УДК 681.518

ВЗАИМОДЕЙСТВИЕ КОМПОНЕНТ В РАСПРЕДЕЛЁННЫХ ГЕНЕТИЧЕСКИХ АЛГОРИТМАХ ГЕНЕРАЦИИ ТЕСТОВ

Иванов Д.Е., к.т.н., с.н.с., доцент, Институт Прикладной Математики и Механики НАН Украины,
Чебанов П.А., ассистент кафедры ИТС Луганского Национального Университета им. Т.Шевченко
E-mail: *ivanov@iamm.ac.donetsk.ua, pavel.tchebanoff@gmail.com*

Abstract

In this paper the problem of construction of the distributed genetic algorithms that based on the «islands» model is discussed. In this model one calculation system is nominated as server and manage the interaction among the clients. Direct search of the problem solution is performed on the clients' machines. The algorithm of the work of the server and clients are considered in detail. Also we propose the schema of the synchronous interaction between the main server and «islands».

1. Введение

Перед разработчиком цифровых схем стоит целый ряд задач построения идентифицирующих входных последовательностей. К ним относятся тестовые, инициализирующие, верифицирующие эквивалентность и т.п. Наиболее сложной из них является задача построения тестов для цифровых последовательностных схем. Для этой задачи разработаны структурные методы, являющиеся расширением для последовательностных схем метода ветвей и границ [1], методы, основанные на символьном моделировании [2], а также псевдослучайные методы [3].

Генетические алгоритмы (ГА) [4] успешно используются при генерации проверяющих тестов цифровых схем [5] (с 90-х годов) наряду с детерминированными структурными методами. Опыт показывает, что генетические алгоритмы дают лучшие результаты для схем, ориентированных на обработку данных, в то время как детерминированные методы более успешно работают для сильно последовательностных схем со сложной управляющей логикой.

Известно, что генерация проверяющих тестов является NP-трудной задачей, т.е. в худшем случае решается путем перебора [5]. Поэтому, несмотря на то, что разработано множество последовательных структурных методов генерации тестов, которые для многих классов схем дают хорошие результаты, были предприняты попытки параллелизации алгоритмов построения тестов [6,7].

Однако рассмотренные подходы либо являются поверхностно описанными, либо имеют узкую направленность (например, рассчитаны на определённый тип оборудования в кластере). В данной статье авторы пытаются подробно описать работу компонент распределённой системы вычислений, реализующих модель островов распределённых генетических алгоритмов (РГА) построения тестов.

Данная статья посвящена вопросам построения алгоритмов работы серверной и клиентской частей распределённых алгоритмов, а также вопросам их корректного взаимодействия. Статья имеет следующую структуру. Во введении подчёркивается актуальность исследования. Следующий раздел посвящён описанию модели островов РГА. В третьем и четвёртом разделах описаны алгоритмы работы серверной и клиентской частей распределённого генетического алгоритма, основанного на модели островов. Пятый раздел посвящён вопросам взаимодействия компонент распределённых вычислений

рассматриваемой модели. В заключении делаются выводы и указываются направления дальнейших исследований.

2. Модель островов распределённых генетических алгоритмов

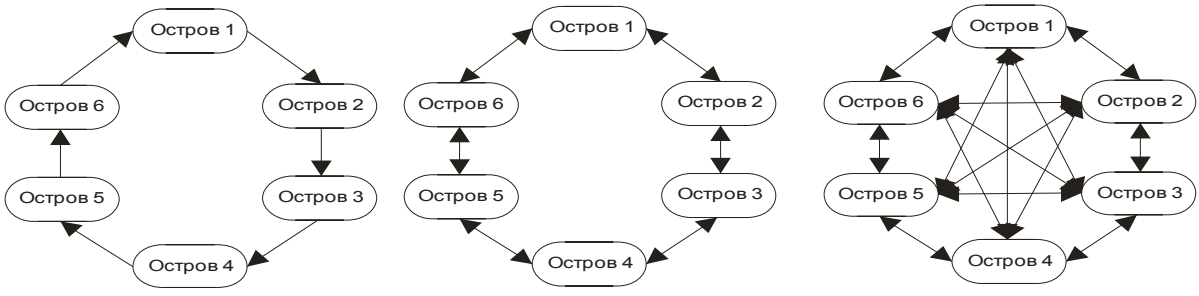
Первоначально, попытки построения распределённых ГА строились на модели «мастер-хозяин». В такой модели существует только одна копия ГА, которая выполняется на главном процессоре, который называется «хозяином». Остальные процессоры в узле осуществляют второстепенные функции: на них лежит задача вычисления оценочных функций особей в популяции. Таким образом, непосредственный поиск осуществляется только на головном компьютере, а параллелизация в модели РГА служит для ускорения работы алгоритма.

Для улучшения качества поиска генетические алгоритмы в распределенной вычислительной среде используют «модель островов» [8]. Модель островов отличается от модели «мастер-хозяин» следующими моментами. В модели островов также используются понятия головного процессора (системы) и вспомогательных (клиентских). Однако построение процесса поиска существенно отличается. Головной процессор выполняет управляющие функции, однако непосредственный поиск перемещён на «острова» (клиентские системы). На каждой из таких систем выполняется своя копия генетического алгоритма (построение новых поколений популяций). При этом генетические параметры работы этих алгоритмов должны несколько отличаться друг от друга. Это позволит направить поиск на каждом таком «острове» в отличном от других направлении.

Через заданное количество итераций (достаточно редко) острова производят обмен лучшими особями. Это позволяет корректировать направление поиска для менее удачных островов. Данный обмен называется миграцией. Выделим основные характеристики такого обмена:

- топология, которая определяет, какие подпопуляции будут считаться соседними и, соответственно, между какими островами будет возможен обмен особями (рис.1);
- время изоляции, которое определяет, сколько эпох ГА не будет производиться миграция;
- степень миграции, которая определяет количество особей, участвующих в миграции;
- стратегия отбора особей для миграции;
- стратегия удаления особей из подпопуляции при проведении миграции;
- стратегия репликации мигрирующих особей.

Классифицировать распределенные генетические алгоритмы можно по методу миграции, по однородности подпопуляций и по параметрам схемы обмена. По методу миграции можно выделить изолированные РГА, где миграция между подпопуляциями не



а) обмен по кольцу б) двунаправленный обмен по кругу в) обмен каждый с каждым

Рис.1. Различные топологи взаимодействия компонент распределенных ГА.

происходит; синхронные РГА, где миграция происходит в одно и то же время; асинхронные ГА, где миграции происходят в разное время, как правило, при готовности 2 соседей. По однородности популяций выделяют однородные РГА, в которых параметры ГА (кодирование, кроссинговер, мутация, редукция и их вероятности) одинаковы для всех островов, и неоднородные ГА, в которых каждый остров может иметь собственные значения параметров. По виду обмена между популяциями существуют статическая и динамическая схема соединений. В статической схеме такие характеристики обмена особями, как, например, степень миграции и время изоляции, неизменны на протяжении работы алгоритма. В динамической же схеме характеристики могут изменяться в зависимости от сходимости в той или иной подпопуляции.

Точная и корректная реализация взаимодействия различных копий ГА в модели островов является сложной задачей. В следующих разделах будут описаны алгоритмы работы и взаимодействия главного и клиентских процессоров в вычислительной среде.

3. Алгоритм работы сервера.

В данном разделе мы опишем подробно алгоритм работы сервера в распределённом ГА, основанном на модели островов.

В данной распределённой модели компонент сервера не производит непосредственного поиска решения: на нём не исполняется главный итерационный цикл ГА – построение новых популяций. Однако на сервер возложены функции организации вычислительного процесса для клиентов – островов.

Сервер начинает работу с чтения описания схемы и построения списка неисправностей. При этом сразу строится сжатый по эквивалентности список [9]. После поиска клиентов сервер отсылает им описание схемы. Также на все клиенты отсылаются параметры ГА: число особей в популяции, вероятности операций скрещивания и мутации, число итераций ГА, которые должны быть выполнены на острове перед обменом информации. Следует заметить, чтобы поиск на всех островах не был идентичен, а направлялся в разные стороны пространства поиска, данные параметры ГА несильно отличаются для всех островов.

Далее происходит итеративный процесс, который заключается в следующем. Из списка выбирается очередная непроверенная неисправность, которая называется целевой. Для неё строится начальная популяция. Принцип построения начальной популяции для задачи тестирования заключается в попытке построить входную последовательность, которая активизирует данную неисправность-цель. При этом предполагается, что с большой вероятностью влияние активизированной неисправности можно распространить на внешние выходы схемы. Данный подход построения начальной популяции подробно описан в [10]. Далее всем островам передаются данные для начала процедуры поиска: целевая неисправность и начальная популяция.

После этого сервер переходит в режим ожидания данных от клиентов. По прошествии некоторого числа итераций ГА на островах они передают на сервер информацию о результатах поиска решения. Если на одном из островов решение обнаружено, то оно (тестовая последовательность) передаётся на сервер. Сервер при этом передаёт на все остальные клиенты сигнал об окончании работы с текущей целевой неисправностью. Для полученной входной последовательности сервер выполняет процедуру моделирования с неисправностями, целью которой является обнаружение других неисправностей (кроме текущей целевой), которые также обнаруживаются данной последовательностью. После этого новая последовательность добавляется в глобальный тест.

Возможна также ситуация, когда ни на одном из островов не найдено решения за заданное число итераций ГА. В этом случае острова передают на сервер лучших особей из своей популяции, а также характеристики своей копии ГА. Сервер на основе этих данный

выбирает лучшие характеристики и последовательности, полученные с острова, который им соответствует, посылаются на другие острова. Также для них производится подстройка параметров поиска. Какие острова могут влиять друг на друга, определяется топологией островов. Заметим, что описанный выше механизм взаимодействия позволяет реализовывать любые заданные топологии связей островов: каждый с каждым, кольцевая, только соседи и т.д. После того как сервер передал островам данные для новой серии итераций ГА, он снова переходит в режим ожидания результатов поиска.

Наконец, возможен вариант, когда после многократного запуска поиска на островах решения не найдено, т.е. фактически превышено максимальное число итераций для данной неисправности. В этом случае неисправность отмечается как непроверенная, а на все острова посылается сигнал об остановке поиска для текущей целевой неисправности. После этого сервер переходит к выбору новой целевой неисправности. Если список неисправностей исчерпан, то сервер посылает всем клиентам сообщение о полном завершении работы.

Описанный словесно алгоритм работы изображен ниже в виде укрупнённого псевдокода.

```
Сервер_распределённый_ГА(описание схемы){
  if( поиск_клиентов() == найдены ){
    чтение_описания_схемы();
    построение_и_сжатие_списка_неисправностей();
    инициализация_параметров_ГА()
    do_in_threads_in_parallel{
      послать_данные_на_остров(ОПИСАНИЕ_СХЕМЫ, ПАРАМЕТРЫ_ГА);
    } // конец выполнения потоков – точка синхронизации
    while( есть_непроверенные_неисправности && не_достигнуто_число_итераций ){
      ЦЕЛЕВАЯ_НЕИСПРАВНОСТЬ=выбор_целевой неисправности();
      ПОПУЛЯЦИЯ=построение_начальной_популяции();
      do_in_threads_in_parallel{
        послать_данные_на_остров(ЦЕЛЕВАЯ_НЕИСПРАВНОСТЬ, ПОПУЛЯЦИЯ);
      } // конец выполнения потоков – точка синхронизации
      while( ЦЕЛЕВАЯ_НЕИСПРАВНОСТЬ->detected != detected ){
        do_in_threads_in_parallel{
          послать_данные_на_остров(ЛУЧШИЕ_ОСОБИ, ЛУЧШИЕ_ПАРАМЕТРЫ_ГА);
        } // конец выполнения потоков – точка синхронизации
        do_in_threads_in_parallel{
          DETECTION_SIGN=принять_параметр_обнаружимости_цели();
          if( DETECTION_SIGN == detected ){
            ЦЕЛЕВАЯ_НЕИСПРАВНОСТЬ->detected = detected;
            остановить_другие_клиенты()ж
            NEW_TEST_SEQUENCE=получить_новый_тест();
            break;
          }
          else{
            ЛУЧШИЕ_ОСОБИ=получить_лучших_особей();
            ПАРАМЕТРЫ_ГА=получить_параметры_ГА();
          }
        } // конец выполнения потоков – точка синхронизации
        сортировать_особей_на_основе_топологии(ЛУЧШИЕ_ОСОБИ);
        ЛУЧШИЕ_ПАРАМЕТРЫ_ГА=настройка_параметров_ГА(ПАРАМЕТРЫ_ГА);
      } // конец цикла – неисправность не проверилась
      if( ЦЕЛЕВАЯ_НЕИСПРАВНОСТЬ->detected == detected ){
        проверить_проверились_другие_неисправности(NEW_TEST_SEQUENCE);
        добавить_в_тест(NEW_TEST_SEQUENCE);
      } // конец if – неисправность проверилась
    }   // конец while – есть непроверенные неисправности
  }     // конец if – есть клиенты-острова
}       // конец алгоритма-сервера
```

Отметим, что обмен данными с островами не является требовательным к ресурсам процессора, однако требует определённого времени. С целью ускорения данных процедур, такой обмен можно организовать параллельно с использование вычислительных потоков. В приведённом выше псевдокоде этот момент отмечен процедурой «do_in_threads_in_parallel{}».

4. Алгоритм работы клиентов.

Перейдём теперь к описанию алгоритма работы клиентов в распределённой модели ГА построения тестов.

Каждый узел вычислительного кластера за исключением мастера выступает в качестве острова в рассматриваемой модели. Именно на островах происходит поиск решения с помощью генетического алгоритма. В отличие от традиционной реализации в данном подходе необходимо предусмотреть механизм взаимодействия с сервером вычислений, а также ряд дополнительных состояний алгоритма. Это связано с несколькими факторами.

Клиент начинает работу с поиска сервера, после чего получает от него описание схемы и параметры ГА. Далее клиент переходит в режим ожидания от сервера статусного сообщения. Данные сообщения говорят о наличии следующих ситуаций:

- сервер выбрал новую целевую неисправность, и на остров будут переданы данные для нового поиска: целевая неисправность и начальная популяция. В этом случае на острове будет запущен ГА поиска решения для новой целевой неисправности.
- сервер инициализирует обмен данными с другими островами, в этом случае клиент передаст на сервер лучших особей, которые получены к текущему моменту времени, а также характеристики совей копии ГА. Далее сервер на основе топологии обмена определит, какие данные необходимо передать клиенту, чтобы улучшить его характеристики поиска. После этого будет продолжен основной цикл выполнения ГА.
- при работе над поиском решения на острове возможна ситуация, когда решение найдено на другом острове. В этом случае серверу необходимо остановить основной цикл ГА всех островов. Острова переходят в режим ожидания данных
- когда сервер уже перебрал в качестве целевых все возможные неисправности и ему необходимо полностью завершить работу всех клиентов.

Основной цикл работы ГА реализуется традиционно для данных типов алгоритмов [5,10]. Единственное его отличие – необходимость регулярной проверки статусного сообщения с сервера. Данный момент легко реализуем с помощью дополнительного вычислительного потока, который, получив данной сообщение, установит соответствующий флаг.

Укрупнённый псевдокод предложенного алгоритма приведён ниже.

```
клиент_распределённый_ГА(){
  if( поиск_сервера() == найден ){
    чтение_описания_схемы();
    чтение_параметров_ГА();
    while(1){
      do_in_thread{
        СТАТУС_ИСПОЛНЕНИЯ=получить_статус_с_сервера();
      }
      switch( СТАТУС_ИСПОЛНЕНИЯ ){ //начинаем работу с новой неисправностью
        case 0:{  //начинаем работу с новой неисправностью
          ЦЕЛЕВАЯ_НЕИСПРАВНОСТЬ=получить_с_сервера_целевую_неисправность();
          НАЧАЛЬНАЯ_ПОПУЛЯЦИЯ=получить_начальную_популяцию();
          break;
        }
        case 1:{  // итерации не исчерпаны, и другой остров не нашёл тест
          читать_новых_особей_с_сервера();
```

```
          оценить_особей();
          break;
        }
        case 2:{  // другой клиент нашёл тест – завершаем работу
          break;
        }
        case 3:{  // итерации исчерпались, отдадим лучших особей
          послать_лучших_особей_на_сервер();
          послать_параметры_ГА_на_сервер();
          break;
        default: {};
        }
      }
      РОДИТЕЛИ=выбрать_родителей();
      ПОТОМКИ=скрещивание( РОДИТЕЛИ );
      ПОТОМКИ=мутация( ПОТОМКИ );
      добавить_потомков_в_промежуточную_популяцию( ПОТОМКИ );
      оценить_промежуточную_популяцию();
      if( ЦЕЛЕВАЯ_НЕИСПРАВНОСТЬ->detected == detected ){
        построить_новую_популяцию();
      }
      else{
        послать_на_сервер_тест();
      } // конец цикла по if – неисправность обнаружена
    }   // конец while – сервер разрешает выполнение ГА
  }     // конец if – сервер найден
}       // конец алгоритма
```

5. Состояния процессов и их взаимодействия.

При построении сложных распределённых алгоритмов особое внимание следует уделить взаимодействию компонент таких систем. В данном разделе мы опишем состояния компонента сервера и компонента клиента с точки зрения сетевого взаимодействия.

Граф состояний сервера изображён на рис.2. Вершины графа имеют следующую семантику:

1. Подготовка к очередной итерации работы ГА на клиентах: выбор очередной целевой функции; построение начальной популяции и передача этих данных на острова.
2. Опрос состояний результатов поиска на островах.
3. Реализация обмена характеристиками процесса поиска и лучшими особями между островами, основываясь на реализуемой топологии.
4. Получение теста и лучших особей с острова, который нашёл решение.
5. Моделирование с неисправностями на новом тесте с целью определения дополнительно проверяемых неисправностей.

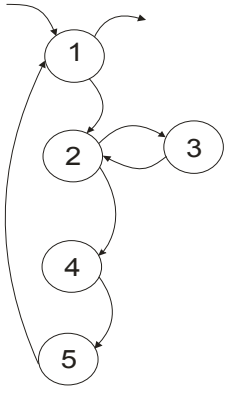


Рис.2. Граф состояний сервера.

Видно, что вершинами обмена для сервера в такой интерпретации являются вершины 1, 2, 3, 4.

При реализации параллельного обмена данными между сервером и клиентами с помощью технологии вычислительных потоков, данные вершины будут являться точками синхронизации.

Граф возможных состояний клиента показан на рис.3. Смысл попадания процесса в вершины графа следующий:

1. Начало работы над новой целевой неисправностью: получение неисправности и начальной популяции.
2. Основной цикл ГА построения новых популяций.

3. Поиск не завершён, но наступило время обмена данными с другими островами. Передача лучших особей и характеристик ГА на сервер для обмена.
4. Тест найден на данном острове. Передача теста, лучших особей и характеристик ГА на сервер.
5. Все возможные неисправности-цели рассмотрены, сервер передал сигнал на остановку.

Таким образом видно, что вершинами обмена информацией с сервером (а значит и вершинами синхронизации) являются 1, 3, 4.

Построение модели обмена с синхронизацией сервера и клиентов во всех точках взаимодействия позволяет обеспечить его непротиворечивость: исключаются случаи, когда и сервер и клиент находятся в одинаковых состояниях ожидания (либо передачи) данных.

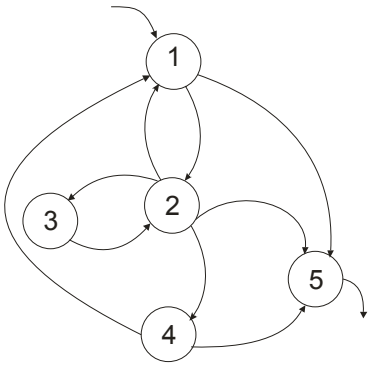


Рис.3. Граф состояний клиента.

6. Заключение.

В данной статье подробно рассмотрены принципы построения распределённых генетических алгоритмов генерации тестов. Предложены алгоритмы работы компонентов в такой модели: сервера вычислений и клиентов. Также описаны механизмы взаимодействия серверных и клиентских компонентов. Сделаны замечания о построении протоколов обмена статусными сообщениями в процессе работы данной сложной системы.

В качестве дальнейших исследований следует отметить практическую реализацию предложенного подхода, а также проведение машинных экспериментов с целью получения оптимальных параметров работы данной распределённой модели к другим задачам построения идентифицирующих входных последовательностей.

Литература.

1. Bryant R.E. *Symbolic Boolean manipulation with ordered binary decision diagrams.* // ACM Comput. Surv.- 1992, vol.24, is.3.- 293-318.
2. Sellers F.F., Hsiao M.Y., Bearnson L.W. *Analyzing errors with the boolean difference* // IEEE Transactions on Computers.- 1967.- №5.- p.675-680.
3. Lisanke R., Brgles F., Degens A.J., Gregory D. *Testability-driven random test pattern generation* // IEEE Trans. Computer-Aided Design.- 1987.- №6.- p.1082-1087.
4. Goldberg D.E., *"Genetic Algorithm in Search, Optimization, and Machine Learning"*, Addison-Wesley.- 1989.- 432p.
5. Y.A.Skobtsov, V.Y.Skobtsov, D.E.Ivanov. *Evolutionary approach to the test pattern generation for the sequential circuits* // Радиоэлектроника и информатика.- 2003, №3.- с.46-51.
6. D.Krishnaswamy, M.Hsiao, V.Saxena, E.M.Rudnick, J.P.Patel. *Parallel genetic algorithms for simulation-based sequential circuit test generation* // IEEE VLSI Design Conference, 1997.- pp.475-481.
7. F.Corno, P.Prinetto, M.Rebaudengo, M.Sonza Reorda, E.Veiluva. *A portable ATPG tool for parallel and distributed systems* // Proc VLSI Test Symp, 1995.- pp.29-34.
8. Ю.О. Скобцов. *Основы эволюционных вычислений*.– Донецк: ДонНТУ, 2008.- 326с.
9. Барашко А.С., Скобцов Ю.А., Сперанский Д.В. *Моделирование и тестирование дискретных устройств*. – Киев:Наукова думка, 1992. – 288с.
10. Иванов Д.Е., Скобцов Ю.А. *Генерация тестов цифровых устройств с использованием генетических алгоритмов* // Труды института прикладной математики и механики НАН Украины. – Т.4. – Донецк, ИПММ. – 1999. – С.82-88.