УДК 004.023

МОДИФИКАЦИЯ МУРАВЬИНОГО АЛГОРИТМА ДЛЯ ЗАДАЧИ ФОРМИРОВАНИЯ МУЛЬТИВЕРСИОННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

И. В. Ковалев, М. В. Карасева, Е. В. Соловьев

Сибирский государственный аэрокосмический университет имени академика М. Ф. Решетнева
Российская Федерация, 660014, г. Красноярск, просп. им. газ. «Красноярский рабочий», 31
E-mail: blackdeathangel@rambler.ru

Исследуется мультиверсионное программное обеспечение и решается задача его формирования как задача оптимизации. Рассматривается муравьиный алгоритм как способ решения задачи формирования мультиверсионного программного обеспечения. Данная методология основывается на введении программной избыточности и позволяет существенно повысить уровень надежности программного обеспечения. Проведены эксперименты с помощью стандартного алгоритма и выполнена модификация алгоритма с повторением экспериментов на тех же данных. Внесенные модификации улучшают алгоритм, что демонстрируют результаты тестовой задачи. Хотя вследствие внесенных изменений скорость расчета на одной итерации замедляется, увеличение скорости схождения алгоритма в область оптимального решения компенсирует данный недостаток. Представлено сравнение полученных результатов.

Ключевые слова: оптимизация, муравьиные алгоритмы, мультиверсионное программное обеспечение.

ANT ALGORITHM MODIFICATION FOR THE PROBLEM OF MULTIVERSION SOFTWARE FORMATION

I. V. Kovalev, M. V. Karaseva, E. V. Solovyev

Siberian State Aerospace University named after academician M. F. Reshetnev
31, Krasnoyarsky Rabochy Av., Krasnoyarsk, 660014, Russian Federation
E-mail: blackdeathangel@rambler.ru

Multiversion software is investigated and an optimization problem of its formation is solved. The ant algorithm is considered as a method for the problem of multiversion software formation. This methodology is based on the introduction of software redundancy that can significantly increase the level of the software reliability. The paper conducts the experiments using a standard algorithm; a modification of the algorithm with the repetition of the experiments with the same data is performed. The introduced modifications improve the algorithm that demonstrates the results of the test problem. Although in consequence of changes the calculation speed on one iteration slows down, the speed increase of the algorithm convergence in the field of the optimal solution compensates this deficiency. The comparison of the obtained results is given.

Keywords: optimization, ant algorithm, multiversion software.

В последнее время очень активно идет развитие «естественных алгоритмов», которые представляют собой алгоритмы оптимизации, основанные на природных механизмах принятия решений. Одним из таких алгоритмов является алгоритм муравьиной колонии (алгоритм оптимизации подражанием муравьиной колонии, ant colony optimization – ACO) [1]. Данный алгоритм является продуктом сотрудничества ученых, изучающих поведение социальных насекомых и специалистов в области компьютерных технологий. В основе данного алгоритма лежит поведение муравьев, а точнее их способность к нахождению кратчайших путей до источника пищи.

Муравьиная колония является распределенной системой, и, несмотря на простоту отдельных её представителей, эта система способна решать сложные задачи. Каждый представитель колонии пытается найти кратчайший путь до источника пищи, при это он не может получить доступ к информации, полученной другими представителями колонии, поэтому у них должен быть механизм, который бы позволил объединить их знания. В качестве данного механизма выступает способность муравьев помечать путь с помощью феромона. Если в процессе поиска муравей находит источник пищи, то на обратном пути он пометит свой маршрут феромоном. Другие муравьи при поиске пищи будут опираться в выборе пути на этот сигнал. Чем большим значением феромона будет помечен путь, тем больше вероятность того, что муравей в своем поиске выберет данный маршрут.

Этот механизм самоорганизации и лег в основу алгоритма муравьиной колонии. Основной идей алгоритма стало то, что набор агентов, поведение которых копирует поведение муравьев, объединяется для решения задачи оптимизации. Агенты координируют свою работу с помощью стигмерции (stigmergy),

которая является механизмом непрямого взаимодействия, осуществляющимся посредством изменения общей среды. В случае ACO таким механизмом стали феромоны. Агенты отмечают пройденный путь с помощью феромона, увеличивая шансы данного пути при выборе альтернатив. Для того чтобы алгоритм не скатывался в область локального экстремума, существует такой механизм, как испарение феромона. Данный механизм отвечает за то, чтобы пути, ошибочно выбранные в качестве решения, постепенно теряли свою привлекательность за счет испарения феромона на них; при этом пути, которые предпочли агенты в процессе принятия решений, будут увеличивать свою популярность, что должно привести к тому, что все агенты в конечном счете выберут общее решение.

Рассмотрим общий алгоритм муравьиной колонии.

1. Создаем муравьев. Стартовая точка, куда помещается муравей, зависит от ограничений, накладываемых условиями задачи, потому что для каждой задачи способ размещение муравьёв является определяющим: либо все они помещаются в одну точку, либо в разные с повторениями (без повторений). На этом же этапе задаётся начальный уровень феромона. Он инициализируется небольшим положительным числом для того, чтобы на начальном шаге вероятности перехода в следующую вершину не были нулевыми.

2. Ищем решения. Вероятность перехода из вершины *i* в вершину *j* определяется по следующей формуле:

$$p_{ij}(t) = \frac{\left[\tau_{ij}\right]^{\alpha}\left[\frac{1}{d_{ij}}\right]^{\beta}}{\sum_{j \in allowedNodes}\left[\tau_{ij}\right]^{\alpha}\left[\frac{1}{d_{ij}}\right]^{\beta}}$$

где $\tau_{ij}(t)$ – уровень феромона; d_{ij} – эвристическое расстояние; α и β – константны. При $\alpha = 0$ выбор ближайшего объекта для включения в итоговое решение, наиболее вероятен, т. е. алгоритм становится жадным. При $\beta = 0$ выбор происходит только на основании феромона, что приводит к субоптимальным решениям.

3. Обновляем значения феромона. Уровень феромона обновляется в соответствии с приведённой формулой:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{\substack{k \in ant\ that\ used \\ edge(i,j)}} \frac{Q}{L_k}$$

где ρ – интенсивность испарения; L_k – цена текущего решения для *k*-го муравья; Q – параметр, имеющий значение порядка цены оптимального решения, т. е. Q/L_k – феромон, откладываемый *k*-м муравьем, использующим ребро (i, j).

4. Дополнительные действия. Обычно здесь используется алгоритм локального поиска, однако он может также использоваться и после поиска всех решений.

5. Проверка окончания поиска. В случае если заданные ограничения были выполнены, поиск останавливается, в противном случае возвращаемся на 1 шаг.

Формирование состава мультиверсионного программного (N-version programming, NVP) обеспечения является задачей, для которой применяется муравьиный алгоритм [2; 3]. Методология мультиверсионного программирования является одной из наиболее перспективных и уже положительно зарекомендовавших себя методологий обеспечения высокой надежности и отказоустойчивости программного обеспечения. Данная методология основывается на введении программной избыточности и позволяет существенно повысить уровень надежности программного обеспечения [4].

Мультиверсионность исполнения программных модулей подразумевает независимую генерацию ряда функционально эквивалентных версий каждого модуля в соответствии с исходными спецификациями. Для версий программного модуля, называемых мультиверсиями, предоставляются средства конкурентного исполнения. Входными данными версий одного модуля являются идентичные наборы данных. Результаты же работы мультиверсий могут отличаться ввиду различных причин. Выбор правильного решения из представленного множества результатов происходит в блоке оценки и принятия решения, где определяется корректный результат [5].

Вследствие этого перед проектировщиком встает задача выбора оптимального набора программных компонент с учетом ряда критериев. Так к какому же классу проблем можно отнести данную задачу? Эта задача относится к классу задач о покрытии множества (set covering problem – SCP) [6]. Нам дается $m \times n$ матрица $A = \left[a_{ij}\right]$, в которой все элементы равны 0 или 1. Дополнительно каждой колонке присвоена положительная стоимость b_j. Можно сказать, что колонка *j* покрывает строку *i*, если $a_{ij} = 1$. Цель SCP – выбрать набор колонок с минимальной стоимость, при этом покрыв каждую строку. Обозначим J набор колонок и y_j обозначим бинарную переменную, равную 1, если $j \in J$, и 0 в противном случае. Формально SCP можно определить следующим образом:

$$\min f(y) = \min \sum_{j=1}^{n} b_j y_j ,$$

$$\sum_{j=1}^{n} a_{ij} y_j \geq 1,\ i = 1,\ ...,\ m ,$$

$$y_j \in \{0,1\},\ j = 1,\ ...,\ n ,$$

Для проведения экспериментов мы возьмем алгоритм MAX–MIN Ant System (MMAS), он является одним из наиболее исследованных и одним из наиболее эффективных алгоритмов, относящихся к семейству муравьиных алгоритмов [1]. Основными особенностями данного алгоритма является наличие верхней и нижней границы на значение феромона, а также способа обновления значения феромона. Обновляется и учитывается только лучшее решение. Схема тестовой мультиверсионной программы представлена на рисунке. Параметры версий в каждом модуле приведены в табл. 1.

Таблица 1

Параметры стоимости и надежности для всех версий каждого модуля

	v1	v2	v3	v4	v5	v6	v7	v8	v9	v10
Module 1	C = 54 R = 0,6	C = 58 R = 0,66	C = 62 R = 0,698	C = 66 R = 0,73	C = 70 R = 0,75	C = 74 R = 0,767	C = 78 R = 0,778	C = 82 R = 0,788	C = 86 R = 0,795	C = 90 R = 0,8
Module 2	C = 22 R = 0,51	C = 24 R = 0,52	C = 26 R = 0,53	C = 28 R = 0,54	C = 30 R = 0,55	C = 32 R = 0,56	C = 34 R = 0,57	C = 36 R = 0,58	C = 38 R = 0,59	C = 40 R = 0,6
Module 3	C = 85 R = 0,6	C = 90 R = 0,61	C = 95 R = 0,625	C = 100 R = 0,65	C = 105 R = 0,68	C = 110 R = 0,74	C = 115 R = 0,81	C = 120 R = 0,865	C = 125 R = 0,89	C = 130 R = 0,9
Module 4	C = 31 R = 0,5	C = 32 R = 0,52	C = 33 R = 0,54	C = 34 R = 0,56	C = 35 R = 0,58	C = 36 R = 0,585	C = 37 R = 0,59	C = 38 R = 0,595	C = 39 R = 0,6	C = 40 R = 0,605
Module 5	C = 52 R = 0,5	C = 54 R = 0,52	C = 56 R = 0,54	C = 58 R = 0,56	C = 60 R = 0,58	C = 62 R = 0,61	C = 64 R = 0,64	C = 66 R = 0,67	C = 68 R = 0,7	C = 70 R = 0,73
Module 6	C = 35 R = 0,6	C = 40 R = 0,64	C = 45 R = 0,68	C = 50 R = 0,7	C = 55 R = 0,72	C = 60 R = 0,74	C = 5 R = 0,76	C = 70 R = 0,79	C = 75 R = 0,82	C = 80 R = 0,85
Module 7	C = 33 R = 0,6	C = 36 R = 0,63	C = 39 R = 0,65	C = 42 R = 0,66	C = 45 R = 0,665	C = 48 R = 0,67	C = 51 R = 0,69	C = 54 R = 0,71	C = 57 R = 0,75	C = 60 R = 0,82
Module 8	C = 72 R = 0,4	C = 74 R = 0,47	C = 76 R = 0,52	C = 78 R = 0,56	C = 80 R = 0,595	C = 82 R = 0,625	C = 84 R = 0,645	C = 86 R = 0,66	C = 88 R = 0,67	C = 90 R = 0,68
Module 9	C = 41 R = 0,5	C = 42 R = 0,51	C = 43 R = 0,53	C = 44 R = 0,57	C = 45 R = 0,62	C = 46 R = 0,69	C = 47 R = 0,75	C = 48 R = 0,77	C = 49 R = 0,79	C = 50 R = 0,8
Module 10	C = 20 R = 0,5	C = 25 R = 0,52	C = 30 R = 0,54	C = 35 R = 0,56	C = 40 R = 0,57	C = 45 R = 0,58	C = 50 R = 0,59	C = 55 R = 0,62	C = 60 R = 0,65	C = 65 R = 0,68

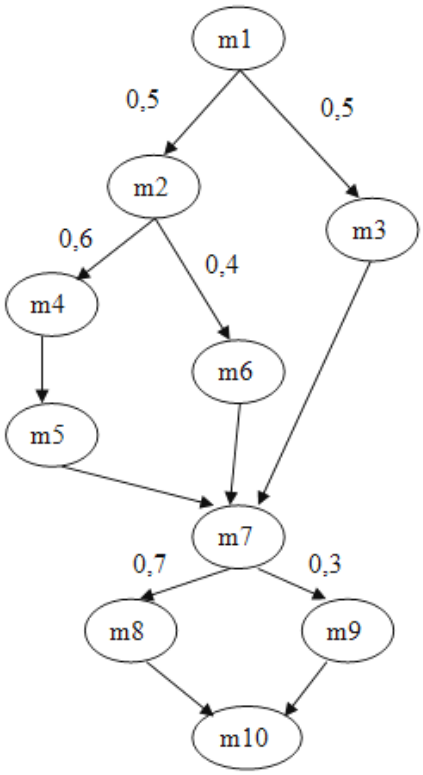

Схема тестового ПО

Для проведения исследования была выполнена программная реализация алгоритма MMAS, которая выглядит следующим образом:

1. Создаем муравьев. Количество муравьев равно количеству модулей ($N = 10$).

2. Для каждой версии выставляется максимальное значение феромона.

3. Создаем минимальное решение. Необходимо, чтобы был задействован каждый модуль. Вероятность выбора версии вычислялась по следующей формуле:

$$p_i = \frac{\tau_i \left[\frac{P_i}{C_i}\right]^{\beta}}{\sum\limits_{\substack{each\,version\\ in\,module}} \tau_i \left[\frac{P_i}{C_i}\right]^{\beta}}$$

4. Вычисляем параметры получившихся решений (P – надежность и C – стоимость). Проверяем, превышены ли ограничения. Если да, то агент считается не удовлетворяющим необходимым условиям, в дальнейшем поиске он не участвует.

5. Пытаемся добавить версию модуля с учетом ограничений. Если ограничения превышены, то агент закончил свой поиск.

6. Проверяем, есть ли еще агенты, не закончившие поиск. Если да, то идем к шагу 5.

7. Сравниваем агентов, которые закончили поиск и удовлетворяют ограничениям с глобальным лучшим решением. В случае нахождения лучшего решения, оно заменяет собой глобальное лучшее решение.

8. Обновляем показатели феромона на версиях. Сбрасываем показатели агентов.

9. Проверяем, превышено ли максимальное количество итераций поиска. Если нет, то идем к пункту 3.

Таблица 2

Результаты эксперимента

Итерация	Параметры
1	Цена: 1609 Надежность: 0,952 Всего версии: 33
198	Цена: 1353 Надежность: 0,954 Всего версии: 25

Была проведена серия экспериментов со следующими параметрами: $P_{\min} = 0,95,\ C \to \min$. Лучший и самый быстрый результат представлен в табл. 2. Затем была проведена серия экспериментов со следующими параметрами $C_{\max} = 1500,\ P \to \max$. Лучший и самый быстрый результат приведен в табл. 3.

Таблица 3

Результаты эксперимента

Итерация	Параметры
1	Цена: 1499 Надежность: 0,938 Всего версии: 25
43	Цена: 1455 Надежность: 0,939 Всего версии: 29
100	Цена: 1463 Надежность: 0,946 Всего версии: 26
181	Цена: 1486 Надежность: 0,961 Всего версии: 29

Как видно из стандартной формулы расчета вероятности выбора версии, там не учитывается вероятность использования модуля, а также количество уже выбранных версий в данном модуле. Были внесены изменения в формулу расчета на 5 шаге, после чего она приобрела следующий вид:

$$p_i = \frac{K_i \tau_i \left[\frac{P_i}{C_i}\right]^{\beta}}{\sum\limits_{\substack{not\,used\\ version}} K_i \tau_i \left[\frac{P_i}{C_i}\right]^{\beta}},$$

$$K_i = G_i \frac{1}{n_i},$$

где G_i – вероятность использования модуля i; n_i – количество уже выбранных версий в модуле i.

Была проведена серия экспериментов со следующими параметрами: $P_{\min} = 0,95,\ C \to \min$. Быстрейший результат представлен в табл. 4.

Таблица 4

Результаты эксперимента

Итерация	Параметры
2	Цена: 1830 Надежность: 0,951 Всего версии: 36
3	Цена: 1601 Надежность: 0,959 Всего версии: 31
5	Цена: 1459 Надежность: 0,953 Всего версии: 27
43	Цена: 1344 Надежность: 0,950 Всего версии: 27
80	Цена: 1340 Надежность: 0,953 Всего версии: 26
536	Цена: 1302 Надежность: 0,952 Всего версии: 25

Как видно из данных, уже на 80-й итерации был достигнут результат, лучший по сравнению со стандартным алгоритмом. Лучший достигнутый результат показан в табл. 5.

Таблица 5

Результаты эксперимента

Шаг	Параметры
1	Цена: 1776 Надежность: 0,954 Всего версии: 34
2	Цена: 1683 Надежность: 0,965 Всего версии: 31
4	Цена: 1521 Надежность: 0,950 Всего версии: 29
8	Цена: 1499 Надежность: 0,951 Всего версии: 28
9	Цена: 1446 Надежность: 0,952 Всего версии: 29
48	Цена: 1438 Надежность: 0,950 Всего версии: 28
166	Цена: 1427 Надежность: 0,953 Всего версии: 27
232	Цена: 1415 Надежность: 0,952 Всего версии: 26
272	Цена: 1396 Надежность: 0,955 Всего версии: 27
287	Цена: 1364 Надежность: 0,950 Всего версии: 26
429	Цена: 1317 Надежность: 0,951 Всего версии: 24
1884	Цена: 1268 Надежность: 0,950 Всего версии: 25

Стоит отметить, что даже самый плохой результат, полученный с помощью модифицированного алгоритма, превосходит самый лучший результат, полученный обычным алгоритмом. Затем была проведена серия экспериментов со следующими параметрами: $C_{\max} = 1500, P \to \max$. Результат, достигнутый максимально быстро, представлен в табл. 6.

Таблица 6

Результаты эксперимента

Итерация	Параметры
1	Цена: 1465 Надежность: 0,946 Всего версии: 26
32	Цена: 1490 Надежность: 0,948 Всего версии: 27
35	Цена: 1440 Надежность: 0,951 Всего версии: 24
85	Цена: 1487 Надежность: 0,954 Всего версии: 28
112	Цена: 1480 Надежность: 0,957 Всего версии: 27
118	Цена: 1487 Надежность: 0,961 Всего версии: 27
364	Цена: 1469 Надежность: 0,962 Всего версии: 27
1142	Цена: 1469 Надежность: 0,963 Всего версии: 30
1552	Цена: 1488 Надежность: 0,964 Всего версии: 27
1651	Цена: 1488 Надежность: 0,964 Всего версии: 27

Как видно, уже на 118-й итерации показатель надежности превышает показатель надежности в аналогичном эксперименте без модификаций. Лучший достигнутый результат показан в табл. 7.

Таблица 7

Результаты эксперимента

Шаг	Параметры
1	Цена: 1435 Надежность: 0,944 Всего версии: 27
20	Цена: 1493 Надежность: 0,948 Всего версии: 26
34	Цена: 1494 Надежность: 0,950 Всего версии: 27
70	Цена: 1494 Надежность: 0,954 Всего версии: 27
80	Цена: 1468 Надежность: 0,955 Всего версии: 27

Окончание табл. 7

Шаг	Параметры
361	Цена: 1496 Надежность: 0,959 Всего версии: 29
406	Цена: 1497 Надежность: 0,963 Всего версии: 29
1278	Цена: 1500 Надежность: 0,964 Всего версии: 28

Итак, внесенные модификации улучшают исходный (базовый) алгоритм муравьиной колонии, что демонстрируют результаты тестовой задачи. Хотя вследствие внесенных изменений скорость расчета на одной итерации несколько замедляется, увеличение скорости нахождения оптимального решения (время схождения на тестовой задаче при использовании модификации до уровня надежности 0,95 было меньше на 79 %) компенсирует данный недостаток.

Библиографические ссылки

1. Dorigo M., Stützle T. Ant colony optimization / The MIT Press. Cambridge, 2004.

2. Ковалев И. В. [и др.] Использование метода роя частиц для формирования состава мультиверсионного программного обеспечения // Приборы и системы. Управление, контроль, диагностика. 2013. № 3. С. 1–6.

3. Ковалев И. В., Царев Р. Ю., Прокопенко А. В., Соловьев Е. В. К вопросу реализации муравьиного алгоритма при выборе состава мультиверсионного программного обеспечения информационно-управляющих систем // Приборы и системы. Управление, контроль, диагностика. 2012. № 2. С. 1–4.

4. Ковалев К. В., Слободин М. Ю., Ступина А. А. Математическая постановка задачи проектирования N-версионных программных систем // Проблемы машиностроения и автоматизации. 2005. № 3. С. 16–23.

5. Ковалев И. В., Новой А. В. Расчет надежности отказоустойчивых архитектур программного обеспечения // Вестник СибГАУ. 2007. № 4. С. 14–17.

6. Lessing L., Dumitrescu I., Stützle T. A Comparison between ACO Algorithms for the Set Covering Problem / Canada Research Chair in Distribution Management. HEC Montreal, 2012.

References

1. Dorigo Marco, Thomas Stützle. Ant colony optimization The MIT Press, Cambridge, Massachusetts, 2004.

2. Kovalev I. V., Solovyev E. V., Kovalev D. I., Bakhmareva K. K., Demish A. V. *Pribori I sistemi. Upravlenie, control I diagnostika*. 2013, no 3, p. 1–6.

3. Kovalev I. V., Tzarev R. Yu., Prokopenko A. V., Solovyev E. V. *Pribori I sistemi. Upravlenie, control I diagnostika*. 2012, no 2, p. 1–4.

4. Kovalev I. V., Slobodin M. Yu., Stupina A. A. *Problemi mashinostroeniya I avtomatizatsii.* 2005, no. 3, p. 16–23.

5. Kovalev I. V., Novoi A. V. *Vestnik SibGAU*. 2007, no. 4, p. 14–17.

6. Lucas Lessing, Irina Dumitrescu, Thomas Stützle. A Comparison between ACO Algorithms for the Set Covering Problem, Canada Research Chair in Distribution Managment, HEC Montreal, 2012.

УДК 62-506.1

ОБ ИССЛЕДОВАНИИ КОМПЬЮТЕРНОЙ СИСТЕМЫ ДИАГНОСТИКИ ЭЛЕКТРОРАДИОИЗДЕЛИЙ НА ОСНОВЕ ДАННЫХ ИСПЫТАНИЙ

Н. В. Коплярова[1], В. И. Орлов[2]

[1]Сибирский государственный аэрокосмический университет имени академика М. Ф. Решетнева
Российская Федерация, 660014, г. Красноярск, просп. им. газ. «Красноярский рабочий», 31
E-mail: aaa@mail.sibsau.ru
[2]ОАО «Испытательно-технический центр – НПО ПМ»
Российская Федерация, 662970, г. Железногорск Красноярского края, ул. Молодежная, 20
E-mail: itcnpopm@atomlink.ru

Предлагается алгоритм оценивания качества классификации в случае отсутствия информации о реальном расположении классов (отсутствие обучающей выборки). Алгоритм основан на том, что моделируется выборка, максимально подобная реальным данным. Таким образом, решается актуальная задача оценки качества классификации данных на основе адекватных моделей. Классификация проводилась на данных, представляющих собой показатели качества транзисторов. Рассматривается алгоритм группировки (оценки качества) электрорадиоизделий (ЭРИ) по данным испытаний. Формулируется также принцип исследования ЭРИ средствами компьютерного моделирования. При этом моделирование процесса кластеризации осуществляется в условиях, максимально приближенных к реальности. Приводятся результаты численных исследований.