ВЕСТНИК ЮЖНОГО НАУЧНОГО ЦЕНТРА РАН Том 1, № 2, 2005, стр. 41–50

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

УДК 628.3 + 621.3

КВАНТОВЫЕ И ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ – НОВАЯ ТЕХНОЛОГИЯ ЭВОЛЮЦИОННОГО ПОИСКА

© 2005 г. В.М. Курейчик

Рассматривается новая технология решения оптимизационных и комбинаторно-логических задач на графовых моделях на основе композитных (интегрированных квантовых и генетических) алгоритмов. Это позволяет получать наборы локально-оптимальных решений и строить эвристические алгоритмы с полиномиальной скоростью роста количества операций в зависимости от объема входных данных.

При решении оптимизационных и комбинаторно-логических задач проектирования, конструирования и искусственного интеллекта на основе графовых моделей эффективно используются стратегии, концепции, методы, механизмы эволюционного моделирования и бионического поиска [1–6]. Бионический поиск с точки зрения преобразования информации при решении задач на графах – это последовательное преобразование одного конечного нечеткого множества альтернативных решений в другое. Само преобразование называется алгоритмом поиска, или генетическим алгоритмом (ГА). В основе ГА лежит случайный, направленный или комбинированный поиск. Такие алгоритмы эффективно используют информацию, накопленную в процессе эволюции, для получения квазиоптимальных и оптимальных решений [4, 5]. Использование идей квантовой механики позволяет при поиске решений в задачах на графах использовать подходы параллельных вычислений. Согласно известному принципу суперпозиции, система (графовая модель) может как бы одновременно находиться во всех возможных состояниях. Производя над одним состоянием графа произвольные действия, мы производим это одновременно над заданным множеством состояний [7–10]. При решении оптимизационных задач на графах предлагается новая технология на основе совместного бионического и квантового поиска. Описаны новые архитектуры и принципы такого поиска. Это позволяет расширить область поиска решений без увеличения времени работы и сократить преждевременную сходимость алгоритмов, повысить эффективность и качество получаемых решений.

[1] Таганрогский государственный радиотехнический университет, г. Таганрог.

КВАНТОВЫЕ АЛГОРИТМЫ

В последнее время появились новые подходы решения NP-полных проблем, основанные на методах квантового поиска [7–10]. Квантовый поиск анализирует неструктурированные проблемы, которые в общем виде формулируются следующим образом [7]. Задана функция $f(x)$, аргументы x – целые числа $x = 1, 2, ..., N$, причем $f(x)$ принимает значение ноль во всех случаях, кроме $x = w$. Необходимо найти значение w, используя наименьшее число запросов к $f(x)$. Задачи такого типа при небольшом $x < 100$ решаются на основе полного перебора (исчерпывающего поиска) или методом проб и ошибок.

Идею и структуру квантового алгоритма предложил Л. Гровер [7, 8]. Согласно [8], при решении неструктурированной проблемы поиска существует "оракул", определяющий, является ли рассматриваемое решение искомым. Л. Гровер рассматривает N целых чисел индекса $x = 1, 2, ..., N$ как набор ортогональных векторов $\vec{x} = \vec{1}, \vec{2}, ..., \vec{k}$ в N-размерном пространстве Хильберта. Этот шаг алгоритма как бы на языке вычисления кванта ставит в соответствие каждому возможному индексу уникальный собственный вектор. Первоначально готовится пространство

$$\vec{S} = \frac{1}{\sqrt{N}} \sum_{x=1}^{N} \vec{x},$$

т.е. как бы квантовый регистр памяти, содержащий определенное количество суперпозиций, равное количеству всех N [7, 8].

Для реализации поиска это квантовое пространство развивается в общую суперпозицию, которая концентрируется в $\vec{t}$ векторе, определяющем путь до цели поиска. Предлагается процедура квантового кругооборота U. Другими словами, если имеется отличное от нуля совпадение между стартовым пространством $\vec{S}$ и целевым $\vec{t}$, то есть $\vec{t}|U|\vec{S} \neq 0$, тогда можно использовать унитарную процедуру U для выполнения классического поиска цели. Л. Гровер предлагает использовать U и $f(x)$, чтобы построить увеличивающий амплитуду оператор Q, который изменяет амплитуду вероятности от не-цели векторов $\vec{S} \neq \vec{t}$ в цель $\vec{S} = \vec{t}$ [7]. Поведение "оракула" в алгоритме квантового поиска моделируется возвратной функцией $f(x) = 0$, для всех x, w и $f(x) = 1$, для $x = w$.

Для решения NP-полных проблем на графах предлагается анализировать структуру графа, чтобы "выращивать" полные решения, рекурсивно расширяя последовательные частичные решения.

Приведем модифицированный алгоритм квантового поиска [11–13].

1. Начало.
2. Ввод исходных данных.
3. Проверка условий существования инвариантных частей в графе.
4. Анализ математической модели и на его основе построение дерева частичных решений.
5. Суперпозиция частичных решений на основе жадной стратегии и квантового поиска.
6. В случае наличия тупиковых решений – последовательный поиск с пошаговым возвращением.
7. Если набор полных решений построен, то переход к 7, если нет, то к 5.
8. Лексикографический перебор полных решений и выбор из него оптимального или квазиоптимального решения.
9. Конец работы алгоритма.

Приведем укрупненный код алгоритма квантового поиска оптимальных решений в графе на Паскале.

```
Алгоритм
begin {основная программа}
generation: = 0 {установка}
initialize;
repeat {основной цикл}
gen: = gen + 1
generation;
    cycle;
    return;
    statistics (max, avg, min, sumfitness, newsol)
    report (gen)
    oldsol: = newsol
    until (gen ≥ maxgen)
    end {конец основной программы}.
```

Здесь generation (gen) – генерации (итерации) алгоритма; max, avg, min, sumfitness – максимальное, среднее, минимальное, суммарное значение целевой функции; oldsol, newsol – старое и новое решение соответственно. Алгоритм квантового поиска (1 итерация выполняется в блоке repeat).

Работа алгоритма начинается с чтения данных, инициализации случайных решений, вычисления статистических данных и их печати. Процедура report представляет полный отчет обо всех параметрах алгоритма. На основе анализа данных из процедуры report строится график зависимости значений целевой функции от числа генераций. Если функция имеет несколько локальных оптимумов, и мы попали в один из них, то увеличение числа генераций может не привести к улучшению значений целевой функции. В этом случае наступила предварительная сходимость алгоритма. Операция return предусматривает пошаговый возврат до выхода из тупика. Алгоритмы квантового поиска весьма чувствительны к изменениям и перестановкам входных параметров исходной модели. Это говорит о том, что, например, для одного вида модели объекта, представленного матрицей, можно получить решение с одним локальным оптимумом. А для этой же матрицы с переставленными строками и столбцами можно получить другое решение с лучшим локальным оптимумом. Следует отметить, что, изменяя параметры, алгоритмы и схему квантового поиска, в некоторых случаях можно выходить из локальных оптимумов. Эта проблема продолжает оставаться одной из важнейших во всех методах оптимизации.

ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

Генетические алгоритмы отличаются от других оптимизационных и поисковых методов и алгоритмов [1–3]:

- анализируют и преобразуют закодированное множество исходных параметров;
- осуществляют поиск из части популяции или множества популяций (множества альтернативных решений), а не из одного решения;
- используют целевую функцию (функцию пригодности или приспособленности), а не ее различные приращения для оценки качества альтернативных решений;

– используют детерминированные, вероятностные и комбинированные правила анализа оптимизационных задач.

В ГА предварительно анализируется множество входных параметров оптимизационной задачи и находится некоторое множество альтернативных решений, которое называется популяцией. Каждое решение кодируется как последовательность конечной длины в некотором алфавите. ГА работает до тех пор, пока не будет получено решение заданного качества или будет выполнено заданное количество генераций, или на некоторой генерации возникает преждевременная сходимость, когда найден локальный оптимум. Процесс эволюции основан на анализе начальной популяции и ГА начинает свою работу с создания исходного множества альтернативных решений. Затем эти "родительские" решения создают "потомков" путем случайных, направленных или комбинированных преобразований. После этого оценивается эффективность каждого альтернативного решения, и они подвергаются селекции. Во всех моделях эволюции используется принцип "выживания сильнейших", т.е. наименее приспособленные решения устраняются, а лучшие решения переходят в следующую генерацию. Затем процесс повторяется вновь [2–6].

Генетические алгоритмы манипулируют популяцией хромосом на основе механизма натуральной эволюции. Приведем формальное определение ГА.

$$\text{ГА} = \left(P_i^o, N, P_{i,k}^T, T, L_j, A, (\text{ЦФ}, \text{ОГР}, \text{ГУ}), \text{ГО}, t\right),$$

где P_i^o – исходная популяция хромосом альтернативных решений, Pio = (Pio1, Pio2, ..., Pion), Pio1 ∈ Pio – хромосома (альтернативное решение), принадлежащее i-й исходной популяции; N – мощность популяции, т.е. число входящих в нее хромосом, N = | PiT |; PiTk ∈ PiT – k-я хромосома, принадлежащая i-й популяции, находящейся в T поколении эволюции; T = 0, 1, 2, ... – номер поколения, проходящего популяцией во время эволюции, иногда число поколений связывают с числом генераций генетического алгоритма, обозначаемых буквой G; L_j – длина i-й хромосомы (альтернативного решения), т.е. число генов (элементов, входящих в закодированное решение, представленное в заданном алфавите), например, | PiT | = L_j; A – произвольный абстрактный алфавит, в котором кодируются хромосомы, например, A1 = {0, 1}, A2 = {0, 1, 2, ..., 10}, A3 = {0, 1, 2, *}, A4 = {A, B, C, D}, здесь * – метка, означающая любой символ в алфавите A2; (ЦФ, ОГР, ГУ) – целевая функция, ограничения и граничные условия, которые определяются на основе заданной модели исходной решаемой задачи; ГО – генетические операторы, t – критерий окончания работы ГА.

Рассмотрим возможные случаи окончания работы ГА. Если значение глобального оптимума ЦФ известно, то условием окончания работы ГА можно считать нахождение значения ЦФ, превышающей глобальное значение на заданную величину ε в случае минимизации ЦФ. Когда значение глобального оптимума ЦФ неизвестно или приоритетом является время работы ГА, условием окончания ГА считают комбинацию условия не превышения предельно допустимого значения времени с условием нахождения удовлетворительного решения.

АЛГОРИТМЫ РЕШЕНИЯ ЗАДАЧ НА ГРАФАХ

Для решения NP-полных проблем на графах предлагается анализировать структуру графа, чтобы "выращивать" полные решения, рекурсивно расширяя последовательные частичные решения. В задачах на графах важным является нахождение инвариантов. Инвариант графа G = (X, U), где | X | = n, а | U | = m – это число, связанное с G, которое принимает одно и то же значение на любом графе, изоморфном G. Очевидно, что число вершин n и ребер m является простейшими инвариантами графа. Полный набор инвариантов определяет граф с точностью до изоморфизма [5, 6]. Основными инвариантами графа являются числа: цикломатическое, хроматическое, внешней и внутренней устойчивости, клик, полноты, ядер, планарности и т.д.

Покажем на примере графа (рис. 1) определение числа клик на основе квантового алгоритма [8]. Клика – это полный подграф, содержащий наибольшее число ребер. Соответственно число полноты это наибольшее число вершин в клике.

При определении клик графа найдем частичные решения, для каждой вершины графа рекурсивно расширяя "хорошие" решения и устраняя тупиковые решения. На первом шаге квантового поиска для вершины 1 получим следующее частичные решения:

{1,2}, {1,3}, {1,4} –	1 уровень,
{1,2,3}, {1,2,4}, {1,3,4} –	2 уровень,
{1,2,3,4} –	3 уровень.

В результате после суперпозиции частичных решений получим клику Q_1 = {1,2,3,4}, | Q_1 | = 4.

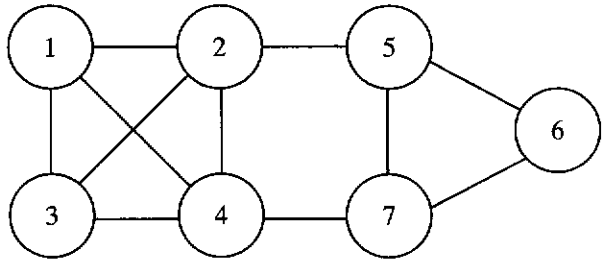

Рис. 1. Граф G

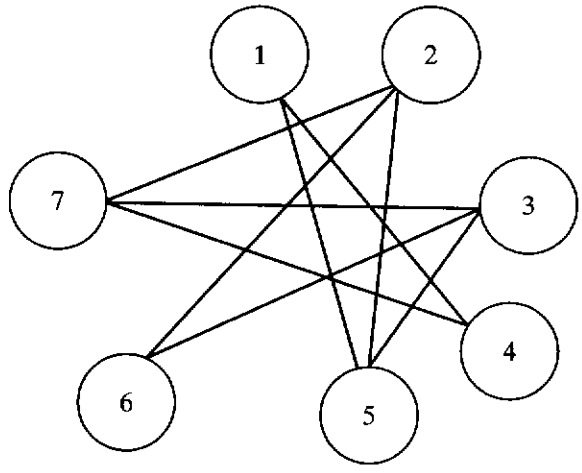

Рис. 2. Граф G = (X, U)

На втором шаге для вершины 2 получим: {2,5}, Q_2 = {2,5}, $|Q_2| = 2$. Для третьей вершины частичных решений нет. Для четвертой вершины имеем {4,7}, Q_3 = {4,7}, $|Q_3| = 2$.

Для пятой вершины получим следующие частичные решения:

{5,6}, {5,7} – 1 уровень,
{5,6,7} – 2 уровень.

После суперпозиции частичных решений получим клику

Q_4 = {5,6,7} $|Q_4| = 3$.

Итак, для графа G (рис. 1) построено семейство клик $Q = \{Q_1, Q_2, Q_3, Q_4\}$. Причем число полноты графа равно 4.

Рассмотрим решение раскраски графа на основе квантового поиска. Раскраской графа G = (X, U) называется разбиение графа на такие непересекающиеся подмножества вершин $X_1 \cap \cap X_2 \cap ... \cap X_e = \varnothing$, $X_1 \cup X_2 \cup ... \cup X_e = X$, что вершины внутри каждого подмножества несмежны. Наименьшее число подмножеств X_i при раскраске называется хроматическим числом графа.

Алгоритм основан на нахождении частичной раскраски для подмножеств вершин. Для определения полной раскраски производится рекурсивно расширение частичных окрасок с возвратом назад в случае тупиковых решений. Элементарный способ состоит в рассмотрении дерева поиска частичных решений заданной глубины. Решение по раскраске находится на "ветвях и листьях" этого дерева [11, 12].

Например, пусть задан граф G = (X,U) из [3], где $|X| = 7$ (рис. 2).

Построим дерево частичной раскраски, приведенное на рис. 3.

Корневая вершина дерева содержит все вершины X графа G. На первом уровне дерева располагаются все вершины графа. Отметим, что возможно случайное и упорядоченное расположение вершин. Наиболее вероятной кажется упорядоченность вершин по уменьшению локальных степеней. Хотя, конечно, требуются экспериментальные исследования на графах различного вида. Выбираем на первом уровне вершину 1 и определяем возможные частичные решения по раскраске, включающие эту вершину. Следуя по дереву вниз, видно, что имеем три частичных решения {1,3}, {1,6}, {1,7}. Дальнейшее расширение этих частичных решений невозможно. Это отмечено знаком '-'. Переходим к вершине 2 и строим два новых частичных решения {2,4,6} и {2,3,4}. Вершина 3 частичных решений не дает. Вершина 4 дает частичное решение {4,5,6} и вершина 5 – {5,6,7}. Вершины 6 и 7 частичных решений не дают. Для получения альтернативных раскрасок предлагается операция суперпозиции. Суть операции в объединении n частичных решений в одно с исключением повторяющихся элементов. Например: {1,2} S {3,4,5} S {1,3,7} = [{1,2}, {3,4,5}, {7}].

Здесь S – знак суперпозиции. Задача – выбрать решение с наименьшим числом подмножеств, что позволит найти раскраску с наименьшим числом цветов.

Для графа G (см. рис. 2) на основе дерева частичных решений выполним следующие операции суперпозиции.

1. Суперпозиция:

{1,3} S {2,4,6} S {4,5,6} – два общих элемента,
{1,3} S {2,4,6} S {5,6,7} – один общий элемент.

Выбираем решение с одним общим элементом. Получим первую раскраску с тремя цветами A,B и C:

{1,3} – A,
{2,4,6} – B,
{5,7} – C.

2. Суперпозиция:

{1,6} S {2,3,4} S {4,5,6} – два общих элемента,
{1,6} S {2,4,6} S {5,6,7} – один общий элемент.

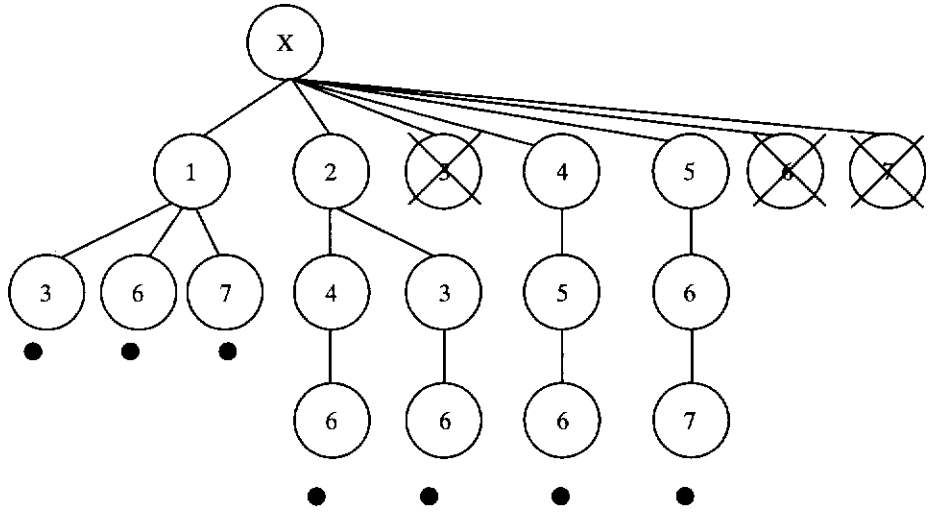

Рис. 3. Дерево частичной раскраски

Выбираем решение с одним общим элементом. Получим вторую раскраску, также с тремя цветами D, E и F :

{1,6} – D,
{2,3,4} – E,
{5,7} – F.

3. Суперпозиция:

{1,7} S {2,3,4} S {5,6,7} – один общий элемент,
{1,7} S {2,3,4} S {4,5,6} – один общий элемент.

Выберем любое решение с одним общим элементом. Получим третью раскраску, также с тремя цветами H, I, K:

{1,7} – H,
{2,3,4} – I,
{5,6} – K.

На рис. 4 показаны три возможных раскраски графа G (см. рис. 2). Соответственно хроматическое число данного графа равно трем.

Приведем эвристическое правило.

При наличии нескольких возможных альтернатив суперпозицию в квантовом алгоритме выполнять для подмножеств, имеющих наименьшее число совпадающих элементов.

Для решения оптимизационных задач на графах в последнее время применяются жадные алгоритмы, которые являются различными модификациями алгоритмов динамического программирования, но жадные алгоритмы проще и быстрее [6]. Основой жадного алгоритма является локально-оптимальный выбор на каждом шаге с прогнозом, что окончательное решение будет оптимальным. Очевидно, что в общем случае жадный алгоритм может привести в локальный минимум, который далек от оптимального. Хотя для многих графовых задач эти алгоритмы дают возможность получать оптимум искомой целевой функции [1–7].

Общая схема псевдокода жадного (Greedy) алгоритма для поиска гамильтонова цикла в графе может быть представлена следующим образом [6].

Greedy – GC (G = (X, U), | X | = n, | U | = m)
1. S ← {1}
2. j ← {1}
3. for i< ← 2 to n
4. do if S ≥ S_i
5. then S ← SU{i}
6. j ← i
7. return S

Здесь множество S состоит из номеров вершин графа от 1 до n. Если поиск ГЦ начинается с первой вершины графа. В противном случае можно перенумеровать вершины или осуществить поиск такого, например, вида: S = 3, 4, 5, ..., n, 1, 2, 3. Величина j – это номер последней вершины, включенной в цикл. В строках псевдокода 3–6 реализуется цикл, ищется путь, начинающийся после вершины j. Если такая цепь найдена, то она включается во множество S и переменной j присваивается ее номер. Такой алгоритм требует в лучшем случае всего $\Theta(n)$ шагов, не считая предварительной сортировки графа. На каждом шаге алгоритм делает выбор, чтобы не было тупиков и повторяющихся вершин.

Следуя [6], считается, что к оптимизационной задаче на графах применим принцип жадного выбора, если последовательность локальных жадных выборов позволяет получить искомое решение. Жадный алгоритм на каждом шаге делает выбор частного оптимального решения. Затем алгоритм делает наилучший выбор среди оставшихся частных решений и т.д.

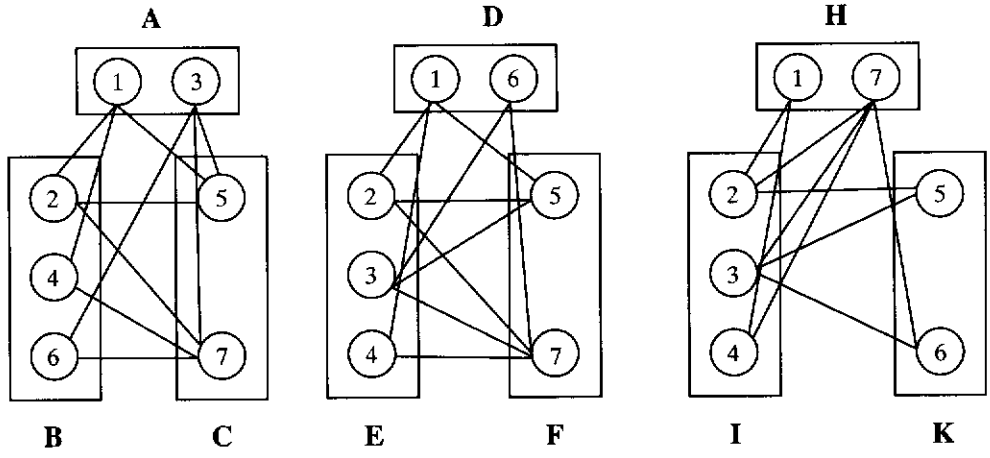

Рис. 4. Возможные раскраски графа G

Приведем ряд эвристик жадного выбора:

Э1 – Жадный выбор на первом и последующих шагах не должен закрывать путь к оптимальному решению.

Э2 – Для упрощения жадного выбора и повышения скорости подзадача, вытекающая после жадного выбора на первом шаге, должна быть аналогична исходной.

Э3 – Задачи, решенные на основе жадных алгоритмов, должны обладать свойством оптимальности для подзадач.

На рис. 5 изображен плоский граф на 20 вершин, изоморфный графу, образованному ребрами додекаэдра, предложенный Гамильтоном. Необходимо найти путь вдоль ребер додекаэдра, проходящий через каждую вершину один раз и возвращающийся в исходную вершину.

Покажем на примере (рис. 5) реализацию композитного алгоритма построения гамильтонова цикла (ГЦ) в графе. Алгоритм использует идеи жадной стратегии и квантового поиска. При этом определяются простые пути из первой заданной вершины, а далее суперпозицией находится гамильтонов цикл, если он существует. На рис. 5 гамильтонов цикл, проходящий по всем вершинам один раз из первой вершины, показан жирными линиями и стрелками.

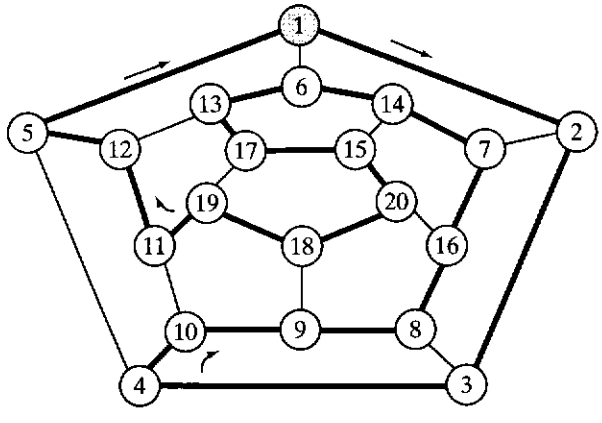

Рис. 5. Гамильтонов цикл в графе

Алгоритм:

1^0	1,2
2^0	$1^0 + 3$
3^0	$2^0 + 4$
4^0	$3^0 + 5$
5^0	$4^0 + 12$
6^0	$5^0 + 11$
7^0	$6^0 + 10$
8^0	$7^0 + 9$
9^0	$8^0 + 8$
10^0	$9^0 + 16$
11^0	$10^0 + 7$
12^0	$11^0 + 14$
13^0	$12^0 + 6$
14^0	$13^0 + 13$
15^0	$14^0 + 17$
16^0	$15^0 + 19$
17^0	$16^0 + 18$
18^0	$17^0 + 20$
19^0	$18^0 + 15$.

В данном алгоритме пункт 1^0 означает, что построен частичный путь из вершины 1 в вершину 2. Пункт 2^0 алгоритма означает, что к пути 1,2 добавлен путь в третью вершину, т.е. имеем 2^0: 1,2,3. Далее все выполняется аналогично. Например, пункт 19^0 означает, что к результату 18^0 пункта (1, 2, 3, 4, 5, 12, 11, 10, 9, 8, 16, 7, 14, 6, 13, 17, 19, 18, 20) добавлен путь в вершину 15.

Теперь необходимо возвращение на шаг назад с попыткой построить ГЦ. Если построение ГЦ

не удается – необходимо осуществлять последовательные возвраты назад до тех пор, пока они не исчерпаны, т.е. построить ГЦ невозможно или будет найден ГЦ.

Тупик в алгоритме, когда необходимо возвратиться на шаг, назад будем помечать знаком ■.

Продолжение алгоритма:

20^0 $17^0 + 20j$
21^0 $16^0 + 18j$
22^0 $15^0 + 19j$
23^0 $14^0 + 17j$
24^0 $13^0 + 13j$
25^0 $12^0 + 6j$
26^0 $11^0 + 14j$
27^0 $10^0 + 7j$
28^0 $9^0 + 16j$
29^0 $8^0 + 8j$
30^0 $7^0 + 9j$
31^0 $6^0 + 10j$
33^0 $5^0 + 11j$
34^0 $4^0 + 12j$ ← здесь возможны несколько различных путей, но в итоге они приведут в тупик, так как не удастся найти выход к вершине 1, пройдя $^\circ\!-\!1^\circ = 19$ вершин
35^0 $3^0 + 5j$
36^0 $2^0 + 4j$.

В пунктах 20^0–36^0 алгоритма мы последовательно спускались от 15 вершины до вершины 4 ($15 \to 20 \to 18 \to 19 \to 17 \to 13 \to 6 \to 14 \to 7 \to \to 16 \to 8 \to 9 \to 10 \to 11 \to 12 \to 5 \to 4$).

Продолжение алгоритма:

37^0 $36^0 + 10$
38^0 $37^0 + 9$
39^0 $38^0 + 8$
40^0 $39^0 + 16$
41^0 $40^0 + 7$
42^0 $41^0 + 14$
43^0 $42^0 + 6$
44^0 $43^0 + 13$
45^0 $44^0 + 17$
46^0 $45^0 + 15$
47^0 $46^0 + 20$
48^0 $47^0 + 18$
49^0 $48^0 + 19$
50^0 $49^0 + 11$
51^0 $50^0 + 12$
52^0 $51^0 + 5$
53^0 $52^0 + 1$.

Итак, после пункта 53^0 алгоритма получен гамильтонов цикл. Приведем его по мере возврата от пункта 53^0 до начала (см. рис. 1).

ГЦ = 1, 5, 12, 11, 19, 18, 20, 15, 17, 13, 6, 14, 7, 16, 8, 9, 10, 4, 3, 2, 1.

Отметим, что в пункте 45^0 после вершины 13 алгоритм сделал выбор вершины 17 из двух возможных вершин: 12 и 17. Если бы алгоритм выбрал вершину 12, то потребовалось бы дополнительно 15 шагов, и алгоритм вернулся бы к выбору вершины 17.

АРХИТЕКТУРЫ ПОИСКА

На рис. 6 приведены схемы взаимодействия квантовых и генетических алгоритмов. Очевидно, что данные схемы можно взять как строительные блоки и наращивать иерархически. При этом возможно построить схему последовательного или параллельного совместного поиска любой сложности.

Отметим также, что квантовый поиск может ускорить классический случайный алгоритм, создавая суперпозиции частотных решений, увеличивая пространство поиска искомого решения.

Предлагается модифицированная схема совместного поиска, состоящая из трех основных блоков (рис. 7). Первый блок назовем препроцессором. Здесь производится создание одной или некоторого множества начальных популяций. Второй блок состоит из четырех этапов: выбор представления решения; разработка операторов случайных, направленных и комбинированных изменений; определение законов выживания решения; рекомбинация. Третий блок назовем постпроцессором. Здесь реализуются принципы эволюционной адаптации к внешней среде (лицу, принимающему решение) и самоорганизации. Отметим, что строительные блоки совместного квантового и генетического поиска (рис. 6) могут эффективно работать в составе процессора и постпроцессора. Преимущество такой архитектуры совместного поиска состоит в том, что в ней все уровни связаны с уровнем внешней среды и могут общаться между собой.

Для управления и реализации процессом совместного поиска используются следующие принципы [6]:

– Принцип целостности. В квантовых и генетических алгоритмах значение целевой функции альтернативного решения не сводится к сумме целевых функций частичных решений.

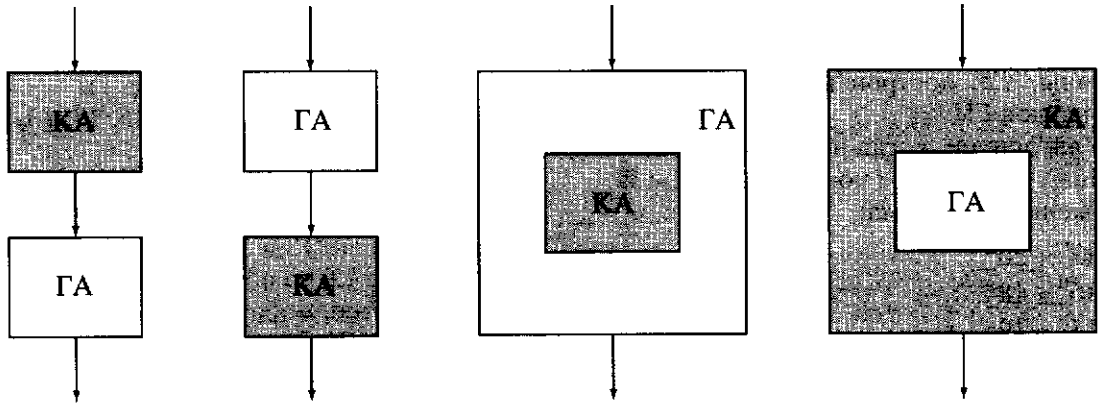

Рис. 6. Схемы взаимодействия генетических и квантовых алгоритмов

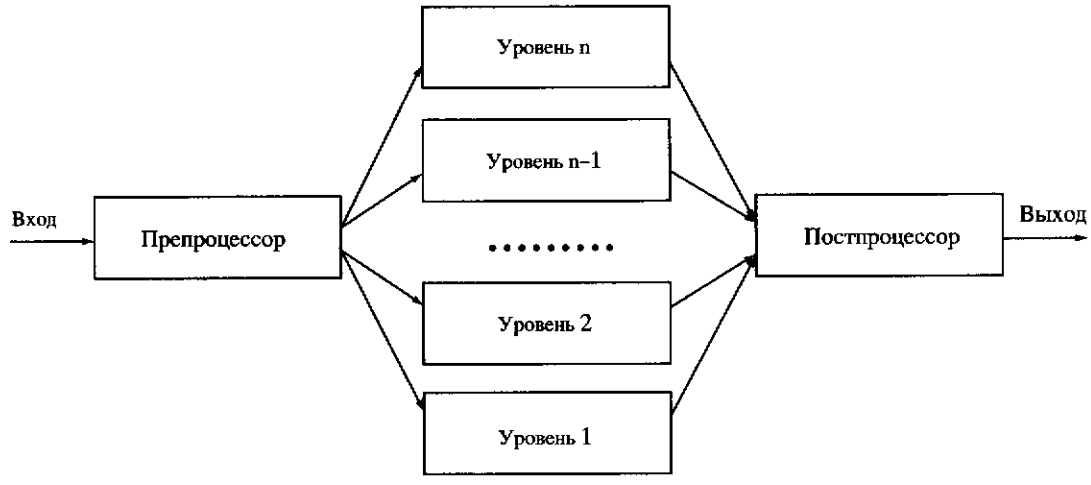

Рис. 7. Горизонтальная архитектура совместного поиска

- Принцип чувствительности к начальным условиям. Результат работы квантовых и генетических алгоритмов зависит от представления входных данных исследуемой модели.
- Принцип дополнительности. При решении оптимизационных задач возникает необходимость использования различных не совместимых и взаимодополняющих моделей эволюции и исходных объектов.
- Принцип неточности. При росте сложности анализируемой задачи уменьшается возможность построения точной модели.
- Принцип управления неопределенностью. Необходимо вводить различные виды неопределенности в квантовые и генетические алгоритмы.
- Принцип соответствия. Язык описания исходной задачи должен соответствовать наличию имеющейся о ней информации.
- Принцип "007". Использовать только те входные данные, которые необходимы для решения задачи.
- Принцип разнообразия путей развития. Реализация квантовых и генетических алгоритмов многовариантна и альтернативна. Существует много путей эволюции. Основная задача – выбрать путь, приводящий к получению оптимального решения.
- Принцип единства и противоположности порядка и хаоса. "Хаос не только разрушителен, но и конструктивен", т.е. в хаосе области допустимых решений обязательно содержится порядок, определяющий искомое решение.
- Принцип совместимости и разделительности. Процесс эволюции носит поступательный, пульсирующий или комбинированный характер. Поэтому модель синтетической эволюции должна сочетать все эти принципы.

– Принцип иерархичности. Квантовые и генетические алгоритмы могут подстраиваться сверху вниз и снизу вверх.
– Принцип "Бритвы Оккама". Нежелательно увеличивать сложность архитектуры поиска без необходимости.
– Принцип гомеостаза. Квантовые и генетические алгоритмы конструируются таким образом, чтобы любое полученное альтернативное решение не выходило из области допустимых.

На рис. 8 показан пример реализации архитектуры соподчинения в совместном поиске.

Здесь все уровни связаны с блоком внешней среды и могут общаться между собой. Эта структура раскрывает подробно блоки совместного поиска. Приведенные схемы ориентированы на эффективное решение оптимизационных задач [1, 3, 5, 6]. Данные блоки могут быть раскрыты и другим способом в зависимости от начальных условий и знаний о решаемой задаче на графах.

Характер поведения алгоритмов определяет скорость роста количества выполненных операций при возрастании объема входных данных. Наилучшим случаем является такой набор данных (числа вершин и ребер графа), при котором алгоритм выполняется за минимальное время. В лучшем случае временная сложность комбинированного (объединенного квантового и генетического) алгоритма (ВСА) ориентировочно составляет $O(n)$. В худшем случае ВСА = $O(n!)$. В среднем случае производят анализ определения различных групп, на которые разбивают возможные входные наборы данных. Тогда среднее время работы алгоритма вычисляется по формуле $A(n) = \sum_{i=1}^{q} p_i t_i$, где n – размер входных данных (число вершин графа), q – число групп входных данных, p_i – вероятность того, что входные данные принадлежат группе с номером i, t_i – время, необходимое алгоритму для обработки вершин графа из группы с номером i. Отметим, что для простоты в данном алгоритме можно выделить 2 группы вершин. В первой существует один возможный путь для дальнейшего построения клик графа. Во второй существует $(q - 1)!$ и меньше путей для построения клик.

ЗАКЛЮЧЕНИЕ

Отметим, что можно организовать различное количество связей внутри схемы совместного поиска между блоками по принципу полного графа, по принципу звезды и т.д. Такие схемы в случае наличия большого количества вычислительных ресурсов могут быть доведены до N блоков. Причем $N - 1$ блоков могут параллельно осуществлять эволюционную адаптацию и через блоки миграции обмениваться лучшими представителями решений. Последний блок собирает лучшие решения, может окончить результат работы или продолжить оптимизацию.

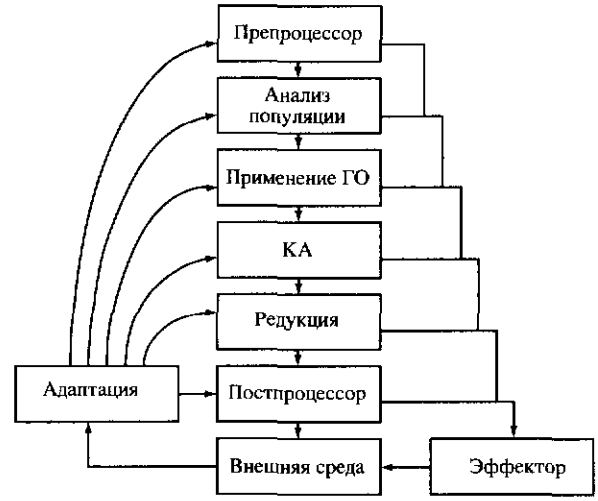

Рис. 8. Пример реализации архитектуры соподчинения в совместном поиске

Такие стратегии решения задач на графах позволяют учитывать влияние внешней среды и знания о решаемых задачах и в отличие от существующих методов позволяют во многих случаях выходить из локальных оптимумов.

СПИСОК ЛИТЕРАТУРЫ

1. *Люггер Д.Ф.* Искусственный интеллект: стратегии и методы решения сложных проблем. М.: Издательский дом "Вильямс", 2003. 864 с.
2. *Holland John H.* Adaptation and Artificial Systems: An Introductory Analysis with Application to Biology, Control, and Artificial Intelligence. USA: University of Michigan, 1975. P. 211.
3. *Goldberg David E.* Genetic Algorithms in Search, Optimization and Machine Learning. USA: Addison-Wesley Publishing Company, Inc., 1989. P. 412.
4. *Koza J.R.* Genetic Programming. Cambridge // MA: MIT Press, 1998. P. 819.
5. *Курейчик В.М.* Генетические алгоритмы и их применение: Монография. Таганрог: Изд-во ТРТУ, 2002. 242 с.
6. *Емельянов В.В., Курейчик В.М., Курейчик В.В.* Теория и практика эволюционного моделирования. М.: Физматлит, 2003. 432 с.

7. *Grover L.K.* // Proc. 28th Ann. ACM Press, New York, 1996. P. 212–219.
8. *Grover L.K.* // Physical Rev. Letters. 2000. V. 85. No. 6. P. 1334–1337.
9. *Williams C.P.* // Computing in sciences and engineering. March–April 2001. P. 44–51.
10. *Валиев К.А., Кокин А.А.* Квантовые компьютеры: надежда и реальность. Москва-Ижевск: Изд-во НИЦ РХД, 2002. 320 с.
11. *Курейчик В.М.* // Перспективные информационные технологии и интеллектуальные структуры. № 2(18). 2004. С. 14–17.
12. *Курейчик В.М.* // Труды конференций IEEE AIS'04, CAD-2004. М.: Физматлит, 2004. С. 12–19.
13. *Курейчик В.М.* // Известия ТРТУ. 2004. № 3. С. 29–34.
14. *Тарасов В.Б.* От многоагентных систем к интеллектуальным организациям: философия, психология, информатика. М.: Изд-во Эдиториал УРСС, 2002. 352 с.
15. *Курейчик В.В.* Эволюционные, синергетические и гомеостатические методы принятия решений. Таганрог: Изд-во ТРТУ, 2001. 221 с.
16. *Макконелл Дж.* Анализ алгоритмов. Вводный курс. М.: Техносфера, 2004. С. 300–302.

QUANTUM AND GENETIC ALGORITHMS AS A NEW TECHNOLOGY OF EVOLUTIONARY SEARCH

V.M. Kureichik

A new technology of decision of optimization and combinatorial logical problems on graph models based on composite (integrated quantum and genetic) algorithms is considered. It allows to get sets of local and optimal decisions and to develop heuristic algorithms with polynomial speed of operations number growth depending on input data volume.

REFERENCES

1. Lyugger D.F. 2003. *Iskusstvennyy intellekt: strategii i metody resheniya slozhnykh problem*. [*Artificial Intelligence: Strategies and methods for solving complex problems*]. Moscow, Publishing House “Williams”: 864 p. (In Russian).
2. Holland John H. 1975. *Adaptation and Artificial Systems: An Introductory Analysis with Application to Biology, Control, and Artificial Intelligence*. USA, University of Michigan: 211 p.
3. Goldberg David E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. USA, Addison-Wesley Publishing Company, Inc.: 412 p.
4. Koza J.R. 1998. *Genetic Programming*. Cambridge, MA, MIT Press: 819 p.
5. Kureychik V.M. 2002. *Geneticheskie algoritmy i ikh primenenie*. [*Genetic algorithms and their application*]. Taganrog, TRTU [TSURE] Publ.: 242 p. (In Russian).
6. Emel'yanov V.V., Kureychik V.M., Kureychik V.V. 2003. *Teoriya i praktika evolyutsionnogo modelirovaniya*. [*Theory and practice of evolutionary modeling*]. Moscow, Fizmatlit Publ.: 432 p. (In Russian).
7. Grover L.K. 1996. In: *Proc. 28 th Ann*. New York, ACM Press: 212–219.
8. Grover L.K. 2000. *Physical Rev. Letters*. 85(6): 1334–1337.
9. Williams C.P. 2001. *Computing in sciences and engineering*. (March–April): 44–51.
10. Valiev K.A., Kokin A.A. 2002. *Kvantovye komp'yutery: nadezhda i real'nost'. Regulyarnaya i khaoticheskaya dinamika*. [*Quantum computers: hopes and reality. Regular and chaotic dynamics*]. Moscow; Izhevsk, NITs RKhD Publ.: 320 p. (In Russian).
11. Kureychik V.M. 2004. *Perspektivnye informatsionnye tekhnologii i intellektual'nye struktury*. 2(18): 1–17. (In Russian).
12. Kureychik V.M. 2004. In: *Trudy konferentsiy IEEE AIS’04. CAD-2004*. [*Proceedings of IEEE AIS’04. CAD-2004*]. Moscow, Fizmatlit Publ.: 1–19. (In Russian).
13. Kureychik V.M. 2004. *Izvestiya TRTU*. (3): 29–34. (In Russian).
14. Tarasov V.B. 2002. *Ot mnogoagentnykh sistem k intellektual'nym organizatsiyam: filosofiya, psikhologiya, informatika*. [*From multi-agent systems to intellectual organizations: philosophy, psychology, computer science*]. Moscow, “Editorial URSS” Publ.: 352 p. (In Russian).
15. Kureychik V.V. 2001. *Evolyutsionnye, sinergeticheskie i gomeostaticheskie metody prinyatiya resheniy*. [*Evolutionary, synergetic and homeostatic methods of decision making*]. Taganrog, TRTU [TSURE] Publ.: 221 p. (In Russian).
16. Makkonell Dzh. 2004. *Analiz algoritmov. Vvodnyy kurs*. [*Analysis of Algorithms. Introductory Course*]. Moscow, “Tekhnosfera” Publ.: 30–302. (In Russian).