2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies

Improving Retrieval Effectiveness using Ant Colony Optimization

Sandeep G. Khode
Dept. of Computer Science and Engineering
Thapar University
Patiala, India.
sgkhode@gmail.com

Rajesh Bhatia
Dept. of Computer Science and Engineering
Thapar University
Patiala, India
rbhatiapatiala@gmail.com

Abstract — **Software reuse is only effective if it is easier to locate and appropriately modify a reusable component than to write it from scratch. It is the use of existing software knowledge or artifacts also known as software components to build new software. There are two main problems in software reuse. First, classifying software modules in a component library is a major problem in software reuse. Second, identifying appropriate software components in a library or software component retrieval is an important task in software reuse: after all, components must be found before they can be reused. Many researchers have proposed various techniques to search and retrieve components. Proposed technique helps re-user to identify and retrieve software component. In its first step it matches keywords, their synonyms and their interrelationships. And then makes use of ant colony optimization, a probabilistic approach to generate rule for matching the component against the re-user query.**

Keywords: **Software reuse, Component Retrieval, Ant Colony Optimization.**

I. INTRODUCTION

Software reuse may be broadly defined as the use of existing engineering knowledge or artifacts to build new software systems. Software reuse can significantly improve software quality and productivity; so many organizations are now trying to implement systematic reuse programs that will allow them to derive maximum leverage from their existing software assets.

Software libraries play a very vital role in software reuse. A software library is a set of software assets that are maintained by an organization for possible browsing and retrieval [1]. Software libraries are repositories where software components are stored and searched; as such, they represent a precious resource for the software engineer. To retrieve appropriate component from the library, we need proper retrieval technique. An engineer can study library components to become familiar with a programming language or a programming style, look for common patterns of usage, get acquainted with an application domain, or reuse library components (rather than have to write them from scratch). Although many methods for component retrieval have been proposed, no method is fully efficient.

As the size of the software library increases, it becomes increasingly difficult to trust the retrieval task to informal procedures, such as those library science and information science. When the size of the library increases, we need to choose effective algorithms that maintain adequate precision and recall. Instead of going on checking each component against the query, we will get better results if find easiest route to relevant components.

In this paper, the application of new paradigm to retrieve software components is being proposed. It works on the methodology based on Ant Colony Optimization as described in [2]. The ant colony optimization algorithm (ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. This algorithm is a member of Ant colony algorithms family, in Swarm intelligence methods, and it constitutes some metaheuristic optimizations. Initially proposed by Marco Dorigo in 1992 [2], the first algorithm was aiming to search for an optimal path in a graph; based on the behavior of ants seeking a path between their colony and a source of food. The original idea has since diversified to solve a wider class of Numerical problems, and as a result, several problems have emerged, drawing on various aspects of the behavior of ants.

In this paper, section 2 will describe related works regarding component retrieval as well as few applications of ant colony optimization paradigm. And in section 3, we are going to propose our retrieval technique which is based on ant colony optimization paradigm.

II. RELATED WORK

A. *Retrieval Techniques*

Although many retrieval techniques have been proposed till date, no solution gives perfect results [3]. Like Clifton and Li [4], explained a method, Classifying software components using design characteristics. They used certain information like discriminators to describe the component and then they used neural network technology to accomplish the match. But this type of techniques has poor coverage ratio [3]. Jürgen Börstler [5] explained a technique of feature oriented classification, which takes very specific features as classes and finds similarity to make their cluster. It also shows investment cost is little bit higher [3].

There are certain methods, like behavior sampling, as explained by Podgurski and Pierce [6, 7], which uses behavior of the component, means its executable characteristics to find out exact or relevant components. But this may be irrelevant technique. Because comparing a large repository against the query may show very large time complexity as it compares internal characteristics of the component.

So we need an efficient mechanism to eliminate the above mentioned shortcomings and make a move towards finding

978-0-7695-3915-7/09 $26.00 © 2009 IEEE
DOI 10.1109/ACT.2009.187

out required components for reusing them with minimum resources to fasten the software development process with software reuse.

B. Ant Colony Optimization

A new heuristic as proposed by Marco Dorigo and Thomal Stutzle [2], ant colony optimization can be applied to retrieve software components from the repository. Ant colony optimization algorithm is based on the characteristic of real world ant. Ants follow their way to food from nest with the help pheromone deposited on their way. And they always follow the shortest route.

The basic idea of this process is illustrated in Figure 1. In the left picture the ants move in a straight line to the food. The middle picture illustrates what happens when an obstacle arrives in the path between the nest and the food. In order to cross that obstacle each first ant chooses its path on random basis. It is assumed that all ants move roughly at the same speed and deposit pheromone in the trail at roughly the same rate. However, the ants that, by chance, choose to turn left will reach the food sooner, whereas the ants that go around the obstacle turning right will follow a longer path, and so will take longer time to cross the obstacle. As a result, there is more pheromone on the shorter path around the obstacle. Since ants prefer to follow trails with larger amounts of pheromone, eventually all the ants converge to the shorter path around the obstacle, as shown in the right picture.

Many researchers have used this methodology for different kinds of work. Like Parpinneli [8] proposed an algorithm Ant Miner (i.e. Ant colony optimization based data miner) to find out rules to mine the data. And this method has been used by Holden [9] for web content classification. In ant miner [8], each ant starts with empty rule and go on construction the classification rule in the following format:

IF <conditions> THEN <class> .

Stockeart, S., Martine De C., Cornelis, C., Kerre, E. E, combined ant based clustering with c-means algorithm. They also used fuzzy IF-THEN rules to control behavior of the artificial ant in a clustering algorithm [10]. Ant colony optimization is also been used for web usage mining and web structure mining as given by Shen, Jie, Lin Ying, Chen Zhimin [11].

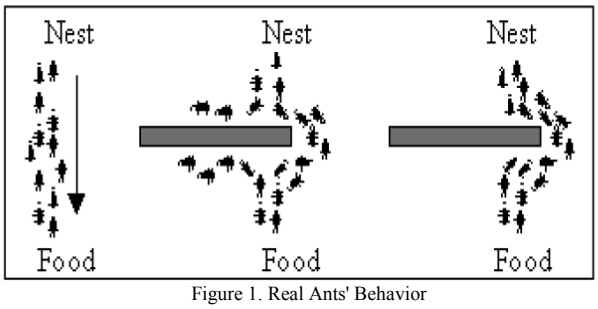

Figure 1. Real Ants' Behavior

III. COMPONENT RETRIEVAL BASED ON ANT COLONY OPTIMIZATION

A. Methodology

As given in previous sections ant colony optimization have been applied in several applications. And we can also take its advantage in retrieval of the software components. We can select candidate component by comparing its keywords available. But this may not be fully efficient way because there may be presence of ambiguities in writing the keywords.

In our work, we define the components with the help of few terms like linked list, tree, graphs etc. and also their domain of application like banking, finance, system software etc. Then we found out relevant terms with these terms to extend our comparability. For this we created a database for terms that are similar to these predefined terms. We follow following algorithm for the retrieval:

- The user enters a query; from his query keywords are identified.
- Using our words database we represent keywords in certain relations.
- Compare these relations with our available table of relations, and finding out best comparisons on the basis of keyword match.
- On the basis of this comparison we get a way towards relevant components.
- Update pheromone of each term.
- Repeat all the steps till we get our desired rule.

B. Pheromone Manipulation

Initially the pheromone for all the terms is set equal that is 1. This is because for our initial setup we say that all the terms are equally likely to be selected.

Selection of any path for matching the relations is totally dependent on the pheromone associated to that term. Probability for a path to be selected is given by following formula:

$$P_{i,j} = \frac{\eta_{i,j}.\tau_{i,j}(t)}{\sum_{i=1}^{a} xi.\sum_{j=1}^{bi}(\eta_{i,j}.\tau_{i,j}(t))} \quad (1)$$

Where:

η_{ij} problem dependent heuristic function, currently we have assumed it equal to 1;

$\tau_{ij}(t)$ is amount of pheromone currently available (at time t) in the position i,j of the trail being followed by the ant;

a is total number of attributes

b_i is total number of values on the domain of the attribute i;

I are the attributes i not yet used by the ant;

Initially the amount of pheromone will be decided on the basis of following formula [12]:

$$\tau_i = \frac{1}{\sum_{i=1}^{a} b_i} \quad (2)$$

Where :

τ_i is the pheromone associated with the i^{th} term

a is total number of attributes in $term_{ij}$

b_i is the number of possible values that attribute can have;

We also need to establish the quality of the rule that is being developed by above mentioned process. This quality is dependent on four terms, which defined as follows:

- True Positives (TP): It is the number of cases covered by the rule that have the class predicted by the rule.
- False Positives (FP): It is the number of cases covered by the rule that have a class different from the class predicted by the rule.
- False Negatives (FN): It is the number of cases that are not covered by the rule but that have the class predicted by the rule.
- True Negatives (TN): This is the number of cases that are not covered by the rule and that do not have the class predicted by the rule.

With the help of the above definitions the term quality (Q) can be formulated as given in [12] :

$$Q = \frac{TP}{TP + FN} * \frac{TN}{TN + FP} \quad (3)$$

Larger the value of the Q, higher the quality of the rule. And value of Q lies between 0 to 1. With the help of calculated quality, the pheromone of the term can be updated with the help of following formula:

$$\tau_{(i+1)} = \tau_i(t) + \tau_i(t) * Q \forall i \in R \quad (4)$$

Where:

R is set of terms occurring in the current rule constructed by the ant in iteration t.

IV. COMPUTATIONAL RESULTS

We have evaluated above mentioned process with the help of 3 programming domains, their attributes and values. Table I shows initial pheromone for identified terms in each domain. These values are identified for the formulation of the problem. The same heuristic can be applied even though number of values increases.

TABLE I. VALUES FOR COMPUTATION

Domain	Attributes	Values	Pheromone
Data Structure	Linked List	Singly	1
		Circular	
		Doubly	
		Pointer	
		Node	
		Head	
		Data	
		Linked	
		List	
	Queue	Priority	1
		Circular	
		Dequeue	
		Front	
		Rear	
		Data	
		Index	
	Stack	Top	1
		Data	
	Tree	Root	1
		Left child	
		Right child	
		Key Value	
		BST	
		B tree	
	Graph	Graph	1
		Adjacent node	
		Adjacency matrices	
		Coloring	
		Shortest Path	
		Spanning Tree	
DBMS	Tables	Primary key	1
		Tuple	
		Attribute	
		Schema	
		Constraints	
	Index	Level	1
		Key	
		Name	
		Constraints	
	File structure	Hierarchical	1
		Flat	
		Size	
		Hybrid	
	Query	Select	1
		Insert	
		Delete	
		Update	
Application software	Banking	Add	1
		Remove	
		Account	
		Customer	
		Withdraw	
		Deposit	
		Money	
	Library Management	Add	1
		Remove	
		Book	
		Record	
		Student	
		Teacher	
		Withdraw	
		Deposit	
	Inventory Management	Add	1
		Remove	
		Item	
		Ship	
		Customer	
		Supplier	
	Airline Reservation System	Booking	1
		Cancel	
		Customer	
		Ticket	

Initially, value of τ_i of each term is set to 1 with the help of (1). It represents that every term is equally likely to be selected for the first query. We assume a query for our computation which is given as follows:

A. *Case 1*

1. **Accepting user query:** User will enter the query in the form of keywords.
   E.g. We take above mentioned query as our example.
   Insert a Node into Circular Linked List
2. **Searching keywords in the database:** We will then search the given keywords in the database for the match; i.e. we will match the keywords Insert, a, Node, into, Circular, Linked, List in our database. And we will go on listing the matches
3. **Computing Quality:** The quality of the match is computed by using the given formula. Following table of matches has been derived, which has fields as term and keyword matched with that term.
   Above determined table will be matched with the available values of the matched terms viz. Query, Linked List, Queue.
   Firstly, values of Query will be considered which are as follows:
   {Select, Insert, Delete, Update}
   And hence Quality for the term *Query* is computed by using (3), we get value of Q:
   Q = 0.16
   For this we will have following values
   TP = 1; FP = 4; FN = 3 ;TN = 7

TABLE II. MATCHED VALUES CASE 1

Term	Value
Query	Insert
Linked List	Node
Linked List	Circular
Linked List	Linked
Linked List	List
Queue	Circular

TABLE III. RESULTS OF THE RETRIEVAL

Term	TP	FP	FN	TN	Q	$\tau_{(i+1)}$
Linked List	4	1	5	6	0.38	1.38
Queue	1	4	6	10	0.10	1.1
Query	1	4	3	7	0.16	1.16

4. **Updating Pheromone:** Using this value of Q and pheromone updating formula we get new value of pheromone for term Query as
   $\tau_{Query} = 1.16$

In the same way, calculations for each of the term are shown in TABLE 3. The table also shows value of the Quality and new pheromone value.

TABLE IV. MATCHED RELATIONS FOR CASE 2

Term	Value
Inventory Management	Ship
Inventory Management	Item

In this way, updated pheromone value for each term is calculated. It is observed that new pheromone value for term Linked List is highest that means its quality of matching is highest. This leads us to more relevant components with our query.

B. Case 2

Consider following query:

Ship Item

With this query, intent is to find the component with functionality of shipment of an item Inventory management system. For this query the matched relations are shown in TABLE IV.

The quality of the query with the help of factors TP, FP, TN, FN is shown below.

TP = 2; FP = 2; FN = 4; TN = 6

And hence Q = 0.33

$\tau_{Inv.Mgt.Sys} = 1.33$

With this query, components retrieved are related to only Inventory Management System. These are components are the only relevant to this query.

V. CONCLUSION

The retrieval method proposed in this paper makes use Ant Colony Optimization algorithm for retrieval of software components. The method shows very good values of precision and recall. As the method directs towards only the relevant components; recall of the method is always seen as 1. Higher value of precision can be achieved by in case of exact queries belonging to single or restricted domain. Precision degrades gracefully in case of indistinct queries, because this system extracts multiple domains belonging to these queries.

REFERENCES

[1]. Mili, H., Mili, A., Yacoub, S., and Addy, E., Reuse Based Software Engineering, Wiley-Interscience Publication, USA, 2002.

[2]. Dorigo, M. and Thomas, S., Ant Colony Optimization, Prentice-Hall of India Publication, New Delhi, 2005, pp 223-244.

[3]. Mili, A., Mili, R., Mittermer, R. T., A survey of reuse libraries, Annals of software engineering 5, 1995, pp 349-414.

[4]. Clifton, C. and W.-S. Li, Classifying Software Components Using Design Characteristics, In Proceedings of the 10th Knowledge-Based Software Engineering Conference, KBSE '95, IEEE Computer Society Press, Los Alamitos, CA, 1995, pp 139–146.

[5]. Boerstler, J., Feature Oriented Classification for Software Reuse, In Proceedings of the 7th International Conference on Software Engineering and Knowledge Engineering, SEKE '95, KSI Knowledge Systems Institute, Skokie, IL, 1995, pp 204–211.

[6]. Podgurski, A. and L. Pierce, Behaviour Sampling: A Technique for Automated Retrieval of Reusable Components, In Proceedings of the 14th International Conference on Software Engineering, ACM Press, New York, NY, 1992, pp 300–304.

[7]. Podgurski, A. and L. Pierce, Retrieving Reusable Software by Sampling Behavior, ACM Transactions on Software Engineering and Methodology 2, 3, 1993, pp 286–303.
[8]. Parpinelli, R. S., Lopes, H. S., and Freitas, A. A., Data Mining with an Ant Colony Optimization Algorithm. IEEE Transactions on Evolutionary Computing, 2002 (6)(4), 2002, pp 321- 332.
[9]. Holden, N. and Freitas,A.A., Web Page Classification with an Ant Colony Algorithm. In Parallel Problem Solving from Nature-PPSN VIII, LNCS 3242, 1092-1102, Springer Verlag, September 2004
[10]. Stockeart, S.,Martine De C. ; Cornelis, C. ; Kerre, E. E., Fuzzy ant based clustering, ANTS 2004: ant colony optimization and swarm intelligence: International workshop on ant colony optimization and swarm intelligence, Brussels , BELGIQUE, 2004, pp 342-349.
[11]. Shen, Jie, Lin Ying, Chen Zhimin, Incremental web usage mining, Wuhan University journal of natural sciences, 2006, pp 1081-1085.
[12]. Bhatia, R., Dave, M., Joshi R.C., Ant colony based rule generation for reusable software component retrieval, ISEC 08 Hyderabad, India, 2008, pp 129-130.