Anything You Can Do, I Can Do Better: Finding Expert Teams by CrewScout

Naeemul Hassan [1], Huadong Feng [1], Ramesh Venkataraman [1]
Gautam Das [1], Chengkai Li [1], Nan Zhang [2]

[1]*University of Texas at Arlington,* [2]*George Washington University*

ABSTRACT

CrewScout is an expert-team finding system based on the concept of *skyline teams* and efficient algorithms for finding such teams [3, 4]. Given a set of experts, CrewScout finds all k-expert skyline teams, which are not dominated by any other k-expert teams. The dominance between teams is governed by comparing their aggregated expertise vectors. The need for finding expert teams prevails in applications such as question answering, crowdsourcing, panel selection, and project team formation. The new contributions of this paper include an end-to-end system with an interactive user interface that assists users in choosing teams and an demonstration of application domains beyond those in [3, 4].

1. OVERVIEW

We introduce CrewScout (`http://idir.uta.edu/crewscout`), a system for finding expert teams in accomplishing tasks. The underpinning concept of the system is *skyline teams* (called *skyline groups* in [3, 4]). The new contributions made in this paper include an end-to-end system with an interactive user interface that assists users in choosing teams among potentially many skyline teams and an extension of application and demonstration scenarios into more general areas ([3, 4] mostly focused on the application of forming teams in fantasy sports games.)

Consider a set D of n experts $t_1, \ldots, t_n$, modeled by m numeric attributes $A_1, \ldots, A_m$ that represent their skills and expertise. Any subset of k experts form a *k-expert team*. CrewScout finds, for a given k, all k-expert skyline teams, i.e., k-expert teams that are not *dominated* by any other k-expert teams. It further assists users in choosing among the skyline teams. The notion of dominance between teams is analogous to the dominance relation between tuples in skyline analysis [2]. CrewScout calculates for each team an aggregate vector, whose attribute values are aggregated over the corresponding attribute values of the experts in the team. While the concept allows arbitrary aggregate functions, CrewScout's algorithms are particularly efficient for four commonly used aggregate functions—AVG (i.e, SUM, since we only compare teams with equal size), MIN and MAX. A team G_1 *dominates* another team G_2 (denoted $G_1 \succ G_2$), if and only if the aggregate value of G_1 on every attribute is better than or equal to the corresponding value of G_2 and G_1 has better value on at least one attribute.

	t_1	t_2	t_3	t_4	t_5
database	3	0	2	2	0
indexing	0	3	1	2	2

(1) Experts

team	AVG	MIN	MAX
$G_{1,2}$	$\langle 1.5, 1.5\rangle$	$\langle 0, 0\rangle$	$\langle$ **3, 3** $\rangle$
$G_{1,3}$	$\langle 2.5, 0.5\rangle$	$\langle 2, 0\rangle$	$\langle 3, 1\rangle$
$G_{1,4}$	$\langle$ **2.5, 1.0** $\rangle$	$\langle 2, 0\rangle$	$\langle 3, 2\rangle$
$G_{1,5}$	$\langle 1.5, 1.0\rangle$	$\langle 0, 0\rangle$	$\langle 3, 2\rangle$
$G_{2,3}$	$\langle 1.0, 2.0\rangle$	$\langle 0, 1\rangle$	$\langle 2, 3\rangle$
$G_{2,4}$	$\langle$ **1.0, 2.5** $\rangle$	$\langle$ **0, 2** $\rangle$	$\langle 2, 3\rangle$
$G_{2,5}$	$\langle 0, 2.5\rangle$	$\langle$ **0, 2** $\rangle$	$\langle 0, 3\rangle$
$G_{3,4}$	$\langle$ **2.0, 1.5** $\rangle$	$\langle$ **2, 1** $\rangle$	$\langle 2, 2\rangle$
$G_{3,5}$	$\langle 1.0, 1.5\rangle$	$\langle 0, 1\rangle$	$\langle 2, 2\rangle$
$G_{4,5}$	$\langle 1.0, 2.0\rangle$	$\langle$ **0, 2** $\rangle$	$\langle 2, 2\rangle$

(2) All possible 2-expert teams

Table 1: Running Example

The need for finding expert teams prevails in several application areas, including question answering, crowdsourcing, panel selection, project team formation, and so on. This is illustrated by the following motivating examples.

CrowdSourcing Consider the task of forming a team of Wikipedia editors to write a new Wikipedia article related to the topics of "database" and "indexing". Suppose Table 1.1 shows all relevant editors $t_1, \ldots, t_5$ and their expertise on the two topics. We want to assign the task to a team of 2 editors. Table 1.2 shows the aggregate vectors under AVG, MIN and MAX, for all possible 2-expert teams where $G_{i,j}$ stands for a team of experts t_i and t_j. A simple scheme such as picking top editors on individual topics does not work. For example, $G_{1,2}$ consists of the top editor on each topic and has an aggregated vector $\langle 1.5, 1.5\rangle$ with regard to AVG. $G_{3,4}$, with vector $\langle 2.0, 1.5\rangle$, dominates $G_{1,2}$ (denoted $G_{3,4} \succ G_{1,2}$) under AVG. Hence, $G_{3,4}$ is a better team in terms of collective expertise. In fact, $G_{3,4}$ is a 2-expert skyline team, since no other team dominates it under AVG. Table 1.2 highlights all 2-expert skyline teams for every aggregate function.

Questing Answering Consider a question-answering platform such as Quora.com. A question is displayed to users who might answer it. The question asker can also explicitly solicit answers from certain users, oftentimes by offering rewards. To receive quality answers, it is necessary to intelligently post the question to users who have the expertise to answer it. More often than not, a question requires expertise on several aspects that cannot be fulfilled by any single user, needing attention from a diverse team of experts who collectively excel on the required expertise. For instance, consider question "Which programming language is best for high-performance computing? C++, Java or Python?" To get a comprehensive answer, we need experts in "programming language", "high performance", "C++", "Java", "Python", and so on.

Other Motivating Applications The need for finding expert teams arises in several other applications. **1)** Consider the task of choosing a panel of a certain number of experts to evaluate a research paper or a grant proposal. An expert can be modeled as a tuple in the multi-dimensional space defined by the paper's topics, to reflect the expert's strength on these topics. The collective expertise of a panel is modeled as the aggregate vector of the corresponding tuples. The goal is to select panels attaining strong aggregates. **2)** Similarly the problem of forming collaborative teams for a software development project can be viewed as finding programmers who are collectively strong in the multi-dimensional space of desired skills for the project. **3)** The problem also has applications in areas where we look for "teams" in more general sense, such as bundles of products, reviews, stocks, and so on. For instance, to summarize a product's many customer reviews, choosing a set of diverse reviews can be modeled as forming a "team" of reviews, where the reviews are modeled by attributes (dimensions) such as "sentiment", "length", "quality", etc. Another example of application area is the industry of online fantasy sports where gamers compete by forming and managing team rosters of real-world athletes, aiming at outperforming other gamers' teams. The teams are compared by aggregated statistics (e.g., "points", "rebounds", "assists" in basketball games) of the athletes in real games.

An attractive characteristic of a skyline team is that no other team of equal size can dominate it. In contrast, given a non-skyline team, there is always a better skyline team. This property distinguishes CrewScout from other team recommendation techniques. The skyline teams consist of the teams that are worth recommending. They become the input to further manual or automated post-processing that eventually finds one team. Admittedly, determining the "best" team is a complex task that may involve more factors than what skyline teams can capture—e.g., which experts are available for a task, whether they have good relationship to work together, and so on. The post-processing is thus crucial. Examples of such post-processing include eye-balling the skyline teams, filtering and ranking them by user preferences, and browsing and visualization of the skyline teams. Particularly, CrewScout provides an interactive tool to assist a human user in exploring and choosing skyline teams.

2. USER INTERFACE

This section explains CrewScout's user interface. In a nutshell, a user starts their interaction with CrewScout by searching available tasks and selecting a task. CrewScout then proposes a set of experts who are eligible for forming teams. Then, the system provides a set of skyline teams and also guides the user to pick a team.

Figure 2 in Appendix shows the GUI of CrewScout, which is comprised of four panels, including a *task panel*, a *skill panel*, a *parameter panel*, and a *display panel*. The task panel presents a list of available tasks. When a user clicks over a task, CrewScout provides more details about it. CrewScout also provides a keyword search box at the top of this panel for searching from available tasks. The skill panel presents the skills required for completing the selected task. It shows a checkbox for each skill. By default, all the checkboxes are checked. The user can check/uncheck some of them according to their preference. When the user clicks the "Show Experts" button, the display panel presents a paginated list of all experts who have expertise in at least one checked skill. If the user further checks/unchecks some skills, the expert list is automatically refreshed to reflect the change. Experts are ordered by summations of their expertise in all selected skills in the current implementation. In the expert list, a filter is provided for each skill. The user can filter the experts by setting the minimum and maximum expertise for one or more skills. The user can also filter the experts by their names through partial string matching.

The parameter panel allows the user to set parameters for skyline team computation. It includes a textbox for specifying the skyline team size and radio buttons for choosing an aggregate function (AVG, MIN, or MAX). Once the user clicks the "Skyline Teams" button, CrewScout calculates all skyline teams (considering all experts satisfying the aforementioned filters) and shows them in the display panel page by page (Figure 3 in Appendix). Similar to the filters on experts, CrewScout also provides filters for the skyline team list, including filters on team members' names and minimum/maximum aggregated expertise on individual skills. The teams satisfying the conditions are called the *qualifying skyline teams*. When the display panel exhibits the skyline teams, a *clustering panel* is added below the parameter panel. It provides a "Pick a Team" button and three drop-down lists that allow the user to choose a clustering algorithm (e.g., *K-means*), a similarity/distance function (e.g., Euclidean distance) for the clustering algorithm, and the number of clusters. When the user clicks the button, CrewScout will display below the current panels a visualization interface (Figure 4) that clusters the experts in the qualifying skyline teams and assists the user in exploring and choosing teams.

The visualization interface has two panels. The left panel visualizes the clusters. Each expert that belongs to at least one qualifying skyline team is represented as a circle. Circles in the same cluster are annotated with the same color. Their positions are automatically determined by the *multi-foci force layout* (`https://github.com/mbostock/d3/wiki/Force-Layout`). The size of a circle is proportional to the number of qualifying skyline teams containing the corresponding expert. At the beginning, only the labels of big circles (containing information of corresponding experts) are visible. When the user hovers the mouse over a circle, the expert's profile (including the name and the number of skyline teams containing the expert) is displayed in a small pop-up window. The user can gradually zoom in to see the labels of smaller circles. The user can iteratively select k experts. Whenever the user selects an expert, CrewScout removes those circles whose corresponding experts do not belong to any common skyline teams with all selected experts so far. The remaining circles are re-clustered and resized, based on only qualifying skyline teams containing the selected experts. The right panel presents a polar-chart. Each polygon in the polar-chart represents a selected expert's expertise on the chosen skills. The aggregated expertise of the selected experts is also represented by a polygon. The selected experts are listed under the polar-chart. The user can remove any expert by clicking the cross sign beside it, and the clusters of circles are refreshed accordingly. Once k experts are selected, a skyline team is chosen. A "Pick Another Team?" button appears in the left panel. If the user clicks it, two more panels are added to the lower portion of the visualization interface—the left one lists all selected teams and the right one presents another polar-chart that compares them.

3. ALGORITHMS

One seemingly possible solution to finding all k-expert skyline teams is to enumerate each possible combination of k experts in D, compute the aggregate vector for each combination, and find all skyline teams by a conventional skyline query algorithm. This approach has two main problems. One is its significant computational overhead, as the input size to the skyline algorithm ($\binom{n}{k}$) is extremely large even under modestly large n and k. The other problem is with the strategy of returning all skyline teams as the output. For certain aggregate functions (e.g., MAX and MIN), even the output size (i.e., the number of skyline teams) may be nevertheless too large to explicitly compute and store.

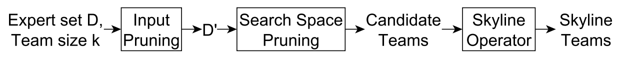


Figure 1: Flow of the Algorithm

CrewScout's underlying algorithm exploits several techniques to address the challenges: *output compression* significantly reduces the output size when the number of skyline teams is large; *input pruning* substantially shrinks the input to the algorithm; and *search space pruning* efficiently finds skyline teams. Figure 1 shows the flow of the algorithm. While details of these ideas and experiment results can be found in [3, 4], we note here that these ideas achieved orders of magnitude gain over the baseline method, in terms of execution time on real and synthetic datasets. Below we provide a brief overview of these ideas.

Output Compression A key observation is, while the number of skyline teams may be large, many of them share the same aggregate vector. For example, $G_{2,4}, G_{2,5}$ and $G_{4,5}$ in Table 1.2 have the same aggregate vector $\langle 0, 2\rangle$ under MIN aggregation function. Thus, our idea for compressing skyline teams is to store only the (much fewer) distinct skyline aggregate vectors (in short, skyline vectors) and one skyline team per skyline vector, as opposed to all skyline teams. However, while the distinct skyline vectors and their accompanying sample skyline teams may suffice in many cases, a user may be willing to spend time on investigating all teams equivalent to a particular skyline vector, and to choose a team after factoring in their knowledge and preference. Hence, CrewScout is also equipped with an algorithm to reconstruct all skyline teams corresponding to a skyline vector.

Input Pruning If an expert t is dominated by k or more experts in D, then we can safely exclude t from the input without influencing the distinct skyline vectors found at the end. This can be proved by contradiction. Assume G is a k-expert skyline team containing t and t is dominated by k or more experts. Since $|G| = k$, there certainly exists such a t' that $t' \succ t$ and $t' \notin G$. Therefore, we can form a k-expert team $G' = (G - t) \cup t'$. G' dominates G and thereby invalidates the assumption that G is a skyline team. Based on the proof, CrewScout remove those experts dominated by k or more experts. In practice, the remaining set of experts (denoted D' in Figure 1) is often orders of magnitude smaller than D.

Search Space Pruning Instead of enumerating each and every k-expert combination, we exclude from consideration a large number of combinations. To achieve that, we identify two properties inspired by the *anti-monotonic property* in the well-known Apriori algorithm [1] for frequent itemset mining. However, it is important to note that the original anti-monotonic property does not hold for skyline teams under AVG, MIN or MAX. More specifically, a subset of a skyline team may not be a skyline team itself. We identify a *order-specific anti-monotonic property*, a generic property that applies for AVG, MIN and MAX, and a *weak candidate-generation property* which applies for MIN and MAX but not AVG. The algorithms in CrewScout iteratively generate larger candidate teams from smaller ones and prune candidate teams by the two properties. Particularly, CrewScout uses a dynamic programming algorithm that applies the order-specific property and an iterative algorithm that leverages the weak candidate-generation property. Finally, CrewScout applies a conventional skyline algorithm over the candidate teams to find skyline teams.

4. DEMONSTRATION PLAN

An online demonstration of CrewScout is hosted at `http://idir.uta.edu/crewscout`. At this moment, the demo runs on a research-publication dataset. We are adding more datasets into the system and we plan to demonstrate it on at least two more datasets during the conference, including a question-answering dataset and an NBA dataset. The system will also eventually support user-uploaded datasets.

The research-publication dataset contains more than 900,000 publications collected through Microsoft Academic Search API. We use this dataset to demonstrate a paper reviewer selection scenario. The publications are review tasks and all the authors are potential reviewers. Each publication is labeled with several topic keywords (such as the ACM keywords), which represent the required expertise for reviewing the publication. Each candidate reviewer is modeled by an expertise vector, of which each attribute corresponds to a topic keyword and its value represents how many of the expert's publications are labeled with the corresponding topic keyword. The values are normalized so that the maximum value on each attribute is 100. We describe the demonstration steps as follows, with an imaginary user Amy.

(1) The task panel shows the available publications. Amy can also search for, say "database", and matching publications are displayed in the task panel. A default publication is highlighted and the corresponding required expertise is shown in the skill panel. The display panel presents the qualifying reviewers for that publication. Figure 2 illustrates this and the next several steps.

(2) Amy clicks a publication to highlight it and see its abstract. Amy clicks the same publication again and its abstract becomes hidden. Amy can do the same on other publications. Whenever a publication is selected, the skill and display panels are refreshed with the corresponding required expertise and qualifying reviewers, respectively. Amy checks/unchecks one or more skills, the qualifying reviewers in the display panel are also automatically refreshed.

(3) Amy filters the reviewers by setting minimum and/or maximum thresholds on one or more skills.

(4) In the parameter panel, a default aggregate function is selected and a default skyline team size is set. Amy can select a different aggregate function and a different team size.

(5) Once Amy clicks the "Skyline Teams" button, the display panel shows the skyline teams (Figure 3). Amy can filter them by reviewer name and minimum/maximum thresholds on aggregated skills.

(6) Amy may be satisfied with a particular team in the display panel. If not, Amy clicks the "Pick a Team" button and the visualization interface presents the reviewer clusters (Figure 4). Amy can also choose different clustering parameters and click the "Pick a Team" button again; the clusters are refreshed.

(7) Amy can move the mouse over the circles to see the reviewers' profiles and can also zoom in and out. Whenever Amy selects a reviewer, a polygon representing the reviewer's expertise is inserted into the polar-chart, the reviewer's name is added to the list below the polar-chart, and the clusters of the remaining qualifying reviewers are refreshed. Amy can remove a selected reviewer by clicking the cross sign beside the reviewer's name. The polar-chart and the clusters are refreshed to reflect the change.

(8) Amy repeats the above step multiple times until a k-reviewer team is formed.

(9) Amy clicks the "Pick Another Team?" button to select another team. In this way, Amy chooses multiple teams and compares them in the polar-chart.

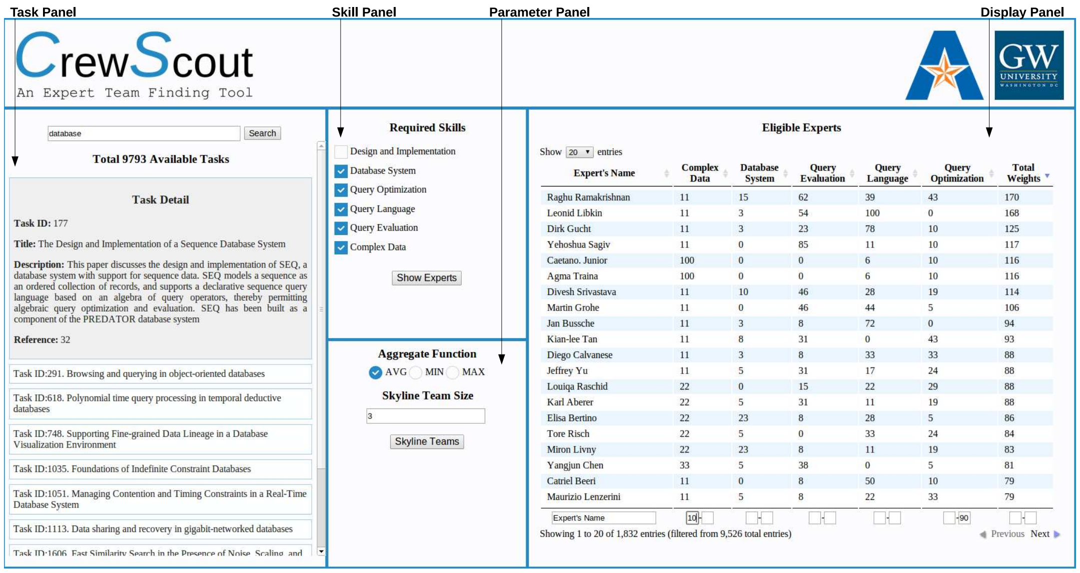
5. REFERENCES

[1] R. Agrawal, R. Srikant, et al. Fast algorithms for mining association rules. In *VLDB*, pages 487–499, 1994.

[2] S. Borzsony, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, pages 421–430, 2001.

[3] C. Li, N. Zhang, N. Hassan, S. Rajasekaran, and G. Das. On skyline groups. In *CIKM*, pages 2119–2123, 2012.

[4] N. Zhang, C. Li, N. Hassan, S. Rajasekaran, and G. Das. On skyline groups. *TKDE*, 26(4):942–956, April 2014.



database Search

Total 9793 Available Tasks

Task Detail

Task ID: 177

Title: The Design and Implementation of a Sequence Database System

Description: This paper discusses the design and implementation of SEQ, a database system with support for sequence data. SEQ models a sequence as an ordered collection of records, and supports a declarative sequence query language based on an algebra of query operators, thereby permitting algebraic query optimization and evaluation. SEQ has been built as a component of the PREDATOR database system

Reference: 32

Task ID:291. Browsing and querying in object-oriented databases

Task ID:618. Polynomial time query processing in temporal deductive databases

Task ID:748. Supporting Fine-grained Data Lineage in a Database Visualization Environment

Task ID:1035. Foundations of Indefinite Constraint Databases

Task ID:1051. Managing Contention and Timing Constraints in a Real-Time Database System

Task ID:1113. Data sharing and recovery in gigabit-networked databases

Task ID:1606. Fast Similarity Search in the Presence of Noise, Scaling, and

Required Skills

- ☐ Design and Implementation
- ✓ Database System
- ✓ Query Optimization
- ✓ Query Language
- ✓ Query Evaluation
- ✓ Complex Data

Show Experts

Aggregate Function

[✓ AVG] MIN MAX

Skyline Team Size

3

Skyline Teams

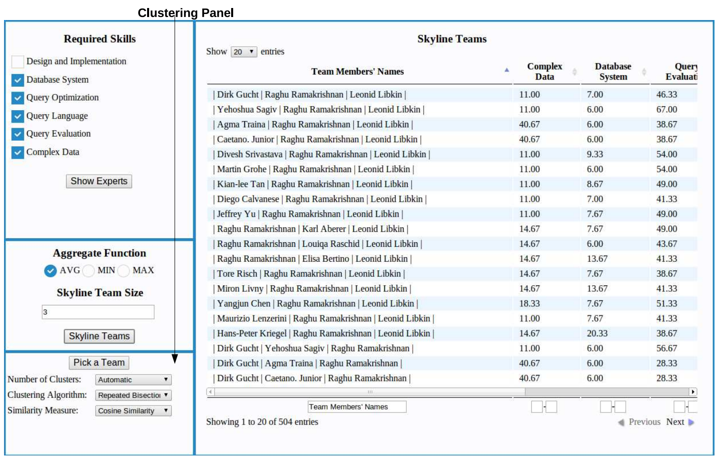
Eligible Experts

Show 20 entries

Expert's Name	Complex Data	Database System	Query Evaluation	Query Language	Query Optimization	Total Weights
Raghu Ramakrishnan	11	15	62	39	43	170
Leonid Libkin	11	3	54	100	0	168
Dirk Gucht	11	3	23	78	10	125
Yehoshua Sagiv	11	0	85	11	10	117
Caetano. Junior	100	0	0	6	10	116
Agma Traina	100	0	0	6	10	116
Divesh Srivastava	11	10	46	28	19	114
Martin Grohe	11	0	46	44	5	106
Jan Bussche	11	3	8	72	0	94
Kian-lee Tan	11	8	31	0	43	93
Diego Calvanese	11	3	8	33	33	88
Jeffrey Yu	11	5	31	17	24	88
Louiqa Raschid	22	0	15	22	29	88
Karl Aberer	22	5	31	11	19	88
Elisa Bertino	22	23	8	28	5	86
Tore Risch	22	5	0	33	24	84
Miron Livny	22	23	8	11	19	83
Yangjun Chen	33	5	38	0	5	81
Catriel Beeri	11	0	8	50	10	79
Maurizio Lenzerini	11	5	8	22	33	79
Expert's Name	10-	-	-	-	-90	-

Showing 1 to 20 of 1,832 entries (filtered from 9,526 total entries) Previous Next

Figure 2: CrewScout **Interface**



Required Skills

- ☐ Design and Implementation
- ✓ Database System
- ✓ Query Optimization
- ✓ Query Language
- ✓ Query Evaluation
- ✓ Complex Data

Show Experts

Aggregate Function

[✓ AVG] MIN MAX

Skyline Team Size

3

Skyline Teams

Pick a Team

Number of Clusters: Automatic

Clustering Algorithm: Repeated Bisectio

Similarity Measure: Cosine Similarity

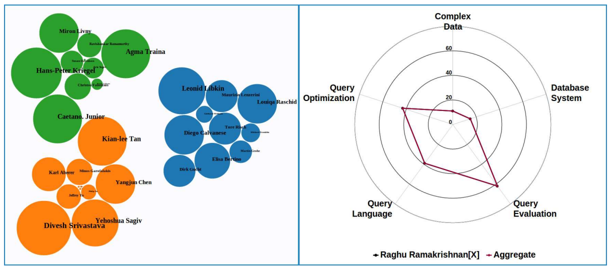
Skyline Teams

Show 20 entries

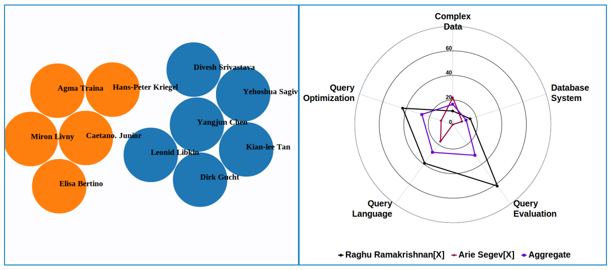
Team Members' Names	Complex Data	Database System	Query Evaluati
| Dirk Gucht | Raghu Ramakrishnan | Leonid Libkin |	11.00	7.00	46.33
| Yehoshua Sagiv | Raghu Ramakrishnan | Leonid Libkin |	11.00	6.00	67.00
| Agma Traina | Raghu Ramakrishnan | Leonid Libkin |	40.67	6.00	38.67
| Caetano. Junior | Raghu Ramakrishnan | Leonid Libkin |	40.67	6.00	38.67
| Divesh Srivastava | Raghu Ramakrishnan | Leonid Libkin |	11.00	9.33	54.00
| Martin Grohe | Raghu Ramakrishnan | Leonid Libkin |	11.00	6.00	54.00
| Kian-lee Tan | Raghu Ramakrishnan | Leonid Libkin |	11.00	8.67	49.00
| Diego Calvanese | Raghu Ramakrishnan | Leonid Libkin |	11.00	7.00	41.33
| Jeffrey Yu | Raghu Ramakrishnan | Leonid Libkin |	11.00	7.67	49.00
| Raghu Ramakrishnan | Karl Aberer | Leonid Libkin |	14.67	7.67	49.00
| Raghu Ramakrishnan | Louiqa Raschid | Leonid Libkin |	14.67	6.00	43.67
| Raghu Ramakrishnan | Elisa Bertino | Leonid Libkin |	14.67	13.67	41.33
| Tore Risch | Raghu Ramakrishnan | Leonid Libkin |	14.67	7.67	38.67
| Miron Livny | Raghu Ramakrishnan | Leonid Libkin |	14.67	13.67	41.33
| Yangjun Chen | Raghu Ramakrishnan | Leonid Libkin |	18.33	7.67	51.33
| Maurizio Lenzerini | Raghu Ramakrishnan | Leonid Libkin |	11.00	7.67	41.33
| Hans-Peter Kriegel | Raghu Ramakrishnan | Leonid Libkin |	14.67	20.33	38.67
| Dirk Gucht | Yehoshua Sagiv | Raghu Ramakrishnan |	11.00	6.00	56.67
| Dirk Gucht | Agma Traina | Raghu Ramakrishnan |	40.67	6.00	28.33
| Dirk Gucht | Caetano. Junior | Raghu Ramakrishnan |	40.67	6.00	28.33
Team Members' Names	-	-	-

Showing 1 to 20 of 504 entries Previous Next

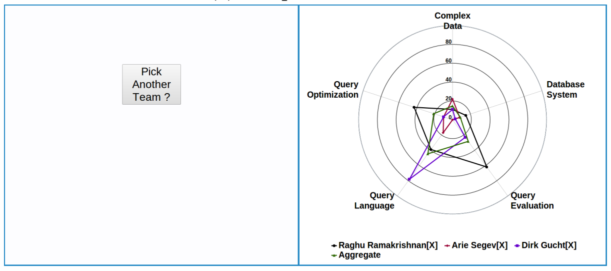
Figure 3: Display Panel Showing Skyline Teams



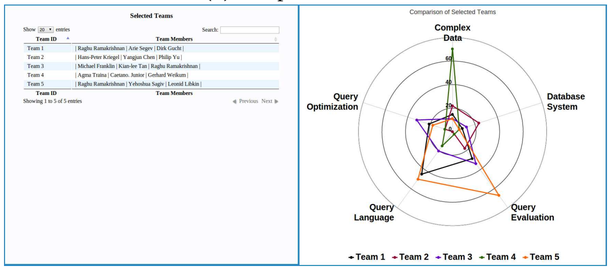
(a) 1 expert selected



(b) 2 experts selected



(c) 3 experts selected



Selected Teams

Show 20 entries Search:

Team ID	Team Members
Team 1	| Raghu Ramakrishnan | Arie Segev | Dirk Gucht |
Team 2	| Hans-Peter Kriegel | Yangjun Chen | Philip Yu |
Team 3	| Michael Franklin | Kian-lee Tan | Raghu Ramakrishnan |
Team 4	| Agma Traina | Caetano. Junior | Gerhard Weikum |
Team 5	| Raghu Ramakrishnan | Yehoshua Sagiv | Leonid Libkin |
Team ID	Team Members

Showing 1 to 5 of 5 entries Previous Next

(d) Selected skyline teams

Figure 4: Selecting and Comparing Teams