Computer Science • 14 (2) 2013 http://dx.doi.org/10.7494/csci.2013.14.2.273

Peter Lavin
Eamonn Kenny
Brian Coghlan

COMPUTING RESOURCE AND WORK ALLOCATIONS USING SOCIAL PROFILES

Abstract *If several distributed and disparate computer resources exist, many of which have been created for different and diverse reasons, and several large scale computing challenges also exist with similar diversity in their backgrounds, then one problem which arises in trying to assemble enough of these resources to address such challenges is the need to align and accommodate the different motivations and objectives which may lie behind the existence of both the resources and the challenges. Software agents are offered as a mainstream technology for modelling the types of collaborations and relationships needed to do this. As an initial step towards forming such relationships, agents need a mechanism to consider social and economic backgrounds. This paper explores addressing social and economic differences using a combination of textual descriptions known as social profiles and search engine technology, both of which are integrated into an agent technology.*

Keywords agents, search, resource allocation, social, economic, Social Grid Agents, social profile

1. Introduction

Assume that several heterogeneous, distributed and disparate computer resources exist, many of which have been motivated by, and created for different and diverse reasons, and that several large scale computing challenges also exist with similar diversity in their motivations and objectives. The problems arising in trying to assemble enough of these resources to address such challenges are in the domain of resource and work allocation.

The different physical and geographically locations, plus distributed control and ownership of resources increases the difficulties faced. For owners of both work and resources to begin to address such challenges, relationships and collaborations need to be formed, i.e. there is a need to align and accommodate the many different motivations, reasons and objectives which may lie behind the existence of both the available resources and the challenges which may be addressed by them. Software agents are offered as a mainstream technology for modelling the types of collaborations and relationships needed for this [27]. However, the social and economic differences found between the different types of resources and work will influence the establishment of these collaborations. As an initial step towards forming such relationships, agents need a mechanism to consider social and economic backgrounds and origins.

This paper explores a solution for addressing social and economic differences using a combination of textual descriptions known as social profiles and search engine technology. These developments are integrated into an embryonic and rudimentary agent technology called 'Social Grid Agents' (SGA) [21] which has been developed in Trinity College Dublin [1]. SGAs were developed to address the diversity and complexity found in the allocation of Grid computing resources.

The remainder of this paper is laid out as follows. Section 2 discusses some prior work, some with automated and some with manual selections for addressing resource allocation problems. In Section 3, the concepts of 'social and economic' differences are explored and relevant examples of each are given. Section 4 describes SGA technology and introduces it both as a use case environment and as the original stimulus for deploying an implementation of social profiles. A solution for describing and differentiating between these differences is described in Section 5, also giving details of a prototype implementation of the solution. Sections 6 and 7 gives details of simple experiments, conclusions and future work.

2. State of the art

This Section briefly describes some prior research on combinations of agents, matching and search technology in the area of work and resource allocation for heterogeneous distributed resources.

Research by Gorodetsky [13] uses *matching* libraries embedded in P2P agents in a multiple agent system (MAS) which incorporates searching and matching of other agents. Services are defined using a key-pair description file and these can be searched and queried for service discovery and matching. This work uses pattern matching to search the expressions used in the matching mechanism but does not incorporate any other search technology.

The European Grid Infrastructure (EGI) [2] employs a job description language and a resource broker based on the ClassAd matching tool [22] (discussed later in Section 4.3) in order to match heterogeneous distributed resources to work. This approach derives from the EU DataGrid project [3], further refined in the EGEE, EGEE-II and EGEE-III projects [4].

Montella [20] describes the integration of a statistical language model called Latent Semantic Indexing (LSI) [12] with the ClassAd matching tool. LSI is frequently used in information retrieval and internet search engines. In this work ClassAd and LSI are used in a two stage process, where initial matches returned by ClassAd are optimised using a customised LSI algorithm. LSI is a similar search technology to the solution used here, but Montella's approach only applies it to terms used in the matching tool's expressions but not to any additional text as proposed here.

Hao [15] uses vector space model technology (VSM) [25] to rank the relevance of web services where large numbers of responses are returned during service discovery. This work uses textual descriptions of web services. Similarity measurements between two or more of these are combined with measurements of the level of connectivity a web services has to others. By doing this, a measure of the relevance and importance of a given web service is produced.

All of the above examples offer automated matchmaking and similarity evaluation mechanisms. Evaluation of computer resource allocations using human interpretation of descriptions and criteria can be seen in how World Community Grid (WCG) [5] make allocation decisions. WCG aggregates computer resources which have been volunteered by members of the public and allocates them to suitable projects. Resources are allocated to projects based on meeting criteria such as being *not for profit* and where *humanitarian benefits* may be forthcoming on completion of the work. For reasons which will be discussed in Section 3.1, computer resources volunteered by individuals (often home PCs) are generally only donated under largely similar conditions. The motivations of the project owners seeking volunteered resources are evaluated by WCG personnel in periodic committee meetings and not by any automated decision making process [6]. In a similar way, SweGrid (the Swedish Grid Initiative) and its subsidiary SNAC (Swedish National Allocations Committee) is another example where 'human' committees convene to evaluate and make decisions about resource allocations under their control [7].

3. Social and economic differences

To illustrate examples of different social and economic outlooks, different types of resources which may be suitable for addressing large computing challenges are listed and briefly discussed below.

1. *Commercial enterprises:* A commercial enterprise invests capital in computer resources. The resources may be used internally or rented to external users. It is likely that the owner's focus will be on maximising investment return or utility for their own benefit.
2. *Public investments:* Governments invest in large computational resources for scientific and academic institutions, doing so for the long-term benefit of society. The resources are owned and controlled by the institutions. Usage may be restricted to educational and research work but society in general may benefit from the results achieved.
3. *Volunteered resources:* In volunteer computing, a member of the public allows their privately owned computer to be used by a third party project. However, they may exercise discretion when choosing how their computer will be used. They may require, for example, that the work aims to bring a humanitarian benefit or increased scientific knowledge and they may be concerned about who will benefit from the outcome and how it will be used.
4. *Environmental factors:* Institutes, groups and corporations who are sensitive to the impact of their carbon footprint may have a preference for using computer resources which are powered (to varying degrees) using renewable energy. This criterion may be critical for both providers and users of commercial and volunteered resources [11].

Considering the cost, impact and potential benefits of accessing such computing resources, owners are likely to have preferences, aspirations and rules which will determine how their resources should be used. Owners of work may also have similar preferences. The following questions may arise:

1. Why are the resources being provided *or* why is the work being done? Who benefits and do they need and deserve those benefits?
2. What are the remuneration and cost expectations? Is the work not-for-profit? What is the financial status of the owner of the work?
3. What policies and ideological orientations exist which may influence allocations? Is there a preference for commercial, educational, humanitarian or scientific work?

Matching resources and work based on technical requirements such as memory requirements, CPU architecture and bandwidth can be done using a Boolean matchmaking mechanism such as ClassAd [23] or similar (as briefly discussed in Sections 2 and 4.3). However, matching based on social and economic outlooks and attempting to capture answers to some of the above questions presents a different challenge. A new approach is needed to accommodate the larger number of considerations which need to be dealt with.

3.1. Creating and measuring descriptions

As a solution to the problem of addressing social and economic differences, this research proposes exploiting the fact that subtle or nuanced differences in the aspirations and preferences discussed above can be described much more effectively using unstructured passages of text. If the 'meaning' of this text (i.e. what the author is attempting to say or describe) can be modelled and measured in some way, this may enable the establishment of similarities or differences between the entities described. One approach to *modelling meaning* is introduced below.

3.1.1. A model of meaning

In spoken language, the meaning of a word or term is often expressed by offering other words which are 'close' in meaning to it. This proximity metaphor suggests that meanings have a spatial property, that they occupy a location in some space occupied by other meanings and nearness in that space can be considered a metaphor for similarity [24].

Meaning can be linked to proximity in a mathematical sense in a way that can be measured and quantified. Early research by the American linguist Harris [16] led to the theory that meaning and distribution of words were closely related, and that the meaning of words could be indicated by the *context* in which they frequently occurred (i.e. words which occur near certain other words frequently mean the same thing). However, the context of a word may vary in scope from a sentence to an entire document. Research by Salton [25] created high-dimensional vectors based on statistical information about frequency and contexts of words in documents. This allows documents to be located geometrically in a high-dimensional vector space known as the *Vector Space Model* (VSM). Each document vector is a statistical representation of word occurrences in that document, and the context for each word is the entire document.

The motivation for locating document vectors in a geometric space is to be able to find other 'similar' vectors which are located nearby using the cosine angle or *cosine similarity* between them (and thereby find documents which have similar words and meanings). Using cosine similarity between vectors, similarity between documents is automatically extracted from the distribution of documents in the geometric space. In practice, the space is searched by creating a new vector using a relatively small number of words or terms known as a *query* and then finding other vectors 'near' to this. It should be noted that this approach of measuring similarity between documents only yields a similarity measurement, and does not offer any form of classification other than to differentiate it from another document relative to the query.

The 'similarity' approach provided by the VSM is widely used in internet search engines where web pages and documents are retrieved based (among other things) on occurrences of the words they contain. A typical VSM implementation maps text documents to individual entries within the space (often known as an *index*). Entries in the VSM or index have a flat structure consisting of one or more fields. Each field can contain the text for different categories of information. Typical categories are 'author', 'contents', 'abstract' and 'main body' (of the article, book, etc.). When searching an index, specific fields can be targeted with query terms, or an entire document can be searched, returning results which have that term present in any field.

3.2. Social profiles

To recap, the overall motivation for defining social profiles suitable for integration into agents is to address social and economic differences, which in turn will facilitate establishing relationships and organisations in a way which respects the requirements of owners of work and resources. This research proposes exploiting the measurement of similarity between documents which describe entities (in this case consumers and providers of resources) and to explore if it is a suitable and feasible way to find similarity between such entities.

For each entity, a collection of text which was compiled either by or about that entity is created. This description is called a *social profile* and aims to capture any aspect of an entity which can be described using text. This in effect allows the full resources of language to be used as opposed to, for example, a restrictive set of key-pair values or a questionnaire (see also Section 3.3). Each textual description document becomes an entry in the VSM index and is a statistical representation of a social profile there.

3.3. Content and design of a social profile

For the purpose of this research, any text which is relevant to the described entities will be used as content for social profiles. In the digital age, sources of such text are plentiful.

By way of discussion, consider the scenario where an e-Commerce company needs to create descriptions of all its sellers in order to present them to potential buyers. An example of such a company is eBay, the internet auction site [8]. A potential buyer is presented with a collection of feedback comments collected from past customers after previous transactions. Taken in total, this is in effect a description of the seller. It consists of a short input where the seller describes itself. Appended to that are several short comments supplied by past customers (this can be called 'reciprocated' or third-party information). In the case of eBay, reciprocated information has a structure imposed on it in the form of a list of questions and a short Likert scale dialogue [19] about delivery times and quality of packaging. What is interesting about this total collection of text is that it (automatically) contains words which describe the seller. If this were indexed in a VSM, statistical information about occurrences of 'superlatives' and other positive terms should differentiate a good seller from one who frequently has negative feedback written about them (however in the case of the eBay website, this is not done, instead they are simply presented to the potential buyer in summary form).

By contrast, another example of a 'product' offered to 'buyers' (by the film making industry in this case) is a feature film. The buyers here are people who wish to go to a cinema to view such a 'product' for their entertainment; they also need a description when deciding on which if any film to view. Film reviews also constitute a textual description but tend not to have a structure similar to that of the eBay feedback details discussed above. Reviewers can write whatever they feel is appropriate and there are several spectra along which to describe a film (for example directing, acting, plot, etc.). However, the aggregation of even a number of film reviews cannot be said to have the same usefulness as an eBay feedback description as personal tastes (among other things) play such a large part in what is a satisfactory film viewing experience. Feedback to eBay sellers and film reviews are very different, one having some structure, while the latter is more likely to be an unstructured and amorphous passage of text.

The origin of the text is also important; and descriptive text from both the entity itself and third parties both have a place in a social profile. In the eBay information discussed above, positive information from a third party is difficult to fake, and negative reviews are difficult to remove and this adds authenticity to the description. In a social profile however, information which comes directly from an entity is also important as they are often best placed to set out what their objectives are or to provide an overview or summary of their mission.

The question arises: what type of information is relevant when describing computing resources and work and what type of structure should it have? For this research, a broad view is taken, insofar as any text which applies to an entity contributes to a description of it, therefore could aid a resource allocation decision.

3.4. Structure of social profile

As seen in the eBay example discussed above, judicious use of Likert scales and structured questions is useful. However, it should be remembered that in that instance, the reason for compiling a description of a seller is to allow a potential buyer to evaluate whether a successful transaction is likely to be completed with that seller or not. Using that definition, the eBay description may be quite narrow and does not need describe a seller beyond that criterion.

The structure of a social profile needs to consider a broader set of criteria. It is not feasible to pose the many questions which could be posed when trying to describe an entity under headings loosely described as ideological, political, economic and sociological. For this reason, at least to begin with, the structure of a social profile must rely mainly on collecting descriptive text. This text may then be indexed under a relatively small number of categories which are dictated by the origins of the text.

For these reasons, the initial prototype social profile contains three sections.

1. Fixed and incontrovertible information or ‘logical data’, such as the creator’s or owner’s name, address, type, URI, address or location.
2. The creator or owner’s input, i.e. what it claims to be a true description of itself, its current and past activities and publications.
3. Third party (reciprocated) information and references, reviews, feedback, replies in blogs and forums.

The rationale for this structure is that its different parts can be mapped to separate ‘fields’ in a VSM entry. Data in the first section is indexed with each item as a separate field but is not statistically analysed for context and frequency. This allows each item to be queried and retrieved using Boolean matching. The second and third Sections consist of large bodies of text and are each indexed in two separate fields. The text is statistically analysed, each becoming part of the vector for the social profile. This allows the profile to have a more general form where different types of data are accommodated.

Although the structure outlined above may suffice for a prototype, extensibility will also be a required feature of any implementation. In the case of VSM entries or documents, at a minimum, it will be required that the number of fields be able to be increased without too much disruption to an agent's abilities or its environment.

4. A use case: social profiles in SGAs

This section further describes a use case agent technology in which social descriptions may be deployed. This use case was the original stimulus for this research.

Social Grid Agents (or SGAs) are essentially Java applications, and were originally implemented as web services in the prototype of Pierantoni in [21]. Each SGA presents a uniform functional message processor interface which receives, processes and returns a message. The use of web services makes them continuously running self-contained entities and allows them to be deployed in distributed locations, using SOAP messages over HTTP for communication. They react to each message they receive and can also sense their environment. Unlike agents described in [27, 9], SGAs do not exhibit reasoning or artificial intelligence attributes and this distinction is made in [21, p.39]. They can be considered 'instruments' which enforce their owners policies and carry out their wishes when interacting with other agents. These policies are defined in messages sent to agents and also in the agents' internal settings.

4.1. SGA architecture

An integral part of an SGA is its *hashmap*, the basis for their modular architecture. The binary abilities of the agent are stored in this hashmap and are instantiated as required when processing a message. *Binary abilities*, like the agents of which they are a part, are essentially message processors too. Each *ability* has the logic or 'knowledge' to deal with a particular type of message and can carry out an action in the environment of the agent, including acting as an interface to underlying services. In a system of multiple SGAs, relationship with other SGAs (and their nature such as ownership and direction of control) is also defined by abilities.

4.2. Social and production topologies

When an SGA acts as an interoperability wrapper for an underlying service, such an agent can be said to 'produce' that service, i.e. it is a 'production agent'. An agent controlling one or more production agents may also interact socially with other similar 'controlling' agents. Therefore, agents can be said to fall into topologies or layers which are termed 'social' and 'production' layers. An agent in the 'social layer' which controls 'production agents' can trade and exchange the 'utility' of these agents with other agents in the social layer.

4.3. ClassAd in SGAs

The functional language ClassAd [22] from the Condor project [26] is integral to the SGA architecture and implementation. Named from an abridgment of '*Class*ified *Ad*vertisements', it was originally developed for *describing and matching* computer jobs and resources. Its main features are:

- Extensible key-pair structure of expressions.
- Matching of expressions and ranking of matches.
- Expressions can be text or numeric values, or (most usefully) can be functional programming statements.

SGAs exploit these three features. ClassAd expressions are used to define all SGA communication messages for both internal and inter-SGA information flows. For communication internal to an agent, messages are passed through the agent's components and are matched against other ClassAd expressions embedded in the agent. The outcome of these evaluations determine the behaviour and response of the agent. The rationale for using a functional language for this is to provide *side-effect-freedom* and *referential transparency*, thereby offering the potential to audit and prove an evaluation outcome from what may be a number of complicated (and possibly nested) expressions. This is desirable, especially for SGAs in an economic oriented market environment.

4.4. The background and motivation for SGA development

The current SGA implementation and its internal matching and evaluation mechanism (ClassAd) are agnostic to the tasks, actions and values which they handle. In the prototype development [21], SGAs were deployed in a metagrid environment. These motivations now also extend to federated clouds and hybrid grid-clouds, and possibly to any computing environment where resources and work are to be allocated.

Although not yet fully developed, it is envisaged that a multiple agent system (or MAS) with large numbers of agents (100s or 1000s) populated by SGAs may exist. This will mean that there will be many different agent creators and owners. Each owner may own and control one or more agents, with each one having an interest or stake in the environment. Candidates represented by agents in such an MAS are entities such as individuals, institutions or enterprises which produce or consume computing services. In effect, the SGAs become producers and consumers of these services, co-existing and interacting with other agents as needed.

4.4.1. Other developments in SGAs

Towards the overall motivation behind SGAs, other developments in this agent technology [18] facilitate the fast creation and deployment of several SGAs within an MAS. This work allows an MAS of SGAs to be enlarged, reduced and reconfigured, all without impacting on other existing agents therein. Serialized binary *abilities* encapsulated in an XML file can be transported to SGAs for configuring and enhancing a new or existing agent. The use of an XML file requires a supporting HTTP server in the MAS environment and this infrastructure will be further utilised in the implementation described below.

5. Prototype implementation

In the scenario outlined in Section 4.4, both an SGA, its abilities and its social profile are created by the agent owner. For this research, all of the different sections of a social profile will be compiled by the owner. Questions of transparency and trust which would naturally arise in the creation of profiles by individual and non-trusted parties are not considered at this stage. These concerns may be addressed at a later stage, for example, by each agent owner publishing its (perhaps digitally signed) social profile, along with references to its various sources as well as embedding it within its agents.

5.1. Lucene as a VSM implementation

Lucene [10] is a Java^{TM} implementation of a Vector Space Model search engine and is used in this research. It provides extensive libraries and an API for interacting with a VSM index.

Central to Lucene's architecture is the concept of an *inverted index*. This index is populated with multiple Lucene documents and this combined data structure has enough information to model the VSM.

A single Lucene document represents a single (text) entity, be that a book, a web page or anything which can be considered as an 'atomic' unit for indexing and retrieval by searching. A Lucene document consists of one or more fields. Depending on the nature of the text that a field holds, it may contain one or several *terms*. As an example, for a (digital) book, content like 'author' and 'title' are usually added to their respective fields as single terms (i.e. not broken down into smaller terms). This allows these fields to be queried and matched (similar to the Boolean matching of values offered by ClassAd). Longer passages such as an abstract or a main body of text are usually assigned to other fields. Such a text is typically broken down (or analysed) into the many individual terms which it contains. Figure 1 is a schema illustration of a simple Lucene document. Overall, the terms in each field and their frequencies make up the vectors which occupy the different locations in the Lucene VSM index.

For clarity, the term *Lucene document* will be used to refer to documents used in the (Lucene) VSM index.

5.2. Content sources for social profile

For experimental social profiles used in this research, data is largely garnered from websites and down-loadable press articles which refer to a selection of entities. How-

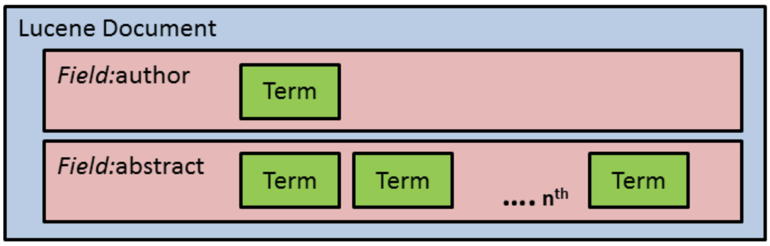

Figure 1. A Sample Lucene Document Schema.

ever, text for the third party area may also be gleaned from relevant reviews, forums and discussion groups as these sources contain descriptive words and terms which apply to the entity.

5.3. XML as a profile container

Section 4.4.1 outlines how separate developments in SGAs use XML files and how this necessitates a HTTP server in the MAS environment. These factors, when combined with the human readability and extensibility makes XML a logical choice of container for the creation of social profiles. Java implementations of XML parsers are ubiquitous and easily integrated in to the SGA development environment and to SGA abilities. In the initial implementation, this XML file contains an element for every field required in the Lucene document. The logic to transfer the content of these elements to the Lucene document fields as described in the structure outlined in Section 3.4 is dictated by the logic in the ability which processes the XML file within the agent. A sample of the structure of this file is shown in Figure 2. Note in this figure that there can be multiple fields for 'logical data' (item 1 in Section 3.4) but only one field each for the *owner's* and *third party* inputs.

5.3.1. Transport and indexing of the social profile XML file

When a social profile is completed by its creator, it is copied to the HTTP server within the MAS. A message containing the URL of this XML file is sent to the agent and used by the agent to fetch its social profile from the HTTP server. The XML is parsed and each element is added to a Lucene document and subsequently added to the VSM index within that agent. At this point, an SGA has a single document in its Lucene index which is in effect a high-dimensional vector representation of a textual description of its owner.

5.4. Integration of Lucene with SGAs

In order to index the social profile within the agent, each SGA needs to have access to the core Lucene libraries. A Java archive file (JAR) which is currently packaged with the agent's web service archive file provides this. The agent also needs to be configured with binary *abilities* which can invoke the API provided by these libraries.

```
<social-profile>
  <profile>
    <documentField>
      <fieldName>agentName</fieldName>
      <fieldContent>agent_0001</fieldContent>
    </documentField>
    <documentField>
      <fieldName>type</fieldName>
      <fieldContent>Academic Institution
                </fieldContent>
    </documentField>
    <documentField>
      <fieldName>ownerDescription</fieldName>
      <fieldContent>Text of owner's description is here...
                </fieldContent>
    </documentField>
    <documentField>
      <fieldName>thirdParty</fieldName>
      <fieldContent>Third party description is here...
                </fieldContent>
    </documentField>
  </profile>
</social-profile>
```

Figure 2. Structure of Original Social Profile XML File.

In keeping with SGA architecture, one ability is added for each specific task. For example, initially, each agent requires the ability to fetch, parse and index its own social profile. Depending on the complexity of the operations which need to be carried out on the index, multiple binary abilities can be added using the mechanism briefly described in Section 4.4.

Although Lucene allows an index to be stored in system memory, for long-term continuous operation the agent also needs access to persistent storage on the hard-disk of its host machine. The disk of the SGA host machine is accessed when social profiles are written and read from the index.

5.5. Extensibility of a social profile

For consistency throughout an MAS, it may not be desirable that the overall structure of the social profile is changed dramatically. However, exploiting more of the potential and flexibility of the Lucene document and search API may be useful. Examples of this may be adding location information such as a GPS location of the agent, and an addition of data fields which may be used to implement directory structures.

In the social profile described in Section 3.4, the structure of the Lucene Document which is indexed in an agent is mapped directly from the structure of the XML. It is also the case that the logic coded into the agent ability used to index this information has to match this structure. For the agent and profile creator, this in effect means that adding further elements to an XML file requires a change in the logic available within the agent. In the current SGA architecture, this means that a new binary *ability* needs to be introduced (i.e. reconfiguration of the agent). While this can be done using the mechanism of Section 4.4, it is restrictive and does not lend itself to the flexibility and extensibility of the social profile structure.

5.5.1. An extensibility solution

The solution developed to overcome this restriction involves being able to define a schema for a social profile XML file and including it within that same file. A typical XML schema is not suitable for this as such a schema would still have to be 'translated' to an actual Java binary code to implement the logic needed for the additional fields. This research implements an *encapsulated binary schema*, where the creator of the XML social profile must also prepare a Java binary class, a representation of which will be added to the social profile. This class contains the required logic needed to index each individual field as specified by the creator (for example whether fields are to be analysed or not). During the indexing operation within the agent, this class is used to add the contents of each XML element to the Lucene Document.

This approach is possible for a number of reasons. A Lucene document has a non-hierarchical or *flat* structure. Therefore, an XML file which reflects this will have a similar flat structure. This means that the elements in the XML file which contain descriptive information can be parsed and used to create a list. Each item in that list can simply be used to call a list of corresponding methods in the Java class (using Java Reflection). As each method is invoked, the contents of the corresponding element in the social profile XML file is added as a field to the Lucene Document. The structure of the XML file required for this is shown in Figure 3. Note the addition of the `<profileClass>` element, and how all `<documentFields>` elements are now children of an `<indexableFields>` element. The overall sequence for social profile creation, and its subsequent indexing are outlined below. This sequence of steps which takes place within an SGA during indexing is further illustrated in Figure 4.

5.5.2. Sequence during profile creation

In the development environment of the agent and social profile creator, the following takes place:

1. The social profile XML file is compiled as before.
2. A Java class is created (`ProfileStructure.java`), having a set of methods which meets the indexing requirements of the created profile.
3. This code is then compiled, whereupon the resulting `ProfileStructure.class` file is converted to a Base64 [17] string.
4. This string is added to the social profile XML file (under a separate element to the social descriptions part).

The names of the methods in the Java binary must match the names of the elements in the social profile.

```
<social-profile>
     <profile>
          <profileClass>
               <base64Code>yv66vgAAADEAnwcAAgEAOgCeRAk=
             </base64Code>
          </profileClass>
          <indexableFields>
               <documentField>
                    <fieldName>agentName</fieldName>
                    <fieldContent>agent_0001</fieldContent>
               </documentField>
                  ...
                  ...
          </indexableFields>
     </profile>
</social-profile>
```

Figure 3. Structure of Modified Social Profile XML File.

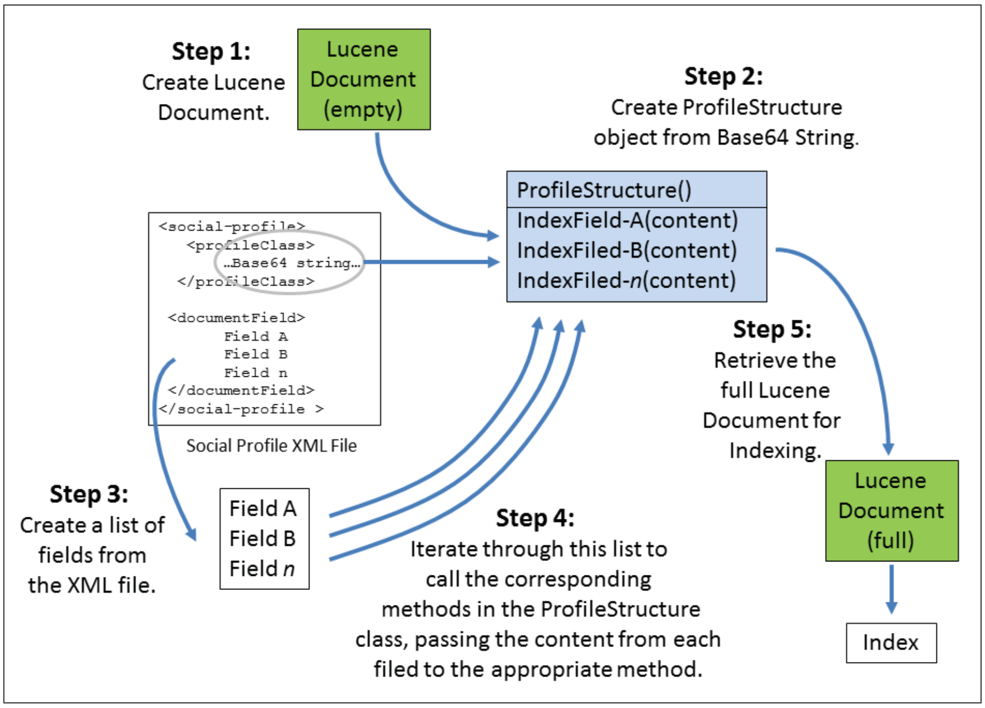

Figure 4. Sequence for Social Profile Indexing Using an Encapsulated Class.

5.5.3. Sequence during indexing

Within the agent, the sequence in the 'indexing ability' for processing a social profile using the *encapsulated binary schema* is as follows:

1. An empty Lucene Document is instantiated (as before).
2. The Base64 string is parsed from the social profile, marshaled to become a Java object and is instantiated with the empty Lucene Document being passed to the constructor as an argument.
3. The elements in the social profile which are children of `<indexableFields>` (containing descriptive information) are then parsed and compiled into a list.
4. Iterating over this list, the corresponding setter methods in the Java class are called, adding the contents of each `<fieldContent>` element to the appropriate fields in the Lucene Document.
5. On completion of this iteration, the agent indexing *ability* calls a getter method on the `ProfileStructure` class which returns the (by now filled) Lucene Document. At this point, it is then added to the agent's index.

The result of this development is that the number of fields in a social profile can now be increased or decreased without making corresponding changes to the indexing ability in the agent; instead, changes are made to the code of the `ProfileStructure` class. The encoded Java class is, in effect, a schema for the social profile and comes encapsulated within the social profile.

6. Experiments and outcomes

6.1. Social profile similarity measurements

This experiment aims to demonstrate the similarity measurements obtainable from Lucene in the context of this application.

16 different entities were selected as subjects for this experiment. All were chosen for their high media profiles, each one being the subject matter of several readily available articles of digital text. These entities were loosely chosen from four different categories as follows: (1) Business corporations, (2) Universities, (3) Government (or government like) agencies or departments, and (4) Charitable/benevolent foundations. Table 1 presents the names and categories of these.

6.1.1. Methodology

For descriptions of the above entities, 16 text files were created using a combination text harvested from each entity's web pages and from related press articles. These files ranged from 930KB to 30KB in size and were indexed to a Lucene index (outside of an SGA) using a custom made Java program. The program had two functions, to insert each one of them into an index as a separate document; and to allow the user to search and retrieve documents from that index using a query formed from a number of terms.

Table 1
Categories and Names of Entities Used.

Category	Entity Name
Commercial Companies and Corporations	Vodafone Ryanair General Electric, USA Virgin Atlantic
Universities, Academic Institutions	Trinity College Dublin, Ireland Imperial College London, UK University of California Berkeley, USA Princeton University, USA
Charities and Benevolent Foundations	Bill And Melinda Gates Foundation Doctors Without Borders The Carnegie Foundation The Global Fund
Government or Similar Departments and Institutions	US Govt, Department of Commerce Irish Govt, Department of Agriculture, Food and Marine Irish Govt, Department of Education Skills European Commissioner for the Environment

To establish if useful similarity measurements could be obtained by searching, multiple searches on this index were carried out using terms associated with each category. Terms were arbitrarily selected and Table 2 gives, as examples, the terms used to search for documents from the categories *Commercial Companies and Corporations* and *Universities, Academic Institutions.* These lists of terms (or words) are passed to the Lucene API and are transformed into the required structure for searching.

Table 2
Terms Queries Used for Searching.

Query	Category	Terms Used
1	Commercial Companies and Corporations	corporate profit loss sales assets industry production goods resources revenue growth executive
2	Universities, Academic Institutions	degree masters research funding science arts humanities university graduate campus student lecturer

6.1.2. Summary of outcome

Table 3 contains an extract from the search results obtained by searching on the index using the sets of terms detailed in Table 2. Table 4 presents (abridged) results returned and the similarity results (or score) for each one. This score indicates how similar (using *cosine similarity*) the vector created from the terms was to the documents returned.

It can be seen that, of the top five results for both searches, four are from the category related to the query terms used. It can also be seen that results close to the bottom of the table have similarity scores which are much smaller than those near the top.

Table 3
Abridged Results from Query Terms 1 in Table 2.

Rank	**Similarity score**	**Entity name**
1	0.129878	General Electric, USA
2	0.128846	Vodafone
3	0.108973	Ryanair
4	0.066199	US Govt Dept of Commerce
5	0.051682	Virgin Atlantic
6	0.045555	Doctors Without Borders
...	...	...
13	0.011785	Irish Govt, Department of Education Skills
14	0.010598	Imperial College London, UK
15	0.008778	Princeton University, USA
16	0.004298	Trinity College Dublin, Ireland

Table 4
Abridged Results from Query Terms 2 in Table 2.

Rank	**Similarity score**	**Entity name**
1	0.189299	University of California Berkeley, USA
2	0.181311	Princeton University USA
3	0.162844	Trinity College Dublin, Ireland
4	0.135752	The Carnegie Foundation
5	0.060671	Imperial College London, UK
6	0.04841	Bill And Melinda Gates Foundation
...	...	...
13	0.005751	Vodafone
14	0.004885	European Commissioner for the Environment
15	0.003274	Virgin Atlantic
16	0.000312	Ryanair

6.2. Proof of concept social profile functionality in SGAs

This work demonstrates, as an example, the type of functionality which may be used in relation to social profiles in SGAs.

Three 'proof of concept' social profiles were selected from the pool of 16 described in Section 6.1 and were indexed in three different SGAs using the agent ability described in Section 5.5.1. Functionality in the form of further abilities relating to social profiles has also been developed and tested in SGAs to request and carry out the exchange of these social profiles. These include:

1. The ability to request another agent's social profile and index it when received.
2. The ability to send an agent's own profile to another agent (e.g. in response to a request as in item 1).
3. The ability to send a social profile to another and request it to return 'similar' profiles it may have stored in its index.
4. The ability to respond to a request for 'similar' profiles (i.e. item 3).

The social profile used by the ability described in item 2 was retrieved from the agents own index using a simple term query, using the agent's own name (known to it) to search its index.

The ability in an agent which responds to the request described in item 3 used a different procedure to retrieve the 'similar' profiles. A Lucene document (a software object) was serialised and sent from the requesting agent to the receiver. On receiving this, the other agent used a *MoreLikeThis* query, a feature included in Lucene. In summary, this allows Lucene to use an existing document as a query to search for other 'similar' documents in an index. Although only tested here on an index which contained a small number of entries, the implementation of this ability allows agents' indexes (containing several social profiles) to be searched for entries which are 'like' or 'similar' to the profile on which the search *query* is based.

6.3. ProfileStructure class

For the testing of the extensibility work described in Section 5.5, varying numbers of XML file elements and corresponding numbers of ProfileStructure class methods were used.

Java Reflection exception catching was used to 'gracefully' catch mis-matches in the structure of the XML file and the methods coded in the binary class. The largest binary class tested had 30 methods and occupied 5.8KB on disk. This converted to a Base64 string of 7.8KB (i.e. a text string of approximately 7800 text characters). With this and other files tested, conversion from the Java binary format to Base64 carried an (linear) overhead of 35% in file size.

7. Future work and conclusions

The similarity score results outlined in Section 6.1.2 show that query terms, when judiciously selected, can be used to retrieve social profiles roughly from the categories that they were originally created in. Whereas this is to be expected from current *search* technology, it also shows that the similarity scores from these results may be useful as a measurement and as a value useful in work allocation decisions made within SGAs.

The flexibility of the Lucene Document when combined with the extensibility solution described in Section 5.5 allows for several additional elements and corresponding fields to be added to a social profile. This may allow features to be implemented within an agent technology which are not directly related to social and economic descriptions. Examples of these are adding *longitude and latitude* coordinates for an agent. These, for example, may be searched using Lucene's numeric range query type.

Given the capacity of a Lucene document to store Boolean or logical type data, the social profile format may also be used as an *record* in a directory service. An agent providing this service could aim to collect a large number of social profiles and could also implement a directory protocol, be it *hierarchical*, *relational* or other.

In agent technology, a *conversation policy* is an agreed protocol used between two agents to carry out a task [14]. For an agent to carry out such a task, each must have a prerequisite set of abilities. Where social profiles for several agents stored in the index of a directory agent, indexing the list of abilities available in an agent will allow those abilities to be searched and retrieved.

7.1. Conclusions

Although similarity measurements between social profile are demonstrated, larger numbers of these profiles will be needed in order to statistically establish their usefulness for establishing measurable differences between the entities described. The MAS architecture of Section 4.4 enables such practical large-scale experiments.

A prototype implementation has successfully been tested which leads the way to developing further agent abilities which will allow for the distribution and exchange of social profiles. The versatility of XML, the Lucene document structure and the *use case* agent technology (SGAs) facilitate the integration and amalgamation of these technologies to an extent that makes further investigation of social profiles feasible.

Acknowledgements

The prototype development of SGA described in Section 4 to 4.3 is the work of Gabriele Pierantoni [21]. Subsequent developments to SGAs outlined in Section 4.4.1 and onwards are by the author.

References

[1] `https://www.scss.tcd.ie/research_groups/cag/`. Computer Architecture and Grid Research Group, Trinity College Dublin
(see also `http://www.grid.ie`).

[2] `http://www.egi.eu/`. The European Grid Infrastructure.

[3] `http://eu-datagrid.web.cern.ch/eu-datagrid/`. The EU DataGrid Project, Retrieved Oct 2012.

[4] http://public.eu-egee.org/. European Grids for Science Accessed Nov 2009.

[5] http://www.worldcommunitygrid.org. Accesssed May 2009.

[6] Personal communication with Kevin Reed of World Community Grid and IBM.

[7] http://www.snic.vr.se/about-snic/documents/snac-strategic-documents/SNAC-policy.pdf. Swedish National Allocations Committee, published by SweGRID, the Swedish Grid Initiative, Accessed 2010.

[8] http://www.ebayinc.com/,http://www.ebay.com/. eBay, Consumer-to-consumer and business-to-consumer online auction website.

[9] http://www.fipa.org/. Foundation for Intelligent Physical Agents, Accessed April 2011.

[10] http://lucene.apache.org/. The Apache Lucene project, Accessed March 2012.

[11] Daouadji A., Nguyen K. K., Lemay M., Cheriet M.: Ontology-Based resource description and discovery framework for low carbon grid networks. pp. 477–482, October 2010.

[12] Deerwester S., Dumais S. T., Furnas G. W., Landauer T. K., Harshman R.: Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

[13] Gorodetsky V., Karsaev O., Samoylov V., Serebryakov S.: P2P agent platform: Implementation and testing. In S. Joseph, Z. Despotovic, G. Moro, S. Bergamaschi, eds., *Agents and Peer-to-Peer Computing*, vol. 5319 of *LNCS*, chapter 4, pp. 41–54. Springer, Berlin, Heidelberg, 2010.

[14] Greaves M., Holmback H., Bradshaw J.: What is a conversation policy? In *Issues in Agent Communication*, vol. 1919 of *Lecture Notes in Computer Science*, chapter 8, pp. 118–131. Springer Berlin/Heidelberg, 2000.

[15] Hao Y., Zhang Y., Cao J.: Web services discovery and rank: An information retrieval approach. *Future Generation Computer Systems*, 26(8):1053–1062, October 2010.

[16] Harris Z. S.: Papers in structural and transformational linguistics. Formal Linguistics Series, vol. 1., 1970.

[17] Josefsson S.: The Base16, Base32, and Base64 Data Encodings. RFC 4648 (Proposed Standard), October 2006.

[18] Lavin P., Coghlan B.: Dynamic proliferation of agents in a multiple agent system. In *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference*. IEEE, 2013 (to appear).

[19] Likert R.: A technique for the measurement of attitudes. *Archives of Psychology*, 22(140):1–55, 1932.

[20] Montella R., Giunta G., Riccio A.: An integrated ClassAd-latent semantic indexing matchmaking algorithm for globus toolkit based computing grids. In R. Wyrzykowski, J. Dongarra, K. Karczewski, J. Wasniewski, eds., *Parallel Processing and Applied Mathematics*, vol. 4967 of *LNCS*, chapter 100, pp. 942–950. Springer, Berlin, Heidelberg, 2008.

[21] Pierantoni G.: *Social Grid Agents.* PhD thesis, University Of Dublin, Trinity College, September 2008.

[22] Raman R., Livny M., Solomon M.: Matchmaking: An extensible framework for distributed resource management. *Cluster Computing*, 2(2):129–138, September 1999.

[23] Raman R., Livny M., Solomon M.: Matchmaking: Distributed resource management for high throughput computing. In *Proc. of the Seventh IEEE International Symposium on High Performance Distributed Computing*, pp. 28–31, 1998.

[24] Sahlgren M.: *The Word-Space Model: using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces.* PhD thesis, Stockholm University, 2006.

[25] Salton G., Wong A., Yang C. S.: A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, November 1975.

[26] Thain D., Tannenbaum T., Livny M.: Distributed computing in practice: The condor experience. *Concurrency and Computation: Practice and Experience*, 17:2–4, 2005.

[27] Wooldridge M., Jennings N. R., Kinny D.: A methodology for agent-oriented analysis and design. In *AGENTS '99: Proc. of the third annual conference on Autonomous Agents*, pp. 69–76, New York, NY, USA, 1999. ACM.

Affiliations

Peter Lavin
School of Computer Science, Trinity College Dublin, lavinp@cs.tcd.ie

Eamonn Kenny
School of Computer Science, Trinity College Dublin, ekenny@cs.tcd.ie

Brian Coghlan
School of Computer Science, Trinity College Dublin, coghlan@cs.tcd.ie

Received: 1.11.2012
Revised: 4.01.2013
Accepted: 11.02.2013