УДК 004.052.2
ВАК 05.13.14
РИНЦ 50.41.01

**В.Л. Ясковский, И.А. Юскевич**

# Методика выбора параметров тестирования и алгоритмического резервирования для обеспечения требований к надежности программного обеспечения систем обработки данных реального времени

*Разработан метод оценки показателя надежности программного обеспечения для системы обработки данных реального времени при комплексном применении как методов тестирования, так и резервных алгоритмов. Приведена постановка задачи выбора параметров тестирования и алгоритмического резервирования для обеспечения требований к надежности программного обеспечения систем обработки данных реального времени как задачи смешанного (целочисленного и непрерывного) программирования с нелинейными ограничениями. Задача решена методами полного перебора и адаптивного сеточного поиска.*

***Ключевые слова:*** *надежность программного обеспечения, показатель надежности, тестирование программного обеспечения, алгоритмическое резервирование, целочисленное программирование, метод сеточного адаптивного поиска, метод полного перебора.*

## METHODS OF CHOICE THE SOFTWARE TESTING AND ALGORITHMIC REDUNDANCY PARAMETERS TO ACHIEVE RELIABILITY REQUIREMENTS FOR THE REAL-TIME PROCESSING SYSTEMS

*We propose methods for the evaluation software reliability parameter of the real-time processing systems at complex application software testing and algorithmic redundancy. A formulation of the problem of choice the software testing and algorithmic redundancy parameters for achieving reliability requirements for the real-time processing systems is given as a problem of the mixed (integer and continuous) programming with the nonlinear constraints. The problem is solved by brute-force and mesh adaptive direct search algorithms.*

***Keywords:*** *software reliability, reliability parameter, software testing, algorithmic redundancy, integer programming, mesh adaptive direct search, brute-force search.*

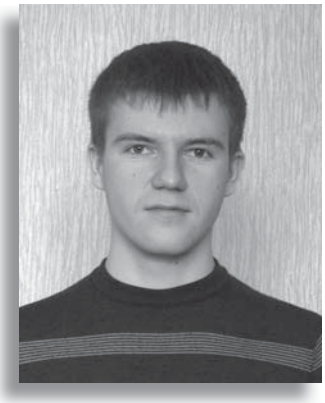
### Введение

Рассмотрен класс систем обработки данных реального времени (СОД РВ), ориентированных на циклическое решение заранее заданного перечня задач над векторами входных данных. СОД РВ применяются в комплексах обработки радиолокационной информации, робототехнических комплексах, системах управления воздушным движением, стационарных и бортовых комплексах военного назначения. В силу того, что СОД РВ решают крайне ответственные задачи, к программному обеспечению (ПО) таких систем предъявляются повышенные требования.

Одним из основных методов обеспечения требуемого уровня надежности СОД РВ, состоящей, в общем случае, из ненадежных подсистем и элементов, является применение избыточности различного вида: аппаратной, временной, информационной, алгоритмической (программной). Применение избыточности позволяет создавать высоконадежные (отказоустойчивые) СОД РВ за счет обнаружения и устранения последствий отказов и сбоев аппаратуры, искажений информации, программных ошибок. Однако применение избыточности различного вида приводит к необходимости увеличения затрат финансовых ресурсов и времени на проектирование СОД РВ. Поэтому возникает задача оптимизации по выбору эффективных средств, методов и способов избыточности, а

***Виктор Людвигович Лясковский,***
*д.т.н., профессор военной кафедры №1*
*Тел.: 8 (903) 800-89-64*
*Эл. почта: dop_big@mail.ru*
*Московский государственный технический университет им. Н.Э. Баумана*
*www.bmstu.ru*

***Viktor L. Lyaskovsky,***
*Doctor of Engineering Science, Professor, Chair of Military Science №1*
*Tel.: 8 (903) 800-89-64*
*E-mail: dop_big@mail.ru*
*Bauman Moscow State Technical University*
*www.bmstu.ru*



***Илья Александрович Юскевич,***
*студент кафедры «Радиоэлектронные системы и устройства»*
*Тел.: 8 (926) 787-12-26*
*Эл. почта: ilya.yuskevich@gmail.com*
*Московский государственный технический университет им. Н.Э. Баумана*
*www.bmstu.ru*

***Ilya A. Yuskevich,***
*student, Chair of Radioelectronic Systems & Devices*
*Tel.: 8 (926) 787-12-26*
*E-mail: ilya.yuskevich@gmail.com*
*Bauman Moscow State Technical University*
*www.bmstu.ru*

также алгоритмов их применения при существующих в процессе проектирования СОД РВ ограничениях.

В известной литературе [1–4] предложен ряд научно-методических подходов, формализованных постановок задач выбора средств и методов избыточности различного вида в условиях ограничений различного вида для этапов проектирования и эксплуатации отказоустойчивых СОД РВ, а также методов их решения. Однако в указанных и других известных авторам работах в явном виде не рассмотрена задача обеспечения требований к надежности ПО для СОД РВ на основе комплексного выбора параметров тестирования функциональных задач и их алгоритмического резервирования.

## 1. Постановка задачи

При проектировании средств СОД РВ особое внимание уделяется надежности встроенного ПО, так как в большинстве случаев сложность ПО СОД РВ превосходит сложность аппаратной части. В свою очередь, доля тестирования и сопровождения от общего числа затрат на ПО для сложных СОД РВ составляет более 75% [5]. В рамках данной статьи под надежностью ПО СОД РВ будем понимать вероятность безотказной работы программ для решения функциональных задач системы.

Для обеспечения требований к вероятности безотказной работы программ для решения функциональных задач СОД РВ целесообразно кроме тестирования применять алгоритмическое резервирование, идея которого заключается в сравнении результатов, получаемых от $N$ ненадежных программ, написанных, по возможности, независимыми командами разработчиков и (или) с использованием различных алгоритмов. При этом для определения правильного результата реализации функциональных задач в СОД РВ применяют различные варианты мажоритарной логики.

Рассматриваемый в настоящей работе алгоритм применения мажоритарной логики для нескольких реализованных программ заключается в следующем – на первом шаге исходные данные для решения функциональной задачи СОД РВ обрабатывает первая (основная) и вторая (резервная) программа, при несовпадении результатов по заранее заданному критерию либо превышении лимита времени решения (зацикливание) исходные данные функциональной задачи СОД РВ обрабатывает третья (вторая резервная) программа. Если все три программы привели к несовпадению результатов по заранее заданному критерию, подключается четвертая. Процесс продолжается до тех пор, пока результаты двух программ не совпадут либо не выполнятся (превысят лимит времени) все $N$ реализованных в СОД РВ программ рассматриваемой функциональной задачи.

Общая стоимость $j$-й программы будет складываться из затрат на её написание и тестирование. Стоимость системы с резервированием может быть оценена как простая сумма общей стоимости всех $N$ программ.

Очевидно, что проектировщик СОД РВ будет стремиться к минимизации затрат при ограничениях на надежность ПО и информационную загрузку элементов СОД РВ (что определятся необходимостью выполнения функциональных задач системы в реальном масштабе времени).

Рассмотрим сущность формализации задачи выбора параметров тестирования и алгоритмического резервирования для обеспечения требований к надежности ПО СОД РВ как задачи смешанного (целочисленного и непрерывного) программирования с нелинейными ограничениями.

Пусть разрабатывается СОД РВ, предназначенная для решения $M$ функций (функциональных задач). Для каждой $i$-й функции ($i = 1 \ldots M$) написано по одной программе, которые характеризуются надежностью прогона $R_{i,1}$ и временем прогона $\tau_{i,1}$ (под прогоном будем понимать однократное выполнение всех заложенных инструкций над единичным вектором исходных данных для определенной функции).

Будем называть эти программы основными. Каждая функция должна иметь заданную надежностью $R_i \geq R_{i_{\text{доп}}}$. В целях повышения надежности решения для каждой $i$-й функции могут быть написаны $v_i$ резервных программ ($v_i$ – целое неотрицательное число). Заметим, что в идеализированном случае $v_i \to \infty$. Практически $v_i$ не превосходит 10.

Каждая $j$-я резервная программа ($j = 2 \ldots (v_i + 1)$) характеризуется надежностью прогона – $R_{i,j}$, временем прогона – $\tau_{i,j}$, временем и стоимостью написания – $T_{i,j}^{(a)}$ и $c_{i,j}^{(a)}$. Кроме того, каждая программа (включая основную) может быть протестирована в течение промежутка времени $T_{i,j}^{(T)}$. Количество резервных алгоритмов функции будет определять кратность резервирования $K_{p,i} = N_i - 1$.

Кроме удовлетворения требований по надежности, при применении мажоритарной логики СОД РВ не должна перегружаться (т.е. загрузка СОД РВ должна быть не больше заданной: $\sum_{i=1}^{M} \lambda_i \cdot \overline{\tau_t(v_i+1)} < \rho_{\text{доп}}$, где $\lambda_i$ – интенсивность входного потока на выполнение $i$-й задачи; $\tau_i(v_i + 1)$ – среднее машинное время прогона. Также не должно быть превышено время $T_{\text{доп}}$, выделенное на разработку и тестирование ПО для СОД РВ.

Задача оптимизации заключается в нахождении булевой матрицы $\mathbf{z}^*$, каждый элемент которой ($z_{i,j}$) равен единице, если $j$-я резервная программа применяется при разработке ПО для реализации $i$-й функции, и нулю в обратном случае, а также действительной неотрицательной матрицы $\mathbf{T}^{(\mathbf{T})^*}$, описывающей время тестирования программ, входящих в ПО для СОД РВ. Для того чтобы решать задачу методами целочисленного программирования, удобно разбить время тестирования на $L$ этапов. В этом случае элементы матрицы $\mathbf{T}^{(\mathbf{T})^*}$ – неотрицательные целые числа, характеризующие количество этапов тестирования $j$-й программы, применяемой при реализации $i$-й функции.

Формализовано описанная задача оптимального выбора $z_{i,j}$, $T_{i,j}^{(T)}$ имеет вид:

$$< \mathbf{z}^*, \mathbf{T}^{(\mathbf{T})^*} >= \underset{z,T}{\text{Argmin}} \left( \sum_{i=1}^{M} \left( \sum_{j=2}^{v_i+1} c_{i,j}^{(a)} \cdot z_{i,j} + \sum_{j=1}^{v_i+1} c_{i,j}^{(T)} \left( T_{i,j}^{(T)} \right) \right) \right), \quad (1)$$

$$R_i(\mathbf{z}, \mathbf{T}^{(\mathbf{T})}) \geq R_{i\text{доп}};$$

$$\sum_{i=1}^{M} \lambda_1 \cdot \overline{\tau_t(\mathbf{z})} < \rho_{\text{доп}};$$

$$\sum_{i=1}^{M} \left( \sum_{j=1}^{v_i+1} T_{i,j}^{(T)} \cdot z_{i,j} + \sum_{j=2}^{v_i+1} T_{i,j}^{(a)} \cdot z_{i,j} \right) \leq T_{\text{доп}}.$$

Последнее ограничение относится к случаю, когда написание и тестирование программ осуществляется последовательно, одной командой разработчиков.

## 2. Оценка показателя надежности ПО для СОД РВ

*2.1. Описание принятой модели надежности ПО для СОД РВ.* На сегодняшний день существует множество моделей надежности ПО автоматизированных и информационных систем [5–7]. В настоящей работе будем использовать модель Нельсона, разработанную специалистами фирмы TRW [6].

Для каждой решаемой задачи рассматриваемая модель может быть представлена в виде функции $\mathcal{F}$, определенной на конечном векторном множестве $\mathbb{E}$, каждый элемент которого $E_e$, является полным набором входных данных, необходимых для одного прогона программы, реализующей заданную функцию.

Вследствие допущенных программистом ошибок существует подмножество, состоящее из наборов входных данных, приводящих к ошибке $m \subset \mathbb{E}$. В этом случае при исполнении программы результат $\mathcal{F}(m_e)$ либо отличается от спецификации $|S(E_e) - \mathcal{F}(m_e)| > \Delta_s$, либо не может быть получен в течение оговоренного промежутка времени (зацикливание).

Пусть в $\mathbb{E}$ содержится $E$ элементов, а в подмножестве $m$ – $M$ элементов. Тогда вероятность безотказной работы программы:

$$R = \frac{E - M}{E}.$$

По этой формуле можно эмпирически оценить надежность программы при её тестировании в качестве «черного ящика».

Если определять надежность программы как вероятность безотказной работы на $n$ прогонах, то её следует искать по формуле

$$R(n) = R^n. \quad (2)$$

В условиях реального функционирования программы вероятность выбора $E_e$ из $\mathbb{E}$ распределена неравномерно, т.е. можно говорить о плотности вероятности $p_e$, определенной на $\mathbb{E}$.

На начальных этапах проектирования СОД РВ можно принять гипотезу о том, что множество $\mathbb{E}$ разбивается на подмножества, содержащие в себе наборы входных данных, встречающиеся в процессе функционирования СОД РВ с равной вероятностью. Дальнейшее рассмотрение ведется для одного из таких подмножеств.

*2.2. Оценка прироста надежности ПО с алгоритмической избыточностью при реализации функций СОД РВ.* Примем гипотезу о том, что программы не содержат ни одной одинаковой ошибки (система при определенных входных данных выдает ошибочный и идентичный результат для двух и более алгоритмов), который может быть принят за истинный результат. Вероятность такого события при независимой разработке программ для решения функциональных задач в СОД РВ представляется незначительной.

Тогда, в случае если результат в 2 из $N$ программ совпал по критерию:

$$(|\mathcal{F}_k(E_e) - \mathcal{F}_l(E_e)| < \Delta_{s_{k,l}}) \wedge \wedge (t_k^{(i)} \leq t_{\text{доп}}^{(i)}). \quad (3)$$

будем считать, что истинное решение найдено ($k, l = \{1 \ldots N\}$).

Так как в общем случае каждый алгоритм имеет собственную точность, можно задать вектор среднеквадратических отклонений результатов $\sigma_j$ при работе $j$-го алгоритма. Следуя правилу трех сигм, предложим один из способов задания матрицы $\Delta_{s_{k,l}}$:

$$\Delta_{s_{k,l}} = 3 \cdot \sigma_k + 3 \cdot \sigma_l.$$

В простейшем случае при выполнении условия (3) за истинное решение может быть принят ре-

зультат программы с наименьшим $\sigma_j$ (как правило, это условие реализуется для основного алгоритма). При возникновении программной ошибки в основном алгоритме при выполнении условия (3) для резервных алгоритмов в качестве оценки результата можно воспользоваться формулой

$$x_{k,l} = \frac{\sigma_l}{\sigma_l + \sigma_k} \cdot x_k + \frac{\sigma_k}{\sigma_k + \sigma_l} \cdot x_l,$$

где $x_{k,l}$ – объединенная оценка искомого параметра;

$x_k, x_l$ – оценка искомого параметра $k$-м и $l$-м алгоритмами.

Если бы множества данных $m_j$, ведущих к ошибке в результате работы $j$-го алгоритма, не пересекались, то общая надежность ПО, благодаря алгоритмическому резервированию, становилась бы равной единице, поэтому интерес представляют именно области пересечения этих множеств.

Диаграммы Эйлера для множеств входных данных, влекущих за собой ошибки при использовании двух-, трех- и четырехкратного резервирования, приведены на рис. 1. Процесс допущения ошибок при программировании $j$-го алгоритма можно представить как многомерную случайную величину выделения подмножества $m_j$ из $\mathbb{E}$. Каждый круг изображает реализацию этой случайной величины. Закрашенные области характеризуют те подмножества данных $m_j$, для которых ошибки в ПО для СОД РВ принципиально не могут быть исправлены при использовании описанной выше логики. Формализовано запишем эти множества в следующем виде (для $N = 3, 4, 5$):

$$N = 3: (m_1 \cap m_2) \cup (m_1 \cap m_3) \cup \cup (m_2 \cap m_3)$$

$$N = 4: (m_1 \cap m_2 \cap m_3) \cup \cup (m_1 \cap m_2 \cap m_4) \cup (m_1 \cap m_3 \cap m_4) \cup \cup (m_2 \cap m_3 \cap m_4)$$

$$N = 5: (m_1 \cap m_2 \cap m_3 \cap m_4) \cup \cup (m_1 \cap m_2 \cap m_3 \cap m_5) \cup \cup (m_1 \cap m_3 \cap m_4 \cap m_5) \cup \cup (m_1 \cap m_2 \cap m_4 \cap m_5) \cup \cup (m_2 \cap m_3 \cap m_4 \cap m_5).$$

Тогда можно найти надежность одного прогона при решении функциональной задачи, для которой реализован один основной и два резервных алгоритма:

$$R_3 = 1 - (P_1 \cdot P_2 + P_1 \cdot P_3 + P_2 \cdot P_3 - - 2 \cdot P_1 \cdot P_2 \cdot P_3),$$

а системы с резервированием, состоящей из $N$ алгоритмов:

$$R_N = 1 - \left( \sum_{k=1}^{N} \prod_{l=1}^{N} \frac{P_l}{P_k} - (N-1) \cdot \prod_{l=1}^{N} P_l \right),$$

где $P_j = 1 - R_j$ – вероятность ошибки при одном прогоне $j$-й программы.

Надежность на $n$ прогонах, соответственно, находят по формуле (2).

Принимая во внимание степенной характер зависимости (2), можно ожидать существенный рост надежности на $n$ прогонах. Исследуем пять программ со случайно выбранной надежностью порядка 0,99. Составив ПО для реализации функциональной задачи в СОД РВ с кратностью резервирования $K_p = 2,3,4$, можно наблюдать существенный рост надежности на $n$ прогонах (рис. 2).

*2.3. Оценка среднего времени работы реализации функции с алгоритмической избыточностью.* Пусть каждая программа затрачивает некоторое количество машинного времени (время прогона программы), т.е. задано среднее время реализации $j$-го алгоритма – $t_j$. Если в результате прогона основной и первой резервной программы выполнилось условие (3), то время прогона системы $t_{min} = t_1 + t_2$. С вероятностью $P = = P_1 + P_2 - P_1 \cdot P_2$ результаты двух программ не совпадут. Для этого

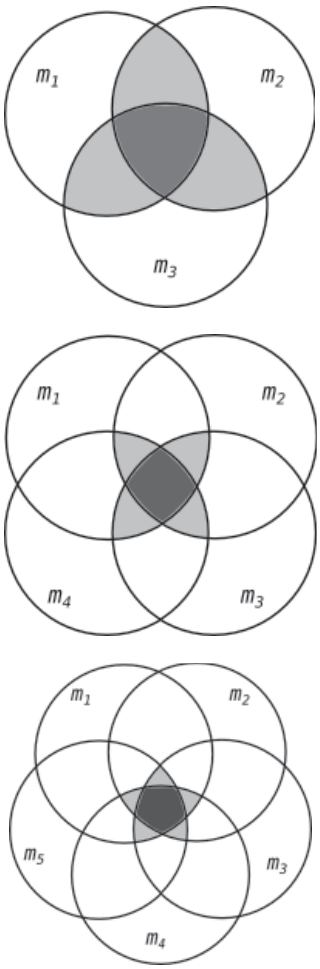



**Рис. 1.** *Геометрическая интерпретация множеств данных, влекущих за собой ошибки, при использовании нескольких алгоритмов при реализации функциональной задачи в СОД РВ*

случая потребуется включение еще одной, резервной, программы, которая увеличит время прогона системы на $t_3$. Далее, с вероятностью $P = P_1 \cdot P_2 + P_1 \cdot P_3 + P_2 \cdot P_3 - 2 \cdot P_1 \cdot \cdot P_2 \cdot P_3$, требуемое машинное время увеличится на $t_4$ и т. д.

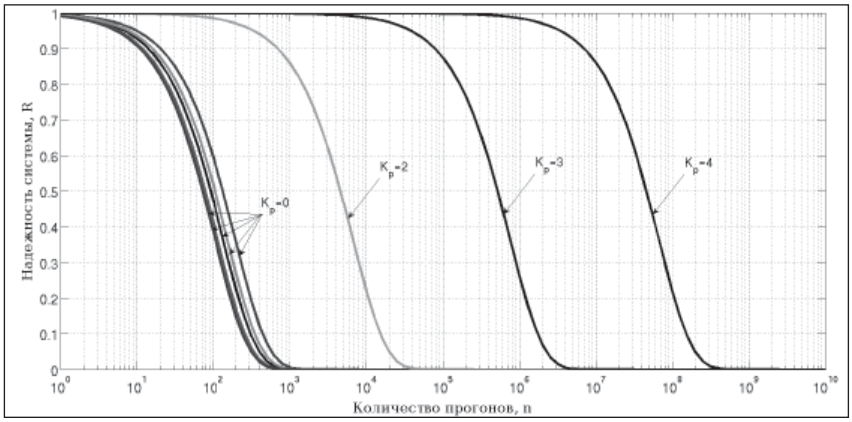



**Рис. 2.** *График зависимости надежности системы программ с разной кратностью резервирования от количества прогонов*

Следуя принципу математической индукции, можно вывести среднее время прогона системы программ в соответствии с принятой в работе логикой алгоритмического резервирования:

$$\overline{t(N)} = t_1 + t_2 + \\ + \sum_{s=3}^{N} t_s \left( \sum_{k=1}^{s-1} \prod_{l=1}^{N} \frac{P_l}{P_k} - (s-2) \cdot \prod_{l=1}^{s-1} P_l \right).$$

*2.4. Оценка параметров тестирования ПО СОД РВ.* Существует множество стратегий и соответствующих им методов тестирования ПО автоматизированных и информационных систем [5–7], каждая из которых имеет свою экономическую модель (зависимость показателя надежности от затрат на тестирование). Майерс в своей монографии [5, с. 177] говорит о спектре стратегий тестирования с двумя крайними подходами – тестирование по отношению к спецификациям и тестирование по отношению к тексту программы. Для оценки изменения параметров надежности ПО СОД РВ выберем подход к тестированию, основанный на оценке соответствия спецификациям (подход «черного ящика»).

Начальная надежность $R_0$ может быть оценена по формуле (2) в процессе поиска ошибок при написании и предварительном тестировании ПО.

Предположим, что исправленная ошибка не породила новой ошибки. Это значит, что подмножество $m \subset \mathbb{E}$ теперь имеет меньший набор элементов. Вероятность найти новую ошибку падает $P_T = \frac{M - M'}{E}$, поэтому со временем эффективность тестирования снижается. Для того чтобы найти среднее количество тестов $x$, необходимое для отыскания ошибки, нужно решить уравнение $P_T \cdot \left( x + \frac{1}{P_0} \right) = 1$, где $P_0 = 1 - R_0$. Из формулы видно, что процесс поиска ошибок в ПО имеет гиперболический характер. Теперь, указав стоимость теста $C_x$, можно перейти к переменной затрат $c = x \cdot C_x$.

$$R(x) = 1 - \frac{1}{x + \frac{1}{P_0}};$$

$$R(c) = 1 - \frac{1}{\frac{c}{C_x} + \frac{1}{1 - R_0}},$$

где $R(x)$ – зависимость надежности программы от количества элементарных тестов;

$R(c)$ – зависимость надежности программы от затрат на тестирование.

## 3. Метод решения задачи

Так как количество этапов тестирования $L$ может быть любым, размерность матрицы не будет адекватно отражать трудоемкость решения задачи. Вместо размерности будем говорить о размере области определения целевой функции:

$$w = 2^{M \cdot N} \cdot (L+1)^{M \cdot N}.$$

Задача (1) решена двумя методами. Первый – метод полного перебора (ПП) – был реализован авторами. Второй – метод сеточного адаптивного поиска (Mesh Adaptive Direct Search, MADS), реализованный в пакете NOMAD, распространяемом по свободной лицензии (GNU LGPL). Постановка задачи и её решение проводилось в среде MATLAB.

Проведем сравнение эффективности метода MADS с методом ПП. Превышение значения целевой функции $C_{MADS}$ для решения, найденного методом MADS над уровнем глобального экстремума $C_{BF}$ (решение, найденное методом полного перебора) в процентах, будем называть ошибкой метода:

$$\delta_{MADS} = \frac{C_{MADS} - C_{BF}}{C_{BF}} \cdot 100\%.$$

Задача оценки точности эвристического метода нетривиальна. Для каждой конкретной постановки задачи ошибка метода будет различной. Пусть, например, имеется несколько алгоритмов для решения ряда функциональных задач. В случае если эти алгоритмы имеют существенный разброс по стоимости написания и тестирования,

*Таблица 1*

**Ошибка эвристического метода**

| [M;N;L] | $w$ | Начальная надежность программ | Требуемая надежность функциональных задач | Стоимость написания программы и этапа тестир. | Поиск по методу гиперкуба | Среднее время решения, с | Ошибка (в ср.), % |
|---|---|---|---|---|---|---|---|
| [2;3;2] | 46656 | $0{,}8 < x < 0{,}9$ | $0{,}999 < x < 0{,}9999$ | $375 < x < 425$ | – | 6,80 | 13,3 |
| [2;3;2] | 46656 | $0{,}8 < x < 0{,}9$ | $0{,}999 < x < 0{,}9999$ | $375 < x < 425$ | + | 10,6 | 8,8 |
| [2;4;2] | 1679616 | $0{,}8 < x < 0{,}9$ | $0{,}9999 < x < 0{,}99999$ | $375 < x < 425$ | – | 19,2 | 12,7 |
| [2;4;2] | 1679616 | $0{,}8 < x < 0{,}9$ | $0{,}9999 < x < 0{,}99999$ | $375 < x < 425$ | + | 20,0 | 6,4 |

*Таблица 2*

**Ошибка эвристического метода при большей дисперсии**

| [M;N;L] | $w$ | Начальная надежность программ | Требуемая надежность функциональных задач | Стоимость написания программы и этапа тестир. | Поиск по методу гиперкуба | Среднее время решения, с | Ошибка (в ср.), % |
|---|---|---|---|---|---|---|---|
| [2;3;2] | 46656 | $0{,}7 < x < 0{,}9$ | $0{,}998 < x < 0{,}99999$ | $300 < x < 600$ | – | 7,4 | 17,8 |
| [2;3;2] | 46656 | $0{,}7 < x < 0{,}9$ | $0{,}998 < x < 0{,}99999$ | $300 < x < 600$ | + | 10,8 | 11 |
| [2;4;2] | 1679616 | $0{,}7 < x < 0{,}9$ | $0{,}998 < x < 0{,}99999$ | $300 < x < 600$ | – | 23,3 | 24 |
| [2;4;2] | 1679616 | $0{,}7 < x < 0{,}9$ | $0{,}998 < x < 0{,}99999$ | $300 < x < 600$ | + | 24,3 | 16,7 |

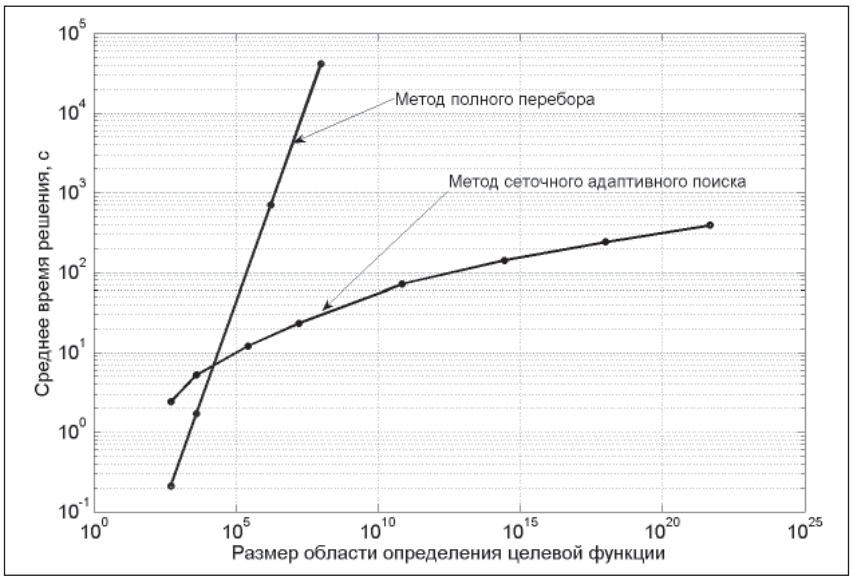



**Рис. 3.** *Скорость решения задачи двумя методами*

начальной надежности, а функциональные задачи имеют существенно различающуюся требуемую надежность, то целевая функция «зашумлена» и ограничена реализацией случайного процесса. В пределе целевая функция полностью случайна, в таких условиях единственный приемлемый метод поиска глобального экстремума – полный перебор.

Сказанное иллюстрируют табл. 1 и 2, в которых приведено среднее время и средняя ошибка решения методом адаптивного поиска для задач со случайными исходными данными (равномерное распределение в указанных интервалах). Для каждого варианта задача решалась по 100 раз методами ПП и MADS. Из проведенных вычислений следует простой вывод – при увеличении дисперсии случайного задания начальных условий ошибка эвристического метода растет.

В MADS возможны 16 алгоритмов выбора пробных точек [8]. Мы использовали метод Generalized Pattern Search (GPS) с 2*n* пробными точками на каждой итерации (где *n* – размерность целевой функции).

Также GPS может быть на каждой итерации дополнен поиском начальной точки по методу латинского гиперкуба, lh-search (мы взяли по 100 дополнительных точек на каждую итерацию). Как показывает наше исследование, такой поиск чаще всего оправдан, поскольку он позволяет существенно снизить вероятность ошибки.

Конфигурация компьютера, на котором производился расчет, – Core i5-2500 CPU 3,30 ГГц, 16 Гб ОЗУ.

Оценка зависимости скорости решения задачи методом MADS от размера области определения целевой функции была выполнена для восьми точек по 100 измерений на каждую, для метода ПП такая оценка была произведена на малых размерностях экспериментально, на больших размерностях – аналитически (рис. 3).

Следуя графику, изображенному на рис. 3, метод полного перебора однозначно эффективнее метода MADS при малых размерностях. При увеличении размерности предпочтение тому или иному методу следует отдавать в зависимости от соотношения цены поиска точного решения к цене ошибки.

## Заключение

Разработанный метод оценки показателя надежности ПО СОД РВ, предложенное формализованное описание задачи выбора параметров тестирования и алгоритмического резервирования, а также рекомендованные методы решения этой задачи позволяют автоматизировать процесс проектирования высоконадежных СОД РВ, приблизить результат проектирования к оптимальному.

Дальнейшие исследования должны проходить в направлении приближения математической модели СОД РВ от идеальной к реальной. Могут быть рассмотрены многомашинные вычислительные комплексы, где в качестве источников ошибок необходимо включать также аппаратные сбои и отказы. В этом случае нужно также рассматривать другие показатели надежности, например коэффициент готовности.

В рамках модели Нельсона могут быть разработаны оптимальные алгоритмы загрузки многомашинных комплексов в условиях изменения интенсивности потока входных данных.

Наконец, необходимо рассмотреть вопросы неравновероятного появления одних и тех же входных данных в процессе функционирования СОД РВ.

## Литература

1. *Коваленко К.А., Лясковский В.Л., Прохоров А.Г.* К вопросу повышения надежности функционирования многомашинных вычислительных комплексов с использованием аппаратурных средств и программно-логических методов // Автоматика и телемеханика. – 1997. – № 3. – С. 226–233.

2. *Лясковский В.Л., Кабардинский А.Ю., Догадов А.А. и др*. Метод обеспечения функциональной надежности комплексов средств автоматизации на основе применения избыточности различного вида // Программные продукты и системы. – 2013. – № 2 (102). – С. 38–41.

3. *Корзун А.Е., Лясковский В.Л., Неплюев Р.Н.* Организация отказоустойчивого функционирования комплексов средств автоматизации системы управления РЭС // Радиотехника. – 2010. – № 11. – С. 60–63.

4. *Лясковский В.Л., Догадов А.А., Колесниченко Р.В*. Методический подход к обеспечению требуемой эффективности реализации функциональных процессов в автономных робототехнических комплексах на основе

применения избыточности различного вида // В сборнике материалов V научно-практической конференции «Технические и технологические системы», Краснодар, 2013 г. – С. 178–180.
5. *Майерс Г*. Надежность программного обеспечения. – М.: Мир, 1980. – 356 с.
6. Надежность программного обеспечения. Анализ крупномасштабных разработок / Т. Тейер, М. Липов, Э. Нельсон. – М.: Мир, 1981. – 323 с.
7. Структурное проектирование надежных программ встроенных ЭВМ / А.А. Штрик, Л.Г. Осовецкий, И.Г. Мессих. – Л.: Машиностроение. Ленингр. отделение, 1989. – 269 с.
8. *Audet C., Digabel S. Le, and Tribes C*. NOMAD user guide. Technical Report G-2009-37, Les cahiers du GERAD, 2009.
9. *Лясковский В.Л., Юскевич И.А.* Методический подход к оценке и обоснованию параметров алгоритмической избыточности в системах обработки данных специального назначения // В сборнике материалов V научно-практической конференции «Технические и технологические системы», Краснодар, 2013 г. – С. 181–184.