ПАРАЛЛЕЛЬНЫЕ МЕТОДЫ РЕШЕНИЯ ЗАДАЧ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ: ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ И ЭВОЛЮЦИОННЫЕ СТРАТЕГИИ[1]

Ю.Н. Сидоренко, С.В. Тимченко

Томский государственный университет, Томский государственный университет систем управления и радиоэлектроники

В последние годы при решении различных задач оптимизации и поиска стали широко применяться различные адаптивные процедуры, среди которых особую популярность завоевали эволюционные и в том числе генетические алгоритмы (ГА) и эволюционные стратегии (ЭС). В той или иной мере они основаны на эволюционных процессах в биологических популяциях, развивающихся поколение за поколением, подчиняясь законам естественного отбора и принципу «выживает сильнейший» (точнее, наиболее приспособленный).

Чрезвычайно важным свойством как ГА, так и ЭС является их естественный внутренний параллелизм. При этом по сравнению с градиентными методами различие во времени расчета целевой функции при различных значениях ее параметров не оказывает влияния на эффективность распараллеливания (эти времена могут отличаться на порядки – например, если можно определить, что точка в пространстве поиска не отвечает наложенным ограничениям, не вычисляя целевую функцию или вычисляя ее только частично).

1. Описание последовательного ГА

Генетические алгоритмы носят итерационный характер и имеют дело с обработкой популяций индивидуумов $P(t) = \left\{x_1^t, x_2^t, ..., x_n^t\right\}$ для итерации t (поколение t). Каждый индивидуум представляет собой потенциальное решение задачи (испытание) и реализуется в некоторой, возможно достаточно сложной, структуре данных S. В этой работе в качестве S рассматриваются строки, составными элементами которых являются вещественные числа (вещественные генетические алгоритмы).

Каждое решение x_i^t оценивается, и определяется мера его «пригодности». Затем формируется новая популяция (итерация или поколение $t+1$). На первом шаге этого формирования – этапе селекции – происходит отбор индивидуумов, обладающих лучшей пригодностью. На следующем шаге

[1] Работа выполнена при поддержке гранта РФФИ 02-01-01022.

некоторые из отобранных таким образом индивидуумов подвергаются преобразованиям с помощью «генетических операторов»: мутации и скрещивания. Оператор мутации создает нового индивидуума путем относительно малого изменения в одном индивидууме, а оператор скрещивания осуществляет более сильные трансформации и создает нового индивидуума путем комбинирования частей из нескольких (двух или больше) индивидуумов. После ряда итерационных шагов алгоритм сходится к лучшему из возможных решений. Остановимся теперь более подробно на трех «генетических операторах» – селекции, скрещивании и мутации.

Селекция. Целью селекции является осуществление выборки индивидуумов в текущей популяции (т.е. из некоторого набора) пропорционально их пригодности. Обычно используют четыре различных механизма селекции – “колесо рулетки”, остаточную стохастическую выборку, равномерную стохастическую выборку и турнирную селекцию. Первые три алгоритма являются вариантами пропорциональной селекции, а последний – непропорциональной.

Наиболее универсальной считается так называемая турнирная селекция, не требующая предварительного ранжирования функции пригодности. При этом последовательно берутся два соседних элемента текущей популяции (первый и второй, третий и четвертый и т.д.) и лучший из них помещается в промежуточную популяцию P'. После первого прохода (пока сформирована только половина промежуточной популяции) исходная популяция случайным образом перемешивается и описанный процесс повторяется еще раз. Здесь лучшие или худшие индивидуумы рассматриваются в смысле их упорядочивания согласно соответствующим значениям целевой функции.

Скрещивание. Наиболее простым является одноточечное скрещивание – каждая выбранная пара строк скрещивается следующим образом: случайно выбирается положение точки сечения (целое число k в промежутке от 1 и $l-1$, где l – длина строки). Затем путем обмена всеми элементами между позициями $k+1$ и l включительно рождаются две новые строки. Например, пусть первая особь – $A=(x_1,x_2,x_3,x_4,x_5)$, а вторая соответственно $B=(y_1,y_2,y_3,y_4,y_5)$ и пусть случайно выбранная точка сечения будет после третьего гена. Тогда в результате скрещивания получим две особи-потомка $A'=(x_1,x_2,x_3,y_4,y_5)$ и $B'=(y_1,y_2,y_3,x_4,x_5)$. После этого потомки замещают родительские особи в промежуточной популяции P'. Схематично этот вариант показан на рис. 1.

Одноточечное скрещивание легко обобщается на *n*-точечное с *n* точками сечения. Предельным случаем является равномерное скрещивание, при котором каждый ген первого из родителей случайным образом переда-

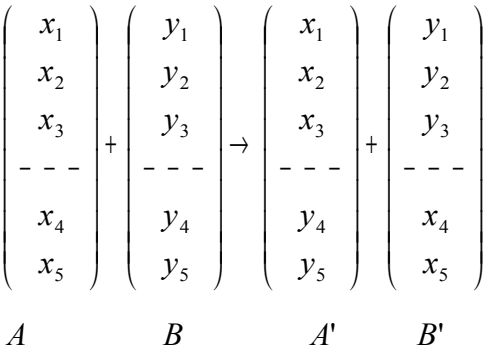


Рис. 1. Скрещивание

ется любому из потомков, при этом другой потомок соответственно получает ген от другого родителя.

Для вещественных ГА наиболее интересно арифметическое скрещивание [2], так как в этом случае сами гены подвергаются модификации. Рассмотрим этот механизм в символьном виде. Пусть у нас есть две родительские особи:

$A = (y_1,...,y_n)$ и $A' = (y'_1,...,y'_n)$.

В результате арифметического скрещивания они заменяются потомками вида

$$A = (\alpha y_1 + (1-\alpha)y'_1,...,\alpha y_n + (1-\alpha)y'_n)$$

и

$$A' = (\alpha y'_1 + (1-\alpha)y_1,...,\alpha y'_n + (1-\alpha)y_n)$$

при некотором случайном значении α из диапазона [–0.5,1.5].

Мутация. Для вещественного ГА использовалась неоднородная мутация [2]. Если ген y_i подвергается мутации, то его новое измененное значение y'_i выбирается внутри интервала $[Min_i, Max_i]$ следующим образом:

$$y_i' = y_i + s(M_i - y_i)\left(1 - \frac{t}{t_{\max}}\right)^b,$$

где s – случайное число из интервала [0,1], M_i случайным образом выбирается из множества $\{Min_i, Max_i\}$, где Min_i и Max_i – нижняя и верхняя границы возможного изменения значения переменной y_i. Такая адаптивная мутация позволяет соблюдать в процессе реализации ГА (эволюции) необходимый баланс между двумя разномасштабными изменениями (мутациями) генов, так как на первоначальных шагах алгоритма в основном преобладали крупномасштабные изменения (обеспечивающие широкую область поиска), в то время как на заключительном этапе (за счет уменьшения масштаба мутаций) происходило уточнение решения.

Полезно рассмотреть выполнение генетического алгоритма как двухстадийный процесс (рис. 2). Начинается он с текущей популяции, к которой применяется оператор выбора, чтобы создать промежуточную популя-

цию. После этого к промежуточной популяции применяются операторы рекомбинации и мутации для того, чтобы создать следующую популяцию. Процесс продвижения от текущей популяции до следующей популяции составляет одно поколение в выполнении генетического алгоритма.

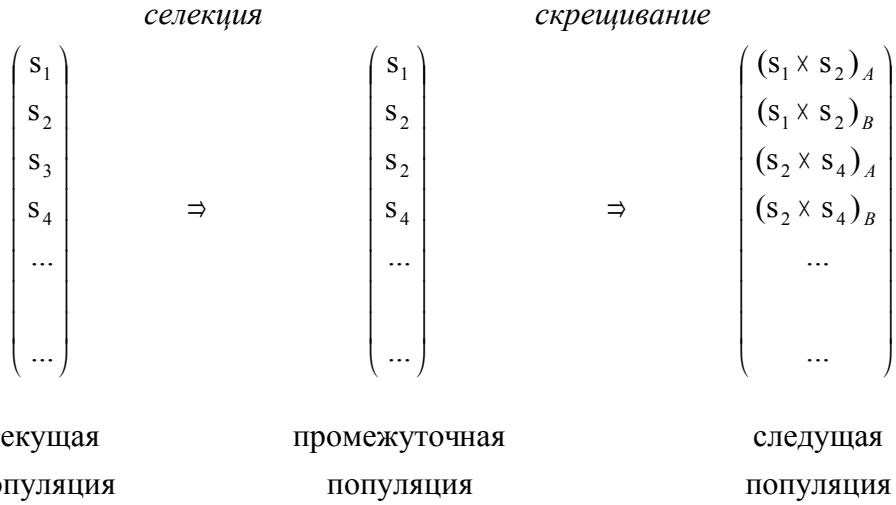


Рис. 2. Генетический алгоритм

2. Параллельные генетические алгоритмы

Как было отмечено выше, у генетических алгоритмов есть чрезвычайно важное свойство. Они обладают естественным внутренним параллелизмом. Причем различие времени расчета целевой функции для разных процессов несущественно. Интересно отметить, что основоположник генетических алгоритмов Холланд [1] задолго до своих первых работ, посвященных ГА, в конце 50-х годов предложил создание мультипроцессора для запуска недетерминированного числа конкурирующих процессов, адаптирующихся во время выполнения (для моделирования эволюции естественных популяций).

Основная идея большинства параллельных алгоритмов заключается в том, чтобы разбить задачу на сравнительно небольшие подзадачи и решать их совместно, используя многопроцессорную вычислительную систему. Стратегия «разделяй и властвуй» может быть реализована большим количеством способов. Одни из них лучше подходят для систем с массовым параллелизмом, другие – для небольших параллельных систем.

2.1. Глобальные ПГА (технология master-slave). В этом случае существует только одна популяция на master-процессоре, как и в последовательном ГА. При этом остальные процессоры (slave-процессоры или workers-процессоры) используются только для расчета целевых функций (рис. 3). Метод очень просто реализуется (так как фактически используется

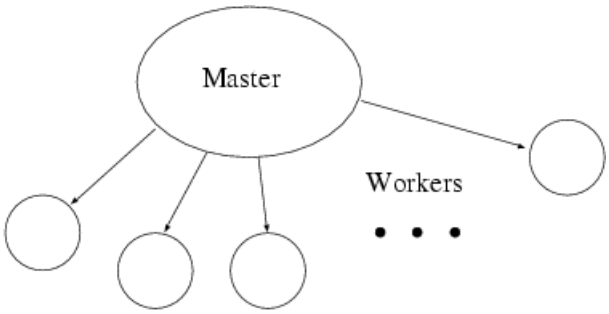


Рис.3. Глобальный ПГА

последовательный ГА) и хорош в тех случаях, когда вычислительная часть задачи доминирует над коммуникационной. При построении глобальных ПГА не делается никаких предположений об архитектуре параллельного компьютера. Число индивидуумов, приходящихся на один процессор, либо константа, либо может динамически меняться с целью обеспечения равномерности загрузки (например, в многопользовательской среде).

Как правило, используются синхронные глобальные ПГА, хотя существуют реализации и асинхронных глобальных ПГА [3]. На компьютерах с MIMD-архитектурой алгоритм, как правило, состоит в размещении популяции на одном процессоре, рассылке индивидуальностей на workers-процессоры, получении на главном процессоре значений целевой функции и применении генетических операторов для следующего шага.

Ниже приводится пример псевдокода асинхронного глобального ПГА.

```
t = 0
initpopulation P(t) /* random or random + initial solution(s) */
Scatter P(t)        /* for evaluation */
while not done do
    P' := ∅
    while size(P') < size(P) do
        Recv_from_j (P_i, f(P_i)) /* from j - th node after evaluation */
        P' = P' ⊕ P_i
        P'' := selectparents P(t)    /* tournament selection */
        recombine P''                /* arithmetical crossover */
        mutate P''  /* nonuniform + distance - dependent mutation */
        Send_to_j(P'') /* to j - th node for evaluation */
    enddo
    P(t + 1) := P'(t) + best(P(t)) /* elitism */
    t := t + 1
enddo
```

2.2. Более сложная модель – крупнозернистый ПГА [4] (другие названия - распределенный ГА или островная модель ГА), в рамках которой

на каждом процессоре располагается изолированная сабпопуляция, обменивающаяся особями с другими сабпопуляциями за счет миграции, регулируемой рядом параметров (рис. 4). Каждая сабпопуляция невелика по размеру, поэтому скорость сходимости весьма высока, хотя качество решения оставляет желать лучшего. Существует критическая скорость миграции, позволяющая получить результат, идентичный последовательному ГА. Один из подходов состоит в том, что на каждом шаге отсылается соседям лучшая особь. Однако в этом случае существует опасность преждевременной сходимости к локальному экстремуму. Другой подход – замена худшего на лучшего от соседей каждые пять (например) итераций. В целом можно отметить, что

• КЗПГА - это достаточно несложное обобщение последовательного ГА, так как фактически имеем несколько последовательных ГА на процессорах, иногда обменивающихся отдельными особями.

• Легко реализуется на кластерах.

• Последовательный ГА легко адаптируется к КЗПГА.

Основными параметрами КЗПГА являются:

• топология связей между сабпопуляциями,

• интенсивность миграции (сколько особей посылается за один раз),

• интервал миграции (количество поколений, после которых происходит миграция).

2.3. *Мелкозернистый ПГА, или клеточный ГА* (рис.5). В этом случае используется большое число очень маленьких сабпопуляций (в пределе - одна особь на процесс), которые интенсивно обмениваются индивидуумами. Эта модель наиболее хороша для MPP-архитектур, но может применяться на любых мультипроцессорах.

2.4. *Гибридные схемы*, объединяющие сразу две из предложенных выше схем (рис.6).

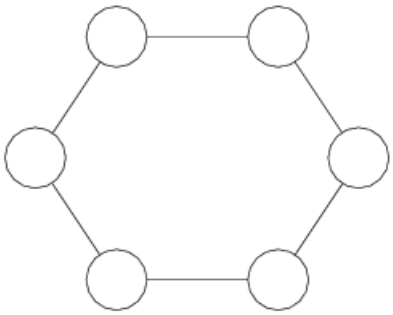


Рис.4. Крупнозернистый ПГА

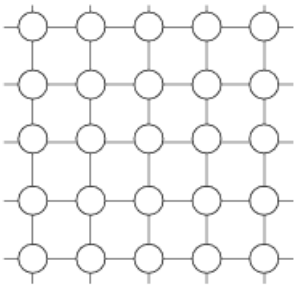
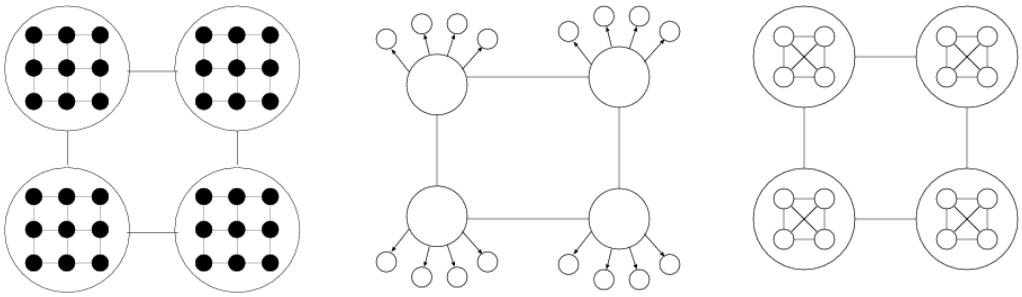


Рис.5. Мелкозернистый ПГА



а) б) в)

Рис. 6. Гибридные схемы:

а) крупнозернистый ПГА+ мелкозернистый ПГА
б) глобальный ПГА + крупнозернистый ПГА
в) крупнозернистый ПГА + крупнозернистый ПГА

Литература

1. Holland J.H. Adaptation in natural and artificial systems. – Ann Arbor: The University of Michigan Press, 1975. – 412 p.
2. Michalewicz Z. Genetic algorithms + data structures = evolution programs. – New York: Springer-Verlag, 1992.
3. Peigin S., Periaux J., Timchenko S. Asynchrone Parallel Genetic Algorithm for Heat Flux Optimization Problem // Parallel Computational Fluid Dynamics. Development and Applications of Parallel Technology. – Amsterdam: Elsevier Science B.V., 1999. – P. 377–384.
4. Grefenstette, J. J. Parallel adaptive algorithms for function optimization (Tech. Rep. No. CS-81-19). – Nashville, TN: Vanderbilt University, Computer Science Department, 1981.