УДК 658.512

ПАРАЛЛЕЛЬНЫЕ ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

П.И. БИДЮК, В.И. ЛИТВИНЕНКО, А.А. ТОКАРЬ

Изложены основные подходы к организации параллельной работы генетических алгоритмов (ГА). Анализируются аппаратные требования, комбинирования различных архитектур в одной системе и построения неоднородных систем. Описаны особенности и схемы глобальных или мелкозернистых, крупнозернистых и гибридных ГА, а также способы организации структуры миграции (топологии) индивидумов между подпопуляциями.

ВВЕДЕНИЕ

Идея оптимальности является центральной идеей кибернетики. Понятие оптимальности получило строгое и точное представление в математических теориях, прочно вошло в практику проектирования и эксплуатации технических систем, широко используется в административной и общественной практике. Основная задача оптимизационного подхода — поиск в исходном множестве наилучших в заданных условиях, т.е. оптимальных, альтернатив. Говоря «наилучшие», предполагается, что известен или задан критерий, способ сравнивания вариантов и нахождения лучшего из них. Однако необходимо также учесть начальные условия, ограничения, так как их изменение может привести к тому, что при том же критерии наилучшим окажется другой вариант. Для поиска оптимального варианта используется множество алгоритмов и методик. Одним из эффективнейших методов на сегодня являются ГА.

ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

Генетические алгоритмы были разработаны Холландом [1] и далее хорошо зарекомендовали себя, при решении задач численной и комбинаторной оптимизации [2–5]. К таким задачам относятся: размещение, упаковка, компоновка элементов, трассировка, управление, оценивание. Базовой задачей проблем комбинаторной оптимизации является классическая NP-полная задача коммивояжера. Она используется при проектировании разводки соединений, разработке архитектуры вычислительных сетей и т.п. [6].

Генетические алгоритмы — это стохастические поисковые алгоритмы, базирующиеся на механизмах природной эволюции. При их реализации ис-

пользуют случайный направленный поиск для построения субоптимального решения данной проблемы. В отличие от эволюции, происходящей в природе, ГА только моделируют те процессы в популяциях, которые являются существенными для развития. Однако точный ответ на вопрос, какие биологические процессы существенны для развития, а какие нет, все еще остается открытым для исследователей.

Каждое решение конкретной задачи (особь) представляется хромосомой — строкой некоторых элементов (генов). Классический «простой» генетический алгоритм использует строки из двоичных элементов 0 и 1 (например, 0011101). На множестве решений определяется целевая (fitness) функция пригодности, которая позволяет оценить близость каждой особи к оптимальному решению. Множество особей (решений) образует популяцию, которая развивается (эволюционирует) от одного поколения к другому.

Таким образом воспроизводится вся новая популяция допустимых решений путем отбора лучших представителей предыдущего поколения, скрещивания их и получения множества новых особей. Новое поколение содержит более высокое соотношение характеристик, которыми обладают лучшие представители предыдущего поколения. Таким образом, из поколения в поколение хорошие характеристики распространяются по всей популяции. Скрещивание наиболее приспособленных особей приводит к тому, что исследуются наиболее перспективные участки пространства поиска. В конечном итоге, популяция будет сходится к оптимальному решению задачи.

Механизм простого ГА достаточно прост [2, 6]. Вначале ГА случайно строит начальную популяцию, к которой далее применяются несколько простых операций и генерируются новые популяции. Простой ГА использует три основных оператора: *репродукция, кроссинговер* (скрещивание) и *мутация*. При репродукции хромосомы копируются согласно значениям их целевой функции. Копирование лучших хромосом с более высокими значениями целевой функции определяет большую вероятность их попадания в следующую генерацию. Оператор репродукции реализует принцип «выживания сильнейших» по Дарвину. При кроссинговере происходит скрещивание двух особей, т.е. обмен последовательностями генов, выбранных случайно. Оператор мутации определяет новое значение особи, изменяя случайным образом один или несколько генов. Операторы кроссинговера и мутации определяют направление развития популяции.

АППАРАТНОЕ ОБЕСПЕЧЕНИЕ

Поскольку ГА применяются для решения больших и наиболее трудных задач поиска, они требуют, соответственно, больших вычислительных ресурсов. При увеличении нагрузки на вычислительную систему появляется желание приобрести новое более мощное аппаратное обеспечение. Однако, покупая все более мощные компьютеры, очень скоро можно приобрести «самый лучший», но и это не решит проблемы так, как этого хотелось бы. Более логичный путь заключается в использовании многопроцессорных вычислительных систем [7]. Существенными преимуществами многопроцессорной системы являются надежность и производительность.

Основным параметром классификации параллельных компьютеров является наличие общей (SMP) или распределенной памяти (MPP). Нечто среднее между SMP и MPP представляют собой NUMA-архитектуры, где память физически распределена, но логически общедоступна. Более дешевым вариантом MPP являются кластерные системы [7]. При поддержке команд обработки векторных данных говорят о векторно-конвейерных процессорах, которые в свою очередь могут объединяться в PVP-системы с использованием общей или распределенной памяти. В настоящее время все большую популярность приобретают идеи комбинирования различных архитектур в одной системе и построения неоднородных систем.

Свободно связанные процессоры или кластеры представляют собой решение с большей масштабируемостью и обеспечивают большую помехоустойчивость, чем многопроцессорные системы. В отличие от MPP и SMP, идея кластеризации отнюдь не нова. Еще в начале 80-х компания Digital Equipment представила машину VAXCluster с операционной системой OpenVMS. Кластеризация Unix-машин началась несколько позже, когда компании начали использовать Unix для критически важных приложений, что поставило на повестку дня необходимость высокой готовности, следовательно, кластеризации. Интенсивное развитие кластерных технологий началось в 90-х годах. Одной из первых фирма DEC в 1993 г. начала разработку коммерческих продуктов в области кластеризации, выпустив основанный на Unix пакет DECAdvantage. В том же году фирма IBM разработала для собственной разновидности Unix — AIX систему High-Availability Cluster Multi-Processing (HACMP). В последующем системы для организации кластеров из Unix-машин начали предлагать около десятка компаний, в их числе Data General, Hewlett-Packard, Pyramid Technology, Sequent Computer Systems, Sun Microsystems.

Кластер состоит из независимых узлов, каждый из которых имеет собственный процессор и память, и работает со своей копией операционной системы. Выход из строя одного узла не приведет к неработоспособности системы в целом. Немаловажным преимуществом кластерных систем перед другими является возможность организации данной системы на базе рабочих станций общего назначения. Иначе говоря, вычислительный кластер — это совокупность компьютеров, объединенных в рамках некоторой сети для решения одной задачи. В качестве вычислительных узлов обычно используются доступные на рынке однопроцессорные компьютеры, двух- или четырехпроцессорные SMP-серверы. Состав и мощность узлов может меняться даже в рамках одного кластера, давая возможность создавать неоднородные системы. Выбор конкретной коммуникационной среды определяется следующими факторами: особенностями класса решаемых задач, доступным финансированием, необходимостью последующего расширения кластера и т.п. Возможно включение в конфигурацию специализированных компьютеров, например файл-сервера, и, как правило, предоставление возможности удаленного доступа к кластеру через Internet.

Очевидно, что простор для творчества при проектировании кластеров огромен [7]. Рассматривая крайние точки, кластером можно считать как пару ПК, связанных локальной 10-мегабитной сетью Ethernet, так и вычислительную систему, организованную в рамках некоторого проекта. Например,

проект Beowulf (www.beowulf.org), название которого превратилось в имя нарицательное для всех последующих кластерных систем такого рода (Beowulf-кластеры). Он был реализован летом 1994 года в научном космическом центре NASA-Goddard Space Flight Center (GSFC), а точнее в созданном на его основе центре CESDIS (Center of Excellence in Space Data and Information Sciences). Первоначальный кластер, который и был назван «Beowulf», создавался как вычислительный ресурс проекта Earth and Space Sciences Project (ESS) и состоял из 16 узлов на процессорах 486DX4/100MHz с 16 MB памяти и 3 сетевыми адаптерами на каждом узле, обеспечивающими связь через 3 «параллельных» Ethernet-кабеля по l0 Mbit. В последующие годы в GSFC и других подразделениях NASA были собраны аналогичные, но существенно более мощные кластеры. Например, кластер HIVE (Highly-parallel Integrated Virtual Environment), содержащий 64 узла по 2 процессора Pentium Pro/200MHz и 4GB памяти в каждом, 5 коммутаторов Fast Ethernet. В рамках проекта Beowulf был разработан ряд высокопроизводительных и специализированных сетевых драйверов (в частности, драйвер для использования нескольких Ethernet-каналов одновременно) или проект Berkeley NOW (Network Of Workstations) — технология, очень похожая на Beowulf, разработанная в Калифорнийском университете в Беркли. В 1997 г. на кластере из 100 рабочих станций на базе UltraSPARC была достигнута производительность в 10GFLOPS по UNPACK, что позволило ему войти в число 200 самых мощных компьютеров мира. Проект официально завершен в мае 1998 г. Через Интернет доступно разработанное в рамках проекта программное обеспечение, в том числе ОС GLUnix (специализированная версия UNIX), система передачи сообщений Active Messages, реализация MPI и др.

ПАРАЛЛЕЛЬНЫЕ ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

Для работы в многопроцессорной или распределенной среде необходимо использовать несколько отличные от обычных генетические алгоритмы — параллельные генетические алгоритмы (ПГА), которые базируются на классических параллельных алгоритмах. Большинство параллельных программ базируются на принципе «разделяй и властвуй». Этот принцип реализуется в ПГА несколькими способами.

По степени манипулирования популяцией ПГА можно классифицировать как глобальные или мелкозернистые, крупнозернистые и гибридные ГА [8].

Глобальные ГА используют такую параллельную модель как «фермер-рабочие». Глобальные ГА используют одну популяцию и распределяют задание по вычислению новой популяции среди процессоров. Например, главный процессор (называемый в соответствии с данной технологией еще фермером) может осуществлять отбор индивидов, а остальные процессоры (для данной технологии ─ рабочие) — выполнять генетические операторы. Чаще всего используется следующий подход: главный процессор выполняет генетические операторы и отбор индивидов, а остальные процессоры подсчитывают функцию пригодности. Главное преимущество глобальных ГА — это сокращение времени вычислений. Если процессор доступен для каждого «решения» в текущей популяции, то время вычисления оценочной

функции для всей популяции уменьшается до значения, несколько большего, чем время, необходимое для оценки одного решения (затраты на обмен данными незначительны по сравнению со временем, необходимым для оценки одного решения). Этот подход может быть полезным, если оценочная функция очень дорога с вычислительной точки зрения; однако, в отличие от методов, описанных ниже, он не уменьшает количества оценок, необходимых для нахождения лучшего решения. Здесь можно выделить *синхронный* и *асинхронный* подходы [8]. При синхронном подходе главный процессор ожидает результатов вычисления целевой функции со всех процессоров и только после этого продолжает исполнение алгоритма. При асинхронном подходе главный процессор не приостанавливает исполнение алгорима для ожидания результатов с медленных процессоров, но работа ГА в этом случае немного отличается от классического простого ГА. Асинхронный подход целесообразно применять в гетерогенной (неоднородной) кластерной системе.

Несколько интересных схем распараллеливания глобальных ГА были предложены Грефенстеттом (Grefenstette) [9].

Первая схема. Главный процессор выполняет селекцию и применяет кроссинговер и мутацию. Затем индивидумы рассылаются остальным процессорам для оценки и возвращаются на главный процессор в конце каждой генерации.

Вторая схема. Данная схема очень похожа на первую, но здесь нет четкого разделения между поколениями. Когда процессоры заканчивают подсчет оценки, результат отсылается на главный процессор, а процессоры получают новых индивидов. Эта схема называется асинхронным глобальным ГА.

Третья схема. В этой схеме популяция хранится в общей памяти. Процессоры получают доступ к популяции независимо друг от друга.

Более привлекательным подходом является **крупнозернистый ГА**. Данный подход еще называется «островным параллелизмом» [10]. В этой схеме большая популяция разбивается на несколько подпопуляций, над каждой из которых выполняется традиционный ГА. Отдельные популяции исследуют их собственное пространство поиска, но иногда обмениваются их наилучшими решениями. Результатом такого «островного параллелизма» является то, что отдельные популяции осуществляют поиск в различных частях пространства поиска относительно независимо, но, основываясь на «предположениях» от других популяций, начинают сходиться и прилагать усилия параллельно в перспективных областях. Этот подход действительно уменьшает общее количество оценок, необходимых для нахождения оптимальных решений, а также менее подвержен вероятности остановки в локальном оптимуме, чем одна популяция при тех же значениях параметров [11].

Среда островного параллелизма также обладает более эффективной защитой от сбоев, так как при выходе из строя одного или нескольких процессоров почти не повлияет на работу алгоритма и поиск решения в целом. В данном подходе Грефенстетт выделил несколько интересных вопросов о частоте миграции индивидумов, расстоянии миграции (топологии) и воздействие миграции на преждевременную сходимость [8, 9]. Решение данных проблем все еще остается открытым.

Существует множество способов для организации структуры миграции (топологии) индивидумов между подпопуляциями. Можно привести следующие наиболее популярные топологии:

- *полный граф* (соединение всех подпопуляций);
- *кольцо*;
- *смежные острова*.

Топология Полный граф [12]. В данной топологии (рис. 1) индивидумы могут мигрировать из некоторой подпопуляции в любую другую случайно или в соответствии с некоторым правилом выбора подпопуляции.

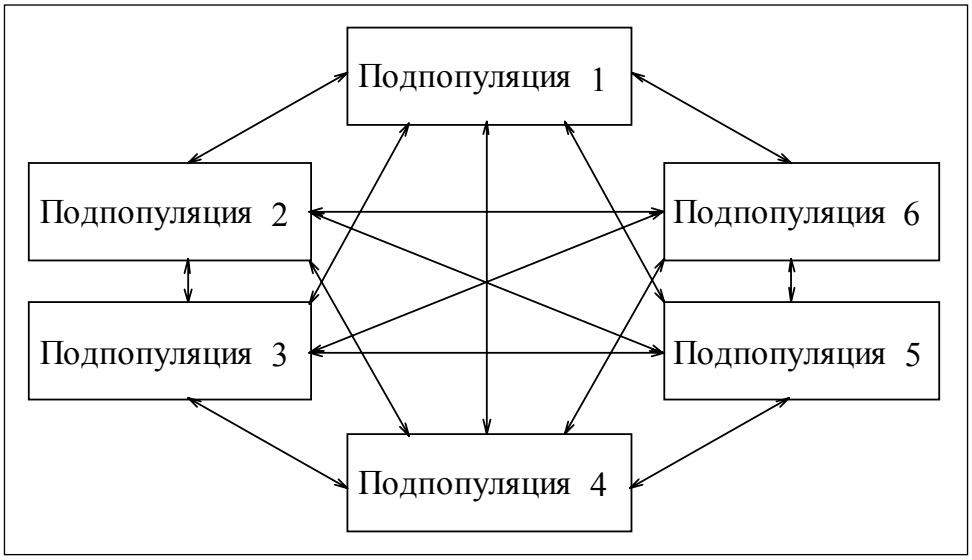

Рис. 1. Топология Полный граф

Топология Кольцо [13], представленная на рис. 2, является наиболее часто используемой. В этой топологии индивиды мигрируют по кругу от

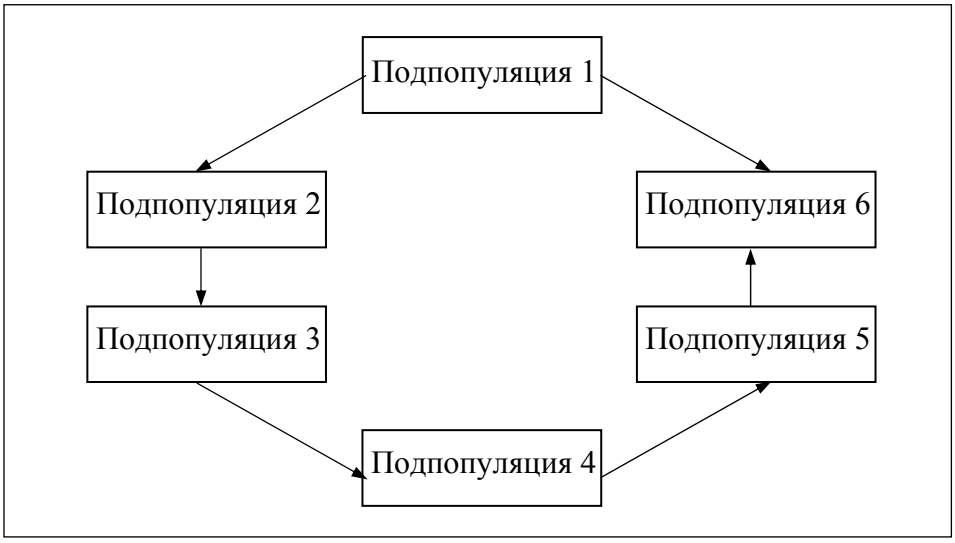

Рис. 2. Топология *Кольцо*

одной популяции в другую. Например, индивиды первой подпопуляции могут мигрировать только во вторую, из второй — в третью и т.д. по кругу.

Смежная топология [14] похожа на предыдущую топологию, однако здесь индивиды мигрируют только между ближайшими соседями данной подпопуляции (рис. 3). Например, в 2-мерной реализации данной топологии у каждой подпопуляции есть три ближайших соседа, к которым могут мигрировать индивиды. Выбор подпопуляции, куда будут мигрировать особи из данной подпопуляции, осуществляется случайным образом или в соответствии с некоторым правилом выбора.

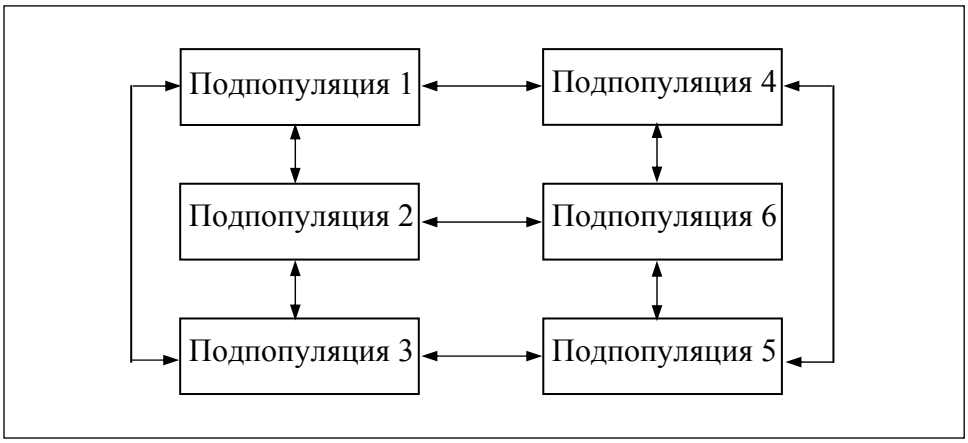

Рис. 3. Смежная топология

Еще более привлекательным, чем островной и гранулярный параллелизм, на наш взгляд, является гибридный подход к распараллеливанию ГА [15].

В данном подходе объединяются оба, рассмотренных раннее, подхода. Основная цель данного подхода — достижение более высокой производительности, и решение некоторых недостатков предыдущих подходов.

Основным недостатком рассмотренных ПГА является то, что после достижения определенного числа процессоров эффективность данного подхода начинает снижаться. Размер популяции играет важную роль и является принципиальным фактором качества конечного решения. Второй немаловажный фактор — время, которое затрачивает алгоритм для поиска конечного решения.

Решением проблемы увеличения процессоров является гибридный подход, который объединяет мелко- и крупнозернистые подходы. В данном случае строится следующая иерархия: на верхнем уровне находится крупнозернистый подход, на нижнем — мелкозернистый. Крупнозернистый подход предоставляет микрогранулированному популяцию подходящего размера и в добавок к этому может осуществлять обмен лучшими индивидумами с другими подпопуляциями. Такое свойство данного подход позволяет существенно расширить круг решаемых задач [14, 15].

В гибридном подходе можно выделить два класса систем: однородные и гетерогенные (рис. 4, 5) [16, 17].

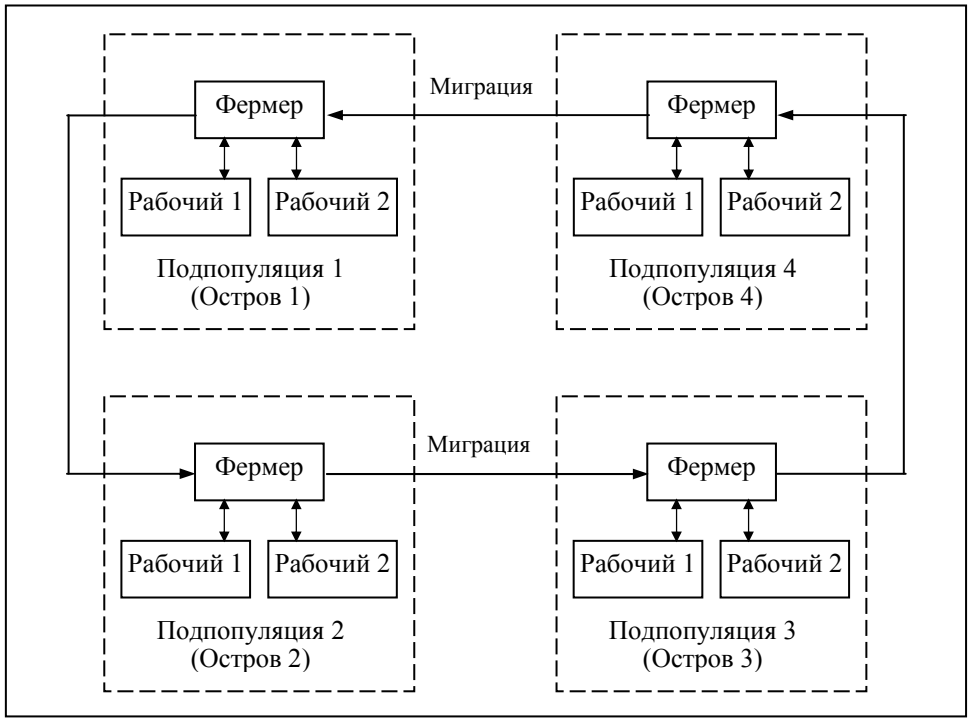

Рис. 4. Однородная гибридная система

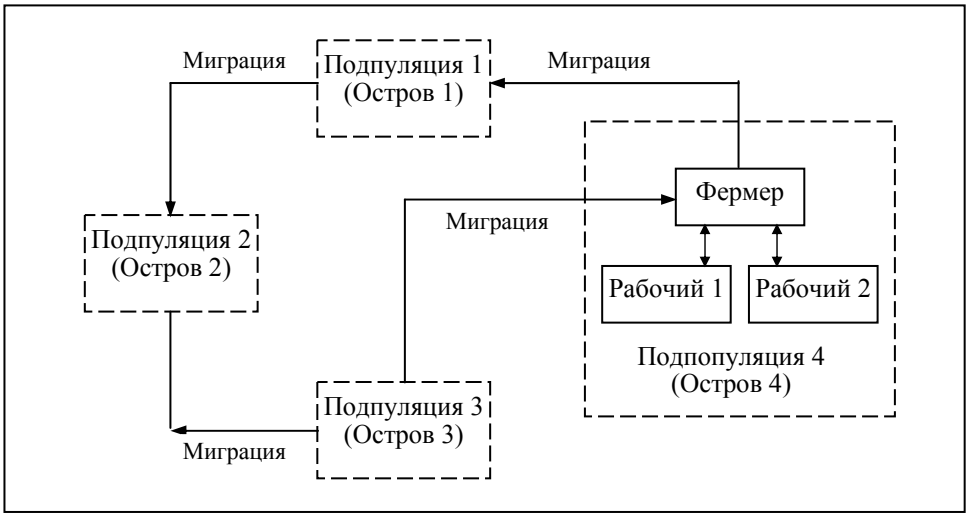

Рис. 5. Гетерогенная гибридная схема

ЗАКЛЮЧЕНИЕ И ВЫВОДЫ

Поскольку ГА применяются для решения больших и наиболее трудных задач поиска, становится необходимым проектировать более быстрые алгоритмы и распределенные алгоритмы, которые сохраняли бы возможность нахождения оптимального решения. С аппаратной точки зрения для реше-

ния таких задач применяются многопроцессорные системы — SMP, MPP, NUMA или кластерные системы, с программной реализацией параллельных или распределенных алгоритмов.

Мелкозернистые алгоритмы ищут решение в пространстве поиска аналогично обычным ГА. Они сохраняют все преимущества линейных ГА и просты в реализации, однако требуют постоянного обмена информацией между главным и подчиненными процессорами. Такая связь уменьшает нагрузку вычислений на один процессор, распределяя ее между несколькими процессорами. Анализ соединения показал, что на практике существует некоторое оптимальное число процессоров, которое минимизирует время выполнения данного алгоритма.

Крупнозернистые алгоритмы исследуют свое собственное пространство поиска и иногда обмениваются наилучшими решениями. Данные алгоритмы вносят в обычные ГА новый генетический оператор — *миграцию*, перемещение лучших особей между популяциями согласно выбраной топологии. Этот подход также уменьшает общее число оценок, необходимых для нахождения оптимальных решений, а также менее подвержен вероятности в локальном оптимуме, чем одна популяция. Так же как и для мелкозернистых алгоритмов у данного алгоритма существует свое оптимальное число процессоров, минимизирующих временной показатель и круг решаемых задач. Для преодоления этого барьера рекомендуется использовать гибридный ГА, с крупнозернистым ГА на верхнем уровне и мелкозернистым — на нижнем. В данном подходе различают однородные и гетерогенные системы.

Основными исследованиями, проводимыми в данной области, являются, на сегодняшний день, поиск более эффективных способов обмена данными между подпопуляциями и приспосабливание обычного ГА под особенности реализации в многопроцессорной системе, т.е. ориентирование основных генетических операторов на их выполнение в многопроцессорной среде.

ЛИТЕРАТУРА

1. *Holland J.H.* Adaptation in natural and artificial systems. Pittsburg: University of Michigan Press, 1975. — 177 p.
2. *Скурихин А.Н.* Генетические алгоритмы // Новости искусственного интеллекта. — 1995.— № 4. — С. 6–46.
3. *Bauer R.J.* Genetic algorithms and investment strategies. — New York: John Wiley & Sons, Inc., 1994. — 308 p.
4. *Rojas I., Gonzalez J., Pomares H., Merelo J.J., Castillo P.A., Romero G.* Statistical analysis of the main parameters involved in the design of a genetic algorithm // IEEE Trans. on Systems, Man, and Cybernetics. Part C. — 2002. — **32**, N 1. — P. 31–37.
5. *Juang Chia-Feng.* A TSK-Type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms // 2002, **10**, N. 2. — P. 155–170.
6. *Скобцов Ю.А.* Применение генетических алгоритмов в интеллектуальных САПР // Наук. пр. Донецького держ. техн. ун-ту. Сер. Обчислювальна техніка та автоматизація. — 1999. — Вип. 25. — С. 101–110.
7. *Основные* классы современных параллельных компьютеров. http://parallel.ru

8. *Cantu-Paz E.* Designing efficient master-slave parallel genetic algorithms, in Genetic Programming 1998: Proc. of the Third Annual Conf. — Madison, Wisconsin: Morgan Kaufmann University of Wisconsin. — 455 p.
9. *Miki M., Hiroyasu T.* New crossover scheme for parallel distributed genetic algorithms // Proc. of the IASTED Intern. Conf., Parallel and Distributed Computing And Systems, November 6-9, 2000. — Las Vegas, Nevada, 2002. — P. 145–150.
10. *Grefenstette J.J.* Parallel adaptive algorithms for function optimization (Tech. Rep. N CS-81–19). — Nashville: Vanderbilt University, 1981. — 129 p.
11. *Литвиненко В.И., Бюргер Ю.А., Мельник А.В.* Сравнительная характеристика работы двух крупнозернистых моделей распределенного генетического алгоритма при решении задач оптимизации без ограничений //МКІМ-2002. Міжнар. конф. з індуктивного моделювання, Львів. 20–25 травня 2002: праці в 4 томах. — **2**. — Львів: державний НДІ інформаційної інфраструктури, 2002. — С. 70–76.
12. *Miki M., Hiroyasu T., Hatanaka A.* parallel genetic algorithms with distributed-environment multiple population scheme. // Proc. 3rd World Congress of Structural and Multidisciplinary Optimization (WCSMO). 1999. — **1**. — P. 186–191.
13. http://www.systemtechnik.tu-ilmenau.de/~pohlheim/GA_Toolbox/algparal.html. Evolutionary Algorithms: Principles, Methods and Algorithms.
14. *Punch W.F., Averill R.C., Goodman E.D., Lin S-C., Ding Y.* Design using genetic algorithms ─ some results for laminated composite structures // IEEE Expert, February. — 1995. — **10**, N 1. — P. 42–49.
15. *Calegari P.* Parallelization of population-based evolutionary algorithms for combinatorial optimization problems // Ph.D. Thesis, September 1999. — Swiss Federal Institute of Technology in Lausanne. — 235 p.
16. *Agapie A.* Genetic algorithms: theory and applications // Int. J. of Computing Anticipatory Systems. — 2000. — **7**. — P. 35–44.
17. *Bussola R., Faglia R., Tiboni M.* A hyperincursive method for the solution of the inverse kinematics of industrial robots based on neural networks and genetic algorithms // Int. J. of Computing Anticipatory Systems. 2000. — **7**. — P. 45–60.

Поступила 05.11.2002