On the problem of version selection to create a graph for the ant colony optimization algorithm

On the problem of version selection to create a graph for the ant colony optimization algorithm

M V Saramud[1, 2]**, I V Kovalev**[1, 2]**, V V Losev**[2]**, D I Kovalev**[2] **and M V Karaseva**[1, 2]

[1] Siberian Federal University, 79, Svobodny Pr., Krasnoyarsk, Russia, 660041
[2] Reshetnev Siberian State University of Science and Technology, 31, Krasnoyarsky Rabochy Av., Krasnoyarsk, Russia, 660037

E-mail: msaramud@gmail.com

Abstract. The article considers various graph creation methods for presenting the structure of a multiversion software package in solving its optimization problem by the ant colony optimization algorithm. The paper presents a method for calculating the weights of arcs for a static version of the graph and the reliability characteristics of the software package, taking into account its architecture. The version of creating a dynamic version of the graph is considered. A comparative analysis of these versions is conducted. The conclusions about the preferred application of the static version of the graph are made.

1. Introduction

Nowadays, evolutionary algorithms have been extensively developing. They are optimization algorithms based on natural decision-making mechanisms [1]. One of these algorithms is the ant colony optimization (ACO) algorithm [2]. This algorithm is inspired by the behavior of real ants; or rather it reduced to their ability to finding the shortest paths to a food source. An ant colony is a multi-agent system, and despite the simplicity of its individual representatives, this system is able to solve complex problems.

2. The application of the ACO to the problem solving

At the moment, there has been no ready-made solution for applying the ant colony algorithm to solving the problem of optimal assembling of the multiversion software package. For this reason, the problem of the basic algorithm modification arises, i.e., it is necessary to form a method for creating a graph representing various structure of the software package and a calculation formula to determine the amount of pheromone on each arc of the graph [3]. Also, it is necessary to select optimal values of the algorithm's coefficients for problem solving.

Two main approaches to graph creation are proposed. The first approach is a static graph of the sufficiently large dimension in which M + 1 levels can be distinguished, where M is a number of modules of the software package. At the first level a start node is located. At each subsequent of the M levels, all possible combinations of versions for the given module are located. For example, let we have 10 versions $V_{1\text{-}1}$-$V_{1\text{-}10}$ and the restriction on the structure of the module is at least 3 modules for the possibility of multiversion voting for the M_1 module [4].

Thus, nodes at the M_1 level will be of all possible combinations of versions from 3 to 10 in each node. Each version can be included only once since the duplication of identical versions does not make sense due to copying errors that may be contained in them. Thus, we have 968 possible combinations for this module (1,2,3; 1,2,4; 1,2,5; ... 6,7,8,9; ... 1,2,3,4,5,6 , 7,8,9,10). Denote the given combination as M_{1-1} - M_{1-968}. As a result, we have 968 arcs that lead to the corresponding nodes from the start node. It is possible to go to any node of the M_2 level from each of the nodes of the M_1 level. For simplicity, we assume the conditions for all modules to be the same: 10 different versions of each module and from 3 to 10 versions inclusive in each module. The conditional image of the graph is shown in figure 1.

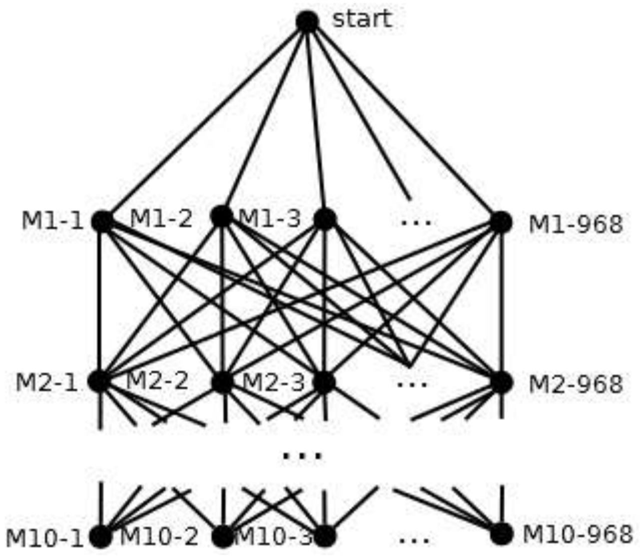

Figure 1. Static version of the graph.

It is not possible to display a graph of a given dimension entirely. The transition scheme shows that a total number of unique combinations will be 968^K, where K is a number of modules in the software package. For example, we take K = 10. For our example, we will get 968^{10} unique paths along which the ant can move. They correspond to the unique combinations of versions in the software package.
It is necessary to determine what version characteristics are the most important for us, and, therefore, on which basis we will make decisions in selecting versions.

Then it is necessary to calculate characteristics of the software package, taking into account the characteristics of each version in the structure and the whole structure of the complex.

3. Calculation of the software package reliability

It is necessary to know the architecture of the designed software package, since it affects the calculation of the most important parameter - reliability, i.e., the probability estimation of the failure-free operation for the given period of time. This is a probabilistic characteristic, taking values from 0 to 1, where 0 is an absolutely unreliable system, and 1 is an absolutely reliable system. The architecture of the software package, considered as an example, is presented in figure 2.

In the diagram one can see that modules M1, M8, M9 and M10 are always applied. Modules M2-M4 are activated with a probability of 0.27, module M5 with a probability of 0.44, module M7 with a probability of 0.29, and module M6 with a probability of 0.73, since two paths with probabilities of 0.44 and 0.29 activate it. The overall reliability of the complex is calculated by the formula $P_{rel} = P_{m1} * (0.27 * (P_{m2} * P_{m3} * P_{m4}) + 0.29 * (P_{m7} * P_{m6}) + 0.44 * (P_{m5} * P_{m6}\ )\ ) * P_{m8} * P_{m9} * P_{m10}$. The element P_{m6} occurs in the formula twice, since two different paths lead to its activation. It becomes clear from the formula that a change in the reliability of various modules will not equally affect the reliability of the entire software package. For example, the reliability of M1, M8, M9, and M10 modules is more important than the reliability of M7 module.

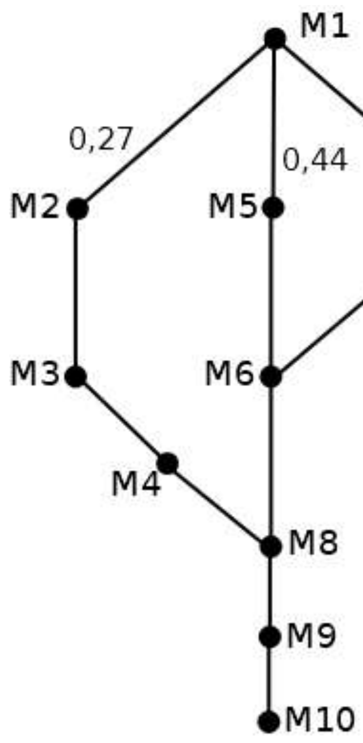

Figure2. Scheme of the software package.

There exist 2 types of the software packages development. They are top-down and bottom-up development [5]. In solving the top-down development problem, we know the requirements for the versions that need to be developed. The following parameters will be used for optimization: C is cost of the version implementation, R is reliability of the version, V is probability of successful implementation of the version (the probability that the version will be implemented with the specified reliability for the expected cost). The weight of each arc will be calculated by the formula $W_{ij} = \frac{(R*V)^{\beta}}{C}$, and the probability of transition along this arc will be calculated by the formula $P_{ij} = \frac{{\tau_{ij}}^{\alpha}*W_{ij}}{\sum({\tau_{ij}}^{\alpha}*W_{ij})}$, where τ_{ij} is value of the pheromone on this arc, α and β are the coefficients that affect the algorithm. The more α, the stronger the solution of the ant depends on the level of pheromones, the more β, the more the solution of the ant depends on the weight of the arc [10]. It is important to note that in our case the arc does not have a length, as in any classical algorithm, and weight is more of an inverse characteristic - the more weight, the “more attractive” an arc is.

In the classical model, after an ant successfully passes the route, it leaves a trace on all the edges it moves, inversely proportional to the length of the moved path. In our implementation, the pheromone value will increase by the given values in two cases - if an ant chose a structure that satisfies the restrictions (for example there are restrictions on the minimum reliability and estimation of the successful implementation of the system, in optimizing the cost) and in the case when the structure replaces the optimal solution.

4. Alternative way to create a graph

There exists a method for creating a graph significantly smaller than the dimension, but changing dynamically with each passage of the ant. First, it is necessary to build the smallest possible system to implement this approach, for the considered earlier case it could be the selection of the first three versions for each module (V_{1-1},V_{1-2}, V_{1-3}, … , V_{10-1},V_{10-2}, V_{10-3}). Then, a graph consisting of possible actions concerning the structure is dynamically created. Since in this case each module contains only 3 versions the minimum required, the actions of removing versions from the structure are not available. As it makes no sense to duplicate versions, the steps of adding the first three versions are also unavailable. Therefore, only arcs corresponding to the actions of adding version V_4-V_{10} are constructed.

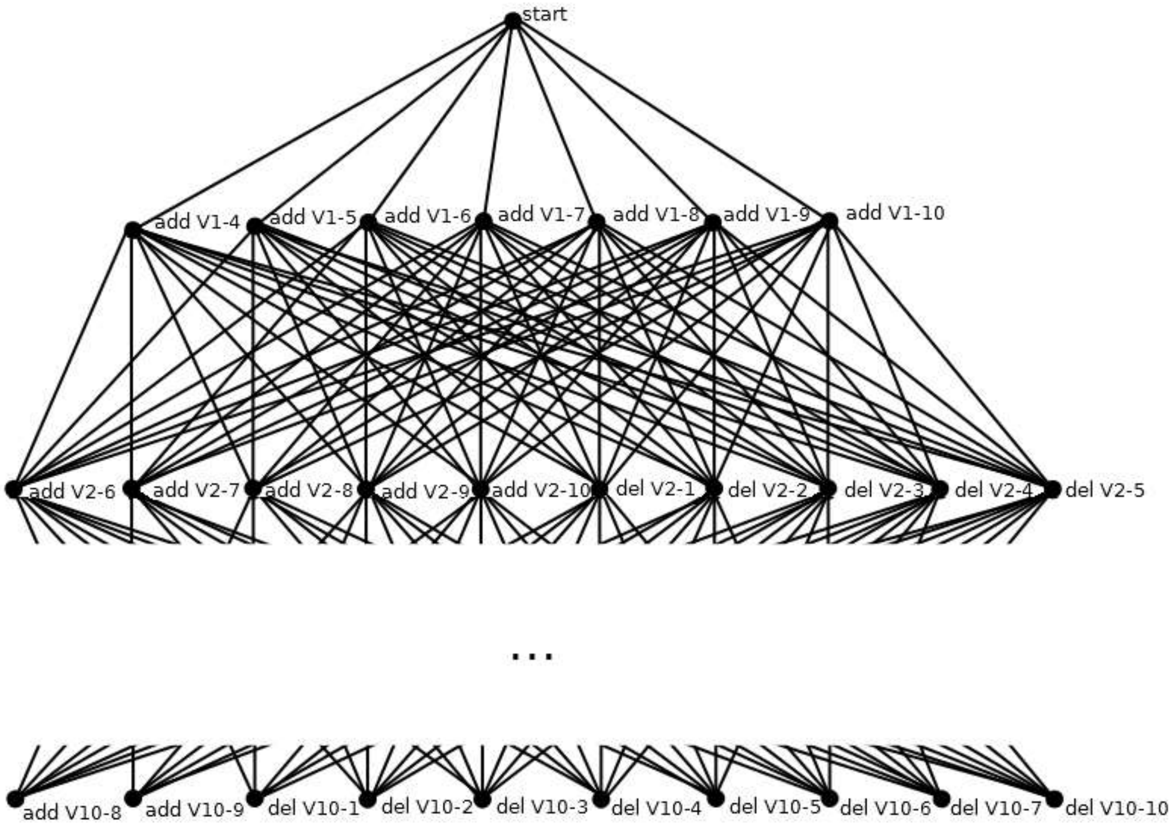

Figure 3. Dynamic version of the graph.

Consider a more illustrative example, presented in figure 3. In this example, versions (V_{1-1},V_{1-2}, V_{1-3}) are selected for module M_1, versions (V_{2-1},V_{2-2}, V_{2-3}, V_{2-4}, V_{2-5}) are selected for module M_2 and versions (V_{10-1},V_{10-2}, V_{10-3}, V_{10-4}, V_{10-5},V_{10-6}, V_{10-7}, V_{10-10}) are selected for module M_{10}. Since modules M_2 and M_{10} contain more than 3 minimum necessary versions, arcs appear corresponding to the variant for removing from any of the current versions, as well as adding the remaining ones.
For a static version of the graph, formulas for calculating the weights of arcs are proposed. The pheromone evaporation is also calculated in an obvious way, i.e., after each pass of a group of ants for arcs that did not participate in the “good” solutions, pheromones evaporate with a given coefficient. This approach is implemented programmatically and described in the article [6]. The obtained results confirm the applicability of the method and its efficiency for the problem solving.

The dynamic graph has a significantly smaller size, as well as, a number of nodes and arcs. However, the considerable problem arises in calculating the pheromone evaporation. The initial weight of the arcs can be calculated based on the characteristics of the versions that they add or remove from the solution. But it is also possible to calculate the addition of pheromone to the passed path. But when calculating the evaporation, a problem arises that in creating a dynamic graph, some arcs are not applied. Consequently, some arcs cannot be applied in solving. The problem arises whether pheromone should be evaporated on them. It is impossible to develop a functioning model using a dynamic graph without solving this problem. There is also an important point. It is necessary to build some initial structures of the software package and change it in the future to work with a dynamic graph. It somehow contradicts the concept of randomness embedded in ACO. There is no any structure at the beginning of the work with a static graph. It is determined randomly passing through the graph. And in the future, new structures are also determined completely randomly based on the weights of arcs and the level of pheromone. In the case of a dynamic graph, only the current structure is changed by no more than 1 version in each module per graph passage.

5. Conclusion

The article considers the application of the ant colony optimization algorithm to solve the problem of optimizing the structure of a multiversion software package. The modifications of the calculated formulas for the weights of arcs are given. Two methods of creating a graph are given to represent the structure of a multiversion software package in solving the problem of its optimization by the ant colony optimization algorithm. For the static graph there exist all the necessary calculation formulas and it was implemented in software that confirms its applicability for this task. Then, for a dynamic graph, there exist some problems with calculating the pheromone evaporation on arcs that do not participate in the current passage of ants. In addition, a dynamic graph does not correspond to the concept of random path search embedded in ACO. Therefore, it is necessary to apply a static version of graph creating to solve a problem of selecting the optimal structure of the multiversion software package by the ant algorithm. The only drawback of this solution is a rather large size of the graph, from which two consequences follow: large resource consumption and a great chance of passing the first group of ants. For modern computers, increasing the consumption of resources by a static graph is insignificant and it is not a real drawback. And one can apply the technique of multiple starts to solve the problem of random passage of the first group of ants,

Acknowledgment
This article was fulfilled with the financial support of the grant from the President of the Russian Federation (agreement No. 075-15-2019-366).

References

[1] Dorigo M and Birattari M 2007 Swarm intelligence *Scholarpedia* **2(9)** 1462
[2] Dorigo M, Di Caro G and Gambardella L M 1999 Ant algorithm for discrete optimization *Artificial Life* **5(2)** 137–72
[3] Zhai Y, Xu L and Yang X 2015 Ant Colony Algorithm Research Based on Pheromone Update Strategy *7th International Conference on Intelligent Human-Machine Systems and Cybernetics* vol 1 pp 38 – 41
[4] Kovalev I, Voroshilova A, Losev V, Saramud M, Chuvashova M and Medvedev A 2018 Comparative tests of decision making algorithms for a multiversion execution environment of the fault tolerance software *Proceedings - 2017 European Conference on Electrical Engineering and Computer Science (EECS 2017)* pp 211-7
[5] Li D and Chang C K 2009 Initiating and Institutionalizing Software Product Line Engineering: From Bottom-Up Approach to Top-Down Practice *33rd Annual IEEE International Computer Software and Applications Conference* **1** pp 53-60
[6] Saramud M V, Kovalev I V, Losev V V, Karaseva M V and Kovalev D I 2018 On the application of a modified ant algorithm to optimize the structure of a multiversion software package Advances in Swarm Intelligence (ICSI 2018) *Lecture Notes in Computer Science* **10941** 91-100