Ant Colony Based Rule Generation for Reusable Software Component Retrieval

Rajesh K. Bhatia
Thapar University
Patiala
91-175-2393228
rbhatiapatiala@gmail.com

Mayank Dave
National Institute of Technology
Kurukshetra
91-1744-238296
mdave67@yahoo.com

R. C. Joshi
Indian Institute of Technology
Roorkee
91-1332-2393228
rcjoshfec@iitr.ernet.in

Abstract

Storage and representation of reusable software components in software repositories to facilitate convenient identification and retrieval has been always a concern for software reuse researchers. This paper discusses and demonstrated an ant colony algorithm based technique that generates rules to store and then identify the component from software repository for possible reuse. Proposed technique help user in organization and storage of components in repository and later can help in identifying most appropriate component for given context. In first stage while searching it makes use of keywords, their synonyms and their inter-relationships. Then it makes use of ant colony optimization; initial pheromone of one is assigned to all domain representative terms of components. By updating pheromone for participating terms and non-participating terms iteratively and by calculating the quality of each rule generated, it leads to quality rules to represent and retrieve the reusable components.

Categories and Subject Descriptors

D.2.13 [Reusable Software]

General Terms

Algorithms, Experimentation

Introduction

The goal of Software Reuse is to construct any software from previously developed components available in software repository. Software repositories usually consist of large number of components of various domains and are very huge in size. Software reuse consists of two types of tasks, development for reuse and development with reuse. Reuse involves four steps in general selection, analysis, customization and instantiation [1]. Component classification is very vital step and it is very difficult to identify appropriate component from repository for given context. As natural language based simple keyword search suffers from inherited ambiguity problem. Because the search space is very large, in this work an ant colony optimization based component classification technique is proposed. In this task major goal is to generate rules for classifying components. For a particular context, various rules are generated and then their quality is checked. Various terms in form of ants are selected for a given context, and then their interrelationships are derived. Checking deposited pheromone on various terms generates rules. Pheromone is deposited on a particular path in this case term and their relationship, if it is contributing for current context otherwise pheromone is evaporated and that term will not play role in rule generation. Quality of a rule is calculated on the basis of various parameters. If quality of two consecutive rules is same or nearly same it means it converged and is included in the list of various rules applicable to present context.

In proposed work 150 terms i.e. ants, their attributes, Characteristics means attribute values and their initial pheromone is considered. These 150 ants may represent a large software repository, as one term may represent similar components. One database to represent various domains of components is created. In this database representative terms of the domain are stored. 10 ant-based terms are considered to form a rule. We have considered “Vehicle” as a representative class. It took three iterations to generate one rule. A set of rules can be generated to satisfy one query and a particular path means a component is selected on the basis of pheromone deposited. The major goal is to discover a good set of classification rules to classify reusable software component from repository for given context.

To the best our knowledge, the use Ant Colony algorithm for discovering classification rules for retrieving reusable software components from repository, is a research area still unexplored. Actually ant algorithms developed for data mining and clustering [2, 3] are different from the classification task addressed in this paper.

Methodology

We usually see natural ants creating paths from food to home; these paths are always shortest possible paths from the food source to their home. Ants leave a chemical substance pheromone, on the path they traverse. Other ants to select the appropriate path use this pheromone, more number of ants traverses the same path, more amount of pheromone will get deposited and chances of selecting this path by the following ants will increase. Ant colony optimization algorithms are based on the idea that “Each path followed by an ant is associated with a candidate solution for a given problem. When an ant follows a path, the amount of pheromone deposited on that path is proportional to the quality of the corresponding candidate solution for the target problem. When an ant has to chose two or more paths, the paths with a larger amount of pheromone have a greater probability of being chosen by the ant.” As a result, the ants eventually converge to a short path, hopefully optimum or a near optimum solution for the target problem.

Similarly, retrieval of appropriate reusable software component for a given context is also a optimization problem. We may select candidate components from the large repository on the basis of keywords but to select most appropriate component keyword search may not work due to ambiguity problem Also we need to know and compare semantic attributes of any component with given user context to return most appropriate component. A component repository containing about one lakh fifty thousand reusable software components have been used for taking results of current algorithm. To represent these components a database of 150 terms like linked list, tree, update etc, is created, Attributes of these terms are considered then their values are assigned. Initially pheromone assigned to each term is one. In this repository components belonging to various 22 domains like banking, systems, hospital, finance etc are stored. To represent each domain, a set of attributes on the basis of their functionality is created. First user enters a query, from his query keywords are identified. Using WordNet various terms to represent these keywords and then their relations are identified. Compare these terms with database of 150 terms and select 10 best terms from the available on the basis of keyword match for given context. Compare these 10 terms with the domain attributes for the said context. On the basis of these comparisons, calculate pheromone for each term. Then calculate quality of these ten terms to represent the current context. Pheromone for each term will get updated. In next iteration only terms with more amount of pheromone will get selected and irrelevant terms means less amount of pheromone will be ignored. These iterations are repeated till we get the rule with desired quality or the quality of two consecutive rules is same [7]. This rule will be stored in domain rules database and then next rule will be generated for the same domain. Rule generation steps are explained in figure 1.

Pheromone Updation

The pheromone of each term occurring in a rule is updated in each iteration. If a term occurs in the rule it means ant visited this term in generating this rule correspondingly its pheromone deposited increases. But if any term does not occur in current rule it means ant has not visited this term and correspondingly pheromone is evaporated. For next iteration these updated pheromone values in database are used to generate next rule. Iterations are repeated till a rule with quality 0.9 or two consecutive rules with same or nearly same quality value are generated. As current ant completes the construction of rule, the rule pruning takes place. The strategy for rule pruning is same as used in [4, 5], but rule quality criteria in this approach is different.

The basic idea here is to iteratively improve the quality of the constructed rule. Initially terms in first rule are selected on the basis of keyword, its synonyms and their interrelationships. These terms are then compared with the domain class terms stored in a separate domain database. The terms, which are present in constructed rule and are also there in the domain class, will play active rule in rule generation. Other terms are iteratively removed one by one. In each iteration the terms, which are matched with domain their pheromone is increased using equation (1) and equation (2). The quality of the current rule is calculated using equation (3). The terms that does not occur in the rule their pheromone has to be decreased to show evaporation. Pheromone evaporation for unused terms is implemented by normalization the pheromone of each term i.e. and by dividing each term's pheromone with sum of pheromone value of all participating terms.

$$\tau_{ij} = \frac{1}{\sum_{i=1}^{a} b_i} \qquad (1)$$

Where τ_{ij} is term$_{ij}$, the amount of pheromone associated with each term found by the ant is increased in proportion to the quality of the rule.

A – Total number of attributes in termij, bi, is the number of possible values that attributes can have.

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \tau_{ij}(t) * Q \forall i, j \in R \qquad (2)$$

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

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

Where Q is the quality of a rule.

TP- True Positives, is the number of terms covered by the rule that have the domain predicted by the rule.
FP- False Positives, is the number of terms covered by the rule but that have the different domain predicted by the rule.
FN- False Negatives, is the number of terms that are not covered by the rule but have the domain predicted by the rule
TN- True Negatives, is the number of terms that are not covered by the rule and do not from the predicted domain

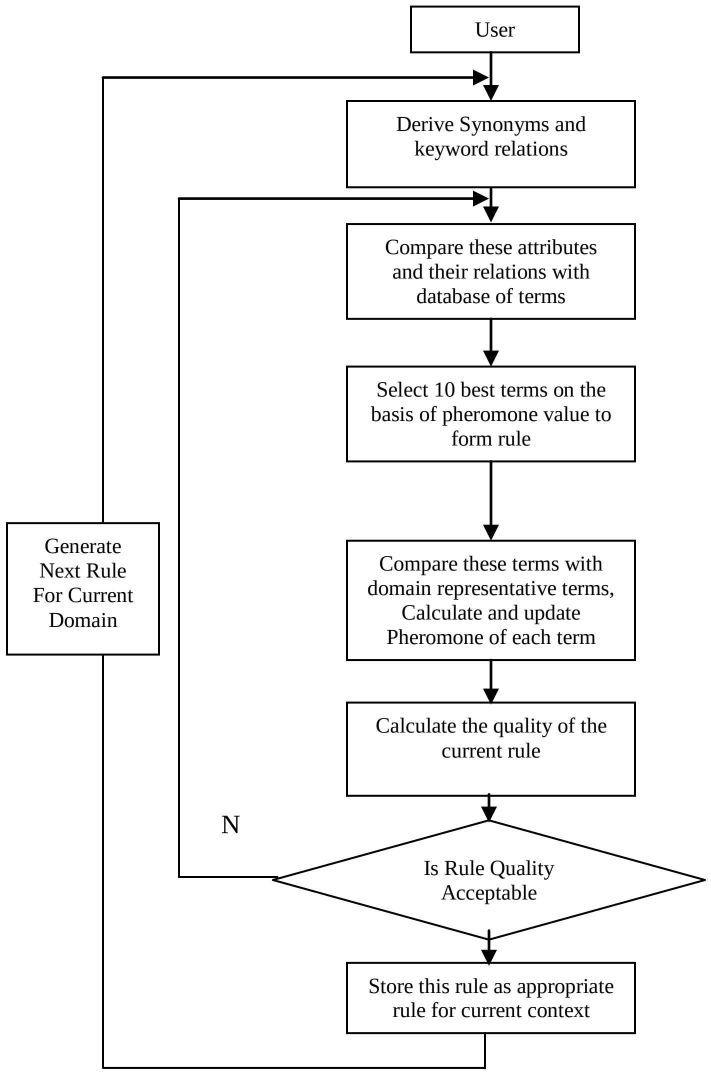
User
Derive Synonyms and keyword relations
Compare these attributes and their relations with database of terms
Select 10 best terms on the basis of pheromone value to form rule
Compare these terms with domain representative terms, Calculate and update Pheromone of each term
Calculate the quality of the current rule
N
Is Rule Quality Acceptable
Store this rule as appropriate rule for current context
Generate Next Rule For Current Domain

Fig 1 Rule Generation Steps

Experimental Results

Consider a database of 150 terms to represent the reusable software components. Initially terms are selected on the basis of keywords, their synonyms and relationships among these terms. Initial pheromone value of 1 is assigned to all terms in database. A user query of 'Linked List' is entered from data structure domain. In domain database there are 11 terms to represent this type of components in the software repository. When we searched our representative terms database on keywords, 10 terms are selected to form Rule 1.

Terms	Attributes	Values	Initial Pheromone
T1	A1 A2 A3 A4	V1 V2 V3 V1 V2 V1 V1 V2 V3	1
T2	A1 A2 A3	V1 V2 V1 V2 V3 V1	1
T3	A1 A2 A3 A4 A5	V1 V2 V3 V1 V2 V1 V2 V3 V4 V1 V1 V2	1
T4	A1 A2	V1 V2	1
T5	A1 A2 A3	V1 V2 V3 V1 V2 V1	1
T6	A1 A2 A3	V1 V2 V3 V1 V1	1
T7	A1 A2 A3	V1 V1 V2 V1	1
T8	A1 A2 A3 A4	V1 V1 V2 V1 V1 V2	1
T9	A1 A2	V1 V1 V2 V3 V4	1
T10	A1 A2	V1 V1	1
T11	A1 A2 A3	V1 V1 V1 V2 V3	1
T12	A1 A2 A3	V1 V2 V1 V2 V1 V2	1
T13	A1 A2	V1 V2 V1 V2	1
T14	A1 A2	V1 V1	1
T15	A1 A2 A3	V1 V1 V1	1
T16	A1 A2 A3	V1 V1 V1 V2	1
T17	A1 A2	V1 V2 V1 V2	1
T18	A1 A2 A3 A4	V1 V2 V1 V2 V1 V2 V1	1
T19	A1 A2 A3	V1 V2 V1 V2 V3 V1 V2 V3 V4	1
T20	A1 A2	V1 V2 V3 V1	1
…….			
T150	A1	V1	1

Table 1 Representative Terms database containing 150 terms

1	T1
2	T3
3	T5
4	T6
5	T8
6	T11
7	T13
8	T16
9	T18
10	T19

Table 2 Rule 1

1	T1
2	T3
3	T5
4	T6
5	T8
6	T9
7	T11
8	T13
9	T16
10	T100
11	T101

Table 3 Data Structure domain terms

TP=8 (T1, T3, T5, T6, T8, T11, T13, T16),

FP=2 (T18, T19), FN=3 (T9, T100, T101),

TN=5 (T18, T19, T9, T100, T101)

Now find the quality of Rule1 using equation (3), Q=0.52. Next step is to update the pheromone of each term occurring or not occurring in the rule using equation (1) and equation (2)

$$\tau_{ij} = \frac{1}{\sum_{i=1}^{a} b_i},$$

Put values of *a* and *b* from database containing 150 terms for all participating terms, we shall get:

T1=1/9, T3=1/12, T5=1/6, T8=1/6, T11=1/5, T13=1/4, T16=1/4

Using equation (2), the amount of pheromone increment for T1=1/9+1/9*0.52=0.17. Similarly we can calculate for other participating terms of Rule1.

To decrease the pheromone of two not occurring terms T18 and T19, it will be computed by dividing the current pheromone value(not modified) of term by the total of pheromone values for all participating terms(modified).

Total = Increased pheromone values (T1+T3+T5+T6+T8+T11+T13+T16)=10.17

So the pheromone value of T18 and T19 is to be decreased by 1/10.17 i.e. by 0.1.

Terms	Attributes	Values	New Pheromone
T1	A1 A2 A3 A4	V1 V2 V3 V1 V2 V1 V1 V2 V3	1.17
T2	A1 A2 A3	V1 V2 V1 V2 V3 V1	1
T3	A1 A2 A3 A4 A5	V1 V2 V3 V1 V2 V1 V2 V3 V4 V1 V1 V2	1.12
T4	A1 A2	V1 V2	1
T5	A1 A2 A3	V1 V2 V3 V1 V2 V1	1.26
T6	A1 A2 A3	V1 V2 V3 V1 V1	1.30
T7	A1 A2 A3	V1 V1 V2 V1	1
T8	A1 A2 A3 A4	V1 V1 V2 V1 V1 V2	1.26
T9	A1 A2	V1 V1 V2 V3 V4	1
T10	A1 A2	V1 V1	1
T11	A1 A2 A3	V1 V1 V1 V2 V3	1.30
T12	A1 A2 A3	V1 V2 V1 V2 V1 V2	1
T13	A1 A2	V1 V2 V1 V2	1.38
T14	A1 A2	V1 V1	1
T15	A1 A2 A3	V1 V1 V1	1
T16	A1 A2 A3	V1 V1 V1 V2	1.38
T17	A1 A2	V1 V2 V1 V2	1
T18	A1 A2 A3 A4	V1 V2 V1 V2 V1 V2 V1	0.9
T19	A1 A2 A3	V1 V2 V1 V2 V3 V1 V2 V3 V4	0.9
T20	A1 A2	V1 V2 V3 V1	1
……..			
T150	A1	V1	1

Table 4 RepresentativeTerms Database with Updated pheromone values after Rule1

Now terms are selected to form Rule2 on the basis of their pheromone value.

1	T1
2	T3
3	T5
4	T6
5	T8
6	T9
7	T11
8	T13
9	T16
10	T17

Table 5. Rule2

Again compare it with domain terms in Table 3 to select occurring and not occurring terms in Rule2.

TP=9, FP=1, FN=2, TN=3, Quality of the Rule2 is Q=0.61.

Now we shall calculate pheromone increment and decrement for participating and not participating terms respectively by using equations 1 & 2. Pheromone increment for participating terms, T1, T3, T5, T6, T8, T9, T11, T13, T16, is calculated. Pheromone decrement for not participating terms, only T17, is calculated.

Terms	Attributes	Values	New Pheromone
T1	A1 A2 A3 A4	V1 V2 V3 V1 V2 V1 V1 V2 V3	1.35
T2	A1 A2 A3	V1 V2 V1 V2 V3 V1	1
T3	A1 A2 A3 A4 A5	V1 V2 V3 V1 V2 V1 V2 V3 V4 V1 V1 V2	1.25
T4	A1 A2	V1 V2	1
T5	A1 A2 A3	V1 V2 V3 V1 V2 V1	1.53
T6	A1 A2 A3	V1 V2 V3 V1 V1	1.62
T7	A1 A2 A3	V1 V1 V2 V1	1
T8	A1 A2	V1 V1 V2	1.53

Terms	Attributes	Values	New Pheromone
	A3 A4	V1 V1 V2	
T9	A1 A2	V1 V1 V2 V3 V4	1.32
T10	A1 A2	V1 V1	1
T11	A1 A2 A3	V1 V1 V1 V2 V3	1.62
T12	A1 A2 A3	V1 V2 V1 V2 V1 V2	1
T13	A1 A2	V1 V2 V1 V2	1.78
T14	A1 A2	V1 V1	1
T15	A1 A2 A3	V1 V1 V1	1
T16	A1 A2 A3	V1 V1 V1 V2	1.78
T17	A1 A2	V1 V2 V1 V2	0.93
T18	A1 A2 A3 A4	V1 V2 V1 V2 V1 V2 V1	0.9
T19	A1 A2 A3	V1 V2 V1 V2 V3 V1 V2 V3 V4	0.9
T20	A1 A2	V1 V2 V3 V1	1
…….			
T150	A1	V1	1

Table 6. RepresentativeTerms Database with Updated pheromone values after Rule1

Select terms to form Rule3

1	T1
2	T2
3	T3
4	T5
5	T6
6	T8
7	T9
8	T11
9	T13
10	T16

Table 7. Rule3

Repeat all above steps to calculate quality and pheromone increment/decrement to form new rule. It is observed that TP=9, FP=1, FN=2, TN=3, Q=0.61, it is same as of Rule2. So it converges here. Now this rule can be selected to represent component Linked List in the repository. This rule will help any user in future to retrieve the Linked List component from the software repository.

Conclusions

We have presented an ant colony algorithm based technique for rule discovery. The major goal is to discover rules to represent various domains of reusable components. Further these rules help in retrieving appropriate component from the reusable components repository. We have considered here one sub domain 'Data Structures' in set of components and then 'Linked List'. It has been found that only three iterations were required to discover a rule to represent or select 'Linked List'. Rule convergence criteria considered here is either attaining quality value of 0.9 or two consecutive rules with same quality. Experimental results shows that terms successfully can generate rules for component repository.

We further plan to experiment ant colony algorithms with formal specifications using Object Constraint Language (OCL) to generate rules for classifying and clustering components. Proper Clustering of components will also help software reuse practitioners.

References

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

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

[3] Holden, N. and Freitas,A.A 2004, Web Page Classification with an Ant Colony Algorithm, Parallel Problem Solving from Nature-PPSN VIII, LNCS 3242, 1092-1102, Springer Verlag, September 2004

[4] Parpinelli,R.S., Lopes,H.S., and Freitas,A.A 2002, Data Mining with an Ant Colony Optimization Algorithm, IEEE Transactions on Evolutionary Computing, 2002 (6)(4), 321-332.

[5] Jiang,W., Xu,Y., and Xu,Y. 2005, A Novel Data Mining Method Based on Ant Colony Algorithm, Advance Data Mining and Applications, LNAI(LNCS) 3584, 284-291, Springer-Verlag Berlin Heidelberg 2005.

[6] Stutzle, T., and Dorigo, M. 2002, A short convergence proof for a class of ACO algorithms, IEEE Transactions on Evolutionary Computation, 6(4), 358-365.

[7] Rajesh Bhatia, M. Dave, and Joshi, R. C. 2008, Ant Colony Based Rule Generation for Reusable Software Component Retrieval, ACM ISEC 08, 129-130.