УДК 004.451.42:.44:004.023

А.В. Ефимов, С.Н. Мамойленко, Е.Н. Перышкова

ОРГАНИЗАЦИЯ ФУНКЦИОНИРОВАНИЯ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ ПРИ ОБРАБОТКЕ НАБОРОВ МАСШТАБИРУЕМЫХ ЗАДАЧ[1]

Предложены последовательный и параллельный генетические алгоритмы оптимизации функционирования распределенных вычислительных систем (ВС) при обработке наборов масштабируемых задач. Представлены результаты моделирования и оценена эффективность алгоритмов.

Ключевые слова: *распределённые вычислительные системы*, *оптимизация функционирования*, *масштабируемые задачи*, *расписание решения задач.*

Распределённые вычислительные системы (ВС) относятся к перспективным средствами обработки информации [1]. Основным функциональным элементом распределённой ВС является элементарная машина (ЭМ). Структура ЭМ допускает варьирование от процессорного ядра до конфигураций, включающих многоядерные процессоры и специализированные ускорители (например, GPGPU). Количество ЭМ в распределённых ВС может варьироваться в широких пределах: от десятков до сотен тысяч.

Повышение эффективности эксплуатации ресурсов распределённых ВС связано с использованием технологии параллельного мультипрограммирования [1, 2]. Одной из актуальных проблем организации функционирования ВС в мультипрограммных режимах является планирование решения пользовательских задач. Здесь, по аналогии с [3], под задачей понимается требование пользователя на выполнение параллельной программы. В общем случае проблема планирования решения задач является трудноразрешимой [4]. Перспективной считается разработка приближённых методов и эвристических алгоритмов управления ресурсами ВС.

Повысить эффективность эксплуатации ресурсов ВС и снизить время нахождения задач в очереди возможно, если, в частности, каждая из них будет допускать решение на различных конфигурациях вычислительных ресурсов. В этом случае задачи называют *масштабируемыми*, или пластичными (moldable). Исследования пользовательских запросов показывают, что свойством масштабируемости обладают более 80 % задач [5].

Системы пакетной обработки, используемые на практике, допускают наличие в очередях масштабируемых задач и позволяют пользователям указывать количество требуемых для их решения ЭМ (рангов задач) в виде диапазонов допустимых значений. В данной работе предполагается, что пользователем каждому допустимому значению ранга задачи задаётся требование на время её решения и указывается предпочтение по выбору именно этих значений. Кроме этого, для каждой за-

[1] Работа выполнена в рамках интеграционного проекта № 113 СО РАН, при поддержке РФФИ (гранты № 08-07-00022, 10-07-00157, 09-07-90403, 09-07-00095, 08-08-00300), Совета по грантам Президента РФ для поддержки ведущих научных школ (грант НШ-5176.2010.9) и в рамках государственного контракта № 02.740.11.0006 с Минобрнауки РФ.

дачи определяется штраф за задержку её решения. Величина штрафа рассчитывается исходя из характеристик задачи: времени поступления, класса пользователя, и т.п.

В основу предложенного алгоритма положен метод [3] формирования укрупнённых задач (пакетов). Планирование выполняется в два этапа: задачи набора распределяются по укрупнённым задачам с целью минимизации общего времени их решения и определяется последовательность решения укрупнённых задач, минимизирующая суммарный штраф за задержку решения задач набора.

1. Постановка задачи

Имеются распределённая ВС, состоящая из N элементарных машин, и набор $I=\{I_i\}$ независимых задач, $i=\overline{1,L}$. Каждая задача характеризуется вектором $P_i=\left\{p_i^k\right\}$ и величиной штрафа c_i за задержку её решения в единицу времени, $c_i>0$. Элементы вектора $p_i^k=(r_i^k,t_i^k,w_i^k)$ задачи I_i задают допустимые значения ранга r_i и времени решения t_i и определяют приоритет w_i^k выбора этих значений, $0<r_i^k\le N$, $t_i^k>0$, $w_i^k>0$, $k=\overline{1,q_i}$, $q_i=|P_i|$. «Удовлетворённость» пользователя выбором для решения задачи значений с приоритетом w_i^k оценивается как $w_i^k\Big/\max\limits_k w_i^k$. Допускается существование в векторе P_i нескольких элементов с одинаковым приоритетом. Считается, что в наборе присутствуют задачи с различным количеством q_i допустимых значений параметров и возможно взаимодействие ЭМ с любой другой машиной ВС; зависимости величин r_i^k, t_i^k, w_i^k, c_i друг от друга отсутствуют.

Необходимо для каждой задачи I_i набора выбрать: k_i – номер элемента вектора P_i, и s_i – время начала решения задачи, а также выделить подсистему $J_i=\{j\in E_1^N\}$ элементарных машин ВС, где $0<k_i\le q_i$, $s_i\ge 0$, j – номер ЭМ, $E_1^N=\{1,2,...,N\}$, $|J_i|=r_i^{k_i}$. В результате должен быть сформирован вектор $S=\langle(k_i,s_i,J_i)\rangle$, $i=\overline{1,L}$, называемый расписанием решения задач на ВС. Характеристиками расписания являются: время $T(S)$ решения всех задач набора и суммарный штраф $F(S)$ за задержку их решения. Кроме этого, качество сформированного расписания оценивается степенью загрузки ресурсов ВС при решении задач.

Требуется найти расписание S^* решения задач на вычислительной системе, такое, чтобы

$$T(S^*)=\min_{S\in\Omega}T(S),\ T(S)=\max_{i=\overline{1,L}}\{s_i+t_i^{k_i}\};\qquad(1)$$

$$F(S^*)=\min_{S\in\Omega}F(S),\ F(S)=\sum_{i=1}^{L}s_ic_i,\qquad(2)$$

при ограничениях

$$\forall i \in \{1,2,...,L\}, |J_i| = r_i^k u \forall j_1 \in J_i, \forall j_2 \in J_i \setminus \{j_1\} \Rightarrow j_1 \neq j_2; \quad (3)$$

$$\forall t \geq 0, \bigcap_{i \in \Xi(t)} J_i = 0, \sum_{i \in \Xi(t)} r_i^{k_i} \leq N; \quad (4)$$

$$L^{-1} \sum_{i=1}^{L} \frac{w_i^{k_i}}{\max\limits_k w_i^k} \geq e, \quad (5)$$

где Ω – область допустимых расписаний S, $\Xi(t) = \{i \in \{1,2,...,N\} \mid s_i \leq t \leq s_i + t_i^{k_i}\}$ – множество номеров задач, решаемых в момент времени t; e – минимально допустимая средняя «удовлетворённость» пользователей. Ограничение (3) определяет, что каждой задаче с учётом выбранных значений параметров должно быть выделено столько ЭМ, сколько требуется для её решения. Ограничение (4) гарантирует, что в каждый момент времени задачи решаются на непересекающихся подсистемах ЭМ и их суммарный ранг не превосходит количества элементарных машин ВС.

Задача (1) – (5) относится к многокритериальной оптимизации [6]. Для решения задачи используем метод приоритетов [6], задавая функции (1) приоритет выше, чем функции (2).

2. Формирование укрупнённых задач

На первом этапе планирования задачи набора разбиваются на подмножества [3] таким образом, чтобы выполнялись ограничения (3) – (5) и достигался минимум функции (1). Подмножества задач будем называть укрупнёнными задачами.

Формально постановка задачи первого этапа приобретает следующий вид. Решением является тройка $S_1 = (K, M, \Im)$, где $K = \langle k_i \rangle$ – вектор выбранных для задач значений k_i, $\Im = \{\Im_m\}$ – множество укрупнённых задач $\Im_m = \{I_i \in I\}$, $m = \overline{1, M}$, $M = |\Im|$ – количество укрупнённых задач. Требуется определить решение S_1^*, такое, чтобы

$$T(S_1^*) = \min_{S_1 \in \Omega_1} T(S_1), \; T(S_1) = \sum_{m=1}^{M} T_m, \; T_m = \max_{i \in \Phi(m)} t_i^{k_i}; \quad (6)$$

при ограничениях

$$\bigcup_{m=1}^{M} \Im_m = \Im, \; \bigcap_{m=1}^{M} \Im_m = 0, \; \sum_{m=1}^{M} |\Im_m| = L; \quad (7)$$

$$R_m \leq N, \; R_m = \sum_{i \in \Phi(m)} r_i^{k_i}, \quad (8)$$

где Ω_1 – область допустимых решений S_1, R_m – суммарный ранг задач каждого подмножества $\Im_m$, $\Phi(m) = \{i \in \{1,2,...,L\} \mid I_i \in \Im_m\}$ – множество номеров задач, помещённых в m-е подмножество. Ограничение (7) требует, чтобы все задачи набора были распределены по укрупнённым задачам. Ограничение (8) задает требо-

вание, чтобы суммарный ранг укрупнённых задач не превосходил количество ЭМ вычислительной системы.

Задача (6) – (8) относится к задачам ортогональной упаковки объектов в контейнеры [7]. При этом задача набора кодируется прямоугольником с шириной, равной r_i^k и высотой t_i^k. Повороты и пересечения прямоугольников не допускаются. Известно, что эта задача является NP-сложной [8].

2.1. Выбор значений параметров задач набора

При выборе для задач значений k_j необходимо контролировать выполнение ограничения (5). Представим набор задач I в следующем виде: $I^0 \bigcup I^1 \bigcup I^2$, где I^0 – подмножество задач набора с одним элементом вектора P_i; I^1 – подмножество задач набора, у которых все элементы вектора P_i имеют одинаковый приоритет; I^2 – остальные задачи набора; $I^0 \bigcap I^1 \bigcap I^2 = \varnothing$. Тогда ограничение (5) можно представить в следующем виде:

$$L^{-1}\Big(\sum_{i\in\Phi^0}\frac{w_i^{k_i}}{\max\limits_k w_i^k}+\sum_{i\in\Phi^1}\frac{w_i^{k_i}}{\max\limits_k w_i^k}+\sum_{i\in\Phi^2}\frac{w_i^{k_i}}{\max\limits_k w_i^k}\Big)\ge e\,, \qquad (9)$$

где $\Phi^0 = \{i \in \{1,2,...,L\} \mid I_i \in I^0\}$, $\Phi^1 = \{i \in \{1,2,...,L\} \mid I_i \in I^1\}$

и $\Phi^2 = \{i \in \{1,2,...,L\} \mid I_i \in I^2\}$.

Очевидно, что для задач из подмножества I^0 выбирать значения параметров нет необходимости и первое слагаемое выражения (9) всегда равно $|I^0|$. Второе слагаемое выражения (9) также равно $|I^1|$ при любом выборе значений параметров для задач из этого подмножества. На выполнение ограничения (5) влияет лишь выбор значений параметров для задач из подмножества I^2.

Утверждение 1. *Если* $|I^0| + |I^1| \ge eL$, *то ограничение* (5) *выполняется при любом выборе значений параметров для задач подмножества* I^2. Доказательство очевидно.

Если для набора задач утверждение 1 не справедливо, то используем следующую процедуру выбора значений параметров задач.

Шаг 0. Считаем, что для задач были выбраны значения параметров, имеющие максимальный приоритет (в случае нескольких значений с максимальным приоритетом выбираем значение, запрашивающее меньшее время).

Шаг 1. Для задач подмножества I^1 значения k_i выбираются произвольно.

Шаг 2. Вычисляются значения средней «удовлетворённости» пользователей при решении задач набора согласно текущему выбору параметров.

Шаг 3. Произвольно выбирается задача из подмножества I^2 и новое значение k_i для неё.

Шаг 4. Если изменение приоритета значений параметров задачи не приводит к нарушению условия (5), то фиксируется k_i и осуществляется переход к шагу 2; в противном случае значение k_i для выбранной задачи не меняется.

Шаг 5. Повторяем шаги 2 – 4 до тех пор, пока не будет перебраны все задачи подмножества I^2.

2.2. Генетический алгоритм формирования укрупнённых задач

Генетический алгоритм предусматривает последовательность жизненных циклов популяции. Каждый цикл состоит из следующих операций: выбора пар особей, их размножения, мутации и селекции наиболее приспособленных особей (формирования новой популяции). Процесс повторяется до тех пор, пока на протяжении заданного количества популяций не будет появляться особь с лучшим значением функции приспособленности. Итоговое разбиение задач набора формируется из особи, имеющей наилучшее значение функции приспособленности.

В терминах генетических алгоритмов [9] будем считать:

ген – укрупнённая задача (пакет);

особь или *хромосома* – допустимое разбиение задач набора при фиксированных значениях k_i;

популяция – несколько особей с различными значениями k_i;

функция приспособленности особи – значение функции (6) для соответствующего разбиения задач набора.

Размер популяции задаётся параметром алгоритма и остаётся постоянным в процессе всей эволюции.

2.2.1. Последовательная версия алгоритма

Начальная популяция формируется следующим образом. Первая особь выбирает для каждой задачи k_i, такое, при котором значения параметров имеют наивысший приоритет w_i^k. Вторая особь – минимум запрашиваемых ресурсов $r_i t_i$. Третья – максимум «удельной площади» $w_i^k / r_i^k t_i^k$. Остальные особи выбирают k_i случайным образом согласно процедуре, описанной в п.2.1. Далее каждая особь формирует укрупнённые задачи (гены), используя алгоритм FFDH [10].

На каждом жизненном цикле из популяции выбираются пары особей. Количество пар задается как половина от количества особей в популяции. Для формирования пары из популяции случайным образом выбирается одна особь. Парой для неё выбирается особь, наиболее удалённая от неё по значению функции приспособленности и не состоящая в других парах. Далее над каждой выбранной парой с заданной вероятностью либо выполняется оператор кроссинговера, либо производится мутация родительских особей.

В качестве оператора кроссинговера предлагается алгоритм, в основу которого положен метод перетасовки генов [11]:

Шаг 1. Гены родительских особей помещаются в общий список и сортируются по степени загрузки ($(R_m T_m)^{-1} \sum_{i \in \Phi(m)} r_i^k t_i^k$).

Шаг 2. Из полученной последовательности выбираются гены, множества задач в которых не пересекаются. Выбранные гены помещаются в новую особь. Оставшиеся гены расформировываются.

Шаг 3. Для расформированных задач случайным образом выбирается один из параметров. Далее эти задачи распределяются по генам с использованием алгоритма FFDH (с учетом уже имеющихся генов).

Мутация выполняется над одной или двумя родительскими особями. Оператор мутации с равной вероятностью либо случайным образом изменяет значения параметров задач, гарантируя выполнение ограничения (5), либо выбирает, для заданной доли задач особи, значения параметров, имеющие максимальный приоритет.

В результате от каждой пары создаются одна или две изменённые особи «потомков», которые могут не выжить, если для них по каким-либо причинам не выполняются ограничения (3) – (5). В новую популяцию попадают родительские и «выжившие потомки», имеющие наилучшие показатели функции приспособленности.

2.2.2. Параллельная версия алгоритма

Очевидно, что время работы последовательной версии генетического алгоритма зависит от количества задач в наборе. Для больших наборов множество задач можно разбить на части, для каждой из которых независимо выполнить последовательную версию генетического алгоритма. Термин «большой набор» следует определять экспериментальным путём для конкретной реализации последовательной версии и аппаратурных возможностей вычислителей, на которых этот алгоритм будет выполняться.

3. Определение последовательности решения укрупнённых задач

На втором этапе определяется последовательность решения укрупнённых задач (пакетов), позволяющая получить минимум функции (2). Состав укрупнённых задач не изменяется.

Пусть $S_2 = m_1,...,m_l,...,m_M$ – некоторая последовательность решения пакетов, где m_l – номер пакета, который будет решаться в *l*-ю очередь. Штраф за решение задач при выбранной последовательности решения пакетов определяется как

$$F(S_2) = \sum_{l=2}^{M} C_{ml} \sum_{r=1}^{l-1} T_{m_r} , \qquad (10)$$

где $C_{ml} = \sum_{i \in \Phi(m)} c_i$ – штраф за задержку решения задач, входящих в *m*-й пакет.

Требуется найти такую последовательность S_2^*, чтобы достичь минимума функции (10).

Запишем функцию (10) в следующем виде:

$$F(S_2) = C * T - f(S_2) ,$$

где

$$C = \sum_{l=1}^{M} C_{ml} , \; T = \sum_{l=1}^{M} T_{ml} ,$$

$$f(S_2) = \sum_{l=1}^{M} T_{ml} \sum_{r=1}^{l} C_{m_r} . \qquad (11)$$

Очевидно, что минимум (10) достигается при максимуме (11).

Известно **утверждение 2** [5, 12]. *Для того чтобы последовательность решения пакетов задач обеспечивала минимум функции* (*10*), *необходимо и достаточно*, *чтобы выполнялись неравенства*

$$\frac{T_{m_1}}{C_{m_1}} \le \frac{T_{m_2}}{C_{m_2}} \le \ldots \le \frac{T_{m_l}}{C_{m_l}} \le \ldots \le \frac{T_{m_M}}{C_{m_M}}. \qquad (12)$$

4. Формирование итогового расписания решения задач

Итоговое расписание $S^* = \langle (k_i, s_i, J_i) \rangle$ формируется исходя из S_1^* и S_2^* следующим образом. Для k_i используются значения из S_1^*. Время начала решения s_i определяется временем начала решения укрупнённой задачи s_r^*, в которую она была помещена. Время s_r^* определяется найденной последовательностью решения укрупнённых задач: $s_r^* = \sum_{l=1}^{r-1} T_{m_l}, s_1^* = 0$. Итак, $\forall m \in \{1, 2, ..., M\}$, $\forall i \in \Phi(m), s_i = s_m^*$. Машины ВС распределяются между задачами каждой укрупнённой задачи в соответствии с требованиями.

5. Результаты моделирования работы алгоритма формирования расписаний решения задач

Моделирование и численные эксперименты осуществлялись с использованием ресурсов пространственно-распределённой мультикластерной вычислительной системы Центра параллельных вычислительных технология ГОУ ВПО «Сибирский государственный университет телекоммуникаций и информатики» [13]. Предложенные алгоритмы реализованы с использованием языка программирования C++. Компиляция программ осуществлялась GNU GCC с указанием максимально возможной степени оптимизации (–03). Наборы задач генерировались для систем с количеством ЭМ от 2^1 до 2^{20} на основе модели рабочей загрузки, предложенной в работе [4]. Рассматривались наборы с количеством задач 10, 100, 1000 и 10000. Приоритеты значениям параметров задач задавались путём их простой нумерации.

ГА требует задания трёх параметров: количества жизненных циклов популяций без улучшения потомства (обозначим как $STEPS_{GA}$), размера популяции (обозначим как PS_{GA}), вероятности выполнения кроссинговера и мутации (обозначим как P_{GA}). Кроме этого, алгоритм зависит от минимально-допустимой средней удовлетворённости пользователей (обозначим как *e*).

Моделирование показало следующее. Для любой минимально допустимой средней «удовлетворённости» пользователей *e* лучшим значением для $STEPS_{GA}$ – 5. При дальнейшем увеличении значении $STEPS_{GA}$ время работы алгоритма растёт, а качество получаемого решения практически не изменяется (рис. 1). $T_{FFD}(S)$ – это время решения задач по расписанию, для формирования которого задачам назначены значения параметров с максимальным приоритетом и однократно выполнен алгоритм FFDH.

С увеличением значения параметра *e* уменьшается время работы алгоритма. Это связано с тем, что количество вариантов значений параметров с увеличением *e* сокращается (рис. 2). Наилучшим размером популяции PS_{GA} является 32 особи

для любого значения минимально допустимой средней «удовлетворённости» пользователей (рис. 3). Вероятность выполнения операторов кроссинговера и мутации существенно не влияет на качество получаемого решения (рис. 4).

Выполнение параллельной версии алгоритма целесообразно для наборов, с количеством задач больше 1000. Ускорение алгоритма практически линейно. Ухудшение значения функции $T(S)$ составило не более 10 %.

Моделирование второго этапа подтвердило на практике необходимость и достаточность (12) для обеспечения минимума функции штрафа за задержку решения задач набора (10).

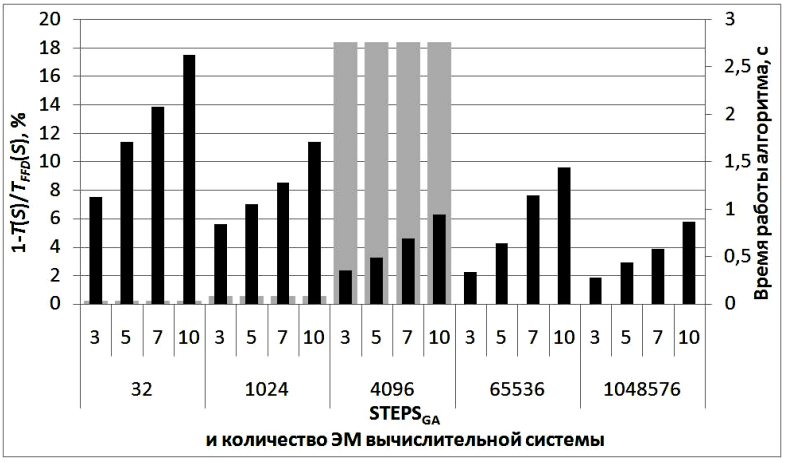

Рис. 1. Влияние параметра $STEPS_{GA}$ на качество работы последовательного ГА ($L = 1000$, $PS_{GA} = 32$, $P_{GA} = 0{,}8$, $e = 0{,}95$): ■ – улучшение значения функции $T(S)$, %; ■ – время работы алгоритма, с

Рис. 2. Влияние e на время работы последовательного ГА ($STEPS_{GA} = 5$, $PS_{GA} = 32$, $P_{GA} = 0{,}8$): – $e = 0{,}5$; – $e = 0{,}7$; – $e = 0{,}8$; – $e = 0{,}9$; – $e = 1$

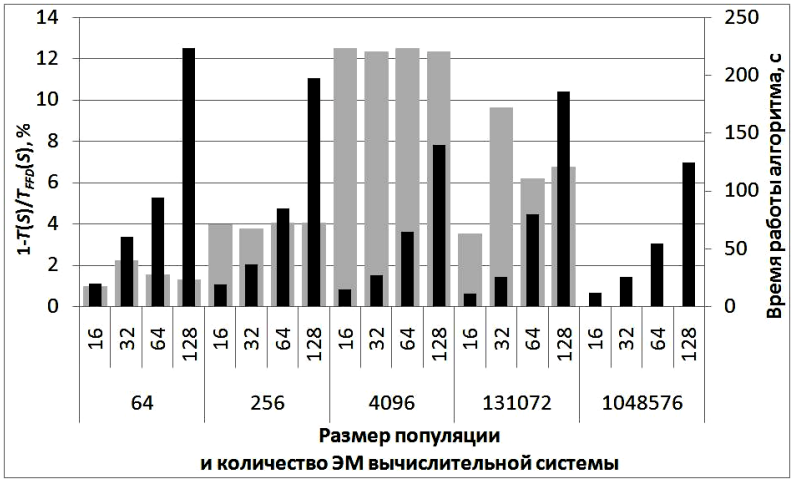

Рис. 3. Влияние размера популяции PS_{GA} на качество работы ГА (L = 10000, $STEPS_{GA}$ = 5, P_{GA} = 0,8, e = 0,95): ■ – улучшение значения функции $T(S)$, %; ■ – время работы алгоритма, с

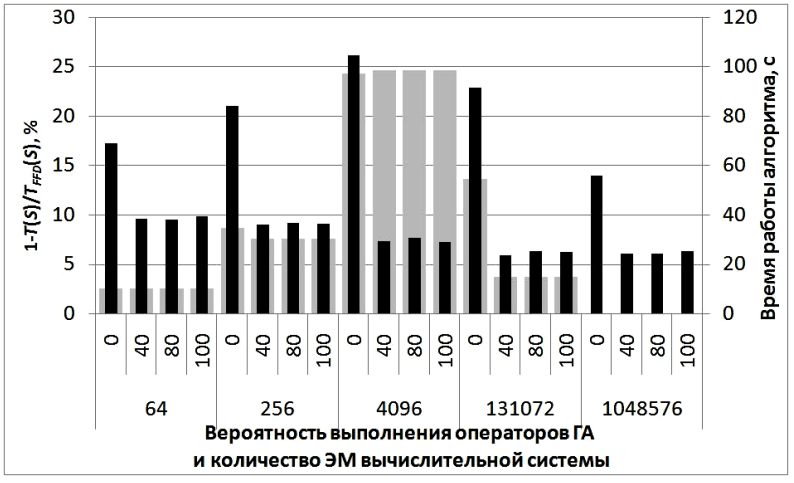

Рис. 4. Влияние вероятности выполнения операторов кроссинговера и мутации на качество работы ГА (L = 10000, PS_{GA} = 32, e = 0,95): ■ – улучшение значения функции $T(S)$, %; ■ – время работы алгоритма, с

Заключение

Разработанные алгоритмы для планирования решения масштабируемых задач на ВС с учётом штрафов за задержку их решения и приоритетов выбора возможных конфигураций подсистем лягут в основу планировщика ресурсов пространственно-распределённой мультикластерной вычислительной системы Центра параллельных вычислительных технологий ГОУ ВПО «СибГУТИ».

ЛИТЕРАТУРА

1. *Хорошевский В.Г.* Архитектура вычислительных систем. М.: МГТУ им. Н.Э. Баумана, 2008. 520 с.
2. *Хорошевский В.Г.* Модели функционирования большемасштабных распределенных вычислительных систем // Электросвязь. М: Электросвязь, 2004. № 10. С. 30–34.
3. *Евреинов Э.В.*, *Хорошевский В.Г.* Однородные вычислительные системы. Новосибирск: Наука, 1978. 319 с.
4. *Бруно Дж.Л. и др.* Теория расписаний и вычислительные машины. М.: Наука, 1984. 336 с.
5. *Cirne W. and Berman F.* A model for moldable supercomputer jobs // 15th Intl. Parallel & Distributed Processing Symp. 2001. URL: http://www.lsd.dsc.ufpb.br/papers/moldability-model.pdf (дата обращения: 12.04.2010).
6. *Таха Х.* Введение в исследование операций. 6-е изд. М.: Вильямс, 2001. 911 с.
7. *Коффман Э.Г.* Теория расписаний и вычислительные машины. М.: Наука, 1984. 336 с.
8. *Гэри М.*, *Джонсон Д.* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. 416 с.
9. *Гладков Л.А.*, *Курейчик В.В.*, *Курейчик В.М.* Генетические алгоритмы. М.: Физматлит, 2006. 320 с.
10. *Coffman E.G.* et al. Performance bounds for level-oriented two-dimensional packing algorithms // SIAM Journal on Computing. 1980. V. 9. P. 808–826.
11. *Rohlfshagen P.*, *Bullinaria J.A.* A genetic algorithm with exon shuffling crossover for hard bin packing problems // Proc. of the 9th Annual Conf. on Genetic and Evolutionary Computation. ACM NewYork, 2007. P. 1365–1371.
12. *Smith W.* Various optimizers for single-stage production. Naval res. Logist. Quart. 3. 1956. P. 59–66.
13. Ресурсы Центра параллельных вычислительных технологий ГОУ ВПО «СибГУТИ». URL: http://cpct.sibsutis.ru (дата обращения: 10.11.2010).

Ефимов Александр Владимирович
Мамойленко Сергей Николаевич
Перышкова Евгения Николаевна
ГОУ ВПО «Сибирский государственный университет телекоммуникаций и информатики» (г.Новосибирск)
E-mail: efimov@cpct.sibsutis.ru; sergey@cpct.sibsutis.ru; e_perishkova@cpct.sibsutis.ru

Поступила в редакцию 3 декабря 2010 г.