ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 38 за 2023 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

В.А. Судаков, Ю.П. Титов, Т.В. Сивакова, П.М. Иванова

Применение метода муравьиных колоний для поиска рациональных значений параметров технической системы

Рекомендуемая форма библиографической ссылки: Применение метода муравьиных колоний для поиска рациональных значений параметров технической системы / В.А. Судаков [и др.] // Препринты ИПМ им. М.В.Келдыша. 2023. № 38. 18 с. https://doi.org/10.20948/prepr-2023-38
https://library.keldysh.ru/preprint.asp?id=2023-38

О р д е н а Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В. Келдыша
Р о с с и й с к о й а к а д е м и и н а у к

В.А. Судаков, Ю.П. Титов, Т.В. Сивакова, П.М. Иванова

Применение метода муравьиных колоний для поиска рациональных значений параметров технической системы

Москва — 2023

Судаков В.А., Титов Ю.П., Сивакова Т.В., Иванова П.М.

Применение метода муравьиных колоний для поиска рациональных значений параметров технической системы

Работа посвящена задачам улучшения поиска рациональных наборов параметров сложной технической системы. Задание наборов параметров осуществляется путем указания границ значений параметров и шага варьирования, а наборы конкретных значений образуют решение. Рассматривается подход с возможностью направленного поиска решения, удовлетворяющего лицо, принимающее решение. Если такое решение не найдено, алгоритм обеспечит полный перебор всех возможных решений. Для управления процессом остановки и взаимодействия с вычислителем используется многопоточное сокетное соединение. Предложены: модификация метода муравьиных колоний с применением хэш-таблицы; новая формула вероятностного выбора вершин. Исследование проводилось на однокритериальных задачах и бенчмарках. Полученные модификации позволяют находить все решения без применения мультистарта, сохраняя преимущество метода муравьиных колоний – быстрый поиск рациональных решений.

Ключевые слова: метод муравьиных колоний, перебор значений параметров, параметрический граф, web-приложение, многокритериальная оптимизация

Vladimir Anatolievich Sudakov, Yuri Pavlovich Titov, Tatiana Vladimirovna Sivakova, Ivanova Polina Mihailovna

Application of the ant colony method to search for rational values of the parameters of a technical system

The work is devoted to the problems of improving the search for rational sets of parameters of a complex technical system. Setting the parameter sets is carried out by specifying the boundaries of the parameter values and the variation step, and the sets of specific values form the solution. An approach with the possibility of a directed search for a solution that satisfies the decision maker is considered. If such a solution is not found, the algorithm will provide a complete search of all possible solutions. A multi-threaded socket connection is used to control the process of stopping and interacting with the calculator. Suggested: modification of the ant colony method using a hash table; new formula for probabilistic choice of vertices. The study was conducted on single-criteria tasks and benchmarks. The resulting modifications make it possible to find all solutions without using a multistart, leaving the advantages of the ant colony method: a quick search for rational solutions.

Key words: ant colony method, enumeration of parameter values, parametric graph, web application, multiobjective optimization

Введение

Достаточно часто для технической системы, инженер-специалист через специальный графический интерфейс изменяет значения параметров для поиска рациональных решений. Результатом такой работы являются значения критерия (чаще всего векторного), полученные с помощью вычислений или визуальной демонстрации. Например, для авиационных систем параметры фюзеляжа и крыла могут определять коэффициенты лобового сопротивления и величину подъемной силы. Для определения оптимальных значений параметров работа с такими системами становится итерационной. Каждый раз после получения значений критериев инженер-специалист изменяет значения параметров до тех пор, пока его не удовлетворяют значения критериев. При появлении больших вычислительных мощностей, суперкомпьютеров и кластеров задачи по определению значений критериев могут быть распараллелены на множество процессов, потоков или вычислителей. В подобных системах задаются границы изменения значений параметров и, как правило, шаг изменения. По результатам запуска вычислений с разными значениями параметров определяются значения критерия, среди которых выбираются рациональные или оптимальные (их может быть несколько). Конкретный набор значений параметров является решением, а набор, соответствующий оптимальному значению критерия, – оптимальным решением. Для задачи поиска оптимального набора параметров используют либо алгоритмы оптимизации, сходящиеся к одному рациональному решению, либо системы полного перебора. В случае многокритериальной задачи (векторного критерия), асинхронности работы вычислительного кластера и неточностей модели сходимость алгоритмов к одному набору значений параметров маловероятна.

Метод муравьиных колоний предложен в 1992 году [1] для решения задачи коммивояжера. Современные разработки в области муравьиных колоний [2-3] уже решают задачи подбора параметров для решателей на основе нейронных сетей и систем машинного обучения. Существуют системы, позволяющие решать задачи кластеризации и классификации с применением модификаций методов муравьиных колоний [4-7]. Современные исследования позволяют применять метод муравьиных колоний для непрерывной оптимизации. Поиск методами CACO (ContinuousAntColonyOptimization – непрерывная оптимизация колонией муравьев) [8], ACOR (Ant Colony Optimization for continuous domain) [9] и CIAC (ContinuousInteractingAntColony – непрерывно взаимодействующая колония муравьев) [10] не предполагает использования графа и активно исследовался, в том числе и российскими исследователями [11, 12].

Очень близкими являются системы оптимизации гиперпараметров, которые определяют рациональные значения параметров алгоритмов. Существуют различные решения оптимизации гиперпараметров, используемые для машинного обучения [13-14]. Среди алгоритмов используют поиск по решетке, случайный поиск, байесовскую оптимизацию, эволюционные методы и т. д. Наиболее близкими являются системы, основанные на байесовском

оптимизаторе, например IBM Bayesian Optimization Accelerator (BOA). Решения на основе BOA используют метаэвристики для создания множества гипотез. Данные метаэвристики являются коммерчески закрытыми и не подлежат анализу, в отличие от открытости и легкости анализа метода муравьиных колоний.

В работе рассматривается применение модификаций метода муравьиных колоний для направленного перебора решений. Для работы метода муравьиных колоний используется специальный, «параметрический» граф. Данный граф состоит из вершин, определяющих конкретные значения конкретных параметров, собранные в слои для параметра. Путь в таком графе будет определять конкретные значения каждого параметра. Поиск в пути в графе методом муравьиных колоний занимает одинаковое время, и при этом не используются сложные вычисления, что обеспечивает минимальную задержку для решения. Метод работает совместно с пользователем. Вероятностное распределение наилучших значений параметров позволяет рассмотреть рациональные решения как можно раньше (по отношению ко всему множеству решений). Но если пользователя (инженера-специалиста) не удовлетворят полученные решения, то метод муравьиных колоний рассмотрит все возможные решения. Таким образом, модификации метода муравьиных колоний позволяют решать две противоположные задачи:

1. Поиск рационального набора значений параметров (решения) на ранних итерациях метода.
2. Если пользователь не останавливает работу программы, то необходимо перебрать все возможные комбинации значений параметров.

1. Модели и методы

Параметрический граф

Пусть N – число работ (считаем, что каждый проект состоит из конечного числа работ). Для решения задачи перебора решений необходим специальный «параметрический граф». Структура параметрического графа предложена в работе, вместе с тем подобные, близкие структуры графов широко применялись в исследованиях метода муравьиных колоний [2-5]. Параметрический граф состоит из групп вершин (слоев) [15-17]. Каждый слой определяет один параметр системы, а каждая вершина – конкретное значение данного параметра (рис. 1). Каждая вершина одного слоя соединена дугами с каждой вершиной следующего слоя. В результате путь в таком графе определяет конкретные значения для каждого критерия, т. е. одно решение. Феромон в таком графе заносится не на дуги, как в оригинальном алгоритме, а на вершины. Чем лучше значение критерия определенного решения, тем больше феромона будет на вершинах, определяющих значения параметров в данном решение и тем больше будет вероятность выбора данного значения параметра.

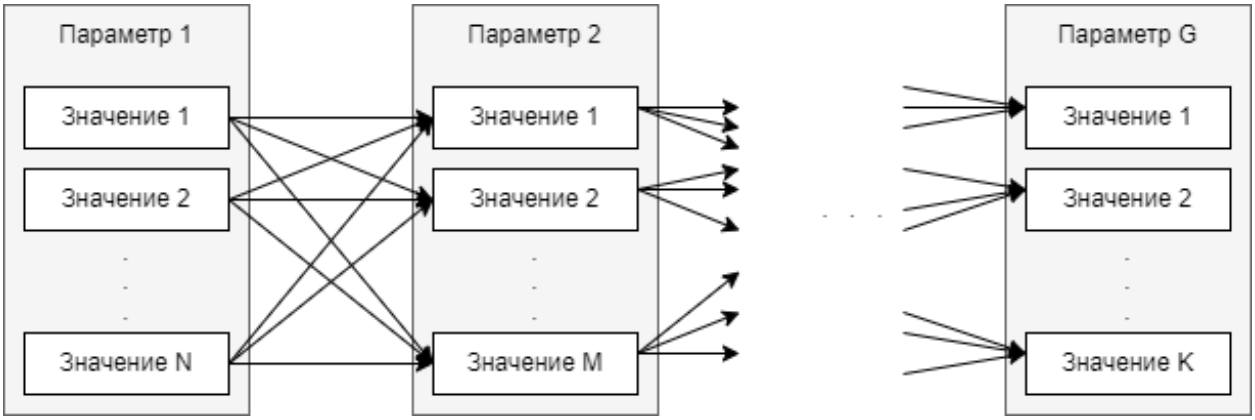

Рис. 1. Структура параметрического графа

Хэш-таблица

Для определения уже рассмотренного решения используется хэш-таблица, в которой хранятся решения. После вычисления решения муравьем-агентом оно проверяется в хэш-таблице. Если его в таблице нет, то можно отправлять на вычислитель и заносить феромон согласно полученному значению критерия. Но если такое решение уже рассмотрено на вычислителе, то необходимо взять значение критерия из хэш-таблицы. Дальнейшие действия агента могут различаться. Рассмотрены следующие алгоритмы [18-19]:

1. ACOCN (ACO Cluster New) – использовать значение целевой функции из хэш-таблицы, занести веса как в оригинальном алгоритме.
2. ACOCNI (ACO Cluster New Ignor) – игнорировать агента. Агент не заносит веса на параметрический граф.
3. ACOCCyN (ACO Cluster Cycle Number) – повторный поиск нового, еще не рассмотренного на вычислителе, решения методом муравьиных колоний с ограничением на количество итераций. Если за установленное количество итераций не найдено новое решение, то агент игнорируется.
4. ACOCCyI (ACO Cluster Cycle Infinity) – повторный поиск нового, еще не рассмотренного на вычислителе, решения методом муравьиных колоний неограниченным количеством итераций. Ограничение в пункте №3 может решить проблему стагнации.
5. ACOCT (ACO Cluster Tree) – повторный поиск нового решения другим алгоритмом. Рассматривалась возможность обхода параметрического графа как дерева.

Для того чтобы метод муравьиных колоний не сходился к одному решению, обычно применяют мультистарт [10,12] – многократный запуск с различными начальными условиями. За счет управления действиями агентов, нашедших решения из хэш-таблицы, можно избежать мультистарта и позволить алгоритму искать только новые решения.

Модификации вероятностной формулы

Кроме предложенных методов рассмотрены различные формулы вероятностного выбора следующей вершины. Оригинальная формула содержит множитель, отвечающий за длину дуги [1]. И, хотя в программном обеспечении имеется возможность установить пользовательские веса на значения параметров, данная функция не является основной, и исследование проводилось без учета предпочтений пользователя. При наличии в формуле только множителя, отвечающего за веса в вершинах параметрического графа, алгоритм быстро начинает стагнировать, сходиться к первому найденному хорошему решению [19]. Рассматривались как мультипликативные, так и аддитивные свертки различных значений, вычисленных на графе. Наилучшей формулой определения вероятности перехода в вершину является

$$P_{ij,k}(t) = \frac{k1*\mu(t)^{\alpha}_{norm,i,j} + k2*(1/kol(t)_{i,j})^{\beta} + k3*(kol(t)_{i,j}/MaxKol_{i,t})^{\gamma}}{\sum_{z\in J_{i,k}}(k1*\mu(t)^{\alpha}_{norm,i,j} + k2*(1/kol(t)_{i,j})^{\beta} + k3*(kol(t)_{i,j}/MaxKol_{i,t})^{\gamma})} \quad . \qquad (1)$$

Формула (1) представляет собой линейную свертку трех слагаемых и весовых коэффициентов. Первое слагаемое, $k1 * \mu(t)^{\alpha}_{norm,i,j}$, определяется количеством весов у i-й вершины j-го параметра (слоя параметрического графа) на итерации t. Для применения данного параметра во взвешенной сумме необходимо использовать нормализованное значение весов в вершинах. Второе слагаемое, $k2 * (1/kol(t)_{i,j})^{\beta}$, определяется количеством посещений агентами i-й вершины j-го параметра на итерации t. Данное слагаемое увеличивает вероятность посетить вершину, которая редко присутствует в решениях, и позволяет избегать стагнации на ранних итерациях метода муравьиных колоний. В третьем слагаемом, $k3 * (kol(t)_{i,j}/MaxKol_{i,t})^{\gamma}$, учитывается максимальное количество возможных посещений вершины – значения $MaxKol_{i,t}$ для параметра j на итерации t. Так как в параметрическом графе на каждом слое необходимо выбрать одну вершину (одно значение параметра), то можно вычислить общее количество решений, наборов значений параметров. Общее количество решений может быть вычислено как произведение количества вершин в каждом слое. Максимальное количество решений, в которых может содержаться конкретная вершина параметрического графа, вычисляется как отношение общего количества решений и количества вершин в данном слое, т. е. для каждой вершины слоя j значение $MaxKol_{i,t}$ будет одинаковым. Третье слагаемое на поздних итерациях позволяет увеличить вероятность выбора вершины, для которой рассмотрены большинство вариантов решений. Если для вершины рассмотрены все возможные решения, то из вероятностного поиска данную вершину можно исключить. Аддитивная свертка позволяет компенсировать значения слагаемых. Вершины с большим количеством весов и частыми посещениями (часто рассматриваемые вершины) могут быть компенсированы

вершинами с небольшим количеством посещений (редкие вершины) или вершинами, для которых рассмотрены практически все решения.

Также предложены и проанализированы другие небольшие модификации алгоритма, например, возврат параметрического графа в исходное состояние, игнорирование некоторых вершин и другие [18].

Взаимодействие с пользователем и вычислителем

Работа метода муравьиных колоний становится бесполезной, если будут рассмотрены все решения. Предложенный метод фактически только переупорядочивает решения, отправляемые на вычислительный кластер, не сходясь к оптимальному. Поэтому важной составляющей работы системы является взаимодействие с пользователем. Для взаимодействия с пользователем разработан web-интерфейс (сайт), позволяющий задать границы изменений значений параметров, подключиться к определенному вычислителю или программному продукту и запустить работу системы. Разработанное приложение само будет формировать набор значений параметров и отправлять на вычислитель, получая от него значения критериев. Результаты работы системы параллельно выводятся пользователю, позволяя остановить работу системы. В результате при непосредственном взаимодействии с пользователем разработанная система позволяет экономить ресурсы вычислителя. Но в случае отсутствия управления от пользователя будут рассмотрены все решения.

Взаимодействие с пользователем (вывод текущего состояния работы системы; ожидание команды остановки от пользователя) реализовано через многопоточную систему. В одном потоке происходит обработка информации от front-end приложения (с ним взаимодействует пользователь), а в другом – работа метода муравьиных колоний. При поступлении сигнала от пользователя возможна остановка работы второго потока для остановки работы перебора решений и взаимодействия с вычислителем. Взаимодействие с вычислительным кластером реализовано через сокеты для универсализации подхода. В результате при остановке работы метода муравьиных колоний необходимо закрыть все сокетные соединения с вычислителем. Сокетное взаимодействие предоставляет универсальный механизм, позволяющий как установить вычислитель на одной ЭВМ, так и иметь доступ к удаленному вычислителю. Кроме того, обеспечивается асинхронное многопоточное взаимодействие с вычислителем путем создания множества сокетных соединений.

Для хранения хэш-таблицы большого объема может не хватать оперативной памяти. В реализации предложено параллельное занесение информации о решениях в базу данных в отдельном потоке. Размер хэш-таблицы регулируется определенным фиксированным размером. Вытеснение страниц осуществляется по принципу FIFO, но возможны и другие, более эффективные методы. Если решение не найдено в хэш-таблице в памяти, то необходимо дождаться проверки наличия решения в базе данных и, только если в ней решение не будет найдено, отправлять запрос на вычислитель.

2. Эксперимент

Параметрические графы

Исследования предложенных методов проводились на параметрических графах различной размерности. Для исследования возможности нахождения всех решений и эффективности работы метода рассматривались бенчмарки [20]. Тестирование и графики будут приведены для бенчмарка «Carrom table function» с двумя параметрами x1 и x2, изменяющимися в пределах от -10 до 10 (2). При шаге 0,1 для обоих параметров в параметрическом графе будет 40401 решение. На таком параметрическом графе исследуем работу предлагаемых методов при поиске всех решений. Исследуется количество дополнительных итераций и скорости работы алгоритма для поиска последних решений.

$$f(\mathrm{x}) = -(\cos(\mathrm{x}_1)\cos(\mathrm{x}_2)\mathrm{e}^{\left|100-\sqrt{\mathrm{x}_1^2+\mathrm{x}_2^2}\right|})^2 \Big/ 30\,;\ \mathrm{x}_1, \mathrm{x}_2 \in [-10..10] \quad (2)$$

Исследование работы метода проводилось и на параметрическом графе большой размерности, представленном на рисунке 2 и имеющем более 10^9 вариантов наборов параметров. Такой граф имеет как числовые, так и качественные значения параметров. При этом существует слой «Нагрузка» с одним значением параметра, а также слой «Решение», который никак не влияет на значение целевой функции. Для слоя «Давление», состоящего из качественных значений параметра, влияние на целевую функцию определяется связками Если-То. Подобные графы более характерны для реальных систем, но их исследование затруднено большими размерностями. На данном графе произведем исследование скорости нахождения оптимального (его при наличии целевой функции можно определить заранее) решения. Для анализа эффективности работы предлагаемых методов вычислялись следующие характеристики:

1. Общее количество рассмотренных путей агентов (решений). Повторный найденный путь требует вычислительной мощности, но не ускоряет перебор всех решений (возможно, ускоряет поиск оптимального).
2. Количество дополнительных итераций для методов ACOCCyN, ACOCCyI и ACOCT.
3. Номер решения, на котором найдено оптимальное значение критерия.
4. Количество прогонов, на которых найдено оптимальное решение.
5. Количество найденных оптимальных решений.

Кроме оптимального решения оцениваются также 99.99% от оптимального 99.9% от оптимального и т. д.

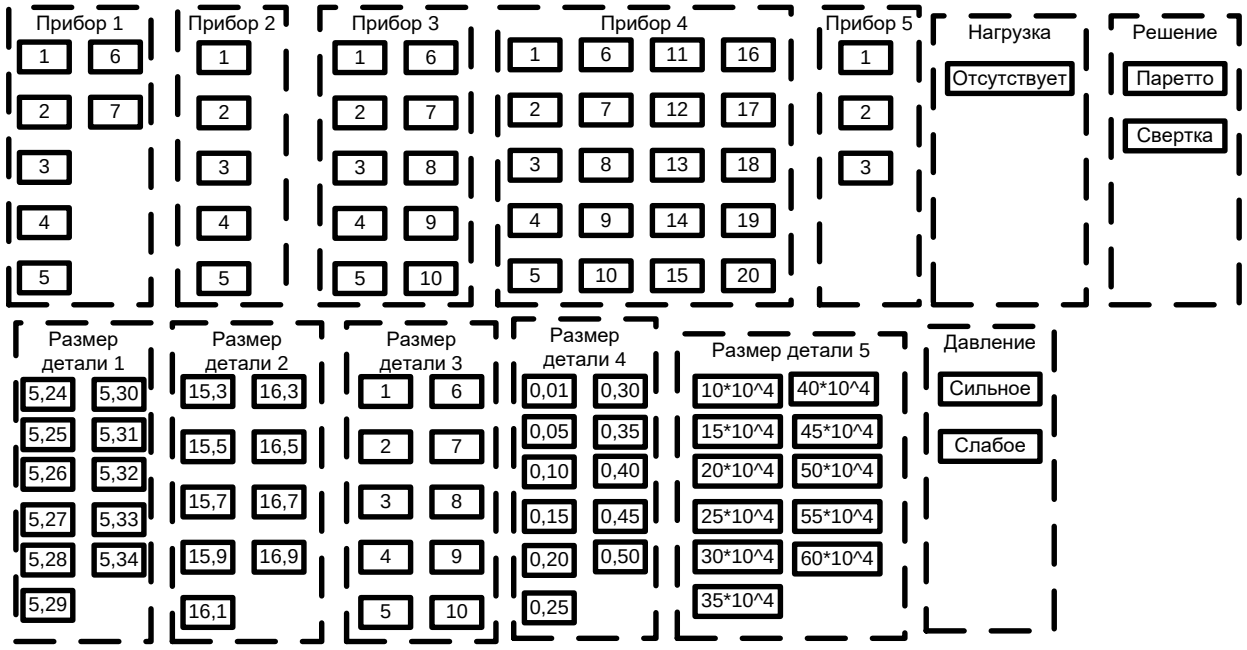

Рис. 2. Параметрический граф для исследования работы предлагаемых модификаций алгоритма

Результаты исследования бенчмарка «Carrom table function»

Так как при рассмотрении бенчмарка «Carrom table function» возможно рассмотрение всех решений для двухпараметрической задачи и возможен вывод целевой функции в виде трехмерного графика, то интересно рассмотреть номер итерации, на которой были рассмотрены конкретные значения параметров x1 и x2 (точка на трехмерном графике). Исследовалась модификация алгоритма ACOCCyI, которая при нахождении агентом решения из хэш-таблицы проводила повторный поиск еще не рассмотренного решения для данного агента методом муравьиных колоний. Проводились 3000 запусков системы (прогонов), и вычислялась оценка математического ожидания номера решения, на котором рассмотрена данная точка. График функции «Carrom table function» и график оценки математического ожидания номера решения (рис. 3) близки по форме. Можно заметить, что все 4 оптимальных значения целевой функции найдены на ранних итерациях (оценочно). Модификация алгоритма не сходится к одному решению, а продолжает поиск других рациональных решений, позволяя в одном прогоне найти все оптимумы на ранних итерациях. Так как пользователю выдаются текущие рассмотренные значения параметров и критериев, то возможна остановка работы алгоритма как при первом оптимуме, так и при последующем рассмотрении рациональных решений.

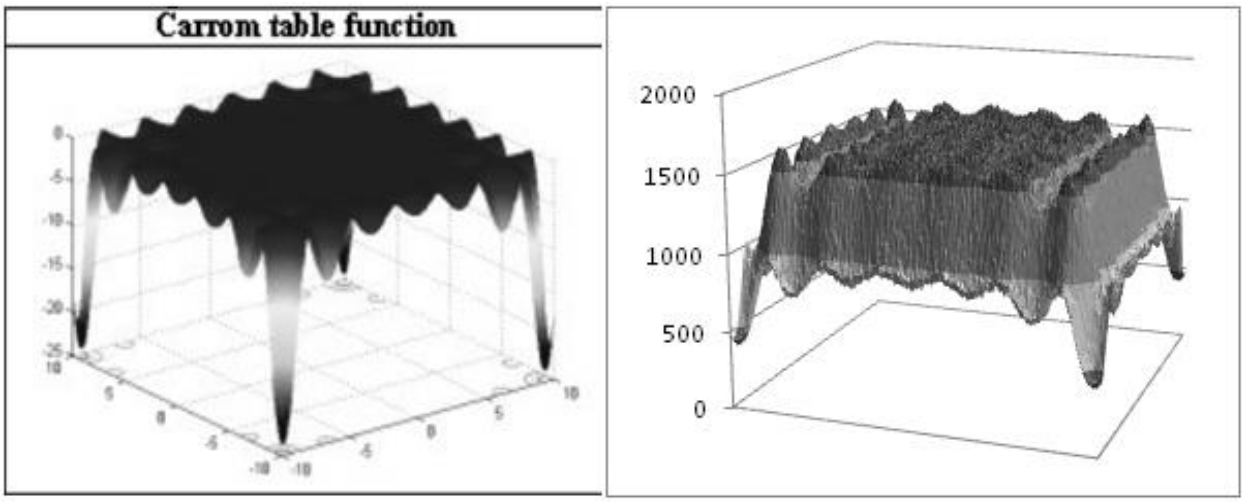

Рис. 3. Исследование эффективности работы метода муравьиных колоний

Приведем результаты анализа модификации ACOCCyI для поиска всех решений. Метод ACOCCyI позволяет вычислить все решения, находя для каждого агента новое решение. При 25 агентах на одной итерации метода муравьиных колоний потребуется 1936 итераций метода, а вот количество дополнительных итераций для одного нулевого агента (если агент нашел уже рассмотренное решение) может быть различным. На рисунке 4 показаны результаты работы метода при различных коэффициентах k2 и k3 в формуле (1) с возможностью игнорирования (исключения из рассмотрения при вероятностном переходе) вершин, для которых найдены все решения. Коэффициенты k2=0 и k3=0 у графиков 1 и 2, у графиков 3 и 4 – k2=0 и k3=1, у графиков 5 и 6 – k2=1 и k3=0, а у графиков 7 и 8 – k2=1 и k3=1. Нечетные номера 1, 3, 5 и 7 – модификации с применением игнорирования вершин. На рисунках по горизонтальной оси отложены итерации метода муравьиных колоний, а статистика собиралась по результатам 200 прогонов метода.

Наименьшее время поиска всех решений требуют алгоритмы с игнорированием вершин параметрического графа, для которых рассмотрены все решения. При этом игнорирование вершин увеличивает время работы алгоритма при нормальной работе алгоритма. Самым эффективным является алгоритм, использующий все слагаемые формулы (1), график 7, представленный на рис. 4. График 9 (рис. 4) – работа алгоритма ACOCNI, который показывает одинаковое время работы на любом номере итерации.

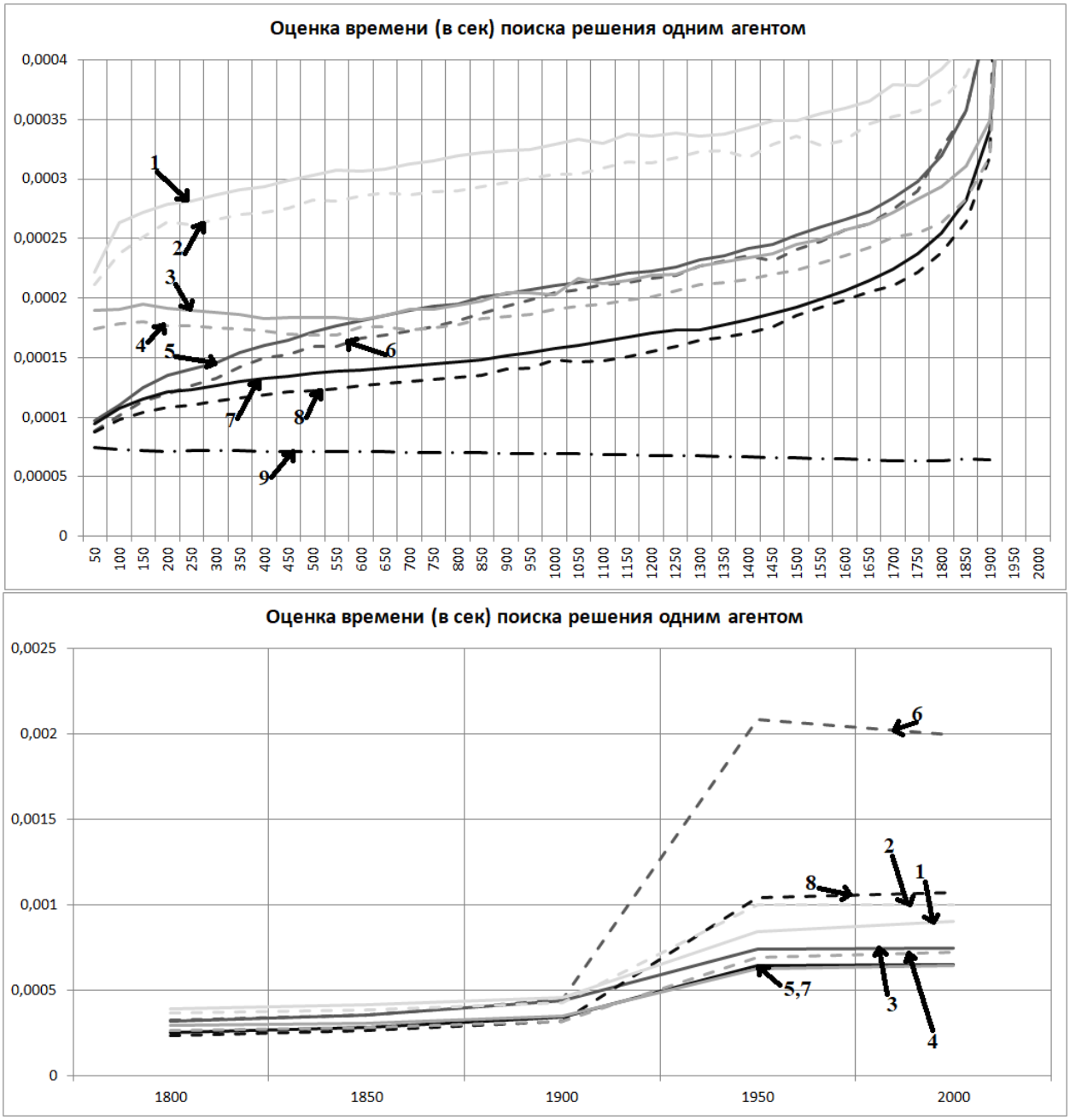

Рис. 4. Графики оценки математического ожидания времени поиска решения одним агентом

На рисунке 5 исследуется необходимое количество дополнительных итераций для агента, нашедшего уже рассмотренное решение (нулевой агент), для нахождения еще не рассмотренного на кластере решения. Форма графиков на рисунке 5 близка к форме на рисунке 4, основные задержки в процессе работы алгоритма связаны с дополнительными итерациями нулевого агента. Следует отметить, что дополнительные итерации нулевого агента отличаются от основных итераций только неизменным состоянием параметрического графа, т. е. алгоритм ACOCNI, в котором просто игнорируются нулевые агенты, тоже рассмотрит все решения. Также видно, что количество дополнительных

итераций практически до последних итераций алгоритма достаточно небольшое, менее 5. В результате использование ограничения на количество дополнительных итераций в алгоритме ACOCCyN не требуется.

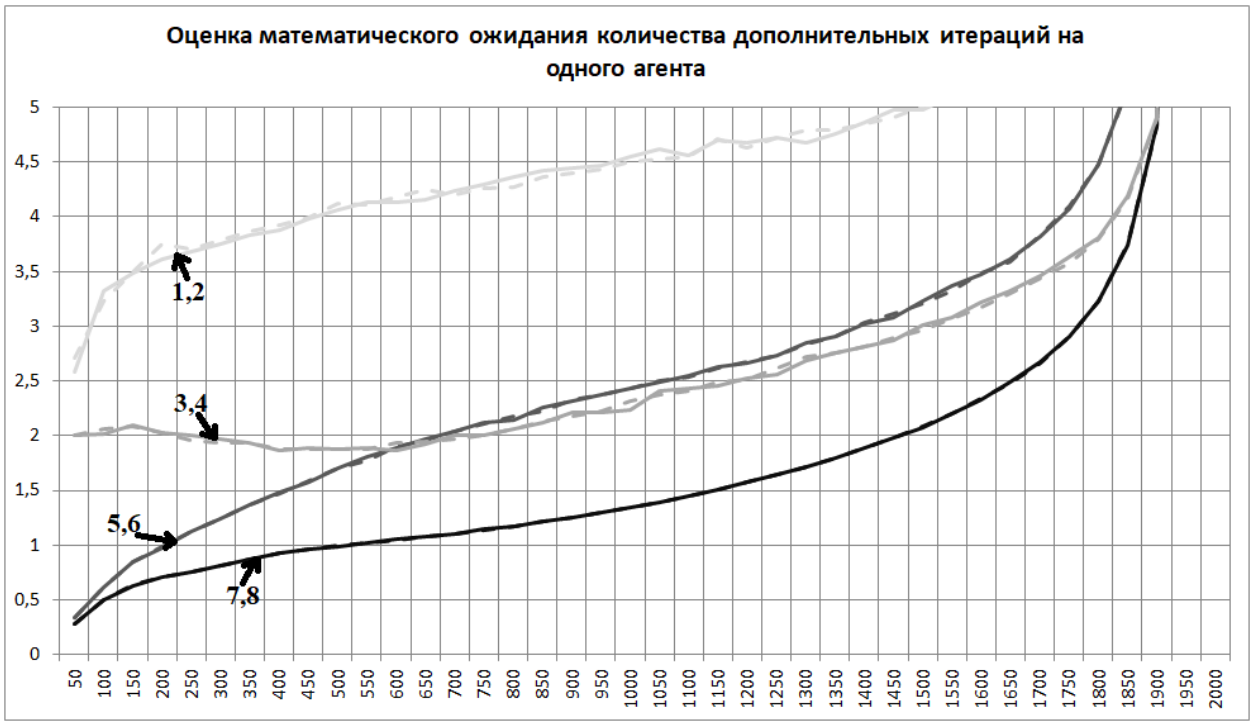

Рис. 5. График оценки математического ожидания количества итераций на одного агента

На рисунке 6 графиками определяется оценка математического ожидания номера решения, на котором было найдено оптимальное решение. Рассматриваются только методы с k2=1 и k3=0 (график 5) или k3=1 (график 7) и игнорированием вершин. При любом количестве итераций оптимальное решение находится после рассмотрения от 0,009% (40401*0,009=363) решений до 0,012% (484) решений – интервал с доверительной вероятностью 0,99. Для нахождения 0,99% оптимального решения требуется в 2 раза меньше рассмотренных решений и наборов значений параметров. Для модификации ACOCNI (график 9) требуется аналогичное количество рассмотренных решений для нахождения оптимального. Остальные алгоритмы находят оптимальное решение позже.

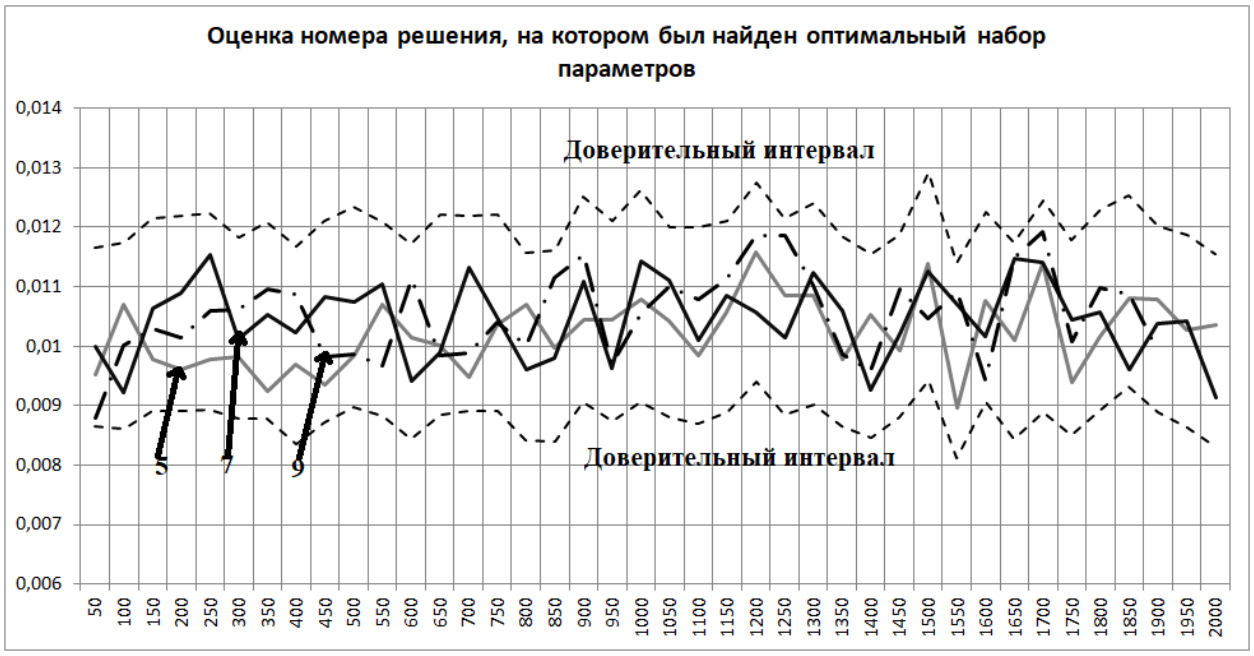

Рис. 6. График оценки математического ожидания номера решения, на котором был найден оптимальный набор параметров

Применение модификации поиска с помощью дерева ACOCT показывает существенно худшие результаты по времени работы, так как требует постоянного рассмотрения графа с начала.

Результаты исследования графа большой размерности

На графе большой размерности исследовались различные модификации метода муравьиных колоний. Исследовались в первую очередь возможность нахождения оптимального решения за ограниченное число итераций и номер итерации, на которой найдено такое решение. На рисунке 7 различными линиями показаны алгоритмы с различными действиями для агента, не нашедшего нового решения. Приписка S означает перевод параметрического графа в начальное состояние, если все агенты на итерации не нашли нового решения. Данная операция похожа на мультистарт (хэш-таблица с рассмотренными решениями сохраняется). На рисунках по горизонтальной оси отложено количество итераций метода муравьиных колоний. Исследование проводилось при 25 агентах на итерации, а статистика собиралась по результатам 200 прогонов метода.

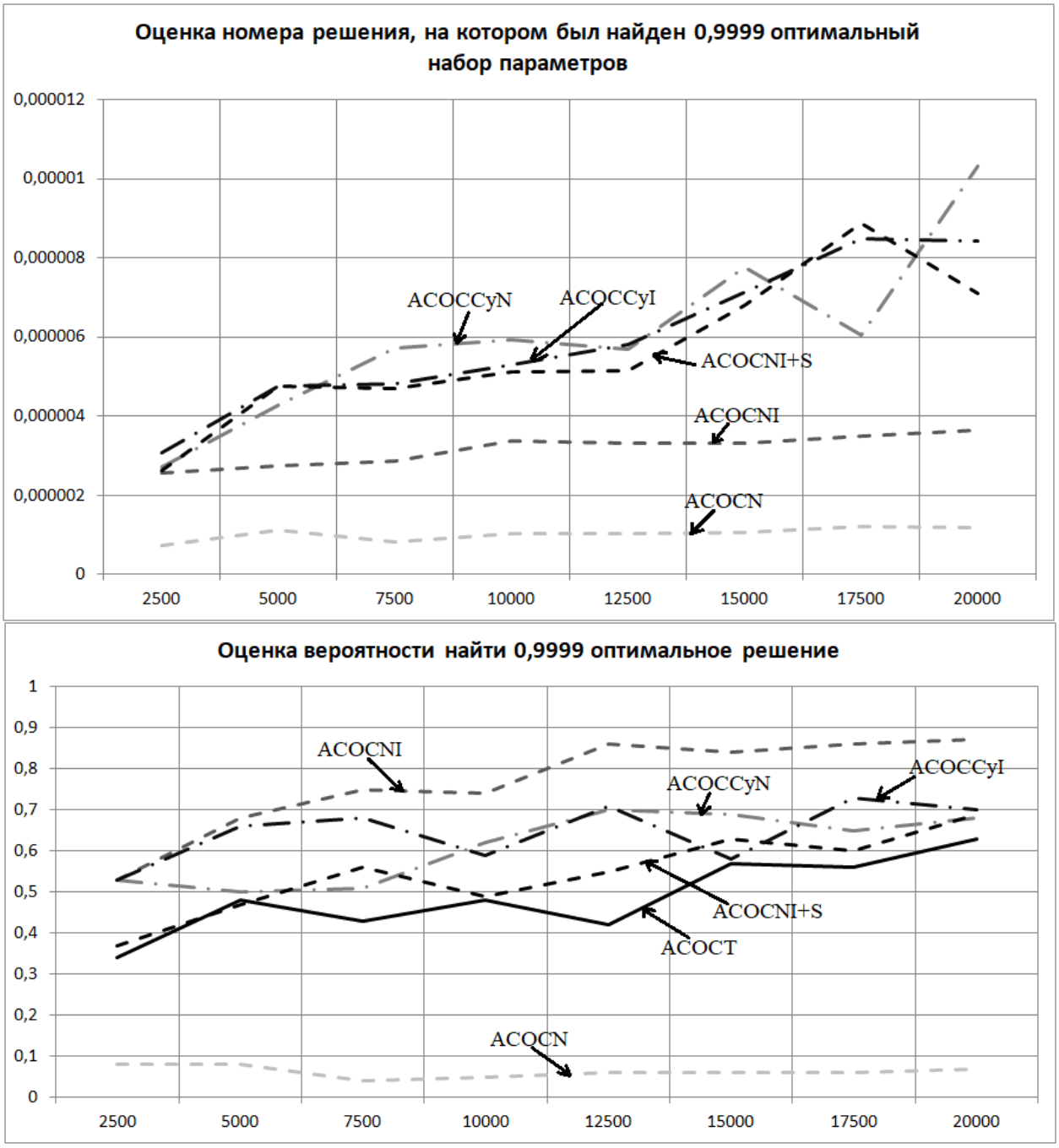

Рис. 7. График оценок математического ожидания и вероятности найти 0,9999% оптимального решения

Раньше всех находит оптимальное решение оригинальный метод муравьиных колоний, в котором агенты заносят веса вне зависимости от того, нашли они новое решение или нет. Агентов, которые не нашли нового решения, будем называть нулевыми. Но при этом данный алгоритм (ACOCN) находит оптимальное решение только в 10% прогонов. Достаточно быстро находит решение и алгоритм ACOCNI, в котором игнорируются нулевые агенты. У данного алгоритма выше вероятность найти оптимальное решение, и для поиска данного решения требуется меньше итераций, чем у других методов.

На рисунке 8 приведен график оценки вероятности нахождения нового решения одним агентом.

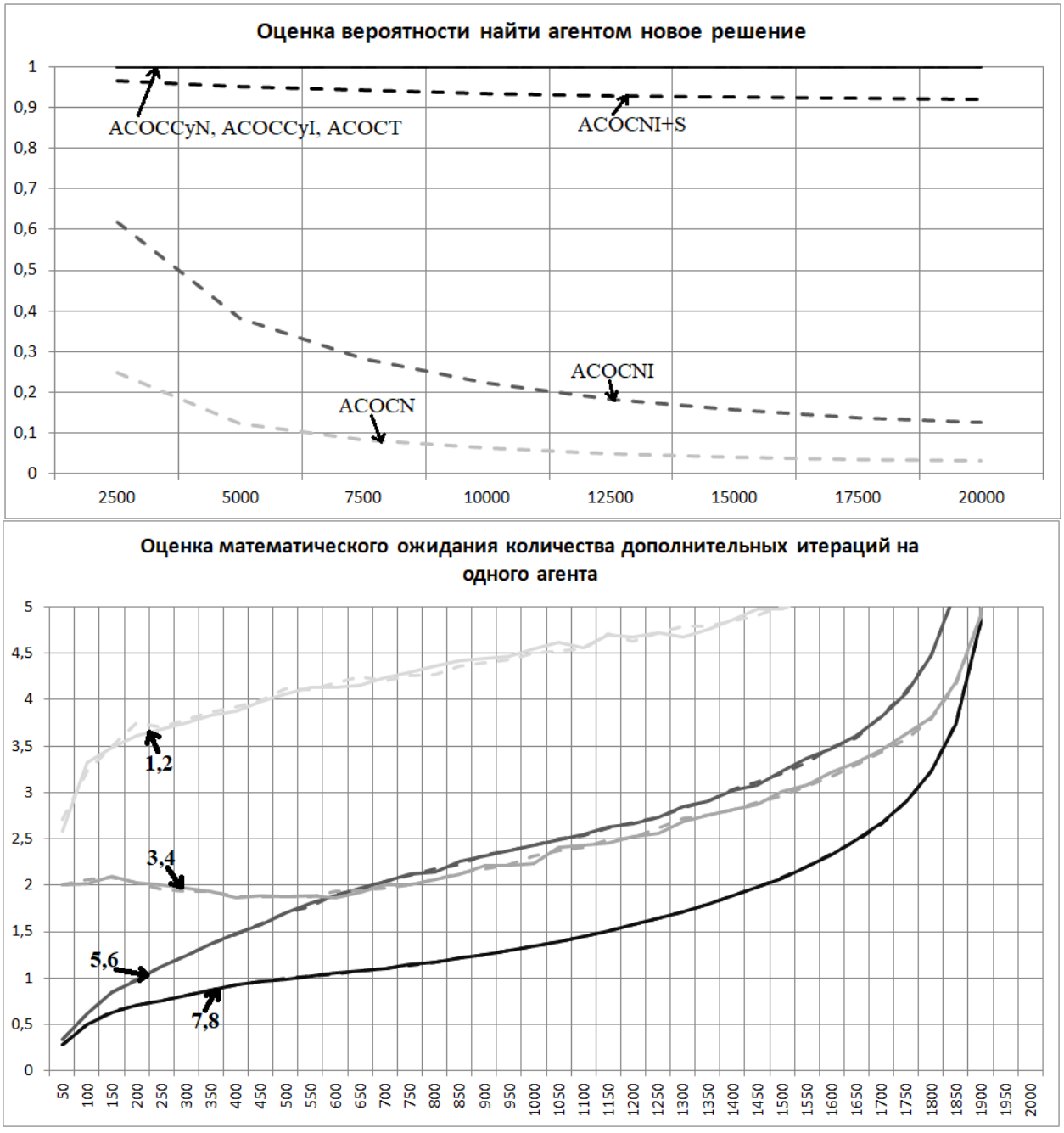

Рис. 8. График оценки вероятности нахождения агентом нового решения

По результатам исследования вероятности нахождения нового решения для каждого агента видно, что вероятность для алгоритма ACOCNI резко уменьшается с увеличением числа итераций, т. е. даже если не давать нулевым агентам заносить веса на вершины параметрического графа, то алгоритм все равно начинает стагнировать, т. е. сходиться к одному решению. Такой алгоритм не может рассмотреть всех решений в параметрическом графе и требует мультистарта (алгоритм ACOCNI+S). Но алгоритм с мультистартом по

эффективности не отличается от алгоритмов повторного циклического поиска (ACOCCyN, ACOCCyI, рис. 7), которые для каждого агента находят новое решение (рис. 8).

3. Заключение

По результатам работы программного обеспечения и проведенных исследований удалось доказать, что предложенные модификации метода муравьиных колоний позволяют рассмотреть все решения в параметрическом графе, не применяя процедуру мультистарта [19]. Предложенные модификации позволяют сократить количество дополнительных итераций и общее время работы алгоритма. При тестировании на бенчмарке «Carrom table function» [20], имеющем 4 оптимальных разнесенных решения, алгоритм находит все 4 значения менее чем за 5% рассмотренных решений. Поиск последних решений требует значительно больше итераций, но добавление третьего слагаемого в вероятностную формулу и отсутствие учета вершин позволяет существенно сократить количество итераций. При этом предложенные модификации позволяют находить оптимальные и рациональные решения на ранних итерациях.

Основным недостатком метода является сложность описания процесса взаимодействия с ним. Если необходимо рассмотреть все решения, то полный перебор будет осуществлен быстрее, так как не требуется работа с параметрическим графом. Но для случая поиска оптимального значения подходы, основанные на байесовском оптимизаторе, генетическом алгоритме и искусственном интеллекте, могут быть быстрее. Необходимость непосредственного взаимодействия с пользователем обычно не рассматривается в данных алгоритмах, но может быть достоинством предлагаемых методов. Другим недостатком является необходимость хранения всех рассмотренных решений в хэш-таблице. При большом количестве решений данная особенность может существенно замедлять работу алгоритма.

Планируется развивать предложенные методы: необходимо рассмотреть вопросы кэширования базы данных, хранящей хэш-таблицу, для более быстрого поиска пути методом муравьиных колоний; рассмотреть применение альтернативных методов поиска новых решений в параметрическом графе. Предполагается на поздних этапах работы алгоритма применять методы, отличные от метода муравьиных колоний.

Также предлагается рассмотреть работу методов для многокритериальной оптимизации и работы с векторным критерием, поиска множества Парето. На данном этапе разработаны способы определения весов для взвешенной суммы критериев при оптимизации на параметрическом графе методом муравьиных колоний [16, 17].

Библиографический список

1. Colorni A., Dorigo M., Maniezzo V. Distributed Optimization by Ant Colonies. // Proc. First Eur. Conf. on Artific. Life, Paris, France, F.Varela and P.Bourgine (Eds.), Elsevier Publishing. pp. 134-142, 1992

2. Dorigo M., Stützle, T.: Ant Colony Optimization // MIT Press, p. 321, 2004

3. Joseph M. Pasia, Richard F. Hartl, Karl F. Doerner. Solving a Bi-objective Flowshop Scheduling Problem by Pareto-Ant Colony Optimization M. Dorigo et al. (Eds.) // ANTS 2006, LNCS 4150, pp. 294–305, 2006

4. Torry Tufteland, Guro Ødesneltvedt, and Morten Goodwin Optimizing PolyACO Training with GPU-Based Parallelization M. Dorigo et al. (Eds.) // ANTS 2016, LNCS 9882, pp. 233–240, 2016. DOI: 10.1007/978-3-319-44427-7 20

5. Parpinelli R., Lopes H., Freitas A.: Data mining with an ant colony optimization algorithm // IEEE Trans. Evol. Comput. 6(4), pp. 321–332, 2002

6. Bremer Jörg and Sebastian Lehnhoff. “Constrained Scheduling of Step-Controlled Buffering Energy Resources with Ant Colony Optimization. // ANTS Conference, 2020.

7. Martens D., De Backer M., Haesen R., Vanthienen J., Snoeck M., Baesens B.: Classification with ant colony optimization. // IEEE Trans. Evol. Comput. 11(5), pp 651–665, 2007

8. Socha K., Dorigo M. Ant colony optimization for continuous domains // European Journal of Operational Research, Volume 185, Issue 3, 2008, pp. 1155-1173. DOI: 10.1016/j.ejor.2006.06.046

9. Mohamad M., Tokhi M., Omar O. M. Continuous Ant Colony Optimization for Active Vibration Control of Flexible Beam Structures // IEEE International Conference on Mechatronics (ICM). Apr., 2011, — P. 803-808.

10. Карпенко А.П. Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой. // М.: изд-во МГТУ им. Баумана. 2-е изд., с. 446, 2017

11. Карпенко А.П., Синицын И.Н. Бионика и системы высокой доступности // Системы высокой доступности. 2022. Т. 18. № 2. с. 25−41. DOI: DOI: 10.18127/j20729472-202202-02

12. Саймон Д. Алгоритмы эволюционной оптимизации: практическое руководство // М.: ДМК Пресс, с. 1002, 2020.

13. Bergstra James S., Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. // In Advances in neural information processing systems, pp. 2546-2554. 2011

14. Akiba Takuya, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. // In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2623-2631. 2019. DOI: 10.48550/arXiv.1907.10902

15. Титов Ю.П. Опыт моделирования планирования поставок с применением модификаций метода муравьиных колоний в системах высокой доступности // Системы высокой доступности. 2018. Т. 14. № 1. С. 27-42.

16. Титов Ю.П. Модификации метода муравьиных колоний для разработки программного обеспечения решения задач многокритериального управления поставками // Современные информационные технологии и ИТ-образование. 2017. Т. 13. № 2, с. 64-74. DOI 10.25559/SITITO.2017.2.222.

17. Sudakov V. A., Titov Y. P. Modified Method of Ant Colonies Application in Search for Rational Assignment of Employees to Tasks // Proceedings of 4th Computational Methods in Systems and Software 2020. Vol.2, Vsetin: Springer Nature, 2020. P. 342-348. DOI 10.1007/978-3-030-63319-6_30.

18. Синицын И.Н., Титов Ю.П. Оптимизация порядка следования гиперпараметров вычислительного кластера методом муравьиных колоний // Системы высокой доступности. 2022. Т. 18. № 3. С. 23-37. DOI: 10.18127/j20729472-202203-02.

19. Синицын И.Н., Титов Ю.П. Исследование алгоритмов циклического поиска дополнительных решений при оптимизации порядка следования гиперпараметров методом муравьиных колоний // Системы высокой доступности. 2023. Т. 19. № 1. С. 59-73. DOI: 10.18127/j20729472-202301-05

20. Layeb Abdesslem New hard benchmark functions for global optimization. 2022 DOI: 10.48550/arXiv.2202.04606

Оглавление