УДК 519.6

ОПТИМИЗАЦИЯ ПРОГРАММНОЙ АРХИТЕКТУРЫ НА ОСНОВЕ ГЕНЕТИЧЕСКОГО АЛГОРИТМА С АЛЛЕЛЯМИ В ШКАЛЕ ПОРЯДКА

Д. А. Шеенок, В. А. Терсков

Красноярский институт железнодорожного транспорта

Поступила в редакцию 18.07.2013 г.

Аннотация. Рассмотрена задача оптимизации программной архитектуры. Описан генетический алгоритм и его параметры для решения поставленной задачи. Проанализированны результаты работы алгоритма с различными параметрами на тестовой задаче.
Ключевые слова: генетический алгоритм, программная архитектура, многокритериальная оптимизация, мультиверсионность.
Annotation. Consider the problem of optimization of the software architecture. Genetic algorithm is described and its parameters to solve the task. In analyzing the results of the algorithm with different parameters on the test task.
Keywords: genetic algorithm, software architecture, multi-criteria optimization, multiversion.

ВВЕДЕНИЕ

Задачи оптимизации постоянно возникают в деятельности человека, в частности при проектировании сложных технических систем. Если существует возможность выбора параметров такой системы, то их следует выбрать оптимальным, с точки зрения цели функционирования системы образом. Классические методы оптимизации накладывают существенные ограничения на целевую функцию задачи оптимизации: выпуклость, аналитическое задание функции и возможность вычисления вектора градиента в любой допустимой точке. Однако, многие возникающие задачи оптимизации характеризуются такими свойствами, как алгоритмическое задание целевой функции, наличие большого количества локальных экстремумов, большая размерность и различный характер параметров.

Важным классом методов, способных решать такие задачи оптимизации являются эволюционные алгоритмы и, в частности, генетические алгоритмы (ГА), основанные на имитации эволюционных процессов, происходящих в природе [1, с. 32].

1. ПОСТАНОВКА ЗАДАЧИ

Основной задачей при проектировании, разработке или модернизации сложных программных систем, как и любых других сложных объектов, для разработки которых привлекаются ресурсы разного характера и объема, является создание соответствующего объекта с заданным уровнем качества в условиях ограниченности ресурсов [2, с. 5].

Для сложных информационных систем, используемых в критических областях (технологические процессы, управление движением, финансовые операции) возникает задача построения такой архитектуры программного обеспечения (ПО), чтобы оно обладало заданной надежностью и требовало на реализацию минимальных трудозатрат.

Компоненты архитектуры программного обеспечения могут быть реализованы с использованием программной избыточности. Если вводится программная избыточность, то ее можно реализовать методом мультиверсионного программирования (*NVP – N-version programming*) или блока восстановления (*RB – recovery block*).

Программный компонент или каждая его версия, в случае применения программной избыточности, могут быть доведены до определенного уровня надежности (вероятности сбоя) путем тестирования. С помощью статистических моделей надежности или экспертных оценок, могут быть определены варианты возможного уровня надежности компонента (или его версии) и соответствующих трудозатрат на достижение этого уровня надежности.

Таким образом, возникает задача оптимизации характеристик определенных компонентов архитектуры ПО. Задача сводится к поиску максимума функции коэффициента готовности проектируемой системы и минимума функции ресурсов, затрачиваемые на разработку или модификацию программной системы:

$$S \to \max,\ T_s \to \min,$$

где S – критерий оценки коэффициента готовности системы, T_s – критерий оценки трудоемкости разработки системы.

Критерии оптимизации заданы алгоритмически, согласно усовершенствованной модели надежности программного обеспечения. Основные обозначения модели надежности ПО следующие [3]:

1) M – число архитектурных уровней в архитектуре ПО;

2) N_j – число компонентов на уровне j, $j \in \{1,..,M\}$;

3) D_{ij} – множество индексов компонентов, зависящих от компонента i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$;

4) F_{ij} – событие сбоя, произошедшего в компоненте i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$;

5) PU_{ij} – вероятность использования компонента i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$;

6) PF_{ij} – вероятность появления сбоя в компоненте i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$;

7) PL^{ij}_{nm} – условная вероятность появления сбоя в компоненте m на уровне n при появлении сбоя в компоненте i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$, $n \in \{1,..,N_m\}$, $m \in \{1,..,M\}$;

8) TA_{ij} – относительное время доступа к компоненту i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$, определяемое как отношение среднего времени доступа к компоненту i на уровне j к числу сбойных компонентов на малых уровнях архитектуры за одно и тоже время;

9) TC_{ij} – относительное время анализа сбоя в компоненте i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$, определяемое как отношение среднего времени анализа сбоя в компоненте i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$, к числу сбойных компонентов на всех уровнях архитектуры, анализируемых в одно и тоже время;

10) TE_{ij} – относительное время устранения сбоя в компоненте i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$, определяемое как отношение среднего времени восстановления в компоненте i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$, к числу сбойных компонентов на всех уровнях архитектуры, в которых происходит устранение сбоев в одно и тоже время;

11) TU_{ij} – относительное время использования компонента i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$, определяемое как отношение среднего времени использования компонента i на уровне j, $i \in \{1,..,N_j\}$, $j \in \{1,..,M\}$, к числу компонентов на всех уровнях архитектуры, используемых в одно и тоже время;

12) Z_{ij} – множество версий компонента i, на уровне j, $k = 1,..,K$;

13) T_{ij} – трудоемкость разработки компонента i на уровне j;

14) T^k_{ij} – трудоемкость разработки версии k компонента i на уровне j, $k \in Z_{ij}$ в чел-часах;

15) NVX_{ij} – трудоемкость разработки среды исполнения версий (приемочного теста для RB или алгоритма голосования для NVP);

16) B_{ij} – дихотомическая переменная, принимающая значение 1 (тогда $NVP_{ij} = 0, RB_{ij} = 0$), если в программном компоненте не используется программная избыточность, иначе равна 0.

17) NVP_{ij} – дихотомическая переменная, принимающая значение 1 (тогда $B_{ij} = 0, RB_{ij} = 0$), если в программном компоненте используется программная избыточность по методу N-версионного программирования, иначе равна 0.

18) RB_{ij} – дихотомическая переменная, принимающая значение 1 (тогда $B_{ij} = 0, NVP_{ij} = 0$), если в программном компоненте используется программная избыточность по методу блока восстановления, иначе равна 0.

19) TR – среднее время простоя системы в большой архитектуре телекоммуникационного ПО реального времени, определяемое как время, в течение которого система не может выполнять свои функции;

20) $MTTF$ – среднее время появления сбоя в большой архитектуре телекоммуникационного ПО реального времени, определяемое как время, в течение которого сбоев в системе не происходит;

21) S – коэффициент готовности;

22) T_s – общая трудоемкость системы;

При построении мультиверсионного компонента из K версий методом N-версионного программирования (*NVP, N-version programming*) для любого K надежность равна [4]:

$$R_{ij} = p^v_{ij}\left(1 - \prod_{k \in Z_{ij}}^{K} (1 - p^k_{ij})\right),$$

где p^v_{ij} – вероятность безотказной работы алгоритма голосования, p^k_{ij} – вероятность безотказной работы версии $k \in Z_{ij}$.

При построении мультиверсионного компонента из K версий методом блока восстановления (*RB, recovery block*) [4]:

$$R_{ij} = \sum_{k \in Z_{ij}}^{K} p^k_{ij} p^{AT}_{ij} \times \times \prod_{l=1}^{k-1} ((1 - p^l_{ij}) p^{AT}_{ij} + + p^l_{ij} (1 - p^{AT}_{ij})),$$

где p^{AT}_{ij} – вероятность безотказной работы приемочного теста для компонента i, $i = 1,..,N$ на уровне $j, j = 1,..,M$; p^k_{ij} – вероятность безотказной работы версии $k \in Z_{ij}$.

Вероятность сбоя таких компонентов рассчитывается как $PF_{ij} = 1 - R_{ij}$.

Трудоемкость разработки системы рассчитывается следующим образом:

$$T_s = \sum_{j=1}^{j=M} \sum_{i=1}^{i=N_j} \left(B_{ij} T_{ij} + + \left(NVP_{ij} + RB_{ij} \right) \times \times \left(NVX_{ij} + \sum_{k \in Z_{ij}}^{K} T^k_{ij} \right) \right).$$

Среднее время сбоя равно [3]:

$$MTTF = = \sum_{j=1}^{j=M} \sum_{i=1}^{i=N_j} \{ PU_{ij} \times (1 - PF_{ij}) \times [TU_{ij} + + \sum_{(m=1)\&(m \neq j)}^{m=M} \sum_{n=1}^{n=N_m} \left((1 - PL^{ij}_{nm}) \times \times \left(TU_{nm} + \sum_{l \in D_{nm}} \left((1 - PL^{nm}_{lm}) \times TU_{lm} \right) \right) \right) + + \sum_{k \in D_{ij}} \left((1 - PL^{ij}_{kj}) \times \times \left(TU_{kj} + \sum_{(m=1)\&(m \neq j)}^{m=M} \sum_{n=1}^{n=N_m} \left((1 - PL^{kj}_{nm}) \right) \times \times \left(TU_{nm} + \sum_{l \in D_{nm}} \left((1 - PL^{nm}_{lm}) \times TU_{lm} \right) \right) \right) \right)] \}.$$

Среднее время простоя системы равно [3]:

$$TR = \sum_{j=1}^{j=M} \sum_{i=1}^{i=M_j} \{ PU_{ij} \times PF_{ij} \times \times [(TA_{ij} + TC_{ij} + TE_{ij}) + + \sum_{(m=1)\&(m \neq j)}^{m=M} \sum_{n=1}^{n=N_m} \left(PL^{ij}_{nm} \times \times \left((TA_{nm} + TC_{nm} + TE_{nm}) + + \sum_{l \in D_{nm}} \left(PL^{nm}_{lm} \times (TA_{lm} + TC_{lm} + TE_{lm}) \right) \right) \right) + \sum_{k \in D_{ij}} [PL^{ij}_{kj} \times [(TA_{kj} + TC_{kj} + TE_{kj}) + \sum_{(m=1)\&(m \neq j)}^{m=M} \sum_{n=1}^{n=N_m} \left(PL^{kj}_{nm} \left(\left(TA_{nm} + TC_{nm} + TE_{nm} \right) + + \sum_{l \in D_{nm}} \left(PL^{nm}_{lm} \times (TA_{lm} + TC_{lm} + TE_{lm}) \right) \right) \right)]]]] \}$$

Коэффициент готовности равен:

$$S = \frac{MTTF}{MTTF + TR}$$

Проведенный анализ мощности пространства оптимизации позволяет сделать вывод о том, что для решения задачи необходимо применение генетического алгоритма (ГА), т.к. задача определения оптимальной архитектуры ПО является *NP*-полной проблемой и перебор всех вариантов решения не представляется возможным [5].

2. ПРЕДСТАВЛЕНИЕ ЗАДАЧИ ДЛЯ РЕШЕНИЯ ГЕНЕТИЧЕСКИМ АЛГОРИТМОМ

Генетический алгоритм представляет собой метод оптимизации, основанный на концепциях естественного отбора и генетики. В этом подходе переменные, характеризующие решение, представлены в виде ген в хромосоме. Генетический алгоритм оперирует конечным множеством решений (популяцией) – генерирует новые решения как различные комбинации частей решений популяции, используя такие операторы, как отбор, рекомбинация (кроссинговер) и мутация. Новые решения позиционируются в популяции в соответствии с их положением на поверхности исследуемой функции. Генетические алгоритмы являются универсальным вычислительным средством для решения серьезных математических задач.

Для части компонентов архитектуры, участвующих в ГА и для которых возможно введение программной избыточности, могут быть изменены следующие характеристики:

1. Метод реализации программной избыточности: мультиверсионное программирование ($NVP_{ij} = 1$, $RB_{ij} = 0$) или блок восстановления ($NVP_{ij} = 0$, $RB_{ij} = 1$). Если выбран метод *NVP*, то устанавливается значение 0, если *RB*, то 1.

2. Var_{v1} – вариант вероятности сбоя компонента и соответствующей трудоемкости для достижения этой вероятности сбоя. Возможные варианты задаются аналитиком ($1 <= Var_{v1} <= Кол\text{-}во\ вариантов\ для\ данного\ компонента$).

3. $Var_{v2}..Var_{v10}$ – номер варианта вероятности сбоя для каждой версии компонента, аналогично пункту 2 ($0 <= Var_{v2..10} <= Кол\text{-}во\ вариантов\ для\ данного\ компонента$, 0 – версии нет). Предельное количество версий программного компонента, если будет применена избыточность, заранее задается аналитиком.

Если аллели $Var_{v2}..Var_{v10}$ принимают значение 0, то считается, что программная избыточность не вводится в данном компоненте ($B_{ij} = 1$).

Для другой части компонентов, участвующих в ГА, недопустимо введение программной избыточности. Поэтому для них изменяется только вариант *Var* вероятности сбоя компонента и соответствующей трудоемкости для достижения этой вероятности сбоя ($1 <= Var_{v1} <= Кол\text{-}во\ вариантов\ для\ данного\ компонента$).

Таким образом, фенотип особи формируется из компонентов, участвующих в ГА, где каждая приведенная характеристика программного компонента представляет собой ген. В таблице 1 представлен общий вид фенотипа и примером аллелей и локусов.

Генетический алгоритм. При определении пригодности особи значения S и T_s рассчитываются по алгоритму, изображенном на рисунке 1.

Каждая особь с рассчитанными критериями записывается в массив. Перед тем как рассчитывать критерии для следующей особи происходит поиск аналогичной уже рассчитанной ранее особи. Это действие целесообразно и позволяет сэкономить время, т.к. расчет критериев значительно дольше поиска идентичной особи в массиве решений.

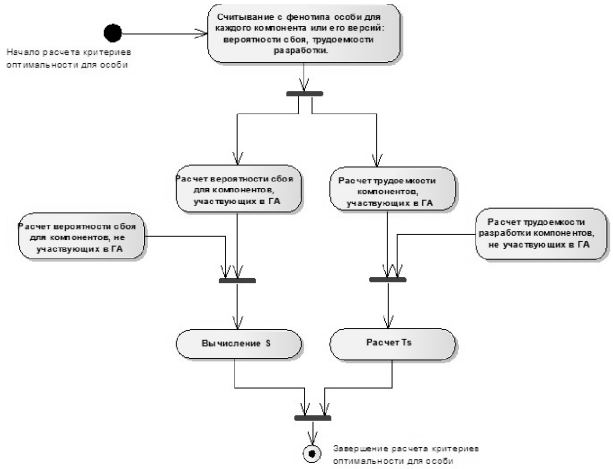

Рис. 1. Алгоритм расчета критериев оптимальности

Скрещивание выбранных особей происходит с заранее заданной вероятностью. Если родители не скрещиваются, то происходит их клонирование.

Скрещивание происходит посредством разрыва хромосомы родителей в заданном количестве точек, при этом потомки получают различные признаки обоих родителей.

В природе существует огромное количество признаков и свойств живых организмов, которые определяются двумя и более парами генов, и наоборот, один ген часто контролирует многие признаки. Кроме того, действие гена может быть изменено соседством других генов и условиями внешней среды. Еще в одной из работ 1928 г. Ю.А. Филипченко, было доказано, что наряду с «основным» геном, определяющим признак, существует ряд генов-модификаторов этого

Таблица 1

Генотип и фенотип особи

Группа компонентов, с возможностью программной избыточности									Группа компонентов, без возможности программной избыточности		
Компонент 1				..	Компонент N				Компонент 1	..	Компонент N
NVP/ RB	Var_{v1}	..	Var_{v4}		NVP/ RB	Var_{v1}	..	Var_{v5}	Var		Var
1	3		0		0	1		0	2		3

признака. Подобный тип наследования встречается часто. Таким образом, фенотип, как правило, представляет собой результат сложного взаимодействия генов [6, с. 50].

Надежность и трудозатраты для компонентов с возможной программной избыточностью определяются взаимодействующими генами версий и метода избыточности. Разрывы хромосомы могут происходить в месте жирной черты (таблица 1). Причем вероятность выбора точки разрыва хромосомы между генами одного программного компонента (связанными генами) заранее задана. Равномерное скрещивание возможно с разрывом во всех точках или только несвязанных генов.

Мутация особей популяции происходит с заданной вероятностью. Кроме вероятности применения мутации к каждой особи, используется вероятность применения мутации к каждому ее гену, величину которой обычно задают от 1 до 10 % [7]. При мутации бинарных генов метода избыточности происходит инвертирование значения.

Аллели генов вариантов надежности/трудоемкости характеризуются шкалой порядка, таким образом, что каждый следующий вариант, лучше по надежности, но хуже по трудозатратам. Т.е. заранее заданные варианты для конкретного программного компонента оптимальны по Парето. Мутация таких генов может происходить двумя способами:

1. Выбор любого варианта надежности/трудоемкости равновероятен.

2. Выбор любого варианта надежности/трудоемкости происходит вероятностью распределенной по закону нормального распределения вероятностей для дискретной случайной величины [8]. При этом выбор текущего значения варианта невозможно. При мутации генов версий программных компонентов, для которых возможно значение 0 (отсутствие версии), такое значение добавляется как дополнительное возможное значение. В таблице 2 приведены вероятности выбора вариантов при мутации гена версии программного компонента, текущее значение которого равно 4, а общее количество заданных вариантов равно 7.

Генетический алгоритм основан на методе с независимой селекцией Шаффера при многокритериальной оптимизации *VEGA* (*Vector Evaluated Genetic Algorithm*) [9].

Селекция происходит с вероятностью, пропорциональной значению критерия. Вероятность индивида быть отобранным по критерию S рассчитывается по формуле [10, с. 36]:

$$p_i = \frac{f_i}{\sum\limits_{j=1}^{N} f_j} = \frac{f_i}{\overline{f} \cdot N},$$

где f_i – пригодность индивида i, $\overline{f} = \frac{1}{N}\sum\limits_{j=1}^{N} f_j$ – средняя пригодность популяции, N – размер попуяции, $\sum\limits_{j=1}^{N} p_j = 1$.

Вероятность индивида быть отобранным по минимизируемому критерию T_s рассчитывается по формуле [11, с. 19]:

$$p_i = \frac{-f_i + C}{N * C - \sum\limits_{j=1}^{N} f_j},$$

где f_i – пригодность индивида i, C – константа, определяющая минимальную пригодность популяции, $C : p_i \geq 0, \forall i$, N – размер попуяции, $\sum\limits_{j=1}^{N} p_j = 1$.

В рассматриваемом алгоритме константа C равна максимальной пригодности особи в популяции. Таким образом, минимальная вероятность быть отобранным индивидом по критерию T_s равна 0.

Входные параметры для ГА следующие:

– размер популяции (N);

– вероятность скрещивания (*prob_cross*);

– вид скрещивания (1,2,3-точечное, равномерное);

– вероятность разрыва связанных генов (*prob_cross_inter*);

Таблица 2

Распределение вероятностей выбора варианта

Отсутствие версии	Вар. 1	Вар. 2	Вар. 3	Вар. 4	Вар. 5	Вар. 6	Вар. 7
0.015625	0.09375	0.234375	0.3125	0	0.234375	0.09375	0.015625

– вероятность мутации особи (*prob_mutate*);

– вероятность мутации гена (*prob_mutate_gen*);

– закон распределения вероятности выбора альтернативы при мутации гена (равновероятный, нормальный);

– критерии останова (максимальное время работы *time_ga*, количество популяций без улучшения решения (стагнация) *stagnancy*, количество популяций pop_count и количество решений solution_count).

Описание алгоритма следующее:

1. Генерация родительской популяции P размером N случайных особей.

2. Расчет критериев для всех особей популяции P (рисунок 1).

3. Пропорциональная селекция $N/2$ особей из P по критерию S в промежуточную популяцию P'.

4. Пропорциональная селекция $N/2$ особей по критерию T_s в промежуточную популяцию P'.

5. Скрещивание (с вероятностью *prob_cross*) $N/2$ случайно выбранных пар особей из промежуточной популяции P'. Добавление N полученных потомков в основную популяцию P.

6. Проведение оператора мутации с вероятностью *prob_mutate* на каждой особи основной популяции P.

7. Расчет критериев для всех особей популяции P.

8. Выбор из популяции P одного из лучших решений. Если в найденных решениях есть лучше, то *stagnancy* = *stagnancy* + 1.

9. Если не сработал хотя бы один критерий останова, то переход на шаг 3.

Полученные в ГА решения отбираются в множество Парето. С точки зрения математики решения множества Парето не могут быть предпочтены друг другу, поэтому после формирования множества Парето задача может считаться математически решенной [9].

3. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

В программной реализации описываемого алгоритма на ПК с процессором 2,1 ГГц для тестовой задачи расчет критериев одной особи требует 1,4–1,6 сек. Такое время обусловлено сложностью алгоритма функции S, для которого считывание значений переменных происходит из БД, так как модель программной архитектуры имеет реляционное представление.

Пространство оптимизации для задачи рассчитывается по следующей формуле:

$$N = \prod_{d=1}^{D}\left(Var_d + 2\cdot\sum_{Ver_d=2}^{L_d} Var_d^{Ver_d}\right)\times \times\prod_{f=1}^{F} Var_f,$$

где N – количество комбинаций; D – количество программных компонентов, в которых возможно введение избыточности; Var_d – количество вариантов надежности с соответствующей трудоемкостью для компонентов с возможностью введения избыточности; Ver_d – количество используемых версий при введении избыточности $(2..L_d)$; L_d – предельно допустимое количество версий; F – количество программных компонентов, в которых не возможно введение избыточности; Var_f – количество вариантов надежности с соответствующей трудоемкостью для компонентов без избыточности.

Условия тестовой задачи для испытаний алгоритма приведены в таблице 3.

Таблица 3

Условия тестовой задачи

№ программного компонента	Участие в ГА	Максимальное кол-во версий	Кол-во альтернатив «вероятность сбоя/трудозатраты»
1	Возможна программная избыточность	3	2
2	Возможна программная избыточность	2	3
3	Возможна программная избыточность	2	3
4	Без программной избыточности	1	3
5	Без участия	1	1
6	Без участия	1	1
7	Без участия	1	1
8	Без участия	1	1
9	Без участия	1	1

Решение задачи полным перебором заняло 15 часов. Задача имеет 34398 решений, из них 41 составляют множество парето-оптимальных решений.

В таблице 4 представлены показатели работы 56 генетических алгоритмов с различными параметрами. На каждый ГА было дано 50 поколений по 50 популяций, вероятность скрещи-

Таблица 4

Показатели работы алгоритмов

№	Параметры ГА			Δf	Точность аппрокс. множества Парето
	Вид скрещивания	Вероятность мутации	Распределение вер-ти выбора		
1	1 точечное	Особи 0.05, гена 0.01	Равномерное	0.20059	4.88%
2			Нормальное	0.20135	0%
3		Особи 0.15, гена 0.1	Равномерное	0.23293	7.32%
4			Нормальное	0.22919	7.32%
5	2 точечное	Особи 0.05, гена 0.01	Равномерное	0.24616	7.32%
6			Нормальное	0.24471	4.88%
7		Особи 0.15, гена 0.1	Равномерное	0.27366	14.63%
8			Нормальное	0.24231	2.44%
9	3 точечное	Особи 0.05, гена 0.01	Равномерное	0.24143	7.32%
10			Нормальное	0.27182	9.76%
11		Особи 0.15, гена 0.1	Равномерное	0.23596	7.32%
12			Нормальное	0.32167	4.88%
13	Равномерное	Особи 0.05, гена 0.01	Равномерное	0.24772	7.32%
14			Нормальное	0.28787	7.32%
15		Особи 0.15, гена 0.1	Равномерное	0.22913	7.32%
16			Нормальное	0.23282	9.76%
17	1-точечное, модификация с вероятностью разрыва связанных генов 0,2	Особи 0.05, гена 0.01	Равномерное	0.22244	4.88%
18			Нормальное	0.25357	0%
19		Особи 0.15, гена 0.1	Равномерное	0.28261	9.76%
20			Нормальное	0.31882	9.76%
21	1-точечное, модификация с вероятностью разрыва связанных генов 0,5	Особи 0.05, гена 0.01	Равномерное	0.23539	7.32%
22			Нормальное	0.26483	4.88%
23		Особи 0.15, гена 0.1	Равномерное	0.28265	7.32%
24			Нормальное	0.21612	7.32%
25	1-точечное, модификация с вероятностью разрыва связанных генов 0,8	Особи 0.05, гена 0.01	Равномерное	0.26384	2.44%
26			Нормальное	0.25002	2.44%
27		Особи 0.15, гена 0.1	Равномерное	0.27336	12.20%
28			Нормальное	0.25930	0%
29	2-точечное, модификация с вероятностью разрыва связанных генов 0,2	Особи 0.05, гена 0.01	Равномерное	0.20993	4.88%
30			Нормальное	0.23056	4.88%
31		Особи 0.15, гена 0.1	Равномерное	0.26518	7.32%
32			Нормальное	0.22881	12.20%
33	2-точечное, модификация с вероятностью разрыва связанных генов 0,5	Особи 0.05, гена 0.01	Равномерное	0.22182	7.32%
34			Нормальное	0.25878	2.44%
35		Особи 0.15, гена 0.1	Равномерное	0.26569	7.32%
36			Нормальное	0.19977	12.20%
37	2-точечное, модификация с вероятностью разрыва связанных генов 0,8	Особи 0.05, гена 0.01	Равномерное	0.26976	0%
38			Нормальное	0.30589	0%
39		Особи 0.15, гена 0.1	Равномерное	0.26375	7.32%
40			Нормальное	0.25216	12.20%

Окончание таблицы 4

41	3-точечное, модификация с вероятностью разрыва связанных генов 0,2	Особи 0.05, гена 0.01	Равномерное	0.30790	9.77%
42			Нормальное	0.19789	4.88%
43		Особи 0.15, гена 0.1	Равномерное	0.23400	14.63%
44			Нормальное	0.22342	7.32%
45	3-точечное, модификация с вероятностью разрыва связанных генов 0,5	Особи 0.05, гена 0.01	Равномерное	0.21736	7.32%
46			Нормальное	0.22206	2.44%
47		Особи 0.15, гена 0.1	Равномерное	0.26236	14.63%
48			Нормальное	0.26294	14.63%
49	3-точечное, модификация с вероятностью разрыва связанных генов 0,8	Особи 0.05, гена 0.01	Равномерное	0.26100	7.32%
50			Нормальное	0.28202	7.32%
51		Особи 0.15, гена 0.1	Равномерное	0.25997	12.20%
52			Нормальное	0.27751	14.63%
53	Равномерное, модификация с разрывом только несвязанных генов	Особи 0.05, гена 0.01	Равномерное	0.20482	4.88%
54			Нормальное	0.29353	7.32%
55		Особи 0.15, гена 0.1	Равномерное	0.23159	7.32%
56			Нормальное	0.20111	9.76%

вания – 0.9. Таким образом, будет просчитано меньше 7 % поискового пространства.

Среднее относительное отклонение (рассеяние решений) в критериальном пространстве Δf рассчитывается по формуле [10, с. 62]:

$$\Delta f = \frac{1}{K}\sum_{k=1}^{K}\frac{\Delta f^k}{M_k - m_k},$$

где $\Delta f^k = \dfrac{\sum_{i,j}^{N'}\left|f_{ij}^k - f^k\right|}{N'}$ – среднее отклонение в целевой пространстве по каждой целевой функции,

$f^k = \dfrac{\sum_{i,j}^{N'} f_{ij}^k}{N'}$ – среднее расстояние в целевом пространстве,

$f_{ij}^k = \left|f_i^k - f_j^k\right|$ – расстояние между индивидами в критериальном пространстве,

i, j – пара индивидов популяции,

N' – количество возможных пар индивидов,

K – количество критериев задачи,

M_k – максимальное значение по k-ой целевой функции,

m_k – максимальное значение по k-ой целевой функции.

Точность аппроксимации множества парето определяется из процентного соотношения количества найденных недоминируемых решений к общему количеству недоминируемых решений задачи.

В 10 из 28 случаев, при нормальном распределении вероятности выбора аллели варианта «надежность/трудозатраты», количество решений из множества Парето меньше, чем при равномерном распределении вероятности. В 8 случаях из 28 при равномерном распределении было найдено меньше решений. В остальных случаях количество решений равно. Но при нормальном распределении рассеяние точек решений в критериальном пространстве больше. Поэтому выбор закона распределения вероятности выбора аллели, при мутации, влияет на эффективность ГА не значительно. На эффективность ГА больше влияет вероятность мутации. При вероятности мутации особи 0.15 и гена 0.1 эффективность алгоритмов более высокая, чем при вероятности мутации особи 0.05 и гена 0.01.

Из таблицы 4 видно, что поиск решений наиболее эффективен при использовании модифицированного оператора 3-точечного кроссинговера алгоритмами 43, 47, 48, 52.

На рисунке 2 представлены найденные этими алгоритмами недоминируемые решения в критериальном пространстве и аппроксимированный фронт Парето тестовой задачи.

Только 14.63 % найденных решений входят в множество Парето задачи. Остальные решения достаточно близки к множеству Парето.

Если сравнивать эффективность ГА при использовании модифицированного оператора скрещивания, то наиболее эффективно отработали алгоритмы с вероятностью разрыва связанных генов 0,5 для выбранной задачи. Модификация равномерного скрещивания с разрывом только несвязанных генов менее эффективна.

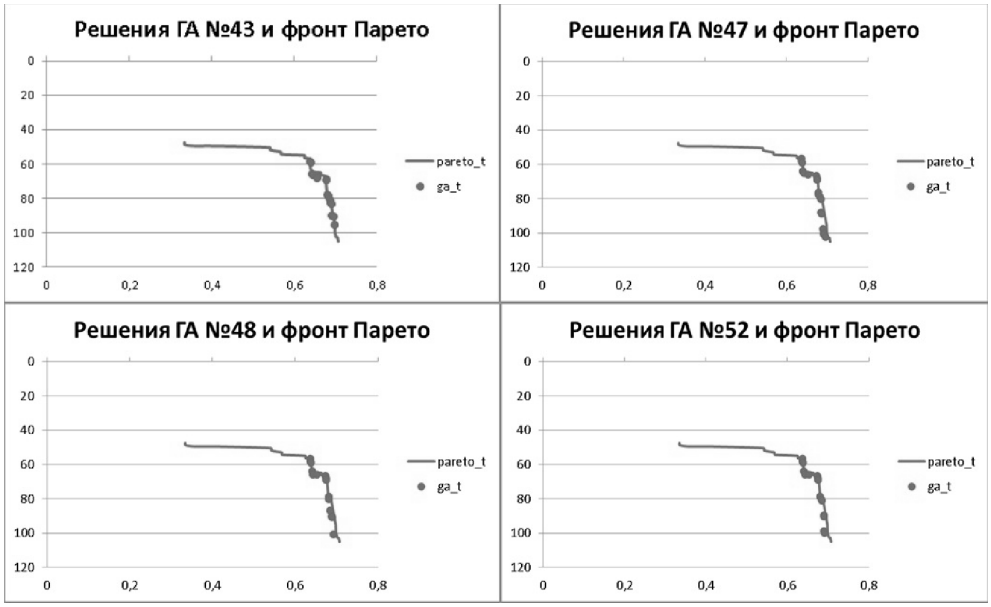

Рис. 2. Решения алгоритмов 43, 47, 48 и 52

ВЫВОДЫ

Разработан ГА для многокритериальной безусловной оптимизации программной архитектуры, основанный на методе VEGA. Описана модификация операторов кроссинговера и мутации генотипа с аллелями в шкале порядка. Применение модифицированного оператора кроссинговера с различной вероятностью разрыва связанных и несвязанных генов увеличивает эффективность ГА. Применение нормального закона распределения вероятности выбора аллели, при мутации, не эффективно. Алгоритм находит решение за приемлемое время, поэтому он может быть использован при решении задач проектирования реальных программных систем.

СПИСОК ЛИТЕРАТУРЫ

1. *Галушин П.В.* Асимптотический вероятностный генетический алгоритм дискретной оптимизации // П.В. Галушин, О.Э. Семёнкина // Вестник Сибирского государственного аэрокосмического университета им. академика М.Ф. Решетнева. № 5 (38). — 2011. — С. 25—29.

2. *Борисенко М.Л.* Использование нечеткой модели при оптимизации характеристик программных средств с помощью многокритериального генетического алгоритма: Дис. канд. техн. наук: Москва, 2002. – 153 с.

3. *Русаков М.А.* Многоэтапный анализ архитектурной надежности в сложных информационно-управляющих системах: Дис. канд. техн. наук: Красноярск, 2005 – 168 с.

4. *Новой, А.В.* Система анализа архитектурной надежности программного обеспечения.: Дис. канд. техн. наук: Красноярск, 2011 – 131 с.

5. *Шеенок Д.А., Жуков В.Г., Терсков В.А.* Повышение надежности программного обеспечения сложных систем. // Вестник СибГАУ. Вып. 5(45). – Красноярск, 2012. – С. 28-33.

6. *Инге-Вечтомов С.Г.* Генетика с основами селекции: Учеб. для биолог. спец. ун-тов. – М.: Высш. шк., 1989. – 591 с.

7. *Панченко Т.В.* генетические алгоритмы [Текст]: учебно-методическое пособие / под ред. Ю.Ю. Тарасевича. – Астрахань: Издательский дом «Астраханский университет», 2007. – 83 с.

8. *Спицын В.Г., Цой Ю.Р.* Представление знаний в информационных системах: учебное пособие. – Томск: Изд-во ТПУ, 2006. – 146 с.

9. *Сергиенко Р.Б.* Коэволюционный алгоритм для задач условной и многокритериальной оптимизации / Р.Б. Сергиенко, Е.С. Семенкин // Программные продукты и системы. – № 4. – 2010. – С. 24-28.

10. *Гуменникова А.В.* Адаптивные поисковые алгоритмы для решения сложных задач многокритериальной оптимизации: Дис. канд. техн. наук: Красноярск, 2006 – 129 с.

11. *Сопов Е.А.* Эволюционные алгоритмы моделирования и оптимизации сложных систем: Дис. канд. техн. наук: Красноярск, 2004 – 129 с.

Шеенок Дмитрий Александрович – аспирант, Красноярский институт железнодорож-

Sheenok D. A. – postgraduate student Department of Mathematics and Computer

ного транспорта. Тел.: 8-923-303-36-37. E-mail: dmitryshkras@rambler.ru

Терсков Виталий Анатольевич – д.т.н., проф. кафедры "Математики и информатики", Красноярский институт железнодорожного транспорта

Science, Krasnoyarsk Institute of Railway Transport. Tel. 8-923-303-36-37. E-mail: dmitryshkras@rambler.ru

Terskov Vitaly A. – Doctor of Technical Sciencies, Professor, Department of Mathematics and Computer Science, Krasnoyarsk Railway Institute.