Моделирование и анализ информационных систем. Т. 23, № 2 (2016), с. 119–136
Modeling and Analysis of Information Systems. Vol. 23, No 2 (2016), pp. 119–136

DOI: 10.18255/1818-1015-2016-2-119-136

УДК 517.9

Метод сбалансированного выбора механизмов обеспечения отказоустойчивости для распределённых вычислительных систем

Волканов Д. Ю.

получена 31 марта 2016

Аннотация. В статье рассматривается задача сбалансированного выбора набора механизмов обеспечения отказоустойчивости для распределённых вычислительных систем (РВС). В данной задаче требуется выбрать сбалансированный набор вариантов модулей РВС максимальный по надёжности при ограничениях на стоимость на множестве возможных вариантов РВС. В статье приводится описание рассматриваемых механизмов обеспечения отказоустойчивости, из которых происходит выбор, рассматривается математическая модель в рамках которой дана постановка задачи и метод её решения. Данная задача широко рассматривается в литературе. Приводится подробное описание метода выбора сбалансированного набора механизмов обеспечения отказоустойчивости для РВС. Предложенный метод представляет собой эволюционный алгоритм с использованием схемы нечёткой логики. Схема нечёткой логики в процессе работы алгоритма анализирует результаты его работы в каждом поколении и, исходя из этой информации, корректирует параметры эволюционного алгоритма. Метод позволяет получить эффективное решение, что показано в экспериментальном исследовании. Ключевой особенностью предлагаемого подхода является использование адаптивной схемы. Метод реализован в виде программного средства, интегрированного со средой моделирования ДИАНА. Заключение статьи содержит краткое описание будущих исследований.

Ключевые слова: надёжность, отказоустойчивость, вычислительные системы, генетический алгоритм, задача оптимизации надёжности, механизмы обеспечения отказоустойчивости, эволюционный алгоритм

Для цитирования: Волканов Д. Ю., "Метод сбалансированного выбора механизмов обеспечения отказоустойчивости для распределённых вычислительных систем", *Моделирование и анализ информационных систем*, **23**:2 (2016), 119–136.

Об авторах:
Волканов Дмитрий Юрьевич, orcid.org/0000-0001-9940-5822, ассистент,
Московский государственный университет имени М.В. Ломоносова,
119991, ГСП-1, Россия, Москва, Ленинские горы, МГУ имени М.В. Ломоносова,
2-й учебный корпус, факультет ВМК, комната 764,
e-mail: volkanov@lvk.cs.msu.su

Благодарности:
Исследование выполнено при поддержке Министерства образования и науки Российской Федерации, уникальный номер (ID) RFMEFI60714X0070, соглашение о предоставлении субсидии 14.607.21.0070.

Введение

В данной статье рассматривается задача сбалансированного выбора набора механизмов обеспечения отказоустойчивости (МОО) для распределённых вычислительных систем (РВС) в следующей постановке. Пусть нам задана РВС в виде набора модулей и структуры связей между модулями РВС. К каждому модулю может применяться один МОО. Каждый модуль содержит не менее одного аппаратного и программного компонента. Каждый компонент может иметь несколько версий. Количество аппаратных и программных компонентов в модуле зависит от МОО, используемого для модуля. Тем самым возникает несколько вариантов РВС. Требуется выбрать сбалансированный набор вариантов модулей РВС, эффективный по определенным критериям на множестве возможных вариантов РВС.

РВС оценивается по четырём показателям: функциональность, производительность, надёжность и стоимость [1]. При разработке РВС основным оптимизируемым показателем является надежность при фиксированных ограничениях на стоимость. Надёжность РВС можно повысить двумя основными способами [2, 3]:

1. Использование более надёжных отдельных модулей, из которых состоит система;

2. Использование МОО на уровне модулей РВС.

Оба этих подхода приводят к увеличению стоимости и/или уменьшению производительности РВС. Поэтому при проектировании РВС необходимо увеличивать надёжность сбалансированно, то есть набор МОО и версии аппаратных и программных компонентов модулей РВС должны быть выбраны так, чтобы обеспечить максимальную надёжность, при ограничениях на стоимость. Данная задача в литературе называется задачей выбора сбалансированного набора МОО для вычислительных систем с целью повышения их надёжности или более кратко задачей оптимизации надёжности [3].

Надо сразу отметить, что задача, в которой максимизируется надёжность вычислительной системы при ограничении на стоимость системы, рассматривалась разными исследователями начиная с 60-х годов прошлого века. В [4] было показано, что эта задача является NP-трудной. Обзоры работ различных временных периодов приведены в [3, 5–10].

В большинстве рассматриваемых работ в качестве МОО рассматривалось исключительно резервирование и оценивалась надёжность только для аппаратных или программных компонентов модуля РВС. Как было показано в работе [11], для РВС рассмотрение аппаратной и программной составляющих РВС по отдельности не позволяет адекватно описывать характеристики системы. Поэтому для оценки надёжности всей системы необходимо рассматривать надёжность аппаратуры и программ совместно. В статье [12] авторы допускают возможность использования различных МОО. В статье [13] те же авторы предлагают решение поставленной задачи с помощью классического генетического алгоритма. Эти работы также не лишены недостатков. В них в качестве алгоритмов, при помощи которых ищется решение, используются классические схемы генетического алгоритма и алгоритма имитации отжига, выбор которых обусловлен простотой реализации данных алгоритмов, а не возможностью нахождения эффективного решения.

Здесь будет рассмотрена задача обозначенная в [3] как задача P_1. Для увеличения вычислительной эффективности или для того, чтобы избегать преждевременного попадания в локальный экстремум, многие исследователи используют эффективные комбинации эволюционного алгоритма (ЭА) с другими эвристическими алгоритмами, нейронными сетями и методами локального поиска экстремума. Эти комбинации называются гибридными эволюционными алгоритмами и являются одними из наиболее перспективных направлений в разработке оптимизационных методов. Если при кодировании задачи в методе, использующем схему генетического алгоритма, используется кодировка отличная от бинарной, то такой алгоритм называется эволюционным алгоритмом. Для решения задачи в данной статье будет рассмотрен адаптивный гибридный эволюционный алгоритм (АГЭА). Этот алгоритм включает в себя процедуру автоматического изменения параметров и является развитием алгоритма, предложенного в статье [14].

Структура работы следующая. В следующем разделе вводятся необходимые понятия и определения. В разделе 2 описываются механизмы обеспечения отказоустойчивости, рассматриваемые в данной работе. Раздел 3 содержит модель функционирования РВС. В разделе 4 приводится постановка решаемой задачи. Раздел 5 содержит алгоритм решения поставленной задачи. Экспериментальное исследование предлагаемого алгоритма приводится в разделе 6. В заключении кратко перечислены полученные результаты и список перспективных направлений дальнейших исследований.

1. Основные понятия

У каждого модуля РВС есть *точки предоставления услуг* другим модулям. Модуль может иметь иерархическую структуру, то есть он может состоять из других модулей, каждый из которых имеет свои точки предоставления услуг. Один и тот же модуль не может входить в состав двух разных модулей и самого себя. Внешняя среда и человек-оператор РВС могут рассматриваться как специфичные модули РВС. *Состоянием* модуля назовем совокупность значений его элементов памяти (ячеек ОЗУ, ПЗУ, регистров и т.д.), доступных в точке предоставления услуг модуля. Состояния модуля в точках предоставления услуг этого модуля будем называть *внешними*, а остальные *внутренними*. То есть, если модуль состоит из нескольких модулей, то внешние состояния внутренних модулей являются внутренними состояниями для внешнего/объемлющего модуля. *Действием* модуля РВС будем называть смену состояния модуля. *Поведением* модуля РВС будем называть последовательность действий модуля [1].

Услугой, предоставляемой модулем, будем называть поведение модуля, влияющее на состояния, а может быть, и на поведение других модулей РВС. Услуга является *корректной*, если внешние состояния модуля соответствуют функциональной спецификации. Услуга может быть распределённой, то есть складываться из внешних состояний нескольких модулей, но мы для простоты изложения не будем рассматривать этот случай.

Будем говорить об *отказе услуги*, если предоставляемая услуга некорректна. Если в модуле произошёл отказ хотя бы одной из услуг, то будем говорить об *отказе*

модуля или просто *отказе*. *Ошибка* – это отклонение одной из частей внутреннего состояния от ожидаемого. Поскольку услуга – это последовательность состояний, то некоторые ошибки могут не изменять внешних состояний модуля и не приводить к отказу. Причину ошибки будем называть *неисправностью*.

Под *надёжностью* РВС будем понимать способность РВС сохранять доступность услуг, при определенных спецификацией условиях, в течение заданного периода времени. Для количественной оценки надёжности вводят *меру надёжности* как вероятность безотказной работы (то есть отсутствие отказов услуг) в течение интервала $[0, t_{end}]$, при условии, что система находится в работоспособном состоянии в момент времени t=0 [15].

Для обеспечения необходимого значения надёжности применяют МОО, предотвращающие отказ РВС при отказах её модулей. МОО работают на этапе эксплуатации РВС и предотвращают отказ при возникновении ошибки. В следующем разделе рассматриваемые МОО описаны более подробно.

2. Рассматриваемые механизмы обеспечения отказоустойчивости

В этой статье будем считать, что в каждом модуле РВС может использоваться один из следующих МОО: один из двух видов N-версионного программирования (NVP/0/1, NVP/1/1), восстановление блоками (RB/1/1) или не использоваться никакие МОО (None) [16].

Структурная схема N-версионного программирования (NVP) [17, 18] состоит из блока принятия решений (голосователя) и N независимо разработанных версий программы. N обычно является нечётным числом. В NVP все N версий программы для одной и той же задачи запускаются параллелельно, а результаты их работы собираются и обрабатываются голосователем. Тот результат, который получается в большинстве версий, принимается голосователем как финальный. Различные аппаратные неисправности могут вызвать ошибки в программе и недопустимый ответ в результате её работы. В этой статье рассматривается механизм NVP для N=3. NVP/0/1 [13] не позволяет предотвратить отказы, возникающие вследствие неисправностей аппаратуры, но позволяет предотвращать одну неисправность в программных версиях. Архитектура этого МОО включает голосователь и 3 версии программы, запускаемых параллельно на одном аппаратном компоненте. Главное отличие МОО NVP/1/1 [13] от NVP/0/1 заключается в том, что каждая программная версия выполняется на отдельном аппаратном компоненте.

МОО RB/1/1 [13, 17, 18] включает в себя две версии программного компонента. Если результат работы одной признан некорректным, то вычисления "откатывают" в начало и запускают вторую версию. Процесс продолжается, пока результат одной из версий не будет принят, либо после определённого числа попыток результат работы всех версий не будет отклонён.

3. Модель функционирования РВС

3.1. Основные предположения

В данной работе будем предполагать:

1. Все модули неремонтируемы, то есть после любого отказа модуль считается неработоспособным.

2. Моменты появления отказов для аппаратных компонентов модулей РВС являются статистически независимыми.

3. Вся аппаратные компоненты модулей РВС вычислительной системы являются активными.

4. Для всех модулей вычислительной системы количество доступных версий программных и аппаратных компонентов фиксировано и постоянно. Набор этих версий для каждого модуля свой.

5. Для каждого компонента модуля РВС интенсивность отказов постоянна, а, соответственно, распределение наработки на отказ является показательным, что позволяет работать с функциями надёжности компонентов системы как со скалярными величинами.

Эти предположения позволят нам ниже трактовать рассматриваемую задачу как задачу дискретной оптимизации.

3.2. Конфигурация РВС

Введем обозначения:

- n – количество модулей в РВС;
- U_i – модуль вычислительной системы, здесь и далее $i \in [1, n]$;
- H_i – аппаратный компонент модуля U_i;
- S_i – программный компонент модуля U_i;
- p_i – количество версий аппаратного компонента H_i;
- q_i – количество версий программного компонента S_i;
- $H_{i,j}$ – j-я версия аппаратного компонента H_i в модуле U_i, $i \in [1, n]$, $j \in [1, p_i]$
- $S_{i,j}$ – j-я версия аппаратного компонента S_i в модуле U_i, $i \in [1, n]$, $j \in [1, q_i]$
- FT_{avail} – множество доступных МОО для РВС. В данной работе $FT_{avail} = \{None, NVP/0/1, NVP/1/1, RB/1/1\}$
- $FT_i \subseteq FT_{avail}$ – для каждого модуля i определён свой набор доступных МОО;

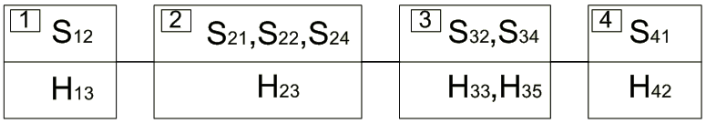

Рис. 1. Пример конфигурации РВС

- $H_i^{F_i}$ – множество версий $H_{i,j}$ аппаратного компонента H_i, фактически используемый в модуле U_i, зависит от типа применяемого МОО;
- $S_i^{F_i}$ – множество версий $S_{i,j}$ программного компонента S_i, фактически используемый в модуле U_i, зависит от типа применяемого МОО;

Конфигурация РВС $System_{RTES} = \{\{H_1^{F_1}, S_1^{F_1}, F_1\}, ..., \{H_i^{F_i}, S_i^{F_i}, F_i\}, ..., \{H_n^{F_n}, S_n^{F_n}, F_n\}\}$, $i \in [1, n]$, включает в себя число и версии используемых программных и аппаратных компонентов и набор МОО. Пример конфигурации РВС приведён на рисунке 1.

Множество $\{H_i^{F_i}, S_i^{F_i}, F_i\}$ является корректным в одном из следующих случаев:

- если $F_i = None$, то $H_i^{F_i} = \{H_{i,k}\}$ $(k \in [1, p_i])$, а $S_i^{F_i} = \{S_{i,j}\}$ $(j \in [1, q_i])$;
- если $F_i = NVP/0/1$, то $H_i^{F_i} = \{H_{i,k}\}$, $(k \in [1, p_i])$, а $S_i^{F_i} = \{S_{i,j1}, S_{i,j2}, S_{i,j3}\}$, причём $j1, j2, j3 \in [1, q_i], j1 \neq j2 \neq j3$;
- если $F_i = NVP/1/1$, то $H_i^{F_i}$ содержит ровно три элемента $H_i^{F_i} = \{H_{i,k1}, H_{i,k2}, H_{i,k3}\}$ $(k1, k2, k3 \in [1, p_i])$, а $S_i^{F_i} = \{S_{i,j1}, S_{i,j2}, S_{i,j3}\}$, причём $j1, j2, j3 \in [1, q_i]$, $j1 \neq j2 \neq j3$;
- если $F_i = RB/1/1$, то $H_i^{F_i} = \{H_{i,k1}, H_{i,k2}\}$ $(k1, k2 \in [1, p_i])$, а $S_i^{F_i} = \{S_{i,j1}, S_{i,j2}\}$, причём $j1, j2 \in [1, q_i], j1 \neq j2$.

Конфигурация РВС называется корректной, если $\forall i \in [1, n]$ множество $\{H_i^{F_i}, S_i^{F_i}, F_i\}$ является корректным.

3.3. Оценка надёжности и стоимости РВС

Будем оценивать:

1. Надёжность РВС

$$R_{system} = \prod_{i=1}^{n} R_i,$$

где R_i – надёжность i-го модуля.

2. Надёжность модуля будем рассматривать в соответствии с формулами, предложенными в [13].

3. Стоимость аппаратных компонентов модуля РВС

$$C_i^{hw} = \sum_{j=1}^{p_i} x_{i,j} * C_{i,j}^{hw},$$

где p_i – количество версий аппаратного компонента H_i $\forall i \in [1,n]$; $x_{i,j}$ – количество экземпляров $H_{i,j}$ в модуле i;$C_{i,j}^{hw}$ – стоимость j-й версии аппаратной компоненты H_i в модуле i, $i \in [1,n], j \in [1,p_i]$.

4. Стоимость программных компонентов

$$C_i^{sw} = \sum_{j=1}^{q} y_{i,j} * C_{i,j}^{sw},$$

где q_i – количество версий программного компонента S_i $\forall i \in [1,n]$; $y_{i,j}$ – количество экземпляров $S_{i,j}$ в модуле i;$C_{i,j}^{sw}$ – стоимость j-й версии программной компоненты S_i в модуле i, $i \in [1,n], j \in [1,q_i]$.

5. Стоимость модуля вычислительной системы:

$$C_i = C_i^{hw} + C_i^{sw}.$$

6. Стоимость РВС:

$$C_{system} = \sum_{i=1}^{n} C_i = \sum_{i=1}^{n} \sum_{j=1}^{p_i} x_{i,j} * C_{i,j}^{hw} + \sum_{i=1}^{n} \sum_{j=1}^{q_i} y_{i,j} * C_{i,j}^{sw}.$$

Все значения стоимости являются натуральными величинами.

4. Задача выбора сбалансированного набора МОО для РВС

Задачу выбора сбалансированного набора МОО для РВС будем рассматривать в следующей постановке:
Дано:

1. n – количество модулей в РВС;

2. p_i,q_i – количество версий программных и аппаратных компонент для каждого модуля i, $i \in [1,n]$;

3. $C_{i,j}^{hw}$, $R_{i,j}^{hw}$ – стоимость и надёжность версий аппаратного компонента $\forall i \in [1,n], j \in [1,p_i]$;

4. $C_{i,j}^{sw}$, $R_{i,j}^{sw}$ – стоимость и надёжность версий программного компонента $\forall i \in [1,n], j \in [1,q_i]$;

5. FT_{avail} – множество доступных МОО для РВС.
   $FT_{avail} = \{None, NVP/0/1, NVP/1/1, RB/1/1\}$

6. $FT_i \subseteq FT_{avail} - \forall i \in [1, n]$ определён свой набор доступных МОО из множества FT_{avail};

7. $P_{v_{i,j}}, P_{rv}, P_d, P_{all}$ – вероятность отказа j-й версии i-го модуля, вероятность одновременного отказа нескольких версий программных компонентов, вероятность отказа схемы принятия решения, вероятность отказа сразу всех версий программного компонента;

8. C^{max}_{system} – максимально допустимая стоимость вычислительной системы.

Требуется определить:
Конфигурацию РВС $System_{RTES}$, такую что:

$$\begin{cases} R_{system_{RTES}} = \max\limits_{system} R_{system} \\ C_{system_{RTES}} \leq C^{max}_{system} \end{cases} \tag{1}$$

В работе [4] было показано, что эта задача является NP-трудной.

5. Эволюционный алгоритм с использованием схемы нечёткой логики

5.1. Общее описание метода

Метод, предлагаемый в данной работе для решения поставленной задачи (1), представляет собой ЭА с использованием схемы нечёткой логики, которая в процессе работы алгоритма анализирует результаты его работы в каждом поколении и, исходя из этой информации, корректирует параметры эволюционного алгоритма.

Предлагаемый метод решения задачи, поставленной в разделе 3, включает в себя следующие шаги:

1. **Кодирование решения.** Каждое решение задачи выбора сбалансированного набора МОО кодируется в виде строки. Строка состоит из блоков, которые соответствуют модулям РВС. Каждый блок (ген) представляет собой тройку вида $<H, S, F>$. H – это номер конфигурации аппаратной составляющей модуля, S – это номер конфигурации программной составляющей, а F – это порядковый номер МОО из множества FT_{avail}, используемого в данном модуле. По каждой строке может быть вычислена целевая функция – надёжность системы R_{system}, которая характеризует качество решения, и однозначно восстановлена соответствующая конфигурация РВС. Количество особей возьмём равным некоторому числу num.

2. **Подготовка популяции решений.** Генерация случайным образом популяции решений $System_k, k \in [1, num]$, для первой итерации алгоритма или популяция с предыдущей итерации запуска алгоритма.

3. **Начальная оценка популяции.** На этом этапе происходит вычисление целевой функции R_{system} и проверка ограничений стоимости $C_{system} \leq C_{system}^{max}$.

4. **Выполнение операции селекции.** В данном алгоритме используется пропорциональная схема селекции [21].

5. **Отбор особей для скрещивания в отдельную промежуточную популяцию.** Популяция сортируется, отбираются лучшие X% (изменяемый параметр) особей, наилучших по значению целевой функции, которые затем участвуют в операции скрещивания.

6. **Выполнение операции скрещивания.** В качестве операции скрещивания используется одноточечное скрещивание [21] модулей РВС. Скрещивание запускается столько раз, сколько особей в первоначальной популяции (num). В результате получается промежуточная популяция, которая состоит из num родителей и num потомков.

7. **Формирование новой популяции.** Промежуточная популяция из *2*num* особей сортируется по возрастанию целевой функции. В новую популяцию берётся доля (изменяемый параметр) лучших особей от исходной популяции, остальная часть популяции формируется из лучших особей промежуточной популяции, полученной после операции скрещивания. Таким образом, количество особей в популяции становится равным первоначальному значению.

8. **Выполнение операции мутации.** Некоторый процент лучших особей популяции (изменяемый параметр) не мутирует. Все остальные особи мутируют с некоторой степенью мутации (изменяемый параметр) и с некоторой вероятностью (изменяемый параметр). Оператор мутации является модификацией одноточечной мутации [21]. Он работает по следующему принципу: случайным образом выбирается ген (тройка $<H,\ S,\ F>$), соответствующий модулю системы. Пусть M_i – количество всевозможных вариантов этого гена для i-го модуля РВС. Далее случайным образом выбирается число n в диапазоне от *0* до $M_i - 1$. На это число n должен быть "увеличен" ген. Далее n раз случайным образом выбираем разряд в гене – либо H, либо S, либо F и увеличиваем соответствующий разряд на единицу, если это возможно. Если выпал разряд H и его уже нельзя увеличить, увеличиваем разряд S на единицу, а в разряде H устанавливаем минимальное значение (подобие операции переноса единицы в десятичной системе счисления). Если выбран разряд S, и его нельзя увеличить, увеличиваем разряд F на единицу, в разрядах H и S устанавливаем минимальные значения для данного F. Если выбран разряд F и его нельзя увеличить, все три разряда устанавливаем минимальные возможные значения. При такой операции может быть получен любой возможный ген, при этом некорректных генов получиться не может.

9. **Оценка популяции.** На этом этапе происходит проверка ограничений стоимости $C_{system} \leq C_{system}^{max}$ и вычисление целевой функции R_{system}. Если решение не удовлетворяет ограничениям, то оно штрафуется. Идея использования

штрафных функций заключается в том, чтобы искусственно понизить значение целевой функции (надежности) для тех решений, которые не удовлетворяют заданным ограничениям. Таким образом, решения, не удовлетворяющие ограничениям, будут взяты в следующую популяцию с меньшей вероятностью. Штрафная функция представляет собой коэффициент, на который умножается значение надёжности. Этот коэффициент должен быть равен 1, если решение удовлетворяет ограничениям, и должен принадлежать (0;1] иначе. Кроме того, логично выбирать его таким образом, чтобы большему значению стоимости соответствовало меньшее значение штрафного коэффициента. В соответствии с такими условиями был выбран следующий вид штрафных функций: функция равна 1, если решение удовлетворяет ограничениям, иначе – обратно пропорциональна стоимости.

Для стоимости:

$$E_{cost} = \begin{cases} 1 & \text{,если } C_{system} < C^{max}_{system} \\ \frac{C^{max}_{system}}{C_{system}} & \text{,если } C_{system} \geq C^{max}_{system} \end{cases}$$

$$R^{**}_{system} = R^{*}_{i} * E_{cost}$$

Также на этом шаге фиксируется лучшее, на текущий момент решения, вычисление среднего значения целевой функции.

10. **Проверка критерия останова.** Если он выполнен, то переход к п. 13, если нет, то переход к п. 12.

11. **Блок нечёткой логики.** Блок нечёткой логики осуществляет автоматическую подстройку параметров алгоритма и переход к п. 5. Подробно работа блока описаны в следующем разделе.

12. **Завершение алгоритма.** В качестве результата выбирается наилучшая из найденных конфигураций $System_{RTES}$.

Данный метод реализован в виде программного средства, сопряжённого с новой версией среды моделирования ДИАНА [19, 20].

5.2. Описание блока нечёткой логики

Блок нечёткой логики нужен для того, чтобы в автоматическом режиме корректировать настройки ЭА, управлять степенью влияния операций селекции, скрещивания и мутации на эволюционный процесс согласно некоторым правилам в зависимости от результатов работы алгоритма, которые получаются в каждом поколении. Правила нечёткой логики для АГЭА зависят непосредственно от значений этих параметров и сформулированы в таблице 1. Каждый из перечисленных в таблице параметров имеет три значения (большое, среднее и малое). Конкретное числовое значение определяется экспериментатором, исходя из параметров оптимизируемой системы.

Таблица 1. Правила работы схемы нечёткой логики

	$F_{av_2} < F_{av_1}$	$F_{av_2} \approx F_{av_1}$ (в пределах 3%)	$F_{av_2} > F_{av_1}$
$F_{max_2} < F_{max_1}$	Малый P_{mut} Большой N_{nmut} Малый S_{mut} Большой N_{sel} Большой P_{cross}	Малый P_{mut} Большой N_{nmut} Малый S_{mut} Большой N_{sel} Средний P_{cross}	Средний P_{mut} Средний N_{nmut} Средний S_{mut} Средний N_{sel} Малый P_{cross}
$F_{max_2} \approx F_{max_1}$ (в пределах 3%)	Средний P_{mut} Средний N_{nmut} Малый S_{mut} Средний N_{sel} Большой P_{cross}	Большой P_{mut} Малый N_{nmut} Средний S_{mut} Малый N_{sel} Средний P_{cross}	Большой P_{mut} Малый N_{nmut} Большой S_{mut} Малый N_{sel} Малый P_{cross}
F_{av_1} и F_{av_2} – среднее значение надёжности текущей и предыдущей популяции F_{max_1} и F_{max_2} – лучшее значение надёжности в текущей и предыдущей популяциях P_{mut} – вероятность мутации N_{nmut} – доля лучших особей текущей популяции, которые не мутируют S_{mut} – степень мутации гена N_{sel} – доля от популяции лучших особей, которые затем будут скрещиваться P_{cross} – вероятность скрещивания			

В зависимости от изменений среднего и лучшего значения надёжности популяции в процессе работы АГЭА происходит изменение значений параметров алгоритма в соответствии с таблицей 1.

Такая схема нечёткой логики имеет малую сложность по сравнению с вычислением целевых функций и проверки ограничений для всех особей популяции, но при этом позволяет производить автоматическую перенастройку ЭА в процессе его работы.

Теорема 1. *Предложенный алгоритм АГЭА обладает свойствами корректности и полноты.*

Доказательство. Идея доказательства данного утверждения состоит в рассмотрении всех операций алгоритма. По построению начальная популяция состоит из корректных решений. Во время работы алгоритма решения изменяются при помощи операций скрещивания и мутации. При скрещивании новое решение получается из старого заменой одного из модулей модулем из другого решения. Поскольку новых модулей не создаётся, то полученное решение корректно. При мутации изменяется один из модулей. По построению эта операция корректна. Полнота вытекает из того, что вероятность попадания корректного решения в следующую популяцию никогда не равна нулю. Для того, чтобы перебрать все решения, достаточно последовательно применять операцию мутации с параметром $M_i = 2$. □

Для сравнения предложенного метода и существующих алгоритмов было проведено экспериментальное исследование.

6. Экспериментальное исследование

6.1. Цель экспериментального исследования

В работе [13] был предложен классический ГА (КГА) для решения рассматриваемой задачи (1). Поэтому с ним и будем сравнивать предложенный АГЭА алгоритм. Сформулируем цели экспериментального исследования:

1. Сравнить эффективность работы АГЭА с классическим ГА (КГА), описанным в статье [13], по следующим критериям:
   - точность как отклонение значения целевой функции от оптимального значения;
   - сложность, вычислительные затраты на выполнение алгоритма (характеризуется количеством итераций);
   - стабильность работы алгоритма (стабильность значения целевой функции на полученном решении и вычислительных затрат на нахождение решения).
2. Исследовать зависимость работы АГЭА от числа элементов области решений, удовлетворяющих разным ограничениям на стоимость системы.

6.2. Методика проведения экспериментов

Для каждого запуска алгоритма фиксируются следующие параметры:

- конфигурация системы $System_{RTES}$, её надёжность $R_{system_{RTES}}$ и стоимость $C_{system_{RTES}}$;
- количество итераций алгоритма N;
- время работы алгоритма T_{alg}.

При заданных условиях эксперимента (значениях фиксированных параметров; распределениях параметров, значения которым присваиваются случайным образом), гипотеза H_0 формулируется следующим образом: «с вероятностью p_0 , значение $R_{system_{RTES}}$ находится в интервале $[R_{system_{min}}$; $R_{system_{max}}]$». В данной работе гипотеза H_0 рассматривалась для значения $p_0 = 0.9$. Определим количество запусков алгоритма для оценки его точности, сложности и стабильности при выбранном значении p_0. На основе метода проверки статистической гипотезы о числовом значении вероятности события в различных источниках даются разные оценки о минимальном количестве запусков алгоритма. В данной работе использована наиболее пессимистическая оценка [22], где объём выборки должен быть не менее 223 для $p_0 = 0.9$. Поэтому для исследования каждого алгоритма будем проводить серию из 225 экспериментов.

Для оценки надёжности по серии экспериментов для каждого алгоритма вычислим среднее значение надёжности, полученное по 225 запускам, отбросив крайние значения. Затем сравним его с оптимальным значением, полученным с помощью алгоритма полного перебора, и оценим отклонение от оптимального значения.

Таблица 2. Стоимость аппаратных компонентов (АК) в системе

Модуль	Версия 1	Версия 2	Версия 3
АК 1	30,0	10,0	10,0
АК 2	30,0	20,0	10,0
АК 3	20,0	30,0	10,0
АК 4	30,0	10,0	10,0
АК 5	30,0	20,0	30,0
АК 6	30,0	20,0	20,0

Для оценки сложности по серии экспериментов для каждого алгоритма вычислим среднее значение количества итераций, выполненных алгоритмом.

Для оценки стабильности работы алгоритма в каждой серии найдём минимальное и максимальное значения надёжности (за исключением отброшенных крайних значений) полученных конфигураций и определим интервал как разность этих значений.

Исследование зависимости работы алгоритмов от размеров области решений заключается в исследовании точности, сложности и стабильности алгоритмов при разных ограничениях на стоимость вычислительной системы.

6.3. Описание экспериментальной вычислительной системы

Экспериментальная вычислительная система имеет следующие параметры [13, 14]:

- количество модулей в системе: 6;
- количество версий аппаратных компонентов в каждом модуле: 3;
- количество версий программы в каждом модуле: 4;
- значения стоимости и надёжности версий программных и аппаратных компонентов для каждого модуля экспериментальной системы приведены в таблицах 2–5;
- набор доступных МОО: *None, NVP/0/1, NVP/1/1, RB/1/1*;
- ограничения на стоимость вычислительной системы: 180, 320, 460, нет ограничения (∞).

6.4. Результаты экспериментов

Результаты исследования алгоритма по точности и сравнение среднего значения надёжности с оптимальным решением приведены в таблице 6. В таблице 6 представлены средние значения надёжности для каждого алгоритма (кроме алгоритма

Таблица 3. Надёжность аппаратных компонентов (АК) в системе

Модуль	Версия 1	Версия 2	Версия 3
АК 1	0,995	0,980	0,980
АК 2	0,995	0,995	0,970
АК 3	0,994	0,995	0,992
АК 4	0,990	0,980	0,985
АК 5	0,995	0,980	0,995
АК 6	0,998	0,995	0,994

Таблица 4. Стоимость версий программных компонент (ПК) в системе

Модуль	Версия 1	Версия 2	Версия 3	Версия 4
ПК 1	30,0	10,0	20,0	30,0
ПК 2	30,0	20,0	10,0	20,0
ПК 3	20,0	30,0	20,0	30,0
ПК 4	20,0	10,0	20,0	20,0
ПК 5	30,0	20,0	30,0	30,0
ПК 6	10,0	30,0	20,0	20,0

Таблица 5. Надёжность версий программных компонент (ПК) в системе

Модуль	Версия 1	Версия 2	Версия 3	Версия 4
ПК 1	0,950	0,908	0,908	0,950
ПК 2	0,965	0,908	0,887	0,908
ПК 3	0,978	0,954	0,860	0,954
ПК 4	0,950	0,908	0,910	0,950
ПК 5	0,905	0,967	0,967	0,905
ПК 6	0,908	0,968	0,968	0,955

Таблица 6. Среднее значение надёжности ВС

Алгоритм	Ограничение на стоимость системы			
	180	320	460	∞
КГА	0,611542	0,824016	0,840537	0,821873
АГЭА	0,625433	0,874313	0,922647	0,9252
Полный Перебор	0,659682	0,891419	0,924541	0,925244

Таблица 7. Отклонение решения от оптимального

Алгоритм	Ограничение на стоимость системы			
	180	320	460	∞
КГА	7,30	5,71	8,40	11,17
АГЭА	5,19	1,92	0,02	0,005

полного перебора) по 225 запускам с разными ограничениями на стоимость вычислительной системы – 180, 320, 460 и без ограничения. В таблице 7 приведено отклонение среднего значения надёжности от оптимального в процентах. Необходимо отметить, что размер популяции в КГА и АГЭА был одинаковым.

Результаты исследования алгоритма по сложности приведены в таблице 8. В таблице указано среднее количество итераций алгоритмов при разных ограничениях на стоимость системы.

Результаты исследования алгоритмов по стабильности приведены в таблице 9. В таблице указано максимальное (max) и минимальное (min) значения надёжности для каждой серии экспериментов, а также длина доверительного интервала (len) при разных ограничениях на стоимость вычислительно системы.

На основании проведённого экспериментального исследования можно сделать следующие выводы:

- точность алгоритма АГЭА оказалась выше алгоритма КГА, а отклонение АГЭА от оптимального решения получилось не превосходящим 5%;

- на малой области приемлемых решений (удовлетворяющих ограничению на стоимость конфигурации системы) предпочтительным оказывается АГЭА, поскольку он имеет наименьшую сложность;

Таблица 8. Количество итераций алгоритмов

Алгоритм	Ограничение на стоимость системы			
	180	320	460	∞
КГА	909,2	109,7	23,6	21,3
АГЭА	75,4	76,29	82,95	97,27

Таблица 9. Стабильность алгоритмов

Алгоритм	Ограничение на стоимость системы					
	180			320		
	min	max	len	min	max	len
КГА	0,57314	0,625008	0,051868	0,748954	0,852204	0,10325
АГЭА	0,544715	0,667985	0,123269	0,807257	0,911711	0,104454
	460			∞		
	min	max	len	min	max	len
КГА	0,765034	0,883054	0,11802	0,712925	0,873224	0,160299
АГЭА	0,898389	0,925049	0,0266	0,924766	0,92544	0,000468

- на большой области приемлемых решений точность, стабильность и сложность алгоритма АГЭА лучше, чем у алгоритма КГА.

Таким образом, можно сделать вывод, что предложенный в статье метод работает не хуже существующего, а на малой области приемлемых решений лучше существующего.

7. Заключение

В рамках данной работы были получены следующие результаты:

- Дана математическая постановка задачи выбора сбалансированного набора МОО для РВС.
- Разработан новый метод решения задачи выбора сбалансированного набора МОО для РВС. Ключевыми особенностями разработанного метода является учёт ограничений последовательно, что позволяет обобщить данный метод для вычислительных систем с дополнительными ограничениями, а также использование разработанного алгоритма АГЭА для поиска решения при ограничениях только на стоимость системы. Предложенный метод реализован в виде программного средства.
- В рамках проведённого экспериментального исследования определена область эффективного применения АГЭА, а именно большая область приемлемых решений. В остальных случаях предложенный в статье метод работает не хуже предложенного в статье [13].

В рамках дальнейших исследований можно выделить следующие направления:

- Исследовать поддержку в популяции большего количества различных решений. Это ухудшит скорость сходимости эволюционного алгоритма. Зато позволит в случае нахождения серии решений, в которых задачи не удовлетворяют ограничениям на стоимость, быстрее получить решения, в которых задачи удовлетворяют ограничениям на стоимость.

- Исследовать возможность применения других адаптивных схем. Это позволит выбрать наилучшую схему адаптации для АГЭА.
- Оценить применимость методов многокритериальной оптимизации для нахождения компромисса между надёжностью и производительностью РВС. На данный момент все характеристики системы, на которые не влияет использование МОО, считаются фиксированными.
- Оценить применимость предлагаемого метода для решения задачи оптимизации надёжности для распределённых систем реального времени, где помимо ограничений на стоимость появляется ограничение на директивное время работы каждого модуля системы.

Список литературы / References

[1] Avižienis A., Laprie J.-C., Randell B., "Dependability and itsthreats: a taxonomy", *Building the Information Society*, **156** (2004), 91–120.

[2] Лупанов О. Б., "Об одном методе синтеза схем", *Изв. ВУЗов, Радиофизика*, **1**:1 (1958), 120–140; [Lupanov O. B., "Ob odnom metode sinteza schem", *Izv. VUZov,Radiophizika*, **1**:1 (1958), 120–140, (in Russian).]

[3] Kuo W.,Wan R., "Recent Advances in Optimal Reliability Allocation", *Handbook of Military Industrial Engineering by Badiru A., Thomas M.*, 2009, 1–24.

[4] Chern M.S., "On the computational complexity of reliability redundancy allocation in a series system", *Building the Information Society*, **11**:5 (1992), 309–315.

[5] Tillman F.A., Hwang C.L., Kuo W., "Optimization techniques for systems reliability with redundancy – A review", *IEEE Transactions on Reliability*, **R-26**:3 (1977), 148–155.

[6] Misra K.B., "On optimal reliability design: A review", *System Science*, **12** (1986), 5–30.

[7] Kuo W., Prasad V.R., "An annotated overview of system-reliability optimization", *IEEE Transactions on Reliability*, **49**:2 (2000), 176–187.

[8] Gen M., Yun Y.S., "Soft computing approach for reliability optimization: State-of-the-art survey", *Reliability Engineering & System Safety*, **91**:9 (2006), 1008–1026.

[9] Aleti A. et al., "Software architecture optimization methods: A systematic literature review", *IEEE Transactions on Software*, **39**:5 (2013), 658–683.

[10] Soltani R., "Reliability optimization of binary state non-repairable systems: A state of the art survey", *International Journal of Industrial Engineering Computations*, **5**:3 (2014), 339–364.

[11] Смелянский Р. Л., "Модель функционирования распределенных вычислительных систем", *Вестник Московского Университета*, **3** (1990), 3–21; [Smeliansky R. L., "Model funkcionirovaniya raspredelennyh vychislitelnyh sistem", *Vestnik Moskovskogo Universiteta*, **3** (1990), 3–21, (in Russian).]

[12] Wattanapongsakorn N., Levitan S.P., "Reliability optimization models for embedded systems with multiple applications", *IEEE Transactions on Reliability*, **53**:3 (2004), 406–416.

[13] Wattanapongsakorn N., Coit D.W., "Fault-tolerant embedded system design and optimization considering reliability estimation uncertainly", *Reliability Engineering & System Safety*, **92**:4 (2007), 395–407.

[14] Bakhmurov A.G. et al., "Method For Choosing An Effective Set Of Fault Tolerance Mechanisms For Real-Time Embedded Systems, Based On Simulation Modeling", *Problems of dependability and modelling*, 2011, 13–26.

[15] Laprie J.C., Coste A., "Dependability: A Unifying Concept for Reliable Computing", *Proceedings of the 12th Fault Tolerant Computing Symposium*, 1982, 18–21.

[16] Xie Z., Sun H., Saluja K., "Survey of Software Fault Tolerance Techniques", *University of Wisconsin-Madison, Department of Electrical and Computer Engineering*, 2006.

[17] Laprie J.C. et al., "Definition and analysis of hardware and software-fault-tolerant architectures", *IEEE Computer*, **23**:7 (1990), 39–51.

[18] Lyu M.R. (Editor-in-Chief), *Handbook of software reliability engineering*, McGraw-Hill: IEEE Computer Society Press, 1996.

[19] Bakhmurov A.G., Kapitonova A.P., Smeliansky R.L., "DYANA: An Environment for Embedded System Design and Analysis", *Proceedings of 5-th International Conference TACAS'99, Amsterdam, The Netherlands*, **1579**, 1999, 390–404.

[20] Бахмуров А.Г. и др., "Интегрированная среда для анализа и разработки встроенных вычислительных систем реального времени", *Программирование*, **5** (2013), 35–52; [Bakhmurov A. G., "Integrated environment for the analysis and design of distributed real-time embedded computing systems", *Programming and Computer Software*, **39**:5 (2013), 242–254, (in Russian).]

[21] Гладков Л. А., Курейчик В. В., Курейчик В. М., *Генетические алгоритмы*, ФИЗМАТЛИТ, 2006; [Gladkov L. A., Kureychik V. V., Kureychik V. M., *Geneticheskie algoritmy*, PHIZMATLIT, 2006, (in Russian).]

[22] Чистяков В. П., *Курс теории вероятностей*, Наука, 1987; [Chistyakov V. P., *Kurs teorii veroyatnostey*, Nauka, 1987, (in Russian).]

Volkanov D. Yu., "Method for Choosing a Balanced Set of Fault Tolerance Techniques for Distributed Computer Cystems", *Modeling and Analysis of Information Systems*, **23**:2 (2016), 119–136.

DOI: 10.18255/1818-1015-2016-2-119-136

Abstract. In the paper we consider a method for a reliability allocation problem (RAP) of distributed computer systems (DCS) under cost constraints. In this problem we maximize reliability of DCS under constraints of system cost. The article describes considered fault tolerance mechanisms. The mathematical formulation of RAP is provided. RAP is widely discussed in the literature. A detailed description of the method is ensured. The applied method is an evolutionary algorithm with an adaptive logic control procedure. The adaptive logic control procedure analyzes the results of evolutionary algorithm work in each generation and, based on this information, adjusts parameters. The key feature of the proposed method is the use of an adaptive hybrid genetic algorithm. The results of experiments with the implemented method are presented. This method was implemented as a pilot system which works in cooperation with DYANA simulation environment. Finally, future plans for the development of the presented method and tools are briefly described.

Keywords: dependability, fault tolerance techniques, genetic algorithm, computer systems, reliability, reliability allocation problem, reliability-redundancy allocation problem, evolutionary algorithm

On the authors:
Volkanov Dmitry Yurievich, orcid.org/0000-0001-9940-5822, assistant lecturer,
Lomonosov Moscow State University,
2nd Education Building, Faculty CMC, room 764, GSP-1, Leninskie Gory, Moscow, 119991, Russian Federation,
e-mail: volkanov@lvk.cs.msu.su

Acknowledgments:
This research is supported by the Ministry of education and science of the Russian Federation, Unique ID RFMEFI60714X0070, agreement No 14.607.21.0070 dated Sep 23, 2014.