ИЗВЕСТИЯ РАН. ТЕОРИЯ И СИСТЕМЫ УПРАВЛЕНИЯ, 2017, № 2, с. 48–56

ДИСКРЕТНЫЕ СИСТЕМЫ

УДК 004.031.43

АЛГОРИТМЫ КОМБИНАТОРНОЙ ОПТИМИЗАЦИИ, СОЧЕТАЮЩИЕ ЖАДНЫЕ СТРАТЕГИИ И ОГРАНИЧЕННЫЙ ПЕРЕБОР[1]

© 2017 г. В. А. Костенко

Москва, МГУ

e-mail: kostmsu@gmail.com

Поступила в редакцию 08.09.16 г.
После доработки 04.10.16 г.

Алгоритмы работают в соответствии с жадной схемой, процедура ограниченного перебора вызывается лишь на шагах, на которых жадный выбор исключает возможность получения оптимального решения. Принцип построения алгоритмов иллюстрируется на примере задачи выбора максимально совместимого числа заявок. Приводятся результаты применения алгоритмов для планирования вычислений в распределенных системах.

DOI: 10.7868/S0002338817020135

Введение. Большинство массовых задач комбинаторной оптимизации относятся к классу недетерминированно полиномиально трудных (*NP*-трудных). Под массовой задачей или просто задачей [1] будем понимать некоторый общий вопрос, на который необходимо дать ответ. Задача определяется следующей информацией:

общим списком всех ее параметров (входными данными),

формулировкой тех свойств, которым должен удовлетворять ответ или, другими словами, решение задачи.

Индивидуальная задача [1] получается из массовой задачи, если всем параметрам массовой задачи присвоить конкретные значения.

Подзадачей (частной задачей) называется такая задача [1], в которой сформулирован тот же вопрос, что и в массовой задаче, но только на некотором подмножестве множества индивидуальных задач данной массовой задачи. Другими словами, в частной задаче на возможные значения параметров наложены ограничения.

Ряд подзадач *NP*-трудной массовой задачи очень часто являются полиномиально разрешимыми. Например, массовая задача о выборе максимально совместимого числа заявок (работ) для одноприборной системы, обслуживающей без прерываний исходно заданный набор заявок с индивидуальными директивными интервалами, является *NP*-тудной. А ее частная задача, для которой время выполнения любой заявки равно ее директивному интервалу, может быть решена точным жадным алгоритмом сложности $O(n \cdot \log n)$ [2]. В этой частной задаче введено следующее ограничение на возможные значения параметров: длительность директивного интервала любой заявки равна длительности ее выполнения.

Для построения алгоритмов решения задач комбинаторной оптимизации наиболее широко используются:

метод динамического программирования [2–4],

жадные стратегии [2, 5],

метод ветвей и границ [2, 5],

итерационные схемы [6]: случайного поиска [7], имитации отжига [8, 9], генетические и эволюционные [10–12],

алгоритмы, основанные на схеме муравьиных колоний [13–16],

алгоритмы, основанные на нахождении максимального потока в транспортной сети [17–20].

[1] Работа выполнена при финансовой поддержке РФФИ (грант № 16-07-01237).

В практических приложениях, как правило, требуется разработать алгоритм решения лишь частной задачи, который обладает требуемой точностью, и его вычислительная сложность не превышает заданную. Другими словами, алгоритм не должен находить обязательно оптимальное решение задачи, а должен находить приемлемое по точности решение за приемлемое время.

Основной проблемой использования итерационных алгоритмов является проблема настройки их параметров на частную задачу таким образом, чтобы обеспечить требуемый баланс между точностью и вычислительной сложностью алгоритма [21]. В настоящее время теоретических результатов решения этой проблемы не существует. Значения параметров алгоритма в практических применениях подбираются экспериментальным путем. Кроме того, о точности и вычислительной сложности этих алгоритмов можно говорить лишь в статистическом плане. Это означает, что для индивидуальной задачи, принадлежащей рассматриваемой частной задаче, могут нарушаться заданные требования к точности и ограничения на вычислительную сложность.

Основной проблемой метода динамического программирования и метода ветвей и границ является то, что верхняя оценка вычислительной сложности алгоритмов для большинства общих задач комбинаторной оптимизации неполиномиальна. Это связано с особенностями самих методов. Так, для метода ветвей и границ из свойств функции нижней границы и функции выбора наилучшего решения никак не следует, что в ходе работы алгоритма хотя бы одно подмножество, отличное от базисного, будет отсечено. Поэтому алгоритм при решении индивидуальной задачи может в худшем случае осуществлять просмотр всех решений.

Жадные алгоритмы основываются на свойстве оптимальности для подзадач. Если оптимальное решение задачи содержит оптимальное решение ее подзадач, то задача обладает свойством оптимальности для подзадач. Жадные алгоритмы, делая на каждом шаге локально-оптимальный выбор (решая подзадачу), предполагают, что в итоге будет получено оптимальное решение задачи. Жадные алгоритмы обладают низкой вычислительной сложностью, но область их применения ограничивается частными задачами, которые имеют свойство оптимальности для подзадач. Доказательство того, что частная задача обладает свойством оптимальности для подзадач, является нетривиальной проблемой и чаще всего для этого используется подход, рассмотренный в [2]. Жадные алгоритмы допускают простой учет ограничений на корректность решения. Это свойство особенно важно при решении практических задач. Например, в задаче построения расписания обменов по шине с централизованным управлением в зависимости от используемого контроллера шины может быть от 3 до 7 дополнительных ограничений на корректность расписания [16] по сравнению с аналогичной классической задачей выбора максимально совместимого числа заявок.

Алгоритмы, основанные на схеме муравьиных колоний, обладают свойством автоматической настройки в ходе работы на индивидуальную задачу. Для применения муравьиных алгоритмов требуется сведение решаемой задачи к задаче нахождения на графе маршрута, обладающего определенными свойствами. Кроме того, как и в итерационных алгоритмах, возникает проблема настройки параметров алгоритма на частную задачу.

Алгоритмы, основанные на нахождении максимального потока в транспортной сети, являются оптимальными и имеют псевдополиномиальную сложность. Однако область их применения ограничивается только классом задач, которые могут быть сведены к задаче нахождения максимального потока в транспортной сети. Также при построении этих алгоритмов возникает проблема учета дополнительных ограничений на корректность решения.

В основу предлагаемых в работе алгоритмов, сочетающих жадные стратегии и ограниченный перебор, положены следующие основные принципы:

на каждом шаге работы алгоритма делается локально-оптимальный выбор в соответствии с используемой жадной стратегией,

на каждом шаге осуществляется проверка того, что “жадный выбор не закрывает пути к оптимальному решению”,

вызов процедуры ограниченного перебора, если проверка условия “жадный выбор не закрывает пути к оптимальному решению” дала отрицательный результат.

В разд. 1 описан предлагаемый метод построения алгоритмов, сочетающих жадные стратегии и ограниченный перебор, приведены две схемы построения алгоритмов и обоснованы их свойства, в разд. 2 и 3 соответственно рассмотрено применение метода для построения алгоритмов решения задач планирования обменов по каналу с централизованным управлением и планирования вычислений в центрах обработки данных.

1. Метод построения алгоритмов, сочетающих жадные стратегии и ограниченный перебор. Предлагаемый в работе метод построения алгоритмов, сочетающих жадные стратегии и перебор, будем рассматривать на примере задач выбора максимально совместимого числа заявок (работ). В общем случае задачу о выборе максимально совместимого числа заявок можно сформулировать следующим образом. Задано множество заявок на обслуживание и множество ресурсов, на которых могут обслуживаться заявки. Каждая заявка характеризуется набором требований к качеству обслуживания. Для каждого ресурса есть набор характеристик. Также задан набор ограничений на допустимое размещение заявок на ресурсах. Параметрами ограничений являются характеристики ресурсов и требования к качеству обслуживания заявок. Нужно разместить на заданном множестве ресурсов максимально возможное число заявок из заданного множества таким образом, чтобы выполнялись ограничения на допустимое размещение заявок. Если система является одноприборной, то требуется лишь выбрать место размещения заявки в расписании выполнения заявок.

Принцип построения алгоритмов, сочетающих жадные стратегии и перебор, заключается в следующем. На каждом шаге алгоритм выбирает заявку из множества еще не размещенных заявок и место ее размещения в соответствии с жадным критерием. В первом подходе к построению алгоритма процедура ограниченного перебора вызывается, если на очередном шаге выбранная заявка не может быть размещена. Во втором подходе к построению алгоритма процедура ограниченного перебора вызывается, если после пробного размещения на очередном шаге выбранной заявки в множестве неразмещенных заявок появляются заявки, которые не могут быть размещены. Для процедуры ограниченного перебора задается максимально допустимая глубина перебора, определяющая максимально возможное количество заявок, которые могут участвовать в переборе.

Ниже приведены схемы алгоритмов, соответствующие этим двум подходам.

Схема алгоритма 1.

1. Выбрать в соответствии с жадным критерием очередную заявку из множества еще не размещенных заявок.

2. Выбрать в соответствии с жадным критерием ресурс (место в расписании) для размещения заявки:

если такой ресурс не найден, то к п. 3,

если такой ресурс найден, то разместить на нем заявку, удалить заявку из множества неразмещенных заявок и, если это множество не пусто, перейти к п. 1, в противном случае – завершение работы.

3. Вызов процедуры ограниченного перебора (или эвристики):

успешное завершение процедуры: сохранить полученное процедурой размещение, удалить заявку из множества неразмещенных заявок и, если это множество не пусто, перейти к п. 1, в противном случае – завершение работы,

неуспешное завершение процедуры: удалить текущую заявку из множества неразмещенных заявок и, если множество неразмещенных заявок не пусто, перейти к п. 1, в противном случае – завершение работы.

Схема алгоритма 2.

1. Выбрать в соответствии с жадным критерием очередную заявку из множества еще не размещенных заявок.

2. Выбрать в соответствии с жадным критерием ресурс для размещения заявки (всегда есть хоть один ресурс), сделать пробное размещение и проверить оптимальность жадного выбора:

если в множестве неразмещенных заявок есть заявки, которые не могут быть размещены после размещения выбранной заявки, то отменить размещение и перейти к п. 3,

если в множестве неразмещенных заявок нет заявок, которые не могут быть размещены после размещения выбранной заявки, то удалить заявку из множества неразмещенных заявок и, если это множество не пустое, перейти к п. 1.

3. Вызов процедуры ограниченного перебора (или эвристики):

успешное завершение процедуры: сохранить полученное процедурой размещение, удалить заявку из множества неразмещенных заявок и, если это множество не пусто, перейти к п. 1, в противном случае – завершение работы,

неуспешное завершение процедуры: удалить текущую заявку из множества неразмещенных заявок и, если множество неразмещенных заявок не пусто, перейти к п. 1, в противном случае – завершение работы.

Рассмотрим свойства предложенных схем алгоритмов, сочетающих жадные стратегии и ограниченный перебор. Функция называется разделяемой [22] на f_1 и f_2, если она представима в виде

$$f(x, y) = f_1(x, f_2(y)).$$

Функция называется разложимой [22] на f_1 и f_2, если она разделяема на f_1 и f_2 и функция f_1 монотонно не убывает по последнему аргументу.

Теорема оптимальности [22] для разложимых функций:

$$\min_{(x,y)} f_1(x, f_2(y)) = \min_{(x)} f_1(x, \min_{(y)} f_2(y)).$$

У т в е р ж д е н и е. Если:

целевая функция задачи $f(x_1, x_2, ..., x_n)$ является разложимой на $f_1, f_2, ... f_n$,

в качестве жадного критерия выбора очередной оптимизируемой переменной (заявки) используется номер функции,

в качестве жадного критерия выбора значения переменной (места размещения заявки) используется глобальный минимум соответствующей функции,

существует решение, содержащее все заявки из исходно заданного набора, то алгоритмы, построенные по схеме 1 или 2, получат оптимальное решение и процедура ограниченного перебора вызываться не будет.

Справедливость этого утверждения следует из теоремы оптимальности разложимых функций и существования решения, содержащего все заявки. Поскольку из теоремы оптимальности следует, что все заявки могут быть размещены в соответствии с жадными стратегиями и нет заявок, которые не могут быть размещены в итоговом решении, то проверка условий необходимости вызова процедуры ограниченного перебора всегда будет отрицательной. Другими словами, решение будет получено только по жадной схеме и вычислительная сложность получения решения будет равна нижней оценке вычислительной сложности алгоритма.

Проиллюстрируем это на частной задаче о выборе максимально совместимого числа заявок для одноприборной системы, обслуживающей без прерываний исходно заданный набор заявок с индивидуальными директивными интервалами, для которой время выполнения любой заявки равно ее директивному интервалу.

Массовая задача может быть сформулирована следующим образом.

Параметры задачи:

множество заявок (работ), которые должны выполняться на системе: $J = \{j\}_{j=1}^{N_{зад}}$, здесь $N_{зад}$ – число исходно заданных заявок,

множество времен выполнения заявок: $\{t_j > 0\}_{j=1}^{N_{зад}}$,

множество директивных интервалов: $\{[s_j, f_j)\}_{j=1}^{N_{зад}}$, таких, что $\forall j$: $f_j - s_j \geq t_j$.

Расписание выполнения заявок представляет собой упорядоченное (по критерию время начала выполнения заявки) множество $H = \{j_k, s_j^*\}_{k=1}^{N_H}, j \in J$. Здесь k – порядковый номер j-й заявки в расписании, s_j^* – время начала выполнения j-й заявки в расписании H, $f_j^* = s_j^* + t_j$ – время завершения выполнения j-й заявки. Множество корректных расписаний H'^* определим набором ограничений:

$$g_1: (\forall j \in H) \Rightarrow ((s_j^* \geq s_j) \wedge (f_j^* \leq f_j)),$$

$$g_2: (\forall j \in H) \Rightarrow (f_j^* - s_j^* = t_j),$$

$$g_3: (\forall (j, l) \in H, \quad j \neq l) \Rightarrow (((s_j^* < s_l^*) \vee (s_j^* \geq f_l^*)) \wedge ((f_j^* \leq s_l^*) \vee (f_j^* > f_l^*))).$$

Ограничения g_1, g_2, g_3 соответственно означают:

1) интервал выполнения каждой заявки располагается в рамках ее директивного интервала;

2) не допустимы прерывания;

3) интервалы выполнения заявок не пересекаются.

Решение задачи – корректное расписание выполнения работ, содержащее максимально возможное число заявок из исходно заданного множества $J = \{j\}_{j=1}^{N_{зад}}$:

$$\max_{H \in H^*} |H|.$$

Данная задача известна в теории расписаний как задача о выборе максимального числа совместимых заявок и является *NP*-трудной.

Частная задача рассмотренной выше массовой задачи о выборе максимального числа совместимых заявок, для которой время выполнения любой заявки равно ее директивному интервалу, имеет те же самые параметры и решение, только наложены ограничения на возможные отношения времен выполнения заявок и их директивных интервалов: $\forall\ j\colon\ t_j = f_j - s_j$.

Пусть в качестве жадного критерия выбора заявки используется правая граница директивного интервала, в качестве жадного критерия выбора места расположения – заявка размещается в конец текущего списка заявок, размещенных на предыдущих шагах. В алгоритме, построенном по схеме 1, заявка всегда будет размещена. Это следует из условия утверждения: "существует решение, содержащее все заявки из исходно заданного набора" и ограничения на возможные значения параметров задачи $\forall\ j\colon\ t_j = f_j - s_j$. Эти условия означают, что директивные интервалы заявок не пересекаются. Следовательно, при решении этой частной задачи процедура ограниченного перебора вызываться не будет.

В алгоритме, построенном по схеме 2 с теми же жадными критериями, процедура ограниченного перебора вызываться не будет, так как при пробном размещении заявки на очередном шаге алгоритма не будет заявок, которые не могут быть размещены после размещения рассматриваемой заявки. Это следует из того, что директивные интервалы заявок не пересекаются.

Схема 1 будет иметь меньшую вычислительную сложность для задач, в которых возможно оптимальное размещение большинства заявок по жадному критерию. Если используемые в алгоритме жадные критерии для частной задачи позволяют получать оптимальное решение, то вычислительная сложность алгоритма для этой частной задачи будет равна вычислительной сложности алгоритма. В случае, когда жадные критерии "плохо подходят" для частной задачи, точность алгоритма, построенного по схеме 1, при одинаковой глубине перебора будет хуже точности алгоритма, построенного по схеме 2.

2. Применение подхода для построения алгоритмов планирования вычислений. В данном разделе рассмотрим алгоритмы, построенные в соответствии с первой схемой, для решения задачи распределения ресурсов в центрах обработки данных [23, 24] и алгоритм, построенный в соответствии со второй схемой, для решения задачи построения расписаний обменов по шине с централизованным управлением [25].

2.1. Алгоритм построения расписаний обменов по шине с централизованным управлением. К шине (каналу обмена) подключен ограниченный набор оконечных устройств, которые являются источниками/приемниками информации, передаваемой по шине. Одно из оконечных устройств назначается контроллером шины, который управляет обменом информации в соответствии с предварительно построенным расписанием обменов и осуществляет контроль состояния других оконечных устройств. Эти оконечные устройства лишь выполняют адресованные им команды контроллера. Обмен информацией осуществляется асинхронно путем поочередной передачи информации по принципу "команда–ответ". Информация передается в виде сообщений, которые могут состоять из командных слов, слов данных и ответных слов [26].

Задача построения расписания обменов по шине может рассматриваться как задача о выборе максисимально совместимого числа заявок для одноприборной системы, обслуживающей без прерываний исходно заданный набор заявок с индивидуальными директивными интервалами (см. разд. 2), в которой на корректность расписания кроме ограничений g_1, g_2, g_3 наложены дополнительные ограничения:

g_4 : число работ в цепочке не должно превышать заданного значения;

g_5 : суммарная длительность выполнения работ цепочки не должна превышать заданного значения;

g_6 : интервал времени между последовательными цепочками должен быть не меньше заданного значения.

Эти ограничения обусловлены особенностями реализации контроллера шины. Цепочка работ – это последовательность работ, следующих непрерывно друг за другом, т.е. $TaskChain = \{[s_i^*, f_i^*)\}_{i=0}^{k}, i = \overline{1, k}: s_i^* = f_{i-1}^*$.

Алгоритм построения расписания обменов [25] разработан в соответствии со второй схемой. В качестве жадного критерия для выбора очередной работы из списка еще не размещенных работ используется $f_i' = s_i + t_i$ – самый ранний срок возможного завершения работы. На каждом шаге осуществляется проверка оптимальности жадного выбора. Она заключается в пробном размещении работы в расписании и проверки того, что после пробного размещения очередной работы в списке неразмещенных работ не появились работы, которые не могут быть размещены в расписании.

На каждом шаге алгоритма выбранная работа размещается в расписании без нарушения условий корректности расписания и вычисляется ближайший такт шины, с которого может выполняться следующая работа. Ближайший такт шины вычисляется с учетом дополнительных ограничений на корректность расписания и времени выполнения размещенной работы. После этого корректируются директивные интервалы еще не размещенных в расписании работ, и на следующем шаге осуществляется выбор работы в соответствии с жадным критерием.

Процедура ограниченного перебора формирует список работ, которые имеют пересечение директивных интервалов с работой, рассматриваемой на очередном шаге алгоритма. Размер списка является параметром алгоритма и определяет глубину перебора. В результате работы алгоритма перебора определяется набор совместимых работ максимальной длины. Если таких наборов несколько, то выбирается набор с наименьшим временем освобождения шины и в качестве работы, размещаемой на текущем шаге, выбирается первая работа из этого набора.

Выбор из множества J работы по критерию минимально возможное время завершения выполнения работы позволяет избежать перебора при проверке дополнительных ограничений $\{g_i, i \in (4,5,6)\}$. Так как если выбранная по этому критерию работа нарушает ограничения, то не существует в множестве неразмещенных работ работа, которая на данном шаге алгоритма может быть размещена в расписание без нарушения этих ограничений.

Для частных задач построения расписаний обменов для бортовых авиационных систем оказалось, что более 90% работ размещаются в конечном итоге в расписании в соответствии с жадным критерием, т.е. для них не вызывается процедура ограниченного перебора. Для этих задач сложность алгоритма составляет в среднем 10% от верхней оценки сложности. Верхняя оценка вычислительной сложности алгоритма равна $O(N_j^2 + N_j F(L))$, где $F(L)$ – сложность операции ограниченного перебора глубины L. В [25] предложены и исследованы различные алгоритмы построения этой операций. Если время выполнения любой работы из исходного набора равно директивному интервалу работы, то алгоритм строит точное расписание.

2.2. Алгоритмы распределения ресурсов в центрах обработки данных. Будем рассматривать центры обработки данных (ЦОД), работающие в режиме инфраструктура как услуга (IaaS – Infrastructure-as-a-Service). Задача планирования вычислений в ЦОД заключается в построении отображения запросов на создание виртуальных машин и виртуальных систем хранения данных, соединенных виртуальными каналами на вычислительные серверы и серверы хранения данных, и построении маршрутов для виртуальных каналов в сети обмена ЦОД. При работе ЦОД в режиме IaaS необходима возможность задания критериев качества сервиса SLA (SLA – Service Level Agreement) для всех типов ресурсов: систем хранения данных, вычислительных и сетевых ресурсов. Вычислительные ресурсы, системы хранения данных и сетевые ресурсы должны рассматриваться как планируемые типы ресурсов и их планирование должно происходить согласованно в смысле соблюдения соглашений о качестве сервиса. Для получаемых отображений виртуальных ресурсов на физические ресурсы ЦОД необходимо гарантированное выполнение запрошенных соглашений о качестве сервиса. В дальнейшем будем использовать математическую постановку задачи из работы [23].

Модель физических ресурсов ЦОД задается размеченным графом $H = (P \cup M \cup K, L)$, где P – множество вычислительных узлов, M – множество хранилищ данных, K – множество коммутационных элементов сети обмена ЦОД, L – множество физических каналов передачи данных. На множествах P, M, K и L определены векторные функции скалярного аргумента, задающие соответственно характеристики вычислительных узлов, хранилищ данных, коммутационных элементов и каналов передачи данных. В дальнейшем будем их называть функциями

разметки графа H. Например, для вычислительного узла может задаваться количество ядер, объем оперативной и дисковой памяти.

Ресурсный запрос будем описывать размеченным графом $G = (W \cup S, E)$, где W – множество виртуальных машин, S – множество виртуальных систем хранения данных (storage-элементов), E – множество виртуальных каналов передачи данных. На множествах W, S и E определены векторные функции скалярного аргумента, задающие характеристики запрашиваемого виртуального элемента (требуемое качество сервиса).

Назначением ресурсного запроса будем называть отображение

$$A: G \to H = \{W \to P, S \to M, E \to \{K, L\}\}.$$

Между характеристиками запросов и соответствующими характеристиками физических ресурсов возможно три типа отношений. Обозначим через x характеристику элемента запроса и через y – соответствующую ей характеристику физического ресурса. Тогда эти отношения можно записать следующим образом.

1. Недопустимость перегрузки “емкости” физического ресурса

$$\sum_{i \in R_j} x_i \le y_j,$$

здесь R_j – множество запросов, назначенных на выполнение на физическом ресурсе j.

2. Соответствие типа физического и виртуального ресурсов: $x = y$.

3. Наличие требуемых характеристик у физического ресурса: $x \le y$.

Отображение A называется корректным, если для всех физических ресурсов и всех их характеристик выполняются отношения 1–3.

Остаточным графом доступных ресурсов H_{res} называется граф H, в котором переопределены значения функций разметки по характеристикам, которые должны удовлетворять отношению 1. Значение каждой характеристики физического ресурса уменьшается на сумму значений соответствующей характеристики виртуальных ресурсов, назначенных на этот физический ресурс.

Входом алгоритма является остаточный граф доступных ресурсов H_{res} и множество ресурсных запросов $\{G_i\}$. В множество $\{G_i\}$, кроме вновь поступивших запросов, могут входить выполняемые виртуальные ресурсы, для которых допустима миграция. Если в $\{G_i\}$ есть виртуальные ресурсы, то для элементов графа H_{res}, на которых выполняются виртуальные ресурсы, переопределяются (увеличиваются) значения функций разметки графа H по характеристикам, которые должны удовлетворять отношению 1.

Требуется: из множества $\{G_i\}$ разместить на выполнение в ЦОД максимальное число запросов, таких, что отображения $\{A_i: G_i \to H, i = \overline{1, n}\}$ являются корректными. Выполняемые виртуальные ресурсы, включенные в множество $\{G_i\}$, сниматься с выполнения не должны.

Выходом алгоритма является множество отображений ресурсных запросов на физические ресурсы $\{A_i: G_i \to H, i = \overline{1, n}\}$ и множество репликаций $\{R_i\}$, $i = 0,1, \ldots$, storage-элементов.

Наиболее полный обзор известных алгоритмов планирования вычислений в ЦОД приведен в [27]. На практике в основном используются жадные алгоритмы. Это обусловлено тем, что они имеют низкую вычислительную сложность. Недостатком жадных алгоритмов является то, что на разных частных задачах их точность может сильно отличаться. В [23, 24] предложены алгоритмы планирования в ЦОД, основанные на сочетании жадных стратегий и ограниченного перебора. Они построены в соответствии с первой схемой и позволяют задавать баланс между вычислительной сложностью и точностью алгоритма. В [24] предложен алгоритм с последовательным планированием запросов (для назначения по жадному критерию выбирается запрос и назначаются все его элементы). Он позволяет отображать запросы на физические ресурсы ЦОД компактно с точки зрения суммарной длины маршрутов для виртуальных каналов запроса. В [23] предложен алгоритм с последовательным согласованным планированием различных типов ресурсов (сначала планируются виртуальные машины всех поступивших запросов, затем storage-элементы и после этого строятся маршруты для виртуальных каналов). Он позволяет обеспечить лучшую точность по сравнению с первым алгоритмом в случае, когда критическими ресурсами являются вычислительные серверы и серверы хранения данных.

Алгоритм состоит из трех шагов:

Ш а г 1. Назначение виртуальных машин на вычислительные серверы.

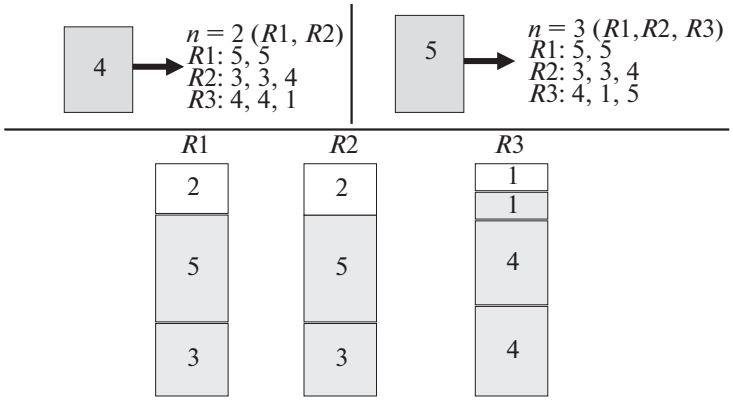

Рисунок. Пример выполнения процедуры ограниченного перебора

Ш а г 2. Назначение виртуальных систем хранения данных на серверы хранения данных.

Ш а г 3. Для полученных отображений виртуальных машин и виртуальных систем хранения данных на физические ресурсы построение маршрутов виртуальных каналов в сети обмена ЦОД.

Общая схема процедуры выполнения шагов 1 и 2 следующая [23].

1. Из множества ресурсных запросов $\{G_i\}$ выбрать очередной запрос G_i в соответствии с жадным критерием K_G.

2. Выбрать очередной элемент e (виртуальную машину $e \in W, W \in G_i$ или storage-элемент $e \in S, S \in G_i$) в соответствии с жадным критерием K_e.

3. Сформировать множество физических ресурсов P_h ($P_h \subseteq P$ или $P_h \subseteq M$ соответственно), на которые может быть назначен выбранный элемент e, т.е. для которого выполнены отношения корректности отображения в случае назначения запроса e на физический ресурс.

3.1. Если множество P_h пусто, то вызвать процедуру ограниченного перебора. Если процедура возвращает неуспех, то запрос G_i не может быть назначен: удалить ранее назначенные элементы запроса G_i и переопределить значения характеристик физических ресурсов, которые должны удовлетворять отношению 1. Если множество $\{G_i\}$ не пусто, перейти на шаг 1; в противном случае – завершить работу алгоритма.

3.2. Если множество P_h не пусто, то выбрать физический элемент $p_h \in P_h$ для назначения в соответствии с жадным критерием K_{ph} и назначить элемент запроса e на физический элемент p_h и переопределить значения его характеристик. Перейти на шаг 2, если есть нерассмотренные элементы запроса. Перейти на шаг 1, если множество $\{G_i\}$ не пусто; в противном случае – завершить работу алгоритма.

Принцип работы процедуры ограниченного перебора проиллюстрирован на рисунке.

Глубина перебор ограничивается заданным числом n, которое указывает максимальное количество физических элементов, среди которых можно производить переназначение (т.е. при $n = 2$ назначенные элементы могут сниматься и переназначаться одновременно не более чем с двух физических серверов). На рисунке слева изображена ситуация, в которой виртуальная машина, требующая четыре ядра, не может быть назначена для выполнения ни на один из физических серверов, несмотря на то, что суммарного количества ядер достаточно для ее назначения. Цветом закрашены назначенные на серверы $R1$, $R2$, $R3$ виртуальные машины, и для каждой машины указано количество занимаемых ядер, белый прямоугольник – количество свободных ядер на сервере. При переназначении виртуальных машин, назначенных на серверы $R1$, $R2$, фрагментация устраняется и виртуальная машина может быть назначена, т.е. в этом случае достаточно глубины перебора $n = 2$. На рисунке справа изображена ситуация, в которой виртуальная машина требует пять ядер. В этом случае, если глубина перебора равна двум, машина не может быть назначена. Но если задана глубина перебора $n = 3$, то виртуальная машина будет назначена. Другими словами, увеличивая допустимую глубину перебора, можно повышать точность алгоритма, но при этом увеличивается его вычислительная сложность. Для частной задачи уменьшить вычислительную сложность алгоритма можно подбором наилучших для задачи жад-

ных критериев K_G, K_e и K_{ph}, что позволяет сократить количество вызовов процедуры ограниченного перебора.

Заключение. Предложенный метод построения алгоритмов, сочетающих жадные стратегии и ограниченный перебор, дает возможность задавать требуемый баланс между точностью и сложностью алгоритма путем выбора значения максимально допустимой глубины перебора. Метод допускает настройку на частную задачу путем подбора жадных критериев. Использование метода для построения алгоритмов планирования вычислений в системах реального времени и центрах обработки данных показало, что для большинства задач более 90% работ размещаются в расписании в соответствии с жадным критерием, т.е. для них не вызывается процедура ограниченного перебора и соответственно время работы алгоритма незначительно превышает время работы жадного алгоритма, но предложенные алгоритмы в отличие от жадных находят оптимальное решение.

СПИСОК ЛИТЕРАТУРЫ

1. *Гэри М., Джонсон В.* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. 416 с.
2. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М.: МЦНМО, 2000. 1296 с.
3. *Мину М.* Математическое программирование. Теория и алгоритмы. М.: Наука, 1990. 488 с.
4. *Беллман Р.* Динамическое программирование. М.: Изд-во иностр. лит., 1960.
5. Теория расписаний и вычислительные машины / Под ред. Э.Г. Коффмана. М.: Наука, 1984. 334 с.
6. *Костенко В.А.* Алгоритмы построения расписаний для вычислительных систем реального времени, допускающие использование имитационных моделей // Программирование. 2013. № 5. С. 53–71.
7. *Растригин Л.А.* Статистические методы поиска. М.: Наука, 1968.
8. *Калашников А.В., Костенко В.А.* Параллельный алгоритм имитации отжига для построения многопроцессорных расписаний // Изв. РАН. ТиСУ. 2008. № 3. С. 133–142.
9. *Зорин Д.А., Костенко В.А.* Алгоритм синтеза архитектуры вычислительной системы реального времени с учетом требований к надежности // Изв. РАН. ТиСУ. 2012. № 2. С. 124–131.
10. *Holland J.N.* Adaptation in Natural and Artificial Systems. Ann Arbor. Michigan: Univ. of Michigan Press, 1975.
11. *Скобцов Ю.А.* Основы эволюционных вычислений. Донецк: ДонНТУ, 2008. 326 с.
12. *Батищев Д.И., Гудман Э.Д., Норенков И.П., Прилуцкий М.Х.* Метод комбинирования эвристик для решения комбинаторных задач упорядочивания и распределения ресурсов // Информационные технологии. 1997. № 2. С. 29–32.
13. *Dorigo M.* Optimization, Learning and Natural Algorithms // PhD Thesis. Dipartimento di Elettronica. Milano: Politechnico Di Milano, 1992.
14. *Костенко В.А., Плакунов А.В.* Алгоритм построения одноприборных расписаний, основанный на схеме муравьиных колоний // Изв. РАН. ТиСУ. 2013. № 6. С. 87–96.
15. *Штовба С.Д.* Муравьиные алгоритмы: теория и применение // Программирование. 2005. № 4. С. 1–15.
16. *Blum C., Sampels M.* Ant Colony Optimization for FOP Shop Scheduling: A Case Study on Different Pheromone Representation // Proc. Congr. on Evolutionary Computation. Los Alamitos, CA: IEEE Computer Society Press, 2002. V. 2. P. 1558–1563.
17. *Federgruen A., Groenevelt H.* Preemptive Scheduling of Uniform Machines by Ordinary Network Flow Technique// Management Science. 1986. V. 32. № 3.
18. *Gonzales T., Sanhi S.* Preemptive Scheduling of Uniform Processor Systems // Association for Computing Machinery. 1978. V. 25. № 1.
19. *Гуз Д.С., Фуругян М.Г.* Планирование вычислений в многопроцессорных АСУ реального времени с ограничениями на память процессоров // АиТ. 2005. № 2. С. 138–147.
20. *Косоруков Е.О., Фуругян М.Г.* Некоторые алгоритмы распределения ресурсов в многопроцессорных системах // Вестн. МГУ. Сер. 15. 2009. № 4. С. 34–37.
21. *Костенко В.А., Фролов А.В.* Генетический алгоритм с самообучением // Изв. РАН. ТиСУ. 2015. № 4. С. 24–38.
22. *Новикова Н.М.* Основы оптимизации. Курс лекций. М.: Изд-во МГУ, 1998. 65 с.
23. *Вдовин П.М., Костенко В.А.* Алгоритм распределения ресурсов в центрах обработки данных с раздельными планировщиками для различных типов ресурсов // Изв. РАН. ТиСУ. 2014. № 6. С. 80–93.
24. *Зотов И.А., Костенко В.А.* Алгоритм распределения ресурсов в центрах обработки данных с единым планировщиком для различных типов ресурсов // Изв. РАН. ТиСУ. 2015. № 1. С. 61–71.
25. *Костенко В.А., Гурьянов Е.С.* Алгоритм построения расписаний обменов по шине с централизованным управлением и исследование его эффективности // Программирование. 2005. № 6. С. 67–76.
26. Государственный стандарт СССР “Интерфейс магистральный последовательный системы электронных модулей” ГОСТ 26765.52-87.
27. *Jiangtao Zhang, Hejiao Huang, Xuan Wang.* Resource Provision Algorithms in Cloud Computing: A Survey Network and Computer Applications. 2016. V. 64. Issue C. P. 23–42.