УДК 658.512

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ И МНОГОУРОВНЕВЫХ СЕТЕЙ ПЕТРИ ПРИ ПРОЕКТИРОВАНИИ КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЕЛЬЧАНИНОВ Д.Б., КРИВУЛЯ Г.Ф., ЛОБОДА В.Г., МЕХАНА САМИ

Рассматривается проблема формализации и алгоритмизации процесса генерации вариантов проектных решений в САПР. Формулируется задача адаптации генетических алгоритмов к решению поставленной задачи. Для представления генетических алгоритмов предлагается использовать многоуровневые структурированные L-сети Петри.

1. Введение

В настоящее время большинство задач проектирования компьютерной техники (КТ) относятся к многокритериальным. При их решении необходимо учитывать большое количество факторов. Кроме того, значительно обострилась конкуренция, и проектные решения надо принимать в режиме реального времени.

Принятие решения в проектировании КТ заключается в генерации возможных структурных и функциональных схем проектируемого вычислительного устройства, оценке этих схем и выборе среди них лучшей. "Правильное" решение – это такой вариант возможной схемы устройства, который оптимизирует некоторую целевую функцию (производительность, стоимость и т.п.) в пространстве рассматриваемых вариантов [1].

С появлением и развитием САПР КТ указанные процессы генерации различных вариантов архитектуры и выбора лучшего решения реализуются в рамках комплексных компьютерных систем проектирования электронных компонентов. Однако генерация производится непосредственно разработчиками "вручную", а не самой системой проектирования. В связи с этим актуальной является задача формализации и алгоритмизации процесса генерации различных вариантов проектных решений.

Формализация заключается в описании параметров схемной реализации вычислительного устройства, взаимодействия этих параметров, их ограничений и критерия, оценивающего альтернативные архитектуры. Критерием является целевая функция, определенная на декартовом произведении значений указанных выше параметров. И наилучшей проектной схеме соответствует экстремум целевой функции.

В общем случае генерация вариантов архитектур приводит к всевозможным комбинациям значений параметров, а выбор наилучшей структуры сводится к определению комбинации, на которой целевая функция принимает экстремальное значение. Размерность пространства поиска определяется числом параметров проектного решения, а точками этого пространства являются все варианты схемной реализации.

В настоящее время применяют три вида методов поиска наилучших решений [2,3]: использующие математические расчеты; перечислительные методы и применяющие элемент случайности.

Методы первой группы можно использовать для целевых функций, которые являются гладкими, дифференцируемыми и унимодальными. Однако формализация задач проектирования КТ редко приводит к целевым функциям такого вида.

Перечислительные методы эффективны при малой размерности пространства поиска. Но размерность задач проектирования современных вычислительных структур из-за увеличения степени интеграции непрерывно растет. При этом время, отводимое на принятие проектного решения, уменьшается. Поэтому данные методы используются мало.

В основе методов, применяющих элемент случайности, лежит случайный поиск в пространстве состояний. Хотя эти методы и не гарантируют получения оптимального решения, они эффективнее первых двух при решении сложных задач, в которых целью является поиск не оптимального решения, а лучшего по сравнению с существующим на данный момент. Именно к такому типу и относится большинство современных задач проектирования КТ.

Для повышения эффективности в методы случайного поиска вводят элементы детерминированности. Примером методов такого типа являются эволюционные вычисления [4], в основе которых лежат принципы природного эволюционного процесса. Эти методы, с одной стороны, моделируют естественные процессы размножения, наследования и отбора, которые происходят по строго детерминированным правилам, а с другой – случайные процессы мутации.

В эволюционных вычислениях существуют три основных направления: генетические алгоритмы, эволюционные стратегии и эволюционное программирование.

При использовании генетических алгоритмов альтернативные решения представляются в виде строки символов, имеющей фиксированную длину. При этом преобразования строки выполняются без какой-либо связи с ее семантическим содержанием.

В отличие от генетических алгоритмов, в эволюционных стратегиях воздействие на альтернативы производится с учетом их семантики.

Наконец, эволюционное программирование основано на описании альтернативных решений при помощи языка конечных автоматов, которые реагируют на внешние воздействия.

Эволюционные вычисления ориентированы не на обнаружение оптимального решения задачи, а на улучшение существующего решения за меньшее количество времени, чем это возможно при применении других методов оптимизации. Несмотря на это, они с успехом применялись для решения реальных задач инженерного проектирования, планирования, размещения, прогнозирования и т.п.

Строго говоря, эволюционные вычисления не являются алгоритмом решения оптимизационной задачи. Правильно было бы говорить об эволюционных вычислениях как о стратегическом подходе к организации оптимизационных процедур. Поэтому к каждому конкретному классу задач эволюционные вычисления необходимо адаптировать, выбирая специальные характеристики и параметры.

На сегодняшний день различные направления эволюционных вычислений настолько переплелись и взаимно проникли друг в друга, что уже рассматривают единую концепцию эволюционных вычислений.

Наиболее популярными и распространенными эволюционными вычислениями являются генетические алгоритмы. Поэтому, в свете сказанного выше, актуальной является задача адаптации генетических алгоритмов к решению задачи генерации различных вариантов реализации вычислительных структур и выбора среди них наилучшей в САПР КТ.

2. Генетические алгоритмы

Генетические алгоритмы отражают принципы естественного отбора и генетики: выживание наиболее перспективных особей, наследование и мутации. При этом человек не вмешивается в процесс поиска, но может опосредственно влиять на него заданием определенных параметров.

Как и все методы случайного поиска, генетические алгоритмы ориентированы не на нахождение оптимального решения, а на поиск лучшего, чем существующие на данный момент решения. Такой подход эффективен для сложных систем, где зачастую необходимо каким-то образом улучшить текущее решение, а задача поиска оптимального решения не ставится из-за сложности системы и, как следствие, невозможности применения традиционных методов, которые направлены на нахождение оптимальных решений.

Основные отличия генетических алгоритмов от традиционных методов заключаются в следующем [3,5]:

1. Генетические алгоритмы оперируют с решениями, представленными в виде кодовой строки. И преобразования кодов производятся вне какой-либо связи с их семантикой.

2. Процесс поиска основан на использовании нескольких точек пространства решений одновременно. Это устраняет возможность нежелательного попадания в локальный экстремум целевой функции, не являющейся унимодальной.

3. В процессе поиска генетическим алгоритмом используется только информация о допустимых значениях параметров и целевой функции, что приводит к значительному повышению быстродействия.

4. Для синтеза новых точек генетический алгоритм использует вероятностные правила, а для перехода от одних точек к другим – детерминированные. Такое объединение правил значительно эффективнее, чем их раздельное применение.

При этом в теории генетических алгоритмов используется ряд биологических терминов.

Кодовая строка, описывающая возможное решение, и ее структура называются *генотипом*. Интерпретация кода с позиции решаемой задачи – *фенотипом*. Для рассматриваемой нами предметной области фенотипом будет некоторое проектное решение в виде структурной схемы вычислительного устройства. Код также называют *хромосомой*.

Совокупность хромосом, одновременно используемых генетическим алгоритмом на каждом этапе поиска, называется *популяцией*. Размер популяции (число хромосом) обычно фиксируется и является одной из характеристик генетического алгоритма. Популяция обновляется созданием новых хромосом и уничтожением старых. Таким образом происходит смена *поколений* популяций.

Генерация новых хромосом основана на моделировании процесса размножения: пара *родителей* порождает пару *потомков*. За генерацию отвечает оператор *скрещивания*, который в общем случае применяется к каждой родительской паре с некоторой вероятностью. Значение этой вероятности наряду с размером популяции является одной из характеристик генетического алгоритма.

К хромосомам новой популяции применяется оператор *мутации*. Вероятность применения этого оператора к хромосоме также является параметром генетического алгоритма.

Оператор *отбора* осуществляет выбор родительских хромосом для порождения потомков, а оператор *редукции* – выбор хромосом, подлежащих уничтоже-

нию. В обоих случаях выбор делается на основании *качества* хромосомы, которое определяется значением целевой функции на этой хромосоме.

Генетический алгоритм прекращает свою работу в следующих случаях:

1. Он обработал число поколений, заданных пользователем перед началом работы алгоритма.

2. Качество всех хромосом превысило значение, заданное пользователем до начала работы алгоритма.

3. Хромосомы стали однородными до такой степени, что их улучшение от поколения к поколению происходит очень медленно.

Работа генетического алгоритма применительно к рассматриваемой предметной области проектирования КТ может быть описана следующим образом.

Начальное множество проектных решений формируется разработчиками КТ. Затем эти решения кодируются в хромосомы, с которыми и будет непосредственно работать генетический алгоритм. Уже на этом этапе возникает актуальная задача выбора способа представления структурной схемы проектируемого устройства в виде строки кода, обрабатываемой генетическим алгоритмом.

Выбор родительских хромосом для размножения и их уничтожение происходит в соответствии с природным принципом "выживает сильнейший". На данном этапе актуальна проблема задания операторов отбора и редукции, отражающих особенности процесса проектирования КТ. Похожая проблема возникает при задании оператора скрещивания, в котором необходимо моделировать передачу свойств между различными структурными схемами устройства.

В генетических алгоритмах часто используется стратегия *элитизма*, заключающаяся в переходе лучших хромосом текущей популяции в следующее поколение без изменений. Такой подход обеспечивает поддержание высокого уровня качества популяции.

Оператор мутации вносит случайные изменения в хромосомы, расширяя область пространства поиска.

Многократное применение операторов редукции, отбора, скрещивания и мутации способствует улучшению качества отдельной хромосомы и, как следствие, популяции в целом, отражая основную цель генетического алгоритма – повышение качества начальной популяции. Основным результатом работы генетического алгоритма является хромосома конечной популяции, на которой целевая функция принимает экстремальное значение.

3. Оптимизация генетических алгоритмов

Как было сказано выше, генетические алгоритмы являются стратегическим подходом к решению проблемы, который необходимо адаптировать к конкретной предметной области путем задания параметров и определения операторов генетического алгоритма. При этом генетический алгоритм становится сильно привязанным к рассматриваемой предметной области и может быть совершенно бесполезен для решения задач в другой предметной области.

От удачного выбора параметров, операторов и вида хромосом зависят устойчивость и скорость поиска – основных показателей эффективности генетического алгоритма. Скорость определяется временем, необходимым для достижения алгоритмом одного из указанных выше критериев останова. Устойчивость – это способность генетического алгоритма увеличивать качество популяции и выходить из локальных экстремумов.

Для увеличения скорости генетические алгоритмы могут подвергаться *распараллеливанию* [3] как на уровне организации работы алгоритма, так и на уровне его реализации на ЭВМ.

На уровне организации работы распараллеливание осуществляется путем структурирования популяции, которое может осуществляться двумя способами.

Первый способ называется "концепция островов" и заключается в разбиении популяции на классы (*демосы*), члены которых скрещиваются только между собой в пределах класса, лишь изредка обмениваясь хромосомами на основе случайной выборки. Второй способ называется "концепция скрещивания в локальной области" и заключается в задании метрического пространства на популяции, хромосомы которой подвергаются скрещиванию только с ближайшими соседями.

Таким образом, учитывая возможность *параллельной* работы генетического алгоритма на *структурированной* популяции, актуальной является задача его представления на *структурированных сетях Петри* [6].

Что касается распараллеливания на уровне реализации, то как указанные выше процессы скрещивания пар родителей, так и процессы вычисления значений целевой функции и применения оператора мутации к хромосомам можно реализовать одновременно на нескольких параллельно работающих процессорах или системах. И если на уровне организации рассматривать сети Петри как способ описания генетических алгоритмов, то актуальной становится задача их реализации на параллельных *процессорах Петри* [7,8].

Устойчивость поиска зависит от параметров операторов генетического алгоритма.

Для оператора скрещивания таким параметром служит степень отличия потомков от родительских хромосом: чем больше это отличие, тем устойчивей поиск, но скорость поиска меньше (лучший результат достигается за более продолжительное время).

Для оператора мутации параметром, влияющим на устойчивость поиска, служит вероятность его применения: малая вероятность обеспечивает устойчивый поиск и не приводит к ухудшению качества хромосом.

Оператор отбора связан с устойчивостью поиска следующим образом: постоянный выбор сильнейших хромосом обычно приводит к сходимости к локальному экстремуму, а выбор слабых хромосом – к ухудшению качества популяции. Аналогичное утверждение справедливо и для оператора редукции.

Что касается влияния размера популяции на устойчивость генетического алгоритма, то увеличение числа хромосом в популяции расширяет область поиска, но при этом время от времени полезно редуцировать популяцию до первоначального размера, иначе скорость генетического алгоритма резко упадет. Подобные алгоритмы называются *поколенческими*.

Развитие поколенческих алгоритмов привело к появлению *адаптивных* генетических алгоритмов, изменяющих свои параметры в процессе работы. Возникла концепция nGA, представляющая *многоуровневые* генетические алгоритмы, в которых нижний уровень улучшает популяцию, а верхний – оптимизирует параметры нижнего уровня, ориентируясь при этом на его скорость и устойчивость.

Таким образом, учитывая возможность многоуровневой организации генетического алгоритма, актуальной является задача его реализации на многоуровневых *L-сетях Петри* [9-11].

Учитывая все сказанное в пунктах 2 и 3, можно указать следующие общие свойства сетей Петри и генетических алгоритмов, обосновывающих преимущество использования сетей Петри для описания генетических алгоритмов.

1. Независимость от предметной области.
2. Параллелизм.
3. Вероятность и детерминированность.
4. Структурированность.
5. Многоуровневость.

4. Структурированные L-сети Петри

L-сеть Петри представляет собой иерархическую многоуровневую сетевую модель, в которой меткам сети Петри верхнего уровня сопоставлены сети Петри нижнего уровня. Структура сети Петри верхнего уровня определяет алгоритм изменения структур сетей Петри нижнего уровня.

Для описания структуры сети Петри любого уровня используется алгебраический подход, основанный на отношениях эквивалентности на множестве дуг сети Петри. Применение отношения эквивалентности для описания структуры сети Петри, вообще говоря, приводит к обычной сети Петри. Однако такой подход позволяет легко адаптировать структуру L-сети Петри к особенностям произвольной предметной области, задавая произвольные отношения на множестве дуг, что при традиционном подходе к описанию структуры обычной сети Петри вызывает принципиальные трудности.

Первый (нижний) уровень L-сети Петри описывается следующим образом.

Пусть A – множество *дуг*. На множестве A задается отношение эквивалентности ρ. Класс эквивалентности ρ называется *позицией*. Таким образом, фактормножество $A/\rho = \{p_i\}_{i=1}^{n}$ является множеством позиций. Затем на множестве A задается отношение эквивалентности τ. Класс эквивалентности τ называется *переходом*. Значит, фактормножество $A/\tau = \{t_j\}_{j=1}^{m}$ является множеством переходов.

Для каждого класса p_i задается разбиение $p_i = I(p_i) \cup O(p_i)$. Множество $I(p_i)$ называется множеством *входных дуг для позиции* p_i, а множество $O(p_i)$ – множеством *выходных дуг для позиции* p_i. Затем для каждого класса t_j задается разбиение $t_j = I(t_j) \cup O(t_j)$. Множество $I(t_j)$ называется множеством *входных дуг для перехода* t_j, а множество $O(t_j)$ – множеством *выходных дуг для перехода* t_j. Разбиение классов эквивалентностей должно удовлетворять следующим условиям:

1. Для любой дуги $a \in I(p_i)$ существует переход $t_j \in A/\tau$, такой, что $a \in O(t_j)$.
2. Для любой дуги $a \in O(p_i)$ существует переход $t_j \in A/\tau$, такой, что $a \in I(t_j)$.
3. Для любой дуги $a \in I(t_j)$ существует позиция $p_i \in A/\rho$, такая, что $a \in O(p_i)$.
4. Для любой дуги $a \in O(t_j)$ существует позиция $p_i \in A/\rho$, такая, что $a \in I(p_i)$.

Пусть N_0 – множество целых неотрицательных чисел. На множестве N_0 рассматриваются следующие отношения: $<$, $>$, $\neq$, $=$, $\leq$, $\geq$. Через R обозначается множество этих отношений: $R = \{<, >, \neq, =, \leq, \geq\}$.

Задается функция *типа* $V: A \to R \cup \{+\}$. Если $V(a) \in R$, то дуга a называется *аналитической* [12]. Если $V(a) = "+"$, то дуга a называется *транспортной*.

Функция этого типа должна удовлетворять следующему условию: если $a \in O(t_j)$, то $V(a)="+"$.

Задается функция *веса* $W: A \to N_0$, такая, что если $V(a)="+"$, то $W(a) \neq 0$.

Наконец, задается функция *маркировки* $M: A/\rho \to N_0$.

Совокупность $N = (A, A/\rho, A/\tau, V, W, M)$ называется *сетью Петри* N.

Транспортная дуга $a \in I(t_j) \cap O(p_i)$ разрешает переход t_j, если $W(a) \le M(p_i)$. Аналитическая дуга $a \in I(t_j) \cap O(p_i)$ разрешает переход t_j, если $(M(p_i), W(a)) \in V(a)$. Переход t_j *разрешен*, если все входные дуги t_j разрешают его. Разрешенный переход может запуститься (сработать). Запуск перехода t_j изменяет маркировку M сети Петри N на новую маркировку M^* следующим образом:

$$M^*(p_i) = M(p_i) - W(a_1) + W(a_2),$$

где $a_1 \in I(t_j) \cap O(p_i)$, $a_2 \in O(t_j) \cap I(p_i)$ и $V(a_1)="+"$.

Транспортная дуга $a \in I(t_j) \cap O(p_i)$ называется *конвейерной*, если $W(a) = 0$. Входная конвейерная дуга всегда разрешает переход t_j. У перехода может быть только одна входная конвейерная дуга. При запуске перехода t_j входная конвейерная дуга a удаляет $M(p_i)$ меток из позиции p_i.

Транспортная дуга $a \in O(t_j) \cap I(p_i)$ называется *конвейерной*, если $W(a) = 0$. Если у перехода t_j есть выходная конвейерная дуга, то у него должна быть входная конвейерная дуга. При запуске перехода t_j выходная конвейерная дуга a добавляет в позицию p_i столько меток, сколько удалила из соответствующей позиции входная конвейерная дуга перехода t_j.

Сеть Петри второго уровня описывается следующим образом.

Структура сети Петри N_C второго уровня определяется аналогично структуре сети Петри первого уровня:

$$N_C = (A_C, A_C/\rho_C, A_C/\tau_C),$$

где A_C – множество дуг сети Петри; ρ_C и τ_C – отношения эквивалентности на множестве A_C; A_C/ρ_C и A_C/τ_C – множества позиций и переходов.

Пусть $M_C = \{m_i\}_{i=1}^k$ – множество меток сети Петри N_C. Распределение меток по позициям задается начальной функцией маркировки:

$$Z_C^0 : M_C \to A_C/\rho_C .$$

Каждой метке m_i с помощью биективного отображения f_1 сопоставляется множество $P_S^i = \{p_{S,j}^i\}_{j=1}^{n_i}$ позиций первого уровня:

$$f_1 : M_C \to P_S, \text{ где } P_S = \{P_S^i\}_{i=1}^k .$$

Затем каждой метке m_i с помощью биективного отображения f_2 сопоставляется множество $N_S^i = \{N_{S,j}^i\}_{j=1}^{k_i}$ сетей Петри $N_{S,j}^i$ первого уровня с общим множеством P_S^i позиций первого уровня:

$$f_2 : M_C \to N_S, \text{ где } N_S = \{N_S^i\}_{i=1}^k .$$

После этого каждой метке m_i с помощью биективного отображения f_3 сопоставляется начальная сеть Петри N_{S,j_i}^i из множества N_S^i соответствующих сетей Петри первого уровня:

$$f_3 : M_C \to N_{S_0}, \text{ где } N_{S_0} = \{N_{S,j_i}^i\}_{i=1}^k .$$

Наконец, каждой метке m_i с помощью биективного отображения f_4 сопоставляется начальная маркировка M_S^i множества P_S^i позиций первого уровня:

$$f_4 : M_C \to M_S, \text{ где } M_S = \{M_S^i\}_{i=1}^k .$$

Таким образом, метке сети Петри второго уровня сопоставлена сеть Петри первого уровня. Такая метка называется *макрометкой* [9, 13].

Множество A_C дуг сети Петри второго уровня разбивается на подмножество A_C^1 входных дуг и подмножество A_C^2 выходных. Каждой входной дуге a сети Петри второго уровня с помощью отображения f_5 сопоставляется некоторая макрометка m_i из множества M_C меток сети Петри второго уровня:

$$f_5 : A_C^1 \to M_C .$$

Затем, если входной дуге a сети Петри второго уровня сопоставлена макрометка m_i из множества M_C меток сети Петри второго уровня, то этой дуге a с помощью отображения f_6 сопоставляется некоторая маркировка M_S^i множества позиций P_S^i первого уровня, сопоставленного макрометке m_i:

$$f_6 : A_C^1 \to M_S .$$

Входная дуга $a \in I(t_j) \cap O(p_k)$ сети Петри второго уровня разрешает переход t_j, если в позиции p_k находится макрометка m_i, соответствующая дуге a, и маркировка M_S^i позиций P_S^i первого уровня, соответствующих макрометке m_i, сопоставлена дуге a:

$$f_1(m_i) = p_k, \; f_5(a) = m_i, \; f_4(m_i) = M_S^i, \; f_6(a) = M_S^i .$$

Переход второго уровня *разрешен*, если все входные дуги перехода t_j разрешают его.

Каждой выходной дуге a сети Петри второго уровня с помощью отображения f_7 сопоставляется некоторая макрометка m_i из множества M_C меток сети Петри второго уровня:

$$f_7 : A_C^2 \to M_C .$$

Затем, если выходной дуге a сети Петри второго уровня сопоставлена макрометка m_i из множества M_C меток сети Петри второго уровня, то этой дуге a с помощью отображения f_8 сопоставляется некоторая сеть Петри $N_{S,j}^i$ из множества N_S^i, сопоставленного макрометке m_i:

$$f_8 : A_C^2 \to N_S .$$

Если в переход входит дуга a, которой соответствует макрометка m_i, то из перехода должна выходить дуга b, которой также соответствует макрометка m_i. Если из перехода выходит дуга b, которой соответствует макрометка m_i, то в переход должна входить дуга a, которой также соответствует макрометка m_i. В переход не может входить две дуги, которым сопоставлена одна макрометка. Аналогично, из перехода не могут выходить две дуги, которым сопоставлена одна макрометка.

Если переход разрешен, то он может быть *запущен*. При запуске перехода t_j макрометка перемещается из входной позиции по соответствующей входной дуге, проходит через переход t_j и по соответствующей выходной дуге помещается в выходную позицию перехода t_j. При этом сеть Петри, которая соответствовала макрометке до запуска перехода, заменяется сетью Петри, сопоставленной выходной дуге, по которой макрометка попала в выходную позицию перехода.

Приведем пример 2-сети Петри.

Пусть $A_C = \{a_c^i\}_{i=1}^8$ – множество дуг сети Петри второго уровня. Зададим на множестве A_C отношение эквивалентности ρ_C, определяемое следующим разбиением:

$$A_C = p_c^1 \cup p_c^2 \cup p_c^3 ,$$

где $p_c^1 = \{a_c^1, a_c^4, a_c^5, a_c^8\}$, $p_c^2 = \{a_c^6, a_c^7\}$ и $p_c^3 = \{a_c^2, a_c^3\}$.

Затем зададим на множестве A_C отношение эквивалентности τ_C, определяемое следующим разбиением:

$$A_C = t_c^1 \cup t_c^2 \cup t_c^3 \cup t_c^4 ,$$

здесь $t_c^1 = \{a_c^5, a_c^6\}$, $t_c^2 = \{a_c^7, a_c^8\}$,

$$t_c^3 = \{a_c^1, a_c^2\},\ t_c^4 = \{a_c^3, a_c^4\}.$$

Входные и выходные дуги определим следующим образом:

$$a_c^1 \in I(t_c^3) \cap O(p_c^1),\ a_c^2 \in O(t_c^3) \cap I(p_c^3),$$

$$a_c^3 \in I(t_c^4) \cap O(p_c^3),\ a_c^4 \in O(t_c^4) \cap I(p_c^1),$$

$$a_c^5 \in I(t_c^1) \cap O(p_c^1),\ a_c^6 \in O(t_c^1) \cap I(p_c^2),$$

$$a_c^7 \in I(t_c^2) \cap O(p_c^2),\ a_c^8 \in O(t_c^2) \cap I(p_c^1).$$

Затем определим множество макрометок $M_C = \{m_1, m_2\}$, а их распределение по позициям зададим начальной функцией маркировки

$$Z_C^0 : M_C \to A_C/\rho_C ,$$

такой, что $Z_C^0(m_1) = p_c^1$ и $Z_C^0(m_2) = p_c^2$.

Структура сети Петри второго уровня с начальной маркировкой показана на рис. 1.

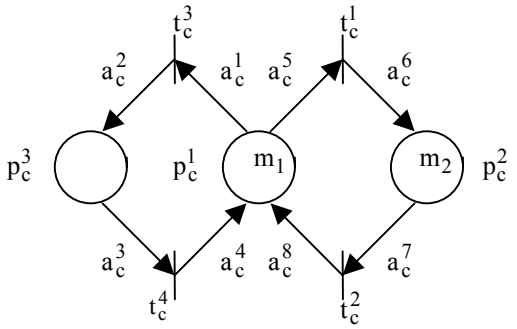

Рис.1. Сеть Петри второго уровня

Каждой макрометке с помощью биективного отображения f_1 сопоставим множество позиций первого уровня:

$$f_1 : M_C \to P_S ,$$

где $P_S = \{P_S^1, P_S^2\}$, $P_S^1 = \{p_{S,1}^1, p_{S,2}^1\}$, $P_S^2 = \{p_{S,1}^2, p_{S,2}^2\}$, $f_1(m_1) = P_S^1$, $f_1(m_2) = P_S^2$.

Затем каждой макрометке m_i с помощью биективного отображения f_2 сопоставим множество сетей Петри первого уровня:

$$f_2 : M_C \to N_S ,$$

здесь

$$N_S = \{N_S^1, N_S^2\},$$

$$N_S^1 = \{N_{S,1}^1, N_{S,2}^1\},\ N_S^2 = \{N_{S,1}^2, N_{S,2}^2\},$$

$$N_{S,1}^1 = (P_S^1, T_S^1, I_{S,1}^1, O_{S,1}^1),\ N_{S,2}^1 = (P_S^1, T_S^1, I_{S,2}^1, O_{S,2}^1),$$

$$N_{S,1}^2 = (P_S^2, T_S^2, I_{S,1}^2, O_{S,1}^2),\ N_{S,2}^2 = (P_S^2, T_S^2, I_{S,2}^2, O_{S,2}^2),$$

$$T_S^1 = \{t_S^1\},\ T_S^2 = \{t_S^2\},$$

$$I_{S,1}^1(t_S^1) = p_{S,1}^1,\ I_{S,2}^1(t_S^1) = p_{S,2}^1,$$

$$I_{S,1}^2(t_S^2) = p_{S,1}^2,\ I_{S,2}^2(t_S^2) = p_{S,2}^2,$$

$$O_{S,1}^1(t_S^1) = p_{S,2}^1,\ O_{S,2}^1(t_S^1) = p_{S,1}^1,$$

$$O_{S,1}^2(t_S^2) = p_{S,2}^2,\ O_{S,2}^2(t_S^2) = p_{S,1}^2,$$

$$f_2(m_1) = N_S^1,\ f_2(m_2) = N_S^2.$$

После этого каждой макрометке m_i с помощью биективного отображения f_3 сопоставим начальную сеть Петри из множества соответствующих сетей Петри первого уровня:

$$f_3 : M_C \to N_{S_0},$$

где $N_{S_0} = \{N_{S,1}^1, N_{S,1}^2\},\ f_3(m_1) = N_{S,1}^1,\ f_3(m_2) = N_{S,1}^2$.

Наконец, каждой макрометке m_i с помощью биективного отображения f_4 сопоставим начальную маркировку множества позиций первого уровня:

$$f_4 : M_C \to M_S,$$

здесь

$$M_S = \{M_S^1, M_S^2\},$$

$$M_S^1(p_{S,1}^1) = 3,\ M_S^1(p_{S,2}^1) = 0,$$

$$M_S^2(p_{S,1}^2) = 1,\ M_S^2(p_{S,2}^2) = 2,$$

$$f_4(m_1) = M_S^1,\ f_4(m_2) = M_S^2.$$

Таким образом, макрометкам сети Петри второго уровня сопоставлены сети Петри первого уровня, как показано на рис. 2.

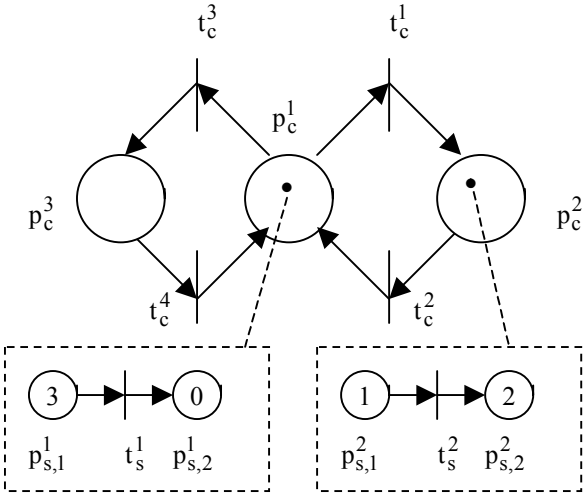

Рис.2. Пример 2-сети Петри

Каждой входной дуге сети Петри второго уровня с помощью отображения f_5 сопоставим некоторую макрометку m_i из множества M_C меток сети Петри второго уровня:

$$f_5 : A_C^1 \to M_C,$$

где $f_5(a_c^1) = m_1,\ f_5(a_c^3) = m_1,\ f_5(a_c^5) = m_2,\ f_5(a_c^7) = m_2$.

Затем входной дуге a сети Петри второго уровня с помощью отображения f_6 сопоставим маркировку множества позиций первого уровня, сопоставленного макрометке m_i:

$$f_6 : A_C^1 \to M_S,$$

здесь

$$f_6(a_c^1) = M_S^3,\ f_6(a_c^3) = M_S^1,$$

$$f_6(a_c^5) = M_S^2,\ f_6(a_c^7) = M_S^4,$$

$$M_S^3(p_{S,1}^1) = 2,\ M_S^3(p_{S,2}^1) = 1,$$

$$M_S^4(p_{S,1}^2) = 0,\ M_S^4(p_{S,2}^2) = 3.$$

Каждой выходной дуге a сети Петри второго уровня с помощью отображения f_7 сопоставим некоторую макрометку m_i из множества M_C меток сети Петри второго уровня:

$$f_7 : A_C^2 \to M_C,$$

где $f_7(a_c^2) = m_1,\ f_7(a_c^4) = m_1,$

$$f_7(a_c^6) = m_2,\ f_7(a_c^8) = m_2.$$

Затем выходной дуге a сети Петри второго уровня с помощью отображения f_8 сопоставим некоторую сеть Петри из множества сетей Петри, сопоставленных макрометке m_i:

$$f_8 : A_C^2 \to N_S,$$

здесь $f_8(a_c^2) = N_{S,2}^1,\ f_8(a_c^4) = N_{S,1}^1,$

$$f_8(a_c^6) = N_{S,1}^2,\ f_8(a_c^8) = N_{S,2}^2.$$

Проследим за перемещением макрометок 2-сети Петри и изменением структур соответствующих им сетей Петри первого уровня.

Когда запустится переход t_s^1, маркировка сети $N_{S,1}^1$ станет равна (2,1), и макрометка m_1 разрешит запуск перехода t_c^3, который удалит ее из позиции p_c^1 и добавит в позицию p_c^3. При этом дуга a_c^2 сопоставит макрометке m_1 сеть Петри $N_{S,2}^1$. Затем,

когда запустится переход t_s^1 и маркировка сети $N_{S,2}^1$ станет равна (3,0), макрометка m_1 разрешит запуск перехода t_c^4, который удалит ее из позиции p_c^3 и добавит в позицию p_c^1. При этом дуга a_c^4 сопоставит макрометке m_1 сеть Петри $N_{S,1}^1$.

Аналогично, когда запустится переход t_s^2, маркировка сети $N_{S,1}^2$ станет равна (0,3), и макрометка m_2 разрешит запуск перехода t_c^2, который удалит ее из позиции p_c^2 и добавит в позицию p_c^1. При этом дуга a_c^8 сопоставит макрометке m_2 сеть Петри $N_{S,2}^2$. Затем, когда запустится переход t_s^2 и маркировка сети $N_{S,2}^2$ станет равна (1,2), макрометка m_2 разрешит запуск перехода t_c^1, который удалит ее из позиции p_c^1 и добавит в позицию p_c^2. При этом дуга a_c^6 сопоставит макрометке m_2 сеть Петри $N_{S,1}^2$.

Таким образом, 2-сеть Петри представляет собой иерархическую двухуровневую сетевую модель, в которой меткам сети Петри верхнего (управляющего) уровня сопоставлены сети Петри нижнего (исполнительного) уровня. Структура сети Петри управляющего уровня определяет алгоритм изменения структур сетей Петри исполнительного уровня.

Макрометке 2-сети Петри можно сопоставить сеть Петри любого класса. Тип класса зависит от множеств значений атрибутов. Например, множества входных и выходных функций определяют кратность дуг и наличие петель в сетях Петри.

Если макрометке 2-сети Петри в свою очередь сопоставить 2-сеть Петри, то получим 3-сеть Петри [9, 13]. Первый (исполнительный) уровень этой сети Петри представляют макрометки 2-сетей Петри, второй (управляющий) уровень – макрометки 3-сети Петри, третий (управляющий) уровень – 3-сеть Петри.

L-сеть Петри строится на основании (L−1)-сетей Петри аналогично тому, как на основе 2-сети Петри строится 3-сеть Петри: если макрометке 2-сети Петри сопоставить (L−1)-сеть Петри, то получится L-сеть Петри [9, 13].

Таким образом, число уровней L-сети Петри зависит от сложности моделируемой системы. (L−1)-сети Петри моделируют подсистемы, на которые первоначально разбита вся система. Эти подсистемы в свою очередь состоят из подсистем, моделируемых (L−2)-сетями Петри и т.д. до сетей Петри, моделирующих подсистемы самого нижнего исполнительного уровня иерархии.

5. Постановка проблемы

Исходя из сказанного выше, перспективной для исследования представляется проблема формализации и алгоритмизации процесса генерации различных вариантов проектных решений. Предлагается адаптировать генетические алгоритмы к решению задачи генерации различных вариантов реализации вычислительных структур и выбора среди них наилучшей в системах автоматизированного проектирования вычислительной техники. В качестве способа представления структурной схемы проектируемого устройства в виде строки кода, обрабатываемой генетическим алгоритмом, предлагается использовать многоуровневые структурированные L-сети Петри, аппаратная реализация которых возможна на параллельных процессорах Петри.

Поставленная проблема включает в себя следующие основные задачи:

1. Описание начальных вариантов структурных схем проектируемых устройств на языке сетей Петри.

2. Описание генетического алгоритма (операторов отбора, скрещивания, редукции и мутации) на языке сетей Петри.

3. Описание оптимизации параметров операторов генетического алгоритма на языке сетей Петри.

4. Описание процесса генерации вариантов структурных схем проектируемых устройств на языке многоуровневых 3-сетей Петри, объединяющих сети Петри, полученные на этапах решения задач 1-3.

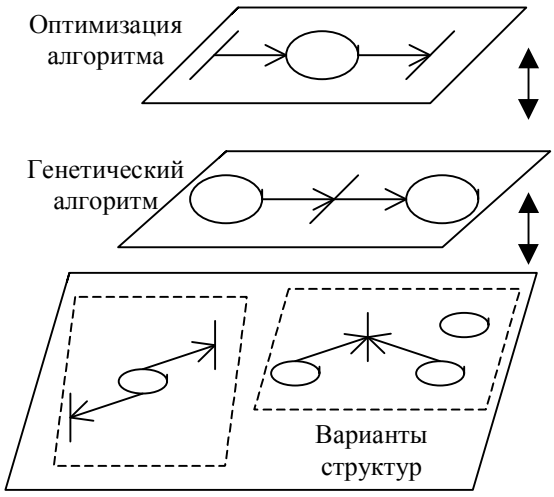

Рис.3. Генерация структурных схем на основе 3-сети Петри

Полученную в результате 3-сеть Петри графически можно представить так, как показано на рис. 3.

Таким образом, 3-сеть Петри должна выполнять следующие функции:

1. Получать на первый уровень начальные структурные схемы проектируемого устройства.

2. Улучшать эти схемы при помощи генетического алгоритма, расположенного на втором уровне.

3. Оптимизировать на третьем уровне работу генетического алгоритма.

4. Выдавать на первый уровень улучшенные схемы проектируемого устройства.

Перспективность поставленной проблемы и предлагаемых подходов к ее решению обусловлена новой концепцией проектирования, которая выражается в лозунге "Компьютер – это сеть", и состоит в том, что современный персональный компьютер рассматривается как совокупность совместимых независимых устройств, находящихся под управлением тонких серверов [14].

Практически эта концепция уже сейчас реализуется в так называемых системах на чипе - System On Chip (SOC). При этом SOC одного семейства (например, Linux On Chip) являются комплектом высокоуровневых модулей, на основе которых можно создавать всевозможные интегрированные однокристальные компьютеры для применения в различных предметных областях.

"Пересчитать производимые разными компаниями системы на чипе очень трудно, добиться же актуальности подобного материала практически невозможно. Чуть ли не каждый день появляются анонсы новых продуктов, проектов и, что самое интересное, новых подходов к формированию инфраструктуры разработки и производства SOC" [14, с. 39].

Литература: 1. *Трахтенгерц Э.А.* Компьютерная поддержка принятия решений. М.: Синтег, 1998. 2. *Васильев Ф.П.* Численные методы решения экстремальных задач: Учеб. пособие для вузов. М.: Наука, 1988. 552с. 3. *Goldberg D.E.* Genetic algorithms in search, optimization and machine learning. Adison Wesley, Reading, MA, 1989. 4. *Эволюционные* вычисления и генетические алгоритмы. Обозрение прикладной и промышленной математики. Вып. 5. М.: ТВП. Т.3. 1996. 5. *Корнеев В.В., Гареев А.Ф., Васютин С.В., Райх В.В.* Базы данных. Интеллектуальная обработка информации. М.: Нолидж, 2000. 352с. 6. *Мурата Т.* Сети Петри: свойства, анализ, приложения // ТИИЭР. 1989. Т. 77. № 4. С. 41-85. 7. *Патент* України на винахід №15213 А. Процесор Петрі / В.В. Матейченко, Г.О. Калінін, О.М. Маркін, С.Ю. Запорожцев, О.М. Гуца // Офіційний бюлетень "Промислова власність". К.: Держпатент України, 1997. №3. С. 3.1.324-3.1.325. 8. *Патент* України на винахід №32183 А. Процесор Петрі / Д.Б. Єльчанінов, В.В. Матейченко, В.Г. Лобода, Ю.С. Петришин // Бюл. №7-II. 9. *Ельчанинов Д. Б.* Моделирование иерархических структур *L*-сетями Петри // НТЖ. Автоматизация, телемеханизация и связь в нефтяной промышленности. М.: ВНИИОЭНГ, 1998. № 2. С. 7–8. 10. *Лобода В.Г., Ельчанинов Д.Б., Цуканов В.Ю.* Модели архитектуры MISC-процессора // Радиоэлектроника и информатика. 1999. № 1. С. 85–89. 11. *Єльчанінов Д.Б.* Структуровані мережі Петрі у системах проектування спеціалізованих процесорів. Автореф. дис. канд. техн. наук. Харків, 2000. 20 с. 12. *Маркін О.М.* Спецпроцесор модифікованих керуючих мереж Петрі для організації безпечного руху на залізничному транспорті. Автореф. дис. канд. техн. наук. Харків, 1997. 22 с. 13. *Ельчанинов Д.Б.* Микропроцессорные иерархические системы управления на базе *L*-сетей Петри // Сб. статей "Актуальные проблемы современной науки в исследованиях молодых ученых г. Харькова". Х.: АО "Бизнес Информ", 1998. С. 20-23. 14. *Зубинский А.* "Первый вал" SOC // еженедельник "Компьютерное обозрение". Киев, 2001. №1-2(271). 17-23 января. С.38-40.

Поступила в редколлегию 23.01.2002

Рецензент: д-р техн. наук, проф. Загарий Г.И.

Ельчанинов Дмитрий Борисович, канд. техн. наук, старший преподаватель кафедры программного обеспечения ЭВМ, сотрудник научно-учебной лаборатории "Приобретение знаний" ХНУРЭ. Научные интересы: математическое моделирование. Увлечения и хобби: музыка. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-95-91.

Кривуля Геннадий Федорович, д-р техн. наук, профессор, заведующий кафедрой автоматизации проектирования вычислительной техники, декан факультета компьютерной инженерии и управления ХНУРЭ. Научные интересы: техническая диагностика, системы автоматизированного проектирования цифровых устройств. Хобби: автомобилизм, туризм, рыбная ловля. Адрес: Украина, 61166 , Харьков, пр. Ленина, 14, тел. (0572) 40-93-26.

Лобода Виталий Гаврилович, канд. техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: коммутация вычислительных структур. Увлечения: мотоцикл. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26.

Механа Сами, аспирант кафедры АПВТ ХНУРЭ. Научные интересы: техническая диагностика компьютерных систем. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 40-93-26.