ИССЛЕДОВАНИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

Методические указания к лабораторной работе

1. ЦЕЛЬ РАБОТЫ

Целью работы является приобретение практических навыков работы с эволюционными методами оптимизации на примере генетического алгоритма.

2. КРАТКАЯ ТЕОРЕТИЧЕСКАЯ СПРАВКА

2.1. Принцип эволюционных вычислений

Эволюционные вычисления - это термин, относящийся к нескольким методам оптимизации и синтеза систем, объединенных тем, что все они используют понятие *эволюции* объектов, входящих в систему. Это понятие допускает биологическую трактовку в смысле эволюции живых существ.

С точки зрения теории систем эволюция представляет собой процесс адаптации системы через изменение ее параметров под воздействием внешних условий. Поэтому эволюционные вычисления трактуются как развитие методов *теории адаптивных систем*.

Суть эволюционного подхода в общем виде сводится к следующему.

1. Фиксируется множество объектов $\boldsymbol{X}$, обладающих параметрами и связанных друг с другом посредством структуры. Среди этих объектов необходимо выбрать наилучшие в смысле некоторого критерия. Критерий оптимальности формируется на основе свойств объектов и не обязательно существует в виде аналитических выражений. Важно, что каждому объекту из множества $\boldsymbol{X}$ сопоставлено значение критерия $\boldsymbol{F}(\boldsymbol{X})$.

2. Природа исследуемого множества объектов произвольна, поэтому необходимо построить *представление* $\boldsymbol{S}$ исходного множества объектов в

другом, конечном множестве, обладающем, например, структурой пространства. Представление $\boldsymbol{\varphi}$**:** $\boldsymbol{X} \rightarrow \boldsymbol{S}$ описывает связь между исследуемыми объектами, которые выступают в качестве потенциальных решений задачи поиска экстремума, и объектами, управлением и манипулированием которых занимается поисковый алгоритм. Существует обратное представление $\boldsymbol{\varphi}^{-1}$: $\boldsymbol{S} \rightarrow \boldsymbol{X}$ и каждому вновь сгенерированному элементу представления $s \in \boldsymbol{S}$ соответствует элемент в множестве $x \in \boldsymbol{X}$.

3. Процесс оптимизации состоит в построении множества объектов - решений $x^* \in \boldsymbol{X}^*$, для которых выполняются условия:

$$x^* = \arg\max_{s \in \boldsymbol{S}} \boldsymbol{F}\,[\boldsymbol{\varphi}^{-1}(s)]$$

В процессе оптимизации исходное множество $\boldsymbol{X}$ развивается или *эволюционирует* к оптимальному состоянию, изменяя свой состав и параметры входящих в него объектов.

Способ построения множества объектов - решений $x^* \in \boldsymbol{X}^*$, то есть способ реализации процесса оптимизации определяется *эволюционным алгоритмом*.

Эволюционные вычисления составляют класс методов оптимизации, эффективных для слабо формализованных задач оптимизации, а также для задач, оптимизируемые объекты в которых не описываются числовыми величинами, например, данные в базах данных.

К эволюционным вычислениям принято относить:

- *генетические алгоритмы,*
- *генетическое программирование,*
- *эволюционное программирование,*
- *эволюционные стратегии.*

Хотя перечисленные походы близки, каждый из них реализует эволюционный алгоритм своим собственным способом. Различия касаются всех аспектов эволюционного алгоритма, включая выбор способа представления индивидуума, используемый механизм отбора, различные формы генетических операторов и измерения пригодности.

2.2. Генетические алгоритмы

Классический генетический алгоритм выглядит следующим образом.

НАЧАЛО /* генетический алгоритм */

Создать начальную популяцию

Оценить приспособленность каждой особи

останов := FALSE

ПОКА НЕ останов ВЫПОЛНЯТЬ

НАЧАЛО /* создать популяцию нового поколения */

ПОВТОРИТЬ (размер_популяции/2) РАЗ

НАЧАЛО /* цикл воспроизводства */

Выбрать две особи с высокой приспособленностью из предыдущего поколения для скрещивания

Скрестить выбранные особи и получить двух потомков

Оценить приспособленности потомков

Поместить потомков в новое поколение

КОНЕЦ

ЕСЛИ популяция сошлась ТО останов := TRUE

КОНЕЦ

КОНЕЦ

В алгоритме применяются следующие понятия.

Популяция - множество (конечное) объектов, природа которых может быть самой разнообразной. Этим обусловлено широкое применение генетических алгоритмов во многих областях.

Популяция состоит из *особей* или *индивидуумов* - элементов множества. С каждой особью связана *функция приспособленности* или *пригодности* по другой терминологии. Эта функция должна иметь численные значения,

различающиеся от особи к особи. Относительно аналитических и иных свойств функции приспособленности первоначальных предположений не делается.

Процесссскрещивания и получения потомства включает операции, называемые *рекомбинацией* и *мутацией*. Они изменяют *генетический код* особей, в качестве которого может применяться обычный двоичный позиционный код. В простейшем случае численное значение функции приспособленности, записанное в двоичной системе счисления, составляет генетический код особи.

Рекомбинация представляет собой обмен генами - фрагментами кода случайной длины.

Мутация - это случайное изменение (инверсия) одного или нескольких разрядов кода. В результате рекомбинаций и мутаций одни особи превращаются в другие с новыми значениями функции приспособленности.

Вся популяция таким образом эволюционирует к состоянию, когда нормированная по числу особей функция приспособленности принимает экстремальное значение и не меняется от поколения к поколению. Тогда говорят, что популяция сходится.

Приведенный здесь достаточно простой алгоритм является основой для множества модификаций. Все модификации получаются изменением параметров алгоритма и его генетических операторов – отбора, рекомбинации и мутации.

2.2.1. Параметры и работа генетического алгоритма

Дадим формальные определения терминам и понятиям, используемым в генетическом алгоритме.

Генетический алгоритм манипулирует решениями задачи оптимизации. Эти решения не обязательно заранее известны и могут генерироваться в процессе работы алгоритма. Поскольку природа объектов – решений произвольна, их следует трактовать как множество, из свойств которого пока фиксируем лишь конечность. Таким образом, решения задачи оптимизации образуют конечное множество X.

В литературе множество X часто называют *пространством поиска*, что не совсем корректно, поскольку, строго говоря, объекты, принадлежащие X,

могут не образовывать пространства. Тем не менее, следуя традиции, и с учетом данного замечания мы будем также называть множество X пространством поиска.

Подмножество $P \subset X$ из r элементов назовем *популяцией*.

Например, в качестве пространства поиска может выступать множество бинарных строк фиксированной длины l. Мощность такого множества равна 2^l . В этом пространстве конкретные наборы из r строк, генерируемые алгоритмом, образуют популяции.

В принципе, размер популяции не обязательно ограничивается множеством X. Объекты из X могут несколько раз встречаться в популяции - это зависит от способа формирования популяции и конкретного варианта применяемого генетического алгоритма.

Кодирование элементов популяции

Важной составляющей частью всякого генетического алгоритма является кодирование элементов популяции. К кодированию относятся следующие термины.

Алфавит $A = \{a_i, i = 1, 2, \ldots, d\}$ – конечный набор символов. Часто для кодирования применяют бинарный алфавит $A_b = \{0, 1\}$.

На практике обычно применяют единственный алфавит. Теоретически возможно существование нескольких алфавитов A_j для кодирования популяции.

Хромосома – это l – местный кортеж, элементами которого являются символы из алфавитов A_j.

Ген – элемент хромосомы; символ a_i из алфавита A_j, стоящий на i – м месте в хромосоме.

Хромосомы образуют популяцию P как отношение вида:

$$P \subset A_1 \times A_2 \times \ldots \times A_l,$$

в котором не обязательно все алфавиты различны. При бинарном кодировании хромосомы принадлежат отношению

$$P_b \subset \underbrace{A_b \times A_b \times \ldots \times A_b}_{l}$$

При этом количество хромосом равно $n = 2^l$.

При бинарном позиционном кодировании биты строки имеют неодинаковый вес, и мутация в старших разрядах приводит к большим скачкам в пространстве поиска по сравнению с мутацией в младших разрядах. Нельзя определенно сказать, как эта особенность влияет на качество работы алгоритма, однако, в ряде случаев применяется непозиционный двоичный код Грэя.

Далее наряду с термином *хромосома* будут употребляться также термины *особь* и *индивидуум*.

Особь – это элемент популяции, конкретный экземпляр решения. Хромосома– это особь, закодированная с применением алфавита *A*.

Алгоритм стартует, формируя первоначальное множество потенциальных решений (гипотез) P_0, которое образует начальную популяцию. В большинстве случаев это множество формируется случайно. Однако, для увеличения скорости сходимости алгоритма в начальную популяцию, могут включаться решения, полученные с помощью другого оптимизационного метода.

Размер начальной популяции определяется экспериментально и является одним из параметров алгоритма. Очевидно, что число потенциальных решений должно превышать одно.

Функция пригодности

Функция пригодностиf есть отображение пространства поиска *X* на пространство R^+ положительных действительных чисел: $f : X \to R^+$. Подчеркнем, что эта функция определена на всем множестве *X*.

Существует несколько подходов к вычислению функции пригодности.

Функция пригодности $f(c_i)$ определена для каждой хромосомыc_i, i= 1, 2, … , r. Это необходимо для выполнения отбора с использованием ряда методов отбора.

Нормализованная пригодность каждой хромосомы представляет собой отношение:

$$\hat{f}\ (c_i) = \frac{f\ (c_i)}{\sum_{i=1}^{r} f\ (c_i)}$$

Так как нормализованная пригодность всегда принимает значения из интервала [0,1], то она рассматривается как вероятность и ее можно непосредственно использовать для отбора хромосом. Однако данная интерпретация верна только в случае использования конкретной процедуры отбора – *отбора с замещением*. Этот тип отбора будет рассмотрен позже.

В генетических алгоритмах используется также понятие *относительной пригодности*, которая рассматривается как пригодность особи, нормализованная относительно средней пригодности популяции:

$$\hat{f}_r\ (c_i) = \frac{f\ (c_i)}{\sum_{i=1}^{r} f\ (c_i)/r}$$

Относительная пригодность характеризует, насколько лучше или хуже пригодность конкретной особи текущей популяции по сравнению со средним значением пригодности.

Очередная популяция (поколение) P_{t+1} генерируется из предыдущей популяции P_t действием операторов отбора, мутации и рекомбинации, образующих набор генетических операторов $\boldsymbol{\Omega}$.

Отбор

Отбор представляет собой процесс выбора особей, которые будут участвовать в воспроизводстве потомков, пропорционально их значению пригодности. Такой отбор называется *пропорциональным*.

Способ отбора непосредственно связан с природой решаемой задачи оптимизации. Очевидно, всегда имеется возможность построить специальный способ отбора, ориентированный на задачу.

Помимо пропорционального отбора, характерного для классической модели генетического алгоритма, существуют и другие способы стохастического отбора. Рассмотрим некоторые из них, наиболее часто применяемые на практике.

Универсальный случайный отбор. Представим, что популяция расположена в случайном порядке на круговой диаграмме, где каждому

индивидууму соответствует пространство, пропорциональное его пригодности. Вокруг диаграммы размещено колесо рулетки с *N* стрелками, находящимися на равном расстоянии друг от друга. Одно вращение рулетки теперь одновременно выберет все *N* членов промежуточной популяции (то есть *N*/2 пар, которые дадут потомство). Результирующий выбор при этом является случайным.

Турнирный отбор. Работает следующим образом: из популяции случайным образом выбирается некоторое количество *t* индивидуумов, и лучший из этой группы заносится в промежуточную популяцию. Данная процедура повторяется *N* раз. Часто турниры проводятся только между двумя индивидуумами, но в общем случае можно создавать для этой цели произвольную группу из *t* кандидатов.

Отбор с отсечением с порогом *T* . Определяются *T* лучших индивидуумов популяции, из них случайным образом выбирается один представитель. Эта операция выполняется *N* раз, в результате чего формируется промежуточная популяция. При этом все *T* индивидуумов имеют одинаковую вероятность выбора, а (*N*-*T*) членов популяции «отсекаются», то есть не участвуют в выборе.

Генетические операторы

Постоянный отбор особей из одной и той же популяции приведет в лучшем случае лишь к появлению множества одинаковых особей в популяции. Для того, чтобы обеспечить возможность развития и улучшения популяции необходимо между этапами отбора вводить некое разнообразие в популяцию.

Генетические операторы позволяют изменять особей популяции путем модификации их генотипа. К генетическим операторам относятся *мутация*и*рекомбинация.*

Мутация

Мутация представляет собой случайное изменение значений генов. Случайно выбирается точка мутации (как правило, одна в хромосоме) и мутация выполняется с заданной вероятностью. Рис. 0 иллюстрирует мутацию хромосомы с бинарным кодированием.Мутация изменяет генотип особей согласно, в общем случае, нескольким вероятностным правилам. Обычно только часть хромосомы изменяется в процессе мутации, что позволяет потомку наследовать большую часть характерных признаков родителей.

Рис. 0. Пример мутации

Роль мутации такова, что она позволяет алгоритму выполнить своеобразный “прыжок” в пространстве поиска и оказаться в области, где, возможно, и находится глобальный экстремум. Таким образом алгоритм может уходить из областей локальных экстремумов.

Очевидно, что в примере поиска экстремума функции действительного переменного с бинарным позиционным кодированием величина “прыжка” определяется позицией мутирующего гена: чем старше разряд, тем дальше выполняется “прыжок”.

Оператор мутации является простым, но весьма важным генетическим оператором. Эволюция популяции только лишь с помощью мутации и отбора не только возможна, но иногда и очень эффективна. Например, эволюционные стратегии первоначально были разработаны как алгоритмы, использующие лишь процессы мутации и отбора.

Возможно также использование не вероятностной, а детерминированной мутации, более известной как *восстановление*.

Рекомбинация

Рекомбинация или скрещиваниеосуществляет обмен генетической информацией между особями популяции путем обмена участками хромосом.

В русскоязычной литературе употребляется также термин *кроссовер*, что представляет собой просто транслитерацию английского слова *crossover*– *скрещивание*.

Мы будем называть операцию скрещивания *рекомбинацией*, а словом *кроссовер* обозначать *оператор* скрещивания.

Существует несколько видов операторов скрещивания или кроссоверов.

Простейшим является одноточечный кроссовер, который иллюстрирует **Рис.**2. Одноточечный кроссовер работает следующим образом. Сначала

случайным образом выбирается одна из точек разрыва, то есть участок между соседними битами в строке. Обе родительские хромосомы разрываются на два сегмента по этой точке. Затем соответствующие сегменты различных родителей склеиваются и получаются два генотипа потомков.

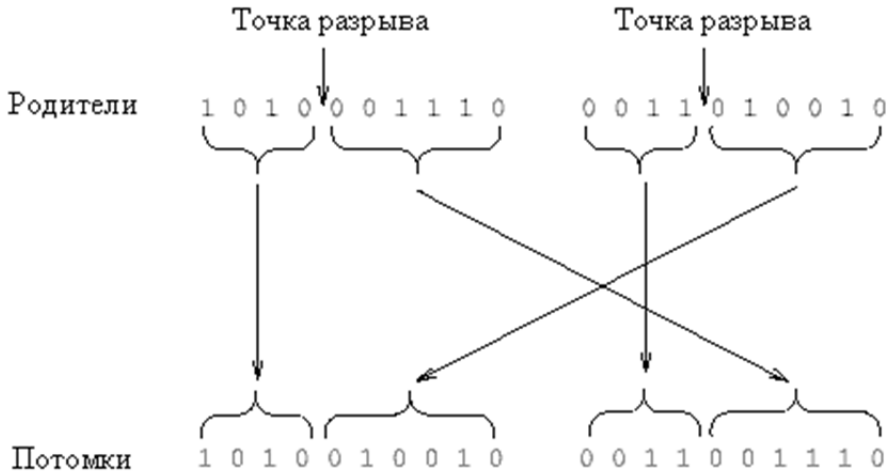

Рис. 2. Одноточечный кроссовер.

Кроме одноточечного существуют различные варианты операторов рекомбинации с двумя и более точками разрыва.

Многоточечный кроссовер. Работает многоточечныйкроссовер следующим образом: случайным образом выбирается p точек разрыва, которые делят строку-родитель на p+1 части. Первый потомок пары наследует от первого родителя все части с нечетными номерами, а от второго – части с четными номерами. Второй потомок, соответственно, наоборот.

Однородный кроссовер. Происходит независимый обмен каждым битом с заданной вероятностью обмена.

Кроссовер по маске. Обобщением многоточечного и однородного кроссовера является *кроссовер по маске*. Маска – это строка длины l-1 (то есть на единицу меньше длины хромосомы), которая состоит из нулей и единиц. Номера позиций маски, соответствующих значению 1, являются номерами точек кроссовера. Например, маска 00100100 определяет двухточечныйкроссовер с точками разрыва после 3-го и 6-го генов хромосомы.

Далее мы будем применять именно этот вариант кроссовера как наиболее общий.

Существуют и более экзотические кроссоверы, из которых упомянем следующие.

Кроссовер с перемещением. Каждая из выбранных для скрещивания хромосом заменяется на новую. Образование новой хромосомы осуществляется путем изменения позиций генов первоначальной хромосомы. Изменение позиций генов производится произвольно и независимо для каждой хромосомы. Новые хромосомы подвергаются одноточечномукроссоверу. Затем в каждой из полученных хромосом происходит возврат первоначальных позиций генов.

Кроссовер частей хромосом .Сначала в хромосомах определяются биты, не являющиеся идентичными по значению и позиции, а затем один из типов кроссовера применяется к определенной таким образом строке битов. Это гарантирует появление потомков, отличных от родительских хромосом.

2.3. Некоторые свойства генетических алгоритмов

Различные способы отбора и разновидности генетических операторов служат основой для конструирования многих вариантов генетических алгоритмов. Однако даже классический генетический алгоритм является сложной вычислительной процедурой, настройка которой на конкретную задачу требует иногда специального исследования.

К настоящему времени исследованы ряд свойств генетических алгоритмов, оказывающих существенное влияние на их работу. Рассмотрим некоторые из них, представляющие интерес в контексте данной работы.

2.3.1. Инвариантные шаблоны и строящие блоки

Для успешной работы генетического алгоритма он должен иметь возможность сохранять хорошие решения, с тем, чтобы дальше улучшать именно их, приближаясь к экстремуму. Данное, неформальное утверждение, справедливо, в общем, для любого метода оптимизации, работающего пошагово, с условием, что оптимизируемая функция удовлетворяет определенным условиям, например, условиям Липшица. Альтернативой является только полный перебор вариантов.

В генетических алгоритмах применяются биологические понятия *генотип* и *фенотип*.

Генотип описывает устройство хромосом – способ кодирования и параметры.

Фенотип – это свойства особи или признаки, определяемые конкретным генотипом.

Если популяция особей из поколения в поколение сохраняет некоторые свои свойства или фенотип, то она, соответственно, сохраняет и генотип.

Данный биологический принцип воплощен в генетических алгоритмах и реализуется посредством таких объектов, как *инвариантный шаблон* и *строящий блок*.

Инвариантный шаблон или просто шаблон – это часть хромосомы, имеющая конкретный генотип. Шаблон определяет все возможные хромосомы с данным генотипом.

Для обозначения шаблонов применяют алфавиты вида {*A*, *}, где *A* – некоторый алфавит, а знак * соответствует любому символу из алфавита *A*.

Например, для хромосом с бинарным алфавитом шаблон 10**1, определяет собой множество из четырех строк {10001; 10011; 10101; 10111}.

В русскоязычной литературе можно встретить термин “ *шима*” вместо *“шаблон”*. Здесь вновь мы имеем пример простой транслитерации английского слова *schema – схема*. Термин *схема* на наш взгляд более подходит для всего алгоритма и мы будем его использовать далее.

У шаблонов выделяют два параметра – порядок и определенную длину.

Порядок шаблона – это число определенных битов ("0" или "1") в шаблоне.

Определяющая длинашаблона – расстояние между крайними определенными битами в шаблоне.

Например, шаблон “*0*1” имеет порядок 2 и определяющую длину 3. Каждая хромосома в популяции является примером из множества 2^l шаблонов.

Пригодность шаблона определяется как средняя пригодность хромосом, которые его содержат.

Строящий блок – это шаблон, обладающий высокой пригодностью, низким порядком, короткой определенной длиной.

Одним из первых объяснений работы генетических алгоритмов была теорема шаблонов Голланда. Теорема шаблонов описывает механизм сохранения генотипа при эволюции популяции следующим образом.

После процедуры отбора в популяции остаются только строки с более высокой пригодностью. Следовательно, строки, которые являются примерами шаблонов с высокой пригодностью, выбираются чаще. Рекомбинация реже разрушает шаблоны с более короткой определенной длиной, а мутация реже разрушает шаблоны с низким порядком. Поэтому такие шаблоны имеют больше шансов переходить из поколения в поколение.

Теорема Голланда.

Пусть r – размер популяции, h – шаблон, $c(h, t)$ - число хромосом в популяции, на шаге t эволюции имеющих в своем генотипе шаблон h.

Нормируем $c(h, t)$ относительно размера популяции: $p(h, t) = c(h, t)/ r$. Величину $p(h, t)$ можно трактовать как вероятность появления шаблона h в хромосомах популяции.

Согласно теореме шаблонов прогноз появления шаблона h в хромосомах в следующем поколении определяется следующим неравенством:

$$p(h,\ t+1)\ \geq p(h,\ t)\frac{f(h,\ t)}{\hat{f}}\ [1\text{-}p_c\ \frac{\Delta(h)}{l\text{-}1}(1\text{-}p(h,\ t)\frac{f(h,\ t)}{\hat{f}})](1\text{-}p_m)^{O(h)}$$

Здесь $f(h, t)$ – средняя по популяции пригодность хромосом с шаблоном h, $\hat{f}$ - средняя пригодность хромосом всей популяции $\Delta(h)$ -определяющая длина шаблона при одноточечной рекомбинации, p_c - вероятность такой рекомбинации, p_m - вероятность мутации, $O(h)$ - порядок шаблона h.

В правой части неравенства содержится нижняя оценка для вероятности $p(h, t+1)$, соответствующая так называемому *пессимистическому прогнозу*, когда рекомбинация и мутация разрушают максимально возможное число шаблонов.

На самом деле эволюция шаблонов происходит более сложным образом. Разрушенные шаблоны могут восстанавливаться из других хромосом. По мере эволюции, популяция становится все более и более однородной и разрушение шаблона в одном месте компенсируется его сохранением в другом.

Потерянные шаблоны могут появляться вновь случайным образом, а теорема шаблонов не предсказывает это явление.

Наконец, величины $f(h, t)$ и $\hat{f}$ не остаются постоянными от поколения к поколению, хотя их вычисляемые на каждом шаге значения можно использовать в неравенстве.

Поэтому теорема шаблонов весьма упрощенно описывает поведение генетического алгоритма. Тем не менее, на протяжении долгого времени она остается одним из основных аналитических результатов в теории генетических алгоритмов.

Важным фактом, который следует из теоремы шаблонов, и подтвержден экспериментально, является сильное отрицательное влияние мутации на шаблоны высокого порядка. В самом деле, как следует из теоремы шаблонов, достаточно большие значения вероятности мутации и порядков $O(h)$ существенно уменьшают величину в правой части неравенства.

Отсюда следует требование к строящему блоку – быть шаблоном низкого порядка.

На теореме шаблонов основана *гипотеза строящих блоков*, предложенная Гольдбергом: производительность генетического алгоритма определяется его способностью генерировать и сохранять строящие блоки. Для этого необходимо выполнение двух условий:

- гены должны слабо взаимодействовать в хромосоме;
- гены, образующие строящий блок, должны быть близко расположены друг к другу.

Взаимодействие или влияние генов в хромосоме, называемое также *эпистазис*, заключается в том, что вклад в величину функции пригодности одного гена зависит от величины других генов.

На практике исключить влияние генов нельзя, особенно при решении задач оптимизации с мультимодальными функциями пригодности.

Тем не менее, гипотеза строящих блоков в общем ее смысле верна: она определяет механизм сохранения генотипа и фенотипа в генетическом алгоритме. Таким механизмом является эволюция строящих блоков.

Привлекая положения, высказанные в начале данного раздела, делаем вывод, что генетический алгоритм обладает необходимым свойством сохранять хорошие решения, что делает его отличным от полностью случайного алгоритма полного перебора решений.

2.3.2. Неявный параллелизм

Обрабатывая в каждом поколении явно n строк, генетический алгоритм неявно обрабатывает порядка n^3 шаблонов низкого порядка и с высокой пригодностью. Это явление названо *неявным параллелизмом*.

В ряде работ приведены геометрические интерпретации явления неявного параллелизма, которые мы и используем в данном разделе.

Пусть необходимо найти экстремум функции $f(x) = 10 + x\sin(x)$, определенной на отрезке $X = [0, 10]$. Пусть кодирование будет осуществляться бинарными строками длины 3, следовательно, отрезок [0,10] нужно разбить на $2^3 = 8$ подинтервалов, каждому из которых будет соответствовать уникальная двоичная комбинация, получаемая переводом номера подинтервала, считая слева направо, в двоичную систему. Длина каждого такого интервала будет $h = 10:8 = 1.25$. **Рис.** 3 иллюстрирует данное кодирование.

Рис. 3. Разбиение отрезка [0, 10] в соответствии с принятым кодированием.

Каждый шаблон определяет множество бинарных строк, имеющих в соответствующих позициях либо 0, либо 1, в зависимости от того, какой бит находится в соответствующей позиции самого шаблона.

В пространстве X шаблону будет соответствовать объединение подинтервалов, бинарные представления которых являются примерами этого шаблона.

Например, в рассматриваемой здесь задаче шаблонам h = (00*) и h = (*10) на отрезке [0,10] будут соответствовать следующие области, которые иллюстрирует **Рис.**4 (на рисунке они выделены темным цветом).

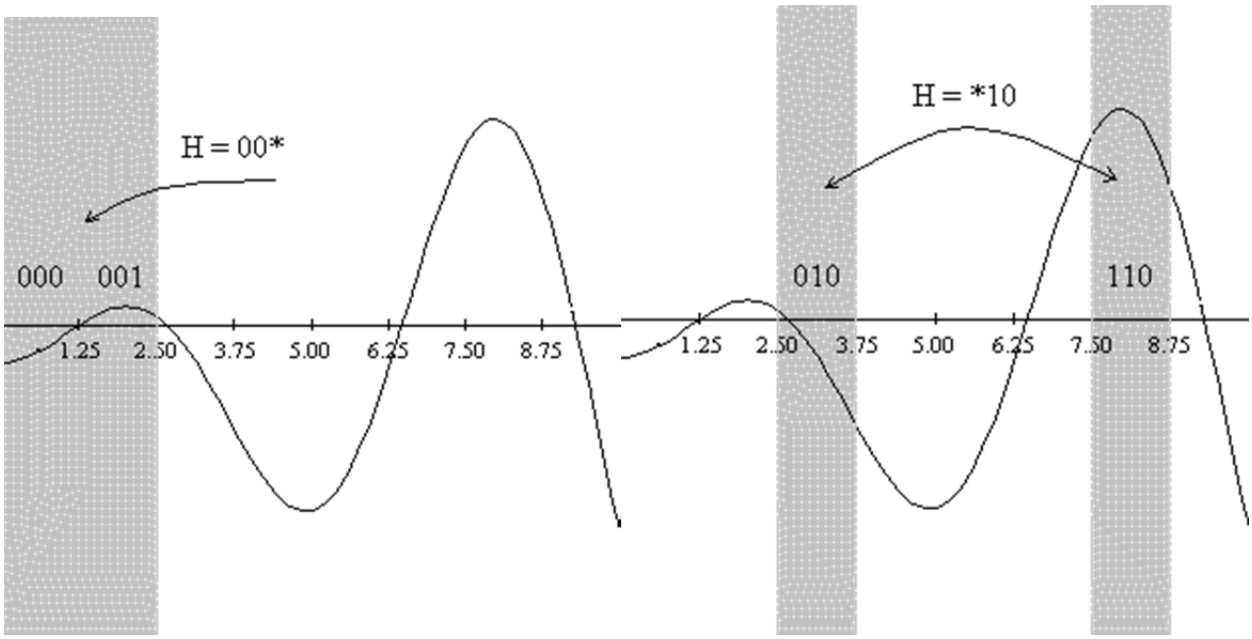

Рис. 4.Интерпретация шаблонов порядка 2 в пространстве X = [0, 10].

Шаблоны с меньшим порядком будут задавать более многочисленное множество бинарных строк, поэтому в пространстве X = [0, 10] они смогут охватить большую область. Рис. 5 иллюстрирует как будут представлены шаблоны порядка 1 в рассматриваемом примере.

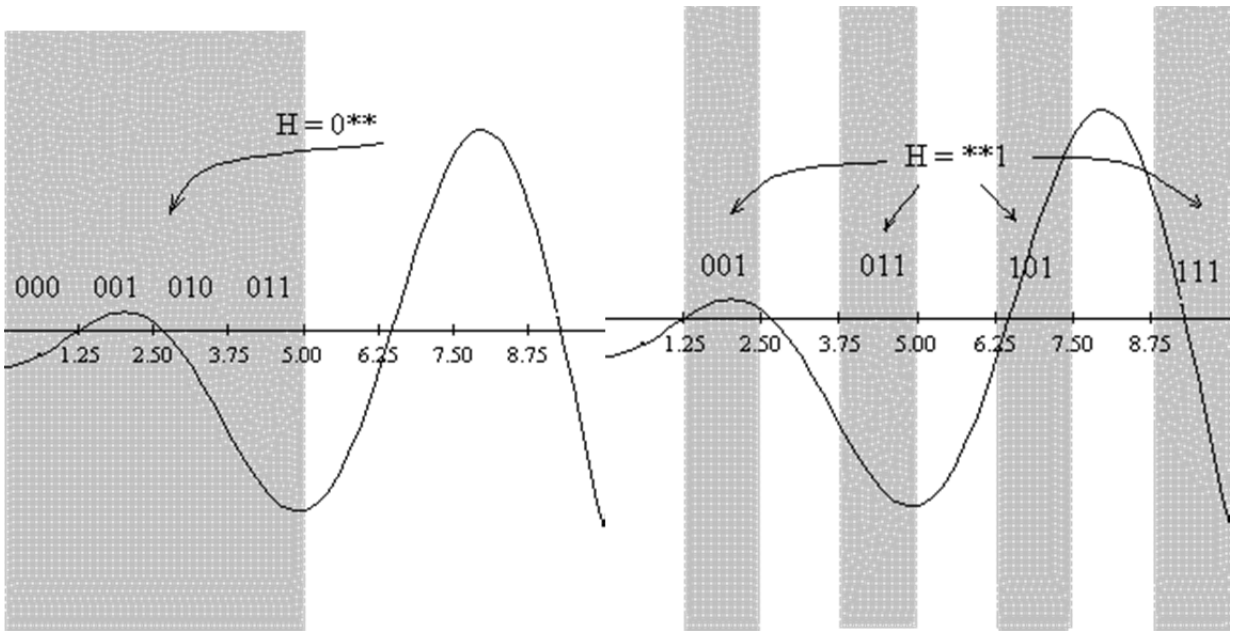

Рис. 5. Интерпретация шаблонов порядка 1 в пространстве *X* = [0, 10].

Неявный параллелизм, как следует из рисунков, заключается в обработке в пространстве *X параллельно* нескольких интервалов, соответствующих шаблонам. Чем меньше порядок шаблона, тем больше таких интервалов и тем шире пространство поиска охвачено алгоритмом.

3. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ РАБОТЫ

Для моделирования генетических алгоритмов используется система *Mathematica*. В системе имеется ряд функций, предназначенных для решения задач оптимизации, например функции FindMaximum, FindMaxValue и т.п. Эти функции можно использовать для сравнения результатов работы классических алгоритмов оптимизации и генетического алгоритма. Классический генетический алгоритм моделирует программа GA_Mma_v9.nb.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ.

1. Получить тестовые функции, применяемые для исследования генетических алгоритмов.

2. Модифицировать программу GA_Mma_v9.nb для заданной тестовой функции.
3. Выполнить решение задачи поиска экстремума тестовой функции генетическим алгоритмом и при помощи соответствующей функции системы *Mathematica*. Сравнить полученные решения.
4. Исследовать влияние параметров генетического алгоритма на результаты поиска экстремума тестовой функции:
   - размера популяции решений;
   - величины мутации;
   - длины строки хромосомы;
   - точности вычисления функции пригодности.
5. Меняя величины параметров в п. 4.4, сравнивать решения, получаемые при помощи генетического алгоритма и функций системы *Mathematica* согласно п. 4.3.
6. Оформить отчет, содержащий описание и результаты экспериментов по исследованию генетического алгоритма. Отчет оформить в системе *Mathematica*, используя ее возможности по созданию гипертекстовых документов.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ И УПРАЖНЕНИЯ.

1. Укажите признаки эволюционных вычислений и генетических алгоритмов, позволяющие их классифицировать среди других алгоритмов оптимизации.

2. Обоснуйте необходимость кодового представления решений в генетическом алгоритме.

3. Как в программе GA_Mma_v9.nb моделировать поиск экстремума функции n переменных для $n > 2$?

4. Какие алгоритмы применяются в функциях системы *Mathematica* FindMaximum, FindMaxValue ?

5. Можно ли использовать неявный параллелизм генетического алгоритма при исследовании функции многих переменных?

6. СПИСОК ЛИТЕРАТУРЫ

1. Богатырёв М.Ю. Генетические алгоритмы: принципы работы, моделирование, применение – Тула, ТулГУ, 2003. – 152 с.
2. Электронный ресурс: http://ru.wikipedia.org/wiki/генетичекийалгоритм.
3. Электронный ресурс: http://mathmod.aspu.ru/images/File/ebooks/GAfinal.pdf