# 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 a novel expert team finding system based on the idea of finding *Skyline Teams* [1, 2] from an input dataset. In this paper, we give a comprehensive overview of CrewScout system, present its GUI and demonstrate its usability. We show how it can be used in a paper-reviewer-recommendation scenario and also explain its potential usage in a wide variety of real-world applications.

## 1. OVERVIEW

This paper introduces CrewScout, a system for finding expert teams in accomplishing tasks. The foundation of the system is the concept of *skyline teams* (called *skyline groups* in [1, 2]). The new contributions made in this paper include an end-to-end system with intuitive user interface, an interactive tool to assist users in choosing teams among (potentially many) skyline teams, and an extension of application and demonstration scenarios into more general areas ([1, 2] mostly focused on the application of forming teams in sports fantasy games.)

Consider a set $D$ of $n$ experts $t_1, \ldots, t_n$, each of which is modeled by $m$ numeric attributes $A_1, \ldots, A_m$ that represent their skills and expertises. Any subset of $k$ experts form a *$k$-expert team*. CrewScout finds, for a given $k$, all $k$-expert skyline teams (denoted $S_k(D)$), 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 [3]. We calculate for each team a single aggregate vector, whose attribute values are aggregated over the corresponding attribute values of the experts in the team. Commonly used aggregate functions are AVG (i.e, SUM, since teams are of 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.

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 topics *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 this editing task to a team of 2 editors. Table 1.2 shows the aggregate vectors under AVG, MIN and MAX aggregate functions, for all possible 2-expert teams ($G_{i,j}$ standing for a team of experts $t_i$ and $t_j$). A simple team formation scheme, such as picking top editors of individual topics, does not work. Specifically, the team $G_{1,2}$, which consists of the top editor on each topic, has an aggregated vector $\langle 1.5, 1.5\rangle$ with regard to AVG. Another team, $G_{3,4}$, with vector $\langle 2.0, 1.5\rangle$, dominates $G_{1,2}$, i.e., $G_{3,4} \succ G_{1,2}$ 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.

| expert | *database* | *indexing* |
|---|---|---|
| $t_1$ | 3 | 0 |
| $t_2$ | 0 | 3 |
| $t_3$ | 2 | 1 |
| $t_4$ | 2 | 2 |
| $t_5$ | 0 | 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**

*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 using virtual currency. 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 job? 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 choos-

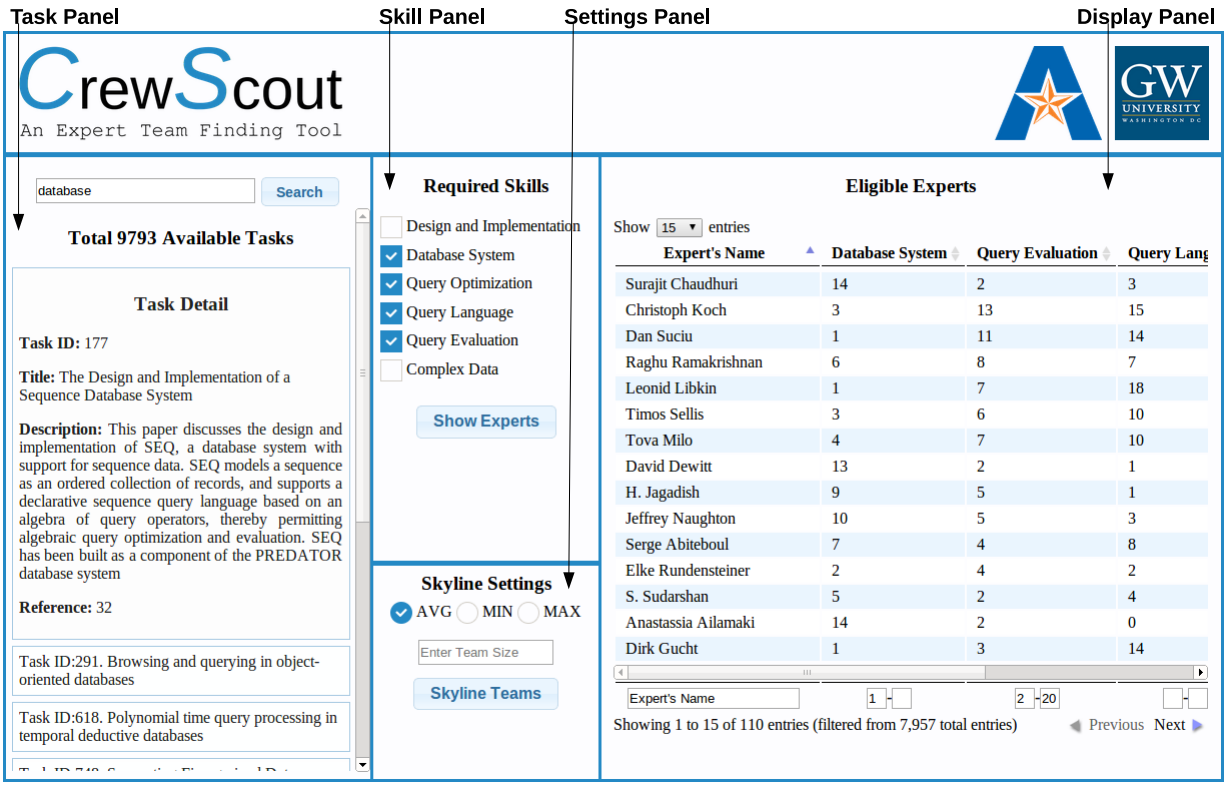



**Figure 1:** CrewScout **Interface**

ing 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 reect 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 captured by attributes (dimensions) such as *sentiment, length, quality*, etc. Another example of applicable 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.

The capability of recommending expert teams is valuable in all above motivating applications. An attractive property of the concept of *skyline teams* is that a skyline team cannot be dominated by any other team of equal size. In contrast, given a non-skyline team, there always exists a better team belonging to the skyline. This property distinguishes skyline-team from other team recommendation techniques. The skyline teams solely consist of the teams which are worth recommending. They become the input to further (manual or automated) processes that ultimately find one team. Examples of such post-processing include eye-balling the skyline teams, more systematic browsing and visualization of the skyline teams, and filtering and ranking them by user preferences.

Admittedly, determining the "best" team can be an extremely complex task which involves many more factors than what skyline teams can capture—e.g., whether *Wikipedia* editors or paper reviewers are available for the assigned work, whether they have good relationship to work together, and so on. Hence, CrewScout finds all skyline teams and provides an interactive tool to assist a human user in exploring the skyline teams and choosing among them her desired team.

## 2. USER INTERFACE

In this section, we explain various functionalities of CrewScout system as well as provide an overview of the user interface. In a nutshell, CrewScout starts by providing option to a user to search for her task of interest. After the user has found a task, CrewScout 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 one team from them. Figure 1 shows the GUI of CrewScout. Overall, we divide the full interface into multiple panels. Below, we describe each of them.

**Task Panel** CrewScout presents a list of available tasks in this panel. When a user clicks over a task, CrewScout provides more detail about it. CrewScout also provides a search box at the top of this panel for searching from available tasks. A user can provide keywords and CrewScout will fetch matching tasks with provided keywords.

**Skill Panel** This panel presents the skills which are required to complete the task selected in *Task Panel*. CrewScout presents a checkbox for each of the skills. By default, all the checkboxes are


**Figure 2: Flow of Algorithm**

checked. The user can check/uncheck some of them according to her preference. The *Show Experts* button allows to see the experts who have expertise in at least one of the checked skills.

**Settings Panel** This panel provides options to set parameters for skyline team computation. There is a radio button for each of the aggregate functions (AVG, MIN, MAX) and a textbox for entering the skyline team size.

**Display Panel** CrewScout uses this panel for displaying multiple types of information. Below, we explain each of them.

*Experts:* When a user selects a task in *Task Panel*, CrewScout presents a table of eligible experts in *Display Panel*. Each of these experts has expertise in at least one of the checked skills in the *Skill Panel*. When the user checks/unchecks some of the skills, the table gets refreshed to reflect that change. Experts in the table are ranked by the summation of expertise in each of the selected skills. CrewScout provides a filter for each of the skills under the corresponding column in *Eligible Experts* table. The user can set the minimum and maximum range of the level of expertise for each skill. User can also use the filter under "Expert's Name" column to search for experts by name.

*Skyline Teams:* CrewScout presents the list of skyline teams in *Display Panel* when a user clicks the *Skyline Teams* button after setting parameters as shown in Figure 3. CrewScout also provides a filter for each column in *Skyline Team* table. These filters help a user to digest large number of skyline teams and focus over her preferred region. We denote the output of these filters as *filtered skyline teams*.

*Visualization:* When a user clicks the "Pick One Team" button of Figure 3, CrewScout visualizes the experts in *Display Panel* using circles as shown in Figure 4(a). The experts who were member of at least one of the *filtered skyline teams*, are shown in the visualization. The experts are clustered using *K-means clustering* algorithm by leveraging *Euclidean distance* between pairs as similarity. We define number of clusters as $\sqrt{n/2}$ using rule of thumb [4]. We set the size of a circle based on how frequent the corresponding expert is in the *filtered skyline teams*. In other words, experts with big circles belong to more *filtered skyline teams*. Initially, CrewScout shows labels of the big circles and allows to zoom-in to see the labels of the small circles. Each time a user selects an expert, CrewScout removes circles whose corresponding experts do not form any skyline team with the selected expert. Then, it clusters the experts and resizes the corresponding circles again. CrewScout shows a polar-chart which visualizes the selected experts' expertise over different skills. It also shows aggregation of the expertise by the function selected in *Settings Panel*. Once the user has selected $k$ experts, she has reached to a skyline team. User can also remove an expert from selection by clicking the cross button beside the expert's name and go back to previous step. By this way, CrewScout guides a user to explore and select experts and at the same time pick a skyline team from many skyline teams.

## 3. ALGORITHMS

One seemingly possible solution to find $S_k(D)$ is to enumerate each possible combination of $k$ experts in $D$, compute the aggregate vector for each combination, and then invoke a traditional skyline-search algorithm to find all skyline teams. This approach

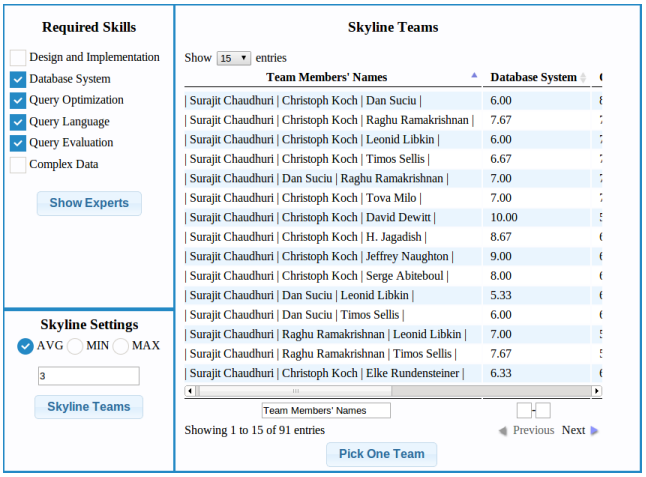


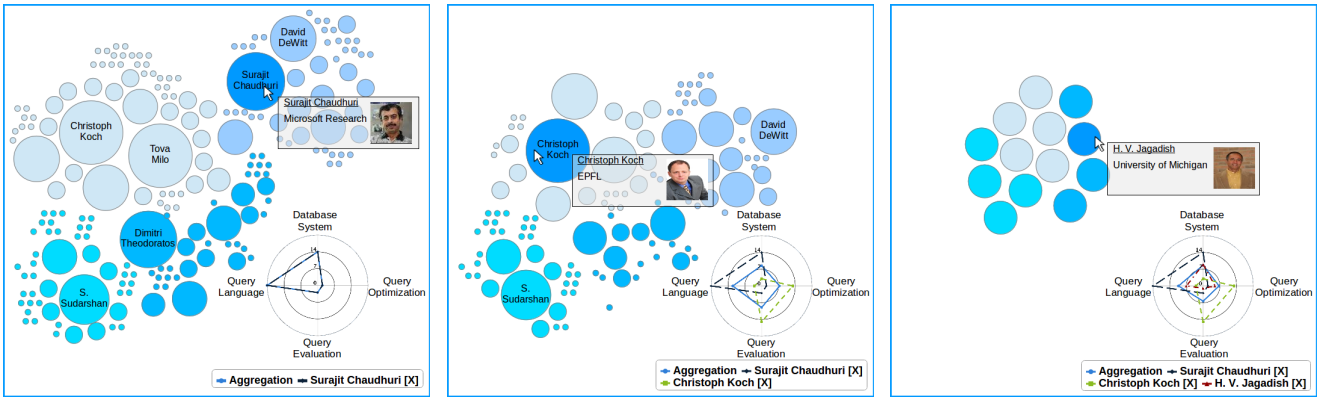

**Figure 3: Display Panel Showing Skyline Teams**

has two main problems. One is its significant computational overhead, as the input size to the final step, i.e., skyline expert search, is $\binom{n}{k}$, which can be extremely large. The other problem is on the seemingly natural strategy of listing 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 produced, may be nevertheless too large to explicitly compute and store.

To address these challenges, we develop several techniques. We start with explaining the *Output Compression* technique that significantly reduces the output size when the number of skyline teams is large. Then, we describe how to efficiently find skyline teams using *Input Pruning* and *Search Space Pruning* techniques. Figure 2 shows flow of our algorithm.

**Output Compression** A key observation is while the number of skyline teams may be large, many of these skyline teams share the same aggregate vector. For example in Table 1.2, $G_{2,4}, G_{2,5}$ and $G_{4,5}$ have the same aggregate vector $\langle 0, 2\rangle$ under MIN aggregation function. Thus, our idea for compressing skyline teams is not to store all skyline teams, but only the (much fewer) distinct skyline aggregate vectors (in short, skyline vector) as well as one skyline team for each skyline vector. 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 her knowledge and preference. So, we have techniques to reconstruct all skyline teams from a given skyline vector. Details of the techniques can be found in [1, 2].

**Input Pruning** An important observation is, if a expert $t$ is dominated by $k$ or more experts in the original table, then we can safely exclude $t$ from the input without influencing the distinct skyline vectors found at the end. We can prove this by contradiction. Assume, $G$ be a $k$-expert skyline team containing expert $t$ and $t$ is dominated by $k$ or more experts. Now, as $|G| = k$, it is always possible to find a $t'$ such that $t' \succ t$ and $t' \notin G$. So, we can form a $k$-expert team $G'$ such that $G' = (G - t) \cup t'$. This $G'$ will dominate $G$ and thereby invalidate our assumption that $G$ is a




(a) 1 expert selected (b) 2 experts selected (c) 3 experts selected

**Figure 4:** CrewScout **Interface: Picking One Skyline Team by Exploring Experts**

skyline team. We leverage this property and prune the experts from $D$ which are dominated by $k$ or more other experts. We find that, this pruning is very efficient in practice and significantly reduces the input size, sometimes by order of magnitude. Figure 2 shows $D'$ as the reduced input after *Input Pruning*.

**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 for frequent item set mining. However, it is important to emphasize here that the anti-monotonic property in Apriori does not hold for skyline teams defined by AVG, MIN or MAX. More specifically, a subset of a skyline team may not necessarily be a skyline team itself. Thus, we identify (a) Order-Specific Anti-Monotonic Property, a generic property that applies to AVG, MIN and MAX, and (b) Weak Candidate-Generation Property which applies to MIN and MAX but not AVG. Based on the two properties, we develop algorithms to compute skyline teams. These algorithms iteratively generate larger candidate teams from smaller ones, and prune candidate teams by these properties. In particular, we develop a dynamic programming algorithm that applies the order-specific property, and an iterative algorithm that leverages the weak candidate-generation property. Finally, we apply traditional skyline algorithm over the candidate teams to find $S_k(D)$.

## 4. DEMONSTRATION PLAN

We will demonstrate CrewScout on several datasets, including a Research article dataset and an NBA dataset. The Research article dataset is collected using Microsoft Academic Search API. It contains information of more than 900,000 research articles. Note that, even though we demonstrate for reviewer selection application from a conference manager's point of view, CrewScout is not by any means bounded for this application only and can be applied for any general team recommendation problem.

**Step 1:** Initially, CrewScout shows a list of available tasks in the *Task Panel*. By default, the first task is selected and its required skills are shown in the *Skill Panel*. The *Display Panel* shows a list of available experts for that task.

**Step 2:** Suppose the manager is interested to see the tasks related to *"database"*. She types *"database"* in the *search box* and click the *Search* button. CrewScout shows a list of tasks matching with the keyword *"database"*.

**Step 3:** Suppose the manager is interested to find 3 experts to complete the first task. She clicks on the task. CrewScout provides more detail about it and refreshes the *Skill* and *Display* panels accordingly. She checks/unchecks the skills according to her preference. CrewScout refreshes the list in *Display Panel*. She further filters the experts by setting minimum or maximum threshold in any skill. She enters *3* as number of required experts in *Team Size* box, chooses one of the aggregate functions (AVG/ MIN/ MAX) and clicks the *Skyline Teams* button.

**Step 4:** CrewScout refreshes the *Display Panel* and shows a list of skyline teams as depicted in Figure 3. The manager can pick any team of experts from this list for the task. She can also filter the teams by expert name or by setting minimum or maximum threshold for a skill.

**Step 5:** Suppose the manager wants to pick a single skyline team. She clicks the *Pick One Team* button. CrewScout presents a visualization of the experts as shown in Figure 4(a). She hovers the mouse over a circle; CrewScout shows profile of the corresponding expert in a small pop-up window and presents expertise in a polar-chart. She selects one circle; CrewScout removes irrelevant circles accordingly.

**Step 6:** Suppose the manager needs to unselect a selected expert. She clicks the cross sign beside the expert's name.

**Step 7:** She repeats **Step 5** and **Step 6** multiple times until she has found her preferred $k$-expert team.

## 5. REFERENCES

[1] C. Li, N. Zhang, N. Hassan, S. Rajasekaran, and G. Das, "On skyline groups," in *CIKM*, 2012.

[2] N. Zhang, C. Li, N. Hassan, S. Rajasekaran, and G. Das, "On skyline groups," *TKDE*, vol. 99, no. PrePrints, 2013.

[3] S. Borzsony, D. Kossmann, and K. Stocker, "The skyline operator," in *ICDE*, 2001.

[4] K. V. Mardia, J. T. Kent, and J. M. Bibby, "Multivariate analysis," 1980.