Saladi Rahul Research Statement

The focus of my research has been the design of *algorithms and data structures for geometric problems*, which is an active field of research in theoretical computer science and databases. Most of my contributions have been to the specific field of *geometric intersection queries* (GIQ), which help answer diverse real-world queries such as: How will a tourist in New York City locate the nearest top rated hotels on his smartphone which can display only a limited amount of information at a time? How can a scientist be quickly provided the latest satellite images of a particular city he is researching? How can a real-estate agent be quickly provided the potential set of buyers for his houses?

In a GIQ problem, the user is not interested in the entire geometric dataset, but only in a small portion of it and requests an *informative summary* of that small portion of data. Formally, the goal is to preprocess a set S of n geometric objects into a data structure so that given a query geometric object q, a certain *aggregation function* can be applied efficiently on the objects of $S \cap q$. The classical aggregation functions studied in the literature are reporting, counting, or approximately counting the objects of $S \cap q$.

If we are interested in answering a single query, it can be done in linear time, by simply checking for each object $p \in S$ whether p lies in the query range q and then computing the aggregate function. However, in many applications (such as MySQL database, Google Maps etc.) the same set S is queried several times, in which case we would like to answer a query faster by preprocessing S into a data structure. The rationale here is that the cost of preprocessing will be more than compensated for by the savings in response time when answering hundreds of thousands of queries (as opposed to using a naïve query algorithm on un-preprocessed data). A central theme of my work is how to organize data into a *small-sized data structure* so that response to any user query can be provided in *real-time*. My work on GIQ problems can be broadly classified into the following two categories:

1. **Classical problems:** GIQ problems have been an active field of research since 1970s. However, there are still many open problems to work on. Solutions to the classical problems form the *basis* for building solutions for newer GIQ problems. References [1, 5] are my key contributions to this class of problems.

2. **Modern problems:** In recent years, there has been an explosion in the volume of digital data that is being generated and stored, and this promises to continue unabated as computing and storage costs drop. Moreover, modern display devices such as smartphones have relatively low computing power and small screens. These challenges make it imperative that new ways be developed to query large datasets and make sense of the results. Traditional query methods that simply report all the data items satisfying a query are no longer sufficient as (a) they place an undue burden on the user to sift through a potentially huge answer space for relevant information, and (b) displaying *all* the items on a small screen would lead to clutter, and hence, will not be informative. My work in [2, 3, 4, 6] addresses these challenges by proposing practically useful aggregation functions, and then providing efficient solutions for them.

My Contributions

In what follows, I provide a brief overview of my work on the two classes of problems discussed above (Details may be found in the cited papers).

Top-k Geometric Intersection Queries: Along with my co-authors I have done an extensive study of top-k GIQ problems [2, 3, 6], some of which has already appeared in top-tier venues. In a top-k GIQ problem, each object in S has a weight associated with it (which is determined by some ranking criteria) and the user would want to know the k largest-weight objects of S intersected by q.

For example, let S be a set of points in the plane and q be an axes-parallel rectangle (see Figure 1). The points could represent restaurants and the rating of each restaurant could be its weight. The query rectangle q can be the downtown area in New York City and the user might want to know the top-5 rated restaurants in that area. This setting is known as *top-k orthogonal range searching*. In [3], we present the first known optimal solution for this problem in the main-memory setting and an almost-optimal solution in the external-memory setting. In main-memory, our data structure has $O\left(n\frac{\log n}{\log\log n}\right)$ size and answers any query in $O(\log n + k)$ time. Previous work on this problem could guarantee an optimal solution only when points lie in $\mathbb{R}^1$.

While the effort in [3] was focused on obtaining an optimal solution for a particular problem, our effort in [2, 6] was to came up with a general framework which can handle top-k GIQ problems for any combination of input objects and query object. Such a generic framework did not exist in the literature before. Another key contribution of [2, 6] was the ease provided by our framework in letting a user to enhance existing software designed for answering GIQs to also answer top-k GIQs. This upgrade involves merely implementing a suitable binary search tree! Indeed, we have demonstrated this in [2] for top-k orthogonal range search and top-k point enclosure query. The final key contribution of [2, 6] was ensuring real-time response and succinct storage. Rigorous theoretical analysis w.r.t. a standard GIQ problem reveal that, one of our technique leads to *no performance loss* in the expected case [6], whereas the other technique leads to just a $O(\log n)$ factor loss in the worst case [2].

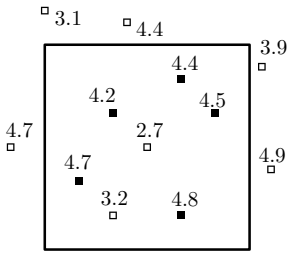
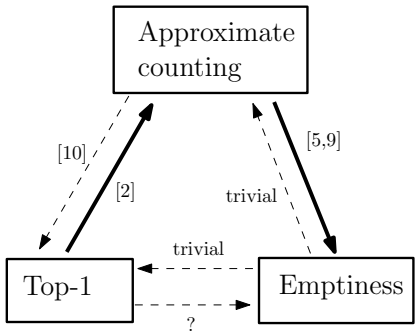


Figure 1: Top-5 rated restaurants shown as solid black squares.

Future Directions: An immediate question to address is whether a top-k GIQ problem can be reduced to its corresponding GIQ reporting problem without any loss of performance in terms of the *worst-case* size of the data structure and the query time.

I pose an even more challenging question: *"For any GIQ problem, consider its corresponding approximate counting, top-1 and emptiness query. For any pair of queries, show that there exist a bi-directional reduction without loss of efficiency (in terms of size of data structure and query time)?"* As shown in the figure, there are examples in the literature which hint positively that the above question could be resolved. However, the non-trivial reductions mentioned in the figure either (i) hold in the *expected* case, or (ii) lead to $O(\text{polylog } n)$ loss in terms of size of the data structure and/or query time. The solid arrows are reductions discovered in papers that I am involved in.



Rectangle Stabbing: In this problem, S is a set of n hyper-rectangles (possibly overlapping) that lie in $\mathbb{R}^d$, so that given a query point q, we can report all the rectangles in S containing q. Rectangle stabbing is useful in applications in which the *preferences of the users* can be modeled as hyper-rectangles in d-dimensional space. For example, consider a real-estate database that contains information on several thousand homes for sale in a large metropolitan area. A potential buyer can specify his preference as a two-dimensional rectangle: "I am looking for houses whose price in the range \$200,000 to \$500,000, and whose age is in the range 3 to 10". Here price is the x-axis and age is the y-axis. Each house can be modeled as a query point: (price, age). The output of the query will be the set of buyers interested in buying that house.

Optimal solutions for rectangle stabbing in $\mathbb{R}^1$ and $\mathbb{R}^2$ were discovered in the 1980s itself. However, for the past three decades an optimal solution in $\mathbb{R}^3$ has been elusive. There is a known lower bound of $\Omega(\log^2 n + k)$ query time for a linear-space data structure in $\mathbb{R}^3$ [7]; whereas, the state-of-the-art linear-space data structure took $O(\log^4 n + k)$ time to answer a query, where k is the number of rectangles reported [8]. At SODA'15 I presented an *almost* optimal solution: an $O(n \log^* n)$ size data structure which can answer the query in $O(\log^2 n \cdot \log\log n + k)$ time.

Future directions: My immediate goals are to (a) *completely* resolve the rectangle stabbing problem in $\mathbb{R}^3$, and (b) discover an optimal (or almost-optimal) data structure for the *dominance query* in $\mathbb{R}^4$; optimal

solutions are known only up to $\mathbb{R}^3$ [8]. In a dominance query in $\mathbb{R}^d$, S is a set of n points and the query q is an octant of the form $[q_1, \infty) \times [q_2, \infty) \times \ldots \times [q_d, \infty)$. I have made significant progress towards reaching these goals.

In the long-run I plan to make progress on the following: *"Are there range-aggregate problems which do not suffer from curse of dimensionality?"* A prime candidate seems to be the so-called *point location* problem in an orthogonal subdivision in $\mathbb{R}^d$. The current state-of-the-art is my solution at SODA'15: an $O(n)$ space data structure which can answer any query in $O(\log^{d-3/2} n)$ time. However, there are no lower bounds to suggest that the exponential dependence on d is necessary. Another prime candidate seems to be the *maximal (or skyline) points* problem in d-dimensional space.

Range-skyline queries: The *skyline* query has been very well studied in the computational geometry and the database community [11, 12, 13]. In this problem, we are given a set S of n points lying in $\mathbb{R}^d$. A point $p = (p_1, \ldots, p_d)$ is *dominated* by another point $r = (r_1, \ldots, r_d)$ iff $p_i \geq r_i, \forall i \in [1, d]$. The skyline of S is a subset $S'(\subseteq S)$ s.t. each point in S' is *not* dominated by any other point in S. For example, consider (again) a real-estate database that contains information on several thousand homes for sale in a large metropolitan area. Each house has a number of attributes associated with it: price, real-estate tax, age, distance to the high school, crime rate, etc. Let us consider each house as a point in S and each attribute as a dimension. Then the skyline of all the houses will represent the set of best houses to choose from. However, in many applications a user would want to know the best (i.e. skyline) entities from only a small portion of data and not from the entire data. In the real-estate example, a user typically wants to find the best houses to buy within a certain locality of his interest and not within the entire city. To capture such scenarios, we introduced the problem of *range-skyline* queries.

Existing solutions in the literature work efficiently only when the *range* restriction and *skyline* are both defined on the same set of attributes, whereas we consider a more general setting where they are defined on different sets of attributes. For the reporting version, we came up with a succinct data structure which incorporates both range restriction and skyline query, leading to an improved query time. For a closely-related problem, we proved a result that a small-size data structure and real-time response are both simultaneously not possible. This work was published at *ACM SIGSPATIAL GIS'12* and also received first place for presentation in the *Fast Forward Preview Session* at the conference (chosen from approximately 55 participants).

Future work: We believe that our data structure for the reporting version is almost-optimal; showing a matching lower bound would be the next step. Similar to top-k GIQ, is it possible to come up with a general technique to efficiently handle any combination of input objects and query object? Our work only focused on the case where input objects are points and the query object is an axis-aligned hyper-rectangle.

Approximate counting: Approximating the information about the objects of S intersecting q is a practical way to obtain improved space-efficient data structures and faster query algorithms, and hence, has received significant attention in the computational geometry and the database community. If k is the number of objects of S intersecting q, then in an approximate counting query one is allowed to report any value in the range $[(1-\varepsilon)k, (1+\varepsilon)k]$, where $\varepsilon \in (0, 1)$.

In [5], I obtain new results on this topic and developed some new techniques. My work initiated the study of approximate rectangle stabbing counting problem and a comprehensive understanding of the problem has been achieved for all the possible settings in $\mathbb{R}^2$. Studying the problem in $\mathbb{R}^3$ would be the next challenge. Perhaps, the most important contribution of this work is a general framework which can handle approximate counting of GIQ problems for any combination of input objects and query object. The first advantage of the framework is that it is *sensitive* to the value of k, i.e., the query time is small whenever k is large. For example, consider the *halfspace range searching problem*, where S is a set of points in $\mathbb{R}^d$, $d \geq 4$, and q is a halfspace. Existing solutions for approximate counting in this setting require a *high* query time of roughly $O\left(n^{1-1/\lfloor d/2 \rfloor}\right)$. By using our framework the query time reduces to roughly $O\left((n/k)^{1-1/\lfloor d/2 \rfloor}\right)$; which is $O(1)$ when $k = \Theta(n)$. The second advantage of our framework is that, unlike previous approaches, it can efficiently handle a larger class of GIQ problems known in the literature as *colored* GIQ problems.

Conclusion: During my Ph.D., my research was mainly focused towards proving upper bounds for various geometric query-retrieval problems. I have been steadily getting interested in the field of geometric approximation algorithms. I have taken a couple of courses in the general field of approximation algorithms and did an internship at Microsoft Research (MSR) on scheduling and fairness algorithms. In my current research, I have developed algorithms in the main-memory model and the external-memory model. Motivated by *big-data*, I plan to expand my horizons by developing sublinear algorithms and algorithms in the streaming model. Finally, I also plan to spend time proving lower bounds for various data structure problems (by building upon the tools developed in the field of communication complexity).

I have had a wonderful experience as a Teaching Assistant (TA) conducting recitations and office hours. I have always got very good reviews from the students for my teaching. I am confident about teaching advanced courses such as Advanced Algorithms and Data Structures, Computational Geometry, and Approximation and Randomized Algorithms. After a few years, I would like to be a professor at a university, so that I can continue to do my research in theoretical computer science and also continue my passion for teaching.

References

[1] Saladi Rahul. Improved bounds for orthogonal point enclosure query and point location in orthogonal subdivisions in $\mathbb{R}^3$. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 200–211, 2015.

[2] Saladi Rahul, and Ravi Janardan. A general technique for top-k geometric intersection query problems. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 26(12):2859–2871, 2014.

[3] Saladi Rahul, and Yufei Tao. On top-k range reporting in 2d space. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 265–275, 2015.

[4] Saladi Rahul, and Ravi Janardan. Algorithms for range-skyline queries. In *Proceedings of ACM Symposium on Advances in Geographic Information Systems (GIS)*, pages 526-529, 2012.

[5] Saladi Rahul. Approximate Range Counting Revisited. *Submitted to a conference, Preprint:* `http://arxiv.org/abs/1512.01713`

[6] Saladi Rahul, and Yufei Tao. Efficient Top-k Indexing via General Reductions. *Submitted to a conference, Manuscript:* `http://www-users.cs.umn.edu/~rahuls/Papers/topk.pdf`

[7] Peyman Afshani, Lars Arge, and Kasper Green Larsen. Higher-dimensional orthogonal range reporting and rectangle stabbing in the pointer machine model. In *Proceedings of Symposium on Computational Geometry (SoCG)*, pages 323–332, 2012.

[8] Peyman Afshani. On dominance reporting in 3D. In *Proceedings of European Symposium on Algorithms (ESA)*, pages 41–51, 2008.

[9] Boris Aronov, and Sariel Har-Peled. On approximating the depth and related problems. *SIAM Journal of Computing*, 38(3):899–921, 2008.

[10] Haim Kaplan, Edgar Ramos, and Micha Sharir. Range minima queries with respect to a random permutation, and approximate range counting. *Discrete & Computational Geometry*, 45(1):3–33, 2011.

[11] Stephan Borzsonyi, Donald Kossmann, and Konrad Stocker. The Skyline Operator. In *Proceedings of International Conference on Data Engineering (ICDE)*, pages 421–430, 2001.

[12] Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. Progressive skyline computation in database systems. *ACM Transactions on Database Systems (TODS)*, 30(1): 41-82 (2005).

[13] H. T. Kung, Fabrizio Luccio, and Franco P. Preparata. On Finding the Maxima of a Set of Vectors. *Journal of the ACM*, 22(4): 469-476 (1975)