Обобщение генетических алгоритмов и алгоритмов схемы МИВЕР

Дегтерев А.С. (1) (nii_suvpt@wave.krs.ru), Канашкин Ф.В. (2), Сумароков А.Д. (2)

(1)ФГУП ЦКБ «Геофизика»,
(2)НИИ систем управления, волновых процессов и технологий

Метод изменяющихся вероятностей (МИВЕР) [1] представляет семейство алгоритмов, имеющих общую схему. В самом общем виде эта схема применительно к задаче псевдобулевой оптимизации:

$$f(X) \to \max_{X \in D},$$

где $f : B_{2^n} \to R^1$ - поле действительных чисел, $B_{2^n} = \{X : x_j \in B_2, j = \overline{1,n}\}$; $B_2 = \{0,1\}$; $D \subset B_{2^n}$ - некоторое подмножество множества булевых переменных, определяемое заданной системой ограничений, выглядит следующим образом.

1.Задаются начальные значения компонент вектора вероятностей $P^0 = \{p_1^0,...,p_n^0\}$, где $p_j = P\{x_j = 1\}$ - вероятность присвоения единичного значения j-й $(j = \overline{1,n})$ компоненте вектора $X \in D$.

2.Случайным образом (в соответствии с распределением вероятностей P^k, $k = \overline{1,R-1}$), независимо выбираются r векторов $X \in D$, $i = \overline{1,r}$.

3.Вычисляются соответствующие значения функционала $f(X^i), i = \overline{1,r}$.

4.Выбираются требуемые (по конкретному алгоритму) значения функционала из совокупности $f(X^i), i = \overline{1,r}$.

5.По результатам п.4 изменяются (в соответствии с правилом конкретного алгоритма) компоненты вектора вероятностей P^k.

6.Пункты 2-5 повторяются R раз.

7.За решение задачи принимается вектор X^*, определяемый из условия $f(X^*) = \max_{k=1,R} f^k_{\max}$ ($f^k_{\max} = \max_{k=1,r} f(X^i)$ - при k-м повторении п.4).

Генетические алгоритмы оптимизации относятся к классу статистических популяционных алгоритмов [2]. В основу генетического алгоритма оптимизации положена упрощённая модель популяции живых существ. В процессе работы алгоритма имитируется жизнь этой популяции в некоторых искусственных условиях. Популяция в генетическом алгоритме представляет собой набор индивидов, каждому из которых соответствуют генотип и фенотип. Генотип (хромосома) – это набор символов внутреннего языка генетического алгоритма (генов), над которыми производятся генетические операции, в состав которых вхо-

дят скрещивание (рекомбинация) и мутация. Фенотип – это внешний облик индивида, в нашем случае решение задачи. Фенотип и генотип связаны операциями кодирования и декодирования. В некоторых случаях фенотип и генотип совпадают. Так, например, при решении рассматриваемой задачи псевдобулевой оптимизации, в качестве генотипа используется непосредственно булев вектор. Таким образом, в общем случае, каждому индивиду соответствует точка в поисковом пространстве, значение оптимизируемого критерия в которой, является мерой качества (пригодности) индивида. Свои же признаки (гены) индивид передаёт в следующее поколение посредством генетических механизмов и шансы его передать эти признаки тем выше, чем больше его пригодность.

Генетические алгоритмы имеют следующую общую схему.

1. Инициализация популяции. На этом этапе хромосомы индивидов заполняются нулями и единицами в случайном порядке с равномерным распределением.
2. Оценка популяции. Вычисляются значения целевой функции в точках, соответствующих индивидам популяции. Отбирается фиксированная доля лучших индивидов в соответствии с заданной «элитарностью» алгоритма. Отобранные называются «выжившими» и допускаются к размножению. Таким индивидам назначается «пригодность». Пригодность индивида тем больше, чем лучше значение оптимизируемой функции в соответствующей ему точке.
3. Условие остановки. Обычно алгоритм останавливают после некоторого числа поколений за время которых не было найдено лучшего решения, либо за время которых не произошло улучшение среднего качества популяции. Возможен также алгоритм с фиксированным числом поколений.
4. Селекция и рекомбинация. Селекция есть выбор одного, двух или более индивидов (в различных вариантах алгоритма) для скрещивания (рекомбинации).
5. Мутация есть случайное (с заданной малой вероятностью) изменение (инвертирование) любого гена (бита) хромосомы. Обычно вероятность мутации задаётся близкой к 1/n, где n – число генов в хромосоме. Мутация нужна для сообщения новых признаков индивидам (возможно полезных), что обогащает генофонд популяции (увеличивает разнообразие) и предотвращает преждевременную сходимость алгоритма (стагнацию).

При ближайшем рассмотрении оказывается, что между алгоритмами схемы МИВЕР и генетическими алгоритмами много общего. И те, и другие работают с булевым вектором, который в генетическом алгоритме называется хромосомой, и может кодировать набор дискретных и вещественных величин. В работе обоих типов алгоритмов можно обобщённо выделить одни и те же этапы: генерация набора решений для изучения оптимизируемой функции, генерация нового набора решений с учётом информации, полученной от вычисления функции в точках (решениях) предыдущего набора. С целью сбора и обобщения информации об оптимизируемой функции генетические алгоритмы осуществ-

ляют набор генетических операций (селекция, рекомбинация, мутация), а алгоритмы схемы МИВЕР – пересчёт вектора апостериорных вероятностей.

Рассмотрим вариант генетического алгоритма с ранговой селекцией и числом родителей разного пола (передающими свои гены одному потомку) равным числу индивидов, отобранных для скрещивания (выживших). При ранговой селекции выбор индивида для рекомбинации производится с вероятностью:

$$r^i = \frac{R^i}{\sum_{k=1}^{s} R^k} = \frac{i}{\sum_{k=1}^{s} k} = \frac{2i}{s(s+1)},$$

где R^i - «ранг» индивида (номер, после сортировки по убыванию критерия пригодности в соответствующей индивиду точке), а s - число выживших индивидов (для остальных r=0). Таким образом, с вероятностью r^i для k-й компоненты вектора $X^{l,next}$ (l-го индивида последующего поколения) выбирается значение $X^i_{k.}$ То есть оказывается, что при рекомбинации в этом алгоритме реализуется разыгрывание каждого гена (бита) потомка с вероятностями: $(p_1, p_2, \ldots, p_n)$, где $p_j = P\{x_j = 1\}$ - вероятность присвоения единичного значения j-й $(j = \overline{1,n})$ компоненте вектора $X \in D$, вычисляется как:

$$\mathrm{p_i} = \sum_{k=1}^{s} X_i^k r^k,$$

где $\sum_{k=1}^{s} r^k = 1$ - ранги (пригодности) родителей.

Для алгоритма с числом родителей одного потомка не равным числу выживших индивидов подобные вероятности выписать нельзя, т.к. они оказались бы зависимы.

Мутация в генетическом алгоритме – это случайное инвертирование любого бита с некоторой малой вероятностью p^m - вероятностью мутации. Если вероятность того, что этот бит установлен в единицу – p_i, то после мутации эта вероятность будет равна: $\hat{p}_i = p^1 + p^2$, где p^1 - вероятность того, что бит был установлен в единицу и при мутации не изменится, а p^2 - вероятность того, что бит был установлен в ноль и при мутации был инвертирован. Таким образом: $\hat{p}_i = p_i(1-p^m)+(1-p_i)p^m = p_i(1-2p^m)+p^m$, то есть мутация - это линейное преобразование вероятности. Следовательно, механизм мутации не позволяет вероятностям p_i обращаться в ноль либо единицу, что означало бы прекращение поиска по i-й компоненте и, возможно, стагнацию.

Приведём алгоритм схемы МИВЕР, который с помощью вектора вероятностей «эмулирует» работу генетического алгоритма с ранговой селекцией и числом родителей каждого потомка, равным числу выживших индивидов.

1. Задаются начальные значения компонент вектора вероятностей $\hat{P}^0 = (\hat{p}_1^0, ..., \hat{p}_n^0)$, где $p_j = P\{x_j = 1\}$ - вероятность присвоения единичного значения j-й $\left(j = \overline{1,n}\right)$ компоненте вектора $X \in D$.

2. Случайным образом в соответствии с распределением вероятностей $\hat{P}^l$, независимо выбираются r векторов $X^i \in D$, $\left(i = \overline{1,r}\right)$.

3. Вычисляются соответствующие значения функции $f(X^i)$, $\left(i = \overline{1,r}\right)$.

4. Выбираются s «лучших» векторов X^i (доставляющих наибольшее значение функции $f(X^i)$). Им назначаются пригодности $r^i = \frac{2i}{s(s+1)}$.

5. По результатам п. 4 вычисляются компоненты вектора вероятностей P^{l+1} по правилу: $p_i^{l+1} = \sum_{k=1}^{s} X_i^k r^k$.

6. В соответствие с заданной вероятностью мутации вычисляются вероятности $\hat{p}_1^{l+1}, ..., \hat{p}_n^{l+1}$ по правилу: $\hat{p}_i^{l+1} = p_i^{l+1}(1-2p^m)+p^m$.

7. Пункты 2-5 выполняются R раз до выполнения условия остановки.

8. За решение задачи принимается вектор X^*, определяемый из условия $f(X^*) = \min_{l=\overline{1,R}} f_{\min}^l$, где: $f_{\min}^l = \min_{i=\overline{1,r}} f(X^i)$ (при k-м повторении п. 4)

Попытаемся сконструировать новый алгоритм, использующий условные вероятности $p_{i|j} = P\{x_i = 1 \mid x_j = 1\}$ – вероятности того, что i-й бит окажется установленным в 1, при условии, что j-й бит уже единица. Из формулы Байеса:

$$P(A \mid B) = \frac{P(AB)}{P(B)},$$

следовательно:

$$p_{i|j} = \frac{p_{(i,j)}}{p_j}, \qquad (1)$$

где $p_{(i,j)} = \sum_{k=1}^{s} X_i^k X_j^k r^k$, $p_j = \sum_{k=1}^{s} X_j^k r^k$.

Из соображений экономии памяти компьютера целесообразно вычислить один раз и хранить величины $p_{(i,j)}$, $i = \overline{(1,n)}$, $j < i$ и p_i, $i = \overline{(1,n)}$, а величины $p_{i|j}$ и $p_{i|\bar{j}}$ (вероятность того, что i-й бит будет установлен в единицу, если j-й бит - ноль) вычислять по мере необходимости. При этом так как

$P(A \mid \overline{B}) = \dfrac{P(A\overline{B})}{P(\overline{B})} = \dfrac{P(A) - P(AB)}{1 - P(B)}$, то:

$$p_{i|\bar{j}} = \frac{p_{(i,\bar{j})}}{p_{\bar{j}}} = \frac{p_i - p_{(i,j)}}{1 - p_j}. \qquad (2)$$

Обратим внимание на то, что в формулах (1) и (2) в делителе стоит величина, которая, очевидно, может быть нулём, а так как $p_{(i,j)} \le p_j$, то в результате мы получаем неопределённость «ноль на ноль». Такая ситуация возникает в случае, если $p_j = 0$, но из за ненулевой вероятности мутации произошло событие $X_j^k = 1$, и нам нужно вычислить величину $p_{i|j}$. Или наоборот, $p_j = 1$, $X_j^k = 0$, и нужно вычислить величину $p_{i|\bar{j}}$. В этом случае (когда нет информации о $p_{i|j}$) будем использовать $p_{i|j} = p_i$.

Одна из сложностей состоит в том, что невозможно точно разыграть многомерную зависимую случайную величину так же как независимую (в алгоритме с вектором вероятностей) используя лишь попарные совместные вероятности ($P(AB)$ или $p_{(i,j)}$). Например, имеем события A, B, C.

Разыгрываем: событие A, используя $P(A)$,

событие B, используя $P(B \mid A) = \dfrac{P(AB)}{P(A)}$,

и теперь, чтобы разыграть событие C нужно знать $P(C \mid AB) = \dfrac{P(ABC)}{P(AB)}$.

Очевидно, что в нашем случае невозможно вычислять и хранить совместные вероятности соответствующие всем возможным сочетаниям событий.

Генерацию решений предлагается производить по следующей схеме. Биты заполняются в случайном порядке, при этом первый бит разыгрывается с вероятностью $\hat{p}_i$ ($\hat{p}_i = p_i(1-2p^m)+p^m$ - преобразование мутации), второй с вероятностью $\hat{p}_{j|i}$ (где $\hat{p}_{j|i} = p_{j|i}(1-2p^m)+p^m$), либо $\hat{p}_{j|\bar{i}}$ (в зависимости от результата розыгрыша *i*-го бита), а все последующие с той $\hat{p}_{k|l}$ или $\hat{p}_{k|\bar{l}}$ которая максимально отличается от $\hat{p}_k$ (перебираются те *l*, соответствующие которым биты уже разыграны). Предполагается, что такой механизм будет генерировать решения с распределением, близким к желаемому. Очевидно, что алгоритм с подобным механизмом генерации решений не будет учитывать информацию, содержащуюся в многоместных совместных вероятностях.

Запишем алгоритм с условными вероятностями.

1. Задаются начальные значения компонент вектора вероятностей $\hat{P}^0 = (\hat{p}_1^0, ..., \hat{p}_n^0)$, где $p_j = P\{x_j = 1\}$ - вероятность присвоения единичного значения j-й $(j = \overline{1,n})$ компоненте вектора $X \in D$ и вероятностей $\hat{p}_{(i,j)}^0$, где $\hat{p}_{(i,j)}^0 = P\{x_i = 1 \wedge x_j = 1\}$- вероятность присвоения единичного значения i-й и j-й компонентам вектора $X \in D$.

2. Случайным образом в соответствии с вероятностями $\hat{p}_{(i,j)}^0$, $\hat{p}_i^0$ и p^m, по вышеприведённой схеме независимо выбираются r векторов $X^i \in D$, $\left(i = \overline{1,r}\right)$.

3. Вычисляются соответствующие значения функции $f(X^i)$, $\left(i = \overline{1,r}\right)$.

4. Выбираются s «лучших» векторов X^i (доставляющих наибольшее значение функции $f(X^i)$). Им назначаются пригодности $r^i = \frac{2i}{s(s+1)}$.

5. По результатам п. 4 вычисляются вероятности $p_{(i,j)}^{l+1}$ и p_i^{l+1} в соответствии с правилом: $p_i^{l+1} = \sum_{k=1}^{s} X_i^k r^k$, $p_{(i,j)}^{l+1} = \sum_{k=1}^{s} X_i^k X_j^k r^k$.

6. Пункты 2-5 выполняются R раз до выполнения условия остановки.

За решение задачи принимается вектор X^*, определяемый из условия $f(X^*) = \min_{l=1,R} f_{\min}^l$, где: $f_{\min}^l = \min_{i=1,r} f(X^i)$ - при k-м повторении п. 4.

На разных типах тестовых функций было проведено исследование сравнительной эффективности генетических алгоритмов и предложенных обобщенных алгоритмов схемы МИВЕР. Тестировались: алгоритм с условными вероятностями, алгоритм схемы МИВЕР с вектором безусловных вероятностей, генетический алгоритм с одноточечным скрещиванием, генетический алгоритм с равномерным скрещиванием.

Все алгоритмы имели следующие параметры: объём популяции – 200, вероятность мутации - 0,01, элитарность - 50%, селекция - ранговая. Размерность задачи псевдобулевой оптимизации – 32.

Число вычислений функции – это произведение объёма популяции на номер такта алгоритма (поколения) во время которого был найден оптимум. Число поколений было ограничено тридцатью. Все алгоритмы запускались на всех задачах 50 раз. Надёжность определялась как процент прогонов алгоритма, приведших к успеху (нахождению оптимума с требуемой точностью за 30 поколений). Число вычислений функции усреднялось по успешным прогонам алго-

ритма (безуспешный запуск алгоритма не вносил вклад в итоговое среднее число вычислений функции, но уменьшал оценку надёжности). На выбранной задаче один алгоритм считался лучше другого, если обеспечивал большую надёжность нахождения оптимума, если надёжности были равны, то во внимание принималось число вычислений функции.

Из анализа результатов можно сделать следующие выводы.

Генетический алгоритм с равномерным скрещиванием лучше алгоритма с одноточечным скрещиванием. Исключение – полимодальная функция, локальные минимумы которой расположены в узлах ортогональной регулярной решётки, что удобно для алгоритма с одноточечным скрещиванием. Так как при таком варианте скрещивания потомок получает хотя бы одну координату от одного из родителей без изменений, то с большой вероятностью потомок попадёт в другой, неисследованный оптимум. Поэтому генетический алгоритм с одноточечным скрещиванием хорошо работает на тех полимодальных функциях, у которых оптимумы расположены на регулярной решётке. Найдя несколько оптимумов, он легко находит и все остальные, в том числе глобальный.

Алгоритм с вектором вероятностей (эмулирующий генетический алгоритм с большим числом полов) работает лучше других алгоритмов на простых задачах.

На самых сложных задачах лучшие результаты показал генетический алгоритм с равномерным скрещиванием.

Алгоритм с условными вероятностями хорошо работает на функциях средней и высокой сложности Исключением является полимодальная функция с локальными оптимумами в узлах регулярной решетки, для которой, как уже указывалось, лучший результат показал генетический алгоритм с одноточечным скрещиванием.

Из практики применения генетических алгоритмов ранее был известен следующий результат: чем меньше полов (родителей) тем более успешно алгоритм справляется с более сложными задачами. Теперь этот результат может быть объяснён на более высоком уровне. Чем меньше число полов (до двух минимум), тем больше алгоритм использует информацию о зависимости битов (генов) и тем более сложные задачи он успешно решает. Большее же число полов ведёт к меньшему использованию информации о зависимости битов и лучшему обобщению информации об оптимизируемой функции, что позволяет такому алгоритму быстрее находить оптимум сравнительно простой функции.

Литература:

1. *Антамошкин А.Н.* Оптимизация фуекционалов с булевыми переменными. – Томск: Изд-во Томск. ун-та, 1987, - 104 с.
2. *Стариков А.* BaseGroup Labs. Генетические алгоритмы – математический аппарат. URL: http://basegroup.ru/genetic/math.htm.