# Using Hierarchical Skills for Optimized Task Selection in Crowdsourcing

Panagiotis Mavridis

ANNÉE 2017

UNIVERSITÉ DE RENNES 1

UNIVERSITE BRETAGNE LOIRE

**THÈSE / UNIVERSITÉ DE RENNES 1**
*sous le sceau de l'Université Bretagne Loire*

pour le grade de

**DOCTEUR DE L'UNIVERSITÉ DE RENNES 1**

*Mention : Informatique*

**École doctorale MATHSTIC**

présentée par


# Panagiotis Mavridis


préparée à l'unité de recherche IRISA – UMR6074
Equipe d'accueil : IRISA/DRUID
Contrat MESR

---

# Using Hierarchical Skills for Optimized Task Selection in Crowdsourcing

**Thèse à soutenir à Rennes**
**le 17-11-2017**

**Alessandro Bozzon**
Maître de conférences à l'Université de Delft, Delft, Pays-bas / *Rapporteur*

**Sihem Amer-Yahia**
Directrice de recherche CNRS, Grenoble, France / *Rapporteuse*

**Anne-Marie Kermarrec**
Directrice de Recherche Inria, Rennes, France / *Examinatrice*

**Gianluca Demartini**
Maître de conférences à l'Université de Queensland, Brisbane, Australie / *Examinateur*

**Phillipe Cudré-Mauroux**
Professeur à l'Université de Fribourg, Fribourg, Suisse / *Examinateur*

**Zoltán Miklós**
Maître de conférences à l'Université de Rennes 1, Rennes, France / *Examinateur*

**David Gross-Amblard**
Professeur à l'Université de Rennes 1, Rennes, France / *Directeur de thèse*

*"I know one thing that I know nothing."*
*- "Socrates"*

*"Everyone is a genius. But if you judge a fish by its ability to climb a tree, it will live its whole life believing that it is stupid."*
*- Credited to "Albert Einstein" but Unknown quoter*

## Acknowledgments

I would like to thank Alessandro Bozzon, Assistant Professor at the University of Delft and Sihem Amer-Yahia, Research Director at CNRS, for accepting the task of reviewing my thesis.

Next, I would like to thank Anne-Marie Kermarrec, Research Director at INRIA, Gianluca Demartini, Senior Lecturer at the University of Queensland and Phillipe Cudré-Mauroux, Professor at the University of Fribourg, for accepting to judge this work.

Then, I would like to thank David Gross-Amblard, Professor at the University of Rennes 1 that directed my thesis and Zoltan Miklos, Assistant Professor at the University of Rennes 1, for supervising my thesis. From the early beginning, they integrated me into their research group, team DRUID, but most importantly provided me with the essential tools to perform research. They also shared with me in abundance their methodologies and skills during our meetings. I will definitely not forget our first late night full paper submission.

I would also like to thank all the members of team DRUID both in Rennes and Lannion for their kindness and precious feedback. Without their support, this work would have been unbearable. I also appreciated very much Team SemLIS and want to thank all of them separately for their support. Especially I would like to thank Pierre Maillot and Carlos Bobed-Lisbona for their scientific advice and feedback.

In addition, I would like to thank my friends Hristina Hristova and Ivaylo Petrov for re-initiating me to a balanced and sporty lifestyle and cheering me up during my ups and downs. Hristina, as a Ph.D. student she could easily sympathize with me. Also, I would like to thank my friend Soda Cissé for our culinary investigations in Les Délices D'Asie!

Moreover, I thank Eddy Maddalena and Alessandro Checco for our adventures and late night (online and offline) existential talks at Sheffield. I also thank Victorien Delannée and Orçun Yildiz for the adventures and the friendly discussions that we shared while in Rennes. Both being Ph.D. students at the same time with me at IRISA and INRIA (respectively), we shared similar challenges to tackle and we understood each other very well.

I want to also mention the financial help from the French government. During all the period of my thesis I have been supported by a MESR grant from the French ministry of Education which was generous enough for me to perform my research and live in France. Also, I would like to thank EIT Digital for financing my international mobility and entrepreneurial seminars both within France and outside France. Finally, I would like to thank ANR project HEADWORK for taking charge of my salary for the

last two months of my research and letting me also participate into several meetings within France.

Last but not least, I would like to thank my beloved parents Michail and Foteini. Without them, I would not be able to even start a Ph.D. thesis. I thank them for putting up with me all this time and supporting me both psychologically and financially from my early childhood up to my early adulthood.

# Table of Contents

# Chapter 1

# Introduction

Crowdsourcing in the form of participative platforms is an overwhelming subject both in Academia [52, 19, 64, 54, 82] and Industry [36, 84, 81]. In this chapter, we will :

— define crowdsourcing,
— introduce crowdsourcing examples,
— motivate the research challenges for modern crowdsourcing platforms in comparison to the related work and
— present our contributions that can improve crowdsourcing and the future market of digital labor.

## 1.1 Crowdsourcing Definition and Types

According to Jeff Howe who came up with the term, crowdsourcing is [51] :

**Definition 1.1.1** (CROWDSOURCING DEFINITION ACCORDING TO JEFF HOWE)

*"Simply defined, crowdsourcing represents the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call. This can take the form of peer-production (when the job is performed collaboratively) but is also often undertaken by sole individuals. The crucial prerequisite is the use of the open call format and the large network of potential laborers."*

The name is a combination of the words crowd and outsourcing and thus it is borrowing characteristics from both terms. As described by Jeff Howe [52] crowdsourcing can be seen as leveraging humankind skills with a perfect, transparent and democratic manner to open up and solve difficult problems.

However, after studying more than *40 definitions* on crowdsourcing two researchers from the Technical University of Valencia came up with a more complete definition [38] :

**Definition 1.1.2** ([38])

*"Crowdsourcing is a type of participative online activity in which an individual, an institution, a nonprofit organization, or company proposes to a group of individuals of varying knowledge, heterogeneity, and number, via a flexible open call, the voluntary undertaking of a task. The undertaking of the task of variable complexity and modularity, and in which the crowd should participate, bringing their work, money, knowledge and/or experience, always entails mutual benefit. The user will receive the satisfaction of a given type of need, be it economic, social recognition, self-esteem, or the development of individual skills, while the crowdsourcer will obtain and use to their advantage that which the user has brought to the venture, whose form will depend on the type of activity undertaken."*

With the use of Internet and crowdsourcing platforms, it is now possible to reach out to connected people with rare skills and gather a wise crowd with combined intelligence. Potential impact has been envisioned for problems that include and are not limited to a cure for cancer [90], protein folding [64, 4], HIV [1], participative democracy, transparency of procedures, open government [2], etc. The applications are numerous and there are many challenges on how to find the hidden talent and the best quality of data that can lead to the best possible results as soon as possible.

However, we can classify crowdsourcing into two main types : explicit and implicit crowdsourcing [34]. Simply put, in "explicit" crowdsourcing participants, contributors of answers for crowdsourcing platforms, are invited to select from a list of numerous tasks to perform, know exactly the task that they work for and they have a clearly stated incentive (monetary or other). Also, the result of the task is clear and the requester of the task should not hide third-party tasks. For instance, tasks posted on Amazon MTurk are a form of explicit crowdsourcing. On the contrary, "implicit" crowdsourcing denotes the scenario where participants do not know that they are working for a task but still contribute for other implicit incentives that have nothing to do with the task. For instance, "reCAPTCHA" is an example of implicit crowdsourcing where participants have to identify images of a certain type or letters of a certain type to get access to a page. The goal of the task is to train a classification system, to which the participants have no access. Our work can be applied to both but here we mainly study explicit crowdsourcing.

## 1.2 First Crowdsourcing Examples

Since the arrival of Web 2.0, online users can use online platforms to express their opinions, comment on products they have bought and taken part in online social groups. Current online participative platforms have evolved though in different forms,

1. http ://2beathiv.org/
2. http ://democracyos.org/

such as structured forums [3] or social networks [4], etc. and now involve the crowd, apart from informal discussions too, data disambiguation, consulting, decision-making processes, calculations, data analysis and more.

Even before scientific applications, in 2005 [84], crowdsourcing had become responsible for innovative product design. As an example, the company Threadless [5] is an online platform with a few employees that leverages the power of the crowd. The main business of Threadless is to invite participants to create T-shirt designs, compete on the designs and then print and sell the winning T-shirts online. Instead of hiring designers or freelancers they give their participants the platform to channel their own designs, receive and send votes for the best design, qualify for different rounds and reward the top three winners of the contest and print the winning T-shirts. The winners get different prizes which are both monetary and material (several T-shirts from their own design). What is very innovative about this process is to obtain a product design that has a proven popularity, scale its production and sell it. This helps to find both a crowd to design the T-shirt but also a crowd to buy it. Similar techniques have been used from companies, such as Ford Motor Company [36] or Newcastle Brewery [81], to crowdsource their marketing campaigns for their new products, which made the products and their campaigns already popular prolonging the attention of the public for the new product and also saving money from paying marketing experts.

The next step, in 2010 [64], was to render this mass of talent that is available online at the disposal of scientists. That resulted in a noteworthy application of citizen science, Foldit [4]. Scientists thought to create a game platform for protein folding to give the opportunity to participants to find new foldings of proteins. The platform was a very simple and well structured online interface, as seen in Figure 1.1 that guided participants into folding different proteins through a game experience (with rankings, competitions, etc.). This experiment was a total success [26] since participants found new foldings of proteins that expert biologists could not find before. The result of this application led to more than one publication in the *Nature* journal [26, 54]. Next, since we presented specific crowdsourcing applications, we will go on with more generic applications.

## 1.3 Generic Crowdsourcing Platforms

Crowdsourcing methodology is also made available to the public through World Wide Web for generic use. Thus the first two generic microtask crowdsourcing platforms were created : Amazon MTurk [6] and CrowdFlower [7].

Amazon MTurk is a microtask crowdsourcing platform that surpassed 500k tasks and 250k participants [8, 49]. In this platform, requesters can publish microtasks with

3. www.stackoverflow.com
4. www.linkedin.com
5. www.threadless.com
6. www.mturk.com
7. www.crowdflower.com

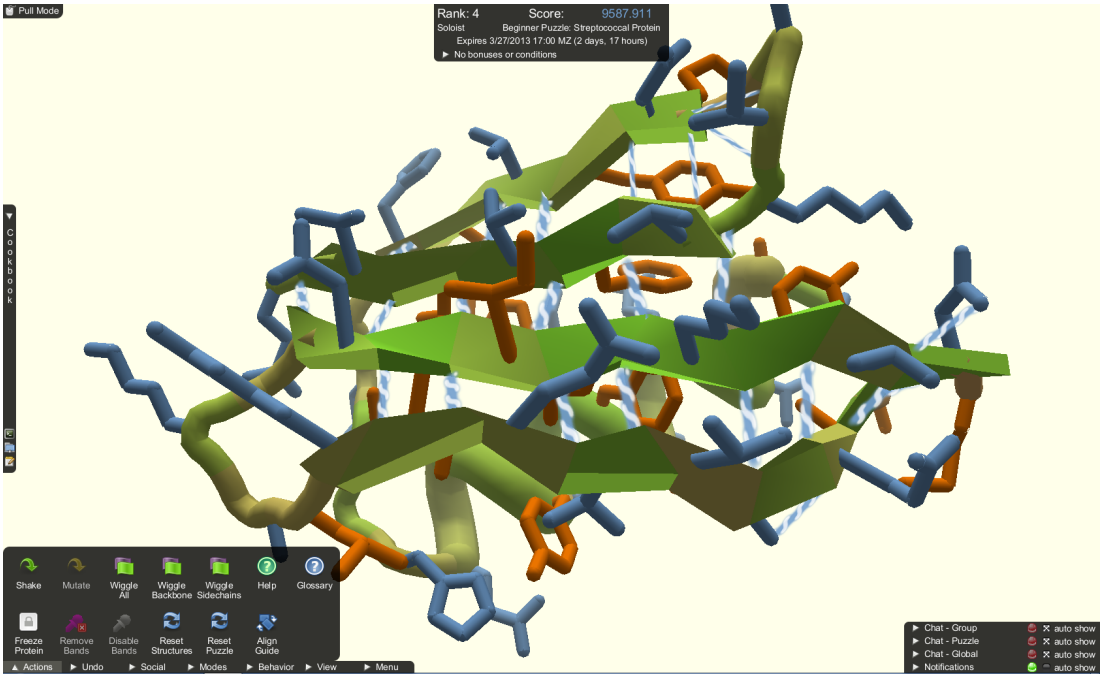



FIGURE 1.1 – Screenshot from FoldIt Interface

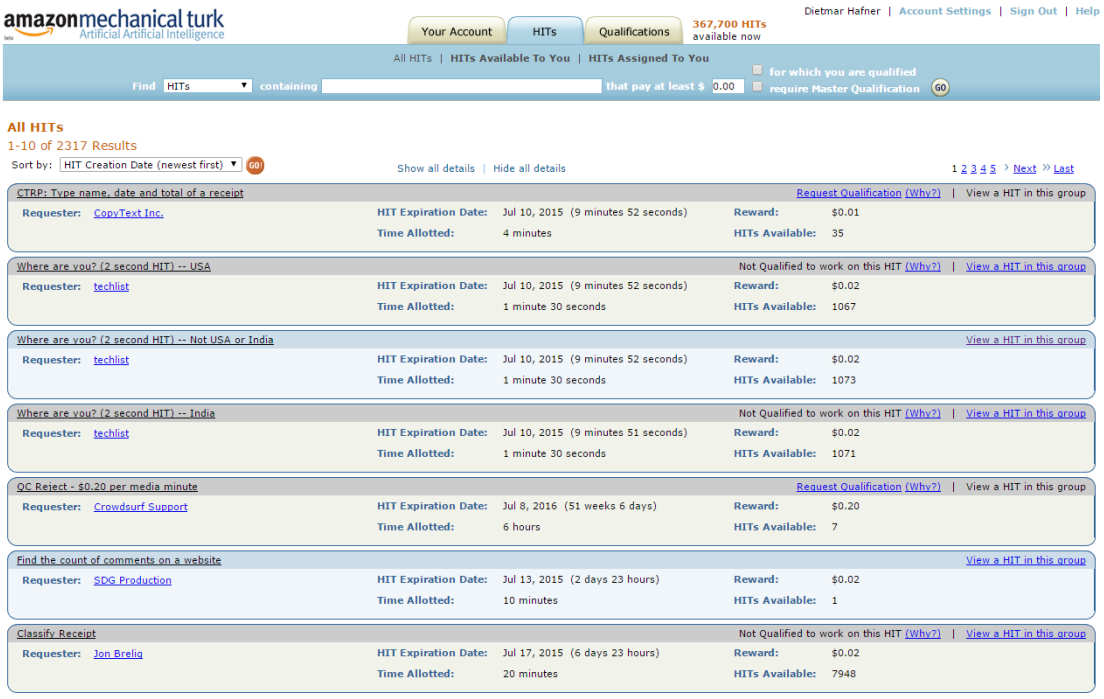



FIGURE 1.2 – Screenshot from Amazon Mturk Interface

a title, detailed description, data (optional), a question, a reward, an expiration date and a number of desired answers from the crowd. Once a participant accepts a task, he has to perform or discard it. All tasks are open to participants with the exception of tasks that need a certificate. Also, some tasks are open only to subscribed participants. An example of tasks in Amazon MTurk can be seen in Figure 1.2.

CrowdFlower [1] is similar to Amazon MTurk but less popular. The main differences, that make CrowdFlower particularly interesting are its user-friendly interface, its crowd, and its built-in features. Participants are ranked from level 0 to level 3, depending on their involvement in the platform and on the tasks that they have performed successfully. Figure 1.3 depicts how these features are combined in their interface. Another three existing generic crowdsourcing platforms are the French platform FouleFactory [5], the Portuguese DefinedCrowd [3] and the Greek Crowdpolicy [2]. For now, we have defined crowdsourcing, underlined its importance and its popularity. In the next section, we will identify current challenges for crowdsourcing systems.

## 1.4 Current Challenges

We mentioned several platforms and cases where crowdsourcing systems have been deployed either for industrial or academic use. For these systems, there are many

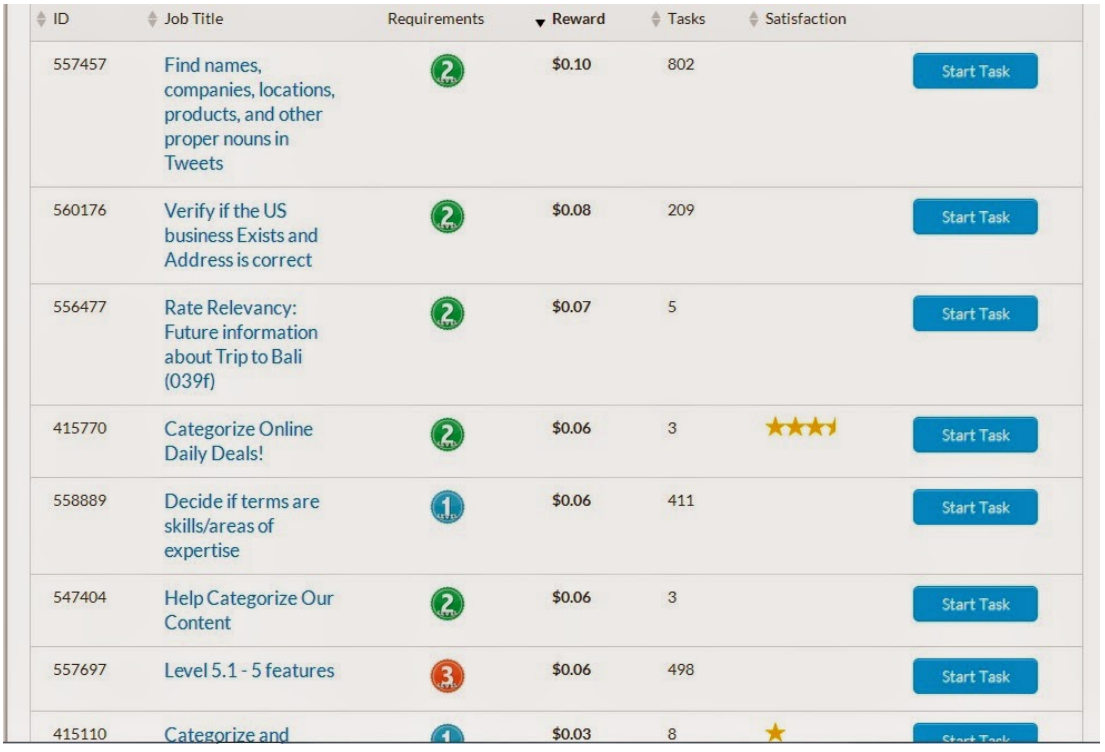



FIGURE 1.3 – Screenshot from CrowdFlower Interface

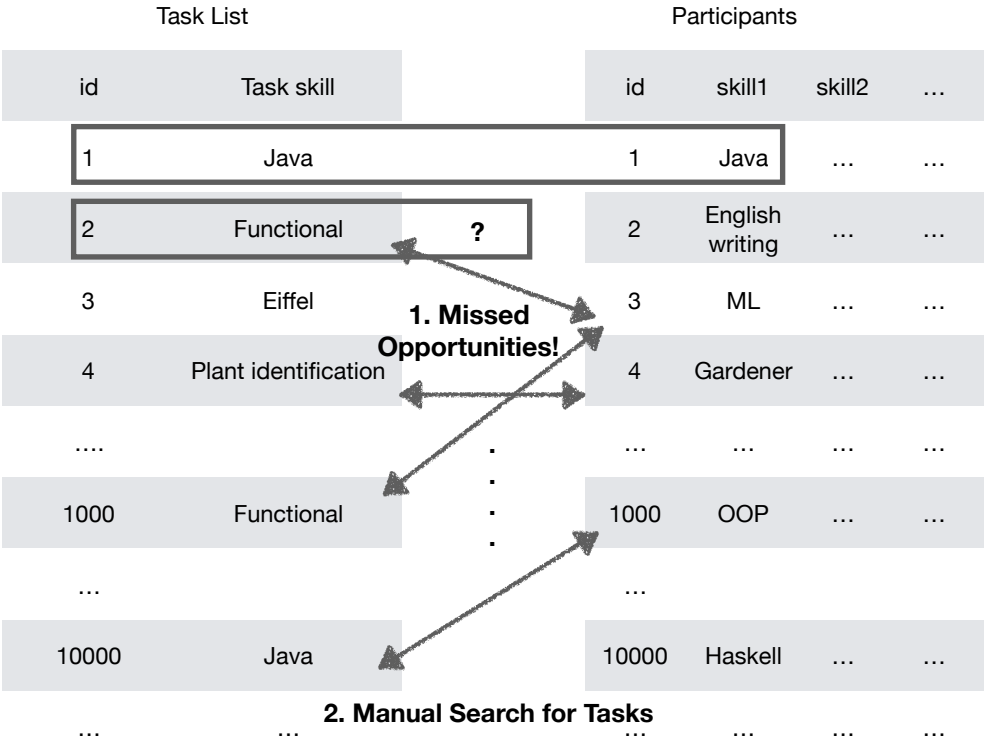



FIGURE 1.4 – Crowdsourcing Challenges

difficulties that arise such as **how to identify, recruit and retain participants, how to distribute relevant tasks to corresponding participants and how to evaluate and pay participants**. Moreover, existing systems lack of generality since most of the time a new platform has to be deployed for a different use.

The current situation in crowdsourcing platforms such as Amazon MTurk and CrowdFlower is depicted in Figure 1.4. Participants having different and diverse skills connect to a platform, but platforms do not support the notion of skills. However, the related work that supports skills ([18, 79]), does not support skill substitution. In the current platforms, participants have to select a task of their taste, given a search bar and several sorting criteria (reward, title, etc.) as seen in Figure 1.4. This results in lost time from creative work since participants take time to select a relevant task or choose according to their best interests.

Concerning the problem of recruiting, identifying and retaining proper participants there is a certain challenge to model the participants with respect to their ability to fulfill a task. In current industrial platforms, participants do not seem to be described in terms of skills. On the other hand, the related academic work does use tags or keywords to describe tasks and participants. However, this keyword representa-

tion does not allow us to substitute similar skills when the names are not similar. For instance, a keyword similarity cannot capture "Programming" and "Java" as related terms. Imagine then a Programming task and a Java programmer, we should be able to assign the "programming" task to the "Java" programmer since a Java programmer is still a programmer. However, this is not possible with a simple keyword matching. The problem is even more prominent in more complex settings where keyword names have no obvious relation.

In order to reassure proper distribution of relevant tasks to corresponding participants, it is a challenge to provide with an optimization method that maps tasks to participants. A given task should be attributed to the correct crowd in order to optimize available time and skills. Now participants have to choose on their own the tasks or even rely on less modern and less structured forums [8] where other participants propose interesting and well-paid tasks.

Challenge 1 : Skill modeling that enables skill substitution and task-to-participant matchings.

Moreover, task lists in modern crowdsourcing platforms can have several thousands of tasks and participants (as seen in Figure 1.4, Figure 1.2 and also mentioned in [82]). In these platforms, participants choose from a huge list of tasks with the help of some simple sorting criteria and a search bar to look for keywords. The challenge here is to be able to help participants with selecting tasks in the form of task propositions. This proposition, however, should be created with the right criteria to keep the motivation of participants and take into account task and participant characteristics.

Challenge 2 : Presenting each participant with a short list of personalized tasks out of a large list of available tasks.

## 1.5 Thesis Contributions

In this work, we mainly focus on a type of tasks called microtasks, tasks requiring only one skill and one can find and work on platforms such as CrowdFlower and Amazon MTurk. Then, our focus is on participant modeling and microtask skill management in crowdsourcing applications which we do by addressing the two main challenges that we mentioned in the previous section (Section 1.4). However, the contributions that we will present that address the above challenges are more generic and can be valuable in more demanding environments and in different platforms such as Wikipedia, Quora [9], Foulefactory, Stack Overflow [10] etc.

Concerning the first challenge, in order to be able to reason and substitute competencies in participative platforms such as crowdsourcing platforms we propose a skill management system :

8. http ://turkernation.com/
9. www.quora.com
10. www.stackoverflow.com

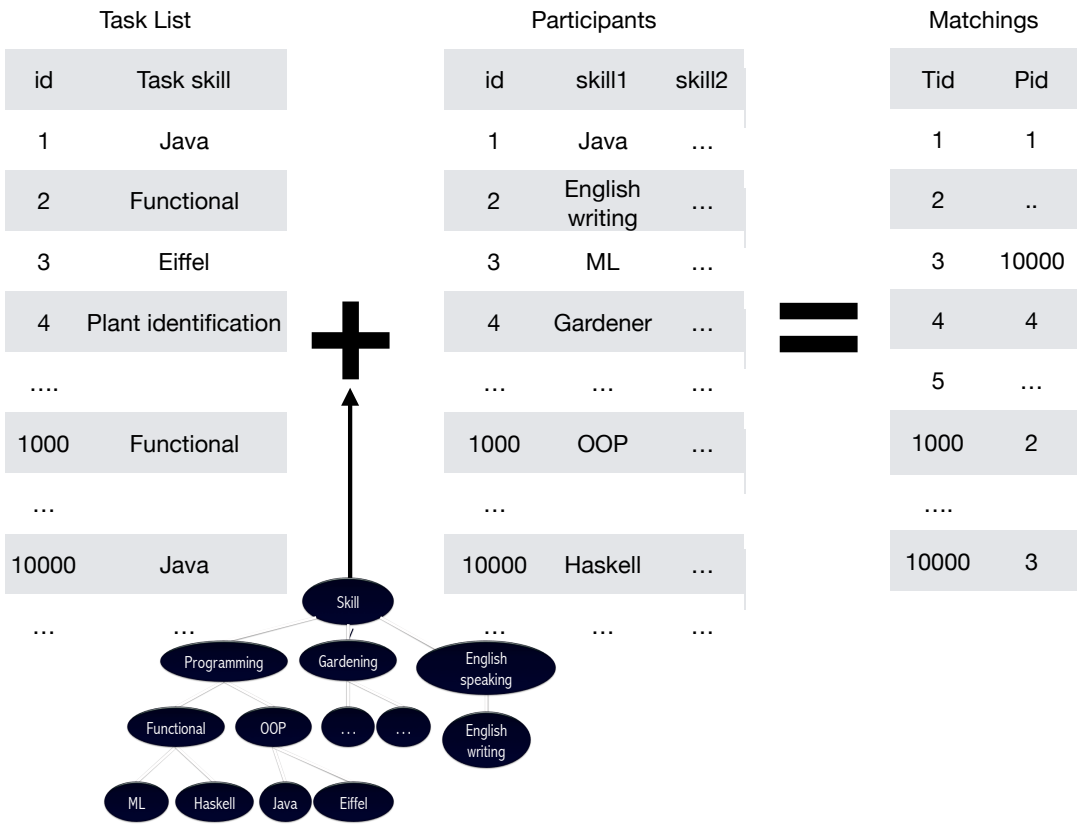



FIGURE 1.5 – Contribution 1 : Matching tasks with participants using taxonomy of skills. Illustration in Figure 1.5

Contribution 1 : We propose a skill substitution model that uses a taxonomy of skills (as the one seen in Figure 1.6) and optimization algorithms that improve one-to-one task to participant mappings for crowdsourcing. Illustration available in Figure1.5.

We want to be able to assign a "core Java task" to a "core Java programmer" (Figure 1.6). This is possible when we use keywords to match tasks with participants, it is called knowledge-intensive crowdsourcing and is already addressed in related work [18, 79]. However, if no core Java programmer is available we will need to find the next best available person. Looking at Figure 1.6 it seems that we could look for either someone with better skills, such as a "Java 1.8 thread" programmer or a "core C" programmer that has slightly less relevant skills. But if we randomly let a "keyword matching" algorithm choose, it will not be able to tell the difference between a "Java 1.8 thread" programmer, a "core C" programmer and a gardener while there was a more suitable person available. With the use of the taxonomy, we could be able to match any of the above depending on the availabilities of participants we have in our crowd. We are also able to avoid if possible the use of a "programmer" for a

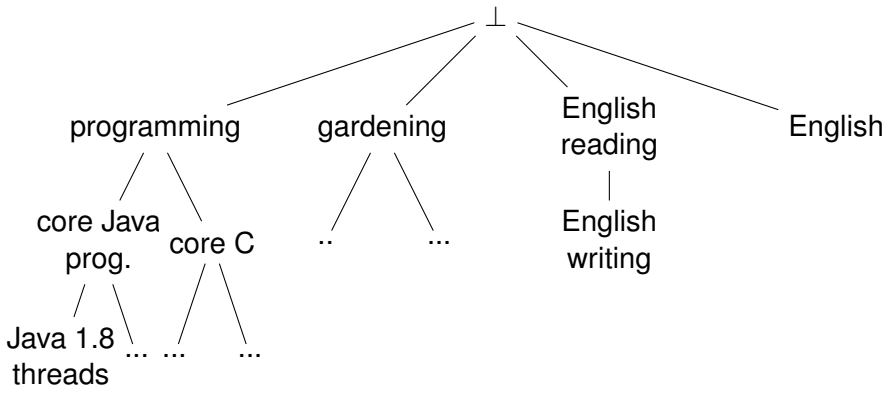



FIGURE 1.6 – A skill taxonomy

"gardening" task and vice versa.

The results of this contribution were published to [65, 66]. To summarize the main points of this contribution we can break it down to simpler ones :

— we use a taxonomy $T$ for the substitution and identification of relevant tasks to relevant participants,
— we propose a taxonomy based distance of assignment between a task and a participant that are less relevant to minimize the trade-off between the different assignments,
— we provide an extensive synthetic experimentation and real experimentation that validates our hypothesis on the importance of the skills substitutions.

After briefing our contribution for the first challenge, we present our contribution to the second challenge :

Contribution 2 : We propose a model that selects out of a huge list of available tasks, a personalized short list of tasks for the participant to choose from. An illustration can be seen in Figure 1.7.

We want to assist participants in finding relevant tasks and provide them with meaningful choice. Specifically, if someone is a specialist in "Java", "Gardening", "OOP" and "ML", we want to propose him with a meaningful subset of the available tasks that correspond to most of his skills. That strategy can save an important amount of time to the participant. Also, we want urgent tasks to be presented first, thus we propose an urgency metric depending on the task deadline and prioritize these tasks first. This strategy guarantees that participants get a list of suitable, and diverse tasks while urgent tasks do not starve of answers.

We could summarize this contribution to the following steps :

— we propose a relevance metric for a given list of tasks to a participant using our taxonomy distance metric,
— we define a diversity metric for a given list of tasks to a participant that is optimal when more skills are taken into account,

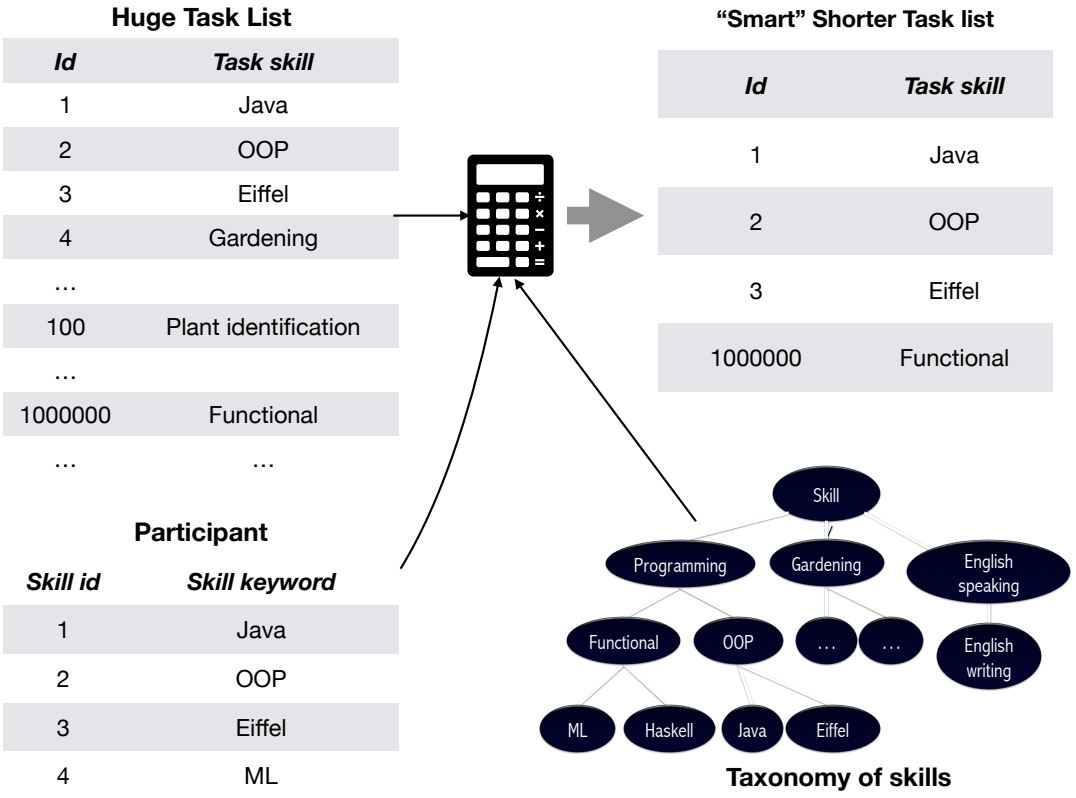



FIGURE 1.7 – Contribution 2 : We present each participant with a short list of personalized tasks out of a huge list of available tasks

— we provide with an urgency metric for a task and then extend it for a list of tasks that takes into account the contributions done for a task, the time left for the task and an average time to complete it,
— we perform an extensive synthetic and real experimentation that validates our hypothesis on the diversity and urgency of tasks compared to other methods and
— we show that participants : (1) select a clever shortlist of tasks over an extensive and exhaustive list of overwhelming tasks and (2) participants select tasks at least 2 times faster in the shortlist.

## 1.6 Structure of the Thesis

The thesis is organized as follows. In Chapter 2, we present the state of the art in crowdsourcing platforms and areas related to skill and preference modeling, participant profiling, resource allocation and complex task management. We also position them with our work and other relevant technologies and tools that could be used for crowdsourcing (such as top-k, skylines, search diversification from information retrieval etc.).

Chapter 3 presents the foundation definitions of our work on concept distance within a taxonomy. This foundation helps understand the substitution of skills on matching between tasks and participants.

Then, on Chapter 4, we focus on hierarchical skill management and optimized task assignments in crowdsourcing [66]. More precisely, we study the problem of matching #N tasks to #N participants, provided a taxonomy of skills $T$, a distance of assignment and a minimum expertise task requirement for each task.

Chapter 5 studies the effect of participant choice in modern crowdsourcing platforms. Specifically, while keeping the taxonomy structure for skill substitution we introduce a model for a ranking function that provides smart shortlists to participants. With our extensive experimentation, we show that while the quality is not compromised the participants stay motivated and save time from searching for relevant tasks. We also study how participants choose tasks in modern crowdsourcing platforms.

Finally, we conclude the thesis in Chapter 6 with a discussion over the results of our work and point out to future works.

# Chapter 2

# State of the Art

We have already motivated the challenges of modern crowdsourcing concerning the skills of participants, the quality of answers and the selection criteria among a huge list of tasks. We have also underlined the importance of skill modeling and participant preference modeling in order to provide them with the best tasks. In this chapter, we will present the state of the art on how they tackle the above challenges of profiling, modeling and mapping tasks and participants in crowdsourcing applications among with information on well-known crowdsourcing platforms. More precisely since we talk about skills we will introduce state of the art concerning :

— crowdsourcing platforms,
— skill modeling,
— user profiling,
— diversity in preference modeling,
— task assignment and
— workflow definition and management.

## 2.1 Academic, Commercial and Community Crowdsourcing Platforms

There exist many crowdsourcing platforms dedicated to a specific task. Several uses and use cases have been mentioned in [33]. Apart from the generic commercial crowdsourcing platforms that we mentioned in Section 1.3, there exist also : academic and community crowdsourcing platforms.

### 2.1.1 Academic Platforms

FoldIt [4] is a popular interface that permits protein folding through an online interface in the form of a game. It gained popularity after non-expert participants found a new protein folding not discovered by biologist experts. This resulted in several scientific publications and the increase of popularity for crowdsourcing as a method for scientific data collection, cleansing and gamification. In e-Science, crowdsourcing

is used to gather huge data sets (participative sensing, for example the "Sauvages de ma rue" [7] project).

Similar to FoldIt, there exist several other crowdsourcing systems that are built with the purpose of solving one specific scientific problem. Even though this approach is successful when deployed and the crowd has been acquired, for each targeted crowdsourcing task, a new system has to be developed and a new crowd to be attracted and sustained. Systems specifically designed for more generic crowdsourcing are on their way (sCOOP at Stanford [6], CrowdDB at Berkeley [41]). A classification of these systems was recently proposed in Communication of the ACM [34]. Other prototypes, using a declarative approach have been proposed as well [74] [30], such that given an abstract description (formal model) of the intended task, the corresponding crowdsourcing system is automatically deployed and managed. This declarative approach is typical in database systems. Such prototypes include the sCOOP project at Stanford [6] or the CrowdDB project at Berkeley [41]. But existing works mention the difficulty to merge crowdsourcing and data management. One has in particular to model participants as a computing resource with variable availability and uncertain answers [28] [30]. In the following we will focus on two more generic systems : a Datalog based declarative language [70] and an active database based system [17].

In [70] authors present a platform named Cylog/Crowd4U for complex crowdsourcing and combined human/machine computation in 2 discrete application examples. The first application is called TweetPecker and encourages users to extract data from particular tweets to predefined tables. What is also interesting is that the users can create rules for what the value of a key in the table will be if a particular keyword appears in the tweets. The second demo example encourages people to work together in order to create a list of publications for the University of Tsukuba. The people can add one or more publications while also declare that the list is complete. Coding in CyLog/Crowd4U is based on Datalog with the addition of open predicates, game definition components and an explicit rule table. The open predicates do not confront to the closed world assumption but profit from the open world assumption of human involvement in computation. The game definition components reassure that during the answers we will keep the answer that will converge to the correct result. In order to be able to add and remove dynamically rules with the crowd, there is an explicit rule table. Finally, the architecture of CyLog consists of :

— an execution controller that takes as input the CyLog program to manage the data-centric complex crowdsourcing handling,
— a logic processor that handles the data,
— a game manager that handles the people and
— an OpenFact API for rapid development with default functions and programs.

As a conclusion, they believe that CyLog can be used as a query processor for complex data-centric crowdsourcing with its extra ability to handle people (rational data sources) and rapid development support.

Reactive Crowdsourcing of [17] presents the notion of crowdsourcing that is overwhelming by defining it, stating the constraints of existing models and proposing a novel solution. Authors define crowdsourcing as the act of people taking part

in a computation. To this extent, they state that existing platforms do not provide a united, flexible and generic way to handle strategies, tasks, results and users. To do this they propose a conceptual framework that works in a reactive execution environment. The reactive environment works with rules, executes tasks and performs task decomposition and routes tasks for execution to different platforms such as Amazon Mechanical Turk, Twitter, Facebook etc. An abstract model describes the way that the tasks will be handled and different model transformations take place to express this meta-model at the level of tasks, objects and users. The main core of the platform is the Reactive Design Language based on the event, condition, action paradigm (ECA). It is a rule language inspired from the active databases and the active rule programming. The termination of the execution is guaranteed from the separation of the stages of control rules, result rules, execution rules and the acyclicity of the graph that executes these stages. The control rules assess the performers to help the result rules find the spammers. Then, the result rules close the execution when the answers are sufficient and detect the spammers. At last, the execution rules remove the spammer's answers and re-plan the tasks that have not met a majority constraint. To evaluate this platform 284 users that performed 3500 microtasks have been asked to contribute. Then they participated into three different politician applications and eight different experiments have been made. In the experiments, the different rules are assessed both on performance and accuracy.

### 2.1.2 Community Platforms

We can consider other examples of crowdsourcing usage such as data acquisition platforms (e.g. Wikipedia). In Wikipedia, participants produce an encyclopedia which is constantly updated and corrected by its own crowd.

Freebase is another example of a crowdsourced platform. Similar to Wikipedia, it is a knowledge base but massive and highly structured containing knowledge about everything. (massive and structured knowledge base about everything).

Zooniverse [1] is another example of community-based crowdsourcing platform. It is one of the biggest and most well-known platforms. Participants can identify animal species of different kinds within the platform through a gamified and interesting platform.

It is also noteworthy that crowdsourcing is used for a crisis such as hurricanes, earthquakes, floods and similar natural disasters [21]. The main uses in such case are basically to identify potential victims and minimize the human life cost of such catastrophes.

## 2.2 Skill Modeling

There are several ways of dividing the skill modeling according to the method that has been used. We identify skill modeling into three main categories :

1. https ://www.zooniverse.org/

— keyword-based methods,
— taxonomy based methods,
— human resource research methods.

### 2.2.1 Keyword-based Methods

In their article [76], the authors propose a conceptual model following a hybrid approach to profile users in order to provide search engines with better results. The motivation is the poor real-life matching efficiency of search engines. Their contribution unites 4 approaches into one hybrid approach and is applied to a document recommendation library system. Their proposal is the combination of static content profiling, static collaborative profiling, dynamic content profiling, and dynamic collaborative profiling. Static content profiling relies on information implicitly given by the user upon his registration (W3 model : Who's the user, What are his objectives and Where is he). This gives a set of $(f_i, v_i)$ tuples that represent the user where $f_i$ is the keyword and $v_i$ is the weight of the keyword (0-1). Dynamic content profiling relies on monitoring user's actions such as browsing patterns and clicking activity. It also monitors the implicit user's feedback such as user's search history among the past keywords and user's interest/disinterest to documents. This provides with the ability to change the weight values on keywords. Then the static collaborative profiling is based on finding implicit similar users from the profiles they have given during inscription. The last method is the dynamic collaborative profiling that clusters similar users with respect to their behavior and their loans and reservation patterns. In order to assess their experiments they use 2 metrics : (1) the improvement of relative precision and (2) the improvement in reducing information overload. Both metrics show significant improvement when their system is used while there is no guarantee when the combined system (with only static methods) is used.

Their contribution on a four-way profiling (static, dynamic, static collaborative, dynamic collaborative) can have real effects on crowdsourcing. As a complementary work, it could be useful to obtain a history of participant interaction with tasks to improve their profiles. However, they use a simple vector-based model (set of $(f_i, v_i)$ tuples) while we propose the use of a taxonomy of skills and a model for effective skill substitution.

### 2.2.2 Taxonomy-based Methods

In their article [67], the authors provide a different approach for recommender systems with the use of ontologies. They suggest that recommender systems provide a lot of useless results because of the nature of the research and the way people are profiled in current systems. User profiling is mostly knowledge-based and behavior based. On the other hand, ontologies are making knowledge both human understandable and machine-readable. To assess the advantage of ontology injection into recommendation systems, they create two systems : Quickstep (recommendation system for researchers within a computer science laboratory) and Foxtrot (a searchable database and

recommender system for a computer science department). In both systems, they use similarity measurements.

Their contribution is motivated within the domain of recommender systems by injecting ontology and taxonomy inference capabilities. They also have some similarity measurements (metrics). Their scope is different since they are not using a taxonomy of skills but data methods for query optimizations to perform better recommendations.

In their article [20] the authors want to create a better and more efficient search engine. They argue that current search engines do not treat well the connections between keywords and two or three keywords that are provided can not really limit the search to the needed results. For this reason, they propose a better alternative that consists of a server-side and a client-side personalization of the results. In their user case, they implement and test their model in one of the most known search engines for jobs in Ireland. For the server side model, they suppose the existence of a taxonomy for different jobs (or skills). In order to answer a certain job query, they perform a similarity-based matching with the taxonomy. They use a distance metric which is calculated as follows. If the query concept (target) is in the same or an ancestor level of the job found then the distance is zero. When the target is at a lower level than the found job then the distance is formed as the sum of the edges needed to access in order to find this target.

On the other hand, on the client side, they perform a k-nearest neighbor classification algorithm to find the best candidate jobs (good job, bad job, candidate job classification) instead of the classic majority classification strategy. To test their model they perform two experiments for 10 users in 58 jobs chosen among a database of 3800 jobs because of their cross-section employment characteristics. The first experiment was made to see the significance of the k (number of nearest neighbors) on the setting measuring the mean classification/personalization accuracy. The second experiment with a standard number k (k=13) showed how close the system performed for each user in comparison with the baseline system. At most cases, the personalized system proposed outperforms the baseline model and at some cases, they are really close to tell the difference.

Their main contribution is to improve search engine results. They apply a taxonomy based model to a popular Irish job search engine. Unlike our definition of distance, their distance is based on a graph distance. Similar to our work it also supports some basic reasoning about jobs. For instance, C++ programming and Java programming are object-oriented languages and thus people looking for object-oriented jobs can be covered from Java jobs. Their distance notion is defined by the graph distance, counting the edges in the taxonomy that connect the two different concepts. An example can be seen in Figure 2.1.

### 2.2.3 Human Resources and Psychology Motivated Research on Skills

In their article [22], the authors provide their empirical insights on competency modeling. They first provide the key differences of job analysis to competency mode-

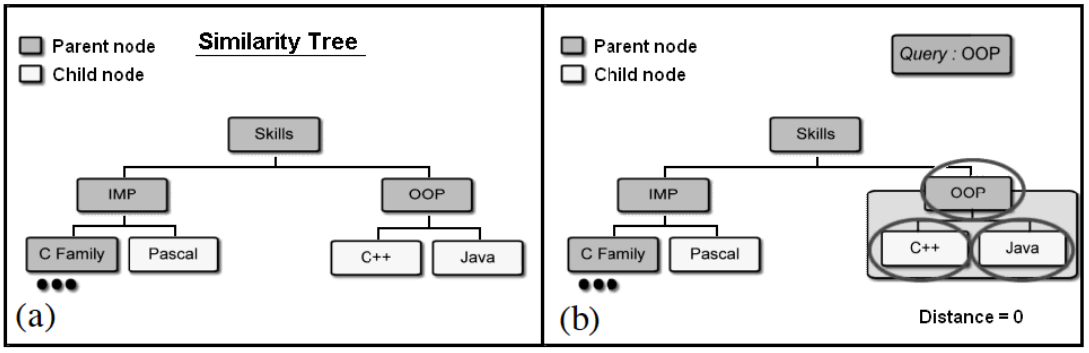



FIGURE 2.1 – Figure taken from [20] where we can notice the notion of distance in a taxonomy of jobs

ling. They define job analysis as the description of progress on competencies claiming and believe that the last complement the former. Such key differences contain :

— the distinguish of top performers to average performers
— the description of progress on competencies
— the building of competencies from a top-down fashion rather than a bottom-up fashion
— the link of competencies to objectives
— a representation that facilitates an ease of use
— a finite number of competencies that can be identified for multiple functions or job families.

They divide the competency modeling into three sections :

— analysis,
— organization and presentation and
— use.

The first consists of competencies procedures like linking competencies with goals and objectives and considering future-oriented job requirements. The second consists of procedures such as adopting or creating a language and a library of competencies. The last contains procedures such as managing usability through Information Technology methods and maintaining the currency of competencies over time. Further uses of the competency modeling for the HR include hiring, training, evaluating, developing careers, promoting, managing and compensating employees.

They mainly perform research on competency modeling and skills from the HR recruiter and management point of view. What is interesting though, is the canonical manner of their propositions (for instance libraries) and collection from know-how knowledge that could be eventually provided with computer science methods.

In their article [29] the authors provide a review of recent learner models that have been found to have the largest known role. They state that individualized learning is proved to be better than classroom learning and they assess different tutor model methods and different student model learning methods. They identify two significant

propositions in the tutor family that is the Cognitive Tutors and the Constraint-based modeling. For the student family, they focus on methods such as the item/skill matrix approach, Bayesian modeling of skills and prediction and also on affect, motivation and disengagement methods for students.

The authors provide a generic theoretical model on student evaluation. Their research is again not motivated from computer science or crowdsourcing but they use Bayesian methods and their skill matrix is an interesting way to monitor the evolution of students' skills.

## 2.3 User Profiling in Crowdsourcing Applications

We have identified user profiling suitable for crowdsourcing mainly into two main categories : the keyword-based and ontology-based profiling.

### 2.3.1 Keyword-based Profiling

In [18] the authors present a way to extract experts in different domains from social network data. Figure 2.2 depicts a visual of the expert extraction system. The method they propose consists of a meta-model to evaluate resources shared in a social network. The resources in this model are assessed with a direct and an indirect way. The direct method considers the direct profile info of the users, while the indirect method relates the resources with a depth distance of two from each corresponding user. They also comprised the bidirectionality of friendship to assess the quality of the relation between people. The resources were assessed in 3 different steps ; the text processing, the entity recognition, and the disambiguation steps. The algorithm they choose to find the best corresponding experts is the top-k search (vector space model). For each resource, they calculate a relevance with different weighted average formulas. Then in order to assess their model, they use three social networks ; LinkedIn, Facebook and Twitter. In their evaluation method, they use a dataset of 30 expertise needs with 330 000 information resources of which the 230 000 where English text and also 40 expert candidates for the ground truth. Afterwards, they provide different metrics to assess the results such as the Mean Average Precision, the 11-Point Interpolated, the Average Precision, the Mean Reciprocal Rank and the normalized discounted cumulative gain. The parameter reference was also studied considering the contribution of the a-value of the references versus the text analysis. The expert candidate references were included also. According to their results, Twitter performed better in the 30 domains of expertise they chose while Facebook was better for TV, movies, and hobby related material. Linked-in had quite good zero distance resources for computer science related material. However, it was outperformed by Twitter on most other domains. Concerning the related work, they mention that the expert finding problem is not a new problem and people have worked previously on it. Relevant problems include the jury selection problem and the expert team foundation problem. Their main contribution is applying resource based methods on social networks using behavior

in order to trace the experts for a particular need. Our work is complementary to their contribution. We assume that the skills have been assessed already and we use a finer model of skills, instead of flat tags or keywords with a probability, to assign tasks to participants for any use that the taxonomy could provide.

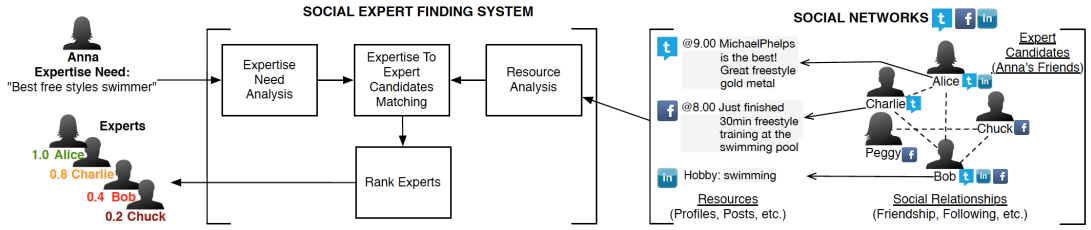



FIGURE 2.2 – Expert extraction approach from [18]

### 2.3.2 Ontology-based Profiling

In [62] present their vision for an ontology-based system with quality management (Figure 2.3). They identify the problem of current crowdsourcing platforms into the uncertainty and the lack of well-defined quality assurance for the answers given by the participants. Given a correct crowd, we can profit from its wisdom if we assign the task to the correct people. They mention the related work that has been done in the field of assessing people for specific tasks such as the question and known answers method and the redundancy and repeated labeling method of Dawid and Skene [27]. Instead of error rate matching techniques, they believe that finding the potential match is more efficient. Thus, they propose a skill ontology and competencies model that given an annotated task question only the corresponding experts of the domain. The quality of the taxonomy can be judged in 4 different levels. The result's quality, the platform's quality, the task's quality and the worker's quality. Respecting this taxonomy they propose a skill ontology-based model consisting of the skill ontology, the ontology merger the skill library of assessments the skill aligner the reputation system and the task assigner. The workflow can be viewed from 3 sides : (1) the requester's, (2) the platform's and (3) the worker's side. In this vision paper, authors believe that uncertainty is inevitable when humans are involved in computation and that is why quality assurance is mandatory especially when there are more chances to find unqualified workers.

## 2.4 Diversity in Queries for Preference Modeling

In this section, we will present the different methods for preference modeling that take into account diversity. On diversity queries for preference modeling, we have identified some common ground, firstly, in top-k and skyline queries and secondly, in search engine queries and crowdsourcing diversification.

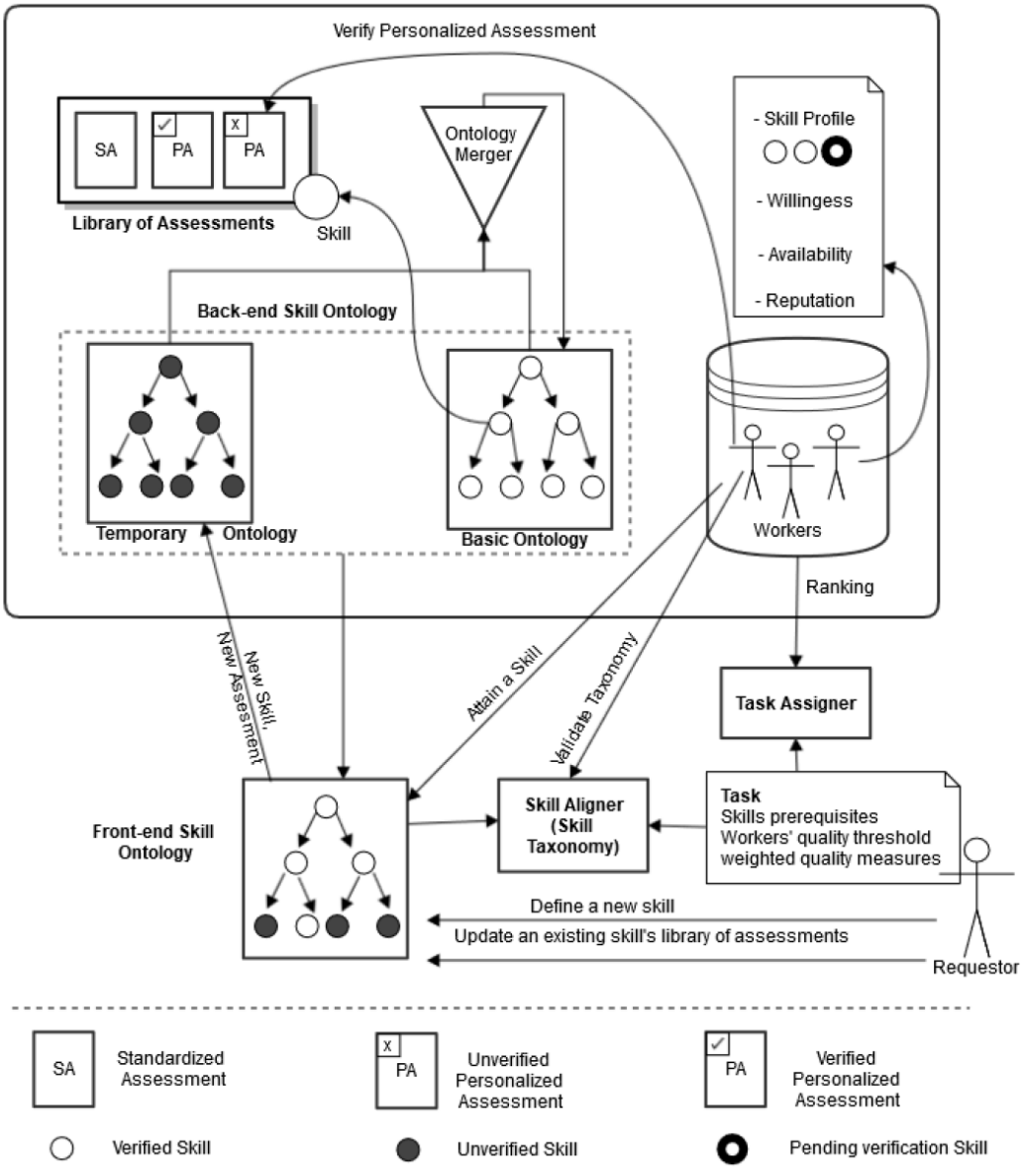



FIGURE 2.3 – Workflow of a vision system [62] based on skill-ontology

### 2.4.1 Diversity in Top-k and Skyline Queries

Diversity has been studied in the context top-k and skyline queries. The following work on top-k queries examines different ways to provide diverse and relevant results.

#### Top-k Queries

The following work [35], [43, 25],[50], [63] is motivated within the domain of databases and more precisely the top-k queries. [35] also proposes a top-k diverse query optimization. The difference in their model is the ability to model top-k queries on continuous data where the data are updated with deletions and insertions. To optimize the diversity of the produced top-k sets they use a data structure called cover trees. Their model can work with streaming data as well.

#### Skyline Queries

In their paper [85], the authors present a framework that permits diversification of skyline queries. They identify the problem of most large data skyline queries as difficult for users to determine manually the most interesting and different results within the queries. For this reason, they propose a natural way to diversify the query results and present with the more diverse sets. The metric they use is the Jaccard similarity for its intrinsic properties. An interesting difference with our approach is their independence of an index. In their case, the use of an index can deteriorate the performance of the query while in our case the index contributes to the accuracy in terms of relevance of results while not neglecting the diversity optimization of the proposed lists. Instead, they provide a signature generation method for their skylines using randomized hashing functions.

[59] also defines subspace skylines as an optimization problem to increase the diversity of results. They examine two different methods for grouping the similar results : skycube and skyline groups. Their contribution focuses on multidimensional large data sets and how to unify these two methods. Their work is motivated within generic skyline queries and optimization while our work is motivated within real-world crowdsourcing optimization on crowdsourcing participant profiles and tasks.

### 2.4.2 Diversity and Relevance in Search Engines and Crowdsourcing

The problem of diverse and relevant queries is a difficult problem (proved to be NP-hard at the worst case [24, 91, 10]). Apart from the database community that studied the problem, from the top-k and skyline query perspective, it has also been studied from the information retrieval community with the goal to diversify search engine results and avoid incomplete queries. Also, the crowdsourcing community has started studying the problem to improve task and opinion diversification [91].

#### Search Engine Query Diversification

In their paper [10] they explore the problem of queries for documents that belong to multiple categories. Similarly to our work, they assume a taxonomy (not necessarily hierarchical) that categorizes documents in various domains. Since a document might belong in different categories, they examine ways to improve the diversity of the results given a certain query. While their work is only taking care of the relevance

and diversity of documents and is motivated within information retrieval there are similarities to our work. Instead of participants, they have queries and instead of tasks documents. However, there is no urgency taken into account, since documents are not tasks that need to be performed and thus there is no motivation to present documents upon an urgency metric.

Similarly, in [80] the authors model reformulations for queries to diversify web search results. In their work, the initial motivation for reformulating a query is the lack of information on the user's needs. Their work is also motivated within information retrieval and they provide an effective model that presents documents to users in a diverse way. They use the TREC 2009 dataset to evaluate their framework which gives strong motivation for diversification of search results. Similarly to our work, they identify that the lack of precise information and preferences for participants and the diversity of existing documents explains the need for diversity into the results into users. However, the work is not directly applicable to crowdsourcing needs and there is no notion of urgency or completion for tasks. Also, they do not take into account the context which is the profile of the user that makes the query. In our case, the user is the participant of the crowdsourcing platform.

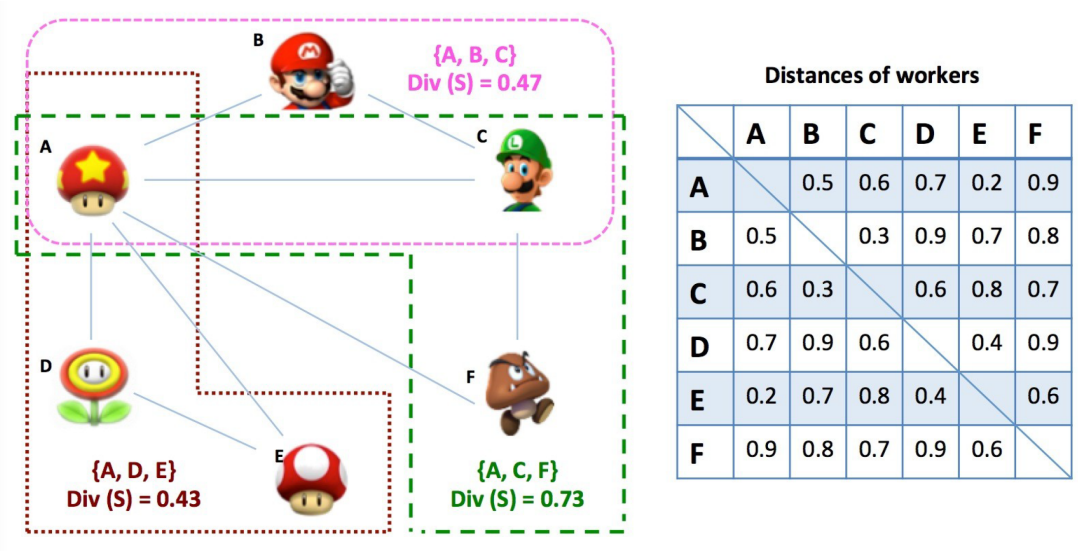
In their work [24], state the importance of getting nuggets of information to the search queries of internet users is mentioned. They focus on the importance of having a reduced redundancy and increased diversity to have a more efficient list of results for the user queries. In this context, they study the different distributions of results and how these can be diversified. Their work is motivated within the domain of information retrieval.

In their paper [89], they also have a diversity setting for ambiguous search results. They provide different metrics and evaluate with the TREC Web Track 2009-2013 dataset to both create their intent hierarchies and evaluate the results. The intent hierarchies are very similar to our use of taxonomies. In their case, the hierarchies are built from the dataset and bring a bias and discrimination which filters out inappropriate results. In our case the hierarchies are made from taxonomies of skills and our metrics are based upon the taxonomy (Section 3.4, Section 4.3 and Section 5.1.2). In our work though, since we do not want to have urgent tasks starving and our context is resource allocation, we need to take into account the urgency of tasks as well though.

**Crowdsourcing Related Diversification**

In their paper [91] oppose to the expert based view for crowdsourcing and they formalize two different methods focusing on the diversity of crowdsourcing opinions. They propose two methods ; (1) the similarity-driven model (S-model) and ; (2) the Task-driven model(T-model). They oppose to the existing expert-driven task assignment by supporting the wise crowd idea. In order to model the smart crowd one has to reassure the diversity, the independence, the decentralization and the aggregation of the opinions. They believe that diversity under proper circumstances can triumph over ability and that it is loosely defined in current papers. As in our case, the optimal solution for their problem is also proved to be NP-hard. That gives ground for proper

approximation algorithms to be used with significant performance advantages over the optimal solution. Their solution makes serious contributions towards the diversity one can obtain in crowdsourcing, however, their model does not take into account the different expertise of the domains and skills of participants. It also does not take into account an optimization of the budget or the timely completion of the tasks. An example of maximizing diversity with their method is shown in Figure 2.4.




| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | | 0.5 | 0.6 | 0.7 | 0.2 | 0.9 |
| B | 0.5 | | 0.3 | 0.9 | 0.7 | 0.8 |
| C | 0.6 | 0.3 | | 0.6 | 0.8 | 0.7 |
| D | 0.7 | 0.9 | 0.6 | | 0.4 | 0.9 |
| E | 0.2 | 0.7 | 0.8 | 0.4 | | 0.6 |
| F | 0.9 | 0.8 | 0.7 | 0.9 | 0.6 | |

FIGURE 2.4 – Finding three participants that maximize diversity [91]

Several works focus on push methods that assign tasks to participants, however, [12] take into account participant choice and preferences. In their work, the main focus of the task proposition for participants is a summary of tasks that can fit participant preferences in terms of skills, preferences and accepted wage to work on. The main goal of the approach is the maximization of the throughput of work by adopting a more worker-centric approach as we also propose in our approach. Unlike our proposition, diversity of tasks is not taken into account and there is neither skill substitution nor a skill taxonomy to model the tasks participants choose. Also, instead of a dynamic urgency, they avoid the starvation of older tasks by making a mix of older and newer tasks in the summary of tasks, they propose.

[95] aims at solving the problem of the lack of choice participants have when being assigned tasks. Authors propose to compute the optimal task assignment taking into account both task requirements for expertise as well as participant preferences based on historical data. While they propose methods to compute a task allocation, our work looks at giving participants a choice from a pre-computed set of tasks which our algorithms estimate being a good fit for them. Thus, the advantage of our approach is the ability to take into account irrational decisions that participants may still make when picking tasks to work on without forcing them to work based on algorithmic decisions. We also use a method to substitute skills with the use of taxonomy while

they rely mainly on keywords.

Another approach on task proposition and assignment from [75] proposes the combination of relevance, diversity and payment preferences for participants. In their approach, they propose three different ways of task recommendation to participants based on (1) relevance of participant skills and task requirements ; (2) diversity of proposed task skills and (3) an on the fly method that gives an optimized compromise for diversity and payment. Similarly to our approach, they propose tasks to participants acknowledging the importance of both *relevance* and *diversity* of types of tasks. However, there is no skill taxonomy used and they focus on payment preferences instead of the urgency of tasks. While they use a fixed top-3 of tasks, we position on the size of the list and propose different size of lists for our synthetic and real experiments. Their method is complementary to ours and strengthens our results and importance of *diversity*.

## 2.5 Task Assignment in Crowdsourcing Applications

Several recent works address the question of assigning tasks to participants in crowdsourcing platforms. Despite the similarity of the overall goal, the settings and assumptions of these works are rather different.

### 2.5.1 Keyword-based Assignment

In their work [39] they provide a complete framework on Crowdsourcing taking advantage of the Amazon Mechanical Turk crowdsourcing platform. Their work consists of a number of different modules and it is based on text and keyword matching and similarity methods. The different modules mentioned are the Warm-up, the Worker Checker and the Adaptive Assigner. In order to classify a task, they use a text mining method (such as TF-IDF) and page ranking similarity methods to extract the stop words and keep the significant words. Thus they limit their current work on small-text tasks. Then a graph is built with these words to build the similarities between several tasks. Workers are ranked with their previous performed tasks and they get for every non-assigned task a certain ranking based on their previous performed tasks. Then to maximize the number of good assignments they use a heuristic and they prove that the optimal is NP-hard (maximum coverage equivalence). To evaluate they use several domains and they see the results on these domains. There is some similar notions but applied and executed with completely different ways. The speed of their algorithm is fast but our complexity of dmax*|T| is essentially lower. In addition, our work is not limited to small-text tasks only. A proper non-automatic annotation though is needed. The evaluation is done on mostly non-knowledge intensive tasks of multiple choice questions.

### 2.5.2 Skill-based Assignment

In the version of the assignment problem studied by Acemoglu et al. [9], the participant skills are described by a hierarchical structure, however (unlike in our setting) the difficulties of the tasks are not known in advance. They also provide a pricing mechanism that results in an optimized result quality after multiple matching rounds. However, their work is motivated and studied within the domain of $Finance$ and not $Computer\ Science$.

In their article [79] formalize the problem of task assignment given a certain pool of workers and propose different algorithms to solve it. To begin with, they argue about the need of knowledge-Intensive crowdsourcing that has already been identified in related work and is the first to have formally treated it. They define the notion of knowledge-intensive crowdsourcing as the process of collaborative creation of knowledge content. At the same time, they oppose it to the micro-tasking classical crowdsourcing and define their optimization problem. In order to define their problem they consider a set or workers $U$ and a set of tasks $T$ that need to be assigned to the workers and they also assume certain constraints such as (1) a preemption of workers (2) minimum number of tasks per user is one and maximum task per user is two. They call their contribution C-DEX (from Crowd index) and they prove that it is at least NP-hard by reducing the multiple knapsack problem to their problem. For the different values of the objective function ($Qtj$) they use they prove their theorem for its modularity and monotony. Then, they provide three different algorithms to solve the problem. That is the C-DEX optimal, the C-DEX approximative (deterministic and non-deterministic) and the C-DEX+. The first uses the reduction form the multiple knapsack problem and thus is using the dual-primal problem properties from integer linear programming. The second is a greedy approximation based on the slightly overflowed knapsack problem with either a deterministic or a stochastic criterion. The third of their algorithms reduces the variables of the problem by grouping the users that have same or equivalent profiles to one virtual worker group represented by the highest amount of money among them and the minimum knowledge among them. The algorithm then solves the reduced variable problem with the same integer linear programming method using the same tasks as in the C-DEX optimal but also the virtual workers instead of all the workers. All algorithms have an offline and an online phase. In the offline phase, the indexes are being created for all users no matter their connectivity status. On the online status, only the online users are taken into account in the reduced problem. For the evaluation, they used both real and synthetic data. For the real data, they used 250 workers (workflow is shown in Figure 2.5) from AMT to demonstrate the improved task quality and the feasibility of the approach. For the synthetic data, they used an event-based crowd simulator to check the scalability and the task quality. The best quality is given to the C-DEX optimal, but the performance is by far the worst on the C-DEX optimal. The C-DEX+ has the best ratio of performance and quality on the results.

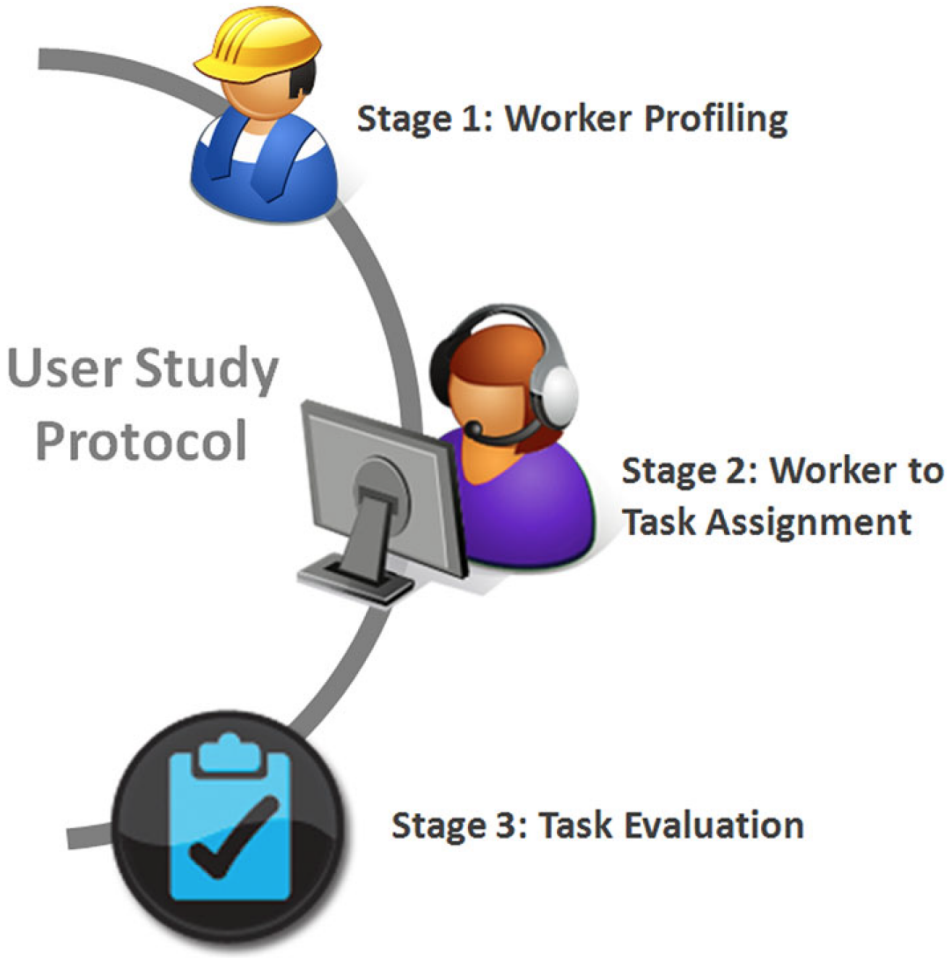



FIGURE 2.5 – User study workflow followed in [79]

### 2.5.3 Quality and Budget-bound Assignment

Cao et al. [23] and Zheng et al. [96] assign each task to multiple participants (in a jury form) and they analyze the trade-off between budget and the expected (aggregated) result that one can obtain this way.

For example, [88] studies the problem of task assignment in a streaming setting. On the other hand, crowd data analytic platforms of Fan et al. [39] and Liu et al. [61], assume a bounded budget for the assignments, that we do not have. Mo et al. [69] also assume a limited budget, and propose methods to optimize the overall quality of (aggregated) responses by varying the number of participants who perform the task.

Karger et al. [53] study the task allocation problem, similar to ours. However, they make completely different assumptions in their work, in particular, they do not use at all skill profiles in their algorithms.

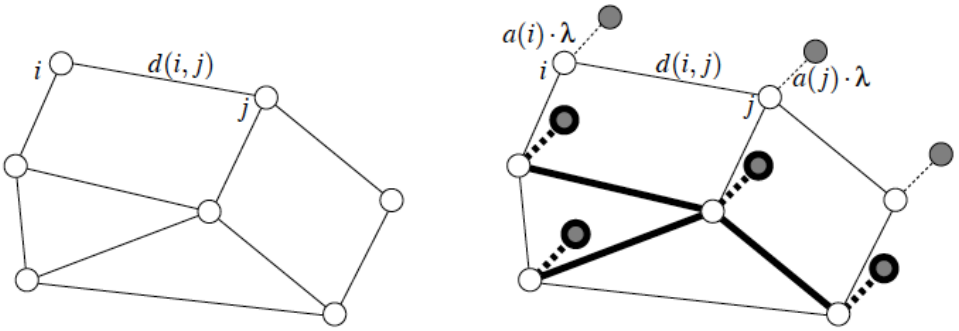
### 2.5.4 Cold-start-aware Assignment

In their article [31] propose a complete system for cold-start crowdsourcing. Without any knowledge of the profile of the participant, their system can propose several crowdsourcing tasks to the participant. The system is composed of several modules that interact with each other and have different responsibilities. Their system takes into account the participants' Facebook profile and having their likes and preferences proposes one by one the next tasks they should perform. Their platform, compared to our approach ranks participants for tasks, while we propose tasks to participants ranking the tasks and letting the participants select. The work could be complementary since we do not study or address the cold-start problem. However, their approach is not using a taxonomy of skills to manage substitution of tasks for participants and there is no explicit notion of skills.

### 2.5.5 Online Team Formation Assignment

In [13] authors study the problem of online team formation. The setting consists of teams of participants and tasks that arrive in an online fashion. The teams of participants are described with different skills. Their problem is to form a team that will perform a task while minimizing the overhead and keeping the load fair among the members of the team. They assume that a task needs several skills, that are modeled as a tag of keywords (one can only possess or not possess a skill), to be performed and that each participant can have several skills as well. The problem is NP-hard since in order to be solved optimally it entails the set-cover and the Steiner-tree problems. However, they provide a random approximation that proves to be suitable for the case of an online solution to the problem that covers the required guarantees. An example of their adapted method is shown in Figure 2.6. Unlike our work, their proposition relies on a flat list of skills and their tasks are not microtasks but complex tasks that need more than one skills and are assigned to teams. Their contribution can be complementary to ours in terms of team formation with taxonomy-substitution-aware team assignments. Since we propose assignment in one round and for all participants we also keep the work balance at a fair level. However, we also consider the fact of choice in case of a list proposition of tasks since this is more common into modern commercial crowdsourcing platforms.

In [77], the authors study the problem of team building in collaborative settings. They consider the optimization problem of building the best team of participants to reach a given task quality. The related skill model is vector-based but enriched with a probabilistic distribution. Their problem is similarly at least NP-hard since it also entails the set-cover problem. In our work, the quality of an assignment is characterized by the overall compatibility of the skills of the participants and the tasks, which is more realistic in platforms such as AMT and Crowdflower. We do not study team assignment but our method could also work on top of this work to support team assignment with skill substitution capabilities.




(a) Original network (b) Modified problem instance

FIGURE 2.6 – Example taken from [13] where they transform the initial problem to the one shown in the figure. We can notice the original network on which they want to solve the Social Task Assignment problem and the modified problem instance on which they solve the group Steiner tree problem. With bold an example solution $(Q, T)$

## 2.6 Workflow Definition and Management in Crowdsourcing

To the best of our knowledge current work on crowdsourcing management systems and workflows can be divided into three different but interdependent categories. These categories are : (1) Microtask Crowdsourcing Systems, (2) Participative Workflow Systems and (3) Macrotask Systems. Since they can support task assignment or decomposition of tasks we believe they are important tools for crowdsourced answers and we find important to study them. A brief comparison is presented on Table 2.1.

| Platform | declarative | workflow | skill taxonomy |
|---|---|---|---|
| AutoMan [14] | - | + | - |
| Jabberwocky [11] | - | + | - |
| CrowdForge [55] | - | + | - |
| Turkomatic [57] | - | + | - |
| Turkit [60] | - | + | - |
| CrowdLang [68] | - | + | - |
| Cylog [70] | + | - | - |
| CrowdDB [42] | + | - | - |
| DeCO [73] | + | - | - |
| Argonaut [48] | - | - | - |

TABLE 2.1 – Comparison of state of the art workflow systems

### 2.6.1 Microtask Crowdsourcing Systems

Mostly known for data completion and data cleansing, there exists academic work that permits participants to provide data or disambiguate existing data. Such platforms are based on either an extension of relational databases such as SQL [42, 73] or an extension of Datalog [70] or an adaptation from Active Databases [17]. They can all support simple spammer detection and answer aggregation methods such as majority voting etc.

To begin with, Deco [73] and CrowdDB [42] propose an extension of SQL to facilitate the participation of the crowd into the computation. Built to bring the open world assumption they fill in the gaps of Database Queries for unknown (data acquisition) and uncertain (data cleansing) table values. On top of the model, a simple spamming retrieval mechanism can be built based on thresholds of previous answers and on data constraints. Despite their declarative approach, they both do not explicit ways for participant collaboration, interdependent task management and participant profile acquisition or specialized crowd focus. At the same time, there are no budget optimization techniques.

On the other hand, Bozzon et al. [17] propose a different way to query the crowd. The main core of their proposition is the Reactive Design Language inspired from the active databases and the active rule programming. It is possible with their method to also detect spammers and terminate tasks that have met their majority constraints. However, it is not clear how one can trigger collaboration and the creation of complex and interdependent tasks. Again budget optimization is not discussed any further.

From another perspective, Morishima et al. [70] present Cylog/Crowd4U an extension of Datalog to include crowd into computations. As with the SQL extension, Cylog has open world predicates. It also has game definition components for the converged answers and an explicit rule table to dynamically add or remove rules. Despite its architecture, Cylog/Crowd4U misses explicit profiles for targeted crowdsourcing and ways to involve collaboration to distributed human computation. Optimization for a budget is not taken into account but possible to program. Workflow creation is also neither directly supported nor easy for the non-programming expert.

### 2.6.2 Participative Workflow Systems

To the best of our knowledge, the several participative workflows enabling systems that exist fall into two different categories and are both based on Map-Reduce (or Man-Reduce as stated in [11]). These are the (1) Man-Reduce Interface Systems ; and the (2) Man-Reduce Programming Systems.

#### Man-Reduce Interface Systems

Division of complex work to form workflows is an intrinsic way to deal with complex tasks. This idea is shared by most of the workflow management systems. However, each one provides different ways of creating and managing the divided work.

Kittur et al. [55] present *Crowdforge* a Map-Reduce framework for complex flows of crowdsourcing. This proposition comes along with a simple to use interface for requesters to create tasks that rely on subtasks. However one can not give access to trusted workers in order to edit the workflow or connect current work with an existing database. Quality control also is an issue as the tasks are addressed directly to the whole crowd community and focalized crowdsourcing is not an option.

Kulkarni et al. [57] in their article propose a platform named *Turkomatic* that also enables the creation of workflows. In the same fashion as CrowdForge, Turkomatic is based on Map Reduce with the difference of its price and divide algorithm. The algorithm invites participants to decompose complex tasks so that is both plausible and acceptable by the workers. However, only the requesters can edit or supervise the workflows. Despite its power and simplicity for users, it lacks particular control and data-driven goals. In addition, people can not participate in editing or monitor the workflow. Finally, it can run over-budget if there is no proper supervision.

**Man-Reduce Programming Systems**

On the other hand, there exist more generic methods that deal with complex crowdsourcing. Such propositions come with either an architecture or a programming language that supports human intelligence commands.

Little et al. [60] present *Turkit* a toolkit for prototyping and exploring algorithmic human computation. Turkit adopts a straight-forward imperative programming style that surpasses the independent tasks that are usually posted on AMT (Amazon Mechanical Turk). Their main contribution is based on crash and re-run principle which introduces local and remote computation principles to reduce redundant crowdsourcing work with the use of a database storing all the human calculation steps. The crash and re-run principle is both an advantage and a disadvantage. The need for a re-run might be needed in case we are not satisfied with the acquired results. Also, it is not obvious how to make people collaborate and form teams or how we can do focalized crowdsourcing based on participants' skills.

In their paper [11] present *Jabberwocky*, a software framework (Figure 2.7) based on the idea of Man-Reduce to handle complex tasks. More precisely their Man-Reduce framework handles the dispatch of tasks to humans and then uses aggregation methods to present the result. Their proposition is a language with four main keywords that facilitate the selection of a crowd and aggregation of the given answers. However, the execution of a Jabberwocky workflow once launched cannot be altered. In addition, crowd optimizations (for a budget, domain-specific crowd, etc) need careful procedural fashioned programming instead of the desired declarative and data-driven approach.

On the other hand, Minder et al. [68] present *CrowdLang*, a framework for hybrid human/machine computation. CrowdLang is a complete computational framework with a library to facilitate new human computation systems, an execution engine and an integrator to incorporate different platforms (such as AMT, Crowdflower etc.). It disposes of basic operators for task decomposition and aggregation has basic blocks

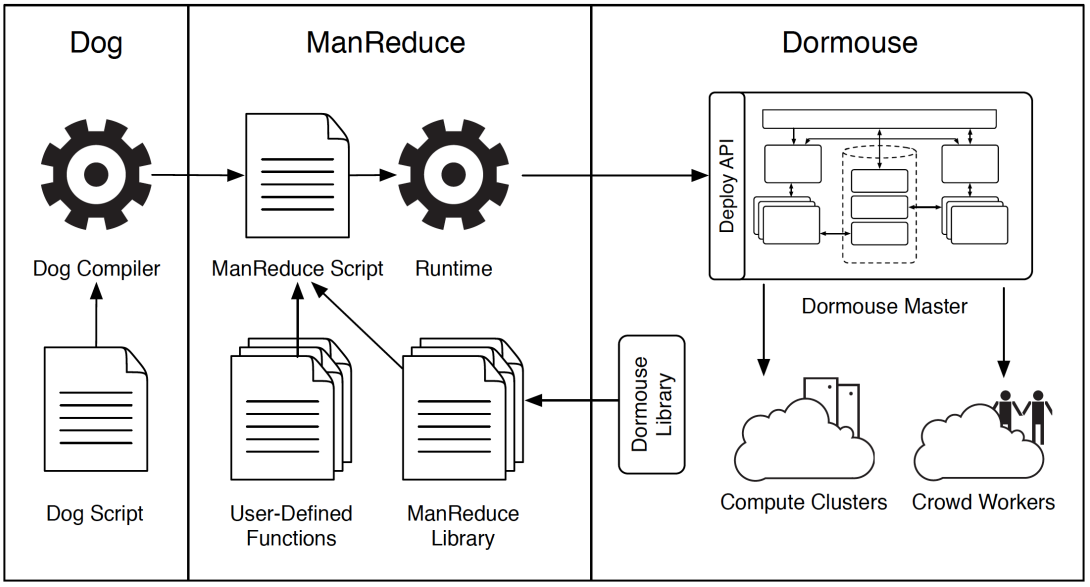



FIGURE 2.7 – Overview of the Jabberwocky system [11]

to create interaction patterns (for a contest and classic collection or collaboration and parallelized subproblem solving) and patterns for decision interaction. However, even though complete it follows a procedural way to proceed to solutions. Optimizations have to be applied by hand and viewed for each problem separately. In addition, problem design is not easy with the different aggregator methods and people can only follow the instructions of the formed workflow.

*Automan* [14] by Barowy et al. is a novel programming language that can perform hybrid human/machine computation. It implements the functional integration of human computation, automatic scheduling, budgeting and automatic quality control into a programming language. In the same track with Turkit, Automan also supports memorization of results that have been previously calculated to avoid redundant recalculations on human performed tasks. It is based on empirical and statistical observations that handle the answer restriction, the automatic budgeting, re-posting and rescheduling of tasks in case there is a lack of answers. However, there is no top-down cost estimation for the desired quality. Even though it helps seamless integration to programs it lacks integration with databases. Also one needs programming skills to contribute to a participative workflow. Participants can not really collaborate (work in teams) as one of their assumptions of the system to work is that workers should be independent in order to be unbiased. Finally, it works only for tasks with bounded answers (multiple choice, finite space of answers) which is not always the case in crowdsourcing (e.x. machine translation, NLP, etc).

### 2.6.3 Macrotask Crowdsourcing Systems

Haas et al. [48] propose *Argonaut*, a system that treats complex tasks as macrotasks instead of multiple interdependent microtasks. Instead of task division Hass et al. propose the assignment of tasks to top-tier workers. After an iterative process of review-and-correct, the system will have a better confidence to return the produced results. It also relies on a hierarchical relationship among participants (workers, reviewers, etc.) that imposes a hierarchical role to each participant. The system is based on a fixed workflow that describes this iterative process of task validation. Despite its novelty, the model, for now, does not handle any explicit notion of peers working together and combining different skills in order to complete a task. Another possible disadvantage is the inequality imposed by the hierarchical view of the participants. However, the latter can lighten up if participants can participate in more than one roles on different macrotasks.

## 2.7 Chapter Conclusion

In this chapter, we have studied methods that are needed to permit modern crowdsourcing in a large scale and to combat some of the main challenges concerning the skill management and the preferences modeling of crowdsourcing. We examined thoroughly the existing related work and we positioned it with respect to our work.

Despite the efforts of the related work, there is still challenges to improve skill modeling and task selection for crowdsourcing. Currently, as we mentioned in "Challenge 1" of Section 1.4, existing crowdsourcing works do not provide a sufficient skill model apart from weighted keywords. Needless to say that skill substitution with the use of a taxonomy is only seen to works such as a job search engine [20] (Section 2.5.5).

In Chapter 3, we will define a skill substitution model suitable for knowledge-intensive crowdsourcing that tries to fill in this gap. To improve task selection, in Chapter 4, we take advantage of our newly defined skill substitution model and present a task assignment optimization framework where participants are matched with a task. The results prove that the use of a taxonomy can significantly improve the quality of the acquired answers.

Moreover, as we state in "Challenge 2" (Section 1.4), tasks are presented to participants in huge lists that make it very difficult to select from. Also, when presented shorter lists, participants are limited to a few tasks that are not sufficiently personalized to match the task deadlines or their diversity of skills (Section 2.4.2). In addition, there is no skill substitution model to cover the lack of certain skills. To improve the task selection process in crowdsourcing, we propose in Chapter 5 a new model that takes into account the different skills of the participants or substitutes of their skills when needed (as in the model of Chapter 3 and Chapter 4). The model also takes into account task deadlines to avoid task starvation. Our experimentation shows that we can keep a high quality of answers and reduce task starvation while we assist participants in selecting tasks.

# Chapter 3

# A Skill Substitution Model

Crowdsourcing platforms such as Crowdflower [1], Amazon MTurk [2] or FouleFactory [3] engage more than 50k participants [82] who perform simple microtasks on a daily basis. More knowledge-intensive tasks can be found on specialized platforms, such as Zooniverse [4], BumbleBeeWatch [5] or SPIPOLL [6]. In this later, benevolent participants upload and annotate insect images according to a precise taxonomy of species.

We have already mentioned the importance of being able to substitute the skills of participants within crowdsourcing management. In order to do this, we believe that a use of a structure is mandatory to be able to reason and substitute the skills of the participants. To improve the probability of having a better management of the participants we need to be able to quantify a similarity among the skills. To do this we will introduce gradually a taxonomy of skills equipped with a distance metric. For the sake of this definitions, we will start by stating our assumptions on defining single skill knowledge-intensive microtasks. Then we will explain in more detail about our contribution in taxonomy aware knowledge-intensive crowdsourcing and define the different distance metrics and examples of uses of distances for participants and tasks.

While one can obtain useful results through these platforms for a number of tasks (that would be otherwise difficult for computers), controlling the quality of the results is a challenging issue, due to the unreliability, volatility or lack of skills of participants. In particular, in knowledge-intensive crowdsourcing [79], where a specific expertise is needed to complete a task, the platform should ideally assign or propose the task to a participant who has this specific skill, or at least some experience in the domain.

Generic crowdsourcing platforms already provide basic skill labeling (such as qualifications in Amazon MTurk [7] : these are short descriptions of qualifications for certain skills a participant might have or the requester might require). Similarly, academic re-

1. http://www.crowdflower.com
2. https://www.mturk.com
3. http://www.foulefactory.com
4. www.zooniverse.org
5. http://www.bumblebeewatch.org
6. Photographic monitoring of pollinator insects, http://www.spipoll.org/
7. http://docs.aws.amazon.com/AWSMechTurk/latest/AWSMechanicalTurkRequester/

search [17, 79, 92, 83] is also considering skill models to improve result quality. These existing approaches rely on flat, unstructured skill models such as tags or keywords.

However, applications often require at least some basic forms of reasoning about the skills (such as, for example, knowing that the skill *English writing* is "more specific" than *English reading*, in the sense that anyone who can write English can also read). Even such simple reasoning operations are not easy to realize with the abovementioned flat skill models. Many platforms could benefit from such a structured skill approach. On the one hand, it would allow a precise and better targeting of tasks. On the other hand, skill reasoning capacities, especially skill substitutions, would enable the participation of the full available workforce of the platform, even if skills do not correspond exactly to requirements. It is noteworthy that rich skill taxonomies are available and used in other contexts, such as ESCO [8], which is used to help European citizens in their job search and represents 5,000 skills in a structured way. Skill taxonomies are also recommended for companies using collaborative solutions so that "employees have a common language to self-profile themselves" (the Wandinc company sells such taxonomies with more than 1,400 personal or business skills [9]).

In this chapter, we propose to finely model tasks and participants using a skill taxonomy. Our contributions are the following :

— we propose an effective taxonomy-based skill model for crowdsourcing, allowing to reason about skill substitutions ;
— we define and discuss difference distance measures between the skills of a participant and the skill required by a task that reflects how well they correspond to each other ;

The rest of the chapter is organized as follows. In Section 3.2 we state our assumptions for the skill modeling of knowledge-intensive crowdsourcing. In Section 3.3 we define knowledge-intensive crowdsourcing. Then in Section 3.4 we define concept distances suitable for taxonomy-aware knowledge-intensive crowdsourcing. Finally we conclude in Section 3.6.

## 3.1 Taxonomies of skills vs. Keywords

Skill management is one of the key issues for crowdsourcing platforms. Being able to gather a huge number of diverse and complementary skills constitutes one of the strengths of these platforms, besides their ability to deliver tasks on a regular basis.

It is noteworthy that most of the existing platforms and most of the related work in the literature rely on skills described as keywords from a free vocabulary, proposed by task requesters or by participants (see for example [39, 77, 79]). To reason about skills, keyword similarity metrics can be used, such as in [39]. First, this approach has the major advantage of being extremely flexible : if a new skill appears (say, an

Concepts_QualificationsArticle.html

8. ESCO : European Skills, Competences Qualifications and Occupations https://ec.europa.eu/esco/home.

9. http://www.wandinc.com/wand-skills-taxonomy.aspx

expertise on a new electronic device), the skill name can be added by any participant and then become available for the overall community. Second, these systems can be seen as self-adaptive and self-tuning, as inappropriate or deprecated keywords can be ruled out by the natural effect of collaborative filtering. But flexibility can also be seen as a drawback for this approach :

— keyword dispersion : participants may use very different keywords for the same skill, leading to a blurred description of available competencies ;
— low precision : for a task requester looking for a precise skill $s$, a keyword-based system will return all skills $s'$ equal to $s$ or similar to $s$ up to a given threshold. Hence the task requester is not guaranteed to obtain only the desired skills. As a naïve example, looking for "Java" experts (computer language) could return "Java" experts (the real Island). Using vocabularies and entity resolution to solve this problem means indeed that a kind of taxonomy as the one we propose is required.
— low recall : due to the openness of the vocabulary, a task requester looking for a skill $s$ is never sure to have exhausted all the possible keywords that correspond to her goal.

In this Chapter, we advocate the use of structured vocabularies for skill management. A structured vocabulary can be a tree (a taxonomy of skills) or a more general ontology of skills (that we do not consider in the present work). We would like to illustrate the various benefits we envision :

— Complementarity : it should be observed that our proposal is complementary to keyword-based systems and that existing methods for skill estimation can be enriched with ours (given nevertheless technical adaptation). Our work proposes to structure the similarity between skills, so it could extend the comparison methods used in the related work.
— Reasoning on skills : being structured as a tree, skills can be substituted by more specific or more generic skills with a clear semantics.
— Availability : participatory platforms in science, especially in biology and ecology, are already using taxonomies to structure image labeling and hence participant abilities (see for example the SPIPOLL platform or BumbleBeeWatch). In these domains, taxonomies are already available to apply our technique. The situation is less advanced for generic applications, but as mentioned in the introduction, several generic skill taxonomies already exist (e.g. ESCO).
— Support for crowd assessment : finally, using taxonomies to structure the crowd allows for an efficient design of targeted qualification tests or quizzes. For example, participants can start by answering generic questions (top of the taxonomy) and go on with more and more specialized questions (bottom of the taxonomy).

Table 3.1 sums up this comparison.

| Criteria | keyword-based | taxonomy based |
|---|---|---|
| flexibility | + | - |
| dynamicity | + | - |
| precision | - | + |
| recall | - | + |
| reasoning | - | + |
| stuctured assessment | - | + |

TABLE 3.1 – Taxonomy- vs. keyword-based skill management

## 3.2 Assumptions, Skills, Tasks and Participants

Before formalizing our problem, we summarize the assumptions we made. We suppose that a skill taxonomy is available at the crowdsourcing platform that we can use to model required skills as well as participant expertise (Figure 3.1). We assume that the concepts present in this taxonomy are used (by the task requester) to annotate the tasks which are listed on the platform, with the required skills. We restrict our attention to tasks that only require a single skill. We believe that this assumption is realistic, as such tasks are more adapted for micro-task crowdsourcing. It is also reasonable to assume that in the launch of a crowdsourcing application there will be no knowledge of the participant profiles. This is known as the cold-start problem, it is inherent to social network applications such as crowdsourcing platforms and it is out of the scope of this study as there is related work that studies it (as mentioned in [79] and as studied in [37, 86, 93, 94]). However, in this current chapter, we suppose that the given skills are correct and assessed. Several methods exist for this purpose, such as qualification tests in Amazon MTurk or ground truth questions like the ones presented in [39] and [79]. Another interesting approach is the endorsement of skills as the one proposed in LinkedIn [10]. Eventually, the profiles of the participants can be corrected and updated by their participation on the platform and requester feedback (as in [39]). In Section 4.3.6 we use a simple method based on the above to obtain safer participant profiles using ground truth questions.

## 3.3 Knowledge-intensive Crowdsourcing

Most crowdsourcing research and platforms have been focusing on generic micro-tasks that can be performed by any human. However, there is an increasing need for more specific crowdsourcing that can be handled by a generic platform and where experts or participants with very specific and rare skills could perform knowledge-intensive tasks. To the best of our knowledge, knowledge-intensive crowdsourcing has been first mentioned in [71] and [72] but first mathematically formulated in [79]. In [79] participants are described with skills in a form of several keywords accompanied with an expertise of $[0, 1]$ for each keyword. On the other hand, tasks are described

10. http://www.linkedin.com

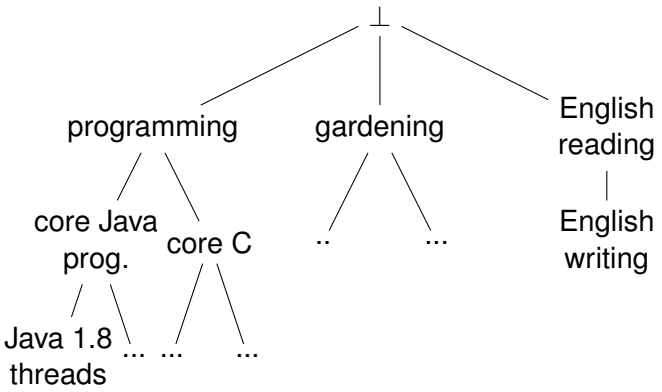



FIGURE 3.1 – A skill taxonomy.

by a requirement of one keyword and an expertise requirement between $[0, 1]$ as well. In our work, [66] the participants and tasks are described with an annotation within a skill taxonomy. Participants can have more than one annotation of skills within the taxonomy while tasks only one. The use of the taxonomy can help us substitute skills that are similar or with more specialized ones.

In Figure 3.1 we can see a sample taxonomy of skills for the purpose of knowledge-intensive crowdsourcing.

More formally, we model a skill taxonomy as a tree $S = (\{s_1, s_2, \ldots\}, \text{is-a}, \leq)$ whose nodes are elementary skills $\{s_1, s_2, \ldots\}$ and $is\text{-}a$ is the parent-child relationship (subclass relationship). Relation $\leq$ denotes the partial order within skills. Taking as example the skill taxonomy of Figure 3.1, for $s = core\ Java$ and $s' = Java\ 1.8\ thread$, then $s \leq s'$. Informally, this partial order means that any participant with skill $s'$ can perform a task requiring skill $s \leq s'$.

Let $depth(s) \in \mathbb{N}$ be the depth of skill $s$ in the taxonomy $S$. We consider only taxonomies with at least two skills (hence $depth(S) > 0$). We use $T = \{t_1, t_2, \ldots\}$ and $P = \{p_1, p_2, \ldots\}$ to denote a set of tasks and participants, respectively. For a given task $t$, we denote the required skill specified by the task requester by $skill(t) \in S$. A skill profile of a participant is the set of skills she possesses. We denote the skill profile of a participant $p$ by $skill(p) \subseteq S$. We insist that $skill(t)$ for a task $t$ refers to a single skill, while a skill profile $skill(p)$ for a participant $p$ might contain several skills.

In this work we suppose that the announced skills are correct (see Section 2 for existing skill estimation techniques). However, we ensure the safety of participant profiles using these simple evaluation methods and calculate their profile. We do not study separately spammers for the sake of simplicity, as they can be ruled out by well-known crowd management techniques, such as majority voting or participant response-quality estimation based on a test set [87].

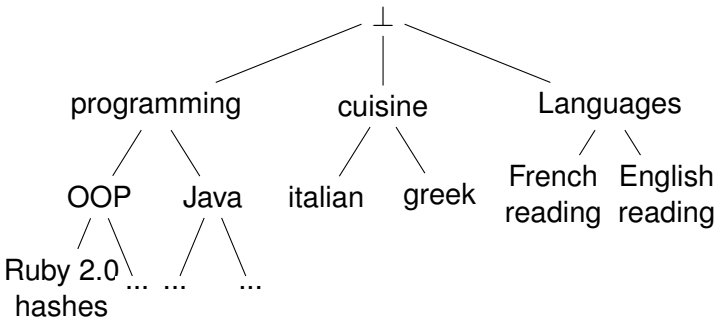



FIGURE 3.2 – Another taxonomy of skills

## 3.4 Skill Distance Definitions

In this section, we will discuss two different distances. One is the graph distance that is relying on the number of edges that separate one task from the other in the taxonomy. The other is a Resnik inspired [78] similarity that we have come up with.

### 3.4.1 Normalized Graph Distance

Firstly, we will define the normalized version of graph distance within a taxonomy. This definition relies on the classic graph distance between nodes in graphs.

**Definition 3.4.1** (NORMALIZED GRAPH DISTANCE BETWEEN SKILLS)

*$S$ is the taxonomy of skills, $d_max$ the maximum depth of the taxonomy and $n_max$ the number of edges that separate the two more distant skills within the taxonomy. Let $n(s, s') \in S$ be the number of edges that connect $s$ and $s'$ in the taxonomy. Then the (normalized)* graph distance *is given by*

$$gd(s, s') = \frac{n(s, s')}{n_{max}}.$$

**Example 3.4.2** *Let us imagine the taxonomy of Figure 3.1. Let $s$ and $s'$ be two different nodes within the taxonomy. A "core Java" and a "gardening" concept. According to the taxonomy in Figure 3.1, $n_max = 5$, $d_max = 3$, then let us consider the following example :*

$$gd(\text{Java 1.8 threads,core Java prog.}) = \frac{1}{5},$$
$$gd(\text{core Java prog.,programming}) = \frac{1}{5},$$
$$gd(\text{Java 1.8 threads,English reading}) = \frac{4}{5},$$
$$gd(\text{Java 1.8 threads,gardening}) = \frac{3}{5}.$$

The "gardening" concept is connected through 3 edges with the "core Java prog." concept. Thus the graph distance of these two notions is 3. To normalize it we divide with the maximum possible distance for this taxonomy which is $n_max = 5$.

### 3.4.2 Normalized Concept Distance within Taxonomy

Our definition is inspired by the classical Resnik similarity [78] that is generally used to measure the similarity between words and concepts while taking into account their relative frequencies. Our definition uses a simplified Resnik similarity for words with uniform distribution.

**Definition 3.4.3** (Distance between skills)
*Let $d_{max}$ be the maximum depth of the taxonomy $S$. Let $lca(s, s') \in S$ be the lower common ancestor of $s$ and $s'$ in the taxonomy. Then the (normalized)* skill distance *is given by*

$$d(s, s') = \frac{d_{max} - depth(lca(s, s'))}{d_{max}}.$$

**Example 3.4.4** *According to Figure 3.1, consider a participant who is knowledgeable in* Java 1.8 threads *and* English reading. *The maximum depth $d_{max}$ of our taxonomy $S$ is 3. Hence*

$$d(\text{Java 1.8 threads,core Java prog.}) = \frac{3-2}{3} = 1/3,$$
$$d(\text{core Java prog.,programming}) = \frac{3-1}{3} = 2/3,$$
$$d(\text{Java 1.8 threads,English reading}) = \frac{3-0}{3} = 1.$$
$$d(\text{core Java prog.,gardening}) = \frac{3-0}{3} = 1.$$

*It is noteworthy that this distance favors close skills that are deeper in the taxonomy. As an example, although* Java 1.8 threads *and* programming *are both separated from* core Java *by one edge,* Java 1.8 threads *is considered closer. Moreover, unrelated skills such as* Java 1.8 threads *and* English reading *have only the root of the taxonomy as a common ancestor, and the distance is then 1 (the maximum).*

### 3.4.3 Concept Distance Selection

We have seen two different distance measures. Let us notice some details about the two definitions and discuss which one seems more suitable for our applications. Let us imagine the example of

$$d(Java\ 1.8\ threads, core\ Java\ prog.) = \frac{3-2}{3} = 1/3$$

and

$$gd(Java\ 1.8\ threads, core\ Java\ prog.) = \frac{1}{5}.$$

At this point, both definitions seem to offer similar properties. However, if we notice :

$$gd(core\ Java\ prog., programming) = \frac{1}{5}$$

and

$$d(core\ Java\ prog., programming) = \frac{2}{3}.$$

we can see that for the graph distance we can not get a different distance for what seems to be deeper in the taxonomy. In other words, this is made even more clear. For Definition 3.4.1 a programming concept has the same distance as a gardening concept which is something that we would like to avoid. Thus we want to favor skills deeper in the taxonomy and give more weight to an edge of a distance that is deeper then, that is only possible with Definition 3.4.3..

## 3.5 Task-to-participant Distance

**Definition 3.5.1** (Distance of task-to-participant assignment)
*By extension, the distance $D(t, p)$ between a task $t$ and a participant $p$ is given by*

$$D(t,p) = \begin{cases} 0 \text{ if } \exists s \in skill(p) \text{ s.t. } s \geq skill(t), \\ min_{s \in skill(p)} d(skill(t), s) \text{ otherwise.} \end{cases}$$

With these definitions, the distance is 0 if the participant has the required skill or she is more specialized. Otherwise, it depends on the distance between the task skill and the best available participant skill. Note however that $d$ and $D$ are not metric distances (no symmetry, no triangular inequality).

## 3.6 Chapter Conclusion

In this chapter, we defined knowledge-intensive crowdsourcing the semantic distance between taxonomy concepts and then built the distance between a single skill microtask to a multi-skill participant respecting the taxonomy of skills.

Briefly, knowledge-intensive crowdsourcing as the name gives away is crowdsourcing of tasks in need of very specific skills. Knowledge-intensive crowdsourcing is suitable for single-skill microtasks and can bring improvements in microtask quality if taken into account.

Our definition of distance between skills in a taxonomy of skills unlike the classic graphic distance is favoring participants that are lower in the taxonomy tree. This is intentional and helps distinguish the differences between a graph distance that will consider a "Programmer" at the same level with a "Gardener".

Since we presented with our taxonomy of skills and distance definitions we focus on our first contribution. In the next Chapter, we introduce the one-to-one task-to-participant assignment problem.

# Chapter 4

# Skill-aware Task Assignment

Besides the simple human intelligence tasks such as image labeling, crowdsourcing platforms propose more and more tasks that require very specific skills, especially in participative science projects. In this context, there is a need to assign tasks to participants based on the required skills for a task and the set of available skills in the crowd, in order to increase the resulting quality. Also, for specific tasks there is a need to require a certain threshold of acceptable quality before we match the task to the participant. As we briefed in Chapter 3, most of the existing solutions rely on unstructured tags to model skills (vector of skills). In this chapter, we use our skill tree model for tasks and participants, that is a taxonomy of skills equipped with a similarity distance within skills (Section 3.4), that enables us to map participants to tasks in a way that exploits the natural hierarchy among the skills and respects task expertise requirements.

In this chapter we formalize the assignment problem of one-to-one task to participant matching with and without expertise requirements. Our contributions are the following :

— we use our previously defined measures for skill distance (Section 3.4) ;
— we formalize the optimization problem of task assignment to most suitable participants in participatory crowdsourcing with and without minimum expertise requirements ;
— we propose several task assignment heuristics that perform better than methods based on unstructured skills ;
— we demonstrate the effectiveness of our heuristics on synthetic and real data sets.

The rest of the chapter is organized as follows. In Section 4.1 we formalize the problem of task assignment in knowledge-intensive crowdsourcing. Then, in Section 4.2 we present our algorithms to address the task assignment problem. We report the results of our extensive experimental evaluation in Section 4.3 and we conclude with a short discussion the chapter in Section 4.4.

## 4.1 Task Assignment

In this section we will model the task assignment problem with and without minimum expertise requirements.

At this point, we remind that we use a taxonomy of skills to represent the skills of both participants and tasks. A taxonomy is depicted in Figure 3.1.

### 4.1.1 Task Assignment without Minimum Expertise Requirement

Given a set of tasks and participants, a task assignment $\mathcal{A}$ is a mapping from $T$ to $P$ that maps a task $t \in T$ to $\mathcal{A}(t) = p \in P$. A task assignment is partial (a task may not be assigned) and injective (a participant can only perform one task during this assignment). As a participant can only participate in one task at a time, the maximum number of tasks that can be assigned is $\min(|T|, |P|)$. Indeed, if there are fewer tasks than participants, some participants may not be assigned. We focus here on *covering task assignments*, where the available workforce is maximally assigned : the number of assigned tasks is $\min(|T|, |P|)$. More subtle models with partial assignments of tasks could be envisioned, where we would prefer not to invite some participants with very low expertise, but we leave these considerations for future work.

If the platform assigns a participant to a task, we assume that she accepts the assignment and completes the task to the best of her knowledge. Remark that to obtain a more realistic model, we could add a probability of acceptance of a task, and a reliability of a participant for a given skill, but this does not lighten the skill mapping problem on its own. For the same reason, we do not map the same task to several participants. However, we can assume as in [9] that there can be several assignments rounds to simulate this feature of multiple assignments. This is more realistic and also used as a classic method for auction-based transactions in economics.

We next model the quality of an assignment. The best situation is to map a task with required skill $s$ to a participant with this exact skill. Note also that a participant with a more specialized skill $s' \geq s$ can perform the task. If such skills are not available in the crowd, more generic participants can be used, but at the expense of a lower quality. In order to capture these situations, we consider a *skill distance* between the required skill and the available ones.

### 4.1.2 Cumulative Distance

Finally, the quality of an assignment $\mathcal{A}$ is measured by the *cumulative distance* $\mathcal{D}(\mathcal{A})$, which is the sum of distances between each pair of assigned tasks and participants, i.e. :

$$\mathcal{D}(\mathcal{A}) = \sum_{(t,p) \text{ s.t. } \mathcal{A}(t)=p} D(t,p).$$

The *normalized cumulative distance* is $\mathcal{D}(\mathcal{A})$ divided by the total number of assigned participants. With this definition, the closer the participants are to the required

skill of their task, the smaller is the distance and the better is the assignment.

We can now define the task assignment problem :

**Definition 4.1.1** (OPTIMAL COVERING TASK ASSIGNMENT PROBLEM)
INPUT *: a taxonomy $S$, a set of tasks $T$ and participants $P$, skill functions.*
OUTPUT *: a covering task assignment $\mathcal{A}$ such that $\mathcal{D}(\mathcal{A})$ is minimized.*

In our model, we want to assign a maximum number of available tasks. If there are more tasks than participants, hence only $|P|$ tasks can be performed during the assignment round. On the contrary, if there are fewer tasks than participants, some participants will not have any task to do on the assignment round. However based on [9] we can assume that there can be several rounds and waiting periods between them which is more realistic for real crowdsourcing and can fill in the gap of more participants or tasks in a given round.

In the following section (Section 4.2) we consider several heuristic algorithms for the task assignment problem.

### 4.1.3 Task Assignment with Minimum Expertise Requirements

The OPTIMAL COVERING TASK ASSIGNMENT PROBLEM tries to find the best assignment possible, given a set of tasks, participants, in the presence of a given taxonomy. In certain cases, task requesters might indicate that they expect a minimum quality of the response. This is indeed a common practice, requesters might choose for example to only accept premium workers for their tasks or request a specific verified skill (such options are available on several, mainstream platforms).

In our model, we can formulate this more general version of the assignment problem as follows. A task requester, at the time he deposes his task $t$, specifies a threshold $\epsilon_t$ that corresponds to his quality expectation. This quality threshold should be respected in any valid assignment, that is

$$D(t, p) \leq \epsilon_t$$

should hold for each task $t$ and assigned participant $p$ (in case this threshold is not specified we assume that the requester has no concerns about the qualifications of a worker, that is $\epsilon_t = +\infty$).

**Definition 4.1.2** (TASK ASSIGNMENT WITH MINIMUM EXPERTISE REQUIREMENTS)
INPUT *: a taxonomy $S$, a set of tasks $T$ (with quality thresholds), participants $P$, skill functions.*
OUTPUT *: a task assignment $\mathcal{A}$ such that :*

- *$\mathcal{D}(\mathcal{A})$ is minimized ;*
- *for each assignment of a participant $p$ to a task $t$, $D(t, p) \leq \epsilon_t$ holds,*
- *the assignment is maximal (we cannot assign any further task without violating the expertise requirement).*

Clearly, assignments with minimum expertise requirements might be partial, that is some workers with poor skill profiles might not get any tasks in the assignment. Specifying quality thresholds can be advantageous for requesters, however, if assignments remain incomplete this might affect the overall appreciation of the crowdsourcing platform. Moreover, it might be not obvious to set appropriate thresholds.

It is noteworthy that solving problems of Definitions 4.1.1 and 4.1.2 also answers natural questions that are of prominent importance for task providers and crowdsourcing platforms. For example :

— Given a set of workers, a set of tasks and a minimum expertise requirement for the overall assignment, what is the percentage of tasks that can be assigned ?
— Given a set of workers, a set of tasks and a minimum expertise requirement for each task, what is the percentage of tasks that can be assigned ?
— Given a set of workers, a set of tasks and a minimum expertise requirement for each task, what is the maximum quality we can achieve for the overall assignment ?

In the following section, we investigate the complexity and potential algorithms for our definitions.

## 4.2 Task-assignment Algorithms

The complexity of the OPTIMAL COVERING TASK ASSIGNMENT PROBLEM we study is in P, as it can be reduced to a Minimum Weight Perfect Bipartite Graph Matching problem. In this encoding, each part of the graph corresponds to tasks and workers respectively, and the weight of an edge between a task $t$ and a participant $p$ is $D(t, p)$. The TASK ASSIGNMENT WITH MINIMUM QUALITY REQUIREMENTS problem can be encoded in a similar way, by setting the weight to $+\infty$ whenever $D(t, p) > \epsilon_t$. There are several variants of the problem (e.g. where participants may take several tasks, or situation where one would like to simultaneously optimize the costs and the quality of the assignment [79]) which are NP-complete. In the following section, we consider first task assignment without minimum expertise requirements.

### 4.2.1 Task-assignment without Minimum Expertise Requirements

As a baseline, we use the Hungarian method [56], the combinatorial optimization algorithm often used to find perfect matchings, to obtain assignments with minimal normalized cumulative distance. We observed that for our specific problem, performance enhancements can be achieved. We considered two different heuristics : MATCHPARTICIPANTFIRST and PROFILEHASH.

Before we go on with the description of the above heuristics, please note that in our implementation, as a natural encoding we represent skills of tasks and participants as a set of words, such that each word denotes a path in the taxonomy $S$. For example, according to Figure 3.1, *core Java programming* is encoded by 00 and *English writing* by 20 (one digit per level).

In MATCHPARTICIPANTFIRST (Algorithm 1), we try to assign the most specialized tasks first to the participants with the lowest number of skills (hence saving most diverse participants for other tasks). More precisely, we reverse-sort the task skills alphabetically, hence the most specific skills of each branch of the taxonomy appear first. For instance, the skill 01243 will appear before 0124. We also sort participants according to their number of skills, so that the least diverse participants appear first. Then, for each distance, starting from 0 to $d_{max}$, and for each sorted task skill, we scan the list of sorted participants and assign the task to the first available participant at this distance. We go on with increasing distances until there is no task or participant left.

The next heuristic, PROFILEHASH (Algorithm 2), uses indexes (hashmap) to organize participants' skills. It implements the following heuristic :

— Try to assign the most specialized tasks first (those that are more difficult to assign).
— For each task, search first for participants with the exact required skill (hence the quality is perfect without wasting more specialized participants).
— If no such participant is available, search for participants with more specialized skills, starting with the least specialized (again, quality will be perfect, and we attempt to save the even more specialized participants).
— If no such participant is available, we progressively relax the skill required for the task (the quality will decrease, but we try to minimize this loss by using the most specialized participants first).

The search order of this heuristic is depicted in Figure 4.1. In order to avoid a systematic traversal of the taxonomy tree, we speed-up the skill search by indexing each participant skills. More precisely, we build hashmaps that associate a skill to a list of participants with this skill. We consider a hashmap for each different skill depth. Also, in order to ease the search of prefixes or extensions of a given target skill, and trading memory for speed, we index also all $prefixes$ of a participant skill. We insert and consume the prefix of each skill in a $FIFO$ order so that we favor the more specialized skills first.

The time complexity of MATCHPARTICIPANTFIRST is composed by the sorting of $T$ and $P$ ($O(|T|\ln|T| + |P|\ln|P|$), and the scan of participants and distances for each task ($O(d_{max}.m_s.|P|.|T|)$), where $d_{max}$ is the maximum distance (depth) of the skill taxonomy and $m_s$ is the maximum number of skills a participant has. Note also that our algorithms require distance computations, that impose lower common ancestor ($lca$) computations in the taxonomy. Constant time algorithms exists [16] (we rely on a simpler, linear time implementation in $O(d_{max})$ in our experiments). Space complexity is limited to input data storage ($O(|T| + m_s|P|)$).

For PROFILEHASH, beside task sorting (time $O(|T|\ln|T|)$), we have to pay for the indexing of all participant skills plus their prefixes. This latter cost a constant time for a hashmap. We also have to include the search for each task skill and for each of its prefix if necessary. This yields a $O(|T|\ln|T| + d_{max}(|P|m_s) + |T|m_s|P|))$ time complexity and $d_{max}m_s|P|$ space complexity for hashmap. We summarize the worst-case complexity of our algorithms in Table 4.1, assuming that $|T|$ and $|P|$ are of the

| Algorithm | Time $O(.)$ | Space $O(.)$ |
|---|---|---|
| HUNGARIAN METHOD | $n^3$ | $n^2$ |
| MATCHPARTICIPANTFIRST | $d_{max}.m_s.n^2$ | $m_s.n$ |
| PROFILEHASH | $d_{max}.m_s.n$ | $d_{max}.m_s.n$ |

TABLE 4.1 – Complexity assuming that $|P| = |T| = n$, $m_s$ is the maximum number of skills of a participant and $d_{max}$ the maximum depth of the skill taxonomy.

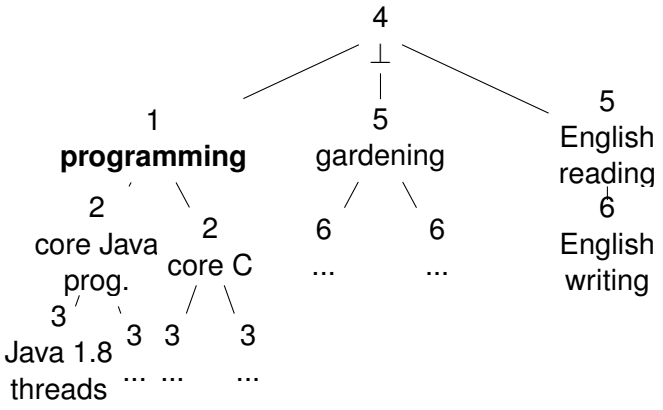



FIGURE 4.1 – Search order, trying to assign a *programming* task (no specified order within skills with the same number on this picture).

same order of magnitude $n$ for readability.

One can observe that the data required to handle our problem is likely to fit in main memory. Indeed, let us consider the realistic Amazon MTurk setting with 500,000 participants and 250,000 tasks. If we suppose one skill per task and a maximum of ten skills per participants and we assume that a skill can be encoded as a ten bytes word (a path in a $d_{max} = 10$ taxonomy), the amount of memory to handle this information is 50 MB for participants and 2.5 MB for tasks, for a total of 53 MB. According to Table 4.1, the maximal space required for running our algorithms for $d_{max} = 10$, $n = 500,000$, $m_s = 10$ and 10 bytes to store a skill is then around ($10.d_{max}.m_s.n =$)500 MB. A typical computer as used in our experiments can handle around thirty-two times this amount.

Note also that the transactional aspect of the problem is negligible : participant and task skills can be stored securely in a database and updated, while the task assignment is performed. The discrepancy between the stored version and the in-memory version will not harm the process, as participant/task skill updates and taxonomy changes are not so frequent.

### 4.2.2 Task-assignment Algorithms with Minimum Expertise Requirements

We consider now adaptations of our algorithms that capture the notion of minimum expertise requirements. We first adapt MATCHPARTICIPANTFIRST (see Algo-

**Data:** Participants $P$, tasks $T$, $skill()$ functions
**Result:** Assignment $\mathcal{A}$
**reverse sort** $T$ according to task skills, alphabetically ;
**sort** $P$ (w.r.t. $|skill(p)|, p \in P$) ;
**foreach** *distance* $i = 0$ *to* $d_{max}$ **do**
  **foreach** *task* $t \in T$ **do**
    **foreach** *participant* $p \in P$*, hence starting with the one with less skills* **do**
      **if** $d(skill(p), skill(t)) \leq i$ **then**
        $\mathcal{A}(p) \leftarrow t$;
        remove $t$ from $T$;
        remove $p$ from $P$;
      **end**
    **end**
  **end**
**end**

**Algorithm 1:** MATCHPARTICIPANTFIRST

rithm 3, where differences are highlighted with symbol $**$). Observe that we consider first tasks with low expertise requirements (i.e. small value of $\epsilon_t$). This means that we try to assign first tasks with constraints easy to satisfy. We continue as in MATCHPARTICIPANTFIRST, but we perform assignments only if the constraint on $\epsilon_t$ is satisfied.

Second, we adapt PROFILEHASH (see Algorithm 4). In the initial version, we favored tasks asking for experts (large skill depth). We choose to favor now higher expertise requirements first, then the expertise depth of the task. We also favor participants with fewer skills, so that to save polyvalent participants for future tasks. As before, we continue as in PROFILEHASH, but we perform assignments only if the constraint on $\epsilon_t$ is satisfied.

## 4.3 Experimental Evaluation

### 4.3.1 Overall Experimental Setting

The evaluation was performed on an Apple MacBook Pro featuring an Intel i7 quad-core CPU running at 2.8 GHz, 16GB of RAM (running on 1600MHz DDR3), 1TB SSD disk and the Mac OS X Yosemite operating system. The code was written in Java and compiled with the latest Java 8 Oracle's compiler. In order to assess our model, we used both a synthetic and a real data set of participants and tasks that we will explain in separate sections. Both the datasets and the code used for the experiments can be provided upon special request.

**Data:** Participants $P$, tasks $T$, $skill()$ functions
**Result:** Assignment $\mathcal{A}$
Initialize $d_{max} + 1$ hashmaps, $M[0], \ldots, M[d_{max}]$;
**reverse sort** $T$ by skill depth ;
**foreach** *distance* $i = 0$ *to* $d_{max}$ **do**
  **foreach** *participant* $p$ **do**
    **foreach** *skill* $s$ *of* $p$ **do**
      /* take $s$ except the last $i$ levels */
      $s' \leftarrow s[0 \ldots length(s) - i]$;
      /* append at the end of the correct skill list */
      $M[|length(s')][s'].append(p)$;
    **end**
  **end**
**end**
**foreach** *distance* $i = 0$ *to* $d_{max}$ **do**
  **foreach** *task* $t \in T$ **do**
    /* take $skill(t)$ except the last $i$ levels */
    $s \leftarrow skill(t)[0 \ldots length(skill(t)) - i]$;
    $C \leftarrow M[length(s)][s]$;
    /* take the first available participant in $C$, while
       respecting the skill specialization order */
    $p \leftarrow C.first()$ ;
    $\mathcal{A}(t) \leftarrow p$;
    remove $t$ from $T$;
    remove $p$ from $P$;
  **end**
**end**

**Algorithm 2:** PROFILEHASH algorithm

**Data:** Participants $P$, tasks $T$, Task requirements $\epsilon_t$, $skill()$ functions
**Result:** Assignment $\mathcal{A}$
**sort** $P$ (w.r.t. $|skill(p)|, p \in P$) ;
****sort** $T$ by expertise requirements ;
**foreach** *distance* $i = 0$ *to* $d_{max}$ **do**
  **foreach** *task* $t \in T$ **do**
    **foreach** *participant* $p \in P$*, hence starting with the one with less skills* **do**
      ****/* take $skill(t)$ except the last $i$ levels */**
      ****$s \leftarrow skill(t)[0 \dots length(skill(t)) - i]$;**
      ****if** *distance* $i < \epsilon_t$ **then**
        **if** $d(skill(p), skill(t)) \leq i$ **then**
          $\mathcal{A}(t) \leftarrow p$;
          remove $t$ from $T$;
          remove $p$ from $P$;
        **end**
      **end**
    **end**
  **end**
**end**

**Algorithm 3:** Adapted MATCHPARTICIPANTFIRST Algorithm. Main adaptations marked with **.

### 4.3.2 Synthetic Data Setting

Our first assessment of the model was carried out with the generation of a synthetic data set that consists of a taxonomy, a set of participants and a set of tasks. The taxonomy $S$ that we used on most cases unless stated differently, was a taxonomy tree of depth ten with ten children per node. Domain-specific taxonomies with $d_{max} = 5$ or more, like the one in SPIPOLL, can be incorporated to more generic taxonomies to create a deeper and greater taxonomy of a $d_{max} = 10$.

Also, for simplicity the taxonomy we used for the Synthetic Data was a balanced taxonomy. However, this is not a limitation because our distance definition (see Section 3.4.2) will favor more specialized skills in case they exist. Please note that in the real setting (see Section 4.3.6) the taxonomy was not balanced.

For the synthetic experiment, we created task skills and participant skills with respect to our taxonomy. Each task is associated with only one skill and is generated as a random path on the taxonomy tree.

Unlike a task, a participant might have multiple skills. To create the participant skills we propose a *budget method*. This method assumes that a participant can have several skills (nodes) on the taxonomy and distributes them randomly, but according to a budget. This is reasonable if we assume that to become an expert in a skill a participant should pass some time on the given domain and that this occupation makes the participant less available to learn another skill in another domain [47]. This method

can eventually create an expert profile or a general knowledge profile. A combination of both in a given domain is also possible.

More precisely we carried out the experiments as follows. First, we generated a number of tasks with the above-mentioned method. Then we generated an equal number of participants, using the budget method. Afterwards, we used the generated tasks and participants as input to both our baseline algorithms and our heuristic propositions. For our experiments, we repeated this procedure ten times so that each point in each figure represents the aggregated result of ten repetitions of random data with the same characteristics (taxonomy, participants, tasks). The variance is also calculated and shown in our figures.

As a baseline we compare with the following algorithms :

— RANDOM and

— EXACTTHENRANDOM.

The RANDOM algorithm assigns randomly tasks to participants. More precisely it shuffles the tasks' ids and the participants' ids and then makes a complete one to one assignment between them. The EXACTTHENRANDOM algorithm matches first each task skill with a participant that has the *exact same skill*, and assigns the remaining tasks and participants randomly. This algorithm recognizes skills, but does not take advantage of the taxonomy structure. Instead, it can be interpreted as a keyword-based (vector of skill) matching of the participant's skills with the tasks.

The figures that follow show how the results we obtained with our different algorithms are affected by the different configurations of the taxonomy, the participants and the tasks. They also demonstrate how in terms of quality our algorithms outperform both the RANDOM and EXACTTHENRANDOM algorithms.

### 4.3.3 Synthetic Data Results and Discussion

To assess our approach we propose *six* different figures, supporting its scalability and improved quality, time cost and effectiveness. We simulate with one curve on each figure the result of each of the five considered algorithms (two baselines, two heuristics and the optimal HUNGARIANMATCH algorithm [1]) that were used :

— RANDOM,

— EXACTTHENRANDOM,

— MATCHPARTICIPANTFIRST,

— PROFILEHASH,

— HUNGARIANMATCH.

On each of the following figures, we assume that the number of participants and tasks are equal which does not harm the generality of our results. For qualitative measurements, we chose a relatively small amount of participants and tasks (1,000 to 3,000). However, in order to show the scalability in terms of participants and tasks for the time needed to perform a complete assignment, we chose higher values of participants and tasks (10,000 to 500,000).

1. Adapted from K. Stern's implementation, `https://github.com/KevinStern`

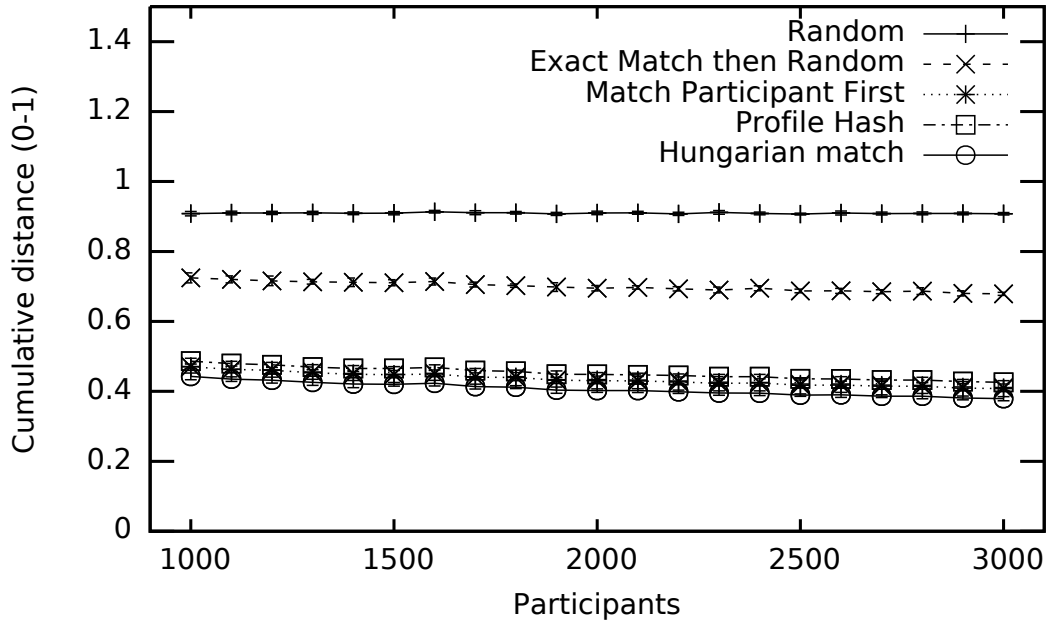



FIGURE 4.2 – Normalized cumulative distance of assignment with respect to the number of participants

Figure 4.2 shows the normalized cumulative distance related to the assignment with respect to the number of participants. Each participant is created with a budget of twenty nodes. We can easily distinguish how RANDOM is outperformed by all the algorithms. This can be interpreted as a motivation for the need of fine skill modeling. On the other hand, the simple exact match assignment of EXACTTHENRANDOM is also outperformed by all our algorithms which strengthens the choice of an inference model for skills. Our PROFILEHASH algorithm perform as good as the exhaustive MATCHPARTICIPANTFIRST algorithm even though with minor differences. It goes without saying that the optimal HUNGARIANMATCH algorithm gives the lower bound of the cumulative distance that we can achieve. It is also noticeable that our heuristic algorithms perform very close to the optimal solution.

Figure 4.3 presents the time (in ms) needed for each algorithm to make a complete assignment of all the participants to all the tasks. We use the same skill budget for every simulated participant as before and the same taxonomy characteristics ($d_{max} = 10$, ten children per node). The x-axis is the number of participants (and tasks) that we keep the same as before. For qualitative reasons, we present the RANDOM, EXACTTHENRANDOM, MATCHPARTICIPANTFIRST and HUNGARIANMATCH algorithms along with the theoretical Hungarian match time for comparison of the time needed for a complete task to participant assignment. Not suprisingly the EXACTTHENRANDOM is the fastest and the MATCHPARTICIPANTFIRST is considerably faster than the HUNGARIANMATCH. We omit the presentation of the PROFILEHASH algorithm here because

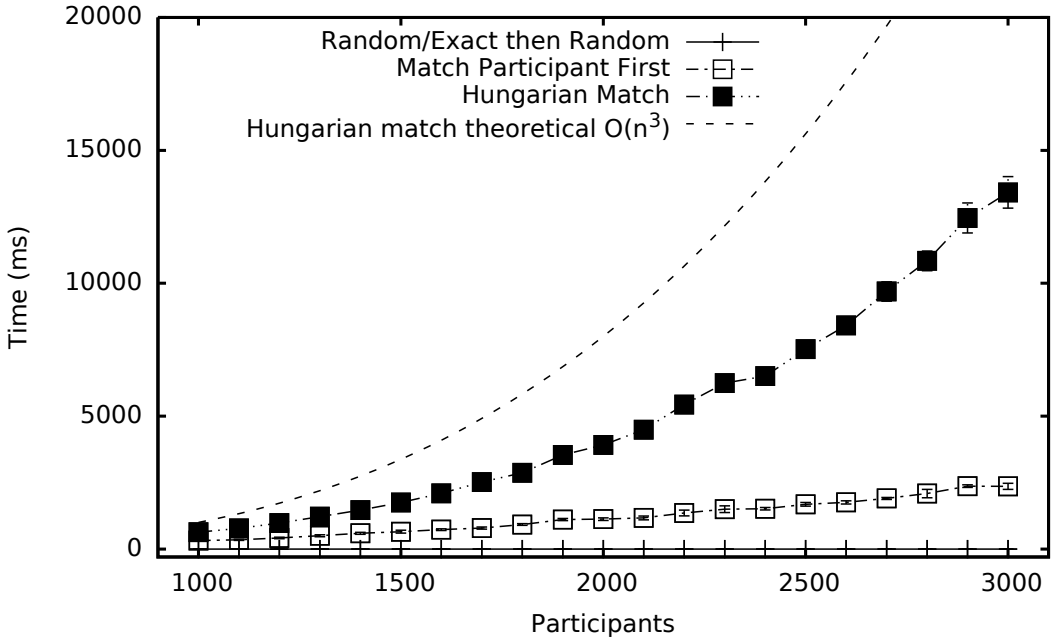



FIGURE 4.3 – Assignment time with respect to the number of participants

of its extreme performance which is the outcome of its efficiently indexed data structures.

In order to show how our algorithms can scale for more participants and tasks in terms of speed, keeping the same taxonomy characteristics and participant budget parameters, we simulate the time needed for a complete mapping of 10,000 to 500,000 participants and tasks shown in Figure 4.4. We compare in this figure only the PROFILEHASH with the baseline RANDOM and EXACTTHENRANDOM algorithms. MATCHPARTICIPANTFIRST and the HUNGARIANMATCH are very slow and their computational complexities and time values show that they are not practical for such high participant and task numbers. We chose these values because currently Amazon MTurk, the oldest and most well-known generic crowdsourcing platform, occupies about 500k participants and hosts about 274k tasks. In our setting, we show that we can accommodate this size with even more tasks (500k instead of 274k). This indicates that both of our algorithms can perform (almost) real-time task mapping on such platforms. Of course RANDOM or EXACTTHENRANDOM algorithms are faster but they lose a lot of potentially good participants (see Figures 4.2, 4.5, 4.6, 4.7). This would be an extreme waste of workforce that should not be missed.

Figure 4.5 shows how the budget on the skills of the participants affects the assignment in finding better mappings. We kept a fixed number of participants (3,000) the same taxonomy properties as before and we experimented with different participant profile budget. The x-axis is the budget for participant skill creation that varies from

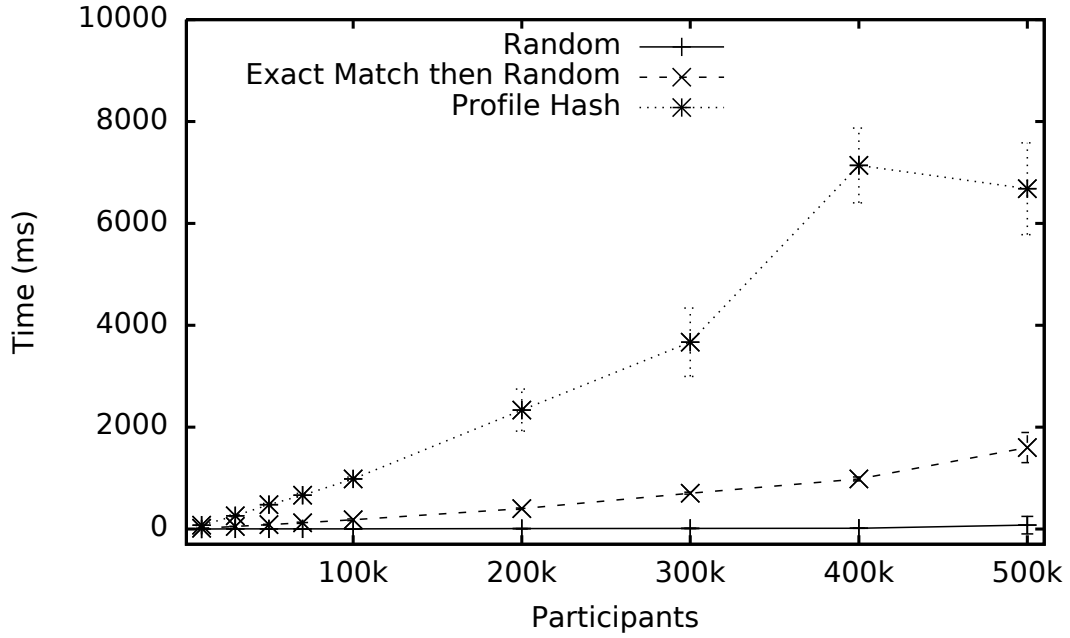



FIGURE 4.4 – Assignment time for larger number of participants

$0.5 * depth$ to $5 * depth$ (5 to 50 for our taxonomy) and the y-axis is the normalized cumulative distance (0-1). We can observe that the quality increased with the availability of more specialized skills. It is equally noticeable that our algorithms outperform the other algorithms in terms of quality at all budget values simulated. Moreover the faster PROFILEHASH perform slightly worse than MATCHPARTICIPANTFIRST. Again we can see that we are not very far from the optimal solution that the HUNGARIAN-MATCH provides.

In Figure 4.6 we show how the depth of the taxonomy affects the quality of the assignment. The x-axis represents the maximum depth of the taxonomy ($d_{max}$) from $d_{max} = 2$ to 20 ($d_{max} = 1$ is not as useful because it would assume falling to the vector-like model). The y-axis is the normalized cumulative distance. Because of the small number of participants and tasks and the great depth of the taxonomy we see that participant skills are quite sparse among the taxonomy and that makes it almost impossible to obtain good matches. Also, it is noteworthy that at about $d_{max} = 8$ we have the maximum gap between EXACTTHENRANDOM and PROFILEHASH algorithms for these number of tasks, participants and current taxonomy setting (10 children per node). While $d_{max}$ is below 4 we can see that both EXACTTHENRANDOM and our algorithms perform very well, practically performing all the possibly good assignments (cumulative distance 0). Again in the general case and for further $d_{max}$ values our algorithms outperform both baseline algorithms. We can again notice that the faster PROFILEHASH perform again slightly worse than MATCHPARTICI-

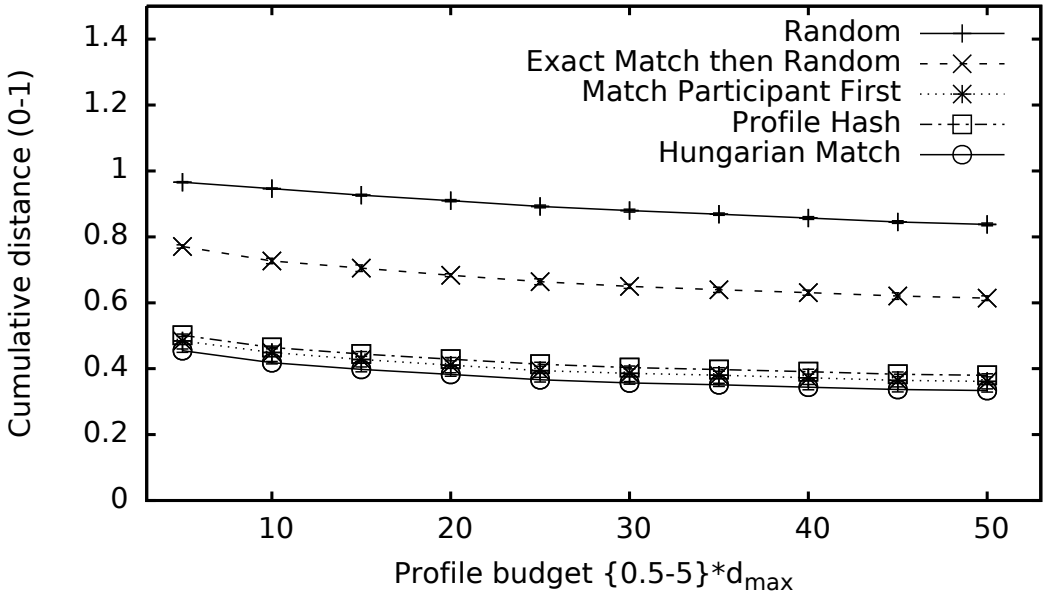



FIGURE 4.5 – Normalized cumulative distance with respect to participant profile budget

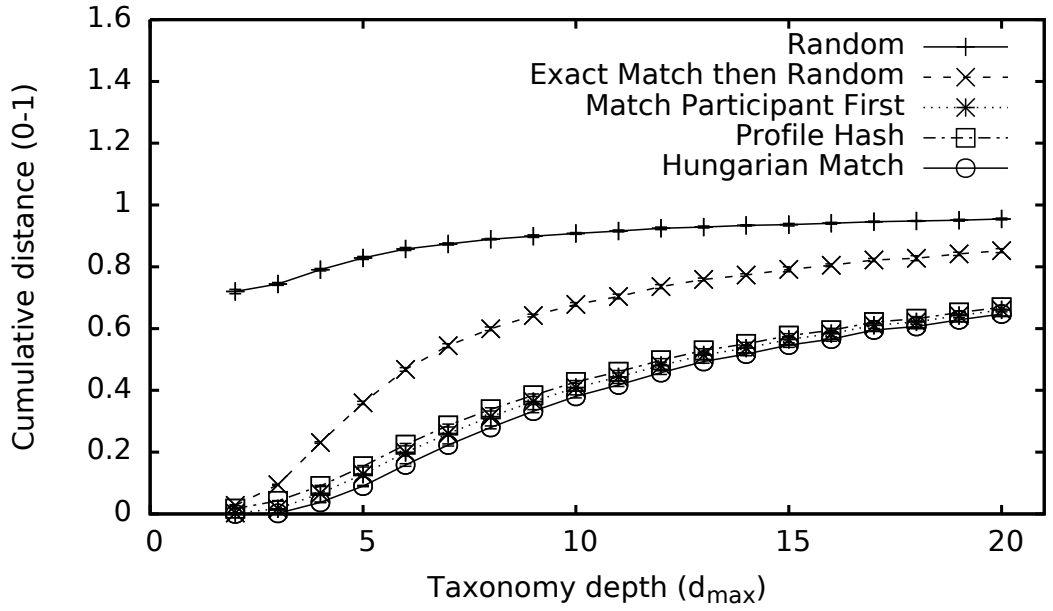



FIGURE 4.6 – Normalized cumulative distance with respect to the $d_{max}$ of the taxonomy

PANTFIRST in this setting. Finally, we can also make another two observations based on this figure. On the one hand, MATCHPARTICIPANTFIRST, PROFILEHASH and HUNGARIANMATCH converge at a great depth which shows the importance of a diversity of skills in a given crowd. An increased budget-to-depth ratio which means having more skillful participants can improve the expected quality and thus decrease the cumulative distance measured. On the other hand, a more numerous crowd assigned to a greater number of tasks can also increase the probability of better task-to-participant matchings which will also lead to a decreased cumulative distance.

Figure 4.7 shows the distribution of assignment distances between algorithms. For this experiment, we keep a fixed number of participants and tasks to 3,000. The x-axis is the distance of quality with respect to the taxonomy while the y-axis is the ratio of tasks assigned per distance. We see that PROFILEHASH algorithm outperforms the other two random based algorithms because they make more assignments to lower distances. The fact that there is an interchange between MATCHPARTICIPANTFIRST and PROFILEHASH has to do with the fact that they are heuristics (do not provide the optimal solution) and that the former is an exhaustive algorithm. It is also noteworthy how the HUNGARIANMATCH would assign to the closest distances in order to give the optimal cumulative distance. We can also notice that due to the Resnik similarity (which is far from the shortest path distance) and the span of task and participant skills, our task and participant generation does not provide a lot of assignments of

short distances (distances 1, 2, 3 and 4 for instance in Figure 4.7). To a greater extent, consider as an example a $d_{max} = 10$ taxonomy, where a depth 4 ($length = 5$) task skill will be matched with a depth 3 ($length = 4$) participant skill (thus having a depth(lca(s,s')=3). Essentially, due to our distance definition, it will obtain a $d(t, p) = 10 - 3 = 7$ distance and not a $d'(t, p) = 4 - 3 = 1$ that would be the value for the intuitive shortest path distance between performed task and participant skills.

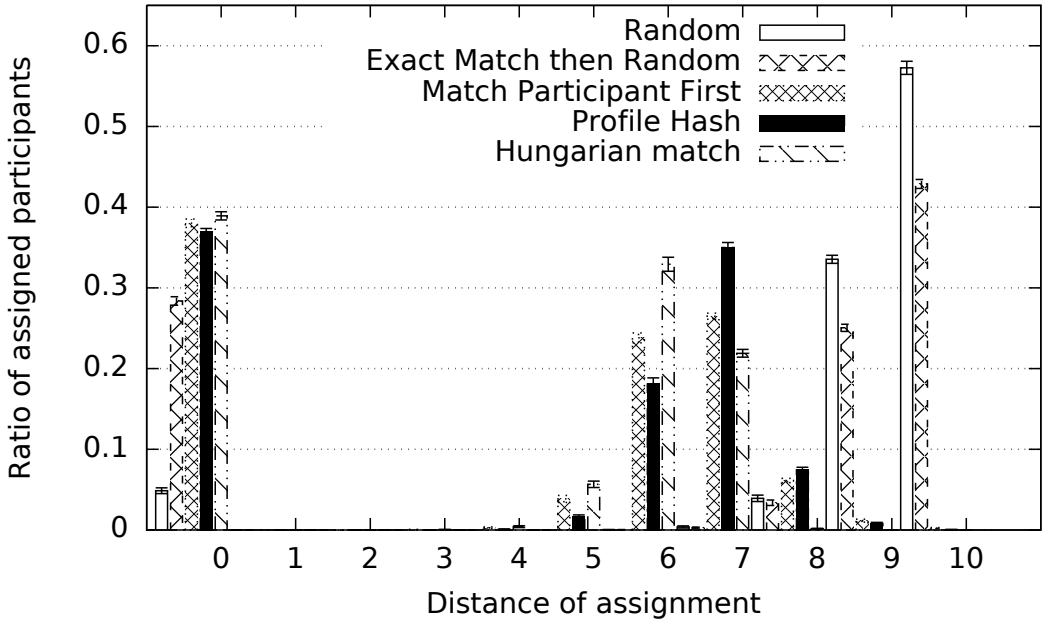



FIGURE 4.7 – Ratio of assigned participants per distance

### 4.3.4 Evaluation with Minimum Expertise requirements

After having evaluated our method without minimum expertise requirements we now focus on the effect of the quality requirements for the tasks. We also adapt apart from our algorithms that we have shown the previous baselines to perform the assignments only when the minimum expertise requirement is met. Please note that for comparison reasons the tasks that do not meet the quality requirement we assign them to maximum distance so that all tasks are assigned to some distance.

Our evaluation uses a synthetic crowd with the same methods as Section 4.3.3(taxonomy of depth ten and ten children per node and a different number of participants) unless differently stated. The experiments are repeated ten times and the average of the results with the standard deviation are presented in the following graphs. The difference now stands on the fact that we generate a strict minimum task expertise requirement for each task by giving the immediately higher level on the taxonomy of the task. For

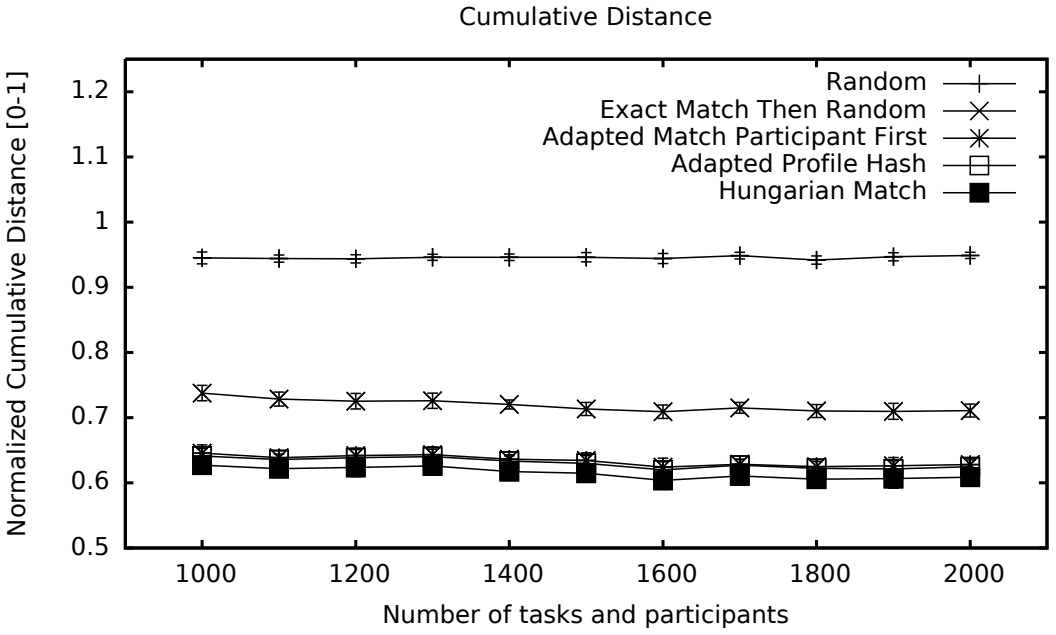



FIGURE 4.8 – Normalized cumulative distance of task assignments for different participant and task numbers. Smaller values are better.

instance on the taxonomy of Figure 3.1 if the task was on "Java 1.8 Threads" we would constrain the assignment to participants that know at least "Java".

Figure 4.8 shows the cumulative distance for each method with this strict constraint for different numbers of participants. For this graph the lower the value the better the performance. We can see that the baselines RANDOM and EXACTTHENRANDOM perform worse than our methods. Also, we can see that our two proposed algorithms, Adapted MATCHPARTICIPANTFIRST and Adapted PROFILEHASH, perform practically equally good with the optimal one (Adapted HUNGARIANMATCH) in terms of quality. The slight differences between the optimal and our two methods are explained in the next figure.

Figure 4.9 focuses on the differences of our the optimal and our methods. As we see Adapted MATCHPARTICIPANTFIRST has a slightly worse performance than the other two Adapted PROFILEHASH andHUNGARIANMATCH but there is no practical difference. Also as we saw in the previous Section (Section 4.3.3) in Figure 4.3 and Figure 4.4, there is an important cost in terms of time for that slight improvements and a memory trade-off (Table 4.1) to have the speed improvements of Adapted PROFILEHASH. We can also observe that when we generate more tasks and participants we get an improved result which is expected.

Figure 4.10 shows the ratio of the number of assignments made by each algorithm normalized with the maximum number of assignments. The bigger the value on this graph the better the method. We can see that we have an important improvement

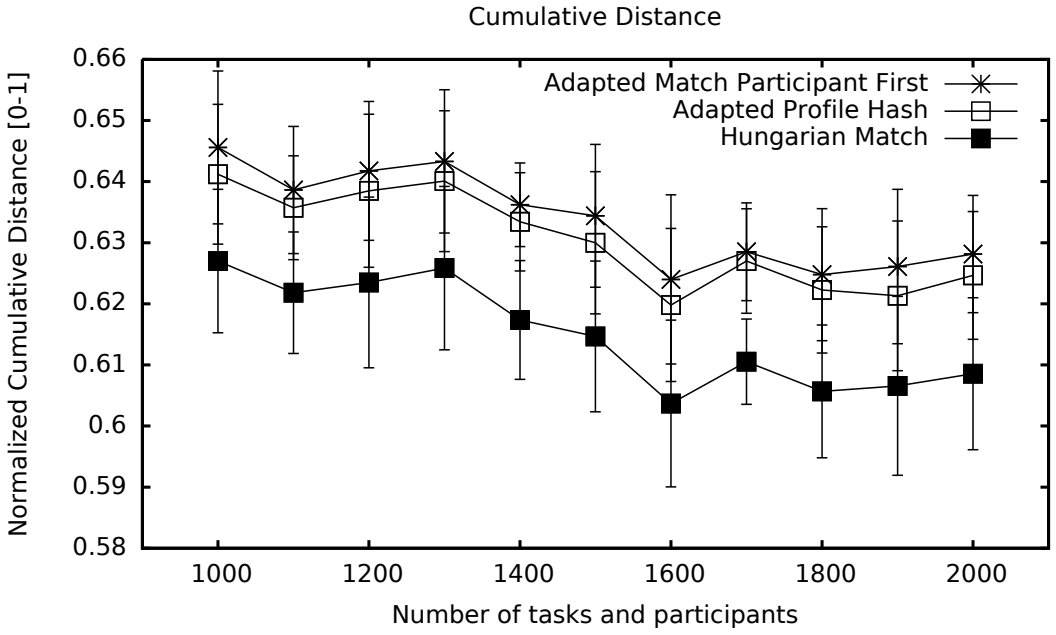



FIGURE 4.9 – Focus on Figure 4.8. Smaller values are better.

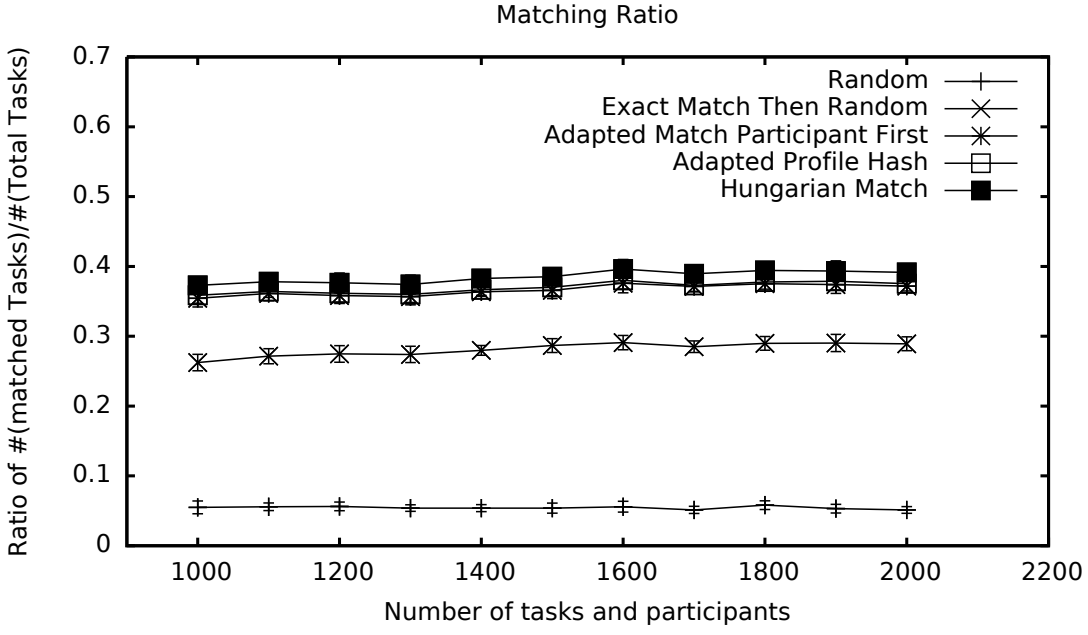



FIGURE 4.10 – Ratio of matched tasks to participants compared to total number of tasks for different participant and task numbers. Bigger values are better.

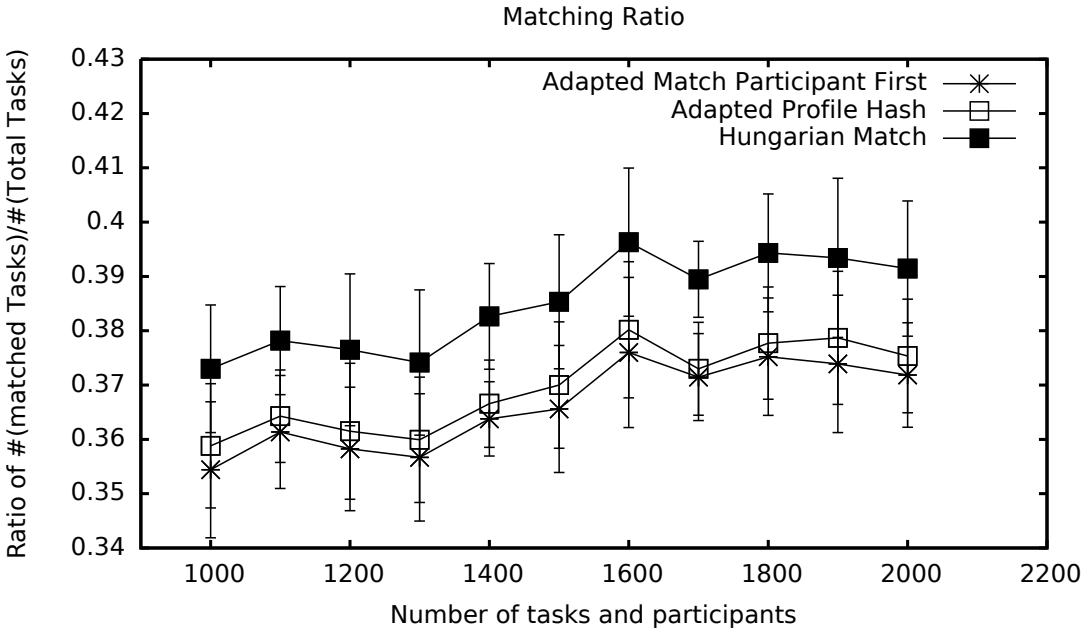



FIGURE 4.11 – Focus on Figure 4.10. Bigger values are better.

on the number of tasks assigned compared to the tasks that are not assigned if we compare our algorithms to the baseline methods. This is partial because the taxonomy will favor some participants with the more specialized knowledge and because we can reason about the immediate level of less specialized participants. As before we can observe that when we generate more tasks and participants we get slightly better results.

In Figure 4.11 we can see that HUNGARIANMATCH is providing with the best ratio and thus more matches. Actually, this is also the best possible for each setting since HUNGARIANMATCH gives the optimal.

Then we generate the Cumulative distance and matching ratio results for different minimum task expertise requirements apart from the strict requirement that we mentioned above. We keep the same setting for the taxonomy but we generate the above metrics for two thousand tasks and participants. For each graph, we gradually relax the task requirement for the tasks and we can see how this influences the cumulative distance and the ratio of the matched tasks.

In Figure 4.12 we can see how the Cumulative Distance for each method on different task requirement levels. We gradually relax the task requirement for the tasks and we can see how this influences the cumulative distance. We are expecting an improvement since the matchings that are not performed are done at a maximum distance. As we can see our methods outperform the baselines each time. We can also observe that the optimal HUNGARIANMATCH is giving the lower bound for the normalized cumulative distance and that the Adapted MATCHPARTICIPANTFIRST and

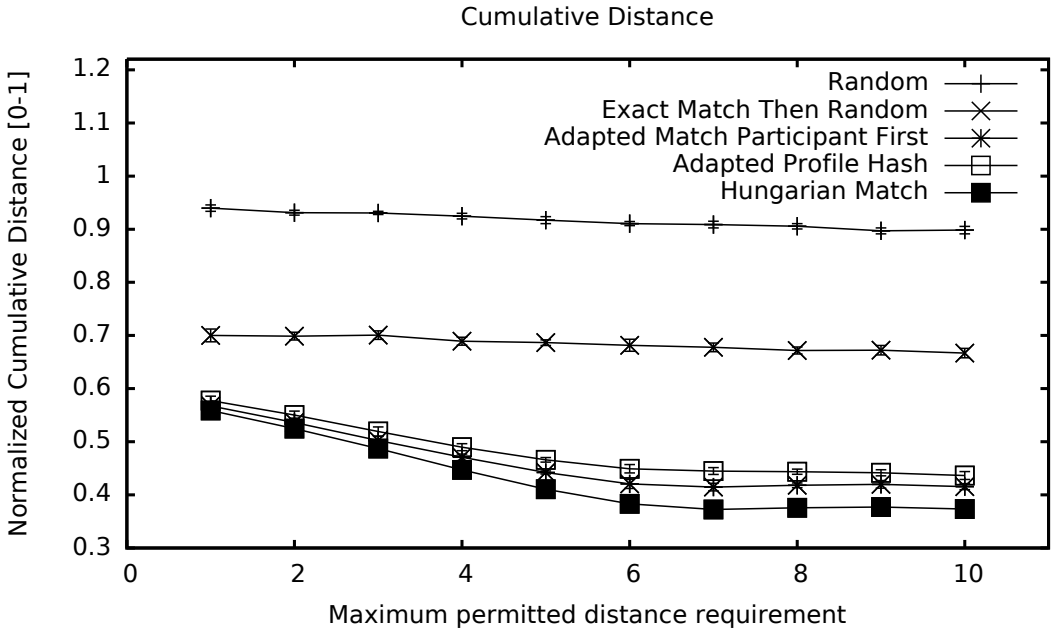



FIGURE 4.12 – Cumulative distance of tasks assignments for different task expertise requirements. Smaller values are better.

Adapted PROFILEHASH are practically indistinguishable.

In Figure 4.13 we can see how the different minimum expertise levels affect the different methods. As in the previous graphs, our methods outperform the baselines and we can see at which point all of the tasks are matched with the participants. On this case, the HUNGARIAN MATCH gets to this point faster than all the other methods even though this is practically indistinguishable from Adapted PROFILEHASH and Adapted MATCHPARTICIPANTFIRST.

### 4.3.5 Synthetic Results Discussion

As mentioned, we performed a series of extensive random generated data experiments to show the significance of our model and assignment algorithms. To the best of our knowledge, we are the first to mention and implement a concrete model that uses fine skills based on a taxonomy. Anticipating the results of our experiments we can observe that in all cases the heuristics we provided give significantly better results than the RANDOM or EXACTTHENRANDOM algorithms. We believe that the results speak loud themselves concerning the need of a finer model of skills for both tasks and participants in knowledge-intensive crowdsourcing. They also show that Adapted PROFILEHASH is an excellent candidate for a scalable, high-quality mapping algorithm while at the same time respecting the minimum task expertise requirements. In Section 4.3.6 we will see how the above is supported by a real experimentation based

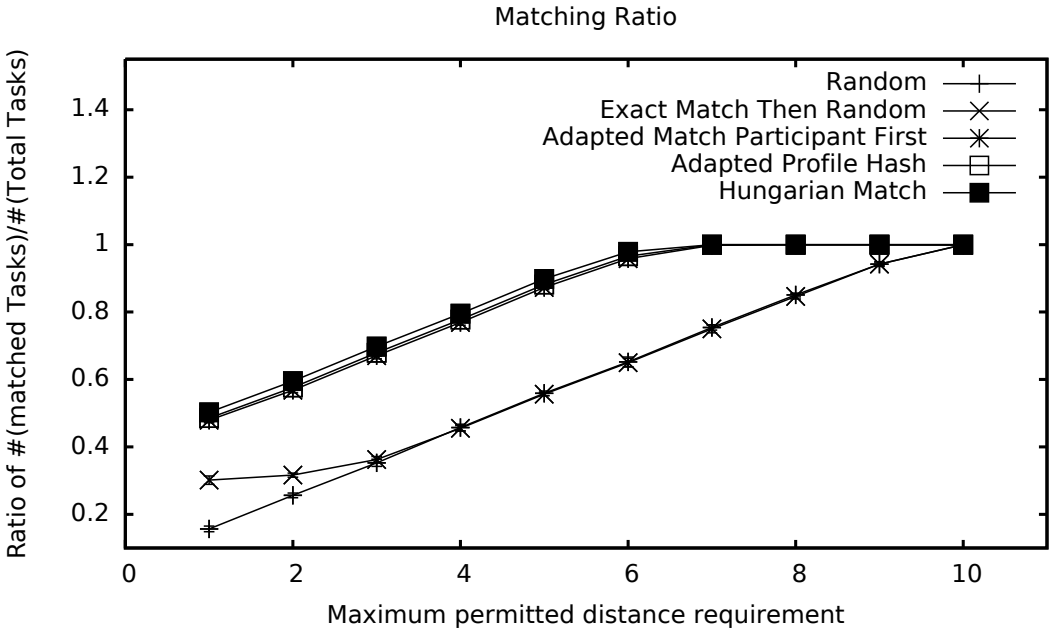



FIGURE 4.13 – Ratio of matched tasks to participants compared to total number of tasks and participants for different task expertize requirements. Bigger values are better.

on a real dataset and different crowds.

### 4.3.6 Real Data Setting

In order to obtain a more realistic data set and test our algorithms we followed a quiz procedure and recruited participants from different sources (our University laboratory and CrowdFlower). As a topic, we chose computer science, a topic where we could elaborate a skill taxonomy, asked participants for their knowledge profile beforehand, assessed their profiles and then gave them the quiz to answer.

More precisely, we chose fifty-eight multiple choice questions on computer science that we could easily elaborate a taxonomy of skills $S$. The taxonomy had a $d_{max} = 4$ and a different number of children per node depending on the node and category which in our case was at least two. In this case, the taxonomy was not symmetrical. Then we asked participants with different backgrounds and mostly computer programming to choose four of their preferred programming languages from a list. This first step provided us with participants with a maximum of four skills each (respecting the fixed profile budget used in Section 4.3.3).

The procedure continued with the real quiz that consisted of fifty-eight multiple choice computer science questions mainly on programming languages, generic computer science, databases and computer architecture questions. All the questions were preannotated with respect to the taxonomy.

In order to carry out the questionnaire, we recruited two different groups of participants. The first group consisted of 31 participants from the University of Rennes 1 that were either students in computer science or computer engineers or computer scientists. All these participants had a proven experience in computer science. The second group was a group of 176 participants that were recruited from CrowdFlower after six qualification questions (gold mining questions in CrowdFlower). In addition, we also considered a third group made of the combination of these two groups (207 participants) which provides with a more diverse case of crowd.

Finally, before applying the algorithms we assessed the profiles given from the participants using test questions on the skills they submitted. Then we applied the two baseline algorithms (RANDOM and EXACTTHENRANDOM) and our PROFILEHASH algorithm in order to obtain the results. Having the ground truth for all the questions and the answers from all the participants we could easily assess the quality of the task to participant assignment. After one hundred repetitions of the assignment algorithms, we obtained results supporting our choice of model and methods.

### 4.3.7 Real Data Results and Discussion

In order to show the feasibility of our approach, we run two baseline mapping algorithms and one of our algorithms (PROFILEHASH) on the questions and the participants. Unless stated differently we also simulate the three different crowds mentioned below :

— Laboratory participants,
— Crowdflower participants,
— Mixed Crowd.

Every bar on the figures presents the average results of one hundred repetitions for the mapping algorithms applied and mentioned below :

— RANDOM,
— EXACTTHENRANDOM,
— PROFILEHASH.

The first crowd (Laboratory participants) consists of the thirty-one participants familiar with Computer science. These participants including graduate students, researchers and engineers were recruited from our laboratory and agreed to participate voluntarily by replying to the survey we made in Google forms for them. The same survey was also given to the Crowdflower participants. The profiles of the participants were verified with test questions. One test question for each skill of the participant. Then the different algorithms were applied to perform the question-to-participant mappings (one to one). Finally, with the help of the ground truth questions, we could calculate the efficiency of the assignments. We discuss the results of these simulations below with the help of two figures.

Figure 4.14 shows the ratio of correct answered questions with respect to the three different crowds. We can observe how our method outperforms the baselines in all the different cases and how the different crowds affect the percentage of correct answers. It also supports the intuition that with crowdsourcing the good use of a greater number

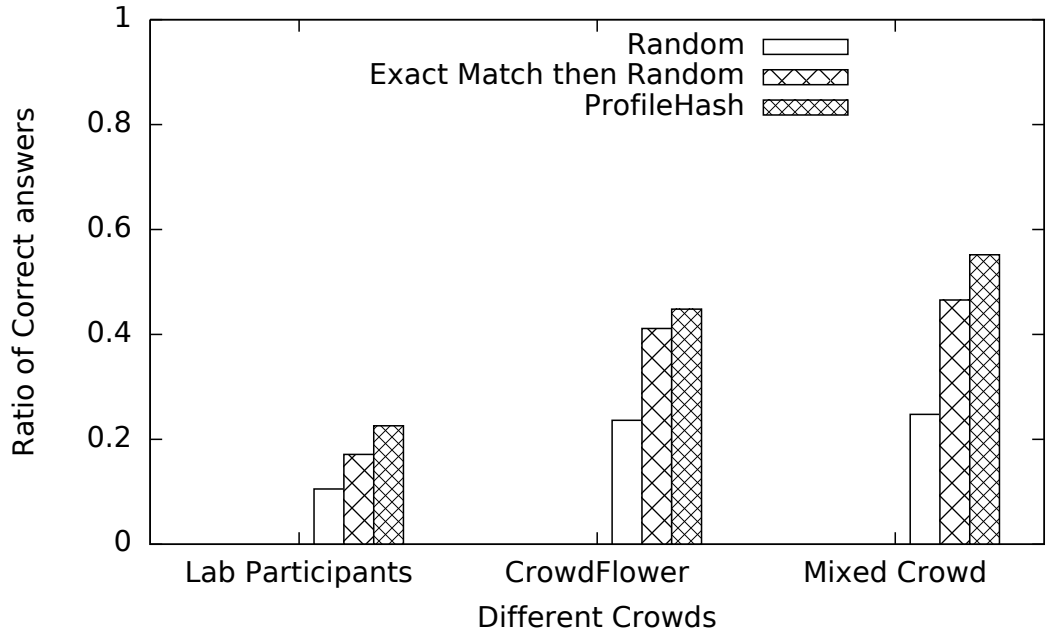



FIGURE 4.14 – Ratio of correct answers with respect to different crowds and algorithm comparisons. Questions are annotated lower in the taxonomy.

of participants (from the laboratory participants towards the mixed crowd) improves the result. In addition, it shows that even a simple model of a vector of skills, such as the one simulated by EXACTTHENRANDOM can improve significantly the expected results compared to the RANDOM method.

Figure 4.15 shows again the ratio of correct answered questions with respect to the different crowds but this time the questions annotated two levels higher in the taxonomy. We performed this test to simulate the realistic case where a question could be annotated at a higher level. For instance, imagine a Java question on sockets. This is clearly a networking task in Java and should be annotated lower in the taxonomy. It should be neither annotated as core Java nor as Object Oriented Programming. However such mistakes or lack of knowledge for a given task could occur in crowdsourcing platforms. With this figure, we show that when the questions are poorly annotated we can obtain a greater result than that of the baselines because our model supports the search for more specialized participants. This is very important especially for real-life crowdsourcing where automatic keyword similarity methods annotating the questions, such as the ones described in [39], could fail capturing the most specialized and important keywords that characterize the question and thus gives more robustness to our model.

To sum up, in this experimental section we saw how the different crowds and task annotations worked in favor of our model and provided with very interesting results. According to Section 4.3.3 and more precisely figure 4.6 we were expecting

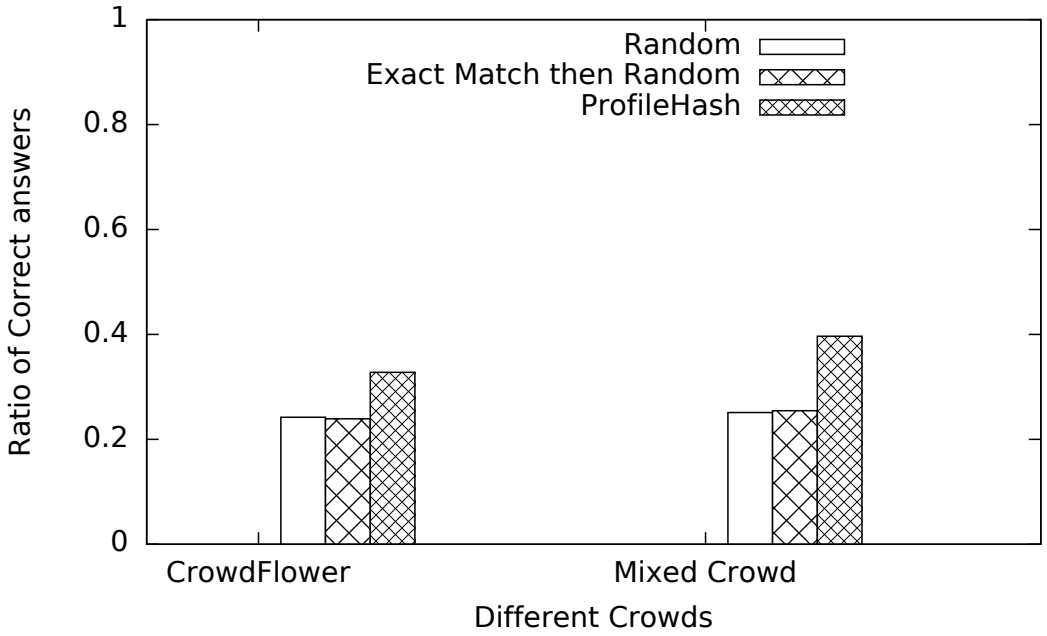



FIGURE 4.15 – Ratio of correct answers with respect to different crowds and algorithm comparisons. Questions are annotated higher in the taxonomy.

to have an observable improvement with the use of taxonomies that have $d_{max}$ >5. However, we could show that even less deep taxonomies equipped with our model and mapping algorithms could give a real life improvement. Even when there was lack of information on the tasks our algorithm performed a better mapping that provided better results. All the above showed the robustness of our model despite the potential existence of spammers (people lying about their skills, or giving bad answers). To the best of our knowledge, we are the first to use such a hierarchy of skills for reasoning on the substitution of participants on crowdsourcing applications and equip it with similarity metrics for better task to participant assignments.

## 4.4 Chapter Conclusion

In this chapter, we have demonstrated the use of taxonomy-based skill modeling for crowdsourcing. Our techniques allow a simple form of reasoning about skills and participant substitution that is particularly useful for optimizing task assignment quality. We proposed several heuristics for task assignment to participants and evaluated their respective performances in terms of quality and scalability through extensive experimentation. Since we have seen the one-to-one task-to-participant optimized matching we will introduce in the next chapter an extended approach where participants are assigned to a list of tasks, given certain criteria and can select one of the tasks.

**Data:** Participants $P$, tasks $T$, Task requirements $\epsilon_t$, $skill()$ functions
**Result:** Assignment $\mathcal{A}$
Initialize $d_{max} + 1$ hashmaps, $M[0], \ldots, M[d_{max}]$;
****reverse sort** $P$ by number of skills ;
****sort** $T$ by task expertise requirement, then by skill depth (descending);
**foreach** *distance* $i = 0$ *to* $d_{max}$ **do**
  **foreach** *participant* $p$ **do**
    **foreach** *skill* $s$ *of* $p$ **do**
      /* take $s$ except the last $i$ levels */
      $s' \leftarrow s[0 \ldots length(s) - i]$;
      /* append at the end of the correct skill list */
      $M[|length(s')][s'].append(p)$;
    **end**
  **end**
**end**
**foreach** *distance* $i = 0$ *to* $d_{max}$ **do**
  **foreach** *task* $t \in T$ **do**
    ****if** *distance* $i < \epsilon_t$ **then**
      /* take $skill(t)$ except the last $i$ levels */
      $s \leftarrow skill(t)[0 \ldots length(skill(t)) - i]$;
      $C \leftarrow M[length(s)][s]$;
      /* take the first available participant in $C$,
         while respecting the skill specialization order
         */
      $p \leftarrow C.first()$ ;
      $\mathcal{A}(t) \leftarrow p$;
      remove $t$ from $T$;
      remove $p$ from $P$;
    **end**
  **end**
**end**

**Algorithm 4:** Adapted PROFILEHASH algorithm. Main adaptations marked with **.

# Chapter 5

# Multi-objective Task Selection

Crowdsourcing has become a popular way [1] [82] to recruit human individuals at scale for different purposes such as simple micro-tasks (e.g. image annotations) or very complex tasks (e.g. innovation or text synthesis). When using crowdsourcing platforms like Amazon MTurk [2], Crowdflower [3] and FouleFactory [4] the main challenges include quality of the data collected from the platform [45] and execution time guarantees [32].

While recent research has focused on improving crowdsourcing quality and throughput using Human Intelligence Task (HIT) allocation [17, 31, 32, 46, 66], such methods assume a *push* crowdsourcing scenario where tasks are unilaterally allocated to available participants by the platform according to different requirements [58]. On the contrary, popular paid crowdsourcing platforms such as Amazon MTurk follow a *pull* model [58] where participants select from available HITs on a first-come-first-served basis. Such paradigm gives a choice to crowd participants about what HIT to work on based on the HIT type, reward, amount of available work, requester reputation, etc.

Typically, such platforms present participants with a list of hundreds of thousands available HITs to choose from. A sorting, in descending and ascending order of particular criteria might also be available along with a search box, without much more to support their HIT choices. This task selection step takes time out of that needed for the HIT completion (as shown in Section 5.3.2).

In this chapter, we propose to combine the push and pull crowdsourcing paradigms by, on the one hand, giving participants a choice between few HITs selected for them and, on the other hand, deal with efficiency and effectiveness requirements by matching HITs to participant profiles (e.g., [12]) and prioritizing urgent tasks. Examples of such an approach include community QA platforms like Quora where 2-3 tasks are suggested to users each time they are visiting the website. This combination of push and pull crowdsourcing allows to present participants with $k$

1. http://www.mturk-tracker.com/
2. http://mturk.com
3. http://www.crowdflower.com
4. http://www.foulefactory.com

HITs that (1) match well their participant profile (mainly skills) ; that (2) is diverse and thus gives the participant a choice between different HIT types and content ; and that (3) ranks higher more urgent HITs giving them a higher chance of being selected by participants. Our approach can, for example, reduce the number of annotations needed for an image annotation HIT given that it will target more relevant and available participants thus reducing the cost and completion time of this job.

More precisely, we propose a method to present crowdsourcing participants with HITs ranked in a multi-criteria manner in order to optimize both the platform and participant requirements. Our contributions are the following :

— we define a HIT ranking problem that aims at optimizing task recommendation relevance while giving participants a choice of prioritized and diverse HITs,
— we provide an extensive synthetic experimental setting that shows the effectiveness of our methods,
— we experimentally compare different HIT suggestion strategies involving 300 participants from a popular crowdsourcing platform,
— we demonstrate that participants prefer our suggestion strategies and select tasks faster with the use of our suggestion mechanism.

We organize the rest of the chapter as follows. In Section 5.1 we formulate the task ranking problem and give details about our task ranking models, given a participant profile. In Section 5.2 we present our algorithm to address the task ranking problem. We report the results of our extensive experimental evaluation in Section 5.3 and we conclude the chapter in Section 5.4.

## 5.1 Task Selection Model for Knowledge-intensive Crowdsourcing

### 5.1.1 Assumptions on Tasks, Participants, Platform

We assume that the style of platform studied is similar to Amazon MTurk and Crowdflower where participants can perform microtasks. The microtasks are described in the platform with fields such as :

— task id,
— task title,
— time left,
— judgments (answers required) left,
— reward.

There might be more depending on the exact platform but we mention only the ones we will need for our model.

We also have no particular assumptions on the distribution of arrival on participants skills and we assume that by the time the participant arrives we know his skills and that they are correct. The other assumptions are the same as the ones mentioned in Section 3.2. We also assume that we have at our disposal a taxonomy of skills like the ones presented on Figure 3.1 and Figure 5.1.

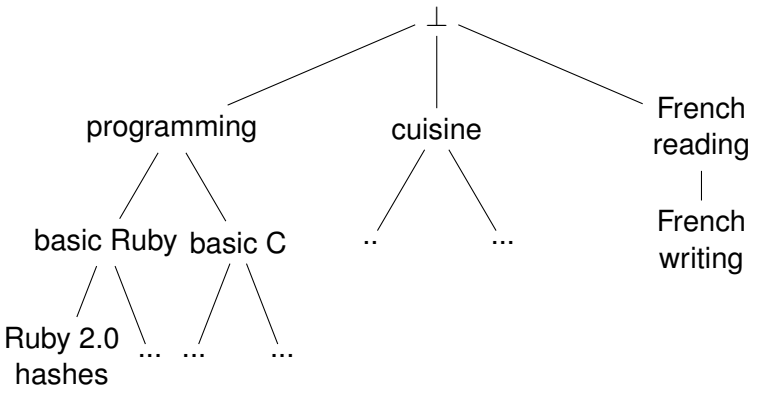



FIGURE 5.1 – Another taxonomy of skills

### 5.1.2 Model and Problem Definition

In the following, we model the task selection setting that we propose. We also describe the particular ranking problem we study. The model that we adopt for task selection is similar to the setting that is currently in use at commercial crowdsourcing platforms. The platform disposes a number of tasks $T = \{t_1, t_2, \ldots\}$ that should be completed by a set of registered participants $P = \{p_1, p_2, \ldots\}$. The task selection is realized in two phases :

1. task proposition : the platform proposes a set of tasks $l_i$ to each participant $p_i$ of size $k$, that is $l_i = \{t_{i_1}, \ldots, t_{i_k}\}$ and
2. task selection : each participant selects a task from the proposed set $l_i$.

We say that a task $t$ is assigned to a participant $p$, if $p$ has obtained the set $l_i$ that the platform proposed to him and has chosen a task from this set. We use $k$ to denote the size of the proposed set.

In this chapter, we focus on task proposition where we provide synthetic data experimentation that shows the improvement of our methods in terms of relevance and diversity of tasks suggested to participants. We also study with real crowdsourcing participants the way they select tasks according to parameters such as task title, reward, requester etc.

Before we present the main problem we study, we discuss some possible ways of assessing the quality of a task proposition. We will use these measures improvements of task proposition strategies.

#### Relevance

To define the relevance of a set of tasks to a participant, we will rely on a distance measure defined in Section 3.4.2. We believe that this distance measure is particularly suitable for knowledge-intensive crowdsourcing as it is related to a hierarchical skill

taxonomy. More precisely, we want to use this definition because it favors skills that are deeper in the taxonomy and as a design principle it favors our prosper definitions for relevance and diversity.

To define the skill distance between two concepts in the skill taxonomy and the assignment distance between task and participant we adopt the following definition from Sectionchap3 :sec :adaptedDistance :

— distance between concepts (Definition 3.4.3),

— distance between task and participant skills (Definition 3.5.1).

Now having in mind these two basic definitions we define the new metrics that will allow us to measure the quality of our task list. We begin with the definition of **Relevance**.

**Definition 5.1.1** *Relevance*
*Relevance of a participant $p$ related to a given task $t$ is defined as $r(t,p) = 1 - D(t,p)$. We define the relevance of a proposed set $l = \{l_1, \ldots, l_k\}$ as an average of the items distance between a participant $p$ and the tasks of the proposed list. $R(l,p) = \sum_{t \in l} \frac{r(t,p)}{|l|}$.*

**Diversity**

We are interested in characterizing the diversity of a set of tasks. There are a number of ways to do this, for example, one could use the number of distinct elements in the list.

**Definition 5.1.2** *Distinct elements of a set*
*We first define the intrinsic diversity of a proposed list as the number of distinct items on it. The normalized diversity is computed dividing the number $N$ of current distinct items in the list $l$ by the number $k$ of items in the list : $D(l) = \frac{N}{k}$.*

While the definition of distinct elements in a list could be useful in certain context, it does not take into account the taxonomy of skills that could offer a more suitable way of defining the diversity. If the tasks on the list are close to each other (w.r.t. the taxonomy) one would consider the list less diverse. For example one would consider the list of 3 elements ["Core Java", "core C++", "core Python"] less diverse than ['"core Java", "Gardening", "Photo editing"]. The following definition relies on the pairwise of distances between the proposed tasks.

**Definition 5.1.3** *Diversity*
*We define the diversity of a list $l$ as : $D(l) = \sum_{i=1}^{k} \sum_{j=1}^{k} \frac{d(t_i,t_j)}{k(k-1)}$, where $t_i$, $t_j$ are different tasks of the list $l$ (of size $k$) and $d(t_i,t_j)$ is their distance w.r.t. the taxonomy.*

As the definition of diversity relies on the task-participant distance, we can observe that the value of the diversity of a set falls between 0 and 1. The values close to 0 correspond to the "less diverse" lists, while higher values characterize "more diverse" lists.

**Definition 5.1.4** *Effective Diversity*
*Effective diversity is the number of different skills of the participant used in the list proposed. We define Effective Diversity, $D_{e_i}$ for a proposed list to a participant $i$ as $D_e(l,p) = R(l,p)D(l)$, where $R(l,p)$ and $D(l)$ are the relevance and diversity of the corresponding list to the participant respectively.*

Note that effective diversity is not the same as relevance. One can achieve maximum relevance with the creation of a task list with tasks that need the exact same skill. On the other hand, maximum effective diversity will be achieved by the use of most of the actual or close to the actual participant's skills to create the proposed task list. Moreover, a proposed list of tasks can be very diverse but the proposed tasks could be unrelated to the participant. We are interested in optimizing the effective diversity rather than the diversity. Figure 5.4 of Section 5.3.3 illustrates this.

**Urgency**

In order to characterize priorities for more urgent tasks, we define an urgency metric. Before defining the urgency $U(l)$ for a list of tasks $l$, we need to define the urgency for each task. In order to define the urgency $u(t_j)$ of a task $t_j$, we have to define two sub-urgencies : (1) u1-urgency, related to the time left for the task to be completed and (2) u2-urgency, related to the judgments pending for the task to finish.

In a classic crowdsourcing platform there is a time deadline for each task. With the following definition if there is a lot of time left for the task, the $u_1(t_j)$-urgency is close to 0. We define $u_1(t_j)$ for a task $t_j$ as :

**Definition 5.1.5** *u1-urgency*
*We define $u_1(t_j)$-urgency as :* $u_1(t_j) = \begin{cases} 1 \text{ if } timeleft(t_j) < avgtime(t_j) \\ \frac{avgtime(t_j)}{timeleft(t_j)} \text{ otherwise.} \end{cases}$ *, where $avgtime$ is the average time needed for the task to be completed and $timeleft$ is the time left for the task to expire. Needless to say that they should both be measured in the same units.*

The u2-urgency is related to the number of required judgments and reflects how many judgments one task needs at a given moment, relative to the requested number of judgments. We define the judgment-wise urgency $u_2(t_j)$ for a task $t_j$ as :

**Definition 5.1.6** *u2-urgency*
*We define $u_2(t_j)$-urgency as* $u_2(t_j) = \frac{judgments(t_j) - judgmentsdone(t_j)}{judgments(t_j)}$*, where $judgments(t_j)$ is the number of answers we require for a task and $judgmentsdone(t_j)$ stands for the number of answers that have already been given for the task.*

We then combine the two above definitions into the task urgency definition $u(t_j)$ :

**Definition 5.1.7** *Task Urgency u*
*We define urgency $u(t_j)$ of a task $t_j$ as $u(t_j) = u_1(t_j)u_2(t_j)$.*

**Definition 5.1.8** *Task-list Urgency*
*We define the urgency of a list $U(l)$, as the average of the urgencies $u(t_j)$ of all tasks present to the list, thus $U(l) = \sum_{j=1}^{k} \frac{u(t_j)}{k}$, where $k$ is the number of tasks in the list proposed to a participant.*

**Ranking function**

**Definition 5.1.9** *Ranking function*
*Combining the above metrics into one function we have the newly defined ranking function $C(l,p) = a_1 R(l,p) + a_2 D(l) + a_3 U(l)$, where $a_1$, $a_2$ and $a_3$ are positive constants, $a_1 + a_2 + a_3 = 1$ and $a_1 = a_2 = a_3 = \frac{1}{3}$.*

However, depending on the application, these weights could be different. Please also note that since we calculate a total ranking for each participant, these values ($a1$, $a2$, $a3$) can also parameters of his profile that we could calculate through feedback on this participation on the platform. We leave this as future work.

Now, after having presented the different definitions and our ranking function, the main problem we study in this chapter is the optimal set selection problem.

**Definition 5.1.10** *Optimal set selection*
*Given a set of tasks $T$, annotated within a taxonomy $S$, with an urgency for each task, the problem is to find a subset of tasks $l$ (of size $k$) that optimizes the objective function $C$, for a given participant $p$. We assume that the profile of the participant $p$ w.r.t. to a skill taxonomy is available.*

Please note that we construct a set of size $k$ for each single participant. We need no assumption on the arrival of the participants as the platform is running and we keep updating the propositions in real time.

## 5.2 RDUList Algorithm

Our problem is more general than the problems studied in [10, 80, 91] that are shown to be NP-hard. Our application needs a fast and real-time approach that will give in a few milliseconds a suitable list for the participant. That means we need to iterate fast on the needed elements. A heuristic approach is promising and given the properties of the problem, it is also suitable. As in [66], we use as a natural encoding schema a representation of skills of tasks and participants presented as a set of words, such that each word denotes a path in the taxonomy $S$. For example, in the taxonomy presented in Figure 5.1, we encode *basic Ruby* with 00 and *French writing* with 20 (one digit per taxonomy level).

The profiles of the tasks can be saved to the main memory and get indexed there. Since the biggest crowdsourcing platform Amazon MTurk holds about a quarter million of tasks at a given point in time we can index the task skills in memory and perform memory intensive algorithms. The above observation plus the encoding make

the resulting representation very efficient, thus we could perform the following observations :

— we can handle saving and encoding of many tasks in the memory. The overall task profiles could account for about $50$ MB for an Amazon MTurk setting [8, 82]. Digital devices such as smartphones have a lot more RAM nowadays,
— this index can run incrementally so that new tasks can be added while they arrive at the index and
— since the task profiles can fit the memory we can index them and perform memory intensive algorithms to speed-up our calculations.

RDULIST is our proposed method and does select for each participant a list of tasks that are at the same time relevant, diverse and urgent. To begin with, the algorithm indexes the tasks according to the urgency and then according to skills, thus starting to look for increasing urgency tasks first. Firstly, we sample the urgencies to k different discrete values from $0$ to $u_max$. Then, we use a list of lists for each discrete urgency value, where each item of the list holds a hashmap. Every hashmap holds the tasks that are subject to this specific urgency and the specific skill-length. This is our task indexing structure. Then we parse this indexing structure, starting with the most urgent tasks and in order to improve diversity, the algorithm will add to the list a new task so that a greater number of the participant's skills are used. If an exact skill cannot be found then a close to his skills will be used. This way we can reassure the urgency and diversity criteria set by the algorithm. In order to take into account the taxonomy hierarchy, as in Section 4.2 it will also relax the participant's skills first to find less specialized tasks for the participant or the task skills to find less specialized participants for the task. We can ensure this way that every participant gets a top-k of the best possible tasks for him. More details on the algorithm are presented in Algorithm 5.

Let's assume a participant profile $p1$={Ruby, C, English writing} and a set of tasks $t$ ={Ruby, Ruby, Ruby, Ruby, C, C, C, English reading, English reading, English writing, Pascal, Pascal}. For the sake of simplicity let us assume that the tasks have the same urgency. However, our algorithm would start with the more urgent tasks first and then relax the urgency to find less urgent tasks if suitable. Now, if we want to give $p1$ a top-3 of tasks then the RDUList algorithm will run as follows :

1. Index the tasks according to urgency and skills.
2. Take the participant's first skill Ruby and try to find tasks that match and put them in a possible list. It will bring these 4 tasks in a list.
3. Then move on with the next skill and bring in another list for the second skill.
4. In the end, it will choose to take the top-3 tasks from the different skills starting with the most urgent.
5. Finally, it will return the list : [Ruby, C, English reading] for the participant to choose from.

A relevant list algorithm would select tasks with only the first skill of the participant and maybe eventually if not more available will go on with the second skill. However, if we want to diversify we should take into account the other skills as well and then select the corresponding tasks.

**Data:** Participant $p$, $k$, tasks $T$, $skill()$ functions
**Result:** List of Tasks $\mathcal{L}$
/* we sample the urgency into k different discrete values, from 0 to $u_max$ for an index optimization */
Initialize $k * d_{max} + 1$ hashmaps, $M[0], \ldots, M[urgency][d_{max}]$;
/* each urgency list has inbuilt a skill hashmap with the task where we can look up for tasks depending on their skill length. */
**foreach** *task* $t$ **do**
  **foreach** *urgency* $i = 0$ *to* $u_{max}$ **do**
    /* take $s$ except the last $i$ levels */
    $s' \leftarrow s[0 \ldots length(s) - i]$;
    /* append at the end of the correct urgency and skill hashmap */
    $M[i][s'].append(t)$;
  **end**
**end**
**foreach** *participant* $p \in P$ **do**
  **foreach** $skill(p)$ **do**
    **foreach** *urgency* $u_{max}$ *downto* $i = 0$ **do**
      /* make a list $C$ for each skill of possible available tasks */
      $s \leftarrow skill(t)[0 \ldots length(skill(t)) - i]$;
      $C[s] \leftarrow M[i][s]$;
    **end**
  **end**
  **foreach** $j = 2$ *to* $k$ **do**
    /* put in the proposition list one task from each task. */
    /* C.size is the size of the list of the tasks that have this particular skill */
    $\mathcal{L}(p) \leftarrow C[j\%C.size()]$;
  **end**
**end**

**Algorithm 5:** RDULIST algorithm

| Algorithm | Time $O(.)$ | Space $O(.)$ |
|---|---|---|
| RANDOM METHOD | $n$ | 1 |
| RELEVANT METHOD | $d_{max}.m_s.n.k$ | $d_{max}.m_s.n.k$ |
| RDU METHOD | $d_{max}.m_s.n.k.u_{max}$ | $d_{max}.m_s.n.k.u_{max}$ |

TABLE 5.1 – Complexity assuming that $|P| = |T| = n$, $m_s$ is the maximum number of skills of a participant and $d_{max}$ the maximum depth of the skill taxonomy. k is the number of tasks in the list.

**Collisions :** With the word collision we mean that the task has been already taken by enough participants and some of them cannot perform the task because it has already gotten the corresponding number of required answers. This is not very likely to happen and can be handled just in time with a sentinel. The likelihood this will happen also depends on the mechanism of task proposition. Provided that the tasks are proposed only to a number of participants that are required to finish the task then the collision will not happen. But if we propose the tasks to k times more participants than the required answers, since the participant will select only one task from the list, then the probability a collision will happen increases but is still very little. Finally, no matter how we will finally choose to propose tasks we should make sure that when the participant selects the task is not already completed. Also, we want to avoid proposing a task that a participant has already done. This can also be handled on the fly by excluding this tasks from the participants.

**Urgency index :** Please note that the urgency index update is not mentioned but on each round or after a significant time we can choose to re-index the urgencies of the tasks. Another strategy could be to make an observation over time for the urgencies of tasks and then re-index at regular time intervals for calibration. These are out of the scope of the present work but are manageable to deal with [15].

## 5.3 Experimental Evaluation

### 5.3.1 Overall Experimental Setting

The evaluation was performed on an Apple MacBook Pro featuring an Intel i7 quad-core CPU running at 2.8 GHz, 16GB of RAM (running on 1600MHz DDR3), 1TB SSD disk and the Mac OS X Sierra operating system. The code for the synthetic experiment was written in Java and compiled with the latest Java 8 Oracle's compiler. The code for the real experiment was written in PHP and for a database we used MySQL. In order to assess our model we simulated a synthetic environment and experimented with parameters such as the different number of tasks proposed within a list, the number of tasks performed by the participants and the average time needed to perform tasks. The exact details of each setting are presented in each section.

In order to get insights on the demographics of a crowdsourcing platform we used and how participants select tasks, we first performed some real experiments on CrowdFlower to understand how they choose tasks. From the demographics, we learned the number of skills, their gender, and on which domains they are competent. From this experiment, we also learned how they choose tasks along different parameters such as :

— the first task in the list,
— the most suitable title for me,
— the most rewarding task,
— the task that takes the least time,
— the urgency of the task.

| Difficulty | Average (0-1) | Standard deviation |
|---|---|---|
| Very difficult | 0.048 | 0.027 |
| Difficult | 0.32 | 0.073 |
| Normal | 0.456 | 0.023 |
| Easy | 0.128 | 0.018 |
| Very easy | 0.048 | 0.025 |

TABLE 5.2 – Results on the difficulty of selecting tasks from a list in a Crowdsourcing Platform.

| Relevance | Average (0-1) | Standard deviation |
|---|---|---|
| Very irrelevant | 0.028 | 0.014 |
| Irrelevant | 0.068 | 0.021 |
| Neutral | 0.372 | 0.031 |
| Relevant | 0.48 | 0.057 |
| Very relevant | 0.052 | 0.040 |

TABLE 5.3 – Results on the relevance of selecting tasks from a list in a Crowdsourcing Platform.

In the real experimentation setting (Section 5.3.6) we proposed several lists to participants to select tasks from. Then from these lists, we deduced the preference of the participants for a top-k recommendation list (shortlist) as compared to a full list. We also removed the possibility of shortlist bias by comparing towards a top-k random list which proved right our intuitions.

### 5.3.2 Task Choice Demographics

We recruited a random set of 25 participants and repeated ten times (unless stated differently) in CrowdFlower asking them to answer three questions concerning the tasks they have taken before in the platform. The results of the experiments can be seen in the tables (Table 5.2, Table 5.3, Table 5.4, Table 5.5). From the tables, we can deduce that there is room for improvement on the recommendation of tasks to participants. Since CrowdFlower is not recommending tasks to participants, providing them with relevant tasks could improve the task selection procedure for participants and reduce the time needed to find a relevant task.

Table 5.2 shows how difficult participants find the current situation into selecting tasks. As we can notice, more than thirty percent of the participants find it difficult to select tasks which correlate into task selection difficulty as a real problem. There is definitely room for improvement in this area.

Table 5.3 shows how relevant participants judge the tasks that are presented to them in CrowdFlower. It seems that at least thirty-seven percent find the tasks neutral. Still, this leaves more than half of the participants unsatisfied with the relevance of the tasks they are proposed.

Table 5.4 presents the average time participants need to select and perform a task.

| Distribution of time | Average (0-1) | Standard Deviation |
|---|---|---|
| 1-5 minutes | 0.69 | 0.065 |
| 6-10 | 0.24 | 0.032 |
| 11-15 | 0.04 | 0.026 |
| 16-20 | 0.01 | 0.036 |
| 21-30 | 0.02 | 0.022 |

TABLE 5.4 – Timely results on selecting tasks from a list in a Crowdsourcing Platform.

| Parameter | Average (0-1) | Standard deviation |
|---|---|---|
| reward | 0.48 | 0.07 |
| title | 0.35 | 0.11 |
| requester | 0.033 | 0.04 |
| expiration time | 0.02 | 0.022 |
| time alloted | 0.04 | 0.05 |
| first task | 0.07 | 0.03 |

TABLE 5.5 – Distribution from CrowdFlower experiment on how participants choose tasks.

We know that participants in order to find well-paid or suitable tasks for them they rely on semi-structured forums [5]. Looking up in such forums can take time (which is also reflected on Table 5.4) that could be used otherwise for another creativity task.

For this reason, we ask 25 participants and repeat six times the experiment asking them to select between criteria on how they select tasks (as seen in Figure 5.5. It was interesting to observe that both title and reward are almost equally important for the choice the participants made. It is natural to assume from this result that participants want to optimize their revenue as long as they feel comfortable with the type and domain of the task.

That said, since the reward incentive has been sufficiently studied in other works such as [40, 14, 46] in the present, we focused rather on the relevance and the diversity that is reflected mainly by the title of each task. The urgency is reflected by the deadline and is mainly important to the requester and the platform.

### 5.3.3 Synthetic Data Setting

Our first assessment of the model was carried out with the generation of a synthetic data set that consists of a taxonomy of skills, a set of participants and a set of tasks.

As mentioned above we do create tasks with skills from a random distribution from a taxonomy of skills. We assume a budget of skills for each participant and, similarly to what we did in Section 4.3.2, give him a random path within the taxonomy of skills to generate his skills. The idea behind the budget of skills is that we have a fixed

5. http ://turkernation.com/

amount of time in our life to acquire knowledge [47]. This time we can either use it to go deeper in a domain or explore more domains of knowledge.

In the following experimental data we simulate, unless differently stated, 3,000 tasks and 3,000 participants with skills from a taxonomy of skills $S$. Each participant is given a list of available tasks to choose from. From this list he has to choose one task.

### 5.3.4 Synthetic Data Results and Discussion

We simulated the experiment for different list sizes and calculated the metrics that characterize each list generated by each algorithm. To compare with our method,RDULIST, we implemented two baselines :

— RANDOMLIST,

— RELEVANTLIST.

RANDOMLIST presents to each participant a random list of available tasks to select from. RELEVANTLIST is an extension of the algorithm PROFILEHASH 4.2 that instead of one task per participant proposes a list of available relevant tasks to the participant according to his skills. Needless to say that this method will not optimize diversity as it will only use a skill to fill in the task list.

First, we want to simulate the quality related metric and how it is affected by the list size proposed to each participant. In Figure 5.2 we can observe the relevance of the lists proposed to each participant. Also, we observe that RDULIST and RELEVANTLIST perform equally well while they both outperform the RANDOMLIST method as expected.

Figure 5.3 shows how the diversity is affected by the different size of lists along the different methods. We can understand that since RELEVANTLIST does not optimize for diversity it will start having a good random diversity after a certain number of items in the list. On the other hand our heuristic RDULIST gives an interesting result that approaches the baseline on most of the cases. This is something we were expecting since the tasks in the RANDOMLIST method are not close to the participants' skills and thus the probability of them being diverse is high. On the other hand, the Effective diversity on Figure 5.4 shows that our method outperforms the baselines all the time. The effective diversity is more important for the overall ranking. Please note that this result may also be extended in the context of content recommendation for users in commercial good platforms (such as Amazon, Ebay etc.).

In Figure 5.5 we see the effect of the list size on urgency applied to the different methods. Both of the baselines are very close in terms of the urgency of tasks which means that they will not be optimal in terms of urgency. As a result a lot of urgent tasks could starve and will not be performed in time (thus will expire before having enough judgments). However, our RDULIST will have tasks eventually move from the less urgent pile to the more urgent pile and then all tasks (provided that they need to be done) will be eventually proposed to the participants.

Figure 5.6 is showing the effect of list size on the overall score (calculated from the ranking function) of the different methods. In any case, it is clear that our method has the best score among the methods that comes from a combination of the above

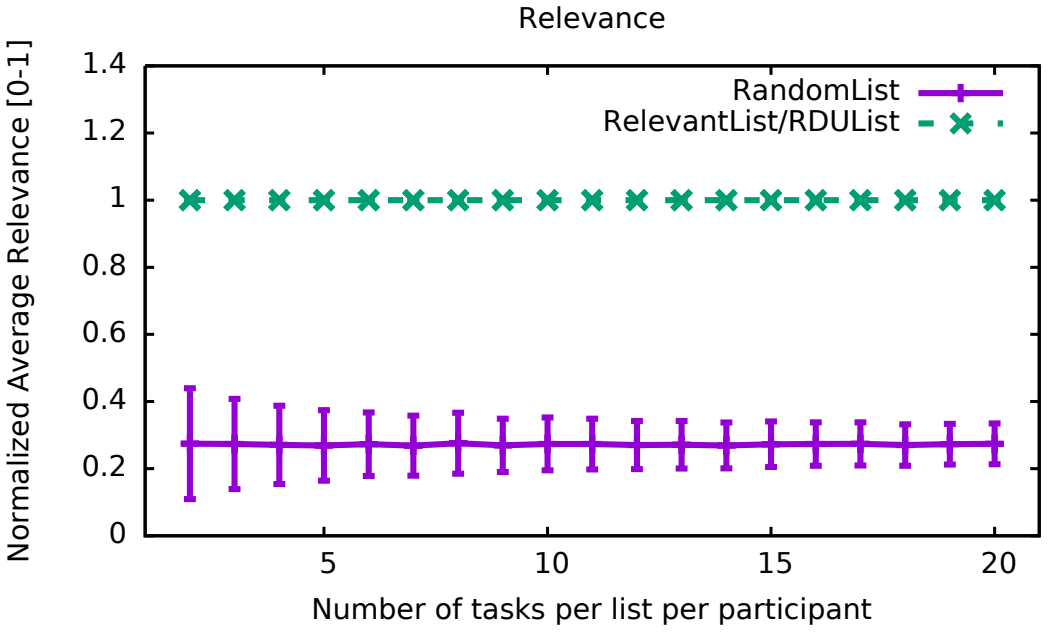



FIGURE 5.2 – Normalized Relevance for the list with respect to the size of the list proposed to the participant.

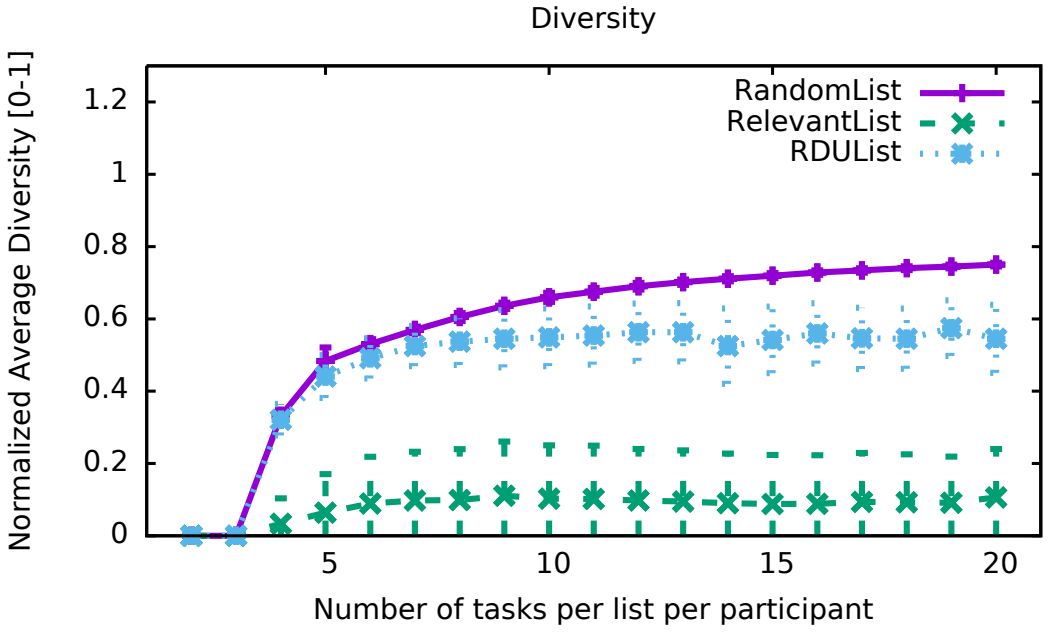



FIGURE 5.3 – Normalized Diversity for the list with respect to the size of the list proposed to the participant.

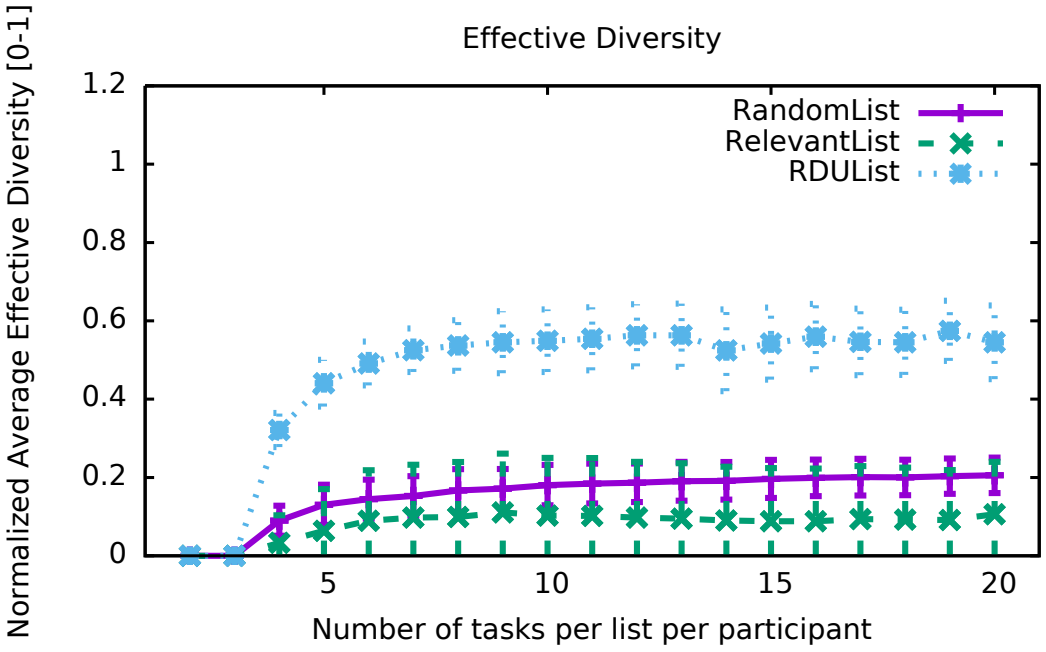



FIGURE 5.4 – Normalized Effective Diversity for the proposed list with respect to the size of the list proposed to the participant.

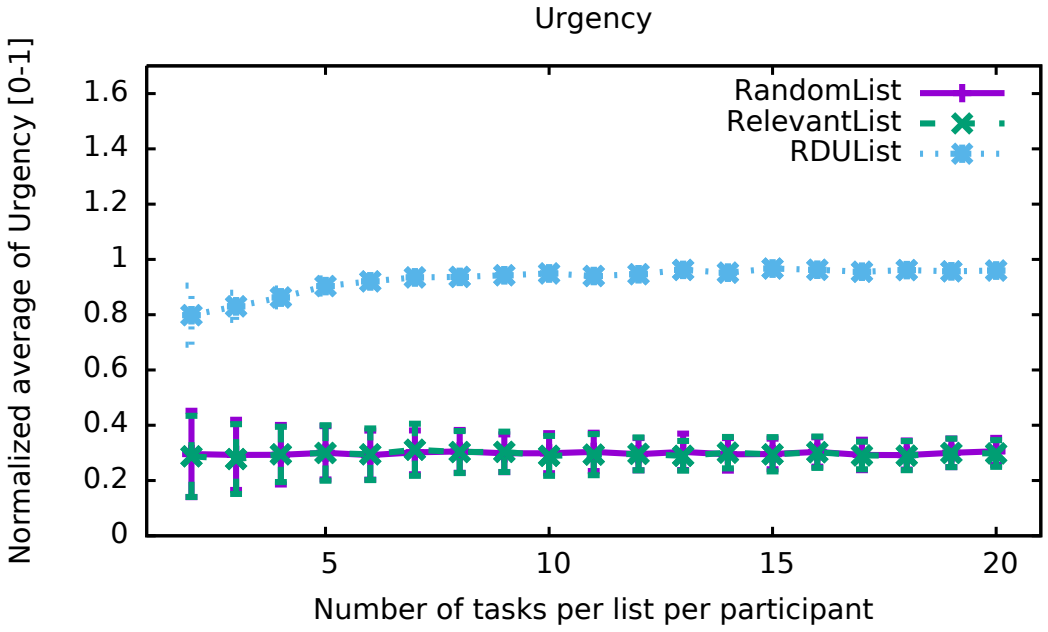



FIGURE 5.5 – Normalized Urgency for the proposed list with respect to the size of the list proposed to the participant.

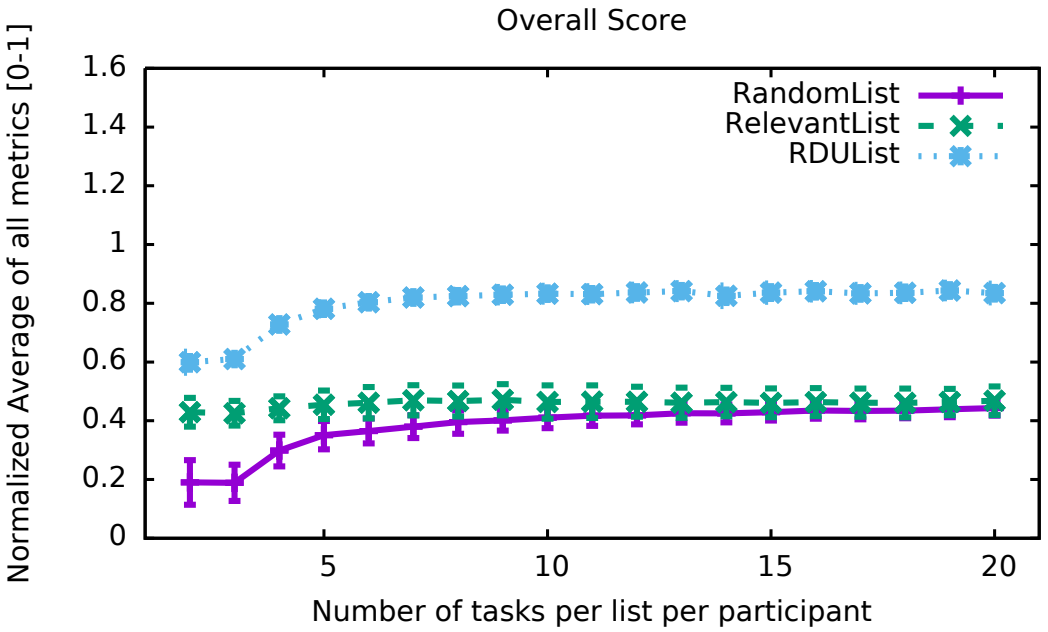



FIGURE 5.6 – Normalized Overall Metric (Relevance, Diversity, Urgency) Average with respect to the size of the list proposed to the participant.

metrics.

Figure 5.7 is showing the effect of list size to the time needed for each participant to get a full list of tasks at hand. As we can see despite the great perturbations in terms of time our method gives a very fast and scalable result combining the advantages of effective diversity and urgency of tasks. With 10 milliseconds as a maximum average time to propose tasks, we have a very scalable and straightforward method to propose tasks to participants. It is expected to have smaller times for the baselines. However, this time improvement came with a sacrifice for diversity and urgency. Instead we propose a scalable method that while give in a comparable amount of time a better solution.

### 5.3.5 Real Data Setting

In this setting, we experiment on how much the participants prefer a full list of tasks as compared to a recommended shortlist of tasks. We experimented with 100 distinct participants that come from more than 20 different countries spread all over the world, from CrowdFlower for three different list sizes ($k = 5$, $k = 10$ and $k = 15$) and asked them to follow our simple workflow. Please note that for each $k$ we recruited a different set of 100 participants to avoid any biases. The participants were redirected then to a page made with PHP and MySql to record their interaction with the system.

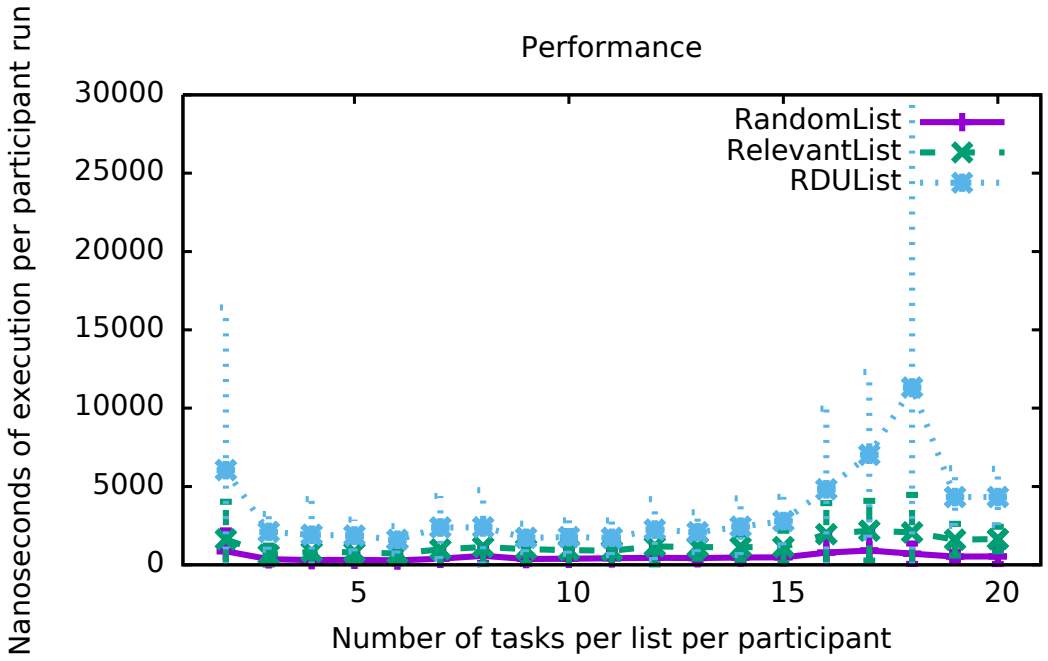



FIGURE 5.7 – Nanoseconds needed to calculate the corresponding list proposed to the participant.

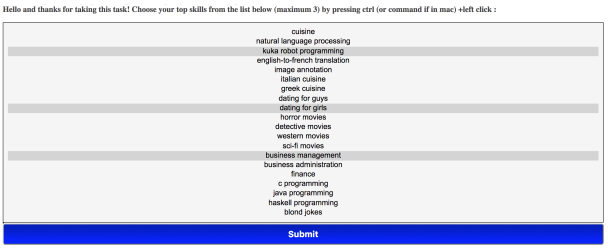



FIGURE 5.8 – First screen of the experiment : Selecting skills from list of skills.

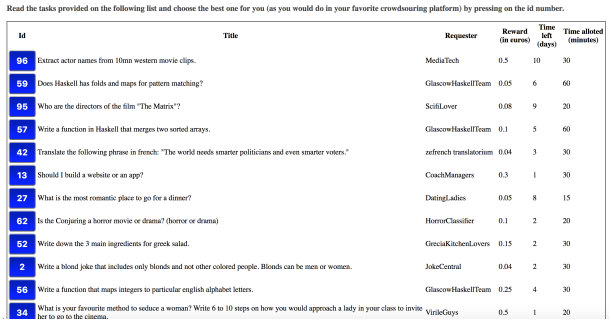
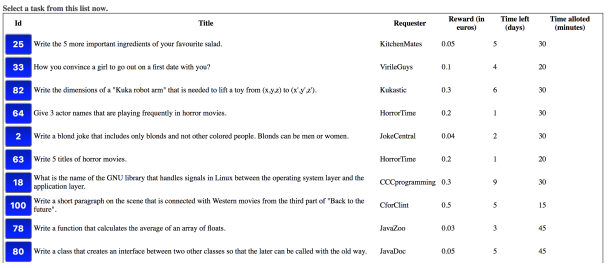



FIGURE 5.9 – Second and fifth screen of the experiment : Selecting tasks from a randomized order Full list of one hundred tasks.

More precisely, we compiled a list with 100 tasks from twenty different generic domains such as dating, cuisine, robotics, programming, films etc. that have similar characteristics with the tasks presented in Amazon MTurk and CrowdFlower. On each domain, we contributed five tasks. The tasks include title, requester name, deadlines, rewards as attributes (also mentioned in Section 5.1.2). We first asked participants to choose their skills (maximum three) from a list of these domains (twenty domains with five tasks on each domain) as seen in Figure 5.8. Then the participants chose from different lists (four lists) their tasks. We first presented them all the tasks that we compiled in a random order and they had to choose a task from that list. Then we gave them a RANDOMLIST of top-5 (and then repeated for top-10 as seen in Figure 5.10) tasks and they had to choose another task. After that point, we asked them to tell us which task they prefer among the two they selected. Next, we gave them again a different randomized full list of all the available tasks and they had to choose a task. Then we gave them a RDULIST of top-5 (and then repeated for top-10 similarly to Figure 5.10) tasks and ask them to choose another task. Finally, we asked them to select which one of the tasks from the two last steps they preferred. The procedure would help us find out whether the participants prefer the whole list of tasks or a smartly selected list of tasks (such as our RDULIST). The purpose of the RANDOMLIST was to remove the confusion bias from the choice between a short list and a very long list.

At the end of the experiment, the participants were given a golden code to insert to the CrowdFlower application and get paid. The workflow of the experiment was to


Select a task from this list now.

| Id | Title | Requester | Reward (in euros) | Time left (days) | Time alloted (minutes) |
|---|---|---|---|---|---|
| 25 | Write the 5 more important ingredients of your favourite salad. | KitchenMates | 0.05 | 5 | 30 |
| 33 | How you convince a girl to go out on a first date with you? | VirileGuys | 0.1 | 4 | 20 |
| 82 | Write the dimensions of a "Kuka robot arm" that is needed to lift a toy from (x,y,z) to (x',y',z'). | Kukastic | 0.3 | 6 | 30 |
| 64 | Give 3 actor names that are playing frequently in horror movies. | HorrorTime | 0.2 | 1 | 30 |
| 2 | Write a blond joke that includes only blonds and not other colored people. Blonds can be men or women. | JokeCentral | 0.04 | 2 | 30 |
| 63 | Write 5 titles of horror movies. | HorrorTime | 0.2 | 1 | 20 |
| 18 | What is the name of the GNU library that handles signals in Linux between the operating system layer and the application layer. | CCCprogramming | 0.3 | 9 | 30 |
| 100 | Write a short paragraph on the scene that is connected with Western movies from the third part of "Back to the future". | CforClint | 0.5 | 5 | 15 |
| 78 | Write a function that calculates the average of an array of floats. | JavaZoo | 0.03 | 3 | 45 |
| 80 | Write a class that creates an interface between two other classes so that the later can be called with the old way. | JavaDoc | 0.05 | 5 | 45 |

FIGURE 5.10 – Third and sixth screen of the experiment : Selecting tasks from a randomized order short list of either random or cleverly selected k tasks.

| Type of list presented to participants | Preference (0-1) |
|---|---|
| k=5 | |
| RANDOMLIST VS Full list | 0.34 |
| RDULIST VS Full list | 0.58 |
| k=10 | |
| RANDOMLIST VS Full list | 0.42 |
| RDULIST VS Full list | 0.52 |
| k=15 | |
| RANDOMLIST VS Full list | 0.37 |
| RDULIST VS Full list | 0.54 |

TABLE 5.6 – Ratio of preference of shortlist over full list.

go to the link, finish the steps presented there and get paid after providing the code.

### 5.3.6 Real Data Results and Discussion

Table 5.6 shows the ratio of preference for each list towards the full list. The Random short list is generated after taking the first five tasks of a random permutation of the tasks in the database. The RDULIST is generated with the same idea as the algorithm proposed in the algorithmic section (Algorithm 5) and as presented in the synthetic results (Section 5.3.4).

We can see that participants prefer the RDULIST rather than the full list or the RANDOMLIST for $k = 5$, $k = 10$ and $k = 15$. For $k = 5$ the difference is significant. We can observe a 24 percent difference comparing RANDOMLIST with RDUList and 16 percent preference of the RDULIST compared to the full list.

Then for $k = 10$ even though the list starts already to fill the screen we still have a preference of 10 percent of the RDULIST over the RANDOMLIST and 4 percent preference of the RDULIST over the full list.

For $k = 15$ the results are consistent. The participants continue to prefer the RDULIST towards the full list or the RANDOMLIST as expected.

Despite the slight decrease for $k = 10$ and $k = 15$ compared to $k = 5$ the results are

| Type of task list | Avg (seconds) | Stdev (seconds) |
|---|---|---|
| k=5 | | |
| Full list | 48 | 47 |
| RANDOMLIST | 22 | 18 |
| RDULIST | 15 | 26 |
| k=10 | | |
| Full list | 51 | 35 |
| RANDOMLIST | 25 | 22 |
| RDULIST | 16 | 16 |
| k=15 | | |
| Full list | 45 | 59 |
| RANDOMLIST | 24 | 24 |
| RDULIST | 13 | 13 |

TABLE 5.7 – Time needed to select a task from a list.

consistent. The RDULIST is the preferred list and as we increase the $k$ we can see that participants prefer slightly the smaller lists.

This result underlines the importance of having a good proposition of tasks that can result in less time to select tasks from the participants and better quality assignments. Also, a recommendation of tasks should be precise and complementary. If it is too exhaustive the participants might prefer a full list.

Table 5.7 shows how much time is needed to select a task from each different list. It is clear that concerning the time needed one needs about less than half the time to select from a shortlist instead of a full list and still prefers the selection from the RDULIST. This is an important finding since the shortlist contains only five percent of the tasks in the full list. At the same time the average time of selecting from RDULIST actually is three times less than the full list and still slightly faster than the RANDOMLIST. This is consistent for both $k = 5$, $k = 10$ and $k = 15$. We can also see that between the different $k$-size list values the results are not that different besides the fact that participants are presented with more options.

The results show the importance of having a recommendation of tasks to participants. The gain is significant both in terms of time selection but also in terms of preference from the participants. The participants clearly preferred the tasks from the RDULIST rather than the tasks in the full list. However, the selection of the tasks in the shortlist is important. Taking into account the skills of the participants increased their preference of the shortlists over the full list. This is promising and shows that we can both assign a list of tasks and respect a choice for the participants. The results in this section complement the results in the synthetic section and support the importance of giving the participants with a choice over the tasks they have to select from.

## 5.4 Chapter Conclusion

In the present chapter, we showed how a realistic simulation of better quality lists could affect the participants' satisfaction and improve the current proposition of tasks. Simulating the skills of participants, we experimented on different lists and observed how this would affect participants' experience through the metrics we defined. The important contribution of our work is the real-time, relevance-diversity-urgency aware model that takes into account several attributes to improve the participant and requester experience in a holistic manner. Finally, after having presented our contributions we will conclude our contributions with a small discussion and state the future work in the final chapter.

# Chapter 6

# Conclusion and Future Work

In the present thesis, we identified crowdsourcing as a challenging, exciting and overwhelming domain. We have defined which applications are considered crowdsourcing and we confirmed the importance of crowdsourcing as a successful tool for several domains. Then we presented the related work and what challenges need to be addressed by current crowdsourcing systems. In the following discussion section, we will summarize the challenges, our contributions and the results that were provided along with what could be improved. Then we conclude with a section discussing some future work that could be carried out to extend our contributions.

## 6.1 Discussion

On Chapter 4 we have shown how we can assign optimally a set of tasks to a set of participants. We also proposed ways to optimize the performance of this assignment with a minimal sacrifice in the quality, as seen on the synthetic datasets (4.3.3). We also demonstrated and experimented on the extended model where tasks require a minimum expertise and showed the impact of the methods proposed.

In order to enhance our contribution, we could also have addressed some other issues such as :

— study how one can obtain taxonomies of skills and

— how to update participant profiles after the requester's feedback.

Later on, the next Chapter 5 given a set of tasks, a taxonomy of skills and an unknown arrival order of participants for a crowdsourcing platform we showed the importance of $relevance$, $diversity$ and $urgency$ of individual tasks on the task skills proposed.

More precisely, we have shown how an overall score of relevance, diversity and urgency is optimized with our methods and how we can improve the total satisfaction and time needed to select short lists of tasks for participants. However, we did not take into account some other parameters :

— we did not include the payment into the overall score and

— we did not take into account the distribution of arriving participants' skills.

To sum up, in our thesis we have shown the following :
— the importance of skill precision and substitution in the context of quality and
— the importance of skill diversity and urgency for the participant satisfaction and the platform guarantees

In the next section, let us propose some future and visionary work that could be carried out in crowdsourcing.

## 6.2 Future Work

Even though we have shown that hierarchical skills are crucial to improve the quality of crowdsourced answers there are more directions one can point for the overall improvement of crowdsourcing. Here, we will discuss some more natural ways that could be pursued to further improve crowdsourcing :
— take into account richer profiles,
— explore different participant incentives,
— explore task reward models,
— model multiple skill tasks and assignment distances for them,
— model continuous crowdsourcing and
— model interactions on complex tasks crowdsourcing.

### 6.2.1 Richer profiles

For instance, we could consider richer profiles for participants. For instance, except for skills, participants also have preferences. The preferences could be from either a partial order of his skills, like his preferred skill or a preferred type of task. Personality tests exist for participant profiling that could show participant traits. As mentioned in [44] there are different types of crowdsourcing microtasks. There are multiple choice questions, paragraph creation, translation, annotation, validation, voting, data completion etc. These different types of skills could be also taken into account as preferences during the task assignment process.

### 6.2.2 Participant Incentives Exploration

Even though hierarchical skills are very important for the quality of the platform, giving necessary incentives for the participant is equally crucial to keep skilled participants in the platform. To this extent, we could explore better and fairer task reward models and different participant incentives that go beyond monetary rewards.

#### Task Reward Models Exploration

We have proposed a taxonomy of skills to improve the search for more suitable participants and take advantage of very specialized niches of participants, such as experts. This approach could be extended to the cost model. A very specialized task that is assigned to a participant with less expertise on the task, thus greater distance

according to our definitions in Section 3.4 could be paid less than an expert participant. The relevance of the task and the participant could also be monetized. The distance metric that we introduced could be used as a cost coefficient for fair payments.

#### Beyond Micropayments

Apart from the cost models and the monetary incentives we could explore how different incentives promote different goals. For instance, Wikipedia does not pay the participants. However, participants get possibly promoted and have the fulfillment of participating to the creation of an open encyclopedia for the community. Also in very active community platforms such as Quora or StackOverflow participants do not get paid directly, but still contribute either for fun or to get recruited or for their reputation in the online community. Unifying incentives for different goals depending on the need of the platform could be another interesting area for research.

### 6.2.3 Multiple Round Assignment and Continuous Crowdsourcing

Instead of having a single assignment of multiple participant assignments to one task we could envision different ways. Already, in well-known platforms such as Wikipedia, Quora, StackOverflow etc. one can update his answer. This is done in a forum based way but it could be more structured and provide with a continuous improvement over the quality of crowdsourced answers.

### 6.2.4 Beyond Single-skill Crowdsourcing

We have explored single skill tasks with specialized skills that come from a taxonomy. A natural extension would be to explore more complex tasks. An exploration of multiple-skill tasks assigned to teams of participants and complex-task crowdsourcing with different plans of execution.

#### Multiple-skill Tasks and Team formation

A natural extension would be to explore tasks that require a vector of different skills from the taxonomy of skills. Then there would be a challenge to match these tasks to one or more participants that have the required skills to perform the task. There is some related work on teams of participants that we mentioned in Section 2.5.5, but the problem of best team formation has not been studied enough. On this track, richer profiles (Section 6.2.1) could be used to make more diverse and skilled teams. However, it would also be interesting to make further research studying the impact on quality of the inclusion of the taxonomy, interactions with the participants and even possible reformatting of the tasks that could lead to the next step of complex-task crowdsourcing and workflows.

**Complex-task crowdsourcing**

In addition, complex tasks that consist of more than one interconnected microtasks have not been sufficiently studied [11]. There is a need for a declarative way to obtain a trade-off for total tasks. We can imagine several different ways that could give a generic result (that will be probably of different quality). One possible direction to achieve this could be to model complex-tasks with a use of a graph and more precisely a **DAG** to permit different possible paths of execution. Then we could include characteristics such as skills, answers with quality guarantees, the safety of optimal execution and total cost of work. To the best of my knowledge, there is no unified and declarative manner to do so.

## 6.3 Vision

The possibilities are endless. I envision the ideal crowdsourcing as a platform where all the above would be embedded into a unified platform. APIs connected with this platform could provide the operating system, mobile and web applications with several crowd capabilities. A simple automatic system with a unified and simple language could find and recruit individuals or teams of experts to solve problems on the fly. This could revolutionize medicine and real-time complex decision-making. It could also help to make technology reach less developed countries. Moreover, it could speed up teamwork and optimize costs through reducing redundant work.

Ideally, we would be able to set a goal to a natural text and then the system will make the necessary translations to be able to disambiguate it, give an estimate of the total cost and time of execution, divide it in a workflow of simpler tasks, find suitable participants for each task and take intermediate decisions in case of failure. Even further, this system could be used as a hybrid human-machine interface for any possible online service, starting from our operating systems, hybrid chatbots, instant translation and much more services.

# List of Publications

## International Posters

[1] Panagiotis Mavridis, David Gross-Amblard, Zoltan Miklos. Skill-Aware Task Assignment in Crowdsourcing Applications. International Symposium on Web AlGorithms (iSWAG), Jun 2015, Deauville, France. Proceedings of the 1st International Symposium on Web AlGorithms, 2015.

## International Conference Papers

[2] Panagiotis Mavridis, David Gross-Amblard, Zoltan Miklos. Using Hierarchical Skills for Optimized Task Assignment in Knowledge Intensive Crowdsourcing. 25th international World Wide Web Conference (WWW), April 11-15 2016, Montreal, Canada.

## Journals (Under Review)

Submitted on Fast-Track of ACM TWEB and pending feedback :

[3] Panagiotis Mavridis, David Gross-Amblard, Zoltan Miklos. Using Hierarchical Skills for Optimized Task Assignment in Knowledge Intensive Crowdsourcing. ACM Journal on Transactions on the WEB.

## Submission Pending

[4] Panagiotis Mavridis, Gianluca Demartini, David Gross-Amblard, Zoltan Miklos. It's Up to You ! Ranking Tasks for Relevance, Diversity and Urgency in Crowdsourcing Platforms

# References


[1] *Crowdflower*. www.crowdflower.com.

[2] *Crowdpolicy*. www.crowdpolicy.com.

[3] *DefinedCrowd*. https://www.definedcrowd.com.

[4] *FoldIt, solve puzzles for science*. http://fold.it/portal/.

[5] *FouleFactory*. www.foulefactory.com.

[6] *Projet sCOOP, Stanford*. http://www-cs-students.stanford.edu/~adityagp/scoop.html.

[7] *Sauvages de ma rue*. http://sauvagesdemarue.mnhn.fr.

[8] *Overview of Mechanical Turk*, 2017. http://docs.aws.amazon.com/AWSMechTurk/latest/RequesterUI/OverviewofMturk.html.

[9] ACEMOGLU, D., MOSTAGIR, M., AND OZDAGLAR, A. Managing innovation in a crowd. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation* (2015), EC '15.

[10] AGRAWAL, R., GOLLAPUDI, S., HALVERSON, A., AND IEONG, S. Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining* (New York, NY, USA, 2009), WSDM '09, ACM, pp. 5–14.

[11] AHMAD, S., BATTLE, A., MALKANI, Z., AND KAMVAR, S. The jabberwocky programming environment for structured social computing. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2011), UIST '11, ACM, pp. 53–64.

[12] AMER-YAHIA, S., GAUSSIER, E., LEROY, V., PILOURDAULT, J., BORROMEO, R. M., AND TOYAMA, M. Task composition in crowdsourcing. In *Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on* (2016), IEEE, pp. 194–203.

[13] ANAGNOSTOPOULOS, A., BECCHETTI, L., CASTILLO, C., GIONIS, A., AND LEONARDI, S. Online team formation in social networks. In *Proceedings of the 21st International Conference on World Wide Web* (New York, NY, USA, 2012), WWW '12, ACM, pp. 839–848.

[14] BAROWY, D. W., CURTSINGER, C., BERGER, E. D., AND MCGREGOR, A. Automan : A platform for integrating human-based and digital computation. In

*Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications* (New York, NY, USA, 2012), OOPSLA '12, ACM, pp. 639–654.

[15] BARSKY, M., THOMO, A., TOTH, Z., AND ZUZARTE, C. Online update of b-trees. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management* (New York, NY, USA, 2010), CIKM '10, ACM, pp. 149–158.

[16] BENDER, M. A., FARACH-COLTON, M., PEMMASANI, G., SKIENA, S., AND SUMAZIN, P. Lowest common ancestors in trees and directed acyclic graphs. *Journal of Algorithms 57*, 2 (2005), 75–94.

[17] BOZZON, A., BRAMBILLA, M., CERI, S., AND MAURI, A. Reactive crowdsourcing. In *Proceedings of the 22Nd International Conference on World Wide Web* (Republic and Canton of Geneva, Switzerland, 2013), WWW '13, International World Wide Web Conferences Steering Committee, pp. 153–164.

[18] BOZZON, A., BRAMBILLA, M., CERI, S., SILVESTRI, M., AND VESCI, G. Choosing the right crowd : Expert finding in social networks. In *Proceedings of the 16th International Conference on Extending Database Technology* (New York, NY, USA, 2013), EDBT '13, ACM, pp. 637–648.

[19] BRABHAM, D. C. *Crowdsourcing*. The MIT Press, 2013.

[20] BRADLEY, K., RAFTER, R., AND SMYTH, B. Case-based user profiling for content personalization. In *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-based Systems* (2000), Springer-Verlag, pp. 62–72.

[21] BUTLER, D. *Crowdsourcing goes mainstream in typhoon response*. Nature, November 2013. `http://www.nature.com/news/crowdsourcing-goes-mainstream-in-typhoon-response-1.14186`.

[22] CAMPION, M. A., FINK, A. A., RUGGEBERG, B. J., CARR, L., PHILLIPS, G. M., AND ODMAN, R. B. Doing competencies well : Best practices in competency modeling. *Personnel Psychology 64*, 1 (2011), 225–262.

[23] CAO, C. C., SHE, J., TONG, Y., AND CHEN, L. Whom to ask ? : Jury selection for decision making tasks on micro-blog services. *PVLDB 5*, 11 (July 2012), 1495–1506.

[24] CARTERETTE, B. An analysis of np-completeness in novelty and diversity ranking. *Information Retrieval 14*, 1 (2011), 89–106.

[25] CATALLO, I., CICERI, E., FRATERNALI, P., MARTINENGHI, D., AND TAGLIASACCHI, M. Top-k diversity queries over bounded regions. *ACM Transactions on Database Systems 38*, 2 (July 2013), 10 :1–10 :44.

[26] COOPER, S., KHATIB, F., TREUILLE, A., BARBERO, J., LEE, J., BEENEN, M., LEAVER-FAY, A., BAKER, D., POPOVIC, Z., AND PLAYERS, F. Predicting protein structures with a multiplayer online game. *Nature 466*, 7307 (08 2010), 756–760.

[27] DAWID, A. P., AND SKENE, A. M. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics) 28*, 1 (1979), 20–28.

[28] DEMARTINI, G., DIFALLAH, D. E., AND CUDRÉ-MAUROUX, P. Zencrowd : leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012* (2012), pp. 469–478.

[29] DESMARAIS, M. C., AND D BAKER, R. S. A review of recent advances in learner and skill modeling in intelligent learning environments. *User Modeling and User-Adapted Interaction 22*, 1-2 (2012), 9–38.

[30] DEUTCH, D., GREENSHPAN, O., KOSTENKO, B., AND MILO, T. Declarative platform for data sourcing games. In *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012* (2012), pp. 779–788.

[31] DIFALLAH, D. E., DEMARTINI, G., AND CUDRÉ-MAUROUX, P. Pick-a-crowd : Tell me what you like, and i'll tell you what to do. In *Proceedings of the 22Nd International Conference on World Wide Web* (Republic and Canton of Geneva, Switzerland, 2013), WWW '13, International World Wide Web Conferences Steering Committee, pp. 367–374.

[32] DIFALLAH, D. E., DEMARTINI, G., AND CUDRÉ-MAUROUX, P. Scheduling human intelligence tasks in multi-tenant crowd-powered systems. In *Proceedings of the 25th International Conference on World Wide Web* (Republic and Canton of Geneva, Switzerland, 2016), WWW '16, International World Wide Web Conferences Steering Committee, pp. 855–865.

[33] DOAN, A., FRANKLIN, M. J., KOSSMANN, D., AND KRASKA, T. Crowdsourcing applications and platforms : A data management perspective. vol. 4, VLDB Endowment, pp. 1508–1509.

[34] DOAN, A., RAMAKRISHNAN, R., AND HALEVY, A. Y. Crowdsourcing systems on the world-wide web. *Communnications ACM 54*, 4 (Apr. 2011), 86–96.

[35] DROSOU, M., AND PITOURA, E. Dynamic diversification of continuous data. In *Proceedings of the 15th International Conference on Extending Database Technology* (New York, NY, USA, 2012), EDBT '12, ACM, pp. 216–227.

[36] ELLIOTT, S. *Ford Turns to the 'Crowd' for New Fiesta Ads*. New York Times, February 2013. `https://mediadecoder.blogs.nytimes.com/2013/02/19/ford-turns-to-the-crowd-for-new-fiesta-ads/?mcubz=1`.

[37] ENRICH, M., BRAUNHOFER, M., AND RICCI, F. Cold-start management with cross-domain collaborative filtering and tags. In *E-Commerce and Web Technologies* (2013), C. Huemer and P. Lops, Eds., vol. 152 of *Lecture Notes in Business Information Processing*, Springer Berlin Heidelberg, pp. 101–112.

[38] ESTELLÉS-AROLAS, E., AND DE GUEVARA, F. G.-L. Towards an integrated crowdsourcing definition. *Journal of Information Science 38*, 2 (2012), 189–200.

[39] FAN, J., LI, G., OOI, B. C., TAN, K.-L., AND FENG, J. icrowd : An adaptive crowdsourcing framework. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (2015), SIGMOD '15, pp. 1015–1030.

[40] FARADANI, S., HARTMANN, B., AND IPEIROTIS, P. G. What's the right price ? pricing tasks for finishing on time. *Human computation 11* (2011), 11.

[41] FENG, A., FRANKLIN, M. J., KOSSMANN, D., KRASKA, T., MADDEN, S., RAMESH, S., WANG, A., AND XIN, R. Crowddb : Query processing with the VLDB crowd. vol. 4, VLDB Endowment, pp. 1387–1390.

[42] FRANKLIN, M. J., KOSSMANN, D., KRASKA, T., RAMESH, S., AND XIN, R. Crowddb : Answering queries with crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 2011), SIGMOD '11, ACM, pp. 61–72.

[43] FRATERNALI, P., MARTINENGHI, D., AND TAGLIASACCHI, M. Top-k bounded diversification. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 2012), SIGMOD '12, ACM, pp. 421–432.

[44] GADIRAJU, U., KAWASE, R., AND DIETZE, S. A taxonomy of microtasks on the web. In *Proceedings of the 25th ACM conference on Hypertext and social media* (2014), ACM, pp. 218–223.

[45] GADIRAJU, U., KAWASE, R., DIETZE, S., AND DEMARTINI, G. Understanding malicious behavior in crowdsourcing platforms : The case of online surveys. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (New York, NY, USA, 2015), CHI '15, ACM, pp. 1631–1640.

[46] GAO, Y., AND PARAMESWARAN, A. Finish them ! : Pricing algorithms for human computation. *PVLDB 7*, 14 (Oct. 2014), 1965–1976.

[47] GLADWELL, MALCOLM, . *Outliers : the story of success*. First edition. New York : Little, Brown and Co., 2008., 2008.

[48] HAAS, D., ANSEL, J., GU, L., AND MARCUS, A. Argonaut : Macrotask crowdsourcing for complex data processing. *PVLDB 8*, 12 (2015), 1642–1653.

[49] HITLIN, P. *Research in the Crowdsourcing Age, a Case Study*. Pew Research Center, 2016. `http://www.pewinternet.org/2016/07/11/research-in-the-crowdsourcing-age-a-case-study/`.

[50] HORAWALAVITHANA, Y. S., AND RANASINGHE, D. N. An efficient incremental indexing mechanism for extracting top-k representative queries over continuous data-streams. In *Proceedings of the 14th International Workshop on Adaptive and Reflective Middleware* (New York, NY, USA, 2015), ARM 2015, ACM, pp. 8 :1–8 :3.

[51] HOWE, J. *The Rise of Crowdsourcing*. Wired, 2006. `https://www.wired.com/2006/06/crowds/`.

[52] HOWE, J. *Crowdsourcing : Why the Power of the Crowd Is Driving the Future of Business*, 1 ed. Crown Publishing Group, New York, NY, USA, 2008.

[53] KARGER, D. R., OH, S., AND SHAH, D. Budget-optimal task allocation for reliable crowdsourcing systems. *Operations Research 62*, 1 (February 2014), 1–24.

[54] KHATIB, F., DIMAIO, F., COOPER, S., KAZMIERCZYK, M., GILSKI, M., KRZYWDA, S., ZABRANSKA, H., PICHOVA, I., THOMPSON, J., POPOVIĆ, Z., JASKOLSKI, M., AND BAKER, D. Crystal structure of a monomeric retroviral protease solved by protein folding game players. *Nature Structural and Molecular Biology 18*, 10 (10 2011), 1175–1177.

[55] KITTUR, A., SMUS, B., KHAMKAR, S., AND KRAUT, R. E. Crowdforge : crowdsourcing complex work. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, Santa Barbara, CA, USA, October 16-19, 2011* (2011), J. S. Pierce, M. Agrawala, and S. R. Klemmer, Eds., ACM, pp. 43–52.

[56] KUHN, H. W. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly 2* (1955), 83–97.

[57] KULKARNI, A., CAN, M., AND HARTMANN, B. Collaboratively crowdsourcing workflows with turkomatic. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work* (New York, NY, USA, 2012), CSCW '12, ACM, pp. 1003–1012.

[58] LAW, E., AND VON AHN, L. *Human Computation*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2011.

[59] LEE, J., AND HWANG, S.-W. Toward efficient multidimensional subspace skyline computation. *VLDB Journal 23*, 1 (Feb. 2014), 129–145.

[60] LITTLE, G., CHILTON, L. B., GOLDMAN, M., AND MILLER, R. C. Turkit : Human computation algorithms on mechanical turk. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2010), UIST '10, ACM, pp. 57–66.

[61] LIU, X., LU, M., OOI, B. C., SHEN, Y., WU, S., AND ZHANG, M. Cdas : a crowdsourcing data analytics system. *PVLDB 5*, 10 (2012), 1040–1051.

[62] MAARRY, K., BALKE, W.-T., CHO, H., HWANG, S.-W., AND BABA, Y. Skill ontology-based model for quality assurance in crowdsourcing. In *Database Systems for Advanced Applications*, Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2014, pp. 376–387.

[63] MAGNANI, M., ASSENT, I., AND MORTENSEN, M. L. Taking the big picture : Representative skylines based on significance and diversity. *VLDB Journal 23*, 5 (Oct. 2014), 795–815.

[64] MARKOFF, J. *In a Video Game, Tackling the Complexities of Protein Folding*. New York Times, August 2010. `http://www.nytimes.com/2010/08/05/science/05protein.html`.

[65] MAVRIDIS, P., GROSS-AMBLARD, D., AND MIKLÓS, Z. Skill-aware task assignment in crowdsourcing applications. In *International Symposium on Web AlGorithms* (2015).

[66] MAVRIDIS, P., GROSS-AMBLARD, D., AND MIKLÓS, Z. Using hierarchical skills for optimized task assignment in knowledge-intensive crowdsourcing. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016* (2016), pp. 843–853.

[67] MIDDLETON, S. E., SHADBOLT, N. R., AND DE ROURE, D. C. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS) 22*, 1 (2004), 54–88.

[68] MINDER, P., AND BERNSTEIN, A. Crowdlang : A programming language for the systematic exploration of human computation systems. In *Social Informatics*, K. Aberer, A. Flache, W. Jager, L. Liu, J. Tang, and C. Guéret, Eds., vol. 7710 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 124–137.

[69] MO, L., CHENG, R., KAO, B., YANG, X. S., REN, C., LEI, S., CHEUNG, D. W., AND LO, E. Optimizing plurality for human intelligence tasks. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013* (2013), pp. 1929–1938.

[70] MORISHIMA, A., SHINAGAWA, N., MITSUISHI, T., AOKI, H., AND FUKUSUMI, S. Cylog/crowd4u : A declarative platform for complex data-centric crowdsourcing. *PVLDB 5*, 12 (Aug. 2012), 1918–1921.

[71] OOSTERMAN, J., BOZZON, A., HOUBEN, G.-J., NOTTAMKANDATH, A., DIJKSHOORN, C., AROYO, L., LEYSSEN, M. H., AND TRAUB, M. C. Crowd vs. experts : Nichesourcing for knowledge intensive tasks in cultural heritage. In *Proceedings of the 23rd International Conference on World Wide Web* (New York, NY, USA, 2014), WWW '14 Companion, ACM, pp. 567–568.

[72] OOSTERMAN, J., NOTTAMKANDATH, A., DIJKSHOORN, C., BOZZON, A., HOUBEN, G.-J., AND AROYO, L. Crowdsourcing knowledge-intensive tasks in cultural heritage. In *Proceedings of the 2014 ACM Conference on Web Science* (New York, NY, USA, 2014), WebSci '14, ACM, pp. 267–268.

[73] PARAMESWARAN, A. G., PARK, H., GARCIA-MOLINA, H., POLYZOTIS, N., AND WIDOM, J. Deco : Declarative crowdsourcing. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management* (New York, NY, USA, 2012), CIKM '12, ACM, pp. 1203–1212.

[74] PARK, H., PANG, R., PARAMESWARAN, A. G., GARCIA-MOLINA, H., POLYZOTIS, N., AND WIDOM, J. An overview of the deco system : data model and query language ; query processing and optimization. *SIGMOD Record 41*, 4 (2012), 22–27.

[75] PILOURDAULT, J., AMER-YAHIA, S., LEE, D., AND ROY, S. B. Motivation-Aware Task Assignment in Crowdsourcing. In *Proceedings of the 16th International Conference on Extending Database Technology* (Mar. 2017), EDBT '17.

[76] POO, D., CHNG, B., AND GOH, J.-M. A hybrid approach for user profiling. In *Proceedings of the 36th Annual Hawaii International Conference on System*

*Sciences (HICSS'03) - Track 4 - Volume 4* (Washington, DC, USA, 2003), HICSS '03, IEEE Computer Society, pp. 103.2–.

[77] RAHMAN, H., THIRUMURUGANATHAN, S., ROY, S. B., AMER-YAHIA, S., AND DAS, G. Worker skill estimation in team-based tasks. *PVLDB 8*, 11 (2015), 1142–1153.

[78] RESNIK, P. Semantic similarity in a taxonomy : An information-based measure and its application to problems of ambiguity in natural language. *J. Artif. Intell. Res. (JAIR) 11* (1999), 95–130.

[79] ROY, S. B., LYKOURENTZOU, I., THIRUMURUGANATHAN, S., AMER-YAHIA, S., AND DAS, G. Task assignment optimization in knowledge-intensive crowdsourcing. *VLDB Journal 24*, 4 (2015), 467–491.

[80] SANTOS, R. L., MACDONALD, C., AND OUNIS, I. Exploiting query reformulations for web search result diversification. In *Proceedings of the 19th International Conference on World Wide Web* (New York, NY, USA, 2010), WWW '10, ACM, pp. 881–890.

[81] STEEL, E. *Newcastle Brown Ale Calls for Other Brands to Join a Sly Super Bowl Ad Campaign*. New York Times, January 2015.

[82] TAMIR, D. *50000 Worldwide Mechanical Turk Workers*. Techlist, September 2014. `http://techlist.com/mturk/global-mturk-worker-map.php`.

[83] TRANQUILLINI, S., DANIEL, F., KUCHERBAEV, P., AND CASATI, F. Modeling, enacting, and integrating custom crowdsourcing processes. *ACM Transactions on the Web 9*, 2 (May 2015), 7 :1–7 :43.

[84] TREBAY, G. *Keeping T-Shirts in the Moment*. New York Times, July 2005. `http://www.nytimes.com/2005/07/21/fashion/thursdaystyles/keeping-tshirts-in-the-moment.html`.

[85] VALKANAS, G., PAPADOPOULOS, A. N., AND GUNOPULOS, D. Skydiver : A framework for skyline diversification. In *Proceedings of the 16th International Conference on Extending Database Technology* (New York, NY, USA, 2013), EDBT '13, ACM, pp. 406–417.

[86] VICTOR, P., CORNELIS, C., TEREDESAI, A. M., AND DE COCK, M. Whom should i trust ? : The impact of key figures on cold start recommendations. In *Proceedings of the 2008 ACM Symposium on Applied Computing* (New York, NY, USA, 2008), SAC '08, ACM, pp. 2014–2018.

[87] VUURENS, J., AND DE VRIES, A. Obtaining High-Quality Relevance Judgments Using Crowdsourcing. *IEEE Internet Computing 16*, 5 (2012), 20–27.

[88] WANG, D., ABDELZAHER, T., KAPLAN, L., AND AGGARWAL, C. C. Recursive fact-finding : A streaming approach to truth estimation in crowdsourcing applications. In *ICDCS'2013*.

[89] WANG, X., DOU, Z., SAKAI, T., AND WEN, J.-R. Evaluating search result diversity using intent hierarchies. In *Proceedings of the 39th International ACM SIGIR*

*Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2016), SIGIR '16, ACM, pp. 415–424.

[90] WICKS, P., VAUGHAN, T. E., MASSAGLI, M. P., AND HEYWOOD, J. Accelerated clinical discovery using self-reported patient data collected online and a patient-matching algorithm. *Nature Biotech 29*, 5 (05 2011), 411–414.

[91] WU, T., CHEN, L., HUI, P., ZHANG, C. J., AND LI, W. Hear the whole story : Towards the diversity of opinion in crowdsourcing markets. *PVLDB 8*, 5 (2015), 485–496.

[92] ZHANG, J., TANG, J., AND LI, J.-Z. Expert finding in a social network. In *DASFAA* (2007), K. Ramamohanarao, P. R. Krishna, M. K. Mohania, and E. Nantajeewarawat, Eds., vol. 4443 of *Lecture Notes in Computer Science*, Springer, pp. 1066–1069.

[93] ZHANG, W., AND WANG, J. A collective bayesian poisson factorization model for cold-start local event recommendation. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2015), KDD '15, ACM, pp. 1455–1464.

[94] ZHAO, Z., CHENG, J., WEI, F., ZHOU, M., NG, W., AND WU, Y. Socialtransfer : Transferring social knowledge for cold-start cowdsourcing. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management* (New York, NY, USA, 2014), CIKM '14, ACM, pp. 779–788.

[95] ZHENG, L., AND CHEN, L. Mutual benefit aware task assignment in a bipartite labor market. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)* (2016), IEEE, pp. 73–84.

[96] ZHENG, Y., CHENG, R., MANIU, S., AND MO, L. On optimality of jury selection in crowdsourcing. In *Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, Brussels, Belgium, March 23-27, 2015.* (2015), pp. 193–204.

# Table of Figures

# List of Tables

# Abstract

A large number of participative applications rely on a crowd to acquire and process data. These participative applications are widely known as crowdsourcing platforms, where amateur enthusiasts are involved in real scientific or commercial projects that requesters have posted online. Most well-known commercial crowdsourcing platforms are Amazon MTurk and Crowdflower. Participants there, select and perform tasks, called microtasks and accept a micropayment in return. Common challenges for such platforms are related to the quality of the required answers, the expertise of the involved crowd, the ease of finding tasks and the respect of tasks' deadlines. Related work focuses on modeling skills as keywords to improve quality while in this work we formalize skills using a hierarchical structure, that can help substituting tasks with similar skills and take advantage of the whole workforce. With extensive synthetic and real datasets, we show a significant improvement in quality when using a hierarchical structure of skills instead of pure keywords. We also extend our work to study the impact of a participant's choice given a list of tasks. While our previous solution focused on improving an overall one-to-one matching for tasks and participants, we also examine how participants can choose from a ranked list of tasks. Selecting from an enormous list of tasks can be challenging, time-consuming and affects the quality of answers to crowdsourcing platforms. Existing related work concerning crowdsourcing uses neither a taxonomy nor ranking methods to assist participants. We propose a new model that provides the participant with a short list of tasks. This short list takes into account the diversity of the participant's skills and the task deadlines as well. Our extensive synthetic and real experiments show that we can meet deadlines, get high-quality answers and keep the interest of participants high while giving them a choice of well-selected tasks.

**Keywords :** crowdsourcing, resource allocation, taxonomy of skills, task to participant matching, task allocation, task ranking, participant modeling, skill modeling, task mapping, user recommendation, diversification, task urgency, task deadlines.

# Résumé français

De nombreuses applications participatives, commerciales et académiques s'appuient sur des volontaires ("la foule") pour acquérir, désambiguiser et nettoyer des données. Ces applications participatives sont largement connues sous le nom de plates-formes de crowdsourcing où des amateurs peuvent participer à de véritables projets scientifiques ou commerciaux. Ainsi, des demandeurs sous-traitent des tâches en les proposant sur des plates-formes telles qu'Amazon MTurk ou Crowdflower. Puis, des participants en ligne sélectionnent et exécutent ces tâches, appelés micro-tasks, acceptant un micropaiement en retour. Ces plates-formes sont confrontées à des défis tels qu'assurer la qualité des réponses acquises, aider les participants à trouver des tâches pertinentes et intéressantes, tirer parti des compétences expertes parmi la foule, respecter les délais des tâches et promouvoir les participants qui accomplissent le plus de tâches. Cependant, la plupart des plates-formes ne modélisent pas explicitement les compétences des participants, ou se basent simplement sur une description en terme de mots-clés. Dans ce travail, nous proposons de formaliser les compétences des participants au moyen d'une structure hiérarchique, une taxonomie, qui permet naturellement de raisonner sur les compétences (détecter des compétences équivalentes, substituer des participants, etc.). Nous montrons comment optimiser la sélection de tâches au moyen de cette taxonomie. Par de nombreuses experiences synthétiques et réelles, nous montrons qu'il existe une amélioration significative de la qualité lorsque l'considère une structure hiérarchique de compétences au lieu de mots-clés purs. Dans une seconde partie, nous étudions le problème du choix des tâches par les participants. En effet, choisir parmi une interminable liste de tâches possibles peut s'avérer difficile et prend beaucoup de temps, et s'avére avoir une incidence sur la qualité des réponses. Nous proposons une méthode de réduction du nombre de propositions. L'état de l'art n'utilise ni une taxonomie ni des méthodes de classement. Nous proposons un nouveau modèle de classement qui tient compte de la diversité des compétences du participant et l'urgence de la tâche. Des expériences synthétiques et réelles montre que nous pouvons respecter les délais, obtenir des réponses de haute qualité, garder l'intérêt des participants tout en leur donnant un choix de tâches ciblé.

**Mots-clés** : crowdsourcing, allocation de ressources, taxonomie des compétences, attribution des tâches, classement des tâches, modélisation des compétences, modélisation des participants, allocation des tâches, diversification des tâches.

# Résumé

## Introduction

Le crowdsourcing sous la forme de plateformes participatives est un sujet qui intéresse à la fois la recherche académique [52, 19, 64, 54, 82] et l'industrie [36, 84, 81]. Dans l'introduction nous allons :

— donner notre définition de crowdsourcing,
— illustrer la notion de crowdsourcing,
— expliciter les défis de recherche pour les plateformes modernes de crowdsourcing en comparaison avec l'état de l'art,
— présenter nos contributions pour l'amélioration des plateformes de crowdsourcing,
— conclure en résumant nos résultats.

### Définition et types de crowdsourcing

Selon Jeff Howe qui a inventé ce terme, le crowdsourcing peut être défini de la façon suivante [51] :

> DEFINITION DU CROWDSOURCING SELON JEFF HOWE
> *"Le crowdsourcing représente l'acte d'une entreprise ou d'une institution qui confie une fonction à un (grand) ensemble de personnes indéfinies sous la forme d'un appel ouvert. Cela peut prendre la forme d'une production collaborative (lorsque le travail est effectué en collaboration) mais souvent aussi par des personnes isolées les unes des autres. La condition préalable cruciale est l'utilisation d'un appel ouvert et le large réseau de travailleurs potentiels."*

Ce mot est une combinaison des mots "crowd" (foule) et "outsourcing" (externalisation), ce qui lui confère les caractéristiques des deux termes. Comme l'a décrit Jeff Howe, le crowdsourcing peut être considéré comme un effet de levier des compétences humaines avec une manière parfaite, transparente et démocratique d'ouvrir et de résoudre des problèmes difficiles.

### Exemples et Cas d'Usage de Crowdsourcing

Grâce à l'utilisation d'Internet et de plateformes de crowdsourcing, il est maintenant possible de trouver des personnes en ligne qui ont des compétences rares et de

rassembler une "foule sage et intelligente". Ces techniques ont déjà eu un impact significatif dans la recherche contre le cancer [90], le repliement des protéines [64, 4], le SIDA [1], la démocratie participative, la transparence et ouverture gouvernementale [2], etc. Les défis à relever pour utiliser au mieux les talents disponibles et optimiser la qualité des résultats du crowdsourcing sont nombreux.

### Défis des Applications de Crowdsourcing

On a mentionné plusieurs exemples et cas d'usage où des systèmes de crowdsourcing ont été déployés pour un usage à la fois industriel et académique. Pour ces systèmes, de nombreuses difficultés existent : **comment identifier, recruter et fidéliser les participants, comment répartir les tâches pertinentes entre les participants correspondants et comment évaluer et rémunérer les participants**. De plus, les systèmes ne sont pas généralisables car la plupart du temps, une nouvelle plateforme doit être déployée pour une utilisation différente.

La situation actuelle dans les plateformes de crowdsourcing comme Amazon Mechanical Turk (AMT) et CrowdFlower est dépeinte dans Figure 1.4. Les participants ayant des compétences différentes et diverses se connectent à une plateforme, mais les plateformes ne tiennent pas compte de cette notion de compétence. Quelques travaux académiques récents étudient la modélisation de compétences sous forme de mots-clés ([18, 79]), mais ce modèle simple ne permet pas de raisonner sur les compétences (substitution des compétences par exemple en remplaçant un expert par un généraliste, etc. ). Dans les plateformes actuelles, les participants doivent sélectionner une tâche de leur goût, en fonction d'une barre de recherche et de plusieurs critères de tri (récompense, titre, etc.) comme le montre la Figure 1.4. Il en résulte une perte de temps au détriment du travail créatif puisque les participants prennent le temps de sélectionner une tâche pertinente ou de choisir selon leurs intérêts parmi un très grand nombre de tâches.

En ce qui concerne le problème du recrutement, de l'identification et de la rétention des bons participants, il y a un certain défi à relever pour modéliser la capacité des participants à accomplir une tâche. Dans les plateformes industrielles actuelles, les participants ne sont pas décrits en termes de compétences. D'autre part, le travail académique connexe utilise des balises ou des mots-clés pour décrire les tâches et les participants. Cependant, cette représentation par mot-clé ne permet pas de substituer des compétences similaires lorsque les noms ne sont pas similaires. Par exemple, une similarité de mots-clés ne peut pas capturer les termes "Programming" et "Java" comme des termes apparentés. Imaginez alors une tâche de programmation et un programmeur Java, on doit pouvoir assigner la tâche "programming" au programmeur "Java" puisqu'un programmeur Java est toujours un programmeur. Cependant, ceci n'est pas possible avec une simple correspondance de mots-clés. Le problème est encore plus important dans des contextes plus complexes où les noms de mots-clés n'ont pas de relation évidente.

1. http ://2beathiv.org/
2. http ://democracyos.org/

Afin de garantir une bonne répartition des tâches pertinentes entre les participants correspondants, il est difficile de proposer une méthode d'optimisation qui permette de faire correspondre les tâches aux participants. Une tâche donnée doit être attribuée à la bonne foule afin d'optimiser le temps et les compétences disponibles. Actuellement, les participants doivent choisir eux-mêmes leurs tâches ou même s'appuyer sur des forums astructurés [3] où d'autres participants peuvent les manipuler en proposant leur tâches comme de tâches intéressantes et bien rémunérées.

Défi 1 : Modélisation des compétences qui permet la substitution des compétences et l'appariement des tâches aux participants.

De plus, les listes de tâches dans les plateformes modernes de crowdsourcing peuvent avoir plusieurs milliers de tâches et de participants (comme vu dans la Figure 1.4, Figure 1.2 et également mentionné dans [82]). Dans ces plateformes, les participants choisissent parmi une énorme liste de tâches à l'aide de critères de tri simples et d'une barre de recherche pour rechercher des mots-clés. Le défi consiste à pouvoir aider les participants à choisir les tâches via des propositions de tâches. Ces propositions devraient toutefois être élaborées en fonction des critères appropriés pour maintenir la motivation des participants et tenir compte de leurs caractéristiques et de celles des tâches.

Défi 2 : Présenter à chaque participant une courte liste de tâches personnalisées parmi une longue liste de tâches disponibles.

Dans ce travail, on se concentre principalement sur un type de tâches appelées micro-tâches, des tâches nécessitant une seule compétence et on peut trouver et travailler sur des plateformes comme CrowdFlower et Amazon MTurk. Ensuite, on se focalise sur la modélisation des participants et la gestion des compétences en micro-tâches dans les applications de crowdsourcing, ce qu'on fait en abordant les deux principaux défis qu'on a mentionnés dans la section précédente (Section 1.4. Cependant, les contributions qu'on présente et qui abordent les défis ci-dessus sont plus génériques et peuvent également être utiles dans des environnements plus exigeants et sur différentes plateformes telles que Wikipedia, Quora [4], Foulefactory, Stack Overflow [5] etc.

### Un Modèle de Substitution des Compétences

Concernant le premier défi, afin de pouvoir raisonner et substituer les compétences dans des plateformes participatives comme les plateformes de crowdsourcing, on propose un système de gestion des compétences :

Contribution 1 : On propose un modèle de substitution des compétences qui utilise une taxonomie des compétences (comme celle que l'on voit à la Figure 1.6) et des algorithmes d'optimisation qui améliorent les correspondances entre tâches individuelles et participants pour l'externalisation publique. Illustration disponible dans la Figure1.5.

3. http ://turkernation.com/
4. www.quora.com
5. www.stackoverflow.com

On veut être en position d'attribuer un "core Java task" à un "core Java Programmer" (Figure 1.6). Ceci est possible lorsque on utilise des mots-clés pour faire correspondre les tâches avec les participants : cela s'appelle le crowdsourcing à forte intensité de connaissances et est déjà abordé dans le travail connexe [18, 79]. Cependant, si aucun programmeur Java de base n'est disponible, on doit prendre la personne la plus compétente suivante. En regardant la Figure **??**, il semble qu'on pourrait chercher quelqu'un avec de meilleures compétences, comme un programmeur "Java 1.8 thread" ou un programmeur "core C" qui a des compétences un peu moins pertinentes. Mais si on laisse un algorithme qui marche avec de "mots-clés" choisir au hasard, il ne sera pas en mesure de faire la différence entre un programmeur "Java 1.8 thread" programmateur, un programmeur "core C" et un jardinier alors qu'il y avait une personne plus appropriée disponible. Avec l'utilisation de la taxonomie, on pourrait être en mesure de correspondre à l'un ou l'autre de ces critères selon les disponibilités des participants qu'on avait dans notre foule. On est également capables d'éviter si possible l'utilisation d'un programmeur pour une tâche de jardinage et vice versa.

L'idée principale est qu'il s'agit d'une sorte de crowdsourcing où on devrait prendre en compte les compétences des participants pour les questions posées. Dans notre cas, on peut même améliorer les résultats en substituant des compétences différentes par des compétences équivalentes, meilleures ou progressivement pires pour optimiser le résultat de la qualité globale de la plate-forme.

Les résultats de cette contribution ont été publiés sur [65, 66]. Pour résumer les principaux points de cette contribution, on la décompose en des points plus simples :

— on utilise une taxonomie $T$ pour la substitution et l'identification des tâches pertinentes aux participants concernés,
— on propose une distance d'affectation basée sur la taxonomie entre une tâche et un participant qui est moins pertinente pour minimiser le compromis entre les différentes tâches,
— on fourni une expérimentation de synthèse étendue et une véritable expérimentation qui valide notre hypothèse sur l'importance des substitutions de compétences.

### Affectation Multi-objective des tâches aux Applications de Crowdsourcing

Après avoir exposé notre contribution pour le prémier défi, on présente notre contribution au deuxième défi :

> Contribution 2 : On propose un modèle qui sélectionne parmi une liste énorme de tâches disponibles, une liste courte et personnalisée de tâches que le participant peut choisir. Une illustration peut être vue dans la Figure 1.7.

On veut aider les participants à trouver des tâches pertinentes et leur offrir un choix significatif. Plus précisément, si quelqu'un est un spécialiste en "Java", "Gardening", "OOP" et "ML", on veut le proposer avec un sous-ensemble significatif des tâches disponibles qui correspondent à la plupart de ses compétences. Cette stratégie peut faire gagner beaucoup de temps au participant. De plus, on souhaite que les tâches urgentes soient présentées en premier lieu, c'est pourquoi on propose une mé-

trique d'urgence en fonction du délai de la tâche et on donne la priorité à ces tâches en premier. Cette stratégie garantit que les participants obtiennent une liste de tâches appropriées et variées, tandis que les tâches urgentes n'ont pas besoin de réponses.

On résume cette contribution aux étapes suivantes :

— On propose une mesure de pertinence pour une liste donnée de tâches à un participant en utilisant notre mesure de distance taxonomique,
— On définit une métrique de diversité pour une liste de tâches donnée à un participant qui est optimale lorsque plus de compétences sont prises en compte,
— On fournit une métrique d'urgence pour une tâche et on l'étend ensuite à une liste de tâches qui prend en compte les contributions faites pour une tâche, le temps restant pour la tâche et un temps moyen pour l'accomplir,
— On procède à une vaste expérimentation synthétique et réelle qui valide notre hypothèse sur la diversité et l'urgence des tâches par rapport à d'autres méthodes et
— On montre que les participants : (1) choisissent une liste de tâches intelligente parmi une liste exhaustive et exhaustive de tâches écrasantes et (2) choisissent les tâches au moins 2 fois plus rapidement dans la liste de présélection.

## Conclusion

Dans la présente thèse, on a identifié le crowdsourcing comme un domaine stimulant, passionnant et écrasant. On a défini quelles applications sont considérées comme crowdsourcing et on a confirmé l'importance du crowdsourcing en tant qu'outil efficace pour plusieurs domaines. Ensuite, on a présenté le travail connexe et les défis que doivent relever les systèmes actuels de crowdsourcing. Dans la section qui suit, on résume les défis, nos contributions et les résultats qui ont été fournis, ainsi que ce qui pourrait être amélioré. On conclut ensuite par une section qui discute de certains travaux futurs qui pourraient être réalisés pour prolonger nos contributions.

Dans le chapitre 4, on a montré comment assigner de manière optimale un ensemble de tâches à un groupe de participants. On a également proposé des façons d'optimiser la performance de cette mission avec un minimum de sacrifices dans la qualité, comme le montrent les ensembles de données synthétiques (4.3.3). On a également démontré et expérimenté sur le modèle étendu où les tâches demandent un minimum d'expertise et montré l'impact des méthodes proposées.

Afin d'améliorer notre contribution, on aura également pu aborder d'autres questions, telles que

— sur la façon d'obtenir des taxonomies de compétences et
— comment mettre à jour les profils des participants après la rétroaction du demandeur.

Plus tard, le chapitre suivant 5 étant donné un ensemble de tâches, une taxonomie des compétences et un ordre d'arrivée inconnu des participants pour une plateforme de crowdsourcing, on a montré l'importance de $relevance$, $diversité$ et $urgence$ des tâches individuelles sur les compétences de la tâche proposée.

Plus précisément, on a montré comment une note globale de pertinence, de diversité et d'urgence est optimisée grâce à nos méthodes et comment on peut améliorer la satisfaction totale et le temps nécessaire pour sélectionner des listes de tâches courtes pour les participants. Cependant, on n'a pas pris en compte d'autres paramètres :

— on n'a pas inclus le paiement dans la note globale et
— on n'a pas tenu compte de la répartition (distribution) des compétences des nouveaux participants.

En résumé, dans cette thèse, on a montré ce qui suit :

— l'importance de la précision des compétences pour une substitution plus effective dans le contexte de la qualité et
— l'importance de la diversité des compétences et de l'urgence de tâches pour promoter la satisfaction des participants et les garanties de la plateforme