Особенности применения алгоритма полного перебора для решения задачи квадратичного назначения

Ай Мин Тайк, С.А. Лупин, С.А. Балабаев

***Аннотация*— Алгоритм полного перебора вариантов – это универсальный метод решения задач дискретной оптимизации, обеспечивающий нахождение всех возможных комбинаций параметров, соответствующих экстремуму критериальной функции. Основным недостатком алгоритма является высокая вычислительная сложность, препятствующая его использованию для решения практических задач большой размерности. В статье рассматриваются некоторые методы ее снижения при решении задачи квадратичного назначения, не приводящие к потере точности. Кроме традиционного подхода, опирающегося на распараллеливание ресурсоемких вычислений, исследованы способы ускорения процесса генерации вариантов и ограничения их числа. Результаты вычислительных экспериментов подтверждают эффективность предложенного подхода и возможность его использования в последовательных и параллельных реализациях алгоритма перебора. Для последовательного приложения получено более чем 2000-кратное ускорение вычислений, а многопоточное приложение дает рост производительности еще в 2,3 раза при запуске на четырех ядерном процессоре.**

***Ключевые слова*— алгоритм полного перебора вариантов, параллельная реализация алгоритмов оптимизации, задача о квадратичном назначении, OpenMP.**

I. Введение

Алгоритмы дискретной оптимизации — это множество математических методов, используемых для решения проблем, заключающихся в нахождении параметров, определяющих экстремум некоторой критериальной функции. Эти алгоритмы используются для решения прикладных инженерных задач, в экономике, финансовом анализе, исследованиях операций, в системах поддержки принятия решений.

Алгоритмы оптимизации разделяют на два класса: детерминированные и стохастические [1]. Первые для поиска оптимального решения используют системный, а вторые - случайный или вероятностный подход. Выбор алгоритма зависит от характера и размерности задачи, характеристик пространства параметров решений.

Благодаря развитию вычислительной техники сегодня даже ресурсоемкие оптимизационные алгоритмы находят применение при решении практических задач [2].

Методы оптимизации часто используют для решения плохо формализуемых задач управления с большим количеством переменных и ограничений. В качестве критерия оптимальности может выступать время решения задачи или выполнения работы, затраченная энергия [3], материальные или финансовые ресурсы.

В проводимых исследованиях в качестве объекта выбрана задача квадратичного назначения (Quadratic Assignment Problem, QAP), а решателем является алгоритм полного перебора вариантов (Brute Force Algorithm, BFA). Основная цель заключается в поиске методов повышения эффективности BFA алгоритма при решении задач, представляемых в форме QAP. В качестве критерия эффективности метода будем использовать его влияние на повышение производительности алгоритма, т.е. на снижение времени решения задачи.

Алгоритм перебора — это универсальный и простой метод решения широкого спектра задач [4], но он обладает высокой вычислительной сложностью. Для QAP это $O(N!)$. Существуют различные методы её снижения, направленные на повышение эффективность алгоритма перебора.

Рассмотрим некоторые из них.

Обрезка — это процесс ограничения неперспективных направлений поиска в методе ветвей и границ (Branch and Bound, BB). Если оценка решений, которые могут быть получены при движении по конкретной ветви, выше, чем известное нам решение, ветвь можно считать неперспективной [5].

Запоминание — это метод, позволяющий избежать избыточных вычислений. Он основан на сохранении результатов предыдущих вычислений в таблице, и их последующего использования вместо пересчета. Это сокращает время работы алгоритма перебора [6].

Эвристики — это методы, ориентированные на конкретную проблему, позволяющие управлять процессом поиска. Например, для задачи нахождения кратчайшего пути в графе, можно использовать эвристику, которая отдает предпочтение узлам, расположенным ближе к месту назначения [7].

Распараллеливание — это традиционный метод повышения производительности вычислений, позволяющий повышать скорость рассмотрения

вариантов в алгоритме перебора [8].

Учёт симметрии — это метод сокращения размера пространства поиска. Он заключается в выявлении и устранении эквивалентных симметричных решений, что уменьшает пространство поиска и сложность решаемой задачи без потери точности [9].

Для повышения эффективности алгоритма перебора такие методы могут использоваться как по отдельности, так и в комбинации. Выбор зависит от решаемой задачи.

В этой статье мы оценим возможность использования методов повышения эффективности BFA, ранее опробованных нами для задачи о неограниченном ранце [10], для решения задачи о квадратичном назначении.

II. Квадратичная задача о назначении

Квадратичная задача о назначении (Quadratic Assignment Problem, QAP) была сформулирована в 1957 году Купмансом и Бекманном. Они решали задачу распределения набора из N объектов по набору из N местоположений [11]. Задача QAP является одной из интересных и наиболее сложных задач комбинаторной оптимизации. Задача коммивояжера, нахождения максимальной клики и разбиение графа, могут быть сформулированы как QAP, и это относит их к классу NP-полных задач. Для QAP не существует точного алгоритма, который мог бы найти решение этой задачи за практически приемлемое время для размерности $N > 35$ [12]. Решение этих проблем требует как совершенствования алгоритмов математического программирования, так и использования мощных вычислительных платформ.

Многие ученые, включая математиков, специалистов по информатике, аналитиков по исследованию операций и экономистов, использовали QAP для формализации различных задач оптимизации. Вот некоторые практические QAP-приложения: проектирование планировки помещений, размещение электронных компонентов на плате, минимизация среднего времени завершения работы, ранжирование археологических данных, экономическое планирование, анализ химических реакций, численный анализ, транспортные системы [13].

Рассмотрим их подробнее.

Транспортные системы. QAP может быть применена к транспортным системам для оптимизации маршрутизации товаров и минимизации транспортных расходов. В частности, для определения оптимального распределения товаров по маршрутам доставки с учетом таких факторов, как транспортные расходы, транзитное время и потребительский спрос. Кроме того, QAP может быть использована для решения, таких практических задач как: маршрутизация товаров от поставщиков до заводов-изготовителей, распределение запасов по складам и распределительным центрам, назначение маршрутов доставки отдельным грузовикам или транспортным средствам доставки. Чтобы применить QAP к транспортным системам, можно использовать критериальную функцию, которая минимизирует общие транспортные затраты на доставку товаров [14]. Критерий может быть усложнен с учетом дополнительных факторов, таких как затраты на хранение запасов и требования к уровню обслуживания.

Планировка производственных помещений и больниц. В медицине, QAP позволяет определить оптимальное распределение оборудования по кабинетам внутри больницы или поликлиники с учетом затрат на перемещение медицинских работников и пациентов [15], сократить время, необходимое для прохождения медицинских процедур. Для производственных объектов, QAP используют для решения таких практических задач как: минимизация расстояния, пройденного рабочими и материалами, сокращение суммарного времени производства и повышение его эффективности, оптимизация размещения оборудования для снижения риска несчастных случаев и повышения безопасности, сокращение пространства, необходимого для оборудования и материалов. В простейших случаях критериальная функция минимизирует общее расстояние, пройденное рабочими или материалами [16].

Проектирование печатных плат. В электронике QAP применяется в качестве математической модели при размещении компонентов на печатной плате или в микросхеме. Спектр используемых при этом критериальных функций достаточно широк. Они позволяют: минимизировать суммарную длину проводников, уменьшать искажения сигналов, снижать размера и вес коммутационной платы, повышать эффективность системы охлаждения электронных компонентов [17].

III. Методы решения QAP

Метод ветвей и границ (Branch and Bound Algorithm, BBA) [18] является хорошо известным подходом для решения задач комбинаторной оптимизации, который сокращает пространство поиска решений за счёт обрезки неперспективных ветвей дерева поиска, используя верхнюю и нижнюю границы целевой функции. BBA разбивает задачу, подлежащую решению, на ряд подзадач, каждая из которых может иметь несколько возможных решений, влияющих на ход дальнейших вычислений. При решении QAP с использованием алгоритма ветвей и границ, необходимо выбрать начальный узел и вычислить верхнюю оценку целевой функции. В процессе поиска решения исследуются ветви, имеющие наилучшую оценку из еще не рассмотренных. В качестве оценки в QAP может использоваться, например, скалярное произведение векторов или решение задачи линейного назначения.

Алгоритм ветвей и границ гарантированно получает точное решение для задач малой и средней размерности ($N < 20$) [19]. При большей размерности BBA может стать непрактичным, как из-за большого пространства поиска, так и из-за вычислительной сложности решения задачи линейного назначения.

Генетические алгоритмы — это методы популяционной оптимизации, имитирующие процесс естественного отбора. Существуют эвристические генетические алгоритмы (Genetic Algorithm, GA) и для нахождения решения QAP. Чтобы применить GA к любой проблеме, пространство решений должно быть представлено в виде хромосомы и должна быть определена целевая функция, которая оценивает качество решения. В QAP, генетический алгоритм работает путем представления возможных решений в виде строк двоичных значений, которые затем подвергаются процессу отбора, скрещивания и мутации. Было показано, что GA эффективен для решения QAP, особенно для задач больших размеров [20], однако он не гарантирует нахождения точного решения.

Алгоритмы имитации отжига — это группа эвристик, которые могут быть использованы для решения задач оптимизации путем моделирования физического процесса отжига в металлах, упорядочивающего их структуру. В QAP алгоритм имитации отжига итерационно изменяет текущее решение с определенной вероятностью, которая уменьшается со временем. Было показано, что имитация отжига эффективна для решения QAP, особенно для задач с высокой степенью симметрии [21].

Поиск с запретами — это алгоритм метаэвристической оптимизации, который может быть также использован для решения задач комбинаторной оптимизации, таких как квадратичная задача о назначении [22].

Поиск с запретами начинается с первоначального решения и итерационно улучшает его, исследуя пространство решений. На каждой итерации генерируется решение-кандидат путем небольшого изменения текущего решения. Кандидат оценивается по значению его критериальной функции, и переход к нему осуществляется, если значение критерия улучшается. В отличие от других итерационных алгоритмов поиск с запретами поддерживает запретный список ранее изученных решений, к которым нельзя возвращаться в течение определенного количества итераций. Это стимулирует исследование новых областей пространства решений, что может помочь избежать застревания в локальных оптимумах.

Эффективность поиска с запретами зависит от выбора его параметров, таких как длина списка табу, размер структуры окрестностей и критерий остановки. Настройка этих параметров сложно формализуется, но на практике было показано, что это мощный алгоритм оптимизации, эффективный для решения QAP и других задач комбинаторной оптимизации [23], особенно в случае большого пространством решений.

IV. BFA И ЗАДАЧА О КВАДРАТИЧНОМ НАЗНАЧЕНИИ

Задача размещения элементов является одним из этапов при проектировании коммутационных плат. Для нахождения оптимальной топологии необходимо решить связанные с этим задачи нелинейного программирования [24]. Системы автоматизированного проектирования для решения QAP используют разные по сложности и точности получаемых решений алгоритмы, однако в случае большого числа элементов мы вынуждены ограничивать их спектр из-за недопустимо высоких временных затрат.

В этой статье исследуются методы повышения производительности алгоритма полного перебора вариантов при решении QAP.

Алгоритм полного перебора вариантов – это универсальный и самый мощный метод решения задач дискретной оптимизации. Более того, BFA – это единственный алгоритм, гарантирующий получение точного решения. Это важно в тех ситуациях, когда не время решения задачи, а необходимость нахождения наилучшего возможного решения имеет решающее значение.

BBA является модификацией BFA, направленной на уменьшение пространства поиска решений. Отметим, что в живой природе BFA реализуется как метод проб и ошибок, определяя способность всех существ приспосабливаться к изменениям окружающей среды. При этом неперспективные варианты развития отсекаются за счёт механизма естественно отбора.

Исследование методов повышения эффективности алгоритма перебора мы проводили на примере решения задачи размещения электронных компонентов на печатной плате (Placement Problem, PP), которая относится к QAP.

Математически задача размещения в виде QAP формулируется следующим образом.

1. Пусть заданы симметричные относительно главной диагонали матрицы $R = \left|\left|r_{ij}\right|\right|$ и $D = \left|\left|d_{ij}\right|\right|$, где элементы r_{ij} определяют количество связей между соответствующими компонентами, а элементы d_{ij} определяют расстояние между посадочными местами на коммутационной плате.

2. Решение задачи заключается в нахождении вектора $P = \left|\left|p_k\right|\right|$, элементы которого определяют размещение компонентов, удовлетворяющего условию:

$$F = min \sum_{i=1}^{N} \sum_{j=i+1}^{N} R_{ij} D_{p(i)p(j)}$$

Пространство поиска решения включает $N!$ вариантов, где N число посадочных мест на коммутационной плате.

При решении задачи размещения элементов BFA реализует следующие процедуры:

1. *Генерация* всех возможных вариантов размещения элементов на печатной плате. Это можно сделать с помощью рекурсивного алгоритма, который генерирует все перестановки в заданном наборе.

2. Вычисление критериальной функции F для варианта размещения (или *оценка* его оптимальности). Для РР оценка определяет суммарную длину соединений всех элементов в Манхэттенской или Евклидовой метрике.

3. *Выбор* в качестве решения задачи варианта размещения с минимальным значением критериальной функции F.

В качестве тестовой задачи в работе используется пример с $N = 14$. Соответствующие матрицы R и D представлены ниже.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	∑
1	0	1	0	1	0	0	0	0	0	0	0	0	4	4	10
2	1	0	3	0	4	0	0	0	0	0	0	0	0	0	8
3	0	3	0	0	0	2	0	0	0	0	0	0	0	0	5
4	1	0	0	0	4	0	3	0	0	0	0	0	0	0	8
5	0	4	0	4	0	4	0	4	0	0	0	0	0	0	16
6	0	0	2	0	4	0	0	0	1	0	0	0	0	0	7
7	0	0	0	3	0	0	0	2	0	0	0	0	0	0	5
8	0	0	0	0	4	0	2	0	1	0	0	0	0	0	7
9	0	0	0	0	0	1	0	1	0	2	0	2	0	0	6
10	0	0	0	0	0	0	0	0	2	0	2	0	0	0	4
11	0	0	0	0	0	0	0	0	0	2	0	2	0	0	4
12	0	0	0	0	0	0	0	0	2	0	2	0	0	0	4
13	4	0	0	0	0	0	0	0	0	0	0	0	0	0	4
14	4	0	0	0	0	0	0	0	0	0	0	0	0	0	4

Рис. 1 Матрица связей элементов R

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	∑
1	0	1	2	1	2	3	2	3	4	5	6	5	1	1	36
2	1	0	1	2	1	2	3	2	3	4	5	4	2	2	32
3	2	1	0	3	2	1	4	3	2	3	4	3	3	3	34
4	1	2	3	0	1	2	1	2	3	4	5	4	2	2	32
5	2	1	2	1	0	1	2	1	2	3	4	3	3	3	28
6	3	2	1	2	1	0	3	2	1	2	3	2	4	4	30
7	2	3	4	1	2	3	0	1	2	3	4	3	3	3	34
8	3	2	3	2	1	2	1	0	1	2	3	2	4	4	30
9	4	3	2	3	2	1	2	1	0	1	2	1	5	5	32
10	5	4	3	4	3	2	3	2	1	0	1	2	6	6	42
11	6	5	4	5	4	3	4	3	2	1	0	1	7	7	52
12	5	4	3	4	3	2	3	2	1	2	1	0	6	6	42
13	1	2	3	2	3	4	3	4	5	6	7	6	0	2	48
14	1	2	3	2	3	4	3	4	5	6	7	6	2	0	48

Рис. 2 Матрица расстояний посадочных мест D

Для этого примера число вариантов, которые необходимо рассмотреть в алгоритме размещения для QAP, составит $N! = 14! = 87\,178\,291\,200 \approx 8{,}7 \cdot 10^{10}$. Вариант счетчика без ограничений реализует программа *Var-1*.

V. Особенности программной реализации BFA

Рассмотрим возможность *ограничения неперспективных направлений поиска* и *распараллеливания* при решении задачи размещения элементов с помощью BFA.

Исключение неперспективных направлений поиска в процессе генерации вариантов без расчета критериальной функции можно реализовать с помощью механизма поразрядного сброса счетчика при достижении в разряде максимального значения.

Для определения неперспективных направлений поиска воспользуемся принципом формирования нижней оценки в методе ветвей и границ. Он базируется на том факте, что минимум скалярного произведения двух векторов обеспечивается в том случае, если их компоненты отсортированы в противоположном порядке. Для задачи размещения это означает, что лучший вариант мы получим, если элементы с максимальным числом связей будут располагаться на центральных местах. Таким образом, сократить пространство поиска можно разделив элементы и посадочные места на несколько групп. От корректности проведения этой процедуры будет зависеть точность получаемых решений.

В рассматриваемом примере мы разделили элементы и посадочные места на две группы: $N_1 = 9$ и $N_2 = 5$, $N_1 + N_2 = N = 14$. Разделение зависит от суммарного числа соединений элементов в матрице R и суммы расстояний между посадочными местами в матрице D. Результат деления на группы отмечен в матрицах серым цветом. В результате такого разделения на группы число вариантов рассматриваемых сократится и будет равно: $N_1! \cdot N_2! = 9! \cdot 5! = 43\,545\,600 \approx 4{,}4 \cdot 10^7$.

Такой вариант счетчика реализует программа *Var-2*.

Достоинством алгоритма перебора является простота его реализации и в последовательном, и в параллельном вариантах, но это справедливо, если мы реализуем простые генераторы вариантов. Использование генераторов с разными значениями разрядов приводит к очевидному усложнению кода. Рассмотрение всей программы в рамках статьи не представляется интересным, ограничимся только фрагментом, реализующим генератор вариантов. Параллельные варианты программ реализованы как многопоточные приложения с использованием библиотеки OpenMP.

Структура программы:

1: Define global variables: min = 5000, resultmatrix[50][14], N;
Main()
2: Initialization (matrix R, D, vector P)
3: *Call permutationfun*
4: *Output optimal results and computing time*

Генерацию вариантов выполняет рекурсивная функция перестановки элементов, в которую передаются параметры из основной функции.

Реализация генерации вариантов в *Var-1*.

```
1: void permutationfun(parameters){
2: computevalue = 0;
3: for k = 0 to size do
4:    call permutationfun(size − 1, parameters)
5:    if (size%2 == 1)
6:       swap(p[0], p[size − 1];
7:    else
8:       swap(p[k], p[size − 1];
9: end
10: if (size == 1)
11:    for i = 0 to n do
12:       for j = i + 1 to size do
13:          computevalue+= R[i][j] ∗ D[p[i]][p[j]];
14:       end
15:    end
16:    if (min ≥ computevalue){
17:       min = computevalue
18: save current permutation variant into the savematrix}
```

Для варианта *Var-2* реализации перестановок отличается от *Var-1*. В этом варианте BFA используются две перестановки: одна для N_1 и вторая для N_2. Функция *permutationfun* реализует перестановки для N_1, а *secondpermutation* для N_2. Эта функция может выполнять перестановки с указанной начальной позиции индекса с помощью итератора объекта.

Для параллельных приложений необходимо обеспечить одновременную перестановку элементов. Следовательно, нужно указать начальную позицию индекса с помощью итератора объекта функции перестановки для каждого вычислительного ядра. Реализация перестановок в *Var-2*.

```
1: Define global variables: min = 5000, resultmatrix[50][14], N;
Main() {
2:    Initialization(matric R, D, vector P, iterator PP, N1, N2)
3: do { for j = 0 to N2 do
4:          P[N1 + j] = pp[j];
5:    Call permutationfun(parameters);
6: }while (secondpermutation(begin(pp) + 0, end(pp)));
7:    Output optimal results and computing time
```

Функция перестановки с указанной начальной позиции индекса с помощью итератора объекта.

```
1: bool secondpermutation(Iterator a, Iterator b){
2: if (a == b)return false;
3: Iterator x = b;
4: if (a == −−x)return false;
5: while(1){ Iterator m = x, mm;
6:    if (∗−−x <∗ m) { mm = b;
7:       while(1)(! (∗−−x < ∗−− mm));
8:          iter_swap(x, mm);
9:          reverse(m, b); return true; }
10:   if (x == a){ reverse(a, b); return false; } } }
```

Экспериментальная программа написана на языке программирования C++.

Фрагмент кода для параллельной реализации четырёхпоточного варианта счётчика *Var-2* представлен ниже.

```
1: Define global variables: min = 5000, resultmatrix[50][14], N;
Main() {
2:    Initialization(matric R, D, vector P, iterator PP, N1, N2)
3: #pragma omp parallel for num_threads(4)
4: for i = 0 to N2 do {
5:    int pp[  ] = {10,11,12,13,14};
6:    int p[  ] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14};
7:    int temp; temp = pp[0]; pp[0] = pp[i]; pp[i] = temp;
8:    sort(begin(pp) + 1, end(pp));
9:    do {
10:      for j = 0 to N2 do {
11:         P[N1 + j] = pp[j]; }
12:   Call permutationfun(parameters);
13:}while (secondpermutation(begin(pp) + 1, end(pp))); }
14:   Output optimal results and computing time
```

В генераторе вариантов для формирования вектора P мы не использовали векторный тип данных и функцию *next_permutation*. Вместо этого для генерации перестановок использован метод *Heap* и данные типа итератор, потому что скорость доступа к итератору выше, чем к другому типу данных.

Простейшим способом распараллеливания вычислений в многопоточных приложениях является использование функции *omp parallel for* из библиотеки OpenMP [25]. В представленных выше фрагментах кодов двух перестановок формирование выходного вектора происходит от младшего разряда к старшему, что требует незначительной трансформации алгоритма при его распараллеливании. Воспользуемся тем, что каждое изменение индекса в цикле *omp parallel for* обеспечивает запуск нового потока, и организуем разделение вычислений в потоках с помощью значения старшего разряда вектора P.

Необходимо отметить, что такое распределение нагрузки между потоками в общем случае не является сбалансированным, поскольку число вариантов вектора для разных значений старшего разряда отличается.

VI. Результаты экспериментов

Для проведения вычислительных экспериментов использовался персональный компьютер с процессором Intel® Core™ i3 9-го поколения, количество физических ядер – 4, тактовая частота – 3,1 ГГц. Последовательные приложения запускались на одном ядре. Параллельные приложения запускались на 4-х физических ядрах, режим гипертрединга не использовался. Программирование проводилось в среде Visual Studio 2019.

Результаты работы приложений в последовательном варианте представлены в таблице 1. Параметр N_{var} это число сгенерированных вариантов решения, а t_{calc} – время вычислений в секундах.

Таблица 1. Последовательное исполнение программ

Код	N_{var}	t_{calc}
Var-1	87 178 291 200	34063,6
Var-2	43 545 600	16,9

Полученные данные подтверждают, что количество сгенерированных вариантов в обоих случаях совпадает с теоретическими оценками. Исключение неперспективных направлений поиска в *Var-2* позволило сократить число рассматриваемых вариантов и время поиска решения более чем в 2000 раз,

Поскольку приложение *Var-1* гарантирует нахождение точного решения, корректность разбиения пространства поиска на части в *Var-2* подтверждается совпадением полученных ответов. В таблице 2 показаны 8 точных решений, полученных программами.

Таблица 2. Полученные решения

1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	4	7	2	5	8	3	6	9	10	11	12	13	14
1	2	3	4	5	6	7	8	9	10	11	12	14	13
1	4	7	2	5	8	3	6	9	10	11	12	14	13
1	2	3	4	5	6	7	8	9	12	11	10	13	14
1	4	7	2	5	8	3	6	9	12	11	10	13	14
1	2	3	4	5	6	7	8	9	12	11	10	14	13
1	4	7	2	5	8	3	6	9	12	11	10	14	13

Оценим теперь, как будут работать счётчики в параллельных приложениях. Критерием оценки в данном случае будет выступать полученное по отношению к последовательной версии ускорение. В таблице 3 это параметр Accel.

Таблица 3. Параллельное исполнение программ

	Var-1		*Var-2*	
Поток	t_{calc}	Accel	t_{calc}	Accel
1	34068,6	-	16,9	-
2	17682,7	1,93	10,46	1,62
3	13171,2	2,59	7,51	2,25
4	11165,1	3,05	7,42	2,27

Запуск приложений в параллельном режиме позволяет ускорить вычисления, но не обеспечивает достижения возможного максимума (для используемой платформы это 4). Причин того, две. Первая связана с методом распределения нагрузки – использование значения старшего разряда счётчика, а вторая характерна для конструкции *omp parallel for*. Максимальное ускорение она позволяет получить, только если число циклов *for* делится на число параллельных потоков (в нашем случае ядер) нацело, иначе на последней стадии будут работать не все ядра. С ростом размерности решаемых задач влияние обеих причин на эффективность вычислений будет снижаться.

VII. Заключение

Проведенные вычислительные эксперименты подтвердили, что эффективность универсального метода решения задач комбинаторной оптимизации, инвариантного к условиям поиска, алгоритма полного перебора вариантов, может быть значительно повышена за счет оптимизации генератора вариантов. Отсечение неперспективных вариантов без вычисления критериальной функции позволяет повысить производительность BFA и увеличить размерность решаемых задач.

Дальнейшие наши исследования в этой области будут направлены на оценку эффективности предлагаемого подхода для задач размещения элементов большой размерности, в частности, для решения тест-задачи Штейнберга [26].

Библиография

[1] Mario Francisco, Silvana Revollar, Pastora Vega, Rosalba Lamanna,“A Comparative study of Deterministic and Stochastic Optimization methods for Integrated Design of Processes”, IFAC Proceedings Volumes, vol.38, Iss.1, pp. 335-340, 2005.

[2] Wendly Saintil, “The Role of Optimization Algorithms in Improving Efficiency and Decision-Making”, [Online]. Available: https://medium.com/codex/the-role-of-optimization-algorithms-in-improving-efficiency-and-decision-making-d61df38ba6bc.

[3] Thomas Weise, “Metaheuristic Optimization Algorithms”, An Introduction to Optimzation Algorithms, pp. 49-150, 26 December 2020.

[4] Aye Min Thike, Sergey Lupin,Yuriy Vagapov, “Implementation of Brute Force Algorithm for Topology Optimisation of Wireless Networks”, 2016 International Conference for Students on Applied Engineering (ICSAE), 20 - 21 October 2016, Newcastle upon Tyne, UK. C. 264-268.

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, “Introduction of Algorithms”, ISBN 978-0-262-03384-8, 31 July 2009.

[6] Madeline Corman, “Memoization”, [Online]. Available: https://medium.com/@madelinecorman/memoization-96ab60464bd2.

[7] M. Fatih Tasgetiren, Quan-Ke Pan, P. N. Suganthan, Ikbal Ece Dizbay, “Metaheuristic algorithms for the quadratic assignment problem”, 2013 IEEE Symposium on Computational Intelligence in Production and Logistics Systems (CIPLS), pp. 131-137, 16-19 April 2013, Singapore.

[8] Milan Vaško, Marián Handrik, Alžbeta Sapietová, Jana Handriková, “Parallelization of computational algorithms for optimization problems with high time consumption”, MATEC Web of Conferences 157, pp. 1-10, 2018.

[9] Elena Castellani. “On the meaning of symmetry breaking”, Symmetries in Physics: Philosophical Reflections, Cambridge University Press, 2003.

[10] Ай Мин Тайк, С.А. Лупин, Мин Тху Кхаинг, “Методы повышения эффективности алгоритма полного перебора на примере решения задачи о неограниченном ранце”, International Journal of Open Information Technologies, Vol. 11, No 5 (2023), pp.41-46.

[11] Santosh Kumar Sahu, Manish Pandey, “Hybrid Ant System Algorithm for Solving Quadratic Assignment Problmes”, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol.5, pp. 5950-5956, 2014.

[12] Tansel Dokeroglu, Ahmet Cosar, Ender Sevinc, “Artificial bee colony optimization for the quadratic assignment problem”, Applied Soft Computing (JoCAI), March 2019.

[13] Eliane Maria Loiola, Nair Maria Maia de Abreu, Paulo Oswaldo Boaventura-Netto, Peter Hahn, Tania Querido, “A survey for the quadratic assignment problem”, European Journal of Operational Research 176 (2007),pp. 657-690, 27 December 2005.

[14] Xin (Bruce) Wu, Jiawei Lu, Shengnan Wu, Xuesong (Simon) Zhou, “Synchronizing time-dependent transportation services: Reformulation and solution algorithm using

quadratic assignment problem", Transportation Research Part B: Methodological, Vol – 152, pp 140-179, October 2021.

[15] Clayton W. Commander, "A Survey of the Quadratic Assignment Problem, with Applications", Morehead Electronic Journal of Applicable Mathematics, 2005.

[16] Cemre Cubukcuoglu, Pirouz Nourian, M. Fatih Tasgetiren, I. Sevil Sariyildiz, Shervin Azadi, "Hospital layout design renovation as a Quadratic Assignment Problem with geodesic distances", Journal of Building Engineering 44(2021).

[17] Ekrem Duman, Ilhan Or, "The quadratic assignment problem in the context of the printed circuit board assembly process", Computers & Operations Research 34 (2007), pp-163-179.

[18] Sharifah Shuthairah Syed-Abdullah, Syariza Abdul-Rahman, Aida Mauziah Benjamin, Antoni Wibowo, Ku-Ruhana Ku-Mahamud, "Solving Quadratic Assignment Problem with Fixed Assignment (QAPFA) using Branch and Bound Approach", 4th International Conference on Operational Research (InteriOR), 2018.

[19] Danny Munera, Daniel Diaz, Salvador Abreu, "Solving the Quadratic Assignment Problem with Cooperative Parallel Extremal Optimization",16th European Conference on Evolutionary Computation in Combinatorial Optimization, EvoCOP 2016, March 2016, Porto, Portugal.

[20] H. Azarbonyad, R. Babazadeh, "A Genetic Algorithm for solving Quadratic Assignment Problem (QAP)", 5th International Conference of Iranian Operations Research Society (ICIORS), Tabriz, Iran, 2012.

[21] Kambiz Shojaee Ghandeshtani, Nima Mollai, Seyed Mohammad Hosein Seyedkashi, Mohammad Mohsen Neshati, "New Simulated Annealing Algorithm for Quadratic Assignment Problem", 4th International Conference on Advanced Engineering Computing and Applications in Sciences, 2010.

[22] Alfonsas Misevicius, "A Tabu Search Algorithm for the Quadratic Assignment Problem", Computational Optimization and Applications, January 2005.

[23] Omar Abdelkafi, Bilel Derbel, Arnaud Liefooghe, "A Parallel Tabu Search for the Large-scale Quadratic Assignment Problem", IEEE Congress on Evolutionary Computation, IEEE CEC 2019, Jun 2019, Wellington, New Zealand.

[24] Hossein Jafari, Abbas Sheykhan, "Using a New Algorithm to Improve the Search Answer in Quadratic Assignment Problem (QAP)", International Journal of Research in Industrial Engineering, Vol. 10, No. 2 (2021), 28 May 2021.

[25] OpenMP. [Online]. Available: https://www.openmp.org/.

[26] G. W. Graves and A. B. Whinston, "An Algorithm for the Quadratic Assignment Problem", Management Science, Vol. 16, No. 7, Theory Series (Mar., 1970), pp. 453-471.

Статья получена 05 мая 2023.

С.А. Лупин, профессор, Национальный исследовательский университет «МИЭТ», (e-mail: lupin@miee.ru);

Ай Мин Тайк, к.т.н., докторант Национального исследовательского университета «МИЭТ» (e-mail: ayeminthike52@gmail.com);

С.А. Балабаев, аспирант Национального исследовательского университета «МИЭТ»; (email: sergei.balabaev@mail.ru).

The features of using a brute force algorithm for solving a quadratic assignment problem

Aye Min Thike, S.A. Lupin, S.A. Balabaev

***Abstract* — The brute force algorithm of the variants is a universal method for solving discrete optimization problems, which provides finding all possible combinations of parameters corresponding to the extremum of the criterion function. The main disadvantage of the algorithm is its high computational complexity, which prevents its use for solving practical problems of large size. The article discusses some methods for reducing its computational complexity when solving the quadratic assignment problem, which do not lead to loss of accuracy. In addition to the traditional approach based on parallelization of resource-intensive computations, methods to accelerate the process of generating variants and limiting their number have been investigated. The results of computational experiments confirm the effectiveness of the proposed approach and its applicability in sequential and parallel implementations of the algorithm. The sequential application was obtained more than 2000x acceleration of calculations, and a multithreaded application provides an additional 2.3x performance boost when running on a quad-core processor.**

***Keywords*— The brute force algorithm, parallel implementation of optimization algorithms, quadratic assignment problem, OpenMP.**

REFERENCES

[1] Mario Francisco, Silvana Revollar, Pastora Vega, Rosalba Lamanna,"A Comparative study of Deterministic and Stochastic Optimization methods for Integrated Design of Processes", IFAC Proceedings Volumes, vol.38, Iss.1, pp. 335-340, 2005.

[2] Wendly Saintil, "The Role of Optimization Algorithms in Improving Efficiency and Decision-Making", [Online]. Available: https://medium.com/codex/the-role-of-optimization-algorithms-in-improving-efficiency-and-decision-making-d61df38ba6bc.

[3] Thomas Weise, "Metaheuristic Optimization Algorithms", An Introduction to Optimzation Algorithms, pp. 49-150, 26 December 2020.

[4] Aye Min Thike, Sergey Lupin,Yuriy Vagapov, "Implementation of Brute Force Algorithm for Topology Optimisation of Wireless Networks", 2016 International Conference for Students on Applied Engineering (ICSAE), 20 - 21 October 2016, Newcastle upon Tyne, UK. C. 264-268.

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, "Introduction of Algorithms", ISBN 978-0-262-03384-8, 31 July 2009.

[6] Madeline Corman, "Memoization", [Online]. Available: https://medium.com/@madelinecorman/memoization-96ab60464bd2.

[7] M. Fatih Tasgetiren, Quan-Ke Pan, P. N. Suganthan, Ikbal Ece Dizbay, "Metaheuristic algorithms for the quadratic assignment problem", 2013 IEEE Symposium on Computational Intelligence in Production and Logistics Systems (CIPLS), pp. 131-137, 16-19 April 2013, Singapore.

[8] Milan Vaško, Marián Handrik, Alžbeta Sapietová, Jana Handriková, "Parallelization of computational algorithms for optimization problems with high time consumption", MATEC Web of Conferences 157, pp. 1-10, 2018.

[9] Elena Castellani. "On the meaning of symmetry breaking", Symmetries in Physics: Philosophical Reflections, Cambridge University Press, 2003.

[10] Ай Мин Тайк, С.А. Лупин, Мин Тху Кхаинг, "Методы повышения эффективности алгоритма полного перебора на примере решения задачи о неограниченном ранце", International Journal of Open Information Technologies, Vol. 11, No 5 (2023), pp.41 -46.

[11] Santosh Kumar Sahu, Manish Pandey, "Hybrid Ant System Algorithm for Solving Quadratic Assignment Problmes", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol.5, pp. 5950-5956, 2014.

[12] Tansel Dokeroglu, Ahmet Cosar, Ender Sevinc, "Artificial bee colony optimization for the quadratic assignment problem", Applied Soft Computing (JoCAI), March 2019.

[13] Eliane Maria Loiola, Nair Maria Maia de Abreu, Paulo Oswaldo Boaventura-Netto, Peter Hahn, Tania Querido, "A survey for the quadratic assignment problem", European Journal of Operational Research 176 (2007),pp. 657-690, 27 December 2005.

[14] Xin (Bruce) Wu, Jiawei Lu, Shengnan Wu, Xuesong (Simon) Zhou, "Synchronizing time-dependent transportation services: Reformulation and solution algorithm using quadratic assignment problem", Transportation Research Part B: Methodological, Vol – 152, pp 140-179, October 2021.

[15] Clayton W. Commander, "A Survey of the Quadratic Assignment Problem, with Applications", Morehead Electronic Journal of Applicable Mathematics, 2005.

[16] Cemre Cubukcuoglu, Pirouz Nourian, M. Fatih Tasgetiren, I. Sevil Sariyildiz, Shervin Azadi, "Hospital layout design renovation as a Quadratic Assignment Problem with geodesic distances", Journal of Building Engineering 44(2021).

[17] Ekrem Duman, Ilhan Or, "The quadratic assignment problem in the context of the printed circuit board assembly process", Computers & Operations Research 34 (2007), pp-163-179.

[18] Sharifah Shuthairah Syed-Abdullah, Syariza Abdul-Rahman, Aida Mauziah Benjamin, Antoni Wibowo, Ku-Ruhana Ku-Mahamud, "Solving Quadratic Assignment Problem with Fixed Assignment (QAPFA) using Branch and Bound Approach", 4th International Conference on Operational Research (InteriOR), 2018.

[19] Danny Munera, Daniel Diaz, Salvador Abreu, "Solving the Quadratic Assignment Problem with Cooperative Parallel Extremal Optimization",16th European Conference on Evolutionary Computation in Combinatorial Optimization, EvoCOP 2016, March 2016, Porto, Portugal.

[20] H. Azarbonyad, R. Babazadeh, “A Genetic Algorithm for solving Quadratic Assignment Problem (QAP)”, 5th International Conference of Iranian Operations Research Society (ICIORS), Tabriz, Iran, 2012.

[21] Kambiz Shojaee Ghandeshtani, Nima Mollai, Seyed Mohammad Hosein Seyedkashi, Mohammad Mohsen Neshati, “New Simulated Annealing Algorithm for Quadratic Assignment Problem”, 4th International Conference on Advanced Engineering Computing and Applications in Sciences, 2010.

[22] Alfonsas Misevicius, “A Tabu Search Algorithm for the Quadratic Assignment Problem”, Computational Optimization and Applications, January 2005.

[23] Omar Abdelkafi, Bilel Derbel, Arnaud Liefooghe, “A Parallel Tabu Search for the Large-scale Quadratic Assignment Problem”, IEEE Congress on Evolutionary Computation, IEEE CEC 2019, Jun 2019, Wellington, New Zealand.

[24] Hossein Jafari, Abbas Sheykhan, “Using a New Algorithm to Improve the Search Answer in Quadratic Assignment Problem (QAP)”, International Journal of Research in Industrial Engineering, Vol. 10, No. 2 (2021), 28 May 2021.

[25] OpenMP. [Online]. Available: https://www.openmp.org/.

[26] G. W. Graves and A. B. Whinston, “An Algorithm for the Quadratic Assignment Problem”, Management Science, Vol. 16, No. 7, Theory Series (Mar., 1970), pp. 453-471.