MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 27, NO. 2, 2020
journal homepage: www.mais-journal.ru

ALGORITHMS

Research and Development of an Algorithm for the Response Time Estimation in Multiprocessor Systems Under the Interval Uncertainty of the Tasks Execution Times

M. G. Gonopolskiy[1], A. B. Glonina[1]

DOI: 10.18255/1818-1015-2020-2-218-233

[1]Lomonosov Moscow State University, 1 Leninskie Gory, Moscow 119991, Russia.

MSC2020: 68M20
Research article
Full text in Russian

Received February 11, 2020
After revision March 9, 2020
Accepted March 11, 2020

The paper presents an algorithm for the worst case response time (WCRT) estimation for multiprocessor systems with fixed-priority preemptive schedulers and the interval uncertainty of tasks execution times. Each task has a unique priority within its processor, a period, an execution time interval [BCET, WCET] and can have data dependency on other tasks. If a decrease in the execution time of the task A can lead to an increase in the response time of the another task B, then task A is called an anomalous task for task B. According to the chosen approach, in order to estimate a task's WCRT, two steps should be performed. The first one is to construct a set of anomalous tasks using the proposed algorithm for the given task. The paper provides the algorithm and the proof of its correctness. The second one is to find the WCRT estimation using a genetic algorithm. The proposed approach has been implemented software as a program in Python3. A set of experiments have been carried out in order to compare the proposed method in terms of precision and speed with two well-known WCRT estimating methods: the method that does not take into account interval uncertainty (assuming that the execution time of a given task is equal to WCET) and the brute force method. The results of the experiments have shown that, in contrast to the brute force method, the proposed method is applicable to the analysis of the real scale computing systems and also allows to achieve greater precision than the method that does not take into account interval uncertainty.

Keywords: genetic algorithm; multiprocessor systems; worst-case response times; response time analysis; scheduling anomalies; scheduling.

INFORMATION ABOUT THE AUTHORS

Mark G. Gonopolskiy correspondence author	orcid.org/0000-0002-0670-8603. E-mail: gmarkmgw@yandex.ru student.
Alevtina B. Glonina correspondence author	orcid.org/0000-0001-8716-4128. E-mail: alevtina@lvk.cs.msu.su programmer.

Funding: This work was supported by RFBR (grant No 19–07–00614).

For citation: M. G. Gonopolskiy and A. B. Glonina, "Research and Development of an Algorithm for the Response Time Estimation in Multiprocessor Systems Under the Interval Uncertainty of the Tasks Execution Times", *Modeling and analysis of information systems*, vol. 27, no. 2, pp. 218-233, 2020.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 27, № 2, 2020
сайт журнала: www.mais-journal.ru

Алгоритм оценки максимального времени отклика задач в многопроцессорных системах с интервальной неопределенностью длительности выполнения работ

М. Г. Гонопольский[1], А. Б. Глонина[1]

DOI: 10.18255/1818-1015-2020-2-218-233

[1]Московский государственный университет им. М. В. Ломоносова, Ленинские горы, д. 1, г. Москва, 119991 Россия.

УДК 519.6
Научная статья
Полный текст на русском языке

Получена 11 февраля 2020 г.
После доработки 9 марта 2020 г.
Принята к публикации 11 марта 2020 г.

В данной статье предложен алгоритм оценки максимального времени отклика задач (Worst Case Response Time — WCRT) в многопроцессорных системах с планировщиками с приоритетом и вытеснением и интервальной неопределенностью длительности выполнения работ. Между задачами имеются зависимости по данным. Для каждой задачи задан приоритет, период и интервал [BCET, WCET], которому принадлежит время выполнения на процессоре работ этой задачи. Если уменьшение времени выполнения задачи A может привести к увеличению времени отклика задачи B, то задача A называется аномальной задачей для задачи B. Согласно выбранному авторами подходу, для получения оценки WCRT некоторой задачи необходимо выполнить два шага. На первом шаге с помощью предложенного в работе алгоритма осуществляется построение множества задач, аномальных для заданной задачи. Приведено доказательство корректности этого алгоритма. На втором шаге с помощью генетического алгоритма выполняется поиск WCRT. Пространство поиска — множество всевозможных наборов длительностей выполнения аномальных задач. Было разработано инструментальное средство на языке Python3, реализующее предложенный подход. Проведено экспериментальное исследование, в ходе которого предложенный метод сравнивался по точности и скорости с двумя известными методами оценки WCRT: методом, не учитывающим интервальную неопределенность, т.е. предполагающим, что время выполнения всех работ равно WCET, а также с переборным методом. Экспериментальное исследование показало, что в отличие от переборного метода, предложенный метод применим для анализа вычислительных систем реальной размерности, а также позволяет достичь большей точности, чем метод, не учитывающий интервальную неопределенность.

Ключевые слова: генетический алгоритм; многопроцессорные системы; наихудшее время отклика; анализ времени отклика; аномалии планирования; планирование вычислений.

ИНФОРМАЦИЯ ОБ АВТОРАХ

Марк Геннадьевич Гонопольский
автор для корреспонденции
orcid.org/0000-0002-0670-8603. E-mail: gmarkmgw@yandex.ru
студент.

Алевтина Борисовна Глонина
автор для корреспонденции
orcid.org/0000-0001-8716-4128. E-mail: alevtina@lvk.cs.msu.su
программист.

Финансирование: Работа выполнена при финансовой поддержке РФФИ (грант № 19–07–00614).

Для цитирования: M. G. Gonopolskiy and A. B. Glonina, “Research and Development of an Algorithm for the Response Time Estimation in Multiprocessor Systems Under the Interval Uncertainty of the Tasks Execution Times”, *Modeling and analysis of information systems*, vol. 27, no. 2, pp. 218-233, 2020.

Введение

В данной статье рассматриваются многопроцессорные системы, представляющие собой набор процессоров, на каждом из которых выполняется набор задач под управлением динамического планировщика со статическими приоритетами и вытеснением. Процессоры соединены некоторой средой передачи данных. Все задачи периодические, в течение каждого периода должен быть выполнен один экземпляр задачи, называемый работой. Каждая задача характеризуется интервалом длительности выполнения работ, приоритетом на сопоставленном ей процессоре и периодом. Между задачами с одинаковым периодом могут существовать зависимости по данным: работа задачи-отправителя посылает сообщения работам задач-получателей и работа задачи-получателя не может начать свое выполнение, пока не получит данные от всех работ задач-отправителей.

Под конфигурацией вычислительной системы понимается совокупность числовых характеристик и взаимосвязей ее компонентов: число процессоров, привязка к ним задач, характеристики задач и зависимости по данным между ними.

Будем называть временем отклика (RT — Response Time) задачи величину, равную максимальному по всем работам этой задачи значению времен их завершения относительно начала периода. Наихудшее время отклика (WCRT — Worst Case Response Time) задачи — максимально возможное время отклика задачи для данной конфигурации системы.

WCRT используется для решения задач, возникающих при проектировании систем жесткого и мягкого реального времени. В частности, значения WCRT необходимы для проверки соблюдения директивных сроков для систем жесткого реального времени [1] и вычисления величины запаздывания для систем мягкого реального времени [2].

Большинство подходов к оценке WCRT основано на предположении, что длительность выполнения каждой работы на процессоре фиксирована и равна максимально возможной [3, 4] (в дальнейшем такой сценарий функционирования систем будем называть *базовым*). Однако этот подход может приводить к заниженным оценкам WCRT задач. Рассмотрим пример, представленный на Рис. 1 и Рис. 2.

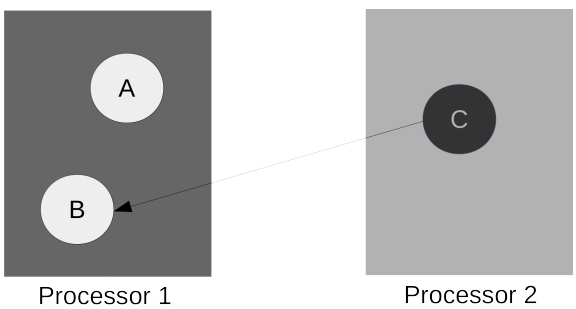

Fig. 1. Scheduling anomaly. Binding tasks to processors and data dependencies

Рис. 1. Пример аномалии планирования. Привязка задач к процессорам и зависимости по данным

Пусть имеется многопроцессорная система, представленная на Рис. 1. Приоритеты задач A и B, привязанных к первому процессору, равны 1 и 2 соответственно. Приоритет задачи C, привязанной ко второму процессору, равен 1. Все задачи имеют одинаковые интервалы длительности выполнения их работ — $[0, 2]$. На левой части Рис. 2 приведен базовый сценарий функционирования системы, то есть сценарий, согласно которому длительность выполнения каждой работы равна максимально возможной длительности (2 единицы времени для задач A, B, C). В этом сценарии работа задачи A успевает завершиться до постановки на выполнение работы задачи B, и WCRT задачи A равно 2.

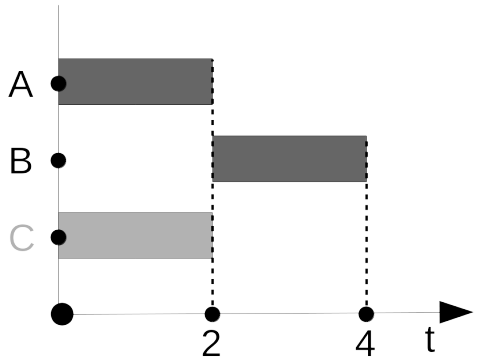

Fig. 2. Scheduling anomaly. Task A completion time increase due to task C execution time reduction

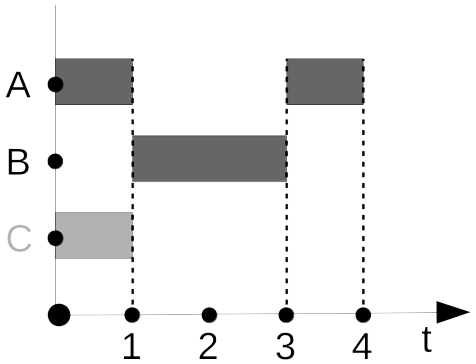

Рис. 2. Пример аномалии планирования. Увеличение времени завершения работы задачи A из-за уменьшения длительности выполнения работы задачи C

На правой части Рис. 2 приведен сценарий функционирования системы, согласно которому длительность выполнения работы задачи C равна 1, а длительности выполнения работ задач B и A по-прежнему равны максимально возможным — 2. В этом сценарии в момент времени 0 начинает выполняться работа задачи A. Работа задачи C завершается в момент времени 1 и отправляет данные работе задачи B, после чего задача B становится готовой к выполнению. Так как задача A имеет приоритет меньший, чем задача B, происходит вытеснение работы задачи A, и на выполнение ставится работа задачи B. Работа задачи B завершает свое выполнение в момент времени, равный 3, после чего происходит возобновление выполнения работы задачи A. Таким образом, работа задачи A завершается в момент времени 4 и WCRT задачи A равно 4.

Приведенный пример демонстрирует, что уменьшение длительности выполнения работы одной задачи может привести к увеличению WCRT другой задачи. И WCRT может достигаться на сценарии функционирования системы, отличном от базового. Поэтому целесообразно применение подходов, учитывающих интервальную неопределенность длительности выполнения работ.

Задачу A будем называть аномальной для задачи B, если уменьшение времени выполнения задачи A может привести к увеличению времени отклика задачи B. Формальное определение аномальной задачи будет введено в разделе 2.

Известен ряд точных методов поиска WCRT, учитывающих интервальную неопределенность длительности выполнения работ. В общем случае сложность всех этих методов экспоненциально зависит от количества задач в системе. Примером таких методов является метод целочисленного линейного программирования (ЦЛП) [5, 6]. Также стоит отметить группу точных методов, основанных на верификации моделей систем, в частности методы, использующие математический аппарат временных автоматов [7, 8]. Еще один точный метод, основанный на полном переборе всех возможных длительностей выполнения задач, приведен в [6]. Из-за экспоненциальной сложности точные методы непригодны для анализа WCRT для систем реальной размерности.

Существуют приближенные методы оценки WCRT [9—13], которые учитывают интервальную неопределенность длительности выполнения работ, но дают завышенную оценку, недостижимую на практике.

Еще один метод оценки WCRT был предложен в работе [14]. В этой работе авторы решают задачу оценки WCRT в три этапа:

1. Нахождение подмножества аномальных задач, для этого используется метод Раку и Эрнста [15].
2. Редуцирование интервалов длительностей выполнения найденных задач из пункта 1.
3. Поиск оценки времени отклика задач с использованием генетического алгоритма.

Однако метод Раку и Эрнста не находит все возможные аномальные задачи. В частности при использовании данного метода для примера, представленного на Рис. 1 и Рис. 2, задача C не будет отмечена как аномальная для задачи A.

Таким образом, все известные методы обладают теми или иными недостатками, поэтому авторами было принято решение разработать новый метод оценки WCRT, основанный на [14]. Предлагаемый метод оценки WCRT для заданной задачи состоит в последовательном выполнении двух шагов:

1. Построение множества задач, аномальных для данной задачи; для этого был разработан новый алгоритм поиска аномальных задач, корректность которого была строго доказана.
2. Поиск WCRT с помощью генетического алгоритма.

Стоит отметить, что получаемая оценка является нижней, однако она имеет большую точность, чем оценки, получаемые с помощью методов, основанных на анализе базового сценария функционирования системы. Поэтому использование предложенного метода для решения задач, возникающих при проектировании систем реального времени, вместо методов, основанных на анализе базового сценария, позволяет сократить количество ошибок при проектировании.

В отличие от методов, имеющих экспоненциальную сложность, предложенный метод применим для анализа систем реальной размерности.

В отличие от приближенных методов, дающих оценки сверху, предложенный метод позволяет найти гарантированно достижимую оценку WCRT, а также сценарий функционирования системы, на котором эта оценка достигается. Поэтому предложенный метод может использоваться для анализа пессимистичности приближенных оценок сверху: получив достижимую нижнюю оценку и убедившись в том, что она слабо отличается от верхней оценки, можно пользоваться последней, не опасаясь, что оценка сильно завышена.

1. Формальная постановка задачи

Пусть дана многопроцессорная система, содержащая N одинаковых процессоров.

Рабочая нагрузка представляется в виде кортежа $WL = (T, M)$, где $T = \{t_i\}$ — набор задач, $M = \{M_i\}$ — набор сообщений, передаваемых между задачами.

Каждая задача представляется в виде кортежа $t_i = ([bcet_i, wcet_i], H_i, P_i) \in T$, где:

- $[bcet_i, wcet_i]$ $(bcet_i, wcet_i \in \mathbb{N})$ — интервал длительности выполнения работ задачи;
- $H_i \in \mathbb{N}$ — период;
- $P_i \in \mathbb{N}$ — уникальный в рамках процессора приоритет.

Каждое сообщение представляется в виде кортежа $M_i = (S_i, R_i, dn_i)$, где:

- $S_i \in T$ — задача-отправитель;
- $R_i \in T$ — задача-получатель;
- $dn_i \in \mathbb{N}$ — длительность передачи сообщения.

Рабочей нагрузке соответствует ациклический направленный граф, далее обозначаемый как G; вершины в нем соответствует задачам, а дуги — сообщениям. Для любой пары вершин в графе G верно, что эти вершины могут быть связаны только одной дугой.

Отображение $Bind : T \longrightarrow \{1, ..., N\}$ задает привязку задач к процессорам.

Конфигурация вычислительной системы — это двойка $(WL, Bind)$, где WL — описание рабочей нагрузки, $Bind$ — привязка задач к процессорам.

Пусть $CONF$ — множество всевозможных конфигураций многопроцессорной системы. При анализе некоторой конфигурации $conf \in CONF$ многопроцессорной системы рассматривается функционирование этой системы в течение интервала планирования длительности L. Как правило, L равно наименьшему общему кратному периодов задач из T.

Для каждой задачи $t_i \in T$ на L определен набор работ $W_i = \{w_{ij}\}$, где $j \in \overline{1, [\frac{L}{P_i}]}$. Пусть $t_i \in T$ — исследуемая задача, Out_{ij} — время завершения ее j-й работы ($j/ge1$). Тогда время отклика R_{ij} этой работы определяется следующим образом:

$$R_{ij} = (Out_{ij} - (j-1) * H_i). \tag{1}$$

Время отклика R_i задачи t_i определяется следующим образом:

$$R_i = \max_j R_{ij}. \tag{2}$$

Пусть $\tau = (\tau_1, ..., \tau_{|T|})$ — набор длительностей выполнения задач для заданной конфигурации, определяющий конкретный сценарий функционирования системы; $\tau_i \in (bcet_i, wcet_i)$.

Пусть $C(conf)$ — множество всевозможных наборов τ для конфигурации $conf$. Тогда задача поиска WCRT для заданной задачи этой конфигурации имеет следующую формальную постановку.

Дано: конфигурация многопроцессорной системы $conf \in CONF$ и задача $t_i \in T$.

Найти:

$$\max_{C(conf)} R_i. \tag{3}$$

2. Алгоритм поиска аномальных задач

Согласно предложенному подходу для получения оценки WCRT заданной задачи необходимо сначала найти множество задач, аномальных для этой задачи. В данном разделе приведено описание разработанного авторами алгоритма поиска аномальных задач, а также обоснование его корректности.

2.1. Теоретическое обоснование алгоритма

Введем ряд определений, необходимых для описания алгоритма поиска аномальных задач.

Согласно разделу 1, рабочей нагрузке соответствует ациклический направленный граф G, вершины в котором взаимно однозначно соответствуют задачам, а дуги — сообщениям. Поскольку для любой пары вершин в графе G верно, что эти вершины могут быть связаны только одной дугой, то в качестве *пути* в графе G можно рассматривать последовательность вершин, в которой каждая вершина соединена дугой со следующей.

В данном разделе для краткости под задачей, находящейся в некотором пути, будем понимать вершину, соответствующую этой задаче в графе G и находящуюся в этом пути.

Нижним путем задачи A назовем путь в графе G, начинающийся с задачи T, которая непосредственно зависит по данным от A, и заканчивающийся задачей P, от которой не зависит по данным ни одна задача.

Верхним путем задачи A назовем путь в графе G, заканчивающийся такой задачей T, что A непосредственно зависит по данным от T, и начинающийся с задачи P, которая не зависит по данным ни от каких задач.

Если задача A и задача B находятся на одном процессоре и не имеют зависимостей друг от друга, то будем говорить, что между задачей B и задачей A имеется *фиктивная* связь.

Два пути будем называть связанными фиктивной связью, если между задачей, которой оканчивается один из этих путей, и задачей, с которой начинается второй путь, имеется фиктивная связь.

Задача B *аномальна* для задачи A, если существуют два сценария функционирования системы, отличающиеся друг от друга лишь тем, что в первом сценарии время выполнения работ задачи B больше, чем во втором сценарии, вследствие чего $WCRT(A)$ в первом сценарии имеет значение меньшее, чем во втором.

Задачу B назовем *аномальной задачей первого типа* для выбранной задачи A, если существуют два сценария функционирования системы, отличающиеся друг от друга лишь тем, что в первом сценарии время выполнения работ задачи B больше, чем во втором сценарии, и существует такая задача P из верхнего пути задачи A, некоторая работа которой во втором сценарии завершается позже, чем в первом сценарии, вследствие чего $WCRT(A)$ во втором сценарии больше, чем в первом сценарии. Обозначим множество аномальных задач первого типа задачи A как $IT(A)$ (Inheritable Tasks).

Задачу B назовем *аномальной задачей второго типа* для выбранной задачи A, если существуют два сценария функционирования системы, отличающиеся друг от друга лишь тем, что в первом сценарии время выполнения работ задачи B больше, чем во втором сценарии, и существует такая задача P, имеющая больший приоритет, чем задача A,находящаяся с ней на одном процессоре, и некоторая работа которой во втором сценарии начинает выполняться раньше, чем в первом сценарии, вследствие чего $WCRT(A)$ во втором сценарии больше, чем в первом сценарии. Обозначим множество аномальных задач второго типа задачи A как $HT(A)$ (Hidden Tasks).

Задачу B назовем *аномальной задачей третьего типа* для выбранной задачи A, если существуют два сценария функционирования системы, отличающиеся друг от друга лишь тем, что в первом сценарии время выполнения работ задачи B больше, чем во втором сценарии, и существует такая задача P, имеющая больший приоритет, чем задача A, находящаяся с ней на одном процессоре, и некоторая работа которой во втором сценарии начинает выполняться позже, чем в первом сценарии, вследствие чего $WCRT(A)$ во втором сценарии больше, чем в первом сценарии. Обозначим множество аномальных задач третьего типа задачи A как $HT(A)$ (Separate Tasks).

Лемма 1. *Задача B является аномальной задачей для задачи A тогда и только тогда, когда она принадлежит $IT(A) \cup HT(A) \cup ST(A)$.*

Доказательство. Докажем необходимость и достаточность.

1. Необходимость. Пусть задача B является аномальной для задачи A, то есть существуют два таких сценария функционирования системы, отличающиеся друг от друга лишь тем, что в первом сценарии время выполнения работ задачи B больше, чем во втором сценарии, и $WCRT(A)$ в первом сценарии имеет значение меньшее, чем во втором. Следовательно, имеет место один из четырех случаев:
   (a) Во втором сценарии из-за более раннего, по сравнению с первым сценарием, окончания работы некоторой задачи B закончится позже работа задачи, от которой зависит задача A. В этом случае задача $B \in IT(A)$, исходя из определения множества аномальных задач первого типа.
   (b) Во втором сценарии работу задачи A вытеснит работа другой, более приоритетной задачи P, из-за более раннего, по сравнению с первым сценарием, окончания работы некоторой задачи B. В этом случае работа задачи P либо закончится позже, по сравнению с первым сценарием, из чего следует, что $B \in ST(A)$ по определению множества аномальных задач третьего типа, либо работа задачи P начнется раньше, чем в первом сценарии, из чего следует, что задача $B \in HT(A)$, по определению множества аномальных задач второго типа.
   (c) Во втором сценарии работа задачи A будет вынуждена ждать окончания работы более приоритетной задачи P, которая начнет выполняться раньше из-за более раннего,

по сравнению с первым сценарием, окончания работы некоторой задачи B. То есть в первом сценарии задача A успеет завершиться до начала задачи P, а в втором — нет. В этом случае $B \in HT(A)$.

(d) Во втором сценарии работа задачи A будет вынуждена ждать окончания работы более приоритетной задачи P, которая начнет выполняться позже, чем в первом сценарии, из-за более раннего окончания, по сравнению с первым сценарием, работы некоторой задачи B. То есть в обоих сценариях задача A становится готовой после начала выполнения задачи P и чем позднее начнет выполняться P, тем позднее начнет выполняться A. В этом случае $B \in ST(A)$.

2. Достаточность. Любая задача, содержащаяся во множестве $IT(A) \cup HT(A) \cup ST(A)$ является аномальной, исходя из определения этих множеств.

□

Пусть $Dep(A)$ — множество всех задач, от которых задача A имеет непосредственную зависимость по данным.

Лемма 2. *Множество аномальных задач первого типа для выбранной задачи A исчерпывается объединением множеств аномальных задач для задач, от которых задача A имеет непосредственную зависимость по данным.*

Доказательство. По определению множество аномальных задач первого типа для задачи A — это все такие задачи, для каждой из которых существуют два сценария функционирования системы, отличающиеся друг от друга лишь тем, что в первом сценарии время выполнения работ соответствующей задачи больше, чем во втором сценарии, и существует такая задача P из верхнего пути задачи A, работы которой во втором сценарии завершаются позже, чем в первом сценарии. Иными словами, это аномальные задачи для задач из верхних путей задачи A. Однако в выбранном пути каждая последующая задача имеет среди своих аномальных задач первого типа аномальные задачи задач, предшествующих ей в пути. Следовательно, чтобы рассмотреть объединение множеств аномальных задач для задач из верхних путей задачи A, достаточно рассмотреть лишь объединение множеств аномальных задач для задач из $Dep(A)$. □

Пусть $I_1(A)$ — все более приоритетные задачи, чем A и находящиеся с ней на одном процессоре. $S(A) \equiv I_1(A) \cup Dep(A)$,

Пусть $M = \{z_i\}_{i=1}^{m}$ — некоторое множество задач. Обозначим как $U(M)$ следующее множество: $U(M) \equiv S(z_1) \cup ... \cup S(z_m)$. Обозначим как $D(M)$ следующее множество: $D(M) \equiv Dep(z_1) \cup ... \cup Dep(z_m)$. Положим $E_0(M) \equiv U(M) \cup M$.

Определим $E_{i+1}(M)$ как $E_{i+1}(M) \equiv U(E_i(M)) \cup E_i(M)$. Тогда, так как множество задач в системе конечно, $\exists n > 0 \in \mathbb{N}$: $\forall i \geq n\ E_i(M) = E_n(M)$. $E_n(M)$ будем обозначать как $E_{comp}(M)$

Лемма 3. *Множество $ST(A) \cup HT(A)$ для задачи A содержится во множестве $E_{comp}(D(I_1(A)))$.*

Доказательство. Пусть A — рассматриваемая задача. Предположим, что задача $B \notin E_{comp}(D(I_1(A)))$ является аномальной задачей второго или третьего типа для задачи A, то есть $B \in ST(A) \cup HT(A)$. Тогда из определения аномальной задачи второго типа и определения аномальной задачи третьего типа следует, что в результате раннего завершения работы задачи B происходит раннее или позднее завершение работы некоторой задачи $P \in I_1(A)$. Поэтому задача B принадлежит конкатенации путей, связанных фиктивными связями и заканчивающейся задачей P. Тогда задача $B \in E_{comp}(D(I_1(A)))$, поскольку будет найдено такое $k > 0 \in \mathbb{N}$: $B \in E_k(I_1(A))$, что противоречит введенному предположению. Следовательно, предположение неверно, и лемма доказана.

□

Множество всех задач из верхних путей задачи A будем обозначать как $Over(A)$.

Для краткости под глубиной задачи в графе G будем понимать максимальную длину пути до вершины в графе G, соответствующей данной задаче.

Лемма 4. *$IT(A)$ для задачи A содержится в $\cup_{P \in Over(A)} (E_{comp}(D(I_1(P))))$.*

Доказательство. Воспользуемся методом математической индукции. Пусть задана некоторая задача A графа G, с глубиной $n \in \mathbb{N}$.

1. Пусть $n = 1$, тогда задача A не имеет аномальных задач первого типа ($IT(A) = \varnothing$), поскольку не имеет зависимостей по данным. Следовательно, утверждение верно.
2. Пусть для $n = k$ утверждение верно. Докажем утверждение для $n = k + 1$. Согласно лемме 1 и лемме 2, $IT(A_{k+1}) = \cup_{P \in Dep(A_{k+1})}(IT(P) \cup HT(P) \cup ST(P))$.
   Исходя из предположения индукции, для каждой такой задачи $P \in Dep(A_{k+1})$: $IT(P)$ содержится в $\cup_{J \in Over(P)} (E_{comp}(D(I_1(J))))$. Далее, $HT(P) \cup ST(P)$ содержится в $E_{comp}(D(I_1(P)))$, согласно лемме 3. Следовательно, утверждение леммы 4 верно.

□

Теорема 1. *Множество аномальных задач для задачи A содержится в множестве $E_{comp}(D(I_1(A))) \cup (\cup_{P \in Over(A)} (E_{comp}(D(I_1(P)))))$.*

Доказательство. Утверждение теоремы напрямую следует из леммы 1, леммы 3 и леммы 4. □

Таким образом, для получения множества задач, аномальных для заданной задачи A, необходимо выполнить следующие действия:

1. Найти множество $\cup_{P \in Over(A)} (E_{comp}(D(I_1(P))))$, далее обозначаемое как $CIT(A)$, и множество $E_{comp}(D(I_1(A)))$.
2. Убрать из $E_{comp}(D(I_1(A))) \cup CIT(A)$ все задачи, которые не являются аномальными для рассматриваемой задачи.

На данный момент авторам не известен общий метод определения, является ли некоторая задача из множества $E_{comp}(D(I_1(A))) \cup CIT(A)$ аномальной для задачи A. Однако авторами был разработан алгоритм (алгоритм перекрестия), позволяющий исключить из множества $E_{comp}(D(I_1(A))) \cup CIT(A)$ ряд задач, гарантированно не являющихся аномальными.

Для задач A и B задано *перекрестие*, если существует такой нижний путь задачи A, в который входит задача B. В таком случае работа задачи B не может начать свое выполнение, пока не закончит свое выполнение работа задачи A. Следовательно, уменьшение времени выполнения работ задачи B не может повлиять на время отклика задачи A, и задача B не является аномальной для задачи A.

Исключив с помощью алгоритма перекрестия из множества $E_{comp}(D(I_1(A))) \cup CIT(A)$ ряд неаномальных задач, получим множество, полностью содержащее все аномальные задачи для рассматриваемой задачи, однако, возможно, содержащее некоторые лишние задачи. Это не приведет к получению некорректных результатов, но может увеличить длительность работы генетического алгоритма, используемого для поиска WCRT.

Для удобства вместо $E_{comp}(D(I_1(A)))$ будем использовать множества:

- $I_2(A)$ — множество всех задач, входящих в верхние пути задач из $I_1(A)$;
- $I_3(A)$ — множество $\cup_{P \in I_2(A)} (E_{comp}(I_1(P))$.

Лемма 5. *Множество $I_2(A) \cup I_3(A)$ для задачи A совпадает с $E_{comp}(D(I_1(A)))$.*

Доказательство. Пусть A — некоторая задача.

1. Докажем, что для любой задачи $B \in I_2(A)$ верно, что $B \in E_{comp}(D(I_1(A)))$.
   M — некоторое множество задач. Положим $Dep_0(M) \equiv D(M)$.
   Определим $D_{i+1}(M)$ как $D_{i+1}(M) \equiv Dep(D_i(M)) \cup D_i(M)$.
   Тогда, так как множество задач в системе конечно, $\exists\ n > 0 \in \mathbb{N}$: $\forall i \geq n\ D_i(M) = D_n(M)$. $D_n(M)$ будем обозначать как $D_{comp}(M)$. Возьмем в качестве M множество $D(I_1(A))$ и получим множество $D_{comp}(D(I_1(A))) = I_2(A)$. По построению $E_{comp}(M) = E_n(M)$, такое что $E_i(M) = E_n(M)$, для любого $i > n$, где $E_{i+1}(M) = U(E_i(M)) \cup E_i(M)$, $E_0(M) = U(M) \cup M$. По определению $U(M) = (I_1(z_1) \cup Dep(z_1)) \cup ... \cup (I_1(z_m) \cup Dep(z_m))$, $D(M) = Dep(z_1) \cup ... \cup Dep(z_m)$, $z_i \in M$, $|M| = m$. Следовательно, $D(M) \in U(M)$ и $D_i(M) \in E_i(M)$. Поэтому $D_{comp}(D(I_1(A))) \in E_{comp}(D(I_1(A)))$.
2. Докажем, что для любой задачи $B \in I_3(A)$ верно, что $B \in E_{comp}(D(I_1(A)))$.
   Пусть $P \in I_2(A)$, $B \in E_{comp}(I_1(P))$. Тогда существует такая, имеющая фиктивную связь с задачей P, задача C (может быть, совпадающая с задачей B), на которую оканчивается конкатенация соединенных фиктивными связями путей, содержащая задачу B. Из рассуждений предыдущего пункта следует, что найдется такое k, что $P \in D_k(D(I_1(A)))$. Следовательно, по построению множества $E_{comp}(D(I_1(A)))$, множество $E_{comp}(S(P))$ полностью содержится во множестве $E_{comp}(D(I_1(A)))$. Так как множество $S(P) = I_1(P) \cup Dep(P)$, тогда множество $E_{comp}(I_1(P))$ содержится во множестве $E_{comp}(S(P))$. Следовательно, $E_{comp}(I_1(P))$ содержится в $E_{comp}(D(I_1(A)))$. Следовательно, $B \in E_{comp}(D(I_1(A)))$.
3. Докажем, что для любой задачи $T \in E_{comp}(D(I_1(A)))$ верно, что $T \in I_3(A) \cup I_2(A)$.
   Напомним, что $I_3(A) = \cup_{P \in I_2(A)} (E_{comp}(I_1(P))$. Проведем доказательство от противного. Предположим, что существует некоторая задача $T \in E_{comp}(D(I_1(A)))$ такая, что $T \notin \cup_{P \in I_2(A)} (E_{comp}(I_1(P)) \cup I_2(A)$. Тогда, по построению множества $E_{comp}(D(I_1(A)))$, возможны два случая:
   (a) Задача T совпадает с некоторой задачей N из $D(I_1(A))$, тогда $T \in I_2(A)$, что противоречит введенному предположению.
   (b) Задача T принадлежит конкатенации путей, связанных фиктивными связями и заканчивающейся задачей N. Тогда существует процессор, на котором наблюдается фиктивная связь между некоторой задачей C_1 из верхнего пути задачи N (или совпадающей с ней) и некоторой отличной от C_1 задачей C_2, которой оканчивается конкатенация путей, связанных фиктивными связями, содержащая задачу T. Из этого следует, что задача $T \in E_{comp}(I_1(C_1))$, так как найдется такое k, что $T \in E_k(I_1(C_1))$. Следовательно, $T \in \cup_{P \in I_2(A)} (E_{comp}(I_1(P))$, что противоречит введенному предположению.
   Таким образом, $T \in I_3(A) \cup I_2(A)$, что и требовалось доказать.

□

Использование такого разбиения упрощает реализацию алгоритма, описанного далее. Удобно найти множество $I_2(A)$ в виде верхних путей задач из $I_1(A)$ и на основании этого множества найти $I_3(A)$.

Итого, для получения множества, содержащего все аномальные задачи для заданной задачи A необходимо найти $I_1(A) \cup I_2(A) \cup I_3(A)$, то есть $(\cup_{P \in Over(A)} (E_{comp}(D(I_1(P))))) \cup E_{comp}(D(I_1(A)))$, а затем исключить из этого множества задачи, заведомо не являющиеся аномальными, согласно алгоритму перекрестия. Корректность предложенного алгоритма поиска аномальных задач (а именно тот факт, что полученное множество будет содержать все задачи, аномальные для заданной задачи A) следует из теоремы 1 и леммы 5.

2.2. Описание алгоритма в виде псевдокода

Уровнями графа G назовем множества задач, каждое из которых содержит лишь задачи, имеющие одинаковую глубину в графе G. Например, *уровень ноль* содержит лишь задачи, не имеющие

зависимостей; *уровень один* — задачи, имеющие зависимости только от задач уровня ноль и т.д. Разбиение множества задач на уровни выполняется при поиске аномальных задач первого типа.

Введем обозначения для вспомогательных функций:

- UnderLine(A) — функция, возвращающая множество всех нижних путей задачи A.
- OverLine(A) — функция, возвращающая множество всех верхних путей задачи A.
- Getanomaltasks(A) — функция, возвращающая множество всех известных на данный момент аномальных задач для задачи A.
- Dep(A) — функция, возвращающая множество $Dep(A)$, то есть множество всех задач, от которых задача A непосредственно имеет зависимость по данным.
- CoreNum($task$) — функция, возвращающая номер процессора, на котором выполняется задача $task$.
- TasksFromCore(HW) — функция, возвращающая множество задач, работы которых выполняются на процессоре с номером HW.
- Sop(A) — функция, возвращающая множество $S(A)$, то есть множество всех задач, с которыми задача A имеет фиктивную связь, и множество всех задач, от которых задача A непосредственно имеет зависимость по данным.
- GetLayers(G) — функция, возвращающая упорядоченное по возрастанию номеров множества уровней.
- I1(A) — функция, возвращающая $I_1(A)$ для задачи A, то есть множество задач, выполняющихся на одном процессоре с задачей A и имеющих приоритет, больший, чем у задачи A.
- Depend(M) — функция, возвращающая $D(M)$ для множества M.
- Priority(M) — функция, возвращающая приоритет задачи A на соответствующем ей процессоре.

Приведем описания на псевдокоде функций, входящих в состав алгоритма поиска аномальных задач.

Алгоритм нахождения перекрестия между задачами T и S реализует функция isCross(T,S). Возвращает значение логического типа данных: если между выбранными задачами есть перекрестие, тогда возвращается *Истина*, иначе — *Ложь*. Функция имеет следующий вид:

```
|bool IsCross(T,S):
|   F := false // флаг наличия перекрестия
|   if T ≠ S:
|      for Path from UnderLine(T)
|         for task from Path:
|            if task = S:
|               F := true // задача не нуждается в рассмотрении
|   else:
|      F := true // задача не нуждается в рассмотрении
|   return F
```

Алгоритм поиска аномальных задач первого типа реализует функция CIT(A). Согласно лемме 1, для того, чтобы найти аномальные задачи первого типа для данной задачи A достаточно выполнить проход по всем задачам, от которых задача A имеет непосредственную зависимость по данным, и объединить все аномальные задачи этих задач в одно множество. В процессе применения алгоритма осуществляется обход графа по уровням — от нулевого до последнего, что позволяет заменить полное вычисление множества $(\cup_{P \in Over(A)} (E_{comp}(D(I_1(P)))))$ обходом задач из множества $Dep(A)$. Функция имеет следующий вид:

```
|set CIT(A):
|   Tasks := ∅
|   for T from Dep(A):
|       Tasks := Tasks ∪ Getanomaltasks(T)
|   return Tasks
```

Алгоритм поиска $E_{comp}(M)$ для множества $\notin \equiv D(I_1(A))$, для некоторой задачи A, реализует функция Ecomp(M). Как упоминалось ранее, множество $E_{comp}(M)$ для некоторого множества M строится путем объединения множеств $I_2(A)$ и $I_3(A)$. Таким образом, для каждой задачи $B \in I_1(A)$ требуется найти множество верхних путей, затем для каждой задачи C из найденных верхних путей требуется провести операцию поиска соответствующей части множества $I_3(A)$, представляющую из себя $E_{comp}(S(I_1(C))) \cup I_1(C)$. Функция имеет следующий вид:

```
|set Ecomp(M):
|   AnomalSet := ∅
|   for T from M:
|       for OverT from OverLine(T):
|           TmpSet := ∅
|           for Task from OverT:
|               AnomalSet.add(Task) // поиск I2
|               HW := CoreNum(Task)
|               CoreTasks := TasksFromCore(HW)
|               for Hidd from CoreTasks:
|                   if Priority(Hidd) ≤ Priority(Task):
|                       CoreTasks := CoreTasks\Hidd // удалить задачу из множества
|               for Hidd from CoreTasks:
|                   TmpSet := TmpSet ∪ Hidd
|                   TmpSet := Ecomp(Sop(Hidd)) ∪ DummySet // поиск I3
|           AnomalSet := TmpSet ∪ AnomalSet
|   return AnomalSet
```

Алгоритм поиска аномальных задач реализует функция FindAnomalTasks(G). Эта функция возвращает массив множеств $E_{comp}(D(I_1(A))) \cup CIT(A)$ для каждой задачи A в системе, описываемой графом G. Функция имеет следующий вид:

```
|set array FindAnomalTasks(G):
|   Layers := GetLayers(G) // разбить граф на уровни
|   for Layer from Layers:
|       for T from Layer:
|           MainSets[T] := CIT(T)
|           Positive := ∅
|           for Susp from I1(T):
|               if IsCross(T,Susp): // перекрестие ⟶ следующая задача
|                   continue
|               else Positive = Positive ∪ Susp
|           MainSets[T] := MainSet[T] ∪ Ecomp(Depend(Positive))
|   return MainSets
```

3. Генетический алгоритм оценки WCRT

Авторами был разработан алгоритм, позволяющий получить оценку WCRT для заданной задачи и заданной конфигурации системы, учитывающий влияние аномальных задач. За основу была взята идея из [14]: оценка максимального времени отклика задач проводится с помощью генетического алгоритма.

За основу взят классический генетический алгоритм [16].

Пусть задана некоторая конфигурация системы и задача T, для которой требуется найти оценку WCRT. *Особь* представляет собой последовательность генов. Количество генов равно количеству задач, аномальных для задачи T. Каждый ген соответствует некоторой задаче A, аномальной для задачи T, и имеет целочисленное значение из интервала длительности выполнения задачи A. В том числе значение гена может быть равно одной из границ этого интервала.

Функция приспособленности особи — время отклика задачи T, вычисленное для случая, когда длительность выполнения каждой аномальной задачи равна значению соответствующего гена, а длительности выполнения остальных задач равны правым границам соответствующих интервалов. Для вычисления функции приспособленности используется средство, описанное в работе [17]. Авторами этой работы предложен метод получения временной диаграммы функционирования вычислительной системы с заданной конфигурацией при фиксированных длительностях выполнения задач. Временная диаграмма содержит события постановки работ на выполнение, события вытеснения и завершения работ. Значение времени отклика задачи получается посредством анализа временной диаграммы согласно формулам, приведенным в разделе 1.

В качестве оператора *скрещивания* используется классическое одноточечное скрещивание. *Мутация* представляет собой изменение некоторого количества генов в особи на случайные значения из соответствующих интервалов длительности выполнения задач. В результате выполнения операторов скрещивания и мутации в популяцию попадают новые особи и ее размер возрастает. В процессе *отбора* формируется новая популяция таким образом, что в нее попадает некоторое количество лучших особей текущей популяции, а также для обеспечения разнообразия, некоторое количество худших особей. При этом итоговый размер новой популяции равен размеру предыдущей популяции до выполнения скрещивания и мутации.

Критерием останова является отсутствие изменения функции приспособленности лучшей особи в течение определенного числа итераций. При возникновении ситуации, в которой итоговый результат работы алгоритма меньше, чем WCRT в базовом сценарии функционирования системы, в качестве результата берется WCRT для базового сценария.

4. Экспериментальное исследование

Предложенный авторами статьи метод сравнивался с методом, основанном на анализе базового сценария функционирования системы, и с методом, основанном на полном переборе. Критериями сравнения выступали время и точность работы алгоритма.

Для проведения экспериментов было разработано инструментальное средство на языке Python3, реализующее предложенный в разделах 2 и 3 подход. Также реализован метод полного перебора и метод, основанный на анализе базового сценария функционирования системы. Кроме того разработано средство генерации конфигураций систем различной размерности.

Эксперименты проводились на данных, соответствующих реальной системе, и на искусственно сгенерированных данных.

Поскольку истинное значение WCRT можно определить за приемлемое время лишь для конфигураций небольшой размерности, сравнение точности генетического алгоритма с точностью метода, основанного на базовым сценарии, было выполнено лишь для таких конфигураций.

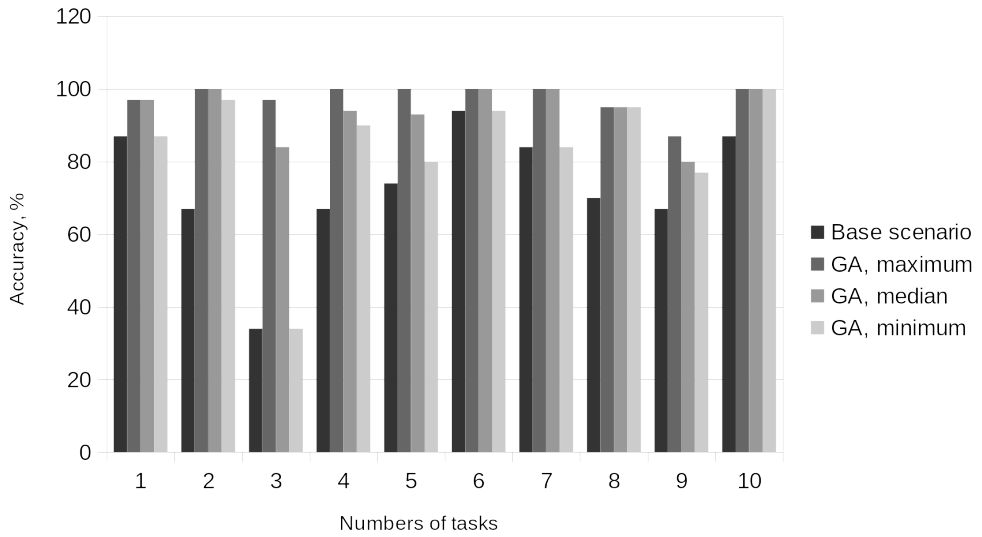

Fig. 3. Comparison of the minimum, median and maximum accuracy of the genetic algorithm with the method based on the base scenario analysis. Artificially generated data

Рис. 3. Сравнение минимальной, медианной и максимальной точности генетического алгоритма (GA) и метода, основанного на анализе базового сценария (Base scenario), на искусственно сгенерированных данных

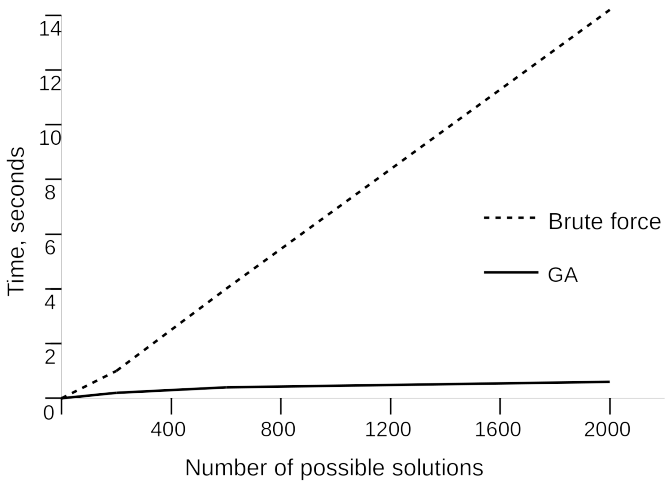

Fig. 4. Comparison of genetic algorithm speed to speed of the brute force depending on the number of possible solutions. Artificially generated data

Рис. 4. Сравнение скорости работы генетического алгоритма (GA) со скоростью работы полного перебора (Brute force) в зависимости от количества возможных решений, на искусственно сгенерированных данных

На Рис. 3 представлены результаты сравнения точности алгоритмов. Во всех экспериментах медианное значение точности генетического алгоритма было не меньше 80%. В 70% случаев алгоритму удалось достигнуть точного значения.

На Рис. 4 представлены результаты сравнения времени работы предложенного генетического алгоритма со временем работы полного перебора на искусственно сгенерированных данных.

Также была проведена серия экспериментов на данных, приближенных к реальным. Конфигурация вычислительной системы, для задач которой производилась оценка WCRT, содержит 164 задачи и 100 сообщений. Результаты этих экспериментов представлены на Рис. 5

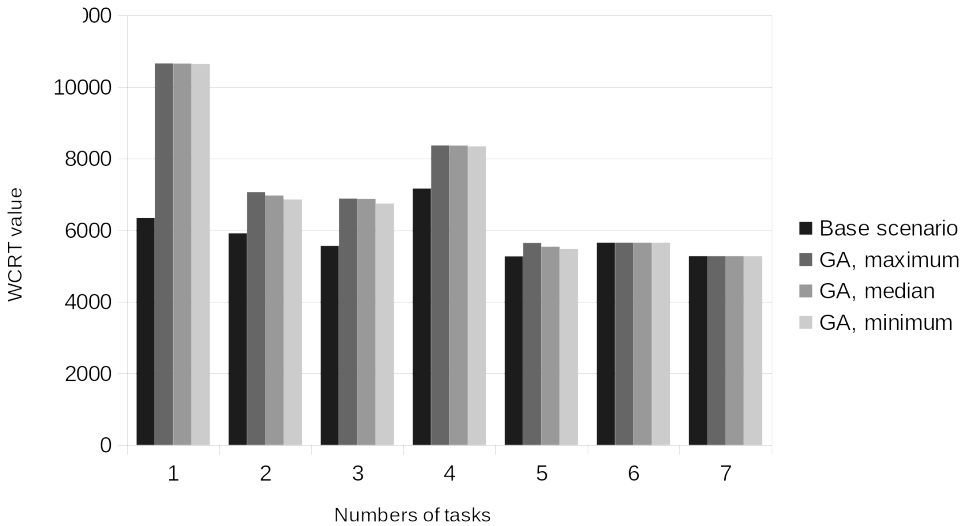

Fig. 5. Comparison of the minimum, median and maximum results of the genetic algorithm with the results of a method based on the base scenario analysis. Realistic data

Рис. 5. Сравнение минимальных, медианных и максимальных результатов генетического алгоритма (GA) и результатов метода, основанного на анализе базового сценария (Base scenario), на данных, приближенных к реальным

Были получены следующие результаты: в зависимости от входных данных точность разработанного алгоритма превышает точность метода, основанного на базовом сценарии в 1.0 — 1.68 раз; при количестве возможных решений, равном 10^{36}, разработанный алгоритм работает в среднем около 12 минут.

5. Заключение

Авторами разработан метод оценки WCRT, заключающийся в последовательном применении двух предложенных ими алгоритмов: алгоритма поиска аномальных задач, позволяющего найти множество задач, включающее в себя все аномальные задачи для рассматриваемой задачи, и алгоритма оценки времени отклика задач на основе множества аномальных задач.

Было выполнено экспериментальное исследование метода на искусственно сгенерированных данных и данных, приближенных к реальным, показавшее, что предложенный метод позволяет повысить точность оценки WCRT по сравнению с методом, не учитывающим интервальную неопределенность (до 1.68 раз на реальных данных).

В качестве направлений дальнейших исследований можно выделить следующие задачи: разработка, исследование и сравнение с ГА других поисковых алгоритмов оценки WCRT, в том числе точных (например, метод ветвей и границ); исследование интервалов длительности выполнения работ для редукции пространства возможных решений; доработка предложенного подхода с целью получения оценок WCRT одновременно для групп задач.

References

[1] N. Holsti, T. Langbacka, and S. Saarinen, "Using a Worst-Case Execution Time Tool for Real-Time Verification of the DEBIE Software", *EUROPEAN SPACE AGENCY-PUBLICATIONS-ESA SP*, vol. 457, pp. 307–312, 2000.

[2] C. Liu and J. H. Anderson, "Task Scheduling with Self-Suspensions in Soft Real-Time Multiprocessor Systems", in *2009 30th IEEE Real-Time Systems Symposium*, 2009, pp. 425–436.

[3] J. C. P. Gutierrez, J. J. G. Garcia, and M. G. Harbour, "Best-Case Analysis for Improving the Worst-Case Schedulability Test for Distributed Hard Real-Time Systems", in *Proceedings of 10th EUROMICRO Workshop on Real-Time Systems*, 1998, pp. 35–44.

[4] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment", *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.

[5] J. Kim and et al., "An ILP-Based Worst-Case Performance Analysis Technique for Distributed Real-Time Embedded Systems", in *Proceedings of IEEE 33rd Real-Time Systems Symposium*, 2012, pp. 363–372.

[6] R. Wilhelm, "Why AI+ ILP is Good for WCET, but MC is not, nor ILP Alone", in *International Workshop on Verification, Model Checking, and Abstract Interpretation*, 2004, pp. 309–322.

[7] A. Brekling, M. R. Hansen, and J. Madsen, "Models and Formal Verification of Multiprocessor System-on-Chips", *The Journal of Logic and Algebraic Programming*, vol. 77, no. 1-2, pp. 1–19, 2008.

[8] J. Kany and S. Madsen, "Design Optimisation of Fault-Tolerant Event-Triggered Embedded Systems", PhD thesis, Doctoral dissertation, Master's thesis, Tech. Univ. of Denmark, Lyngby, Denmark, 2007, 176 pp.

[9] J. Kim and et al., "A Novel Analytical Method for Worst Case Response Time Estimation of Distributed Embedded Systems", in *Proceedings of 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2013, pp. 1–10.

[10] K. Tindell and J. Clark, "Holistic Schedulability Analysis for Distributed Hard Real-Time Systems", *Microprocessing and microprogramming*, vol. 40, no. 2-3, pp. 117–134, 1994.

[11] R. I. Davis and et al., "Controller Area Network (CAN) Schedulability Analysis: Refuted, Revisited and Revised", *Real-Time Systems*, vol. 35, no. 3, pp. 239–272, 2007.

[12] J. C. Palencia and M. G. Harbour, "Schedulability Analysis for Tasks with Static and Dynamic Offsets", in *Proceedings 19th IEEE Real-Time Systems Symposium*, 1998, pp. 26–37.

[13] O. Redell, "Analysis of Tree-Shaped Transactions in Distributed Real-Time Systems", in *Proceedings of 16th Euromicro Conference on Real-Time Systems*, 2004, pp. 239–248.

[14] S. Samii, S. Rafiliu, P. Eles, and Z. Peng, "A Simulation Methodology for Worst-Case Response Time Estimation of Distributed Real-Time Systems", in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2008, pp. 556–561.

[15] R. Racu and R. Ernst, "Scheduling Anomaly Detection and Optimization for Distributed Systems with Preemptive Task-Sets", in *12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06)*, 2006, pp. 325–334.

[16] V. M. Kureychik, *Geneticheskiye algoritmy i ih primeneniye*. Taganrog: Izd-vo TRTU, 2002, 244 pp.

[17] A. B. Glonina, "Programmnoye sredstvo modelirovaniya modulnyh vychislitelnih sistem dlya proverki dopustimosti ih konfiguratsiy", *Programmniye produkty i sistemy*, vol. 30, no. 4, pp. 574–582, 2017.