РЕШЕНИЕ ЗАДАЧ ОБЕСПЕЧЕНИЯ ОТКАЗОУСТОЙЧИВОСТИ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ С ПОМОЩЬЮ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

Е.Н. Турута, Н.Ю. Альбов, Р.Г. Давыдов

Московский Технический Университет связи и информатики, ул. Авиамоторная. 8, Москва, 111024, РОССИЯ, тел.(095) 716-10-55, turuta_en@mail.ru

АННОТАЦИЯ

Рассматривается применение генетических алгоритмов для решения задач дискретной оптимизации, возникающих при разработке методов обеспечения отказоустойчивости распределенных вычислительных систем (РВС).

1. ОБЕСПЕЧЕНИЕ ОТКАЗОУСТОЙЧИВОСТИ РВС НА ОСНОВЕ РАЦИОНАЛЬНОГО СТАТИЧЕСКОГО ПЕРЕРАСПРЕДЕЛЕНИЯ ЗАДАЧ (РСПЗ/ОУ)

Один из перспективных подходов к обеспечению отказоустойчивости распределенных вычислительных систем (РВС) основан на оптимальном перераспределении выполняющихся в системе задач между ее процессорными модулями (ПМ) при отказах некоторых из них. Этот подход реализуется посредством отказоустойчивого размещения задач, т.е. такого статического распределения задач по ПМ, при котором определенное число копий каждой задачи размещается в различных ПМ [1,2]. Система рассматривается как совокупность n однотипных ПМ, объединенных подсистемой связи. Каждый ПМ имеет один процессор и память ограниченного объема. Система выполняет фиксированное задание Г - известное множество задач $\Omega^0=\{U_1,\ldots U_j,\ldots, U_L\}$ с заданными требованиями к порядку их выполнения и взаимосвязям.

Каждая задача U_j характеризуется следующими показателями:

- b_j - *вес* задачи, т.е. величина, определяющая важность задачи U_j для системы (более важной задаче соответствует больший вес);
- v_j - объем оперативной памяти, требуемый для хранения и выполнения задачи;
- τ_j - время выполнения задачи.

В процессе работы РВС возможны отказы ПМ с известным распределением вероятностей. *Состояние системы* определяется как $s_v=\delta_1,\ldots,\delta_n$, где $\delta_i=0$, если M_i – *работоспособный ПМ (р-ПМ)* и $\delta_i = 1$, если Mi - *отказавший ПМ (о-ПМ)*; $s^0 = 00\ldots0$ – *начальное состояние* системы; все состояния $s_v \neq s^0$ называются *искаженными*.

Известно начальное *распределение задач(РЗ)* по ПМ при отсутствии их отказов (для *состояния* s^0), описываемое матрицей $\mathbf{D}^0=\|d^0_{ji}\|$, где: $d^0_{ji} = 1$, если задача U_j назначена для выполнения в ПМ M_i и размещена в нем, $d^0_{ji} = 0$ в противном случае; каждая задача назначена *в один и только в один ПМ*. Задачи, назначенные в ПМ M_i в соответствии с начальным РЗ, назовем *собственными задачами* этого ПМ.

Качество работы системы оцениваем ее *функциональной мощностью* E_v в состоянии s_v, определяемой как сумма весов всех задач, назначаемых для выполнения в работоспособные ПМ в состоянии s_v (и составляющих множество Ω^v).

Требуемый уровень отказоустойчивости системы задается *множеством* $\mathbf{S} = \{s_v\}$ *работоспособных состояний*, которое определим как множество *всех* таких состояний, число k отказавших ПМ для которых *не превосходит* заданного значения d, очевидно $\mathbf{S} = s^0 \cup \mathbf{S}^\omega$, где $\mathbf{S}^\omega=\{s_\omega\}$ - множество *искаженных* работоспособных состояний.

Требуется путем организации при отказах ПМ надлежащего *перераспределения задач (ПЗ)* между различными р-ПМ обеспечить выполнение заданных требований к отказоустойчивости системы и к другим ее показателям. Для этого при проектировании РВС или при подготовке к реализации в ней определенного задания, необходимо найти *оптимальные планы* $\mathbf{D}_\omega = \|d^\omega_{ji}\|$ *распределения задач* (РЗ) *для каждого состояния* $s_\omega \in \mathbf{S}^\omega$ и результирующее *отказоустойчивое размещение задач* $\mathbf{Z} = \|z_{ji}\|$ для всех ПМ, где $z_{ji}=1$, если задача U_j в *размещена* в ПМ M_i, (т.е. ее программный модуль загружен в память ПМ M_i), $z_{ji} = 0$ в противном случае; z_{ji} определяется как дизъюнкция значений d^v_{ji} для всех состояний $s_v \in \mathbf{S}$ (включая начальное).

После того, как *оптимальные планы* $\mathbf{D}_\omega$ найдены, каждый ПМ заранее обеспечивается ап-

паратными и программными ресурсами, необходимыми для выполнения задач, которые *могут* назначаться ему в любом состоянии $s_\nu \in \mathbf{S}$. При переходе ВС вследствие отказов некоторых ПМ в любое *искаженное состояние* $s_\omega \in \mathbf{S}^\omega$ и обнаружении этого факта во всех р-ПМ начинается выполнение задач, назначенных им в соответствии с планом $\mathbf{D}_\omega$.

Эта задача относится к классу задач дискретной оптимизации с булевыми переменными, для решения которых в принципе могут применяться точные методы математического программирования. Однако, поскольку для многих задач практической сложности применение таких методов затруднено ввиду их чрезвычайно большой трудоемкости, то целесообразно использование эвристических методов, дающих не точное, а некоторое "хорошее" решение. К ним относятся различные модели *эволюционных вычислений*, в частности *генетические алгоритмы*.

2. Генетические алгоритмы (ГА)

Алгоритмы этого класса [3, 4] - поисковые алгоритмы, использующие механизмы, прототипом которых являются процессы эволюции популяций живых организмов, подчиняющиеся принципам естественного отбора. Алгоритм осуществляет поиск лучшего решения *в пространстве поиска решений*. Это пространство под воздействием операторов алгоритма, имитирующих биологические механизмы эволюции, изменяется (эволюционирует) в направлении "улучшения" содержащихся в нем решений. Результатом такой эволюции должно быть пространство поиска, содержащее наилучшие (или приемлемые) решения, которые и обнаруживаются алгоритмом.

В терминологии ГА пространство поиска решений интерпретируется как *популяция* $\mathbf{P} = \{\mathbf{A}_1,.., \mathbf{A}_k,.., \mathbf{A}_N\}$, а каждое *решение* из этого пространства – как *хромосома* (или *особь*) $\mathbf{A}_k$, представляемая в виде строки символов, называемых *генами*:

$$\mathbf{A}_k = \{\mathbf{a}_1,\ldots,\mathbf{a}_j,\ldots,\mathbf{a}_L\}.$$

С каждым *геном* $\mathbf{a}_j$ сопоставлено множество $\mathbf{W}_j=\{w^j_h\}$ его возможных значений w (*"аллелей"*). На множестве решений (хромосом) задается *целевая функция (ЦФ)* - функция "полезности": $\boldsymbol{F}(\mathbf{A}_k)$ - "полезность" ("эффективность") решения $\mathbf{A}_k$. Генетический алгоритм осуществляет поиск хромосомы $\mathbf{A}^*$, для которой $F(\mathbf{A}^*) = \max_{\mathbf{P}} F(\mathbf{A}_k)$. *Размер* (численность) N популяции может изменяться в процессе работы алгоритма, но наиболее распространены ГА, для которых N = Const. От размера популяции зависит время поиска решения и его «качество». Важно, что размер популяции значительно меньше числа всех допустимых решений. Именно это позволяет находить решение за приемлемое время.

Работа ГА - итеративный процесс, продолжающийся вплоть до выполнения *заданных условий остановки* (близость полученных решений к заданному значению, прекращение роста средней "полезности" популяции, минимальная численность популяции, число итераций и др.).

Этот процесс включает генерацию (случайным образом) *начальной популяции* $\mathbf{P}_0 = \{\mathbf{A}^0_{r1},.., \mathbf{A}^0_{rk},..,\mathbf{A}^0_{rN_0}\}$ (выбор ее размера – отдельный вопрос), применение к начальной популяции *последовательности генетических операторов*, порождающей новую популяцию $\mathbf{P}_t$, проверку условий остановки работы алгоритма и либо выдачу результата (при выполнении этих условий), либо переход к созданию следующей популяции $\mathbf{P}_{t+1}$ (в противном случае).

Последовательность генетических операторов обычно включает операторы отбора (репродукции), кроссинговера (скрещивания), мутации и инверсии. Все эти операторы являются вероятностными, т.е. их параметры - случайные величины.

1. *Отбор* (репродукция), т.е. формирование следующей популяции из имеющейся в данный момент, имитирует принцип «выживания сильнейших» и осуществляется путем стохастической, но целенаправленной *селекции*, т.е. отбора *лучших* хромосом на основе их значений функции "полезности". Основные виды селекции: *пропорциональная и турнирная*.

   При пропорциональной селекции вероятность выбора *r*-й хромосомы определяется как:

$$P(A_r) = F(A_r) \Big/ \sum_{k=1}^{N} F(A_k)$$

при условии, что $\boldsymbol{F}(\mathbf{A}_k) > 0$ для всех $k=1...N$. Этот принцип программно реализуется на основе метафоры «колеса рулетки».

При *турнирной селекции* из популяции $\mathbf{P}_t$ случайным образом выбирается подмножество $\mathbf{P}_t^* \subset \mathbf{P}_t$ и хромосома из этого подмножества, имеющая максимальное значение функции приспособленности, выбирается в следующую популяцию $\mathbf{P}_{t+1}$.

Применяются также «элитные» методы отбора, гарантирующие обязательное выживание при отборе лучшей или группы лучших особей популяции. Элитизм может быть внедрен практически в любой стандартный метод отбора.

2. *Кроссинговер* (скрещивание): (1) в популяции, полученной после отбора, выбираются две хромосомы (по какому-либо правилу, возможно, случайно); (2) случайным образом выбирается *точка разрыва* (точка кроссинговера), т.е. позиция m в строках, описывающих хромосомы, $1<m<L$,; (3) выбранные хромосомы (родители) обмениваются своими частями, формируя таким образом две новых хромосомы (потомков); (4) выбор *одной* из двух полученных хромосом-потомков в качестве результата кроссинговера (как правило, операторы, порождающие более одного потомка, отбирают из них по какой-либо стратегии одного или более в качестве результата операции, отбрасывая остальные). Это - одноточечный кроссинговер. Используются также различные варианты многоточечного кроссинговера, когда хромосомы-родители разрываются не в одном месте, а в нескольких. В отличие от живой природы в ГА применяется кроссинговер трех или более хромосом: образующиеся хромосомы, содержат участки генов от двух родителей и более.
3. *Мутация* применяется с некоторой достаточно низкой вероятностью P_m (обычно: $P_m \approx 0{,}001$) к хромосомам, полученным после кроссинговера. Оператор *одноточечной мутации* выбирает случайным образом ген в хромосоме и меняет его значение на другое из области допустимых значений. Применяется и *многоточечная мутация*, которая является композицией нескольких одноточечных мутаций.
4. *Инверсия* может применяться с ненулевой и очень низкой вероятностью наряду с мутациями. Она изменяет не значения генов, а только порядок их следования в случайно выбранной хромосоме. В простейшем случае в такой хромосоме случайном образом выбирается точка разрыва, и меняются местами начало и конец хромосомы. Допустимо применение многоточечной инверсии.

Кроссинговер, мутация и инверсия могут порождать новые хромосомы (решения), которые никогда не встречались в предыдущих популяциях. Их следует оценить значениями функции приспособленности.

Последующая популяция $\mathbf{P}_{t+1}$ может быть образована как целиком только из потомков хромосом предыдущей популяции $\mathbf{P}_t$, полученных в результате применения к $\mathbf{P}_t$ операторов (2), (3), (4), так и с частичным сохранением наилучших особей из $\mathbf{P}_t$. В первом случае новые хромосомы полностью заменяют старые, а во втором – только менее ценные из них.

Работа ГА прекращается при достижении текущей популяцией *состояния адаптации*, которое идентифицируется по стягиванию ядра популяции сначала в "плотное облачко", а затем - в точку. Алгоритм не гарантирует получение глобального экстремума (хотя это и возможно), а дает некоторое "хорошее" решение за приемлемое время.

3. Применение ГА для решения задач РСПЗ/ОУ

В зависимости от требований к системе задача обеспечения отказоустойчивости с помощью РСПЗ/ОУ может иметь различные формулировки. Рассмотрим применение ГА для решения следующих двух задач.

Задача 1. Для каждого искаженного работоспособного состояния $s_\omega \in \mathbf{S}^\omega$ найти такой план $\mathbf{D}_\omega = \|d^\omega_{ji}\|$ *распределения задач* (РЗ), при котором достигается *максимальное значение функциональной мощности системы* в состоянии s_ω

$$E^f_\omega = \sum_{U_j \in \Omega^f_\omega} \sum_{M_i \in H_\omega} d^\omega_{ji} b_j \to \max \qquad (1)$$

где

Ω^f_ω - множество *собственных задач отказавших ПМ*, $\mathbf{H}_\omega$ – множество р-ПМ для состояния $\mathbf{s}_\omega$,

при ограничениях:

- *на суммарное время* выполнения в ПМ M_i всех задач, назначенных ему в состоянии $\mathbf{s}_\omega$:

$$\Delta T^\omega_{\Sigma i} = \sum_{U_j \in \Omega^f_\omega} \tau_j d^\omega_{ji} \le \Delta T^*_{\Sigma i}, \quad i = 1,\ldots,g_\omega, \qquad (2)$$

где

g_ω – число р-ПМ в состоянии s_ω

$\Delta T^*_{\Sigma i} = T^*_{\Sigma i} - T^0_{\Sigma i}$ – ресурс времени для размещения в ПМ M_i копий задач отказавших ПМ,

$T^*_{\Sigma i}$ – заданное граничное значение *суммарного времени* выполнения *всех* задач, назначенных в р-ПМ M_i ;

$T^0_{\Sigma i}$ – *суммарное время* выполнения всех *собственных* задач ПМ M_i;

- *на объем памяти* каждого р-ПМ M_i, доступный для размещения копий задач, дополнительно назначаемых этому ПМ для выполнения в искаженном состоянии s_ω:

$$\Delta V_{\omega i} = \sum_{U_j \in \Omega^f_\omega} d^\omega_{ji} v_j \le \Delta V^*_{\omega i}, \quad \forall M_i \in H_\omega,\ i = 1,\dots, g_\omega, \qquad (3)$$

где

$\Delta V^*_{\omega i} = (V^m_i - V^0_i) / N(S^{\omega r}_i)$, V^m_i - максимальный объем памяти ПМ M_i,

V^0_i - объем памяти ПМ M_i, занятый его собственными задачами;

$N(S^{\omega r}_i)$ - число таких состояний $s_\omega \in S^\omega$, в которых данный ПМ M_i *не отказал*) и при учете, что *в каждом* состоянии S_ω задача $U_j \in \Omega^f_\omega$ может быть либо назначена *только в один р-ПМ,* либо отброшена, т.е.

$$\sum_{i=1}^{n} d^\omega_{ji} \le 1, \quad \forall U_j \in \Omega^f_\omega. \qquad (4)$$

Рассматриваем случай, когда в каждом состоянии s_ω подвергаются *перераспределению* (т.е. передаются для выполнения в какие–либо р-ПМ, либо отбрасываются) только *собственные задачи отказавших ПМ,* а *собственные задачи всех р-ПМ* продолжают выполняться в "своих" ПМ (решение задачи для случая, когда в каждом состоянии s_ω *заново перераспределяются все задачи* начального множества Ω^0, аналогично описываемому ниже).

Задача 2. Для каждого состояния $s_\omega \in S^\omega$ найти такой план РЗ $D_\omega = \| d^\omega_{ji} \|$, при котором достигается *минимальное значение затрат* на размещение в работоспособных ПМ резервных копий тех задач, которые назначаются в эти ПМ для выполнения в *данном* состоянии:

$$\Delta C_\omega = \sum_{U_j \in \Omega^f_\omega} \sum_{M_i \in H_\omega} d^\omega_{ji} c_{ji} \to \min \qquad (5)$$

где

c_{ji} - *затраты* на размещение *копии* задачи U_j в ПМ M_i (дополнительная память, интерфейсы, каналы связи и др.), при условии, что эта задача не является собственной задачей ПМ M_i; если все ПМ *идентичны,* то $c_{ji} = c_j$ для любого ПМ M_i.

при ограничениях

- *на объем доступной памяти* каждого р-ПМ M_i, в данном состоянии s_ω (3);
- *на минимально допустимую функциональную мощность системы* в состоянии s_ω :

$$E_\omega = \sum_{U_j \in \Omega^0_\omega} \sum_{M_i \in H_\omega} d^\omega_{ji} b_j \ge E^*_\omega \qquad (6)$$

При использовании ГА для решения этих задач каждая *хромосома,* т.е. возможное *решение* для состояния s_ω, описывает некоторый план РЗ D_ω для данного состояния. Рассматривая *произвольное* состояние s_ω, опустим индекс состояния и представим хромосому в виде строки $A_k = \{a_1,\dots,a_j,\dots,a_L\}$, где L – общее число задач, выполняемых системой, *ген* a_j - номер р-ПМ, в который назначена задача U_j *в данном* состоянии, множество возможных значений *любого гена* a_j - это множество W^r номеров р-ПМ для *данного* состояния, т.е. $a_j \in W^r$, $W^r = \{0, w_h, h=1,2,..,g\}$, где w_h – номер р-ПМ, g – число р-ПМ в данном состоянии, 0 – номер "фиктивного ПМ", размещение задачи в котором означает ее отбрасывание. Пример хромосомы:

№ задачи	*1*	*2*	...	*j*	...	*L*
№ р-ПМ	**1**	**0**	..	a_j	..	a_L

Решение задачи 1. В качестве "функции полезности" $F(A_k)$ принята *функциональная мощность системы* в состоянии s_ω - сумма весов тех задач U_j, для которых $a_j \neq 0$ для данной хромосомы A_k. Решение задачи состоит в вычислении с помощью ГА *плана РЗ* D_ω *для каждого состояния* $s_\omega \in S^\omega$. Затем на основе всех полученных планов D_ω и плана начального РЗ формируется план $Z = \| z_{ji} \|$ результирующего *отказоустойчивого размещения задач* для всех ПМ как указано выше.

Общая блок-схема выполнения генетического алгоритма, формирующего план D_ω, приведена на Рис. 1. Рассмотрим его шаги.

1. Создание начальной популяции P^0 состоит в выполнении *операции инициализации* для *каждой* из ее N_0 хромосом, т.е. случайного распределения задач по р-ПМ некоторого данного состояния с одновременной проверкой заданных ограничений на объем памяти ПМ (3) и на максимальное суммарное время выполнения задач в каждом ПМ (2); N_0 задается пользователем. Алгоритм *инициализации одной хромосомы:*
   1) формирование случайного списка задач;
   2) выбор очередной задачи из этого списка, если он не пуст, в противном случае - окончание работы алгоритма;
   3) попытка размещения выбранной задачи в *i-й* р-ПМ данного состояния ($i = 1,\dots, g$,

g - число р-ПМ), начиная с $i = 1$, при проверке ограничений (2) и (3): данная задача назначается для решения в *i-й* ПМ, если ограничения (2) и (3) для него не нарушены, иначе – попытка разместить ее в *(i+1)-й* ПМ; процедура продолжается до наибольшего номера Р-ПМ;

4) если данную задачу не удается разместить ни в один ПМ - отбрасывание этой задачи и переход к шагу 2.

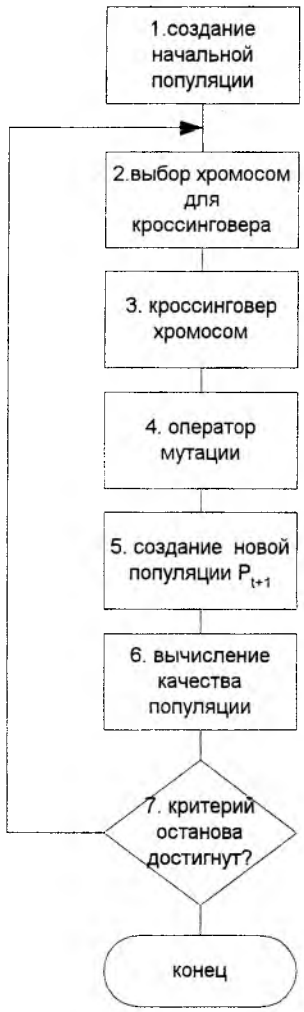

Рисунок 1 Общая блок-схема работы алгоритма

2. Выбор хромосом для кроссинговера. В данной реализации ГА отбор хромосом (репродукция) не выделен в отдельный этап, а совмещен с кроссинговером. Две различных хромосомы, являющиеся кандидатами для формирования пар, над которыми возможно выполнение кроссинговера, выбираются по принципу пропорциональной селекции ("алгоритм рулетки") непосредственно из предшествующей популяции (на первой итерации - из начальной), из них случайным образом формируются пары различных хромосом, и над каждой из этих пар с заданной вероятностью выполняется операция кроссинговера.

3. Операция кроссинговер над данной парой с заданной вероятностью $\mathbf{P_{cr}}$ реализуется следующим образом. Если случайно сгенерированное число $r < \mathbf{P_{cr}}$, то над данной парой выполняется кроссинговер. Потомки проверяются на ограничения и сравниваются с родительскими хромосомами. Если потомок не удовлетворяет ограничениям или если полезность потомка меньше полезности родительской хромосомы, то потомок отбрасывается и ищется новая родительская пара для кроссинговера. Если $r >= \mathbf{P_{cr}}$, то над данной парой кроссинговер не выполняется, а "несостоявшиеся родители" переходят в следующую промежуточную популяцию **R** (для выполнения следующего оператора - мутации). Если потомок удовлетворяет ограничениям и полезность потомка больше полезности родителя, то потомок переходят в промежуточную популяцию **R**, а родители отбрасываются.

Используется два варианта кроссинговера.

I. Одноточечный кроссинговер: значения первых *m* элементов первой строки записываются в соответствующие элементы второй, а значения *первых m* элементов второй строки - в соответствующие элементы первой (*m* - случайная точка разрыва).

II. Использование формируемого случайным образом битового вектора, элементы которого принимают значение 0 или 1. Принцип передачи наследственной информации от родителей к потомкам показан на Рис. 2.

Рисунок 2. Кроссинговер с помощью битого вектора

4. Мутация (одноточечная): для каждого элемента каждой хромосомы из популяции **R** с заданной вероятностью $\mathbf{P_{mut}}$ выполняется мутация: попытка обмена задачи U_m, соответствующей выбранному элементу A_m , на какую-либо из отброшенных задач U_s (т.е. назначенных согласно данной хромосоме в "фиктивный "ПМ с номером 0). Данная хромосома переходит на следующий этап алгоритма, если после некоторой попытки такого обмена она удовлетворяет ограничениям (2) и (3), иначе ищется другая задача из множества отброшенных задач. В случае неудачи таких попыток для всех отброшенных задач хромосома переходит на следующий этап без изменения.

Пусть $\mathbf{P^c_t}$ – дочерняя популяция, полученная из популяции $\mathbf{P_t}$ в результате *кроссинговера* ее особей и последующей *мутации*. В следующую (новую) популяцию $\mathbf{P_{t+1}}$ могут попасть как особи только дочерней популяции $\mathbf{P^c_t}$, так и *лучшие особи* из дочерней $\mathbf{P^c_t}$ и родительской $\mathbf{P_t}$ популяций. В последнем случае используется *элитизм*.

5. Создание новой популяции $\mathbf{P_{t+1}}$: хромосомы, полученные в результате операций *отбора, кроссинговера и мутации* над хромосомами текущей популяции $\mathbf{P_t}$, переходят в $\mathbf{P_{t+1}}$.
6. Вычисление качества популяции (необходимо для ведения статистики и для определения значения критерия останова): после каждого шага ГА вычисляется полезность $F(\mathbf{A_k})$ каждой хромосомы и полезность всей популяции, как сумма всех $F(\mathbf{A_k})$ при $k = 1,\ldots,N$.
7. Замена предыдущей популяции на новую и проверка критерия останова (Рис.1). Использованы критерии: (а) заданное максимальное количество популяций, (б) заданное максимальное значение $F(\mathbf{A_k})$ полезности хромосомы (заданный процент от суммы весов всех задач начального множества). Самая «полезная» хромосома в популяции, полученной перед остановом, является решением задачи.

Решение задачи 2. Используется процедура ГА, аналогичная представленной на Рис.1.

Функция приспособленности хромосомы A_k задана как

$$F\left(A_k\right) = 1 \Big/ \sum_{j=1}^{L} c^*_{ji},$$

где

$$c^*_{ji} = c_{ji}, \text{ если } a_j = wh = M_i,$$

$$c^*_{ji} = 0, \text{ если } a_j = 0$$

Каждая популяция $\mathbf{P_t}$ характеризуется *минимальным значением затрат* (т.е. затрат для лучшей хромосомы популяции) $\mathbf{C_{min}(P_t)}$ и средним значением затрат $\mathbf{C_{avg}(P_t)}$. Предусмотрено применение одноточечного кроссинговера с отбором особей для него путем *пропорциональной селекции* и трех видов мутации (по выбору пользователя): одноточечная (описана выше), двухточечная (случайный выбор в хромосоме двух позиций (задач), например r и s, и обмен между ними значениями $\mathbf{a_r}$ и $\mathbf{a_s}$ (т.е. значениями номеров ПМ, в которые назначаются эти задачи), направленная (в хромосоме случайным образом выбирается ген из множества генов, соответствующих отброшенным задачам, и его значение, т.е. 0, случайным образом заменяется одним из номеров реальных р-ПМ).

В качестве условия остановки работы алгоритма может быть принято достижение одного из следующих заданных значений: максимально допустимого значения $\mathbf{C_{min}^*(P_t)}$ минимальных затрат для популяции, минимально допустимой численности популяции N_t, максимального значения числа популяций, а так же ситуации, когда среднее значение затрат $\mathbf{C_{avg}(P_t)}$ не уменьшается для последних N_ξ популяций, где N_ξ задано. Можно также потребовать остановки алгоритма при выполнении хотя бы одного из нескольких заданных условий. Остановка алгоритма происходит всегда, если для некоторой популяции $\mathbf{C_{min}(P_t) = 0}$ (т.е. найдено наилучшее решение).

Результат работы алгоритма - хромосома $\mathbf{A^*}$ (план РЗ для данного состояния), которой соответствуют минимальные затраты из всех хромосом для всей эволюции.

Применение ГА для решения подобных задач отказоустойчивости РВС не обеспечивает гарантированного нахождения лучшего результата, но позволяет найти «достаточно хорошее» решение за «достаточно малое» время. Это особенно важно для "больших" систем, когда может потребоваться перераспределение тысяч задач по тысячам ПМ, причем, поскольку обеспечение высокой отказоустойчивости таких "больших" систем требует задания достаточно большого допустимого числа d отказавших ПМ, число таких планов РЗ чрезвычайно велико. Алгоритм наиболее эффективен в тех случаях, когда целевая функция может иметь не-

сколько локальных экстремумов и другие методы (в частности градиентный метод) могут остановиться на одном из них.

Для решения описанных выше задач РСПЗ/ОУ была разработана программа на языке С++ с использованием стандартных библиотек языка С++ и выполнены расчеты для ряда структур РВС.

Литература

[1].Турута Е.Н. Организация распределения задач в вычислительных системах, обеспечивающая их отказоустойчивость// Автоматика и вычисл. техника. 1985. № 1. С.5-14.

[2].Tourouta E.N. The methods for ensuring fault-tolerance of distributed control systems// IFAC Symp. on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS' 94). 1994, Helsinki, Finland. Preprints, v. 2, pp.772-777.

[3].Корнеев В.В., Гарев А.Ф., Васютин С.В., Райх В.В. Базы данных. Интеллектуальная обработка информации. М.: "Нолидж", 2000.

[4].Goldberg D.E. Genetic algorithms in search, optimization and machine learning. N.Y. Addison-Wesley Pub. Comp., 1989.

ИСПОЛЬЗОВАНИЕ ЭКСПЕРТНЫХ СИСТЕМ ДЛЯ ЗАЩИТЫ ИНФОРМАЦИОННО-ВЫЧИСЛИТЕЛЬНЫХ СЕТЕЙ ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА

А.М. Курносов

Аннотация

Рассматриваются вопросы использования экспертных систем для классификации информационного оружия, применяемого в телекоммуникационных сетях. Анализируются существующие методы задания классов состояний объекта, а также алгоритмы кластер-анализа. Алгоритм нечеткой кластеризации Данна-Бездека выбран, как наиболее оптимальный, позволяющий системе функционировать в условиях неопределенности.

В настоящее время широкое внедрение телекоммуникационных систем, прежде всего автоматизированных систем управления (АСУ) бизнес-процессами, становится определяющим фактором эффективности любой экономической структуры. Вместе с тем, как показывает мировая практика, растет потенциальная уязвимость экономики по отношению к информационным воздействиям.

АСУ бизнес-процессами представляет собой человеко-машинную систему, использующую современные экономико-математические методы, автоматические средства обработки данных (ЭВМ, автоматических устройств накопления, регистрации, отображения) и связи, а также новые организационные принципы управления. АСУ предназначена для принятия оптимальных решений по управлению производственно-хозяйственной деятельностью корпорации.

Одним из основных элементов системы управления, является информационно-вычислительная сеть (ИВС), объединяющая следующие элементы:

- сеть обмена данными;
- локальные вычислительные сети (ЛВС) и АСУ региональных представительств, подключенных к базовой сети;
- подсети рабочих групп в рамках подразделений.

К особенностям функционирования перспективной ИВС можно отнести:

- разнородность технических средств и программного обеспечения;
- высокая доля разнородных сетевых информационных технологий;
- расширение спектра услуг и, как следствие, увеличение объема и качественного разнообразия информации, хранимой, обрабатываемой и передаваемой в ИВС;
- увеличение числа пользователей ИВС.

Для перспективной ИВС любой корпорации может стать актуальным следующее:

- сопряжение с глобальными сетями типа Internet;
- сопряжение с сетями других корпораций в результате расширения международного сотрудничества.

Перечисленные особенности корпоративной ИВС, обуславливают высокую степень уязвимости информации, хранимой, обрабатываемой и передаваемой в такой сети.

Очевидно, что при создании единой ИВС корпорации в ее системе защиты информации