Requirements Eng (2011) 16:209–226
DOI 10.1007/s00766-011-0121-4

Application of swarm techniques to requirements tracing

Hakim Sultanov · Jane Huffman Hayes · Wei-Keat Kong

Received: 13 November 2010 / Accepted: 15 May 2011 / Published online: 11 June 2011
© Springer-Verlag London Limited 2011

Abstract We posit that swarm intelligence can be applied to effectively address requirements engineering problems. Specifically, this paper demonstrates the applicability of swarm intelligence to the requirements tracing problem using two techniques: a simple swarm algorithm and a pheromone swarm algorithm. The techniques have been validated using two real-world datasets from two problem domains. The simple swarm technique generated requirements traceability matrices between textual requirements artifacts (high-level requirements traced to low-level requirements, for example). When compared with a baseline information retrieval tracing method, the swarm algorithms showed mixed results. The swarms achieved statistically significantly results on one of the secondary measurements for one dataset compared with the baseline method, lending support for continued investigation into swarms for tracing.

Keywords Information retrieval · Requirements traceability · Swarms · Software engineering

1 Introduction

Despite the importance of software requirements, far too many practitioners forge ahead with coding before understanding the problem that needs to be solved. If requirements are captured, they may not be documented, analyzed, kept up to date, or traced as the software development life cycle progresses. There is much evidence to show that the lack of requirements, or of quality requirements, inevitably leads to low software quality [1–4]. Requirements are crucial when developing mission- or safety-critical systems, where the consequences of poor quality can be the loss of life or damage to the environment.

Activities can be undertaken to improve the quality of requirements, including, but not limited to, requirements consistency checking (functional and non-functional), interface requirements consistency checking, requirements reading (to look for fault types such as ambiguous requirements, incomplete requirements, etc.), and requirements tracing (to ensure that requirements are addressed in subsequent artifacts). Some of these activities have been supported by automated techniques. These techniques, though, are not fully automatic, are not general-purpose, have not been validated on large, real-world systems in numerous domains, and still require much effort on the part of the human analyst [5]. As a result, researchers continue to search for new and better techniques to improve the quality of requirements.

Swarm techniques apply swarm intelligence, a property of a non-centralized group of non-intelligent self-organized agents that collectively behave to perform work [6]. Swarm techniques have been demonstrated to work well on problems in nature such as finding the shortest route to a food source, cooperating in carrying large items, building a nest, etc. In computer science, swarm techniques have assisted with problems such as network routing/network management, power plant maintenance scheduling, load balancing in distributed systems, image compression,

H. Sultanov · J. H. Hayes (✉) · W.-K. Kong
University of Kentucky, 301 Rose Street,
Lexington, KY 40506-0495, USA
e-mail: hayes@cs.uky.edu

H. Sultanov
e-mail: hakim.sultanov@uky.edu

W.-K. Kong
e-mail: wkkong1@uky.edu

www.manaraa.com

personalized web search, etc. Swarm techniques exhibit emerging behavior not found in deterministic techniques and often overcome issues such as the problem of local minima.

Swarm techniques have been successfully applied to problems in software maintenance. For example, Antoniol et al. [7] applied a number of such techniques to the problem of project planning for a large maintenance project. Lam et al. [8] applied ant colony optimization (ACO) to the problem of test sequence generation. Reitz [9] examined the use of an ant colony in building a software evolvability component model. Ayari et al. [10] used ACO to perform test input generation. Such applications are still lacking in requirements engineering.

These techniques may be more computationally complex than other techniques that have been widely applied in requirements engineering (such as information retrieval techniques for requirements tracing). However, it is quite possible that the techniques can be adapted in such a way as to still permit practical application. An example of method adaptation can be found in requirements tracing: the application of information retrieval techniques was adapted to take advantage of unique properties of the requirements engineering domain, namely small datasets, queries that may be dependent on each other, i.e., related queries known in advance, etc.

Similarly, we have examined the characteristics of the requirements engineering domain and have found that swarm techniques may be a good fit: small problem size or search space, decomposable problem, a certain degree of non-determinism, etc. Of specific applicability to requirements engineering is the emergent behavior feature of swarm intelligence methods. Emergent behavior is a byproduct of the main activity of an agent in a swarm. For example, an ant prefers a path with a stronger pheromone over one with a weaker pheromone. As a result of this behavior of an individual ant, the colony of ants can establish shorter routes.

Porting this idea to requirements engineering, we can model a swarm in such a way that an individual "software ant," with limited reasoning logic, limited knowledge of its environment, and limited awareness of its immediate surroundings, can discover a target item and bring the whole colony to it. Then, a group of "ants" can make a decision about the item in a collective manner. For example, consider the problem of classifying requirements (as non-functional, functional; or as high risk, low risk). We can provide some logic for the swarm to cluster the requirements by determining certain important characteristics of what makes a requirement functional/non-functional or low/high risk, thus adapting the agents to target requirements engineering problems of interest. This paper demonstrates such an adaptation to the problem of requirements tracing. The goal of this research is to build a prototype model for applying swarms to the requirements traceability problem and to evaluate its viability.

We choose requirements tracing as the first problem on which to demonstrate the swarm techniques as it is well known and there are well established benchmarks for acceptable performance of methods. There is prior work in information retrieval that successfully uses swarm techniques to rank retrieved information [11], an important aspect of requirements tracing.

In addition, the idea of term proximity and "phrasing" for requirements tracing expressed by Zou et al. [12] was inspirational. This idea can be examined from a different angle. Swarm agents can establish and promote a link between two textual documents based on the terms that are relatively close to each other, proximally, in both documents. In a sense, swarm agents will have awareness of their immediate surroundings and cast their votes based on the small segments of documents that are "visible" to them.

Similar ideas pertaining to 'lexical affinities' were expressed in work by Maarek et al. [13] and Niu and Easterbrook [14], each considering two term units within a single sentence. The 'lexical affinities' limit the neighborhood window to a maximum of 5 terms apart. In other words, terms occurring relatively close to each other in two documents form related phrases. The related phrases can be viewed as common segments creating a logical link between the documents.

This idea of using small common segments between two documents as a link appears to be a valid starting point for investigating swarm behavior on the traceability problem. The software ants traverse the search space, consisting of documents and common vocabulary, with limited knowledge about the environment. They have knowledge about the immediate surroundings, i.e., some local text segments. The software ants exchange their knowledge of these text segments through local interactions. To make the knowledge dissemination widespread among the colony members, a certain amount of non-determinism needs to be present in the search and discovery behavior of the software ants.

We start by introducing a simplified ant colony algorithm for tracing textual pairs of requirements artifacts and validating it on two sets of software requirements from real projects, comparing the results to those of a typical information retrieval (IR) tracing technique. In addition to the primary measures used in many tracing papers (recall, precision, F, F2), we also examine the quality of the techniques from the analyst perspective via the secondary measures mean average precision (MAP) and the difference between average similarity (DiffAR) [15]. We then introduce a more sophisticated ant colony technique that uses pheromone deposits, also validating it on two datasets. We found that the simple technique performs comparably

www.manaraa.com

to the IR method for the smaller dataset. We found that the pheromone technique enhanced the performance of the simplified ant colony algorithm [16] in some instances but decreased it in others. Both methods fared well in terms of DiffAR. Both methods performed well for MAP at low thresholds.

The paper is organized as follows. Section 2 provides background on ant colony optimization. Section 3 discusses requirements tracing. Section 4 presents related work. Section 5 discusses the two approaches to tracing using the ant colony algorithms. Section 6 presents the validation of the techniques on two datasets. Section 7 presents results and analysis. Section 8 concludes and discusses future work.

2 Ant colony optimization

Insects such as bees and ants, small and simple individually, can accomplish tremendous tasks in a collective effort. Of particular interest to computer scientists is that the insects' achievements and actions are all accomplished through local peer-to-peer interactions.

A number of scientists have studied the behavior of ants in foraging for food. Jean-Louis Deneubourg described a self-organizing behavior of ant colonies, where ants used pheromone communication [17]. The idea of using pheromone trails as a method of communicating through the environment is the heart of the ant colony optimization (ACO) algorithm [18]. This algorithm has been used in a number of computer science applications, such as the traveling salesperson problem, and has applicability to requirements engineering problems. We present the algorithm below.

Consider a graph G = (V, E) and a task of finding the shortest path between two nodes in the graph. For each edge between the nodes i and j in the graph, we assign a pheromone value τ_{ij}. In the initial step, the ACO will assign to each edge in the graph a zero pheromone value, $\tau_{ij}(0)$. Also, a group of «ants» $k = 1,\ldots,n$ is positioned at the source node.

For each iteration of the algorithm, each ant builds a path to the destination node. Also, at each node, each ant decides the next link to take. If ant k is at node i, the probability p of selecting the next node $j \in N_i^k$, which belongs to a set of nodes adjacent to i [18] is:

$$p_{ij}^{k}(t) = \begin{cases} \frac{\tau_{ij}^{\alpha}(t)}{\sum_{j \in N_i^k} \tau_{ij}^{\alpha}(t)} & \text{if } j \in N_i^k \\ & \text{or} \\ 0 & \text{if } j \notin N_i^k \end{cases}. \quad (1)$$

In the formula above, «α» is a parameter which amplifies the attractiveness of the pheromone trail.

In formula 1, when α and $\tau(t) = 1$, the algorithm simplifies down to navigating graph G in a non-deterministic fashion. The probability of selecting the next node *j* for ant *k* located in node *i* is the inverse of the number of nodes accessible from *i*.

3 Requirements tracing

Requirements tracing is defined as "the ability to describe and follow the life of a requirement, in both a forwards and backwards direction" [19]. A typical process used for requirements tracing of natural language artifacts, manual or automated, generally consists of a number of steps: document parsing, candidate link generation, candidate link evaluation, and traceability analysis [5]. For example, if a requirements document is being traced to a set of use cases, document parsing extracts elements from the two artifacts resulting in uniquely identified requirements elements and uniquely identified use case elements. At this point, a human analyst or tool establishes relationships or links between the elements, perhaps by selecting one requirement element and then performing string searches (using important words or terms in that element) into the collection of use case elements. If a tool is being used, it may be the case that the human analyst or tool assigns keywords to the requirements and use case elements and then performs keyword matching in order to generate what are called "candidate links".

Candidate link evaluation deals with assessing the links generated by a tool to ensure that they are correct. Traceability analysis, often performed manually by an analyst, deals with deciding if the high-level artifact has been "satisfied" by the lower level artifact, e.g., does this use case satisfy the given requirement? In this work, we concentrate on adapting the swarm technique to the candidate link generation problem.

3.1 Terminology

First, we define some terminology. The high- and low-level textual elements are called *documents*. Documents contain words or *terms*. The collection of all terms from all documents is called the *dictionary* or *vocabulary*. The collection of all terms in a document is called the document *corpus*. The *inverted index* lists all documents where a particular term occurs. *Term frequency* $TF_{t,d}$ is the count of how many times a particular term occurs in a document. *Inverse document frequency*, IDF_t, is a calculated value:

$$IDF_t = \log\left(\frac{N}{DF_t}\right), \quad (2)$$

where *N* is the total number of documents in the collection, and DF_t is document frequency, e.g., the number of documents where a given term occurs.

www.manaraa.com

To trace high-level textual elements (say from a requirements document) to low-level textual elements (say to a design document), we use swarm agents that traverse the collection of all documents and the vocabulary shared by the documents.

3.2 Measures

The tracing results are compared with an answer set of correct or "true" links, prepared by experts. Results are then evaluated using recall and precision [20].

Recall R is evaluated as the total number of relevant retrieved documents divided by the total number of relevant documents in the whole collection:

$$R = \frac{\#\text{of\_relevant\_retrieved}}{\#\text{\_relevant\_in\_collection}} \tag{3}$$

Precision P is evaluated as the total number of relevant retrieved documents divided by the total number of retrieved documents:

$$P = \frac{\#\text{of\_relevant\_retrieved}}{\#\text{\_retrieved}} \tag{4}$$

Precision and recall can be combined into a weighted harmonic mean:

$$F = \frac{(\beta^2 + 1)P * R}{\beta^2 P + R}, \quad \text{where } \beta^2 \in [0, \infty). \tag{5}$$

When $\beta^2 = 1$, precision and recall are balanced in the measure, this is called F_1 measure. When $\beta^2 = 2$, recall has more weight than precision, this is called F_2 measure.[1]

A number of researchers have posited the importance of evaluating candidate link list quality from the perspective of the analyst who must examine such a list (and make the final determination of whether a retrieved item is relevant or not) [5, 12, 15, 21]. Toward that end, we apply two additional measures DiffAR and MAP, sometimes referred to as *secondary* measures [15]. DiffAR is evaluated as the difference between the average similarity of the relevant matches (true positives) and non-relevant matches (false positives). DiffAR examines the internal structure of a candidate link list. Similar to how humans work with results returned from a web search, tracing analysts will examine the highest ranking elements in the candidate link list first. It is important that the relevance weights correctly separate true positives from false positives. Toward that end, DiffAR is the difference in the average relevance between true-positive links and false-positive links. Formally, it is defined as:

$$\text{DiffAR} = \frac{\sum_{(d,h)\in L_T} \text{sim}(d,h)}{|L_T|} - \frac{\sum_{(d',h')\in L_F} \text{sim}(d',h')}{|L_F|}, \tag{6}$$

where h and d belong to sets of textual artifacts $H = \{h_1, \ldots, h_n\}$ and $D = \{d_1, \ldots, d_k\}$; $L = \{(d, h) | sim(d, h)\}$ is a set of candidate links. L_T is a subset in L of true links; L_F is a subset in L of false links.

MAP measures "the quality across the recall levels" [20]. The higher the MAP, the closer the true links are to the top of the candidate link list. For h_j in a set of textual artifacts $H = \{h_1, \ldots, h_n\}$, a subset of relevant documents $\{d_1, \ldots, d_{m_j}\}$, *and* $L_{Jt} \in L = \{(d,h)|sim(d,h)\}$ *a subset of true links ranked by relevance,* MAP is evaluated as follows:

$$\text{MAP(H)} = \frac{1}{|H|} \sum_{j=1}^{|H|} \frac{1}{m_j} \sum_{k=1}^{m_j} \text{Precision}(L_{jT}) \tag{7}$$

A high value of MAP implies that true links are ranked higher in the list of the returned results.

Zou et al. [12] use average precision change (AP) to measure the internal quality of candidate link lists. It looks at a number of recall levels (such as 10% recall, 20% recall, etc.) and averages the precision change at each, thus, returning one value. We find MAP to be a better measure as it is more widely accepted in the IR community.

In summary, recall is a coverage measure, precision is a signal-to-noise ratio, F measure combines recall and precision, F_2 combines recall and precision with recall being valued over precision, DiffAR looks at relevance weights within candidate link lists, and MAP examines how precision changes in the internals of a candidate link list. The first four measures are important when evaluating traceability techniques from a researcher's perspective: how accurate is our tool and are we able to generate candidate link lists that are comparable in quality or better than existing methods? DiffAR and MAP examine the quality of the internal structure of candidate links, so this is from the perspective of a human analyst who must work with the lists. It is possible to generate candidate link lists that have high recall and precision, but have all the true positives listed in the lower half of the list. This will surely frustrate analysts as they look through highly ranked items that are NOT links. A better automated tool would retrieve high-recall and high-precision lists that ALSO have high DiffAR (the average relevance of true positives is higher than that of false positives) and high MAP (meaning that precision stays high at various recall levels (10, 20%, etc.) within the candidate link lists).

4 Related work

We address related work in the areas of traceability link generation and swarm techniques below.

[1] Note that F and F2 values reported in this paper differ from the RE 2010 conference paper due to a formula error (β^2 was not squared in the RE 2010 results but is here).

www.manaraa.com

4.1 Candidate link generation/text analysis

As mentioned in Sect. 3, candidate link generation is concerned with retrieving relevant elements from a given artifact pair. We focus on textual requirements artifacts.

Much work has been accomplished in applying information retrieval techniques to the candidate link generation problem. Antoniol et al. [22] used the vector space model (VSM) and a probabilistic model to recover traceability from source code modules to man pages and functional requirements. With VSM, they achieved the highest recall (100%) for the Albergate dataset by setting the threshold to 10% of the highest similarity measure, but only achieved precision of 11.98%.

Marcus and Maletic [23] applied latent semantic indexing (LSI) to the same datasets used by Antoniol et al. They achieved precision of 16.38% at 100% recall for the Albergate dataset. Hayes et al. [24] applied VSM with thesaurus to a dataset and compared this method to manual tracing and to a proprietary tool. They found that manual tracing resulted in higher precision (46%) than the proprietary tool (38.8%) or the VSM + thesaurus method (40.7%), but that the VSM + thesaurus method outperformed the other two approaches in terms of recall (85.4% compared with 43.9% for manual and 63.4% for the proprietary tool). Egyed et al. [25] found that it takes, on average, 1–2 min to manually recover code to requirements traces. They also found that recovery of method traces costs 3–6 times more than recovering class traces (also manually). Panis [26] surveyed 26 engineers at Teradyne and found that engineers prefer to see a traced requirement's content versus just an identifier. He found that engineers who depend on traceability information when creating documents most value it.

Zou et al. [27] examined ways to improve the precision of automated IR traces by using phrasing, obtaining improvements of almost 20% for one dataset when examining the Top 5% of the returned candidate links. Sundaram et al. [15] studied the effect of vocabulary base on traceability accuracy (using both artifacts versus just the low-level artifact to build the vocabulary) and found support for using only the low-level artifact. Zisman and Spanoudakis [28] examined ways to generate traceability links by applying rules to artifacts that had been tagged with parts of speech.

In general, the above techniques have been able to achieve excellent recall [24] but often at the expense of precision that is not acceptable or is only borderline acceptable. Our work differs in that it uses a greedy algorithm approach to generate candidate link lists; it does not require parts of speech tagging, phrasing, or specification of probabilities.

4.2 Swarm techniques

There are a number of researchers who have applied the particle swarm optimization (PSO) algorithm to the problem of analyzing textual documents. PSO is a direct search method for some optimal solution in a search space. The main characteristic of the PSO algorithm is that each member of the swarm adjusts its behavior based on the information obtained from its neighbors in the search space. The swarm agents are modeled to have a position in a search space and a velocity. The agents iteratively evaluate some fitness function where the agents' position and velocity are used as input parameters. The agents operate on the premise of their own "best" position and the swarm's and the neighbors' "best" position. The "best" implies a point in the search space where the fitness function has reached some optimal value [29].

Merwe and Engelbrecht applied data clustering using PSO on six different classification problems [30]. For example, 400 vectors were randomly created in two-dimensional space from the Wisconsin breast cancer database, with the objective of classifying data as representing benign or malignant tumors. Diaz-Aviles and Nejdl proposed a swarm ranking method for IR using particle swarm optimization on the benchmark database LETOR. The swarm first undertook a learning phase to rank IR results [11]. Cui et al. used PSO to cluster text documents [31].

In the above work, the researchers model the search space as a hyperspace of words or terms. The fitness function is, in some form or fashion, a Euclidian distance in the vector space of terms between the multidimensional points. The proposed vector space model treats each term as a dimension of the multidimensional space. For example, for data clustering, Merwe and Engelbrecht [30] used a variation of a distance vector to randomly seed centroid vectors, e.g., to seed some starting search points in the search space. The drawback of a VSM approach, in general, is that it treats terms as separate dimensions of the search space. Each new term increases the vector space dimension size and hence increases the complexity and number of necessary computations.

Diaz-Aviles and Nejdl [11] used training (learning) for a collection of queries and resulting retrieved documents. They used a training set as well as a validation set to attempt to reduce over-fitting. They proposed the method of *Swarm Ranking* to optimize the combination of content and links. The method used the mean average precision as the fitness function to evaluate the results. They found that the approach significantly outperformed standard approaches. Our method is similar in that we use a swarm algorithm to rank retrieved low-level requirement elements that may be relevant to a given high-level requirement. Our

www.manaraa.com

approach differs in that we do not take a semi- or supervised learning approach and thus do not require a training set.

Aghdam et al. [32] used ACO to select text features. In this work, the ACO algorithm did not have any a priori knowledge about the text features. Our method is similar in that it does not involve "learning" and in that the agents do not have predetermined knowledge about the space they traverse. The first method that we apply is a *simple* version of ACO. The second method uses pheromone deposits on the links and terms to influence the path selection behavior of a swarm agent. We use the phrase "*simple*" because there is no actual use of pheromones. The swarm agents are given freedom to operate on their own, determining the search path based on the environment, i.e., term frequency, weight, etc. The second method uses pheromone deposits on the links and terms to influence the path selection behavior of a swarm agent. Note that there is no "learning" or predetermined knowledge about the space being traversed. For both methods, the search and discover phase of the algorithm is "random roulette" and is greedy. The term and document frequencies of the text collection are used as guiding heuristics for agent behavior. Technically, the algorithm is still a swarm, but it is not as intelligent and cooperative as ACO. In the simple approach, the swarm agents are given freedom to operate on their own, determining the search path based on the environment, i.e., term frequency, weight, etc. In the second method that we apply, pheromones are deposited, but there is still no "learning" or predetermined knowledge about the space being traversed.

5 Methodology

Two techniques are presented in this paper. Preprocessing techniques that are common to both are discussed, followed by a more detailed look at each technique.

A swarm agent starts from a high-level textual document and follows a word or term that is present in the high-level document to a low-level element via the common vocabulary (the inverted index for the collection of both sets of documents).

First, the documents are parsed, stop words (words such as 'the' and 'of') are removed, and each remaining term is stemmed using Porter's algorithm [33]. Stemming is a fast technique to reduce terms to their stem such as 'comput-' for 'computer' and 'computing'. Term frequencies for each document and document frequencies for each term in the vocabulary are calculated. The VSM method, mentioned in Sect. 4.1, uses TF-IDF weighting schema (VSM using TF-IDF weighting is often called simply TF-IDF). Recalling the definitions for term frequency (TF) and inverse document frequency (IDF) from Sect. 3.1, we can say that TF-IDF assigns more weight to the most relevant terms within a document. The TF-IDF weight for each term is calculated using the following formula:

$$\mathrm{TF}-\mathrm{IDF}_{t,d} = \mathrm{TF}_{t,d} * \mathrm{IDF}_t. \quad (8)$$

The documents are then tagged as high- or low-level elements. The preprocessing also builds the inverted index. The constructed inverted index indicates not only the textual element associated with a given term, but also the type of the element: high or low. In applying the swarm of agents, each high-level element is assigned a fixed number of agents roughly equal to or greater than the number of low-level elements.

5.1 Simple swarm

We refer to the simple ant colony algorithm as simple swarm hereafter. The simple swarm technique is described as follows in Listing 1:

When all agents reach the low-level elements, we can determine candidate links. To establish and quantify candidate links, we need to count the number of agents that "made it" to the low-level elements, grouping them by their origin. The origin is the name of the high-level element from where the agents started their "journey". If a low-level element B has at least one agent that came from element A, we consider this "count" of at least one (1) as a potential candidate link between A and B. The candidate links for each high-level element are ordered by the count of the agents at the low-level elements. Agent counts are normalized to a value between 0 and 1, with the top low-level link for each high-level element having a value of 1. Links are filtered out at fixed threshold intervals to calculate recall and precision values at each cutoff threshold. Table 1 in "Appendix" lists the recall and precision values with regard to the cutoff threshold.

Figure 1 depicts the application of the algorithm to a small example (select terms were chosen for illustrative purposes). Assume that we have two high-level requirements Req1.txt and Req2.txt and use cases UC5.txt and UC8.txt:

Req1.txt: "The system shall support personal distribution lists".
Req2.txt: "The system shall be able to add a contact to the address list".
UC5.txt: "User edits personal distribution list by adding new contact".
UC8.txt: "List email contacts".

After preprocessing these elements, we determine that Req1.txt has the terms *personal, distribution,* and *list* and that Req2.txt has the terms *list, address,* and *contact*.

www.manaraa.com

Listing 1 Pseudo code for simple swarm

```
SIMPLE SWARM TRACELINKS (H, L)
    // Input High and Low level documents H and L
    // Output list of agent count (h,l,n) - from h in l, where n is the count
    1.  For each document h in high level collection H
    2.      // T = {t1, ....,tn} sorted terms in doc h
    3.      T ← h.Terms.sortBy(TFIDF)
    4.      For each agent s in swarm S
    5.          i ← Random[1.10]
    6.          t ← T[i]
    7.          // E is a record in the inverted index listing occurrences of
    8.          // term t in low level documents
    9.          E ← Vocabulary[t].LinksToLowLevelDocuments
    10.         E.sortBy(t.TermFrequency)
    11.         j ← Random[1.10]
    12.         e ← E[j]
    13.         e.countAdd(h,l)
    14.     EndFor
    15. EndFor
    16. For each document h in high level collection H
    17.     For each document l in low level collection L
    18.         list agent count from h in l
    19.     EndFor
    20. EndFor
```

Similarly, we know that the low-level element UC5 has the terms *edit*, *personal*, *distribution*, and *list* and that UC8 has the terms *contact*, *list*, and *email*. The inverted dictionary for the collection of all documents is used as the common vocabulary. The terms in the common vocabulary contain links pointing to the documents in which the terms are encountered. The vocabulary term links contain the term frequency count TF and a tag indicating if it is a high- or low-level element.

As the algorithm starts, a group of agents is assigned to Req1.txt. In the high-level element, the terms are then ordered by the TF-IDF weight of each term in the document. The agent randomly selects a term, for example, the agent may pick the term *personal*. The agent then "positions" itself in the common vocabulary at the term *personal*. The agent then inspects the links from the term *personal* to low-level elements. These links are also sorted in descending order by term frequency. The agent picks the next link to follow randomly from the top 10 or less candidate links. At the last leg of its journey, the agent arrives at the low-level element. At the end of this loop, the result composition will have all agents from all high-level documents located at the low-level elements.

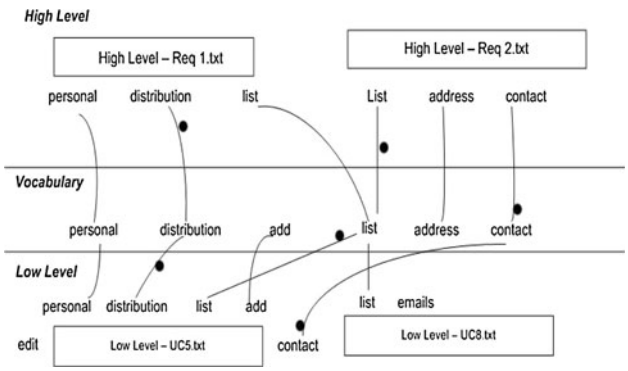

Fig. 1 Agents tracing links from high-level to low-level elements via vocabulary

A discussion of thresholds is in order. For the swarm method, candidate link lists are generated after applying a threshold filter varying from 0.1 to 0.9. The threshold indicates a percentage above which links are considered to be part of the candidate link list. For example, assume that one hundred agents starting from element Req1.txt traverse to documents UC1.txt, UC2.txt, UC3,txt, and UC4.txt, of which 50, 35, 10, 5 agents reach UC1.txt, UC2.txt, UC3.txt, and UC4.txt, respectively. If 0.7 is selected as the threshold, then only UC1.txt and UC2.txt are selected for the candidate link list (normalized values are 1, 0.7, 0.2, and 0.1, respectively).

The simple swarm method we tested used the TF-IDF weight and term frequency as the guiding heuristic for the agents. This version of the algorithm does not use any pheromones. Formula (1) is not applicable in its classical sense. This version of the algorithm appears to be a more focused version of TF-IDF. Nevertheless, the simple swarm is a stepping stone for the next method, pheromone swarm.

5.2 Pheromone swarm

The pheromone swarm method we tested used the TF-IDF weight amplified by log_2 of pheromone count on terms and links as the guiding heuristic for the agents. The distinction between the simple swarm method and the swarm with pheromone method lies in the selection of the terms and links by the swarm agents. A simple swarm agent is driven to consider, select, and focus on the most important terms in the document mostly at random (with some heuristic based on TF-IDF value of a term in a document). The agents in the pheromone swarm take into consideration pheromone deposits on the links and terms to choose the next step of their journey. This allows the pheromone swarm to search, discover, and guide swarm members to a target location via local interactions in the search space. The agent's decision on what term to select or what path to take is influenced by presence of pheromone markings on the inspected object, e.g., terms or links. For example, when an agent starts from a high-level document, the agent has a higher chance of selecting a term; if the term has some pheromone markings. The pheromone markings on a term in a high-level document indicate an established fact that this particular term is a neighbor to some other term in some low-level document. This idea of marking the neighbors and selected terms is based on treating textual documents as collections of "phrases" rather than as "bags

www.manaraa.com

of words." A similar idea was expressed by Zou et al. [12], where the authors focused on "two word phrases." Our approach is different in a sense; we allow phrases to be loosely defined in a neighborhood of a linking term.

The swarm with pheromones technique is described as follows in Listing 2:

Once all agents reach the low-level elements, they stay there. The pheromone deposit can spread further up the graph to the terms. We use the same methodology to generate candidate links using the cutoff threshold.

As the algorithm starts, a group of agents is assigned to Req1.txt. In the high-level element, the terms are then ordered by the product of TF-IDF weight and log_2(Pheromone $count_{Term} + 1$) in the document. To smooth the influence of the pheromone count on terms, we use a smoothing technique by taking a $log_2(count + 1)$ of the nonzero count. Related to Formula 1, the parameter t is a sequential order in which agent s was released (Listing 2, line 1). The amplification constant $\alpha = 1$. The agent randomly selects a term, say, from the top ten sorted terms. Going back to our original example, the agent may pick the term *personal*. The agent then "positions" itself in the common vocabulary at the term *personal.* Then, the agent inspects the links from the term *personal* to low-level elements. In this algorithm, the links may contain pheromone deposits. The pheromone deposits on the link serve as attractors for the agents for path selection. The pheromone deposits on the links indicate that there is another agent at the low-level document that came from a particular high-level document. Furthermore, the residing agent in the low-level document is in the neighborhood of the term *personal*. If the source document of the residing agent is Req1.txt, then our current agent will have a higher probability of selecting this pheromone-marked link. Once a link to the low-level document has been selected, the agent crawls down to a low-level element. There, the agent diffuses the pheromones on the neighbors of the linking term. These pheromone deposits will attract "future" agents traveling from the Req1.txt high-level document.

Swarm agents can be instructed to deposit the pheromones in the low-level documents beyond the immediate neighbors. To indicate how far the pheromones are deposited, we use a delta value. When delta is equal to one, we deposit the pheromones on the immediate neighbors. When we set the delta to 3, the agents deposit the pheromones up to three neighbors to the left and right of the linking term in the low-level document. When the delta is set to 5, five neighbors on either side of the linking term receive pheromone deposits. If the linking term is at the end or beginning of a document, and there are no "next 3 neighbors" on the right or left, only the present side of the linking term's neighborhood receives pheromone deposits.

The pheromone swarm algorithm is complexity of $O(N^3 \log N)$. The algorithm has to iterate through each swarm agent, each high-level document, and sort term within high-level document by weight and pheromone deposit. A possible speed up can be achieved by using a smaller size swarm.

In order to validate the approach, we applied it to two sets of requirements from software systems. The study design and threats to validity are presented in the next section.

Listing 2 Pseudo code for simple swarm

```
PHEROMONE SWARM TRACELINKS (H, L)
    // Input High and Low level documents H and L
    // Output list of agent count (h,l,n) - from h in l, where n is the count

1.  For each agent s in swarm S
2.      For each document h in high level collection H
3.          // T = {t1, ....,tn} sorted terms in doc h
4.          T ← h.Terms.sortBy (TFIDF, PheromoneCount)
5.          i ← Random[1.10]
6.          t ← T[i]
7.          E ← Vocabulary[t].LinksToLowLevelDocuments
8.          E.sortBy (tTermFrequency, PheromoneCount);
9.          j ← Random[1.10]
10.         e ← E[j]
11.         N ← l.neighborsOf(t)
12.         For each neighbor n of t
13.             Vocabulary[n].link[e].addPheromone[h]
14.             if (Vocabulary[n].links.Contain[h]) and
15.               (h.Terms[n].isNeighborOf(t)) then
16.                   h.Terms[n].addPheromone()
17.         EndFor
18.     EndFor
19. EndFor
20. For each document h in high level collection H
21.     For each document l in low level collection L
22.         list agents from h in l
23.     EndFor
24. EndFor
```

6 Evaluation

This section will present the study design as well as threats to validity.

6.1 Study design

Requirements from two projects were used for the study. The first project is Pine [34], a text-based email system developed by the University of Washington. This project consists of a requirements document (49 requirements) and a set of textual use cases (51). There are 246 true links in the answer set. Development of answer sets will be discussed further in the next section. The second project is CM1, a NASA scientific instrument [35]. The project consists of a complete requirements document (235 requirements) and a complete design document (220 design elements). There are 361 true links in the answer set.

www.manaraa.com

The independent variable in the study is the method (TF-IDF, simple swarm, pheromone swarm). The dependent variables are recall, precision, F, F2, DiffAR, and MAP. The 11-point interpolated precision–recall graph is used to evaluate the statistical significance of the results (sign test). In addition, the Wilcoxon signed-ranks test is applied to the DiffAR and MAP results to test for significance at the 0.05 level. In cases where the number of queries returned with relevant links is different, the Mann–Whitney *U* test is used instead of the Wilcoxon.

The study was conducted as follows for the simple swarm method. After preprocessing, each high-level element was selected one at a time and the simple swarm method was applied. The output was captured as a candidate RTM and was compared with the answer set. We captured information on the number of true links identified, true negatives, false positives, and false negatives. From this, we calculated recall and precision (as discussed in Sect. 3). The F, F2, DiffAR, and MAP measures were calculated as well. We then compared these measures to those obtained for the same datasets using the Vector Space Model with TF-IDF weighting (called TF-IDF hereafter).

The pheromone study was identical to the above except that all agents for all high-level elements are released at the same time versus one at a time.

6.2 Threats to validity

Threats to conclusion validity threaten the ability to draw the correct conclusions from the study results. Using two datasets and applying similar treatments, we address the reliability of the treatment implementation. Both datasets were analyzed using the simple swarm and pheromone swarm methods, with the same list of delta values: 1, 3, and 5.[2]

There is a possible threat to internal validity due to experimenter bias. We reduced this threat by using datasets for which answer sets have been independently verified by more than one analyst and in some cases more than one research group (CM1). There was a potential for bias though in that the answer sets were created by human analysts that are familiar with the traceability research domain. We also used a vetted tool, RETRO [36], and adapted it in order to implement the swarm techniques. There is a possible threat to internal validity due to repeated testing. The swarm methods randomly select links to follow. To mitigate this threat, we ran each method ten times and examined the mean recall and precision values. Each method produced average recall and precision values with variances ranging from 0.003 to 0.06. Variance decreased significantly as threshold values increased.

We reduced threats to construct validity by using standard information retrieval measures to evaluate effectiveness, such as recall, precision, F and F2 measures, as well as MAP. A further threat involves the parameters used by the swarm methods. To enhance the selection method of a "next" linking term, we chose $\log_2$ of pheromone count on a given term. This $\log_2$ provides a steady slow increase in the importance of the weight of a term with pheromone deposit, in contrast to simple counts of pheromone deposits.

Threats to external validity deal with whether the results can be generalized. The study used two datasets for validation. Though both datasets are real projects (not student projects), one of the datasets is relatively small (49×51). Also, though the datasets do represent two different domains, it is not possible to state that the study sufficiently validated all domains or all projects [36].

7 Results and analysis

In this section, we present the results for the two swarm methods and the TF-IDF method on the Pine and CM1 datasets. In Sects. 7.1–7.3, we evaluate the two swarm methods on the Pine dataset using the primary measures of recall, precision, F, and F2 as well as secondary measures for the Pine dataset. In Sects. 7.4–7.6, we evaluate the two swarm methods on the CM1 dataset, along with a discussion on secondary measures. Section 7.7 provides an overall summary of the results. Data points for the figures presented in this section are presented in Table 1 of "Appendix".

7.1 Simple swarm applied to the Pine dataset

The results for the simple swarm method are presented using the measures from Sect. 3.2. Overall precision and recall are examined first, followed by F and F2 measures.[3]

Figure 2 presents the 11-point interpolated precision–recall curve for the simple swarm and TF-IDF methods on the Pine dataset. Simple swarm has higher precision than TF-IDF at 6 out of the 11 recall points, with most of the points near the middle to high end of recall. The difference

[2] We used delta ranging from 1 to 5. We present the results for 1, 3, and 5 only. The results corresponding to delta of 2 and 4 followed a linear trend and fit between the selected values.

[3] Note that the RE 2010 paper presented average precision and average recall for the swarm and TF-IDF methods (which was mislabeled as overall precision and recall). The results in this paper use overall precision and recall for all methods.

www.manaraa.com

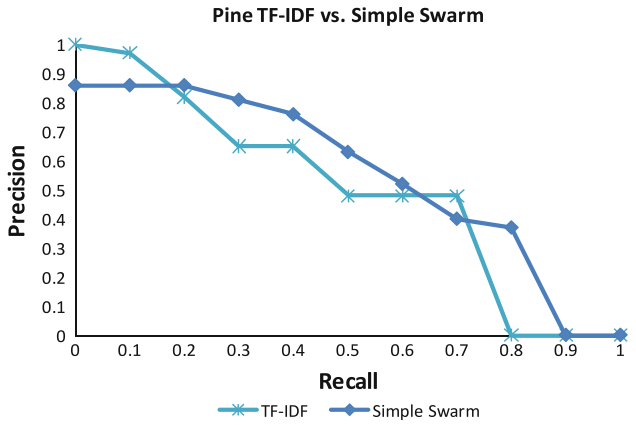

Fig. 2 11-point interpolated precision–recall curve for TF-IDF and simple swarm for the Pine dataset

in precision however is not statistically significant using the sign test.

Figure 3 depicts the F and F2 measures for both simple swarm and TF-IDF methods on the Pine dataset across the different thresholds. This figure presents a different view of how the two methods performed when threshold filtering is applied. F and F2 values for TF-IDF start off high but degrade as threshold values increase. Simple swarm F and F2 values, on the other hand, do not degrade as quickly as TF-IDF, performing best between threshold values of 0.2 and 0.4. Figure 3 also shows that simple swarm has a more consistent recall/precision tradeoff compared with TF-IDF when using threshold filtering. While TF-IDF shows a consistent decline in performance as threshold values increase, simple swarm shows an increase in performance at lower threshold values before degrading at a slower rate compared with TF-IDF as threshold values increase. This behavior can be explained by the fact that agents tend to gather around a smaller subset of elements as threshold values increase. The simple swarm method "directs" each swarm agent to consider and focus on the most important terms in the document, allowing agents to perform a more focused search. After passing an optimum threshold, agents start missing correct targets, e.g., low-level elements that are part of the correct links to the high-level element from which the agents started the journey.

Another explanation for the difference in F and F2 measure behavior between TF-IDF and simple swarm is how each link's weight is calculated. TF-IDF link weights are measures of cosine similarity between the weighted keyword vectors of two documents [20]. Link weights above 0.8 are uncommon using TF-IDF due to the fact that multiplication of two numbers between 0 and 1. Swarm methods, on the other hand, calculate link weights by dividing each link's agent count by the largest agent count. Using this method, the top-most link always has a weight of 1. The difference in how weights are calculated does not prevent the methods from being compared appropriately as links are filtered using the same threshold values for both methods. The difference in F and F2 behavior indicates that TF-IDF achieves peak scores at lower threshold values compared with swarm. Both methods achieved comparable peak F and F2 values at different threshold values, e.g. TF-IDF at 0.1 and simple swarm at 0.2 and 0.4.

Pheromone swarm precision deteriorates below the 0.2 threshold but still remains near the 0.9 range.

Figure 4 presents the 11-point interpolated precision–recall curve for the pheromone swarm and TF-IDF methods on the Pine dataset. Pheromone swarm gains a slightly higher precision than TF-IDF at several points for various delta values. The difference in precision, however, is not statistically significant using the sign test.

Figure 5 depicts the graph of the F measure for TF-IDF and pheromone swarm. Peak F values for pheromone swarm delta = 1 and delta = 3 are comparable to the TF-IDF Peak F value, e.g., 0.58, 0.56, 0.58, respectively. Pheromone swarm does not exhibit the same F/F2 trend as simple swarm when threshold values increase. The decrease in F values for the pheromone swarm is still slower than TF-IDF, indicating that the precision/recall tradeoff does not decrease as fast with each increasing threshold value.

Figure 6 depicts the graph of the F2 measure for TF-IDF and pheromone swarm for the Pine dataset. The trend in the F2 graph is similar to Fig. 5, with TF-IDF outperforming pheromone swarm 0.66–0.61, respectively, at the 0.1 threshold. Even so, the recall/precision tradeoff is still slower compared with TF-IDF.

7.2 Secondary measures for the Pine dataset

Figure 7 shows DiffAR performance for simple swarm, pheromone swarm, and TF-IDF methods. All swarm methods have consistently higher DiffAR values compared with TF-IDF. Simple swarm performed the best among all methods, with DiffAR going from 0.41 to 0.93 as threshold values increase. This suggests that link weights from swarm methods correlate more with link correctness. Achieving higher DiffAR represents work that is less frustrating for human analysts, who must ultimately vet all candidate links to form the final traceability matrix.

Figure 8 plots MAP versus Recall for the simple swarm, pheromone swarm, and TF-IDF methods. The simple swarm method returns more correct links at higher MAP with the first three thresholds compared with all the other swarm methods. Compared with TF-IDF at the 0.1 threshold, simple swarm achieved 0.76 MAP at 0.86 recall while TF-IDF achieved 0.75 MAP at 0.72 recall.

www.manaraa.com

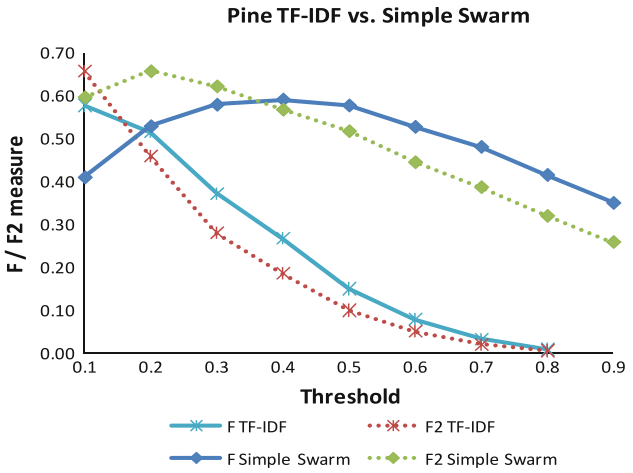

Fig. 3 F and F2 measures for TF-IDF and simple swarm on the Pine dataset

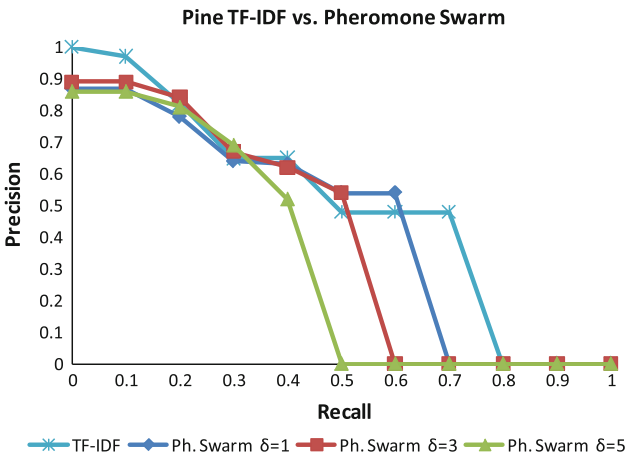

Fig. 4 11-point interpolated precision-recall curve for pheromone swarm and TF-IDF for the Pine dataset

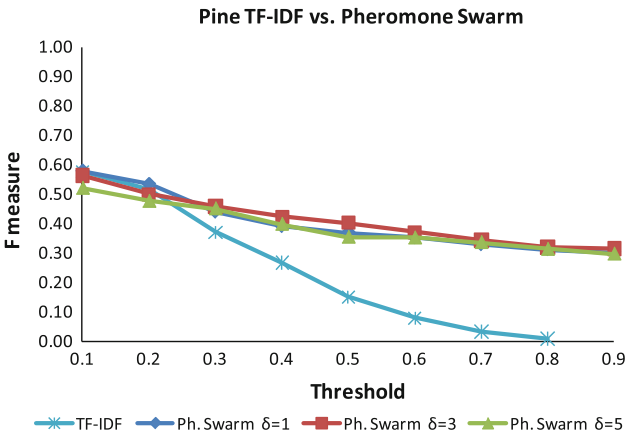

Fig. 5 F measure for TF-IDF and pheromone swarm for the Pine dataset

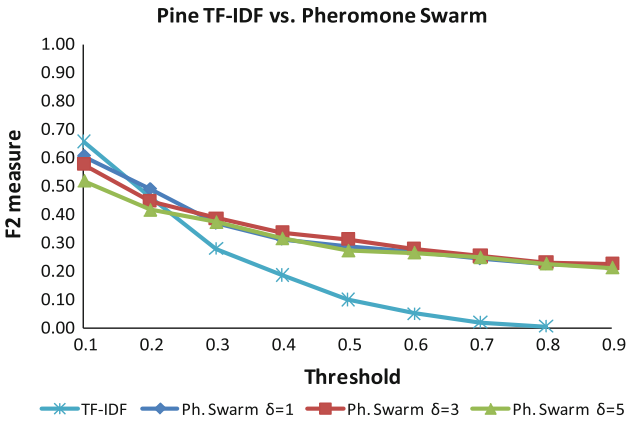

Fig. 6 F2 measure for TF-IDF and pheromone swarm for the Pine dataset

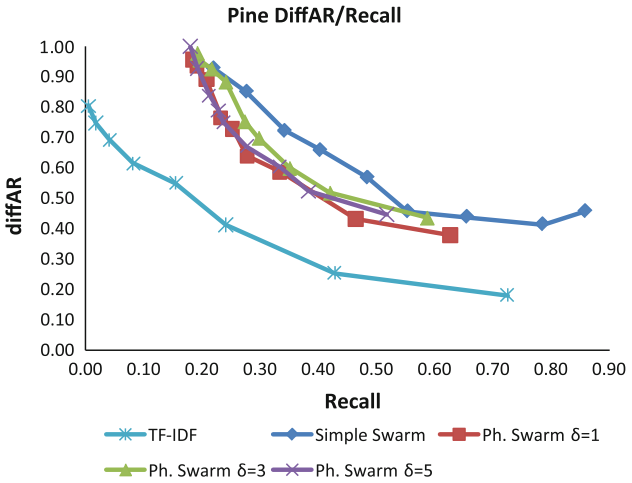

Fig. 7 DiffAR versus recall for simple swarm, pheromone swarm, and TF-IDF for the Pine dataset

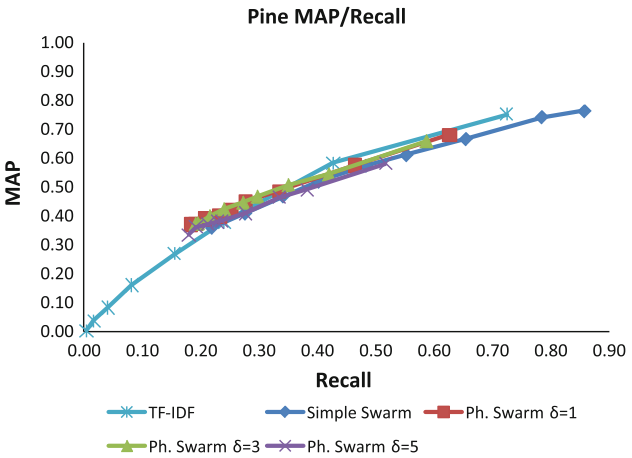

Fig. 8 MAP versus recall for simple swarm, pheromone swarm, and TF-IDF for the Pine dataset

www.manaraa.com

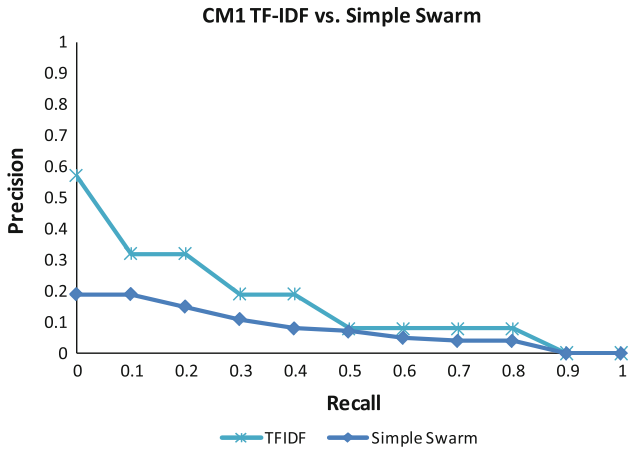

Fig. 9 11-point interpolated precision–recall *curve* for the simple swarm and TF-IDF methods on the CM1 dataset

7.3 Simple swarm on CM1 dataset

Next, we examine the results for the CM1 dataset. Figure 9 shows the recall and precision values for the simple swarm and TF-IDF methods. Note that precision values for this dataset are significantly lower than Pine due to the larger size of the dataset. This is a common phenomenon for IR methods (that larger datasets yield smaller precision values).

The precision/recall tradeoff between the two methods is slightly different than the tradeoff seen in the Pine dataset. Precision increases slowly when recall decreases, e.g., for simple swarm, precision only increases from 0.04 to 0.07 while recall drops from 0.8 to 0.5. This indicates that simple swarm agents are not picking the correct low-level elements as threshold values increase. It is apparent that search options given to the swarm agents restricted their options to explore and directed them to a limited number of low-level elements.

Figure 10 shows the F and F2 measures for the simple swarm and TF-IDF methods. The F and F2 measurement for simple swarm on CM1 did not exceed 0.25. Note that the F measure for simple swarm did not change significantly; it varied from 0.15 to 0.24. TF-IDF achieved a peak F value of 0.28 and peak F2 value of 0.37, significantly outperforming simple swarm. For CM1, the TF-IDF method performed better than simple swarm for both F and F2 measurements. TF-IDF performed best at the 0.2 threshold value, while simple swarm performed best at the 0.8 threshold for F and the 0.5 threshold for F2. Precision for simple swarm ranged from 0.04 to 0.19, contributing to the low F/F2 values and indicating that the two document levels contained many "coincidental matches"; that is to say, even if the elements contained many similar terms, they were not necessarily classified as true links in the answer set.

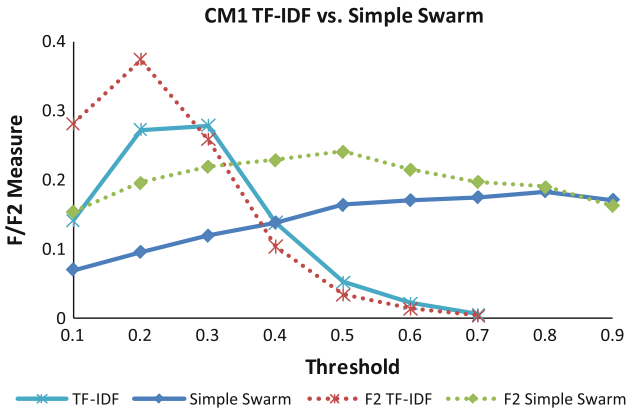

Fig. 10 F and F2 for the simple swarm and TF-IDF methods on the CM1 dataset

7.4 Pheromone swarm on the CM1 dataset

Figure 11 shows the precision–recall curve for the pheromone swarm and TF-IDF methods where agents deposit the pheromones up to 1, 3, and 5 neighbors away, e.g., delta = 1, 3, and 5. Pheromone swarm performs worse at almost all recall points except for 0.5 recall where pheromone swarm delta = 1 and 3 ties with TF-IDF. Note that delta does not have much of an effect on precision for most of the recall points.

Simple swarm in Fig. 9 performs comparably with pheromone swarm, with pheromone swarm method yielding a more focused search compared with simple swarm. The pheromone enabled swarm agents still narrow their options to explore (compared with the simple swarm), yet this time the selection of the candidate links is done a bit more accurately.

Figure 12 shows the F and F2 measures for the pheromone swarm and TF-IDF methods. The F measurement stayed under 0.19; at the same time, F2 reached 0.26 at the

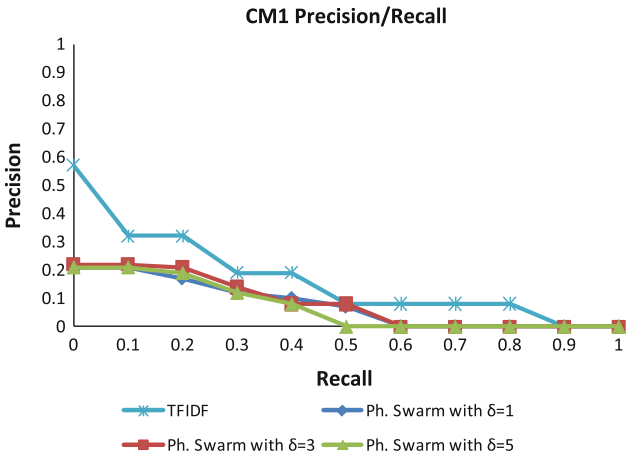

Fig. 11 11-point interpolated recall–precision curve for pheromone swarm, delta = 1, 3, 5, and the TF-IDF methods for the CM1 dataset

www.manaraa.com

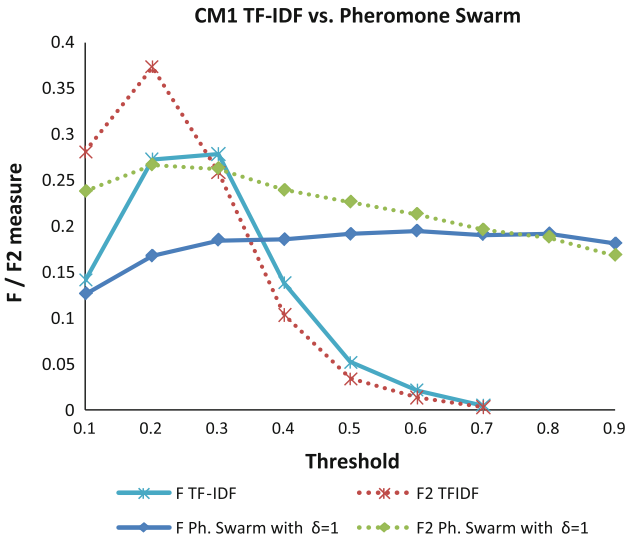

Fig. 12 F and F2 measures for pheromone swarm, delta = 1, and TF-IDF methods for the CM1 dataset

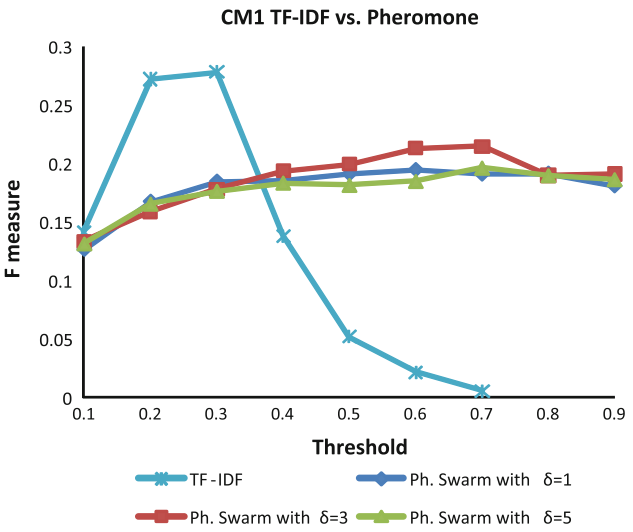

Fig. 13 F measure for the pheromone swarm, delta = 1, 3, 5, and the TF-IDF methods for the CM1 dataset

threshold value of 0.3. Note that the F measure mostly remained in the narrow "corridor" between 0.12 and 0.19. The "corridor" of F2 values was between 0.17 and 0.26 in the CM1 dataset. TF-IDF outperforms pheromone swarm, with similar results compared with simple swarm, although, the peak F2 value for pheromone swarm is at the 0.2 threshold.

Figures 13 and 14 show the F and F2 measures for the TF-IDF and pheromone swarm methods for CM1. F measure for pheromone swarm increases slowly with each threshold increase while F2 measure slowly decreases instead. Pheromone swarm delta = 3 seems to perform better than the other two delta values, achieving peak F value of 0.20 and peak F2 value of 0.25. Expanding the

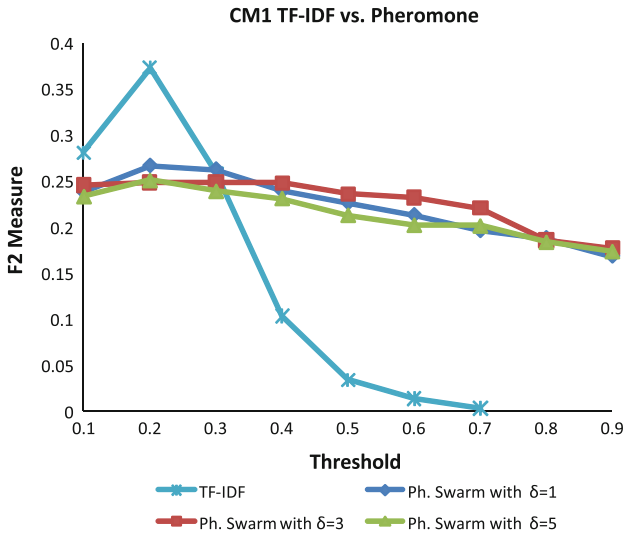

Fig. 14 F2 measure for the pheromone swarm, delta = 1, 3, 5, and the TF-IDF methods for the CM1 dataset

pheromone affected neighborhood does not seem to improve the performance of the method.

7.5 Secondary measures for the CM1 dataset

Figure 15 shows DiffAR performance for simple swarm, pheromone swarm, and TF-IDF methods. Similar to Pine, all swarm methods have higher DiffAR values compared with TF-IDF. All swarm methods performed about the same, with simple swarm performing worse between threshold values of 0.1–0.3.

Figure 16 plots MAP versus Recall for the simple swarm, pheromone swarm, and TF-IDF methods. Simple swarm performed better than TF-IDF at the 0.1–0.3 threshold. Pheromone swarm delta = 3 also performed better than TF-IDF at the 0.1 threshold. Pheromone swarm delta = 1 performs worse than TF-IDF, but as delta increases, performance is comparable to TF-IDF. Note, however, that MAP is still quite low at 0.23, indicating that, on average, each document (high-level element) has an average precision of 23%.

7.6 Overall summary

Though the swarm link weights in TF-IDF are not calculated the same, they serve a similar role, i.e., they serve as a basis for filtering the candidate links. The higher the filter (a close cosine similarity in documents in TF-IDF or a higher agent count in swarm methods), the more the F values decrease for TF-IDF and swarm methods on both datasets.

Figure 3 shows that F values for TF-IDF perform better than simple swarm below threshold values of 0.2 on the

www.manaraa.com

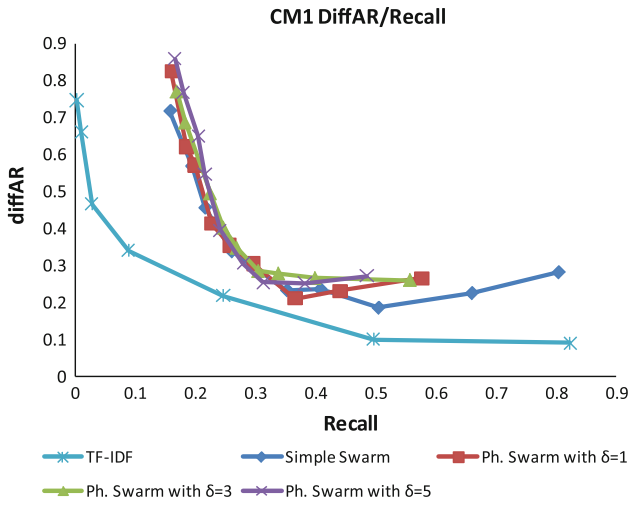

Fig. 15 DiffAR versus recall for simple swarm, pheromone swarm, and TF-IDF methods on the CM1 dataset

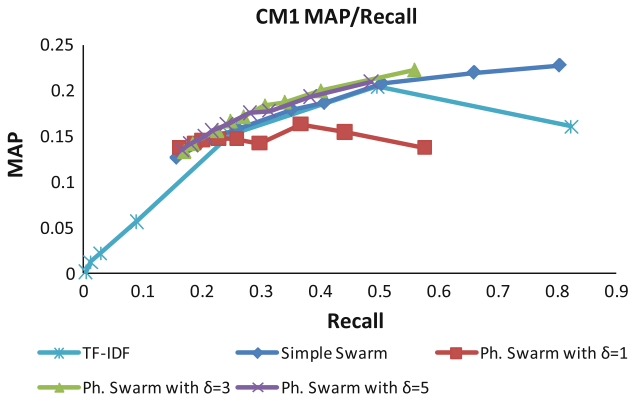

Fig. 16 MAP versus recall for the simple swarm, pheromone swarm, and TF-IDF methods on CM1

Pine dataset. After the threshold is increased, the swarm's (with/without pheromones) F values are better than TF-IDF as seen in Figs. 5 and 6. Furthermore, TF-IDF exhibits a steep decline in F and F2 as threshold values increase. Swarms demonstrate better values for F measurements for higher threshold values.

Figure 10 shows better performance for the TF-IDF method than simple swarm on the CM1 dataset, achieving 0.28 for F and 0.37 for F2. Simple swarm performs better than TF-IDF past the 0.4 threshold.

Pheromone swarm, in general, performed better than simple swarm on the CM1 dataset. Pheromone swarm with delta = 3 reached the highest value for F of 0.21 at the 0.6 threshold. Furthermore, pheromone swarm exhibited a gradual increase in F value as the threshold increased. TF-IDF reached its peak F value of 0.28 at the 0.3 threshold and then declined sharply as threshold values increased. The same trend is observed with TF-IDF in the CM1 and Pine datasets for the F measurement. The F2 values for the swarm methods exhibited a slightly different behavior. F2 values slowly declined as the threshold increased for all swarm methods. Even in these instances, the swarms displayed a more gradual change in performance as the threshold increased. Pheromone swarm F2 values gradually decreased from 0.25 to 0.18.

In summary, the simple swarm approach showed some advantage over the TF-IDF method on the Pine dataset, yet it did not fare as well on CM1. At the same time, with pheromone swarm, any advantage visible on the Pine dataset was lost. Pheromone swarm performance on the CM1 set improved over simple swarm but still underperformed TF-IDF. A possible explanation for this is the way that the high and low elements are connected. The Pine dataset contains 49 high-level and 51 low-level elements, with 2,499 possible links. The CM1 dataset contains 235 high and 220 low elements, creating a search space of 51,700 possible candidate links. The answer set for the Pine dataset has 246 links, about 10% of all possible links. In the CM1 dataset, the ratio of true links over possible links goes down to less than 1% (361 true links divided by 51,700). CM1 also uses a significant amount of technical terms and acronyms, causing the swarm agents to end up at incorrect low-level elements.

It appears that in a compact dataset such as Pine the pheromones make the agents "over-choose" certain links. This leads to lower starting recall and higher precision as seen in Fig. 4. CM1, on the other hand, the more focused selection in a sparsely linked set, delivers better precision than simple swarm. Agents get to pick proper links based on the pheromone markings previously deposited by other agents.

For the CM1 dataset, the MAP measurements exhibited some variance with regard to the pheromone swarm method. Pheromone swarm at delta = 3 performed just "above" the TF-IDF and all other swarm methods. As we saw earlier, in CM1 dataset, increasing the affected neighborhood delivered some performance gains up a certain point using delta = 3. Simple swarm had better MAP at lower thresholds for both datasets.

Another interesting result we observed was related to the size of the neighborhood of a linking term. When we increased the delta from 3 to 5 for the pheromone swarm, we noticed a slight drop in performance across all measurements and both datasets. Apparently, by depositing pheromones on neighbors that are "too remote", the agent introduces too much noise for future agents. For example, on the CM1 dataset with delta = 3, the starting recall and precision values were 56 and 8%, respectively. When we increase the delta to 5, i.e., 5 neighboring terms on either side of the linking term received deposits, the starting recall and precision became 48 and 8%, respectively. Maarek et al. [13] and Niu and Easterbrook [14] experimented with a neighborhood of size five (5) using 'lexical affinities'. Our work differs from 'lexical affinities' in

www.manaraa.com

several ways. Unlike 'lexical affinities,' the swarms consider neighbors that may cross sentence boundaries. 'Lexical affinities' picks up two word units, whereas the swarm considers all terms within the limits of the inspected neighborhood. That may explain why we obtained an optimal neighborhood of three (3) as opposed to five (as in 'lexical affinities'). To achieve high-recall and high-precision results for the CM1 dataset, the collection of candidate links has to be tightly focused and highly precise. The use of a thesaurus might have directed the swarm agents to the proper document. In addition, a method of handling acronyms might assist. In this case, the thesaurus may become project specific.

In the case of TF-IDF at low threshold values, the method considers a greater number of the low-level elements as possible candidate links, thus yielding higher recall at the cost of precision. The swarm method, a more focused approach than TF-IDF, limited the "discovery horizon" for the agents by focusing on the top terms in a textual element, hence limiting the possible search alternatives.

8 Conclusions and future work

The paper presents two swarm methods in support of requirements tracing: simple swarm and pheromone swarm. The methods showed mixed results for the Pine dataset, achieving recall of 78% with precision of 40% at a 0.2 threshold and slightly outperforming TF-IDF with 72% recall and 48% precision at a 0.1 threshold. At the same time, limitations with the simple swarm method were discovered on the CM1 dataset.

The pheromone swarm displayed an improvement on the sparsely connected CM1 dataset. The variations of delta value (size of the affected neighborhood) implied the importance of the linked term proximity in influencing the agent behavior. A larger neighborhood in which to deposit pheromones does not imply better search results.

The swarm methods performed better than TF-IDF for the DiffAR measure for both datasets. This indicates that human analysts who must vet candidate links will find swarm links less frustrating to examine, as the higher link weights correlate with correctness.

As mentioned earlier in the paper, these shortcomings may be a result of the limitations imposed on the swarm agent search behavior. As the TF-IDF method compares documents based on term similarity, it treats a document as a "bag of words" compared with another document, i.e., another "bag of words". The similarity between two "bags" is evaluated through the common terms. For the swarm methods, we also used common terms to establish links between documents. We noticed that F and F2 measurements for the swarm methods were higher above the 0.4 threshold. This implies that a subset of all common terms between two documents can be used to establish a true link. This observation reiterates the reasoning for enabling the swarm to concentrate on the top terms in a document, rather than exploring all of them. The neighborhood size variation suggests that one should consider the term proximity of the linked term to determine candidate links between two documents.

The goal of this work is to build an initial model for applying swarms to the requirements traceability problem and to show its viability. The initial findings, though mixed, urge us to experiment further with the method. We plan to expand this work by using a thesaurus, permitting the agents to discover links not only through a single term but through term synonyms. We also plan to address the use of term proximity as a possible attributing factor for agent search behavior. In light of the observations made that between 20 and 40% of common terms contribute the highest level of true candidate links, it would be interesting to see whether we can achieve the same results using a smaller swarm population. A smaller population size reduces the computation time. Another idea is to use more than one type of pheromone, just like real ants do in nature [37]. The expanded pheromone vocabulary may lead to a greater variety in agent responses and search strategies.

Acknowledgments This work is funded in part by the National Science Foundation under NSF grant CCF-0811140.

Appendix: Results for Pine and CM1 datasets

This Appendix provides results for the experiments run on the Pine and CM1 datasets. Table 1 lists each Method under each dataset with columns for Threshold, Recall, precision, F, F2, DiffAR, and MAP. Table 2 presents statistical analysis of the secondary measures.

Table 1 Detailed results for the TF-IDF, simple swarm, and pheromone swarm methods on the Pine and CM1 dataset

TF-IDF							TF-IDF						
Threshold	Recall	Precision	F	F2	DiffAR	MAP	Threshold	Recall	Precision	F	F2	DiffAR	MAP
Pine							*CM-1*						
0.1	0.72	0.48	0.58	0.66	0.18	0.75	0.1	0.82	0.08	0.14	0.28	0.09	0.16
0.2	0.43	0.65	0.51	0.46	0.25	0.58	0.2	0.50	0.19	0.27	0.37	0.10	0.20

www.manaraa.com

Table 1 continued

TF-IDF							TF-IDF						
Threshold	Recall	Precision	F	F2	DiffAR	MAP	Threshold	Recall	Precision	F	F2	DiffAR	MAP
0.3	0.24	0.82	0.37	0.28	0.41	0.38	0.3	0.25	0.32	0.28	0.26	0.22	0.15
0.4	0.15	0.97	0.27	0.19	0.55	0.27	0.4	0.09	0.31	0.14	0.10	0.34	0.06
0.5	0.08	0.95	0.15	0.10	0.61	0.16	0.5	0.03	0.45	0.05	0.03	0.47	0.02
0.6	0.04	1.00	0.08	0.05	0.69	0.08	0.6	0.01	0.57	0.02	0.01	0.66	0.01
0.7	0.02	1.00	0.03	0.02	0.75	0.04	0.7	0.00	0.50	0.01	0.00	0.75	0.00
0.8	0.00	1.00	0.01	0.01	0.80	0.00							
Simple swarm							*Simple swarm*						
0.1	0.86	0.27	0.41	0.60	0.46	0.76	0.1	0.80	0.04	0.07	0.15	0.28	0.23
0.2	0.78	0.40	0.53	0.66	0.41	0.74	0.2	0.66	0.05	0.10	0.20	0.23	0.22
0.3	0.65	0.52	0.58	0.62	0.44	0.67	0.3	0.50	0.07	0.12	0.22	0.19	0.21
0.4	0.55	0.63	0.59	0.57	0.46	0.61	0.4	0.41	0.08	0.14	0.23	0.24	0.19
0.5	0.48	0.71	0.58	0.52	0.57	0.57	0.5	0.35	0.11	0.16	0.24	0.23	0.18
0.6	0.40	0.76	0.53	0.44	0.66	0.52	0.6	0.26	0.13	0.17	0.22	0.34	0.16
0.7	0.34	0.81	0.48	0.39	0.72	0.47	0.7	0.22	0.15	0.17	0.20	0.46	0.15
0.8	0.28	0.83	0.41	0.32	0.85	0.41	0.8	0.19	0.17	0.18	0.19	0.57	0.14
0.9	0.22	0.86	0.35	0.26	0.93	0.36	0.9	0.16	0.19	0.17	0.16	0.72	0.13
Pheromone swarm $\delta = 1$							*Pheromone swarm with* $\delta = 1$						
0.1	0.63	0.54	0.58	0.61	0.38	0.68	0.1	0.58	0.07	0.13	0.24	0.27	0.14
0.2	0.46	0.63	0.53	0.49	0.43	0.57	0.2	0.44	0.10	0.17	0.27	0.23	0.15
0.3	0.33	0.64	0.44	0.37	0.58	0.48	0.3	0.37	0.12	0.18	0.26	0.21	0.16
0.4	0.28	0.66	0.39	0.31	0.64	0.45	0.4	0.30	0.13	0.19	0.24	0.31	0.14
0.5	0.25	0.69	0.37	0.29	0.73	0.42	0.5	0.26	0.15	0.19	0.23	0.35	0.15
0.6	0.23	0.73	0.35	0.27	0.76	0.40	0.6	0.23	0.17	0.19	0.21	0.41	0.15
0.7	0.21	0.78	0.33	0.24	0.89	0.39	0.7	0.20	0.18	0.19	0.20	0.57	0.15
0.8	0.19	0.84	0.31	0.23	0.93	0.37	0.8	0.19	0.20	0.19	0.19	0.62	0.14
0.9	0.18	0.87	0.30	0.22	0.95	0.37	0.9	0.16	0.21	0.18	0.17	0.82	0.14
Pheromone swarm $\delta = 3$							*Pheromone swarm with* $\delta = 3$						
0.1	0.59	0.54	0.56	0.58	0.44	0.66	0.1	0.56	0.08	0.13	0.25	0.26	0.22
0.2	0.42	0.62	0.50	0.45	0.52	0.54	0.2	0.40	0.10	0.16	0.25	0.27	0.20
0.3	0.35	0.67	0.46	0.39	0.60	0.51	0.3	0.34	0.12	0.18	0.25	0.28	0.19
0.4	0.30	0.74	0.42	0.34	0.70	0.47	0.4	0.30	0.14	0.19	0.25	0.29	0.18
0.5	0.27	0.77	0.40	0.31	0.75	0.45	0.5	0.27	0.16	0.20	0.24	0.35	0.17
0.6	0.24	0.81	0.37	0.28	0.88	0.42	0.6	0.25	0.19	0.21	0.23	0.40	0.17
0.7	0.22	0.84	0.34	0.25	0.93	0.40	0.7	0.22	0.21	0.21	0.22	0.50	0.16
0.8	0.20	0.87	0.32	0.23	0.95	0.37	0.8	0.18	0.20	0.19	0.19	0.69	0.14
0.9	0.19	0.89	0.31	0.23	0.98	0.37	0.9	0.17	0.22	0.19	0.18	0.77	0.13
Pheromone swarm $\delta = 5$							*Pheromone swarm with* $\delta = 5$						
0.1	0.52	0.52	0.52	0.52	0.45	0.58	0.1	0.48	0.08	0.13	0.23	0.27	0.21
0.2	0.38	0.63	0.48	0.41	0.52	0.49	0.2	0.38	0.11	0.17	0.25	0.25	0.19
0.3	0.33	0.69	0.45	0.37	0.60	0.46	0.3	0.31	0.12	0.18	0.24	0.25	0.18
0.4	0.28	0.71	0.40	0.31	0.67	0.41	0.4	0.28	0.14	0.18	0.23	0.31	0.18
0.5	0.24	0.71	0.35	0.27	0.75	0.38	0.5	0.24	0.15	0.18	0.21	0.39	0.16
0.6	0.23	0.77	0.35	0.26	0.79	0.38	0.6	0.22	0.16	0.18	0.20	0.55	0.16
0.7	0.21	0.81	0.34	0.25	0.84	0.37	0.7	0.20	0.19	0.20	0.20	0.65	0.15
0.8	0.19	0.85	0.31	0.23	0.93	0.36	0.8	0.18	0.20	0.19	0.18	0.77	0.14
0.9	0.18	0.86	0.30	0.21	1.00	0.33	0.9	0.17	0.21	0.19	0.17	0.86	0.13

www.manaraa.com

Table 2 Statistical analysis for the TF-IDF, simple swarm, and pheromone swarm methods on the Pine and CM1 dataset

		MAP		DiffAR	
CM1	tfidf@0.2	0.204	Wilcoxon signed-rank (TF-IDF vs. swarm)	0.101	Mann–Whitney (TF-IDF vs. swarm)
	ss@0.3	0.227	$W+ = 4{,}806$, $W- = 6{,}519$, $N = 150$, $P \leq 0.1083$	0.281	**$U = 9{,}839$, $z = -4.65$, $P < 0.0001$**
	delta1@0.3	0.163	**$W+ = 4{,}191$, $W- = 2{,}479$, $N = 115$, $P \leq 0.01696$**	0.212	**$U = 5{,}979$, $z = -1.99$, $P < 0.0466$**
	delta3@0.1	0.222	$W+ = 6{,}117$, $W- = 6{,}924$, $N = 161$, $P \leq 0.4964$	0.261	**$U = 7{,}989$, $z = -3.2$, $P < 0.0014$**
	delta5@0.1	0.209	$W+ = 4{,}297.50$, $W- = 4{,}747.50$, $N = 134$, $P \leq 0.6181$	0.272	**$U = 7{,}722$, $z = -2.92$, $P < 0.0035$**
Pine	tfidf@0.1	0.75	Wilcoxon signed-rank (TF-IDF vs. swarm)	0.179	Wilcoxon signed-rank (TF-IDF vs. swarm)
	ss@0.1	0.76	$W+ = 272.50$, $W- = 322.50$, $N = 34$, $P \leq 0.6753$	0.456	**$W+ = 40$, $W- = 1{,}088$, $N = 47$, $P \leq 3.037e{-}08$**
	delta1@0.1	0.68	**$W+ = 397$, $W- = 164$, $N = 33$, $P \leq 0.0382$**	0.377	**$W+ = 232$, $W- = 896$, $N = 47$, $P \leq 0.0004516$**
	delta3@0.1	0.66	**$W+ = 425$, $W- = 170$, $N = 34$, $P \leq 0.0299$**	0.436	**$W+ = 162$, $W- = 966$, $N = 47$, $P \leq 2.151e{-}05$**
	delta5@0.1	0.58	**$W+ = 591$, $W- = 112$, $N = 37$, $P \leq 0.0003$**	0.445	**$W+ = 151$, $W- = 977$, $N = 47$, $P \leq 1.271e{-}05$**

Bold values indicate the improved or better results for swarms vs. VSM TF-IDF method

References

1. Boehm B (1976) Software engineering. IEEE Trans Comput 25(12):1226–1241
2. Johnson J (2000) Turning chaos into success. IEEE Softw Mag 19(3):21–27
3. Top Software Engineering Issues within Department of Defense and Defense Industry (2006) National Defense Industrial Association
4. Tun TT, Jackson M, Laney R, Nuseibeh B, Yu Y (2009) Are your lights off? Using problem frames to diagnose system failures. In: Proceedings of the IEEE international conference on requirements engineering, vol 0, pp 343–348
5. Hayes J, Dekhtyar A, Sundaram S, Howard (2004) Helping analysts trace requirements: an objective look. In: Proceedings of the 12th IEEE international conference in requirements engineering, 2004, pp 249–259
6. Engelbrecht AP (2006) Fundamentals of computational swarm intelligence. Wiley, USA
7. Antoniol G, Penta MD, Harman M (2005) Search-based techniques applied to optimization of project planning for a massive maintenance project. In: Proceedings of the 21st IEEE international conference on software maintenance, pp 240–249
8. Lam CP, Xiao J, Li H (2007) Ant colony optimisation for generation of conformance testing sequences using a characterising set. In: Proceedings of the third conference on IASTED international conference: advances in computer science and technology, pp 140–146
9. Reitz M (2006) Software evolvability by component-orientation. In: Proceedings of the second international IEEE workshop on software evolvability, pp 66–73
10. Ayari K, Bouktif S, Antoniol G (2007) Automatic mutation test input data generation via ant colony. In: Proceedings of the 9th annual conference on genetic and evolutionary computation, pp 1074–1081
11. Diaz-Aviles E, Nejdl W, Schmidt-Thieme L (2009) Swarming to rank for information retrieval. In: Proceedings of the 11th annual conference on genetic and evolutionary computation, pp 9–16
12. Zou X, Settimi R, Cleland-Huang J (2009) Improving automated requirements trace retrieval: a study of term-based enhancement methods. Empir Softw Eng 15(2):119–146
13. Maarek Y, Berry D, Kaiser G (1991) An information retrieval approach for automatically constructing software libraries. IEEE Trans Softw Eng 17:800–813
14. Niu N, Easterbrook S (2008) Extracting and modeling product line functional requirements. In: Proceedings of the IEEE international conference on requirements engineering, vol 0, pp 155–164
15. Sundaram S, Hayes JH, Dekhtyar A, Holbrook A (2010) Assessing traceability of software engineering artifacts. Requir Eng J 15(3):211–220
16. Sultanov H, Hayes JH (2010) Application of swarm techniques to requirements engineering: requirements tracing. In: Proceedings of the paper presented at the 18th international requirement engineering conference, Sydney, Australia
17. Deneubourg J-L, Aaron S, Goss S, Pasteels JM (1990) The self-organizing exploratory pattern of the argentine ant. J Insect Behav 3(2):159–168
18. Engelbrecht AP (2006) Fundamentals of computational swarm intelligence. Wiley, USA
19. Gotel O, Finkelstein A (1997) Extended requirements traceability: results of an industrial case study, p 169
20. Manning CD, Raghavan P, Schütze H (2008) Introduction to information retrieval, 1st edn. Cambridge University Press, Cambridge
21. Cleland-Huang J, Settimi R, Romanova E, Berenbach B, Clark S (2007) Best practices for automated traceability. Computer 40(6):27–35
22. Antoniol G, Canfora G, Casazza GA, Lucia D, Merlo E (2002) Recovering traceability links between code and documentation. IEEE Trans Softw Eng 28(10):970–983
23. Marcus A, Maletic JI (2003) Recovering documentation-to-source-code traceability links using latent semantic indexing. In: Proceedings of the 25th international conference on software engineering, pp 125–135
24. Hayes JH, Dekhtyar A, Osborne J (2003) Improving requirements tracing via information retrieval. In: Proceedings of the 11th IEEE international conference on requirements engineering, p 138
25. Egyed A, Grünbacher P, Graf F (2010) Effort and quality of recovering requirements-to-code traces: two exploratory experiments. In: Proceedings of the 18th international IEEE requirements engineering conference

www.manaraa.com

26. Panis M (2010) Successful deployment of requirements traceability in a commercial engineering organization…really. In: Proceedings of the 18th international IEEE requirements engineering conference
27. Zou X, Settimi R, Cleland-Huang J (2006) Phrasing in dynamic requirements trace retrieva. In: Proceedings of the paper presented at the computer software and applications conference, 2006. COMPSAC '06. 30th annual international, pp 265–272
28. Spanoudakis G, Zisman A, Perez-Minana E, Krause P (2004) Rule-based generation of requirements traceability relations. J Syst Softw 72(2):105–127
29. Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm intelligence: from natural to artificial systems, 1st edn. Oxford University Press, USA
30. van der Merwe D, Engelbrecht A (2003) Data clustering using particle swarm optimization. In: Proceedings of the 2003 congress on evolutionary computation. CEC '03, pp 215–220
31. Cui X, Potok TE, Palathingal P (2005) Document clustering using particle swarm optimization. IEEE swarm intelligence symposium, The Westin
32. Aghdam MH, Ghasem-Aghaee N, Basiri ME (2009) Text feature selection using ant colony optimization. Expert Syst Appl 36(3):6843–6853
33. Porter M (1980) An algorithm for suffix stripping. Program 14(3):130–137
34. Pine Email System. [Online]. Available: http://www.washington.edu/pine/. Accessed 19 Feb 2010
35. NASA IV&V facility metrics data program: glossary and definitions. [Online] Available: http://mdp.ivv.nasa.gov/mdp_glossary.html#CM1.%207. Accessed 19 Feb 2010
36. Hayes J, Dekhtyar A, Sundaram S, Holbrook E, Vadlamudi S, April A (2007) Requirements tracing on target: improving software maintenance through traceability recovery. Innov Syst Softw Eng 3(3):193–202
37. Hölldobler B, Wilson EO (1998) Journey to the ants: a story of scientific exploration. Belknap Press of Harvard University Press, USA

www.manaraa.com

Copyright of Requirements Engineering is the property of Springer Science & Business Media B.V. and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.

www.manaraa.com