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

Stable marriage is well known term in computer science and is used in many fields such as Job assignment, making marriage etc. Given preference of two sets over each other, our goal is to find a match which is stable. Gale-Shapley has given algorithm for finding a stable match. In this paper, we present a new method which uses Gale-Shapley as a subroutine to find the stable matching. Our algorithm saves a lot of computation by using the concept of skylines.

I. Introduction

Stable marriage is a well known term in the field of computer science. Given some preferences of two data sets over each other, our goal is to find stable matched pairs. There are algorithms for giving stable match but are inefficient for big data. Skylines are used to get our preferred data subset from a big data. Using this preferred set, we can save a lot of computation in many practical problem such as stable marriage problem. In this paper, we present an algorithm which solve the problem of stable marriage efficiently using skyline. We start with taking an example to illustrate the problem. After this we state the problem formally.

Stable Matching problem is stated in term of preferences of data sets over each other. These preferences can be calculated in many ways. For example, preferences can be given directly or there can be a monotonic linear function associated with each element to calculate the preference for it over other elements. In our algorithm, we have the second approach to calculate the preferences which is more practical, as each element can give some values for each attributes of other set's attributes and then we can calculate a preference over each element of other set by using a function. Let's look at an example to understand it more clearly.

Example 1:

Lets start our discussion with an example. We will use familiar terms to introduce our example but the notation can be extended to other notations also. Let M and W be two sets for 'men' and 'women' respectively. Elements within the sets are given in Table 1 and Table 2.

In data sets Q_i represents the quality attributes like beauty, smartness, height etc. and R_i represents the requirement values which are applied over qualities of other set i.e. R_i of M will be over Q_i of women. As we can clearly

see that requirements values are normalized over 1 to create a normalized monotonic linear function.

id	Q_1	Q_2	R_1	R_2	R_3
1	6	5	0.3	0.4	0.3
2	4	7	0.5	0.5	0.0
3	7	3	0.2	0.5	0.3
4	5	3	0.6	0.2	0.2

Table 1: *Set M*

id	Q_1	Q_2	Q_3	R_1	R_2
1	5	4	5	0.4	0.3
2	5	7	6	0.5	0.0
3	4	6	4	0.6	0.4
4	5	6	3	0.5	0.5

Table 2: *Set W*

But this can be extended to any monotonic linear function which is fair to all elements. Here by fair, we mean that a candidate should not be allowed to give too much values to attributes to maximize his/her profit. So we calculate the preference function for a candidate 'a' over 'b' as follows:

$$f(m_{1w_1}) = 0.3 * 5 + 0.4 * 4 + 0.3 * 5 = 4.4$$
$$f(m_{1w_2}) = 0.3 * 5 + 0.4 * 7 + 0.3 * 6 = 6.1$$
$$f(m_{1w_3}) = 0.3 * 4 + 0.4 * 6 + 0.3 * 4 = 4.8$$
$$f(m_{1w_4}) = 0.3 * 5 + 0.4 * 6 + 0.3 * 3 = 4.8$$

The same way, we can calculate values for other men m_2, m_3, and m_4 over all women. Now let's calculate preference of w1 over all men.

$$f(w_{1m_1}) = 0.7 * 6 + 0.3 * 5 = 5.7$$
$$f(w_{1m_2}) = 0.7 * 4 + 0.3 * 7 = 4.9$$
$$f(w_{1m_3}) = 0.7 * 7 + 0.3 * 3 = 5.8$$
$$f(w_{1m_4}) = 0.7 * 5 + 0.3 * 3 = 4.4$$

And the same way, preferences for other women w_2, w_3 and w_4 over all men. After getting these preferences, we can use some stable marriage algorithm to compute the stable marriage. In this paper, our aim is to minimize this computation of computing preference.

I. 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

II. Preliminaries

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.

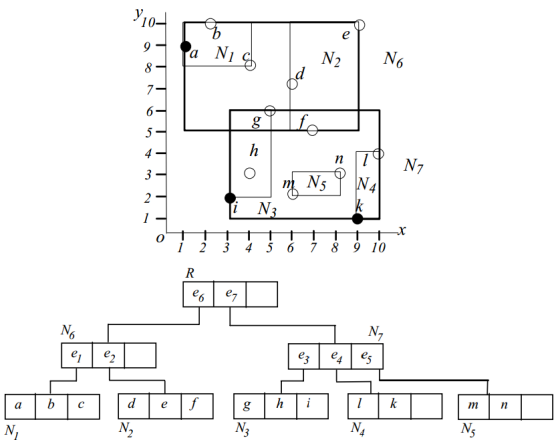

Figure 1: *R-Tree*

We have following 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 neigh-

bors. 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.

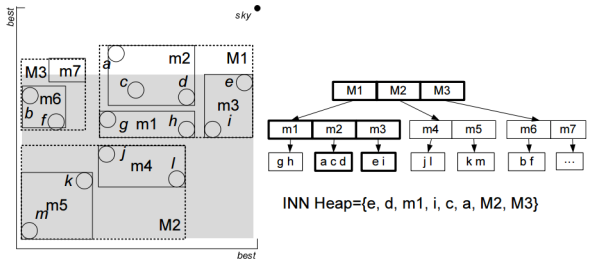

Figure 2: *Example of Branch-and-Bound Skyline*

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.

```
Algorithm BBS (R-Tree R)
S = φ  //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. 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.

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.

```
Algorithm GSA (M,W)
Inititalize all m ∈ M and w ∈ W to free
while ∃ 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
end
```

Algorithm 2: Gale-Shapley Algorithm

III. Problem Statement

Given two sets M and W with following structure:
if m ∈ M, then m has following structure:

$$m = (Q_1, Q_2, \ldots, Q_q, R_1, R_2, \ldots, R_r)$$

and w ∈ W has the following structure:

$$m = (R_1, R_2, \ldots, R_q, Q_1, Q_2, \ldots, Q_r)$$

where $0 \leq R_i \leq 1$ and $\sum R_i = 1$

Our aim is to find stable matched pairs over this two datasets. Preferences function for m ∈ M can be defined as follows:

$$f(m) = R_1 * Q_1 + R_2 * Q_2 + \cdots + R_r * Q_r$$

where R_i is requirement value of m for Q_i quality of women. and for w ∈ W as :

$$f(w) = R_1 * Q_1 + R_2 * Q_2 + \cdots + R_q * Q_q$$

where R_i is requirement value of w for Q_i quality of men.

One thing to note here is that number of quality attributes in a set (M or W) must

be equal to number of requirements attributes in another set (W or M). Order doesn't matter as long we are able to extract the attributes.

IV. Previous Work Used

Stable matching a well studied problem in computer science. Some algorithms have also presented and Gale-Shapley is one of them. In our algorithm we have used a modified version of this algorithm as a subroutine. Let's first take a look at some properties of Gale-Shapley algorithm which are useful in correctness and work-flow of our algorithm. Algorithm Gale-Shapley takes two sets (M and W) and their preferences as input. Size of both input sets must be same. The algorithm iterates over first set (may be M or W) and this set is prefered over another set. Algorithm is not guaranteed to give the same result for all the times over the same input but stability of result matched pair is guaranteed. And that's the key idea to use this algorithm as a subroutine.

I. Modification in Gale-Shapley Algorithm (M-GSA):

You will see later that our algorithm given calls to Gale-Shapley algorithm with different size set but genuine algorithm doesn't work on different size set. So we need some modification in this algorithm but keeping in mind the stability of output should not change.

```
Inititalize all m ∈ M and w ∈ W to free
if sizeOf(A) < sizeOf(B) then
    Interchange A and B variables.
end
while ∃ 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 3: Modified Gale-Shapley Algorithm

We noted a key point in Gale-Shapley algorithm that it prefers one set, say M, over which it iterates, over another, say W and if candidates in W set is more than in M, then it will fully match candidates from set M with elements of W till its same size but some items of W will remain unmatched.

Correctness: Stability of result matched output pairs is guaranteed by Gale-Shapley algorithm as we haven't modified any procedures in that algorithm. We just want to iterate the algorithm over smaller set (smaller set should be preferred) and it should affect the output. The modified algorithm's pseudocode is given above.

V. Stable Marriage using Skylines(SMS)

In recent year, skylines have a got a great attention in many big data industries. As skylines exhibits a great property of most desired data, use of skylines have

increased because of increased in data size in practical application. In general words, skylines are our most preferred subset from a set. This is the key idea to avoid a lot of computation of preferences for stable marriage problem that we have used in our algorithm. Before stating our algorithm, let's first see how we can use the skylines to reduce the computation. Let's M and W be the two data sets. These data sets follows the same pattern as stated in problem statement. Sky(M) represent the skyline set for M over quality attributes (dimensions) and likewise Sky(W) represent the skyline set for W over its quality attributes. Higher values are prefered while calculating skylines.

Lemma V.1. *A $m \in M$ will prefer a woman from Sky(W) over W-Sky(W) (non-skyline women) and likewise $w \in W$ swill prefer a man from Sky(M) over M-Sky(M) (non-skyline men).*

Proof. The proof of the lemma follows from the definition of skylines and monotonic linear preference of men and women. We present proof for a man here as proof for w will be exactly the same. Let m in M and fm be the preference function of m, w_{sky} in Sky(W) and w in W-Sky(W). Let's calculate the preference of m over wsky and w:
$f(m)_w = (R_1)_m * (Q_1)_w + (R_2)_m * (Q_2)_w + \ldots + (R_r)_m * (Q_r)_w$

$f(m)_{wsky} = (R_1)_m * (Q_1)_{wsky} + (R_2)_m * (Q_2)_{wsky} + \ldots + (R_r)_m * (Q_r)_{wsky}$
As

$$w_{sky} \geq w$$
$$\implies (Q_i)_{w_{sky}} \geq (Q_i)_w for i \in [1, ..., r]$$
$$\implies (R_i)_m * (Q_i)_{w_{sky}} \geq (R_i)_m * (Q_i)_w$$

From linearity and monotonicity of preference function,

$$\sum((R_i)_m * (Q_i)_{w_{sky}}) \geq \sum((R_i)_m * (Q_i)_w)$$
$$\implies f(m)_{w_{sky}} \geq f(m)_w$$

□

The result of above lemma is helpful in avoiding computation of preferences as at a time, we just need to calculate the preferences of candidates from skyline set and men and women will be more happier if they are matched with Skyline candidates.

Now we present flow of our algorithm in a common language. Input of our algorithm is same as stated in problem statement. We start by computing skyline set for both M and W over their quality attributes using Branch and Bound Skyline algorithm which uses R-Tree as data structure. After getting skyline set, we send these sets to our M-GSA (Modified Gale-Shapley Algorithm) and get the matched and unmatched pairs from it.

```
Algorithm Stable-Marriage-Skyline (M,W)
// create R-Tree R_M for M and R_W for W
R_M = RTree(M)
R_W = RTree(W)
// now loop until either of the trees is empty
while R_M not empty and R_W not empty do
    // get skylines for R_M and R_W using BBS
    Sky(M) = BBS(R_M)
    Sky(W) = BBS(R_W)
    // get stable pairs using Modified-GSA
    (matchedM,matchedW),unmatched = M-GSA(Sky(M),Sky(W))
    // delete matched items from their corresponding trees
    for each m in matchedM do
        delete m from R_M
    end
    for each w in matchedW do
        delete w from R_W
    end
end
```

M-GSA only computes preferences for input dataset, so have saved a lot of computation for preferences. And furthermore, matched pair from M-GSA are also stable, so we can output the final matching

for matched pair as the current matching and also can remove them from further computation so that none of other candidates have to calculate preferences for these matched pairs. And so our algorithm also provides "Progressive" nature which is very useful. Removing these matched pairs and keeping other as it, we again start by calculating skylines and doing the same procedure until all pairs are matched. Below is the pseudocode for our algorithm.

VI. Experimental Results and Discussion

Experiments for genuine Gale-Shapley Algorithm and SMS (Stable Marriage using Skylines) are performed over different dimensions and different data sizes. Experimental result are plotted below.

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 3: *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 4: *Data Size vs Time for SMS at dimension=8*

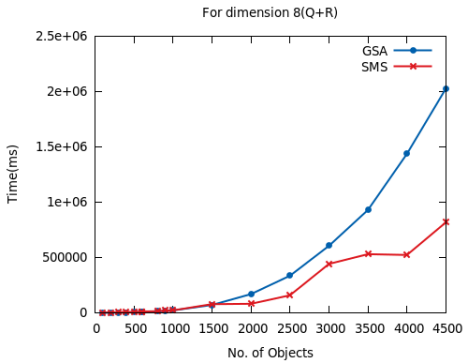

Figure 3: *Number of Object Pairs VS Runtime*

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

Table 5: *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 6: *Dimension vs Time for SMS at datasize=1000*

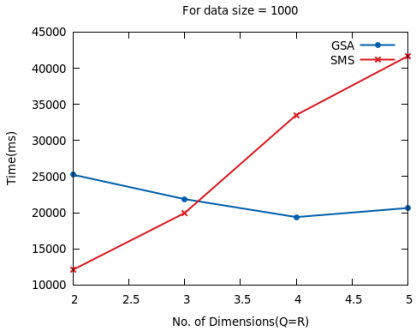

Figure 4: *No of Dimensions VS Runtime*

From these experimental results, we can see that for large size data sets, our algorithm gives good performance. But if we change the size of dimensions, keeping the data size same, our algorithm slows down more as compared to the Gale-Shapley Algorithm because of increasing computation for computing skylines and making of R-Tree. But changing the dimension is very rare and not more practical.

From our experiments, we have seen that output of Gale-Shapley Algorithm and our algorithm don't matched if preference overlaps. This is because, as discussed in section 2, Gale-Shapley Algorithm itself don't give the same result for same data set as it depends on the order in which the input (m_1, then m_2 or m_2, then m_1) is processed and also on which set (M or W) is prefered while processing the input. In our algorithm, we can clearly see that preference of sets (M or W) for M-GSA is determined according to the size of the input (lower size is prefered). And also processing of input is different from Gale-Shapley. So results set may not match of both the algorithm, but the stability is guaranteed.

VII. 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.

VIII. 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