УДК 681.3.06+519.68

ГЕНЕТИЧЕСКИЙ АЛГОРИТМ СОСТАВЛЕНИЯ РАСПИСАНИЙ ДЛЯ РАСПРЕДЕЛЕННЫХ ГЕТЕРОГЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Т. С. Шаповалов[1], **В. В. Пересветов**[1]

Рассмотрена задача составления расписаний в распределенных гетерогенных вычислительных системах. Предложен генетический алгоритм поиска расписаний, учитывающий особенности предметной области. Представлены результаты численных экспериментов. Работа выполнена при финансовой поддержке Дальневосточного отделения РАН (код проекта 06–I–П15–056).

Ключевые слова: составление расписаний, генетический алгоритм, распределенные гетерогенные вычислительные системы.

1. Введение. Эффективность использования вычислительных систем высокой производительности, таких как вычислительные кластеры или распределенные гетерогенные вычислительные системы (РГВС), существенно зависит от методов планирования заданий. Планирование — процесс распределения вычислительной работы между процессорами с целью уменьшения общего времени выполнения заданий. Частью указанного процесса является решение задачи составления расписаний. Для достаточно сложных задач составления расписаний число вариантов возможных назначений на процессоры столь велико, что решение методом перебора получить за приемлемое время невозможно. В общем случае задачи составления расписаний являются NP-полными [1]. Это означает, что для решения сложных задач составления расписаний быстрый (за полиномиальное время) и точный алгоритм не может быть найден, однако могут быть разработаны эффективные алгоритмы нахождения решения с ослаблением некоторых требований в постановке задачи. В нашем случае достаточно получить за некоторое ограниченное время субоптимальное решение — расписание, длина которого будет больше длины оптимального (точного решения) на некоторую небольшую величину, например не более 3%.

Для решения задач составления расписаний нашел применение генетический алгоритм (ГА) — один из видов стохастических алгоритмов, который включает в себя эвристические методы, моделирующие эволюционные механизмы отбора и поиска возможных решений. Эвристические (heuristic) методы основаны на опыте решения похожих задач и используются для решения задач, не имеющих строго формализованного алгоритма, или в случаях, когда исходных данных недостаточно для применения подобного алгоритма. В большинстве случаев эвристика — прием, позволяющий сокращать число рассматриваемых вариантов при поиске решения задачи, однако этот прием не гарантирует получение точного (наилучшего) решения [2].

В настоящей статье предлагается ГА составления расписаний для РГВС. Проводится численное исследование его эффективности для двух вариантов конфигурации вычислительных ресурсов РГВС и различных размеров очередей заданий. Выделенное время на поиск расписания может быть различным, но ограниченным. Такие требования типичны в задачах составления расписаний для РГВС, функционирующей в режиме коллективного доступа к ее ресурсам. В случае применения ГА можно прервать процесс решения задачи составления расписаний в любой момент времени после некоторого числа поколений.

2. Методы составления расписаний. Методы составления расписаний классифицируются на точные и эвристические. В первой группе можно выделить метод ограниченного перебора, в котором поиск осуществляется в дереве возможных решений, составленном на основе зависимостей между задачами. Одним из вариантов составления оптимального расписания данным методом является перебор всех ветвей дерева до тех пор, пока не будет найдено удовлетворительное расписание. При изменении ограничений на расписания переборные методы требуют отдельного алгоритма, приводящего ветви дерева в согласованное состояние. Кроме того, в больших задачах дерево может приобрести слишком большой размер. Другой точный метод — ветвей и границ, предложенный в работе [3], был впоследствии адаптирован в различных формах для составления расписаний [4–6].

[1] Вычислительный центр Дальневосточного отделения РАН (ВЦ ДВО РАН), ул. Ким-Ю-Чена, 65, 680063, г. Хабаровск; Т.С. Шаповалов, научн. сотрудник, e-mail: shapovalovts@gmail.com; В. В. Пересветов, ст. научн. сотрудник, e-mail: peresv@as.khb.ru

Эвристические методы также широко используются при решении задач планирования. В этом случае эвристики трактуются как правила планирования. Данный вид эвристик, оперируя множеством задач, устанавливает порядок их запуска на выполнение. Если задачу можно сопоставить с различными ресурсами, то эвристика должна указывать эти ресурсы. В работе [7] приводится обзор ряда эвристик для задач планирования. В [8] сравниваются восемь стандартных эвристик на множестве однотипных ресурсозависимых задач. В [9] сравнению подвергалась производительность программ, основанных на эвристиках с методом ограниченного перебора. Результаты сравнения показали, что алгоритмы, основанные на эвристиках, плохо приспособлены для ситуаций, в которых ресурсы сильно ограничены. В работе [10] производится сравнение мультиэвристических (использующих несколько эвристических правил для интерпретации ситуации и стратегий поиска решения) подходов. Акцентируя внимание на важности развития мультиэвристических подходов, автор отмечает, что эти алгоритмы лучше приспособлены для многомерных топологий пространств поиска. Применение эвристических правил в алгоритмах планирования описывается также в работах [11–14]. В работе [10] описываются методы решения задач планирования, использующие экспертные и основанные на знаниях (knowledge-based) системы. Каждый набор экспертных правил детально прорабатывается под ту или иную задачу, поэтому алгоритмы данной группы являются специализированными. Системы, основанные на знаниях, делят задачу на подзадачи, решением которых занимаются подпрограммы-агенты. Отметим следующие реализации систем планирования, построенные на методах искусственного интеллекта: ISIS [15], OPIS [16], MicroBOSS [17], DSS [10].

Одна из первых попыток использования ГА для составления расписаний была предпринята в работе [18]. В ней отмечается, что существует множество реальных задач планирования, которые достаточно сложны в случае применения традиционных методов решения. Отмечается также, что ГА, будучи по природе своей стохастическими, имеют возможность уходить от субоптимальных решений. В ряде случаев ГА, эффективный для одного класса задач, может быть успешно использован и для других. Часто для увеличения производительности используется предметнозависимая информация. В работе [19] приводится сравнение нескольких различных вариантов ГА для составления расписаний и делается вывод, что чем больше предметнозависимой информации используется в алгоритмах, тем выше может быть их эффективность. Кроме того, находят применение смешанные методы, в которых ГА комбинируются с традиционными. В [20] исследуются сочетания ГА с методами, основанными на эвристиках упорядочения. Использование ГА для усовершенствования эвристик в различных эвристических методах описано в [21].

В работе [22] приводятся различные классификации алгоритмов составления расписания заданий в РГВС и требования, предъявляемые к ним. В [23] описан ряд испытаний ГА и различных детерминированных методов составления расписаний. Они показали, что использование ГА вместо детерминированных алгоритмов может улучшить эффективность использования ресурсов РГВС. Используются также и другие методы составления расписаний для РГВС [24–26]. Сравнение различных эвристических методов составления расписаний в гетерогенных вычислительных средах приводится в работе [27]. В ней автор на 16 узлах и 512 тестовых задачах подвергает сравнению такие эвристические методы, как адаптивная балансировка загрузки, ГА, имитации отжига и др. Из всех методов лучшие результаты по различным критериям показали ГА.

3. Постановка задачи. Решаемая задача состоит в нахождении такого порядка запуска заданий на определенных вычислительных ресурсах в РГВС, который позволил бы завершить все задания в кратчайший срок. Другими словами, алгоритм решает задачу поиска расписания для РГВС, удовлетворяющего условию минимума длины расписания. Под длиной расписания понимается отрезок времени между запуском первых заданий и временем завершения последнего задания.

Математическая формулировка задачи состоит в следующем. Пусть для множества $U = \{u_0, \ldots, u_N\}$ бесприоритетных вычислительных процессов с определенным отношением предшествования "$\prec$" задан отрезок планирования $[0, T]$ и $t \in [0, T]$ — независимая переменная времени. Пусть $\Phi = \{\phi_0, \ldots, \phi_G\}$ — множество процессоров, принадлежащих РГВС; $\alpha_i(\phi_g)$ и $\beta_i(\phi_g)$ — моменты начала и завершения выполнения процесса $u_i \in U$, $i = \overline{1, N}$ на процессоре ϕ_g. Расписание определяется [28] как множество

$$s = \Big\{\big(\alpha_i(\phi_g),\, \beta_i(\phi_g)\big)\Big\}, \quad \alpha_i(\phi_g),\, \beta_i(\phi_g) \in [0, T], \quad \alpha_i(\phi_g) < \beta_i(\phi_g), \quad i = \overline{1, N}, \quad g = \overline{1, G}. \tag{1}$$

Зададим уровни наличия ресурсов, доступных процессору $\phi_g \in \Phi$ в момент времени t, следующим образом: $r_t(\phi_g) = \{r_t^j(\phi_g)\}$, $j = \overline{1, J}$, $t \in [0, T]$, где J — число различных типов ресурсов. Ресурсами являются: оперативная память, тип и версия операционной системы, архитектура процессора и др. Верхнюю границу каждого ресурса, доступного процессору $\phi_g \in \Phi$, будем обозначать через $r^j(\phi_g)$, $r_t^j(\phi_g) \leqslant r^j(\phi_g)$ для всех t и j. Каждая задача $u_i \in U$ характеризуется потребностью $\overline{r}_i^j$ в ресурсах типа j.

Ограничения на составление расписаний задаются исходя из имеющихся ресурсов и набора заданий. Пусть $U_t = \{u_i \in U \mid \alpha_i(\phi_g) \leqslant t \leqslant \beta_i(\phi_g)\}$ — множество процессов, выполняемых в момент времени t, тогда решение задачи нахождения расписания (1) является допустимым, если для всех $u_i, u_j \in U$ и для всех $\phi_g \in \Phi$ выполняются следующие условия:

$$\beta_i(\phi_g) \leqslant T, \quad u_i \prec u_j \Longrightarrow \beta_i(\phi_g) \leqslant \alpha_j(\phi_g), \quad \sum_{u_i \in U_t} \overline{r}_i^j \leqslant r_t^j(\phi_g) \quad \forall t \in [0, T]. \tag{2}$$

Длительность $\tau_i(\phi_g)$ выполнения вычислительного процесса u_i на процессоре ϕ_g рассчитывается во время работы алгоритма исходя из значений, указанных пользователем на этапе постановки задания в очередь. При поиске расписания для гетерогенных систем необходимо масштабировать длительность выполнения процессов в соответствии с фактически используемым типом процессора. Новое значение длительности $\tau_i(\phi_g)$ вычисляется по следующей формуле:

$$\tau_i(\phi_g) = \tau_i(\phi_h)\, \kappa^c(\phi_h, \phi_g)\, \kappa^m(\phi_h, \phi_g)\, \frac{C(\phi_h)}{C(\phi_g)}, \tag{3}$$

где $\kappa^c(\phi_h, \phi_g)$ — коэффициент перерасчета производительности различных типов процессоров одинаковой тактовой частоты, $C(\phi_g)$ и $C(\phi_h)$ — значения тактовых частот этих процессоров. Коэффициент $\kappa^c(\phi_h, \phi_g)$ зависит от числа конвейеров и иных различий в архитектурах процессоров. Проводить перерасчет производительности следует также с учетом эффективности работы с памятью, которая определяется типом процессора, тактовой частотой и объемом запрашиваемой программой памяти. Эффективность работы с памятью учитывается в формуле (3) с помощью коэффициента $\kappa^m(\phi_h, \phi_g)$. Учет всех аспектов производительности не является предметом внимания данной работы.

Эффективность расписаний, на основании которой рассчитывается функция пригодности ГА, оценивается по следующему критерию длины расписания:

$$l(s_k) = \max_{1 \leqslant i \leqslant N} \beta_i^k(\phi_g),$$

где $k = \overline{1, M}$, $g = \overline{1, G}$ и $\beta_i^k(\phi_g)$ — момент завершения выполнения i-го процесса в расписании s_k на процессоре ϕ_g. Тогда в качестве наилучшего расписания выбирается такое, при котором $l(s') = \min\limits_{1 \leqslant k \leqslant M} l(s_k)$.

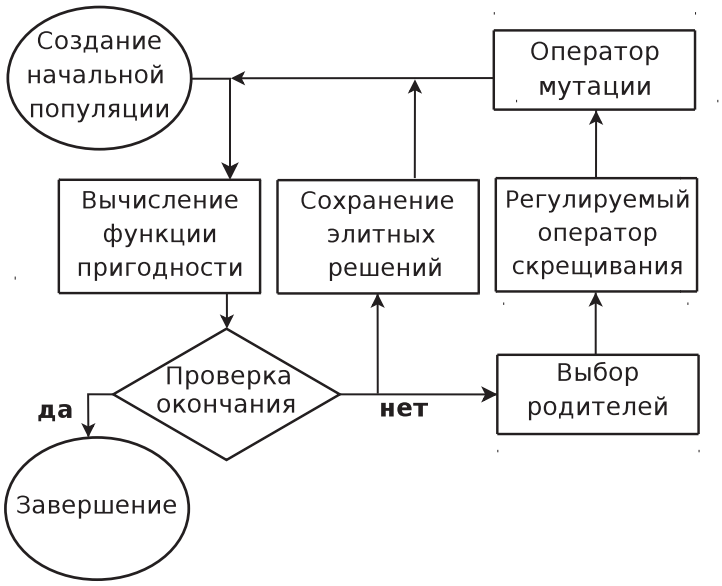

Рис. 1. Блок-схема алгоритма

В связи с тем, что решаемая задача является NP-полной [1], для которой сложность решения при увеличении ее размерности растет экспоненциально, рассматриваемая задача формулируется с ослаблением: достаточно для задач большой размерности получить только субоптимальное решение. При этом длина расписания наилучшего решения уменьшается с ростом отпущенного времени на расчеты или держится на некотором достигнутом уровне. Поэтому в данной работе большое внимание уделяется нахождению параметров ГА, обеспечивающих максимальную скорость получения субоптимальных решений.

4. Описание алгоритма. Блок-схема предлагаемого алгоритма поиска расписания в РГВС представлена на рис. 1.

В ГА каждое из расписаний s_k должно быть представлено в закодированной форме. Для этого необходимо оперировать тремя параметрами: процессором ϕ_g, вычислительным процессом u_i и временем $\alpha_i(\phi_g)$ начала выполнения i-го вычислительного процесса на процессоре ϕ_g. При этом необходимо отметить, что задания могут состоять из одного или нескольких процессов. Под процессом понимается один поток выполнения программы пользователя с отдельным адресным пространством. Тройка параметров $\langle \phi_g, u_i, \alpha_i(\phi_g) \rangle$ определяет место и момент запуска процесса в РГВС и является геном. Хромосомой тогда может служить последовательное расположение указанных троек в порядке запуска соответствующих процессов. В предлагаемом алгоритме идентификаторы процессоров вынесены за пределы области

кодирования генов в отдельный вектор. В каждом элементе данного вектора хранится число последовательно расположенных генов в хромосоме, представляющих собой пары $\langle u_i, \alpha_i(\phi_g)\rangle$. Помимо экономии оперативной памяти при нахождении расписаний для больших популяций, описанный вид хромосом позволяет оставлять гены одного процессора расположенными физически последовательно в оперативной памяти после применения оператора мутации обмена генов. Если бы идентификаторы процессоров были расположены непосредственно в гене, то в результате обмена двух генов местами необходимо было бы либо менять идентификаторы процессоров местами, что нарушало бы непрерывность расположения генов одного процессора, либо производить копирование генов, которые в данном случае состоят из трех компонент, что замедлило бы работу программы. Сохранение непрерывного расположения генов дает возможность генетическому оператору скрещивания копировать в дочернюю хромосому за раз непрерывные участки оперативной памяти родительских хромосом, что позволяет существенно ускорить процесс скрещивания. Необходимым свойством алгоритма составления расписания является возможность сохранения целостности расписания. Под целостностью понимается неизменность количества процессов в расписании, определенных на этапе постановки заданий в очередь. Классические варианты генетических операторов могут нарушать целостность расписаний, изменяя число тех или иных генов. В результате в расписании одних процессов может оказаться больше или меньше, чем изначально требовалось. Так, классический вариант оператора мутации заменяет один ген другим случайным образом, не учитывая того, что полученный в результате ген может уже содержаться в хромосоме. Аналогична ситуация и с классическим вариантом оператора скрещивания. Поэтому в описываемом в настоящей работе алгоритме применяются специальные генетические операторы, которые сохраняют целостность расписаний.

Для сохранения целостности при скрещивании гены должны исключаться из обеих родительских хромосом, чтобы не попасть в дочернюю хромосому повторно. В результате алгоритм гарантирует, что никакой ген при скрещивании не попадет в дочернюю хромосому дважды и все расписания в дочерней популяции останутся целостными. В численных экспериментах использовался одноточечный оператор скрещивания, так как алгоритм показал приблизительно равную производительность на исследуемых задачах при различном числе точек скрещивания. На рис. 1 оператор скрещивания назван регулируемым, так как вероятность скрещивания, как показали численные эксперименты (см. раздел 5 настоящей статьи), целесообразно менять в зависимости от числа задач в очереди.

Оператор мутации реализует обмен генов, что позволяет сохранить целостность. При однократной мутации два случайных гена в хромосоме меняются местами. При этом необходимо вносить изменения в составляющие времени генов в соответствии с формулой (3), так как после обмена гены могут быть сопоставлены процессорам с отличными от прежних характеристиками. Для увеличения вероятности равномерного распределения вычислительных процессов по процессорам при выборе первого и второго гена для обмена были использованы различные вероятности их выборов. В результате вычислительным процессам чаще сопоставляются менее загруженные вычислительной работой процессоры. Для определения вероятностей выбора генов представим хромосому в следующем виде. Подмножества генов хромосомы, относящиеся к одному процессору, выравниваются до максимального числа генов по всем процессорам путем вставки в конец данного подмножества дополнительных (заполняющих) генов. Такая форма хромосомы дает возможность изменения числа генов, относящихся к тому или иному процессору, в то время как при обычном обмене двух генов их число на том или ином процессоре не изменяется. Заполняющие гены кодируют нулевую задачу нулевой длительности, не нарушая структуру расписания. На рис. 2 показан пример подобной формы хромосомы, включающей для наглядности всего два процессора.

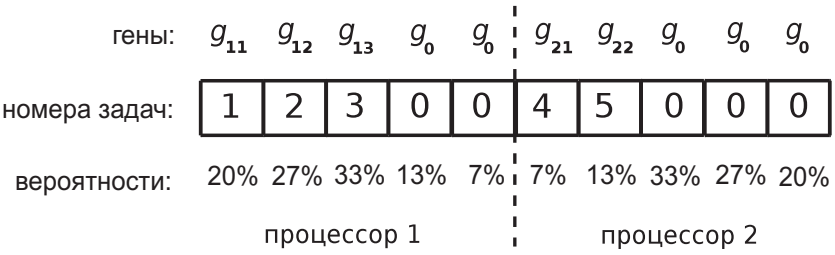

Рис. 2. Пример формы хромосомы с двумя процессорами для определения вероятностей выбора генов для мутации обменом

На данном рисунке гены g_0 — заполняющие. Для упрощения рассуждений предположим, что первый и второй гены будут выбраны из частей хромосомы, относящихся к процессорам 1 и 2 соответственно. При выборе генов для мутации обменом в данном случае необходимо оперировать двумя различными значениями вероятностей p_{m_1} и p_{m_2}, которые вычисляются для обычных и заполняющих генов по следующим формулам: $p_{m_1} = \dfrac{\delta_1 + m}{S_n}$, $p_{m_2} = \dfrac{\delta_2}{S_n}$, $p^0_{m_1} = \dfrac{m + m_0 - \delta_1 + 1}{S_n}$, $p^0_{m_2} = \dfrac{2m + m_0 - \delta_2 + 1}{S_n}$,

$S_n = \sum_{n=1}^{m+m_0} n = \frac{1}{2}(m + m_0 + 1)(m + m_0)$, где m и m_0 — число обычных и заполняющих генов, δ_1 и δ_2 — номера следования генов на процессоре, где произошел первый и второй выбор соответственно. В результате вычисления вероятностей по данным формулам для первого выбора гены получают возрастающую вероятность для обычных генов и убывающую для заполняющих, в то время как для второго выбора вероятности распределяются обратно первому.

Предлагаемый алгоритм является ресурсозависимым, так как учитываются потребности заданий в ресурсах по формуле (2). В программе данный учет производится при создании начальной популяции и на каждой итерации при применении оператора мутации. В операторе мутации гены, подвергшиеся изменению, проверяются на соответствие указанным выше ограничениям и в случае, если находятся гены, не прошедшие проверку, последняя операция над хромосомой повторяется. Аналогичное действие производится при создании хромосом начальной популяции, за исключением того, что проверяются все гены хромосом.

Основным критерием останова является отведенное время на решение задачи. Оно должно быть ограниченным или приемлемым для больших задач. В случае задач небольшой размерности критерием останова служат одинаковые значения длин расписаний для нескольких лучших решений в популяции при условии неизменности их на протяжении нескольких поколений. За отведенное время требуется получить расписание, иначе вся вычислительная система будет простаивать. ГА позволяет находить уже через несколько поколений приемлемое решение, которое может при продолжении работы программы становиться только лучше. В алгоритм включена проверка на группирование лучших решений по показателю длины расписания, эта проверка не является сложной в вычислительном отношении. Если имеет место группирование решений вокруг длины расписания, но отведенное время для решения задачи составления расписания не исчерпано, то программа не останавливается, а продолжает работу, что в некоторых случаях даст еще лучшее решение. Поэтому проверка на группирование лучших решений по показателю длины расписания имеет характер индикатора качества полученного решения.

Здесь рассматривается непараллельный ГА по следующей причине. Задача составления расписаний решается на управляющим узле, другие узлы должны предоставляться для решения заданий (или задач пользователей). Однако это не означает, что нецелесообразно разрабатывать параллельные варианты ГА решения поставленных задач, они могут использоваться в случае наличия свободных узлов на некотором интервале времени.

5. Результаты численных экспериментов. Описанный алгоритм реализован на языке программирования C++. Проведены численные эксперименты с целью проверки эффективности предложенного алгоритма и настройки его основных параметров: вероятности применения оператора мутации p_m, вероятности применения оператора скрещивания p_c, размера популяции M и числа элитных хромосом E.

Для формирования очередей использованы задачи из набора тестов NASA Advanced Supercomputing Parallel Benchmark [29] (NPB, v. 3.3). Тесты NPB состоят из ряда задач, являющихся фрагментами реальных приложений, требующих интенсивных параллельных вычислений. Каждый из тестов NPB может производить вычисления в нескольких классах сложности. В таблице представлены использованные нами в численных экспериментах задачи из NPB для классов сложности “W”, “A”, “B”, “C” и количества процессов 4 и 2 при применении технологии параллельных вычислений MPI.

В данной таблице указаны параметры, учитывающиеся алгоритмом составления расписаний: объем оперативной памяти в МБ и время счета в секундах. Данные параметры получены на вычислительном кластере, работающем в режиме с общей памятью, процессор Intel Core2 Quad 6600 2.4 ГГц, операционная система — Linux CentOS 4, библиотека MPI — Intel MPI Library.

Количество заданий в очереди может быть различным, поэтому программа составления расписаний должна надежно работать при любом их числе. Сначала изложим результаты проведенных экспериментов для небольшой задачи составления расписания с малым числом заданий.

Для исследования алгоритма при решении небольшой задачи составления расписаний выбраны следующие тесты из таблицы и их варианты для одного процесса: “lu” класса A с четырьмя, двумя и одним процессом (значение потребляемой памяти и длительность в секундах: 50 и 92 соответственно), один тест “lu” класса B с четырьмя процессами и тест “mg” класса A с одним процессом (значение потребляемой памяти и длительность в секундах: 450 и 5 соответственно). Для данной задачи получено точное решение методом перебора, которое используется для исследования работы созданной программы. Таким образом, задача состояла в нахождении расписания для 12 процессов. Рассматривалась РГВС из двух вычислительных кластеров, в общей сложности включающих пять процессоров. Первый кластер содержал два процессора Intel Pentium4 3.2 ГГц, и второй — три процессора того же типа, тактовая

Тесты NPB с их характеристиками “оперативная память / время решения”

	W		A		B		C	
Тест	4	2	4	2	4	2	4	2
bt	6/2	8/4	35/45	50/80	100/190	200/340	420/750	80/1375
cg	7/1	11/1	15/1	30/2	110/60	220/80	300/170	600/220
ep	3/1	3/2	3/8	3/13	3/27	3/55	3/105	3/210
ft	10/1	20/1	115/4	230/6	450/55	900/80	—	—
is	3/1	6/1	25/1	50/1	100/3	200/4	400/5	770/12
lu	6/5	8/8	17/34	35/56	50/175	100/327	190/1160	360/1400
mg	20/1	35/1	120/3	230/4	115/12	230/14	880/105	1700/115
sp	10/8	15/12	26/75	50/100	100/360	330/430	350/1376	700/1680

частота 2.2 ГГц. Время расчетов корректировалось в соответствии с производительностью процессоров по формуле (3).

Численное решение рассматриваемых задач нахождения расписаний с использованием ГА считалось найденным, если полученное значение длины расписания отличалось от точного решения не более чем на 3%. Получены зависимости средних значений времени T_p работы программы по результатам 100 запусков от значения уровня мутации p_m. Было установлено, что лучшего результата по времени алгоритм достигает при $p_m = 6\%$, остальные параметры: $p_c = 20\%$, $M = 50$, $E = 4$. Важным параметром является размер популяции M. Представлена зависимость T_p (рис. 3а) от M при $p_m = 6\%$ и $p_c = 4$. Из графика видно, что значение $M = 16$ является лучшим. Другим важным параметром является число элитных хромосом E в популяции. Экспериментально установлено, что число элитных хромосом для предложенного ГА при решении рассматриваемых задач должно быть небольшим: $E = 4$. Такой уровень элитизма характерен и для других вариантов ГА [30]. Экспериментально найдена и показана на рис. 3б зависимость T_p от уровня скрещивания p_c при $p_m = 6\%$. Наилучшие результаты получены при $p_c = 4$, эту величину обозначим p_c°.

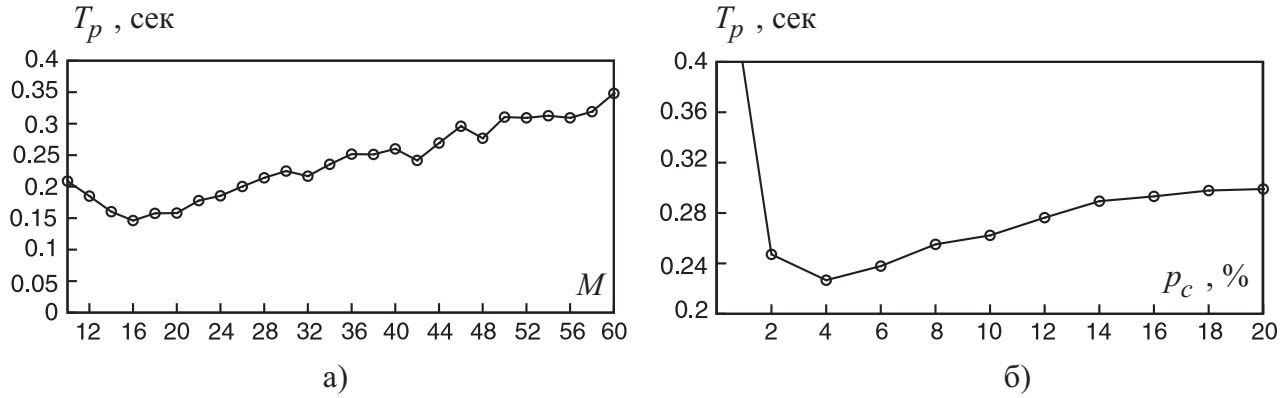

Рис. 3. Время решения задачи составления расписания при размерах популяции M (а) и при уровнях вероятности скрещивания p_c (б)

На рис. 4 представлены значения длин расписаний в зависимости от номера поколения I_p: для лучшего решения в популяции (l_1), второго по длине расписания (l_2), третьего (l_3), четвертого (l_4) и среднего значения для всех особей в популяции (l_c). При этом использовались параметры ГА, для которых время решения близко к минимальному: $p_m = 6\%, p_c = 4\%, M = 20, E = 4$. Значения длин l_2, l_3 и l_4 в 21 поколении достигают известного точного решения.

Следующая экспериментально исследованная задача состояла в нахождении расписания запуска всех тестов и классов сложности, представленных в таблице. Данная задача составления расписания может быть отнесена к задачам средней сложности. Точное решение для этой задачи методом перебора по-

лучить за приемлемое время невозможно, поэтому в дальнейшем принималось во внимание наилучшее решение, которое было получено в результате проведенных экспериментов. Численное решение нахождения расписания с использованием ГА считалось найденным, если оно отличалось от наилучшего решения не более чем на 3%.

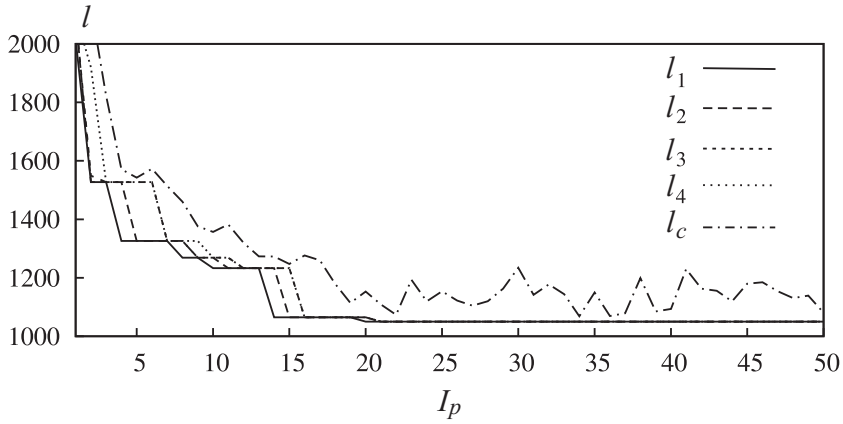

Рис. 4. Зависимости длин расписаний от номера поколения для небольшой задачи

Для описания вычислительных ресурсов в РГВС были использованы два вычислительных кластера ВЦ ДВО РАН. Первый вычислительный кластер [31, 32] — с четырьмя узлами: один процессор Intel Pentium4 3 ГГц и 1 Гб оперативной памяти. Второй вычислительный кластер состоит из восьми узлов, каждый из которых содержит по два двухъядерных процессора Intel Xeon 5060 3.2 ГГц и 4 Гб оперативной памяти. Результаты исследований производительности указанных вычислительных кластеров ВЦ ДВО РАН с использованием NPB даны в [32] и [33] соответственно. Результаты проверки эффективности алгоритма на задаче нахождения расписаний показали, что значения эффективных параметров ГА не зависят от размера задачи, кроме уровня скрещивания. Выявлена также близость результатов для двух использованных нами вариантов РГВС. Тестирование велось сериями из 30 запусков. По зависимости T_p от p_m установлено, что лучшего результата алгоритм достигает при значениях $p_m = 6\%$.

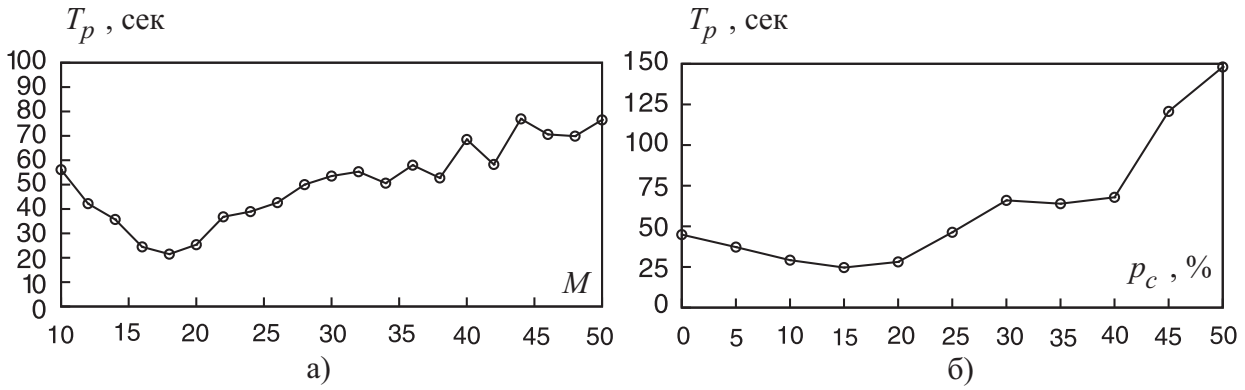

Рис. 5. Время решения задачи составления расписания при различных размерах популяции M (а) и при различных уровнях вероятности скрещивания p_c (б) для средней задачи

На рис. 5а представлена зависимость T_p от размера популяции M. Можно видеть, что лучшего значения по времени программа достигает при размере популяции $M = 18$. При исследовании параметра доли элитизма экспериментально установлено, что лучшего значения по времени программа достигает при $E = 4$. На рис. 5б представлена зависимость T_p от p_c, в этом случае скорость работы программы максимальна при $p_c^\circ = 15\%$. Это значение больше примерно в 4 раза, чем полученное для малой задачи с более простой РГВС, поэтому были дополнительно проведены численные эксперименты для шести других очередей задач из таблицы, но меньших размеров. Для минимальной очереди из 48 задач получено $p_c^\circ = 4\%$. Меньшее число задач для рассматриваемой РГВС с общим числом процессорных ядер, равным 36, не было смысла исследовать. В результате экспериментов установлено, что полученную зависимость p_c° от числа заданий в очереди можно аппроксимировать линейной функцией. Из рис. 5б видно, что времена решений при $p_c^\circ = 4\%$ и $p_c^\circ = 15\%$ отличаются примерно в 1.5 раза, поэтому целесообразно поставить в зависимость значение параметра p_c от величины очереди задач.

На рис. 6 представлены результаты сходимости алгоритма, аналогичные показанным на рис. 4, с параметрами, при которых достигается минимальное время решения: $p_m = 6\%$, $p_c = 15\%$, $M = 20$, $E = 4$. Средняя длина расписаний в популяции (l_c) с ростом номера поколения останавливается на некотором уровне благодаря относительно высоким значениям p_c и p_m.

Детальный анализ динамики решений (особей) от номера поколений показал следующее. Для средней задачи находится много субоптимальных решений вблизи глобального минимума длины расписания. Некоторые решения с равными значениями длины расписания отличаются перестановками порядка следования заданий в расписании, однако эти решения в данном случае являются равноценными. Поэтому группирование решений возле конкретных субоптимальных решений не может служить критерием сходимости к глобальному минимуму, который в некоторых случаях также может состоять из нескольких расписаний, отличающихся перестановкой и имеющих одинаковую длину расписания. В этом смысле одному глобальному минимуму соответствует несколько одинаково приемлемых решений. Критерием сходимости может являться группирование решений вокруг длины расписания. На графиках (рис. 4 и 6) динамики длины расписания показывается процесс сближения и объединения по показателю длины расписания четырех лучших решений.

Из полученных результатов численных экспериментов для задач различной сложности следует, что параметры ГА, при которых программа находит приемлемые решения за наименьшее время, примерно совпадают, кроме значений вероятности скрещивания p_c°.

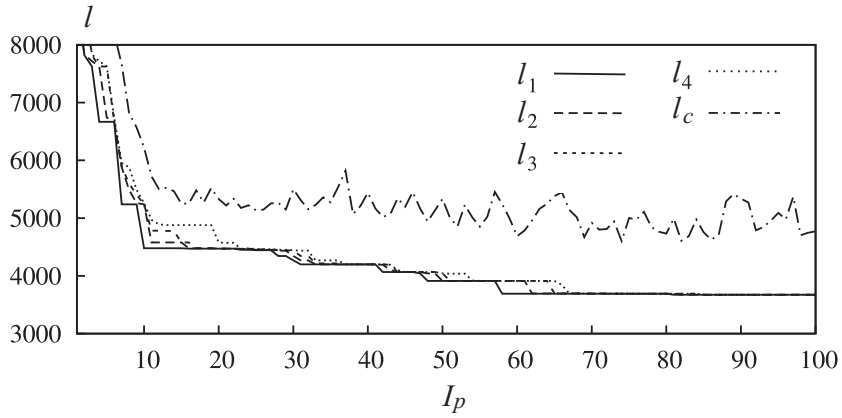

Рис. 6. Длины расписаний в процессе сходимости для задачи среднего размера

6. Заключение. Для решения задачи составления расписания в РГВС предложен и программно реализован ГА, учитывающий особенности предметной области. Проведены экспериментальные исследования ГА для двух вариантов конфигурации вычислительных ресурсов и различных размеров очереди заданий. Выявлены параметры ГА, при которых программа работает с максимальной скоростью при условии надежного достижения точного решения для небольшой задачи и субоптимальных решений, которые отличаются от полученного наилучшего значения не более чем на 3%, для задачи средней сложности. Установлено, что вероятность скрещивания в ГА целесообразно менять в зависимости от размера очереди заданий, что может дать прирост производительности около 50% для исследованного нами диапазона размера очереди заданий и более, если этот диапазон будет шире.

СПИСОК ЛИТЕРАТУРЫ

1. *Коффман Э.Г.* Теория расписаний и вычислительные машины. М.: Наука, 1984.
2. Энциклопедия кибернетики / Отв. ред. В. М. Глушков. Т. 2. Киев: Главная редакция УСЭ, 1974.
3. *Land A.H., Doig A.G.* An autmatic method of solving discrete programming problems // Econometrica. 1960. N 28. 497–520.
4. *Lawler E.L., Wood D.E.* Branch and bound methods: a survey // Operations Research. 1966. N 14(4). 699–719.
5. *Johnson T.J.R.* An alorithm for the resource-constraned project scheduling problem. Doctoral Thesis. Massachusetts Institute of Technology. Cambridge, 1967.
6. *Müller-Merbach H.* Ein Verfahren zur Planung des optimalen Betriebsmitteleinsatzes bei der Terminierung von Großprojeckten // Zeitschrift für Wirtschaftliche Fertigung. 1967. N 62. 83–88.
7. *Panwalker S., Iskander W.* A survey of scheduling rules // Operations Research. 1977. N 25(1). 45–61.
8. *Davis E.W., Patterson J.H.* A comparison of heuristic and optimum solutions in resource-constrained project scheduling // Management Science. 1975. N 21(8). 944–955.
9. *Davis E.W., Heidorn G.E.* An algorithm for optimal project scheduling under multiple resource constraints // Management Science. 1971. N 17(12). 803–817.
10. *Hildum D.* Flexibility in a knowledge-based system for solving dynamic resource-constrained scheduling problems. Umass CMPSCI Technical Report N 94. University of Massachusetts. Amherst, 1994.
11. *Boctor F.F.* Some efficient multi-heuristic procedures for resource-constrained project scheduling // European Journal of Operational Research. 1990. N 49(1). 3–13.
12. *Harvey W.D., Ginsberg M.L.* Limited discrepancy search // Proc. of the 14th International Joint Conference on Artificial Intelligence. Montreal: Morgan Kaufmann, 1995. 607–615.
13. *Sampson S.E., Weiss E.N.* Local search techniques for the generalized resource-constrained project scheduling problem // Naval Research Logistics. 1993. N 40(5). 665–675.

14. *Patterson J.H.* A comparison of exact approaches for solving the multiple constrained resource project scheduling problem // Management Science. 1984. N 30(7). 854–867.
15. *Fox M.S., Smith S.F.* ISIS — a knowledge-based system for factory scheduling // Expert Systems. 1984. N 1(1). 25–49.
16. *Smith S.F., Ow P.* The use of multiple problem decompositions in time constrained planning tasks // Proc. of the 9th International Joint Conference on Artificial Intelligence. Vol. 2. Calif.: Morgan Kaufmann, 1985. 1013–1015.
17. *Sadeh N.* Look-ahead techniques for micro-opportunistic job shop scheduling. PhD Thesis. School of Computer Science, Carnegie Mellon University. Pittsburgh, 1991.
18. *Davis L.* Job shop scheduling with genetic algorithms // Proc. of the International Conference on Genetic Algorithms and their Applications. Pittsburgh: Lawrence Erlbaum Associates, 1985. 136–140.
19. *Bagchi S., Uckum S., et al.* Exploring problem-specific recombination operators for job shop scheduling // Proc. of the 4th International Conference on Genetic Algorithms. San Mateo: Morgan Kaufmann, 1991. 10–17.
20. *Syswerda G.* The application of genetic algorithms to resource scheduling // Proc. of the 4th International Conference on Genetic Algorithms. San Mateo: Morgan Kaufmann, 1990. 502–508.
21. *Hilliard M.R., Liepins G.E., et al.* Machine learning applications to job shop scheduling // Proc. of the AAAI-SIGMAN Workshop on Production Planning and Scheduling. New York: ACM, 1988. 728–737.
22. *Pugliese F., Talia D., Yahyapour R.* Modeling and supporting grid scheduling // Journal of Grid Computing. 2008. N 6/2. 195–213.
23. *Doğan A., Özgüner F.* Genetic algorithm based scheduling of meta-tasks with stochastic execution times in heterogeneous computing systems // Cluster Computing. 2004. N 7. 177–190.
24. *Buyya R.* Economic-based distributed resource management and scheduling for grid computing. Ph. D. Thesis. School of Computer Science and Software Engineering. Monash University. Melbourne, 2002.
25. *Derbal Y.* Entropic grid scheduling // Journal of Grid Computing. 2006. N 4(4). 373–394.
26. *Junwei C., Spooner D., Turner J.D., Jarvis S., Kerbyson D.J., Saini S., Nudd G.* An agent-based resource management system for grid computing // Proc. of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid. Washington: IEEE Computer Society, 2002. 350–350.
27. *Braun T.D., Siegel H.J., Beck N., Boloni L.L., Maheswaran M., Reuther A.I., Robertson J.P., Theys M.D., Yao B., Hensgen D., Freund R.F.* A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems // Journal of Parallel and Distributed Computing. 2001. N 61(6). 810–837.
28. *Топорков В.В.* Модели распределенных вычислений. М.: Физматлит, 2004.
29. *Bailey D., Barszcz E., Barton J., et al.* The NAS parallel benchmarks. Technical Report N RNR–94–007. Washington, 1994.
30. *Hart W.E.* Adaptive global optimization with local search. Ph. D. Thesis. University of California. San Diego, 1994.
31. *Пересветов В.В., Сапронов А.Ю., Тарасов А.Г., Шаповалов Т.С.* Удаленный доступ к вычислительному кластеру ВЦ ДВО РАН // Вычислительные технологии. 2006. **11**. 2006. 45–51.
32. *Пересветов В.В., Сапронов А.Ю., Тарасов А.Г.* Вычислительный кластер бездисковых рабочих станций. Препринт № 83 ВЦ ДВО РАН. Хабаровск, 2005.
33. *Щерба С.И., Пересветов В.В.* Сравнительный анализ эффективности программного обеспечения для вычислительных кластеров // Тр. Межрегиональной научно-практической конференции “Информационные и коммуникационные технологии в образовании и научной деятельности” (г. Хабаровск, 21–23 мая 2008). Хабаровск: Изд-во Тихоокеанского гос. ун-та, 2008. 363–369.

Поступила в редакцию
17.11.2008