such that $b_1 \in$ Sky(B) and $b_2 \notin$ Sky(B)*
Let f_a be the requirement function for a.

$$f_a = b_{i_1} \cdot \alpha_1 + \ldots + b_{i_d} \cdot \alpha_d$$

where b_{i_j} is value of b in j^{th} dimension and α_j is requirement coefficient of a for b_{i_j} such that

$$0 \le \alpha_j \le 1 \text{ and } \sum_{j=1}^{D} \alpha_j = 1$$

then

$$f_a(b_1) \ge f_a(b_2)$$

Proof. Since $b_1 \in$ Sky(A) and $b_2 \notin$ Sky(B)

$$\implies \forall i \in \{1,\ldots,D\} : b_{1_i} \ge b_{2_i}$$

Since α_i is normalized over [0,1],

$$\alpha_i \cdot b_{1_i} \ge \alpha_i \cdot b_{2_i} \forall i \in \{1,\ldots,D\}$$

$$\implies \sum_{i=1}^{D} \alpha_i \cdot b_{1_i} \ge sum_{i=1}^{D} \alpha_i \cdot b_{2_i}$$

$$\implies f_a(b_1) \ge f_a(b_2)$$

□

Corollary V.1.1. *If $b_1, b_2 \in B$ such that $b_1 \in$ Sky(B) and $b_2 \notin$ Sky(B) and $a \in A$, a will prefer b_1 to b_2.*

$$b_1 \succ_a b_2$$

Lemma V.2. *Given set A and B, modified-GSA always return stable matched pairs.*

Proof. It follows from the correctness of Gale-Shapley algorithm. □

From above two lemmas, we can say that finding skylines set and then applying modified-GSA over skyline sets will return stable matched pairs and is correct.

VI. Problem variants

- **JEE Seat Allocation** Every year we have the highly competitive Joint Entrance Exam. There is always involved a cumbersome procedure in allocating the seats to the students. Students have some preference over institutes over their qualities like placements,Academics, Campus Infrastructure while institute prefers students with good JEE rank, their educational qualifications, work experience,etc. So every year we see a huge rush on servers during this period. Our algorithm can help in reduce the load to the servers by employing this more efficient stable matching system.

- **DDA housing allocation** Delhi Development Authority every year comes with new sets of houses and flats for sale at cheap rate. It become hard for DDA officials to make a good allocation system. Buyers go for cheap houses,good location and better amenities. While DDA prefers buyers who have a good collateral for the loans taken for buying houses as they want no problems from banks in the future. They look into the economics activity of the buyer and prefer them who are wel

VII. Experiments With Data

In this section we empirically evaluate the performance of our algorithm. Real life data is given to the new modified algorithm SMS(Stable Marriage using Skylines). These data is generated through a randomized algorithm.

We compare our SMS algorithm with the Old Gale and Shapley Algorithm on dimension and number of object pairs. We have two types of dimensions, one is quality dimension and other is preference dimension for both male and female and their sum is equal for both of them.

We plot the runtime vs dimension graph for both SMS and Old Gale Shapley Algorithm.Also plot the runtime vs number of object pair for both of the algorithms.

VIII. Results and Discussion

- **Performance over increasing number of object pairs:**
  Keeping the quality dimensions constant we observe the variation of runtime with increasing number of object pairs. Initially with less number of object pair the runtime for both of them is nearly same but as the number of objects increases we see the better performance of SMS algorithm over Galse Shapley. It thus comes out to be sufficient. Given below is the result table and graph.

Data Size	Time(ms)
100	44.0649986267
200	264.034986496
300	721.927165985
400	1563.74907494
500	2944.24295425
600	4690.94610214
600	4686.1770153
800	11124.3948936
900	15004.7609806
1000	20375.7810593
1500	69525.5560875
2000	168032.291889
2500	333879.097939
3000	604487.416029
3500	928510.026932
4000	1436655.93195
4500	2027516.62111

Table 1: *Data Size vs Time for GSA at dimension=8*

Data Size	Time(ms)
100	119.868993759
300	4115.5500412
400	6625.64992905
500	6772.97711372
600	9224.0550518
600	10498.0890751
800	13072.9961395
900	19943.8180923
1000	20599.7519493
1500	76703.5870552
2000	80224.5101929
2500	157267.459869
3000	440899.13106
3500	529125.560999
4000	521719.007969
4500	817114.859104

Table 2: *Data Size vs Time for SMS at dimension=8*

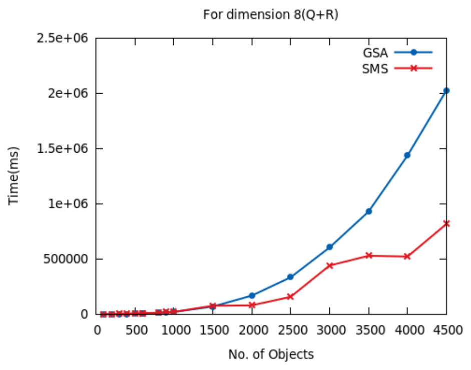

Figure 4: *Number of Object Pairs VS Runtime*

- **Performance over increasing number of quality dimensions:**
  Keeping the number of object pairs constant we see that for some number of less dimensions runtime of SMS algorithm is better that Gale Shapley but as the dimensions increases the SMS runtime goes on increasing as the R-Tree takes more time in comparision and the tree becomes more complex. Hence for higher dimensions we found SMS to be less efficient than Gale Shapley.Given below is the result table and graph.

Dimension	Time(ms)
2	25253.3271313
3	21875.2689362
4	19390.8989429
5	20651.9780159

Table 3: *Dimension vs Time for GSA at data size=1000*

Dimension	Time(ms)
2	12119.3020344
3	19964.1339779
4	33531.7699909
5	41672.4250317

Table 4: *Dimension vs Time for SMS at datasize=1000*

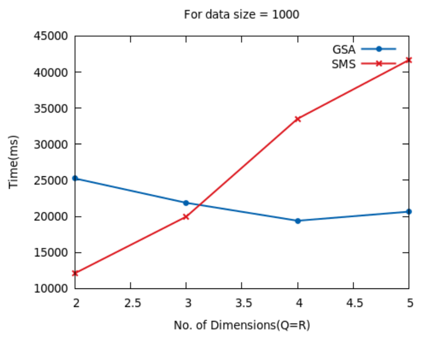

Figure 5: *No of Dimensions VS Runtime*

IX. Conclusion

In this report, we developed a new Stable Marriage algorithm using skylines and compared it with the old Gale Shapley Algorithm. We found our newly developed algorithm beating the old one in efficient with increasing number of object pairs. We also concluded that as the dimension cardinality increases R-Tree becomes less efficient and give a poor time performance VS Old Gale Shapley Algorithm.

X. Future Work

In future we can implement disk based R-Tree in computing skylines. We can also research more on this algorithm and try to reduce time complexity more and make this new modified version better.

References

[1] Gale-Shapley Algorithm: D.GALE L.S.SHAPLEY
paper link

[2] Fair Assignment based on multiple prefrence queries: Leong Hou U,Nikos Mamoulis,Kyriakos Mouratidis
paper link

[3] An Optimal and Progressive Algorithm for Skyline Queries: Dimitris

Papadias, Yufei Tao, Greg Fu, Bernhard Seeger
paper link

[4] R-TREES. A DYNAMIC INDEX STRUCTURE FOR SPATIAL SEARCHING: Antomn Guttman
paper link

[5] https://en.wikipedia.org/wiki/R-tree