УДК 004.021

К вопросу оценки точности алгоритмов дискретной оптимизации

С.А. Лупин, Тан Шейн, Тхан Зо У, Чжо Мью Хтун

Национальный исследовательский университет «МИЭТ», lupin@miee.ru

***Аннотация* — Рассматривается возможность использования параллельных реализаций случайных алгоритмов для оценки точности алгоритмов дискретной оптимизации.**

***Ключевые слова* — алгоритмы дискретной оптимизации; алгоритмы случайного поиска; параллельные алгоритмы.**

I. Введение

При решении задач цифровой обработки информации, проектировании электронных устройств, разработке систем управления вычислительными комплексами часто используются алгоритмы дискретной оптимизации. В общем случае они направлены на нахождение некоторых координат в *N*-мерном пространстве, при которых значение функционала, характеризующего состояние системы, достигает оптимума. Практических задач (например, проектирование цифровых фильтров с дискретными весовыми коэффициентами, разработка алгоритмов управления адаптивными антенными решетками с дискретными фазовращателями и т.п.), которые сводятся к дискретной оптимизации, как и методов их решения, существует довольно много, однако большинство из них позволяют получить не точное решение задачи, а лишь некоторое приближение к нему. Для каждого класса задач, точность такого приближения оценивается с помощью сравнения результатов работы алгоритмов с заранее найденным точным или известным нам лучшим решением. Во многих приложениях для его нахождения используется ресурсоемкий метод полного или усеченного (ветвей и границ) перебора. Например, для задачи о ранце, которая относится к *NP*-сложным и используется в системах криптозащиты, опыт использования метода ветвей и границ описан в [1]. Для *NP*-полных задач уже при размерности порядка $N = 20$ такой подход не может быть реализован даже на современных вычислительных системах, поскольку число вариантов решения при этом будет составлять $20! \approx 2.4*10^{18}$. С учетом числа операций, необходимых для вычисления оценки каждого варианта, число машинных команд составит $10^{20} - 10^{21}$. Это значит, что время выполнения программы даже на суперкомпьютерах с петафлопсной производительностью, займет более 1 года.

В качестве алгоритма, который позволяет найти координаты оптимального решения в *N*-мерном пространстве, авторами настоящей статьи предлагается использовать параллельные реализации алгоритмов случайного поиска. В качестве задач дискретной оптимизации рассматриваются задачи назначения на узкие места и квадратичного назначения.

II. Задачи дискретной оптимизации

A. Задача квадратичного назначения

К задаче квадратичного назначения сводится, в частности, задача размещения элементов на коммутационной плате, которая формулируется следующим образом [2]. Пусть задано множество элементов $e_1, e_2, \ldots, e_N$. Вся схема соединений задается квадратной матрицей соединений $\mathbf{R}$ с числом элементов $N \times N$, номера строк и столбцов которой соответствуют элементам схемы, а элементы матрицы r_{ij} определяют связи элементов друг с другом. Матрица $\mathbf{D}$ определяет расстояния d_{ij} между посадочными местами элементов на коммутационной плате. Алгоритмы размещения минимизируют квадратичный функционал, косвенно оценивающий суммарную длину печатных соединений:

$$F = \sum_{i=1}^{N} \sum_{j=1}^{N} r_{ij} d_{p(i)p(j)}, \quad (1)$$

где $p(i)$ задает номер позиции, присвоенной i -му элементу.

Для нахождения точного решения необходимо произвести перебор $N!$ различных вариантов размещения элементов. В алгоритмах случайного поиска решение получается путем выбора лучшего варианта из некоторого множества случайно сгенерированных размещений.

B. Задача назначения на узкие места

К задаче назначения на узкие места сводится задача диспетчеризации в распределенных системах обслуживания, которую можно представить в следующем виде [3]: пусть N – число заявок, поступивших от обслуживаемых объектов; M – число обслуживающих объектов; y_{ij} – целевая переменная, равная 1, если j-ый объект используется для обслуживания i-ого вызова и равная 0 в противном случае; r_{ij} – время следования j-ого объекта к источнику i-ой заявки.

Тогда целевая функция системы (2), минимизирующая время реакции, имеет вид:

$$T_{react} = \max(r_{ij} y_{ij}, \forall i = \overline{1,N}; \forall j = \overline{1,M}) \to \min . \quad (2)$$

При этом:

$$\sum_{i=1}^{N} y_{ij} \le 1,\ j = \overline{1,M} . \quad (3)$$

В такой постановке, число обработчиков заявок должно быть больше или равно числу самих заявок $M \ge N$. Исходной информацией для алгоритмов распределения заявок служит матрица $\mathbf{R}$, элементы которой r_{ij}, кроме времени движения обработчика к объекту, могут характеризовать и возможность выполнения объектом своей функции по отношению к источнику заявки. В этом случае минимизируя T_{react}, мы оптимизируем не только время движения, но и комплексный показатель эффективности системы обслуживания.

III. Программная реализация алгоритмов

На каждой итерации алгоритма случайного поиска осуществляется генерация случайного решения и находится значение целевой функции для него. Если это решение лучше известного нам, то оно запоминается и процесс продолжается заданное число итераций. Сложность алгоритма определяется числом итераций. Точность решения, получаемого случайными алгоритмами, в общем случае зависит от количества итераций.

При параллельной реализации алгоритма исходят из того, что его итерации независимы по данным [4]. Это позволяет распределить число итераций между отдельными процессами или потоками, реализующими алгоритм, которые не будут взаимодействовать между собой в процессе вычислений. Синхронизация необходима только после завершения каждым из них расчетов. В качестве общего решения берется лучший вариант, из найденных всеми параллельными ветвями программы.

A. *Использование библиотеки MPI*

Параллельная программа решения задачи квадратичного назначения реализуется для кластерной многопроцессорной вычислительной системы и использует библиотеку MPI [4]. Общее число итераций в алгоритме равномерно распределяется между процессорами кластера, пересылка промежуточных результатов вычислений в процессе работы алгоритма не производится.

B. *Использование библиотеки OpenMP*

Параллельная программа решения задачи назначения на узкие места реализована для многоядерной вычислительной системы и использует библиотеку OpenMP [4]. Общее число итераций в алгоритме равно произведению количества ядер в системе на количество итераций, выполняемое каждым из них.

Одной из основных проблем при реализации алгоритмов случайного поиска является быстрое получение случайного вектора с неповторяющимися элементами, описывающего решение. Если заполнять элементы вектора P случайными числами и при этом проверять, чтобы элементы не повторялись, то чем больше будет заполненных элементов, тем чаще будут выпадать повторяющиеся цифры. В результате на генерацию каждого нового вектора затрачивается много времени. Теоретическая оценка математического ожидания числа бросков Q (операций *rand*), необходимых для заполнения вектора длиной N уникальными числами, дает следующий результат:

$$Q = N \sum_{k=1}^{N} \frac{1}{k} \approx N(\ln N + 0.557).$$

При $N = 36$ значение $Q = 150$. В реализованных авторами алгоритмах используется метод ускорения генерации случайного вектора [5]. Это достигается за счет того, что случайное число получается как комбинация двух случайных чисел меньшей размерности, что снижает вероятность повторов. При этом, например, вектор длиной 36 удается получить уже с помощью 90 операций *rand*.

IV. Результаты работы алгоритмов

Поскольку случайные алгоритмы ведут неупорядоченный поиск, то вероятность нахождения точного решения будет зависеть от соотношения числа рассмотренных вариантов к общему числу возможных вариантов решений. При $N = 18$ число вариантов решений будет равно $18! \approx 6.4 \cdot 10^{15}$ и современные вычислительные системы еще позволяют провести полный перебор вариантов, однако даже при незначительном увеличении размерности это потребует больших затрат времени, уже при $N = 24$, число вариантов составит $24! \approx 0.6 \cdot 10^{24}$. С практической точки зрения представляют интерес решения, которые дают случайные алгоритмы за $10^6 - 10^{10}$ итераций. При соотношении числа итераций к числу вариантов меньшим 10^{-6}, вероятность нахождения точного решения будет пренебрежимо мало отличаться от нуля. Задачей проводимых исследований является определение точности параллельных алгоритмов случайного поиска при решении *NP*-полных задач дискретной оптимизации.

A. *Задача квадратичного назначения*

В качестве тестовой для задачи квадратичного назначения используется *тест-задача Штейнберга* [1]. Размерность задачи составляет $N = 36$, точного решения для нее в литературе не приводится. Она используется в качестве оценочной для алгоритмов решения задачи размещения элементов. Особенностью задачи является разреженная матрица **R** и слабо выраженная кластеризация элементов.

Результаты работы параллельного алгоритма [6] на многопроцессорной кластерной вычислительной системе представлены в табл. 1. В каждом случае эксперимент повторялся 5 раз, а полученные данные усреднялись.

Таблица 1

Решение задачи квадратичного назначения ($N = 36$)

Число итераций	Число процессоров			
	1	8	16	32
10^3	6820	6120	6156	5817
10^4	5901	5639	5780	5400
10^5	5566	5200	5532	5418
10^6	5150	4954	5084	4935
10^7	4751	4850	4830	4699

B. Задача назначения на узкие места

В идеальном случае оценку точности решений алгоритмов нужно давать, сравнивая их результаты с точным решением, но далеко не для всех задач такое решение может быть получено. Для задачи назначения на узкие места нижняя граница точного решения может быть получена с помощью последовательного решения задачи линейного назначения для преобразованной матрицы **R** . Этот фактор повлиял на выбор ее в качестве объекта исследований. Для рассмотренных в работе примеров такое решение составляет: $T_{react} = 141$ (для $N = 18$) и $T_{react} = 151$ (для $N = 24$).

Исследования проводились на рабочей станции, имеющей два четырехядерных процессора Intel. Число задействованных потоков совпадало с числом доступных приложению ядер. Результаты работы алгоритма представлены в табл. 2 и 3.

Таблица 2

Решение задачи назначения на узкие места ($N = 18$)

Число итераций	T_{react}					
	Эксперименты					Среднее значение
	1	2	3	4	5	
10^4	189	177	178	180	176	180
$5*10^4$	167	155	170	167	162	164
10^5	146	162	169	153	132	152
$5*10^5$	141	146	143	146	141	143
10^6	141	143	143	143	143	143
$5*10^6$	143	143	141	141	141	142
10^7	141	141	143	141	143	142

Как и в предыдущем случае, было проведено пять серий вычислительных экспериментов. Поскольку решение является целочисленным, полученные данные были усреднены с округлением. Ячейки, соответствующие точному решению задачи, выделены серым цветом. Результаты исследований быстродействия алгоритма для $N = 24$ отражены в табл. 4.

Таблица 3

Решение задачи назначения на узкие места ($N = 24$)

Число итераций	T_{react}					
	Эксперименты					Среднее значение
	1	2	3	4	5	
10^4	342	353	342	307	348	338
$5*10^4$	330	333	245	345	321	314
10^5	316	342	316	339	312	325
$5*10^5$	314	293	285	299	196	277
10^6	245	204	225	239	225	227
$5*10^6$	221	187	221	230	225	217
10^7	204	204	199	194	199	200
$5*10^7$	196	187	181	188	188	188
10^8	185	193	186	185	185	187
$5*10^8$	186	174	178	185	185	182
10^9	171	185	178	151	171	171
$5*10^9$	151	170	173	169	151	163

Таблица 4

Время решения задачи назначения на узкие места ($N = 24$)

Число итераций	Время решения (сек)					Среднее значение
	1	2	3	4	5	
10^4	0.1	0.08	0.1	0.09	0.1	0.094
$5*10^4$	0.12	0.1	0.11	0.1	0.12	0.11
10^5	0.14	0.15	0.14	0.13	0.14	0.14
$5*10^5$	0.39	0.39	0.39	0.38	0.39	0.39
10^6	0.78	0.72	0.67	0.74	0.71	0.72
$5*10^6$	3.3	3.59	3.28	3.28	3.34	3.36
10^7	6.42	6.82	6.24	6.49	6.21	6.47
$5*10^7$	32.1	31.1	32.1	32.9	32.1	32.1
10^8	66.8	61.4	66.7	63	61.5	64
$5*10^8$	319	322	319	320	320	320
10^9	662	639	626	622	642	638
$5*10^9$	3264	3322	3292	3434	3431	3349

V. Обсуждение результатов

Проанализируем полученные данные с точки зрения точности алгоритмов и эффективности их распараллеливания.

A. Задача квадратичного назначения

Результаты эксперимента (табл.1) подтверждают, что и при параллельной реализации алгоритмов с увеличением числа итераций точность решения повышается для любого количества процессоров, участвовавших в эксперименте. Итерации равномерно распределялись между узлами системы и зависимость точности решения от числа процессоров менее очевидна. Хотя лучшее решение и получается всегда при максимальном числе узлов системы, худшее не всегда соответствует системе с одним узлом. Наблюдаются и некоторые другие отклонения. Объяснить это можно тем, что инициализация генератора случайных чисел в программе происходит от системного таймера. Кластер собран на двухпроцессорных модулях, для которых характерно то, что оба процессора используют общий системный таймер. Это приводит к тому, что в процессах, работающих на таких узлах, генераторы случайных чисел достаточно сильно коррелированы. Этот факт и снижает эффективность вычислений. Для устранения этого явления следует использовать внешний генератор случайных последовательностей.

B. Задача назначения на узкие места

Во всех экспериментах был реализован алгоритм, использующий все доступные для вычислений ядра процессоров. В отличие от многопроцессорных систем, все потоки многоядерного процессора используют единственный таймер, поэтому инициализация генераторов случайных чисел проводилась с помощью случайного массива, полученного за пределами потока.

Результаты исследования точности решений параллельной реализации (табл. 2 и 3) не противоречат общепринятой теории – увеличение числа итераций повышает точность. Отметим, что точные решения задачи алгоритм стабильно находит даже при соотношении числа итераций к числу возможных вариантов решений 10^{-10} для размерности $N = 18$ и 10^{-15} для размерности $N = 24$. Казалось бы, при таком низком соотношении точные решения получить невозможно. Однако это справедливо лишь для тех задач, в которых точное решение является единственным. В работе не ставилась задача определить количество различных вариантов решений, обладающих минимальным значением T_{react}, но полученные данные косвенно подтверждают, что существует некоторое множество таких решений. Это и определяет высокую точность алгоритмов.

Рассмотрим еще один аспект, отражающий временные характеристики работы случайных алгоритмов. В табл. 4 представлены результаты исследования быстродействия алгоритма случайного поиска при решении задачи с $N = 24$. Точное решение получается алгоритмом при числе итераций 10^9 и более. При этом многопоточное приложение затрачивает на расчеты около 1 часа при использовании рабочей станции, имеющей два четырехядерных процессора Intel. Для алгоритмов, которые позволяют получить точное или близкое к нему решение, используемое для оценки точности других алгоритмов, это является приемлемым.

Но как определить необходимое число итераций, в том случае, если нам неизвестна нижняя граница оптимизируемого функционала? Одним из косвенных признаков получения алгоритмом локального оптимума, который может быть использован на практике, является тот факт, что в течение определенного числа итераций найденное решение не изменяется. В силу причин, отмеченных выше, необходимое число итераций будет зависеть не только от типа оптимизационной задачи, но также и от ее условий.

На наш взгляд, в качестве критерия окончания работы для алгоритмов случайного поиска целесообразно только временное ограничение. При необходимости реализации длительных вычислений, работу алгоритма можно разделить на несколько этапов. Например, 10^{10} итераций можно получить путем 10 запусков алгоритма с числом итераций 10^9. Кроме того, можно использовать вычислительные системы с большим числом ядер.

VI. Заключение

Таким образом, проведенные исследования подтвердили высказанное предположение о высокой эффективности параллельных реализаций алгоритмов случайного поиска при решении *NP*-полных задач дискретной оптимизации, встречающихся при цифровой обработке информации. Получаемые с их помощью решения могут быть использованы как в качестве эталонных для оценки работы быстрых алгоритмов, так и в самостоятельном виде в информационных системах, не критичных ко времени обработки.

Литература

[1] Колпаков Р.М., Посыпкин М.А. Об оценках вычислительной сложности варианта параллельной реализации метода ветвей и границ для задачи о ранце // Известия РАН: Теория и системы управления. 2011. № 5. С. 74-83.

[2] Селютин В.А. Машинное конструирование электронных устройств // М.: Сов. радио, 1977.

[3] Лупин С.А., Мью Мьинт Ту. Оценка вычислительной сложности основных этапов моделирования распределенных систем // Естественные и технические науки. 2004. № 4.

[4] Лупин С.А., Посыпкин М.А. Технологии параллельного программирования. Учебное пособие // М.: ИД «ФОРУМ»: ИНФРА-М, 2007.

[5] Лупин С.А., Бажанов Е.И., Дорошенко Е.С., Подкопаев И.В. Практическая работа на кластере под управлением Microsoft HPC Server 2008. Учебное пособие // М.: МИЭТ, 2011. 60 с.

[6] Зей Яр Вин Распараллеливание итерационных алгоритмов для многопроцессорных систем // Системный анализ и информационно-управляющие системы. Сборник научных тр. / под ред. д.т.н., проф. В.А. Бархоткина. М.: МИЭТ, 2008. С. 164-168.