В.И. Аверченков, П.В. Казаков

ЭВОЛЮЦИОННОЕ МОДЕЛИРОВАНИЕ И ЕГО ПРИМЕНЕНИЕ

Монография

3-е издание, стереотипное

Москва
Издательство «ФЛИНТА»
2016

УДК 004.89+004.021
А19

Аверченков В.И.

А19 Эволюционное моделирование и его применение: монография [электронный ресурс] / В.И. Аверченков, П.В. Казаков. — 3-е изд., стереотип. — М. : ФЛИНТА, 2016. — 200 с.

ISBN 978-5-9765-1264-1

Рассматриваются принципы и методы эволюционного моделирования. Особое внимание уделяется главному методу эволюционного моделирования – генетическому алгоритму. Приводятся конкретные примеры его применения к решению различных задач оптимизации.

Монография предназначена для специалистов в области информационных технологий, а также для студентов, магистрантов и аспирантов, обучающихся по направлениям «Информатика и вычислительная техника», «Информационные системы».

УДК 004.89+004.021

ISBN 978-5-9765-1264-1

ПРЕДИСЛОВИЕ

В настоящее время неотъемлемой составляющей переднего края исследований научного направления «Искусственный интеллект» являются так называемые «мягкие вычисления» (soft computing), объединяющие нечеткие системы, искусственные нейронные сети и эволюционные алгоритмы. Большое количество русскоязычных публикаций, посвященных первым двум разделам мягких вычислений, свидетельствует о неослабляющем интересе к ним, подтверждает большие перспективность их исследований и эффективность применения при решении соответствующего класса задач. Что касается эволюционных алгоритмов, то объем исследований, связанных с ними, выглядит в сравнении гораздо скромнее. В то же время эта неотъемлемая составляющая мягких вычислений образует целое фундаментальное научное направление, известное как эволюционное моделирование. Оно находит применение при решении различных классов задач, включающих оптимизацию, моделирование адаптивного поведения, моделирование эволюционных процессов в социальных, экономических и технических системах, которым наиболее свойственны основные принципы эволюции. Заимствование законов развития природы позволяет подойти к решению этих задач в более широком, универсальном смысле без ограничений ранее разработанных математических моделей их описания.

Зарубежные исследования в области эволюционного моделирования объединены в направление, которое называется эволюционные вычисления (evolutionary computation). Научными и прикладными разработками в этой области занимаются во многих научных школах стран Европы, Америки и Азии, а также научно-исследовательских центрах крупных компаний. Что отражает существенно большее, чем в Российской Федерации число публикаций и монографий по данной области искусственного интеллекта.

В монографии предпринята попытка рассмотреть основные вопросы, связанные с особенностями эволюционного моделирования, применением его методов при решении задач, где их использование наиболее эффективно. Монография состоит из трех глав, ее содержание основывается на исследованиях в области эволюционного

моделирования ведущих научных школ РФ, публикациях в иностранных изданиях, а также некотором опыте авторов в рассматриваемой области знаний.

Первая глава посвящена анализу возможностей эволюционного моделирования как самостоятельной методологии оптимизации. Рассматривается постановка задачи оптимизации и приводятся традиционные методы ее решения. Проводится аналогия между процессом оптимизации и эволюционным моделированием. Акцентируется внимание на ограниченность традиционных методов оптимизации, существенную их зависимость от математической модели задачи. Отмечаются возможности методов эволюционного моделирования, позволяющие им решать оптимизационные задачи с произвольными математическими моделями. Особое внимание уделяется понятиям эволюционного моделирования, заимствованным из существующих концепций естественной эволюции. В связи с этим приводятся правила интерпретации соответствующих терминов для эволюционной и математической моделей. Основные принципы эволюции - наследственность, изменчивость и естественный отбор - моделируются средствами эволюционных алгоритмов, основы работы которых также приводятся в этой главе. Прикладные свойства эволюционного моделирования формируются вокруг компьютерной имитации по определенным законам коллективного поведения объектов и охватывают не только оптимизацию, но и моделирование самоорганизации, адаптивного поведения, «искусственной жизни» и эволюционное проектирование. Кратко особенности этих задач и перспективы их решения с помощью эволюционных алгоритмов отмечаются в конце главы.

Вторая глава посвящена описанию эволюционных алгоритмов. Основное внимание уделяется главному методу эволюционного моделирования – генетическому алгоритму. В нем наиболее полно реализованы механизмы естественных эволюционных процессов, включающих селекцию, воспроизводство и наследование. Компьютерная реализация этого позволяет применять генетический алгоритм для любых задач эволюционного моделирования. Практически это может быть сделано при использовании стандартного генетического алгоритма, его существующих модификаций либо посредством разработки специальной версии генетической алгоритма под решаемую задачу. Рассматриваются способы представления информации генетического алгоритма, его операторы, некоторые модификации, а также

особенности настройки и мониторинга функционирования. Глава содержит математический вывод и описание фундаментальной теоремы генетического алгоритма - математическое обоснование его эффективности как оптимизационного метода. Представлены особенности и принципы реализации других эволюционных алгоритмов – генетического программирования, эволюционных стратегий и эволюционного программирования. В конце главы приводится и описывается структура обобщенного эволюционного алгоритма.

Третья глава посвящена применению эволюционного моделирования для решения практических задач оптимизации. Рассматриваются такие задачи, как оптимизация сложной нелинейной функции многих переменных, параметрический синтез технических объектов, оптимизация комбинаторных математических моделей в задачах коммивояжера и оптимального инвестиционного планирования. При решении каждой задачи важно правильно преобразовать исходную математическую модель в эволюционную модель, а также выбрать или создать генетический алгоритм. Оптимальность найденного решения во многом определяется настройкой управляющих параметров генетического алгоритма, что достигается серией его тестовых запусков. Примеры подобных исследований также приводятся в этой главе. Кроме отмеченных, рассматривается задача моделирования адаптивного поведения, в ее решении генетический алгоритм демонстрирует возможности самоорганизации популяции при динамическом изменении условий оптимизации без перезапуска самого алгоритма. Применение эволюционного моделирования при решении практических задач предполагает использование соответствующего программного обеспечения. При этом наиболее эффективным считается программирование эволюционного алгоритма непосредственно под решаемую задачу. Однако существуют и специализированные среды для автоматизации создания и исследования эволюционных моделей. Глава кратко знакомит с такими системами и особенностями их применения.

Монография содержит приложения, включающие пример объектно-ориентированной реализации генетического алгоритма, позволяющий создавать на его основе необходимые модификации или интегрировать в существующие информационные системы, а также определения часто употребляемых понятий эволюционного моделирования.

ОСНОВНЫЕ УПОТРЕБЛЯЕМЫЕ ОБОЗНАЧЕНИЯ

$x = \{x_1, x_2, ..., x_n\}$ - переменные оптимизации математической модели;

$f(x)$ – целевая функция;

$\varphi_i(x_1, ..., x_n)$ - ограничения на параметры математической модели;

x_j^-, x_j^+ - нижняя и верхняя границы области определения переменной;

$f(x^*), f^*$ – значение целевой функции в точке экстремума;

$R(x)$ – функция штрафа;

$C_i = \{g_1, g_2, ..., g_n\}$ - хромосома с номером i, описывающая одно решение и состоящая из n числа генов;

$g_j^{(i)}$ - ген номер j хромосомы C_i;

$f(C_i)$ – fitness-функция;

$C_i(t)$ – хромосома C_i поколения t;

$L(C_i)$ – число бит в хромосоме C_i;

$L(g_j)$ - разрядность гена с номером j;

Np – размер популяции;

Ng –число поколений;

Nf – число вычислений функции;

$Np(t)$ – размер популяции в поколении t;

Nt – число элементов в группе для турнирной схемы отбора;

Pm – вероятность применения оператора мутации;

Pc – вероятность применения оператора кроссинговера;

$Pinv$ – вероятность применения оператора инверсии;

k – сайт кроссинговера;

K – фрагмент длины K хромосомы;

$G(t)$ – поколение номер t;

C^* - хромосома, соответствующая оптимальному решению;

(C^*, t) – хромосома, соответствующая лучшему решению в поколении t;

$\tilde{f}(C_i)$ - нормированное значение fitness-функции;

$\overline{f}(C_i)$ - относительная оптимальность хромосомы C_i;

$\overline{f}(t)$ - средняя оптимальность поколения $G(t)$;

$\overline{f}(H(t))$ - средняя оптимальность хромосом, содержащих шаблон $H(t)$;

H – шаблон хромосомы;

$\delta(H)$ – порядок шаблона;

$L(H)$ – длина шаблона;

$P(H,d)$ – вероятность уничтожения шаблона;

$P(H,s)$ – вероятность сохранения шаблона;

$Pm(H,s)$ – вероятность сохранения шаблона после оператора мутации;

$N(H(t))$ – число хромосом популяции $G(t)$, содержащих шаблон H;

Nq – число запусков генетического алгоритма, когда он нашел оптимальное решение;

V – скорость сходимости генетического алгоритма;

η - эффективность генетического алгоритма.

ВВЕДЕНИЕ

> В природе не происходит ничего, в чем не был бы виден смысл какого–нибудь максимума или минимума.
>
> *Леонард Эйлер*

Процессы, протекающие в живой природе, постоянно привлекают исследователей из самых различных областей знания, благодаря тем завидным результатам, которые удается достигать природе в процессе своего существования. Искусственное моделирование удивительной по своей эффективности способности биологических систем адаптироваться с целью наилучшего приспособления и выживания в окружающих условиях может позволить по-новому взглянуть на решение проблем в области философии, психологии, управления, социальных, экономических и технических наук. Разработкой теоретических основ, методов моделирования эволюционных процессов, а также средств их компьютеризации занимается такое направление в области искусственного интеллекта как эволюционное моделирование (ЭМ).

Эволюционное моделирование можно определить как направление в искусственном интеллекте, в основе которого лежат принципы и понятийный аппарат, заимствованные из популяционной генетики и объединяющее компьютерные методы (генетические алгоритмы, генетическое программирование, эволюционное программирование и эволюционные стратегии) моделирования естественных эволюционных процессов. Происходящие в естественных системах эволюционные преобразования направлены на достижение аналогичной цели, что и в искусственных системах при решении оптимизационных задач. Только в последнем случае критерием «приспособленности» решения является его качество, оптимальность относительно других исследованных точек поискового пространства. Заимствование остальных ключевых свойств эволюции – наследственности, изменчивости и естественного отбора - выделило ЭМ в отдельную методологию решения оптимизационных задач. В том числе и таких, которые невозможно решить точными математическими методами из-за

неполноты исходных данных, наличия ограничений и высокой размерности, динамично изменяющихся внешних условий и необходимости функционирования в конкурентной среде.

Основоположниками развития эволюционного моделирования как самостоятельного направления в области искусственного интеллекта принято считать Дж. Холланда, Л. Фогеля, А. Овена, М. Уолша, И. Букатову, Л. Расстригина и других исследователей. Их работы связаны с попытками воспроизвести с помощью математических моделей и алгоритмов основные механизмы эволюционных преобразований в естественных системах, характеризующихся постоянным стремлением достижения оптимальных форм существования в сочетании с минимальными затратами ресурсов. Созданная в итоге теория эволюционного моделирования (эволюционных вычислений) стала фундаментом для разработки принципиально новых компьютерных методов оптимизации в том числе задач, которые до этого считались неразрешимыми.

Трудно назвать естественную или искусственную систему, которой бы не были свойственны эволюционные процессы развития. Главным из них является стратегия «выживает сильнейший». Применение методов ЭМ для исследования таких систем позволяет наиболее точно, не прибегая к модификации их математических моделей, определить оптимальные характеристики функционирования этих систем, новые структуры и формы их развития (эволюции). При этом используется главная концепция ЭМ искусственных систем: вместо моделирования системы в уже готовом виде следует заниматься исследованием эволюции образующих ее простых объектов. В итоге наблюдается их самоорганизация в направлении повышения качества свойств системы в целом.

Практические возможности эволюционного моделирования прежде всего рассматриваются на основе применения главного метода эволюционных вычислений – генетического алгоритма. В нем наиболее полно реализованы механизмы естественных эволюционных процессов, включающих селекцию, воспроизводство и наследование. При этом наличие гибких средств регулирования интенсивности этих процессов позволяет управлять процессом поиска решений, гарантируя определение самого лучшего из них.

Отличительной чертой эволюционных алгоритмов от других численных методов является особая стратегия поиска оптимального решения путем моделирования развития некоторой ситуации вместо непосредственного вычисления ответа по детерминированным формульным зависимостям. В итоге применение генетического алгоритма дает преимущество в неопределенных ситуациях, где существует несколько достаточно хороших, хотя и неочевидных решений, а другие методы поиска решений оказываются непригодны или малоэффективны. Помимо этого, такие алгоритмы эффективно формируют шаблоны адаптивного поведения, так как хромосомы, кодирующие «выжившие» в процессе эволюции решения, хранят, по сути, коллективный опыт многих поколений.

Таким образом, эволюционное моделирование может быть эффективно использовано в следующих случаях:

- применение эволюционных алгоритмов для изучения и моделирования отдельных процессов естественной эволюции;
- совершенствование существующих искусственных систем путем наделения их свойствами адаптивного поведения и самоорганизации на основе методов эволюционного моделирования;
- автоматизация решения различных оптимизационных задач науки и техники.

Методы эволюционного моделирования можно рассматривать не только как эффективные средства решения задач оптимизации, самоорганизации и моделирования адаптивного поведения в разных прикладных областях, но и как базу для упражнений в совершенствовании техник программирования отдельных алгоритмов и программных систем. Особое место здесь занимает генетический алгоритм, содержащий все существующие лингвистические конструкции современных языков программирования (включая возможности параллельного программирования) и одновременно являющийся отправной точкой для создания его модификаций – новых генетических алгоритмов для решения специальных прикладных задач. Применение для этого технологии объектно-ориентированного программирования позволяет реализовать это максимально гибко и удобно для программиста при дальнейшем внедрении в информационные системы и сопровождении.

Эволюционное моделирование было и остается одним из перспективных направлений искусственного интеллекта. Становление и развитие ЭМ в России связано с теоретическими и практическими исследованиями ряда ведущих научных школ, география которых охватывает МГТУ им. Н.Э. Баумана (Москва); Институт прикладной математики им. М.В. Келдыша, РАН (Москва); Нижегородский государственный университет (Нижний Новгород); Таганрогский технологический институт Южного федерального университета (Таганрог); Томский политехнический университет (Томск); Сибирский аэрокосмический университет им. М.Ф. Решетнева (Красноярск) и др. Особое место здесь занимает научно-педагогическая школа в области «Теории и принципов разработки интеллектуальных САПР», руководимая заслуженным деятелем науки РФ, доктором технических наук, профессором Курейчиком В.М.(Таганрогский технологический институт Южного федерального университета). Во многом благодаря его работе эволюционное моделирование получило в нашей стране статус фундаментального научного направления, результаты теоретических и практических исследований по которому находят отражение в изданных учебных пособиях и монографиях, публикациях трудов профильных конференций по искусственному интеллекту.

Результаты международных исследований по эволюционным вычислениям также широко представлены в сети Интернет. Среди таких русскоязычных ресурсов наиболее развитым является сайт [12], созданный и поддерживаемый Ю. Цоем (Томский политехнический университет). Представленная на сайте, хорошо структурированная и обновляемая информация по методам эволюционного моделирования существенно упрощает поиск необходимой информации, обеспечивает электронную поддержку развития исследований по эволюционному моделированию в нашей стране.

ГЛАВА 1. ЭВОЛЮЦИОННОЕ МОДЕЛИРОВАНИЕ – НОВАЯ ТЕХНОЛОГИЯ ОПТИМИЗАЦИИ

Как известно, конечной целью любой процедуры оптимизации является отыскание наилучшего, или «оптимального» решения. Однако далеко не всегда, в силу ряда причин, существует реальная возможность осуществить это. Поэтому видится более правильным подразумевать под оптимизацией некое стремление к совершенству в улучшении решения задачи, которое, возможно, так и не будет достигнуто.

В связи с этим всегда большое внимание уделялось поиску и разработке новых методов, алгоритмов оптимизации с помощью ЭВМ. Одним из перспективных и во всех отношениях «новым» здесь является применение технологии эволюционного моделирования, в которой эффективно сочетаются исключительно высокий потенциал к решению оптимизационных задач (или задач, которые могут быть к таковым сведены), а также богатые возможности реализации с помощью средств вычислительной техники и программирования.

1.1. Анализ задачи оптимизации и методов ее решения

Под термином «оптимизация» обычно понимают процесс или последовательность операций, позволяющих получить приемлемое (при отсутствии известных решений), уточненное, улучшенное (при наличии известных решений) решение [44]. В свою очередь, теория оптимизации включает совокупность результатов фундаментальных математических исследований и численных методов, ориентированных на поиск (выбор) наилучших вариантов решений из множества альтернатив, при этом позволяя избежать полного перебора и сравнения возможных вариантов.

1.1.1. Постановка задачи оптимизации

Под решением задачи оптимизации будем понимать процесс поиска значений переменных $x = \{x_1, x_2, ..., x_n\}$, принадлежащих некоторой допустимой области *D* и обеспечивающих оптимальные значения функции *f(x)*. Эта функция показывает относительное «предпочтение» одного варианта решения по отношению к другим и называется критерием оптимальности или целевой функцией (ЦФ). Если имеется только одна переменная, то целевую функцию можно представить кривой на плоскости. Если переменных две, то ЦФ будет изображаться поверхностью в трехмерном пространстве. При трех и более переменных поверхности, задаваемые ЦФ, называются гиперповерхностями и не поддаются изображению обычными средствами. При этом топологические свойства поверхности ЦФ играют важную роль в процессе оптимизации, так как от них зависит выбор наиболее эффективного алгоритма.

В формализованном виде [1, 22] задача оптимизации заключается в определении значений независимых переменных $x_1, ..., x_n$, при которых критерий оптимальности задачи оптимизации $f(x) = f(x_1, x_2, ..., x_n)$ принимает экстремальное (максимальное или минимальное значение) при условиях

$$\varphi_i(x_1, ..., x_n) \le 0, i = \overline{1,k}; \quad (1.1)$$

$$x_j^- \le x_j \le x_j^+, \quad (1.2)$$

где x_j^- - нижняя граница области определения переменной x_j;

x_j^+ - верхняя граница области определения переменной x_j.

В дальнейшем будем предполагать, что критерий оптимальности должен быть минимален. Это не накладывает дополнительных ограничений на выполнение процедуры оптимизации, так как возможен переход (рис. 1.1) между задачами минимизации и максимизации: *max(f(x)) = - min(- f(x))*.

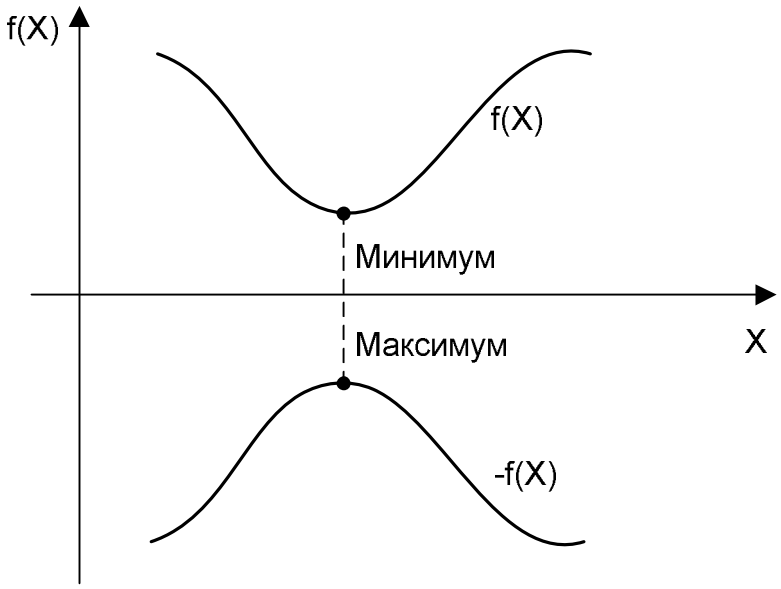


Рис. 1.1. Графическая интерпретация перехода между задачами минимизации и максимизации

Ограничения вида

$$\varphi_i(x_1,...x_n) \geq 0 \qquad (1.3)$$

путем умножения левой части неравенства на минус единицу могут быть приведены к выражению (1.1).

Ограничения (1.1) и (1.3) в своей исходной постановке могут быть заданы в виде уравнений. При задании системы неравенств всегда можно перейти от неравенств к уравнениям

$$\varphi_i(x_1,...,x_n,x_{n+i}) = 0 \qquad (1.4)$$

путем введения дополнительных переменных x_{n+i}, причем для неравенств вида (1.1) $x_{n+i} > 0$, а для неравенств вида (1.3) $x_{n+i} < 0$.

В задаче оптимизации, в которой ограничения имеют вид уравнений, количество ограничений k не может быть больше числа переменных n. Разность n - k определяет число степеней свободы в данной задаче. При этом $n - k$ переменных берутся произвольными, значения же остальных переменных определяются из системы ограничений. Если $n = k$, то число степеней свободы равно нулю и задача в этом случае является алгебраической. Оптимизация целевой функции $f(x)$ при этом не требуется.

В зависимости от числа n переменных оптимизации, структуры области допустимых решений, а также вида целевой функции $f(x)$ задача оптимизации приводится к различным классам экстремальных задач. Ниже отмечаются основные из них [1, 5].

В общем случае математическая модель задачи оптимизации может быть классифицирована по принадлежности к одной либо одновременно к нескольким характеристикам задачи оптимизации, представленных на рис. 1.2. В последнем случае речь идет о наиболее трудно решаемой классическими методами смешанной задачи оптимизации.

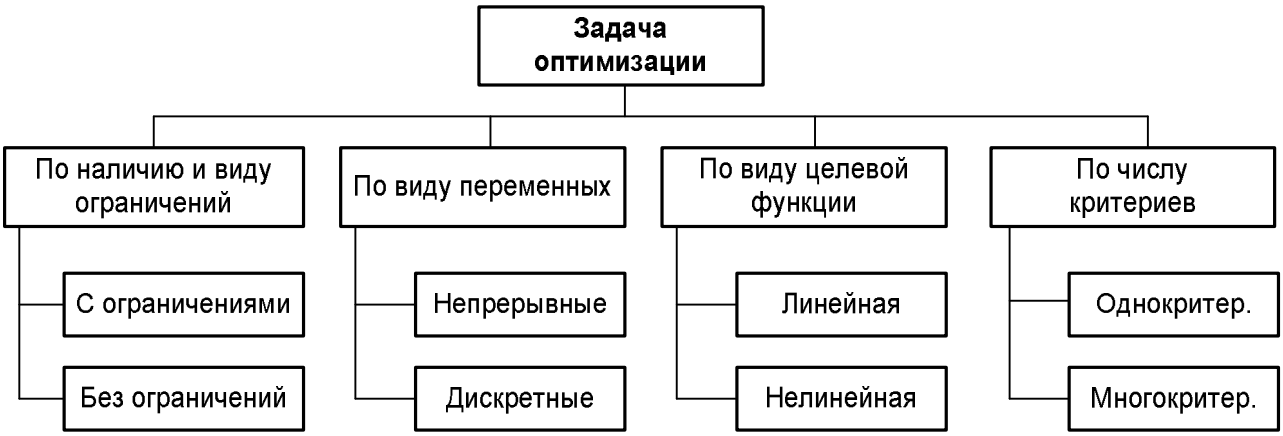


Рис. 1.2. Обобщенная классификация задач оптимизации

Задача оптимизации называется задачей линейного программирования, если критерий оптимальности и ограничения являются линейными функциями параметра x

$$\sum_{j=1}^{n} c_j x_j \to \min; \tag{1.5}$$

при условии

$$\sum_{i=1}^{k} a_{i,j} x_j \geq b_i, \quad i = 1,2,\ldots, k;$$

$$x_j \geq 0, j = 1,2,\ldots,n.$$

В тех случаях, когда критерий оптимальности или ограничения являются нелинейными функциями, говорят о задаче нелинейной оптимизации, которая имеет несколько различных постановок.

При отсутствии нелинейных ограничений (1.3) задача оптимизации сводится к поиску минимума функции $f(x)$, определенной в n-мерном евклидовом пространстве R^n

$$\min_{x \in R^n} f(x). \tag{1.6}$$

Задача (1.6) называется задачей нелинейной оптимизации без ограничений (или задачей поиска безусловного минимума). При наличии ограничений, связывающих переменные x, такая задача называется задачей нелинейного программирования.

Общим для обоих типов задач оптимизации является то, что в зависимости от числа варьируемых переменных они могут быть одномерными (n=1), когда выполняется поиск минимума произвольной кривой *f(x),* или многопараметрическими ($n \geq 2$), связанными с минимизацией некоторой n – мерной гиперповерхности *f(x)*. При этом оптимальное решение x^* в зависимости от вида функции *f(x)* может быть либо точкой локального, либо глобального минимума.

Вектор x^* называется точкой локального минимума, если для всех точек x, принадлежащих ε-окрестности $D(x^*, \varepsilon)$ этой точки, значение $f(x)$ не принимает меньшего значения

$$f(x^*) \leq f(x) \text{ для всех } x \in D(x^*, \varepsilon). \quad (1.7)$$

Точка x^* является точкой глобального минимума, если ни в одной другой точке допустимой области D функция $f(x)$ не принимает меньшего значения

$$f(x^*) \leq f(x) \text{ для всех } x \in D. \quad (1.8)$$

Другими словами, глобальный минимум – это наименьший из всех локальных. На рис. 1.3 приведены точки локальных (x_1^*, x_2^*) и глобального (x_3^*) минимумов для произвольной одномерной функции.

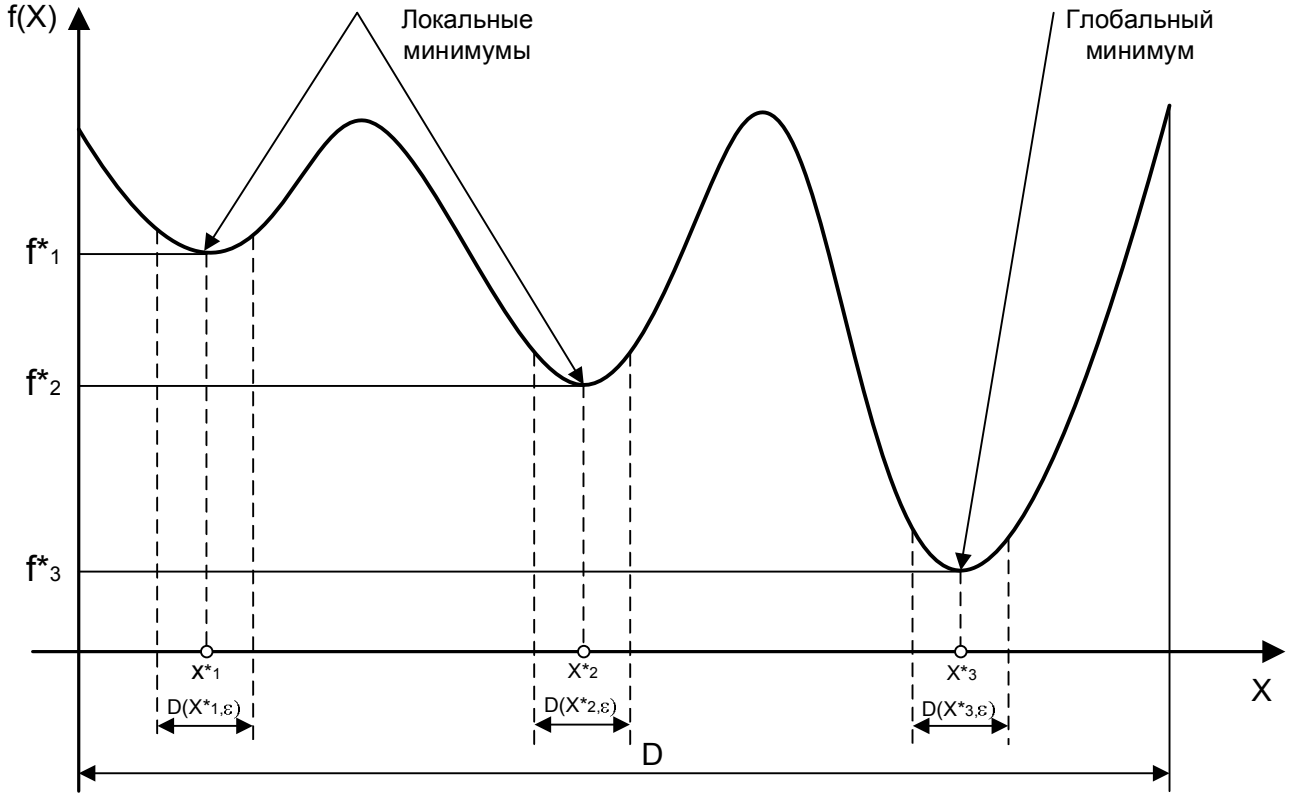


Рис.1.3. Пример локальных и глобального минимумов произвольной кривой

Если *f(x)* имеет в области *D* единственный локальный минимум, то задача оптимизации называется унимодальной (одноэкстремальной), в противном случае – многоэкстремальной.

Говоря о локальных и глобальных оптимумах, важно отметить, что нахождение последних является достаточно сложной задачей и очень зависит от характера оптимизируемой целевой функции. Действительно, поиск глобального экстремума некоторой задачи из разряда тестовых (такие задачи используются для проверки эффективности методов оптимизации), про которые известно как о рельефе поверхности функции, так и о распределении на нем точек оптимумов, будет значительно отличаться от оптимизации неизвестной математической модели какой-либо прикладной задачи. В этом случае, как правило, ставится цель найти рациональное (не худшее) решение, соответствующее локальному оптимуму. Подтвердить же тот факт, что это решение абсолютно оптимально, то есть соответствует глобальному экстремуму, может только экспериментальное исследование применения выбранного численного метода либо некая априорная информация об оптимизационной задаче.

Задача оптимизации с дополнительным требованием, чтобы переменные *x* принимали только дискретные значения, составляет отдельный класс задач дискретной оптимизации.

В ряде задач возникает необходимость получить наилучшие значения одновременно для нескольких критериев. Обычно эти критерии противоречивы и оптимизация по каждому из них приводит к разным значениям переменных *x*. В связи с этим для совместного учета всей совокупности частных критериев рассматривают векторный критерий оптимальности $f(x)=[f_1(x),...,f_m(x)]\to\min$, приводящий к задаче многокритериальной оптимизации.

Оптимальное решение этой задачи в общем случае не является точкой минимума ни для одного из частных критериев и выбирается так, чтобы обеспечить компромисс между частными критериями. Один из путей решения этой задачи состоит в сведении ее к однокритериальной с помощью конструирования специальной, как правило, нелинейной скалярной функции посредством введения системы приоритетов и назначения каждой ЦФ некоторого безразмерного множителя.

1.1.2. Краткий анализ классических методов оптимизации

После постановки задачи оптимизации и разработки ее математической модели может быть выбран соответствующий метод ее решения, которым выступает, как правило, численный метод оптимизации.

В настоящее время существует множество методов оптимизации, которые предназначены для решения своего класса задач оптимизации (линейное, нелинейное, динамическое и т.д. программирование) и с точки зрения стратегии поиска оптимума образуют четыре группы: аналитические, рекурсивные, итерационные и стохастические. Традиционным методам оптимизации, анализу их применимости к различным классам задач посвящен целый ряд работ [5, 13, 32, 40]. Поэтому ниже кратко рассматриваются лишь принципиальные отличия их друг от друга.

Аналитические методы находят применение при решении классических задач оптимизации и задач с ограничениями в виде уравнений. Для решения задач без ограничений используют методы исследования производной функции. Путем приравнивания производной нулю отыскиваются точки экстремума, а затем исследуются точки с помощью второй производной для отыскания оптимума.

Рекурсивные методы относятся к методам, позволяющим определить одну переменную за одну расчетную операцию. Решение всей задачи осуществляется путем поочередного определения переменных. Наиболее распространенным среди этих методов является динамическое программирование. Этот метод можно использовать при анализе многоэтапных процессов принятия решения при небольшом числе ограничений, вводимых в математическую модель.

Итерационные методы объединяют наибольшую группу методов поиска оптимумов. К ним относятся способы расчета функции цели в одной или нескольких вероятностных точках для определения «лучшей» точки. Расчет выполняют до тех пор, пока не приблизятся к назначенному критерию на расстояние, меньшее некоторого заданного значения. Эти методы позволяют устанавливать только локальные

оптимумы, однако они могут применяться тогда, когда оптимизацию проводят в различных исходных точках. Оптимумы, определяемые этим способом, представляют собой достаточно точное решение относительно глобального экстремума.

Различают два больших класса итерационных методов: методы линейного и нелинейного программирования.

Линейное программирование применяют для решения линейных задач, когда функции цели и ограничения являются линейными, а все переменные - непрерывными функциями. В основу этого программирования положено утверждение, что точка оптимума целевой функции находится в одной из вершин выпуклого многогранника, определяющего область возможных решений. Наиболее известным итерационным методом решения линейных задач является симплекс-метод.

Для методов нелинейного программирования характерно непосредственное отыскание оптимума. Эти методы разделяются на две группы: прямые и косвенные. Прямые методы основаны на сравнении вычисляемых значений ЦФ в различных точках, а косвенные – на использовании необходимых и достаточных условий математического определения максимума и минимума функции. Стратегия поиска прямых методов заключается в постепенном приближении к оптимуму, косвенных – стремление найти решение, не исследуя неоптимальные точки посредством определения наиболее перспективного направления локализации оптимума с помощью вычисления производных. К первой группе относятся методы Фибоначчи, Хука-Дживса и др., а ко второй - методы наискорейшего спуска с различными модификациями [5, 13, 40]. Все методы непосредственного поиска оптимума включают операции выбора направления поиска и длины шага, при этом отдельные методы имеют разные критерии выбора этих двух параметров.

Большинство методов непосредственного отыскания оптимума не может быть применено к математическим моделям с ограничениями. В этом случае осуществляют переход к задаче без ограничений с помощью конструирования специальной функции, например на основе методов штрафных функций, множителей Лагранжа.

Следует отметить, что в общем виде задача нелинейного программирования не имеет строгого математического решения и соответственно универсального (например, такого как симплекс-метод в линейном программировании) численного метода. Поэтому для ее решения применяется несколько методов поиска, но даже в этом случае успех нахождения абсолютного оптимума может быть сопряжен не только со знанием физической сущности рассматриваемой проблемы, но и со значительными трудностями при вычислениях. Действительно, среди оптимизационных задач существует широкий класс, который не может быть отнесен к одному из рассмотренных, потому что сочетает в себе свойства двух и более. Подобные математические модели характерны для задач, возникающих в конкретных прикладных областях. Так, в задачах оптимального проектирования процесс оптимизации характеризуется наличием как непрерывных параметров (управляющих переменных), так и необходимостью оптимального выбора значений конкретных дискретных компонентов проектируемой системы, например заданных таблично. В результате возникает задача оптимизации в смешанной постановке, где сочетаются как непрерывные, так и дискретные переменные. В таких условиях применение градиентных методов оптимизации может быть затруднено из-за отсутствия у математической модели аналитических зависимостей для определения первых и последующих производных целевой функции. В такой ситуации поиск рациональных решений ведется с помощью методов прямого поиска. Стоит отметить, что подобные методы наиболее полно приближены к реальным оптимизационным проблемам, в частности в области оптимального проектирования [41]. Недостатком таких методов являются несколько большие по сравнению с градиентными вычислительные затраты, однако он нивелируется хорошей адаптируемостью прямых методов под параллельные архитектуры.

К настоящему времени разработано множество методов прямого поиска и их модификаций, среди который наибольшее распространение получили релаксационный метод Гаусса-Зейделя, метод конкурирующих точек, комплекс-метод и др. Во многом принципы функционирования подобных методов похожи и основываются на поочередном изменении значений каждой переменной целевой функции по определенным правилам. Рассмотрим их на примере комплекс-

метода как наиболее простого по реализации и эффективного по качеству решения оптимизационных задач [57, 62, 63, 66].

Комплекс-метод (Complex method) был предложен в [62] как попытка создания универсального численного метода для решения оптимизационных задач в любой постановке (подобно симплекс-методу в линейном программировании). В комплекс-методе главная сложность связана с его особенностью, а именно необходимостью манипулирования несколькими возможными решениями (точками в пространстве поиска) одновременно. Число таких точек n_c должно быть больше числа оптимизируемых параметров, то есть $n_c \geq n+1$ (обычно n_c принимают в два раза большим, чем n). Алгоритм комплекс-метода в общем виде включает следующие операции.

1. Случайным образом генерируется набор точек с проверкой на соответствие заданным ограничениям. Будем считать, что $x^*_{\max}$ и $x^*_{\min}$ представляют точки соответственно с максимальным и минимальным значениями целевой функции.

2. Вычисляется центроид $\bar{x}$

$$\bar{x} = \frac{1}{n-1}\sum_{j=1}^{n} x_j , \quad x^*_{\max} \neq x^*_{\min} . \tag{1.9}$$

Главная идея алгоритма состоит в замене худшей точки новой лучшей точкой. Новая точка определяется как отражение худшей точки относительно центроида оставшихся точек по следующему выражению:

$$x_r = \bar{x} + \alpha(\bar{x} - x^*_{\min}), \tag{1.10}$$

где α – коэффициент отражения, рекомендуется значение 1,3.

Полученная точка x_r проверяется на соответствие ограничениям, в случае успеха ею заменяется $x^*_{\min}$. Если же не удалось получить лучшую точку, то процесс повторяется. Подобная ситуация может быть следствием концентрации точек около экстремума либо слишком близкого расположения их друг относительно друга. Все это может привести к преждевременной сходимости метода.

Выходом здесь может служить присваивание случайного значения новой точке. В этом случае алгоритм получает дополнительные

возможности поиска лучшей точки в новой окрестности пространства решений. На рис. 1.4 представлена графическая иллюстрация процесса работы комплекс-метода в двумерном пространстве поиска. Окружности представляют уровни значений целевой функции с оптимумом в центре.

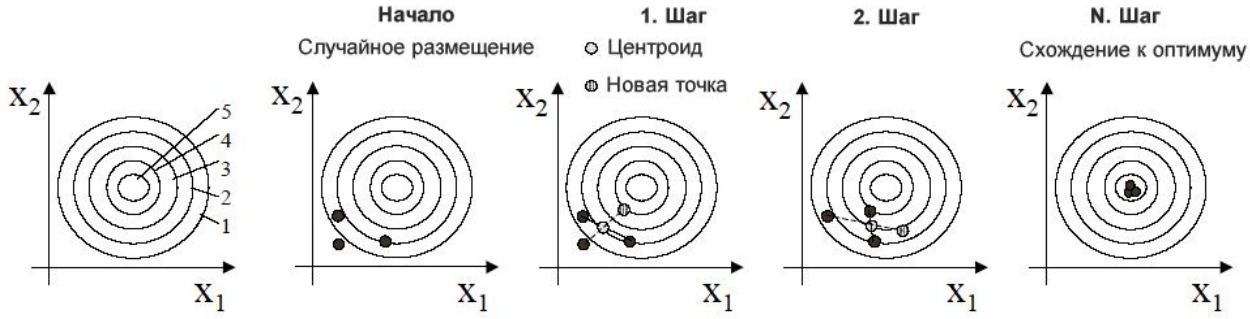


Рис. 1.4. Процесс работы комплекс-метода

Решить задачу в смешанной постановке можно и перебором. Известно, что практически любая задача оптимизации может быть сведена к дискретной путем дискретизации интервалов значений переменных и решена с помощью полного перебора, который гарантирует нахождение глобального оптимума. Однако, помимо того, что математическую модель задачи нужно модифицировать под дискретную, еще возникает главная проблема неэффективности полного перебора, связанная с его большой трудоемкостью. Действительно, объем вычислений, необходимых для сужения неопределенности в многомерном пространстве, является степенной функцией, показатель которой связан с размерностью пространства.

Компромиссом в такой ситуации могут служить методы стохастического или случайного поиска, которые реализуют идею сокращенного перебора и предусматривают использование в процессе нахождения экстремума только значений самих функций без определения их производных.

Стохастические методы оптимизации образуют отдельный класс методов прямого поиска и включают процедуры накопления и обработки информации, в которые сознательно вводится элемент случайности. Преимущества этих методов заключаются в их простоте, достаточной точности и легкости программирования. В результате методы случайного поиска стали одними из наиболее эффективных

методов оптимизации для рассматриваемого класса задач. Подробное рассмотрение таких методов приведено в [32].

Принципы работы методов случайного поиска основываются на следующем. На каждом шаге по известному закону распределения r и заданным верхним x_j^+ и нижним x_j^- пределам рассматриваемых переменных x_j определяются их значения

$$x_j = x_j^- + r_j(x_j^+ - x_j^-), \qquad (1.11)$$

где j – номер переменной;

r_j - случайное число, распределенное на интервале [0, 1].

Полученная таким образом точка x_r проверяется на допустимость и в случае несоответствия, вычисления продолжаются до тех пор, пока не будет найдена точка, которая удовлетворяет всем ограничениям. Далее в этой точке вычисляется значение функции $f_r = f(x_r)$. Точка с наименьшим значением функции запоминается, если выполняется условие:

$$f^* = \begin{cases} f_r, \ \textit{если } f_r < f^*; \\ f^* \ \textit{в противном случае.} \end{cases} \qquad (1.12)$$

$$x^* = \begin{cases} x_r, \ \textit{если } f_r < f^*; \\ x^* \textit{- в противном случае.} \end{cases}$$

Полученная точка далее проверяется на допустимость путем вычисления значений функций, входящих в ограничения. Вычисления продолжаются до тех пор, пока не будет найдена точка, которая удовлетворяет всем ограничениям.

После некоторого числа шагов поиска получаем точку x^*, которая принимается за приближенное значение глобального минимума.

Пусть p – вероятность того, что при N испытаниях точка приближенного значения глобального минимума x^* будет определена с точностью ε. Под точностью здесь понимается объем n-мерного параллелепипеда, выраженный в долях от общего объема области D. Тогда величину ε можно интерпретировать как вероятность попадания каждого испытания x_r в область объемом, равным ε. После N испытаний вероятность того, что одно из них попадет в ε, будет $p = 1 - (1 - \varepsilon)^N$. Откуда

$$N = \lg(1 - p) / \lg(1 - \varepsilon). \qquad (1.13)$$

Из (1.11) видно, что для решения конкретных задач с высокой точностью ε при помощи этого алгоритма требуется проводить большое число испытаний.

Развитие таких методов связано с выбором эффективного способа генерации новой точки, например путем выборок из нормального распределения с центром в наилучшей точке и дисперсией, корректируемой эвристическим методом.

Подобно градиентным методы случайного поиска исследуют пространство решений от точки к точке, что не всегда эффективно для задач большой размерности.

Нельзя не отметить возможности использования гибридных методов оптимизации, сочетающих в себе совместное использование градиентных и прямых методов. Это бывает эффективно для целевых функций со сложной поверхностью, исследование которой требует как определения направления поиска (глобальная стратегия поиска), так и наилучшей длины шага (локальная стратегия поиска).

Отмеченные методы оптимизации относят к классическим, их развитие связано с постоянной модификацией под конкретные задачи, которые эти численные методы позволяют в целом эффективно решать. В то же время круг таких задач, область их применения достаточно узкая, что приводит к необходимости адаптации математической модели задачи под конкретный метод оптимизации. Все это в результате приводит к уменьшению значимости результата от решения поставленной задачи.

Выходом из такой ситуации могло быть использование стохастических методов оптимизации, особенно в задачах смешанного характера, однако практика использования не подтвердила их претензии на первенство, что связано с отмеченными недостатками.

Обобщая изложенное, можно утверждать, что каждой задаче оптимизации свойственны свои специфические особенности, затрудняющие выбор и применение традиционных алгоритмов оптимизации. Поэтому качество результатов во многом определяется выбором метода оптимизации, которому должна предшествовать всесторонняя оценка математической модели задачи, включающая анализ топологических свойств поверхности ЦФ, а также характер переменных оптимизации. Так, для гладких мультимодальных, а также поверхностей

с глубокими впадинами целесообразно использовать разные алгоритмы оптимизации, которые к тому же сами по себе могут сильно отличаться по таким критериям, как эффективность и универсальность. Эффективность алгоритма определяется числом вычислений ЦФ, необходимых для локализации решения с требуемой точностью. Свойство универсальности означает возможность применения алгоритма для решения разнообразных задач. С этой точки зрения классические методы оптимизации могут быть названы универсальными только применительно к своему классу задач. В то же время частое отсутствие априорной информации о топологический свойствах поверхности оптимизируемой функции приводит к необходимости привлекать разные алгоритмы для увеличения доли удачных решений.

Таким образом, отмеченное многообразие задач и методов оптимизации при отсутствии единого подхода к их решению позволяет сделать вывод о сильной зависимости эффективности решения имеющейся задачи от набора алгоритмов поисковой оптимизации, с которыми знаком исследователь, что далеко не всегда положительно сказывается на окончательном результате. Действительно, сейчас, наряду с повсеместным, с точки зрения прикладных областей, использования ЭВМ, возникает потребность в решении оптимизационных задач в самой различной постановке, даже при отсутствии полной информации о ней. В такой ситуации становятся актуальными попытки применения оптимизационных процедур из других научных направлений. Так, в последнее время, наряду с традиционными подходами к оптимизации, применяются технологии научного направления «искусственный интеллект» (ИИ), а именно методы эволюционного моделирования (ЭМ).

Методы эволюционного моделирования представляют собой группу методов адаптивного поиска и многопараметрической оптимизации, копирующие механизмы естественной эволюции и обладающие следующими особенностями:

- наряду с обычным чаще всего используется закодированное представление параметров задачи;
- поиск ведется не из единственной точки, а из множества (популяции) точек;
- специфика работы позволяет накапливать и использовать знания об исследованном пространстве поиска и, следовательно, проявлять способность к самообучению;

- в процессе поиска используется значение ЦФ, а не ее приращения;

- применяются вероятностные, а не детерминированные правила поиска и генерации решений;

- выполняется одновременный анализ различных областей пространства решений, в связи с чем возможно нахождение новых областей с лучшими значениями ЦФ путем объединения субоптимальных решений из разных популяций.

Следствием приведенных свойств является то, что в отличие от традиционных методов поиска оптимального решения, ориентированных на задачи с «хорошим поведением», методы ЭМ позволяют охватить гораздо более широкий круг задач (рис. 1.5) в сочетании со следующими преимуществами:

- независимость от вида функции, включая поддержку неаналитического задания функции;

- независимость от области задания переменных оптимизации;

- применение к широкому диапазону задач без модификации алгоритма;

- высокую помехозащищенность и как следствие адекватную робастность (способность лишь постепенно снижать качество работы по мере приближения к границам допустимой надежности данных).

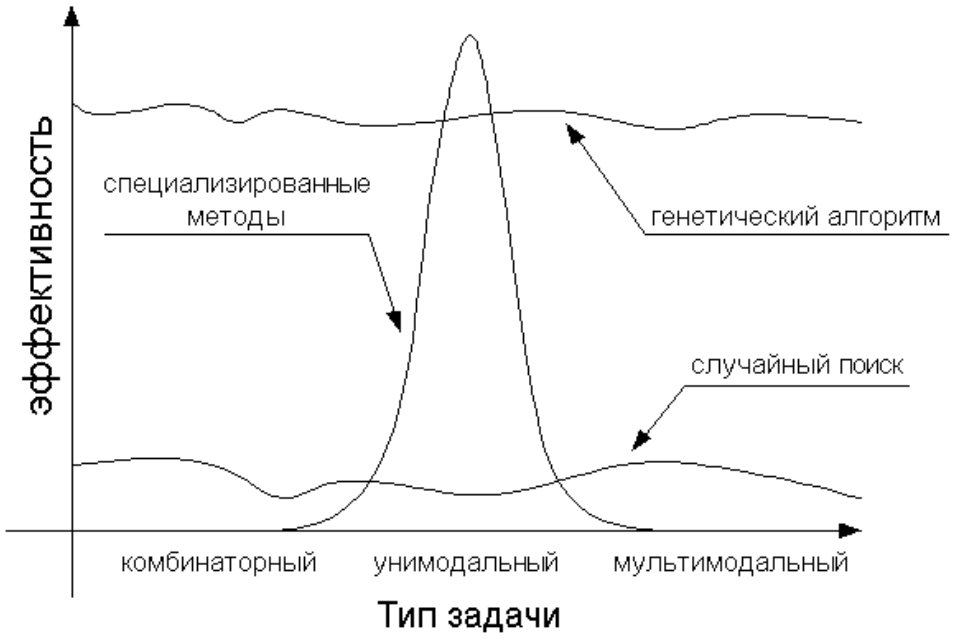


Рис. 1.5. Сравнительная эффективность алгоритмов оптимизации

1.2. Концепция и принципы эволюционного моделирования

В настоящее время эволюционное моделирование образует самостоятельную бионическую ветвь искусственного интеллекта, развитие которой происходит на стыке двух наук: информатики и биологии (рис. 1.6).

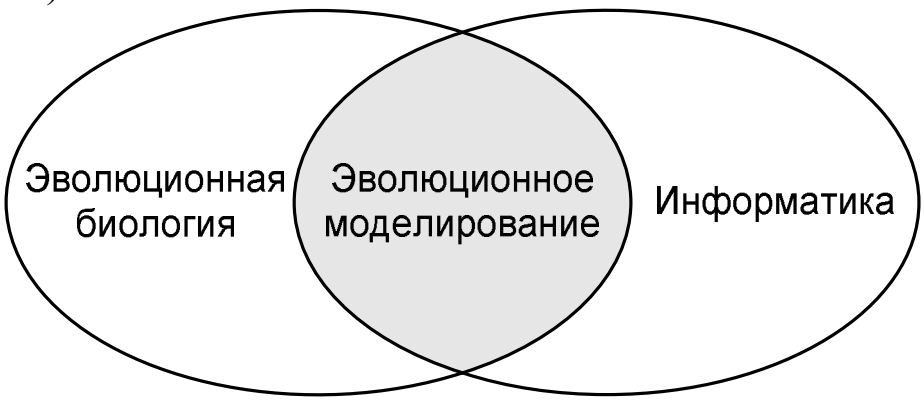


Рис. 1.6. Место эволюционного моделирования среди наук

При этом под термином «эволюционное моделирование» обычно подразумевается направление в искусственном интеллекте, в основе которого лежат принципы и понятийный аппарат, заимствованные из популяционной генетики и объединяющее компьютерные методы моделирования естественных эволюционных процессов (генетические алгоритмы, генетическое программирование, эволюционное программирование и эволюционные стратегии).

1.2.1. Основные понятия эволюционного моделирования

Впервые концепция ЭМ была сформулирована в 1960-х годах Л.Фогелем (L. Fogel) в его работе «Искусственный интеллект и эволюционное моделирование» [39]. Однако достойного внимания ЭМ тогда не получило, что может быть связано как с достаточно декларативным характером этой работы, так и с активно развивающимися другими методами ИИ, в частности, эвристическим программированием.

Позже, в конце 1990-х годов, в зарубежных исследованиях наблюдается новый виток интереса к этой области новых информационных технологий, результатом чего стало формирование отдельного направления ИИ – «эволюционные вычисления» (Evolutionary Computations). Прежде всего это связано с появлением исключительно прикладных методов ЭМ, известных как эволюционные алгоритмы (ЭА), которые стали активно применяться при разработке программного обеспечения различного назначения.

Однако независимо от периодов исследований этого направления основополагающим тезисом ЭМ оставался следующий: «заменить процесс моделирования сложного объекта моделированием его эволюции» [9]. Именно отбор наилучших объектов по результатам эволюции их коллективного поведения является ключевой эвристикой всех эволюционных методов. Если попытаться выразить эту эвристику на естественном языке, то получим: сложно получить самое лучшее решение, модифицируя плохое. Скорее всего, оно получится из нескольких лучших на данный момент.

Термин «эволюция» в ограниченном смысле, касающийся только смены поколений организмов, известен уже очень давно. Однако фундаментальное толкование этот термин получил в работе Чарльза Дарвина «Происхождение видов», в которой он заложил основные принципы эволюционной теории. Самым важным его выводом было заключение об основной направляющей силе эволюции, ею признавался естественный отбор, при котором выживает сильнейший («Survival of the fittest») в широком смысле этого слова. Вторым не менее важным выводом Дарвина был вывод об изменчивости организмов.

Рассмотрим основные принципы, которые лежат в основе биологической составляющей ЭМ [26]. Более же детальный обзор этого вопроса приведен в [16].

Известно, что первую последовательную и непротиворечивую теорию эволюции разработал в начале XIX века Жан Батист Ламарк. В ее основе лежали два допущения: о наследовании приобретенных признаков и о внутренне присущем всему живому «стремлению к совершенству».

Первая гипотеза объясняла, почему организмы так хорошо приспособлены к условиям обитания. В течение жизни они используют свои органы по-разному: одни чаще, другие реже. Те органы, которые все время «тренируются», - крепнут и растут, а «оставшиеся без работы» - уменьшаются и слабеют. Небольшие изменения, возникающие вследствие такой избирательной тренировки, передаются по наследству. Так, если животное питается листьями высоких деревьев, ему приходится все время вытягивать шею. Шея тренируется, крепнет и немножко удлиняется. Потомство такого животного уже от рождения получит чуть более длинную шею. Так, по мнению Ламарка, появились жирафы. Если какая-то птица перестает летать, переходит к наземной жизни, то ее крылья от долгого неупотребления атрофируются. Так возникли нелетающие птицы с рудиментарными крыльями. В подтверждение этой гипотезы можно упомянуть эффект, описанный в 1896 г. американским биологом Дж. Болдуином и заключающийся в том, что искусственный навык, полностью выработанный в ходе эволюции в дальнейшем закрепляется на генетическом уровне и начинает передаваться по наследству.

Второе предположение Ламарка – внутренняя «тяга к совершенству» - объясняло постепенное усложнение организмов, появление новых органов и тканей.

Дарвин внес коррективы в теорию Ламарка. Он отказался от второй посылки своего предшественника – от «тяги к совершенству» - и предложил такой механизм эволюционных изменений, которого теория Ламарка не предусматривала, а именно естественный отбор. Механизм естественного отбора основан на борьбе за существование (причина, которой в том, что живые существа производят больше потомков, чем может выжить), изменчивости (ее причины Дарвин, не зная генетики, не мог сформулировать и принимал ее просто как данность) и наследственности, благодаря которой свойства, помогающие данной особи выжить, передаются ее потомству.

В дальнейшем наука генетика на основе общепринятой сейчас «синтетической теории эволюции» внесла коррективы в эти постулаты и доказала многие существовавшие предположения о развитии организмов. В частности, ученым И.И. Шмальгаузеном были выделены

и остаются аксиомой следующие необходимые и достаточные условия возникновения эволюции [9]:

- наследственная изменчивость (мутации) как предпосылка эволюции;

- борьба за существование как контролирующий и направляющий фактор;

- естественный отбор как преобразующий фактор.

На рис. 1.7 представлено взаимодействие указанных факторов (главные выделены пунктирной линией) с учетом многообразия форм их проявления, взаимосвязей и взаимовлияния.

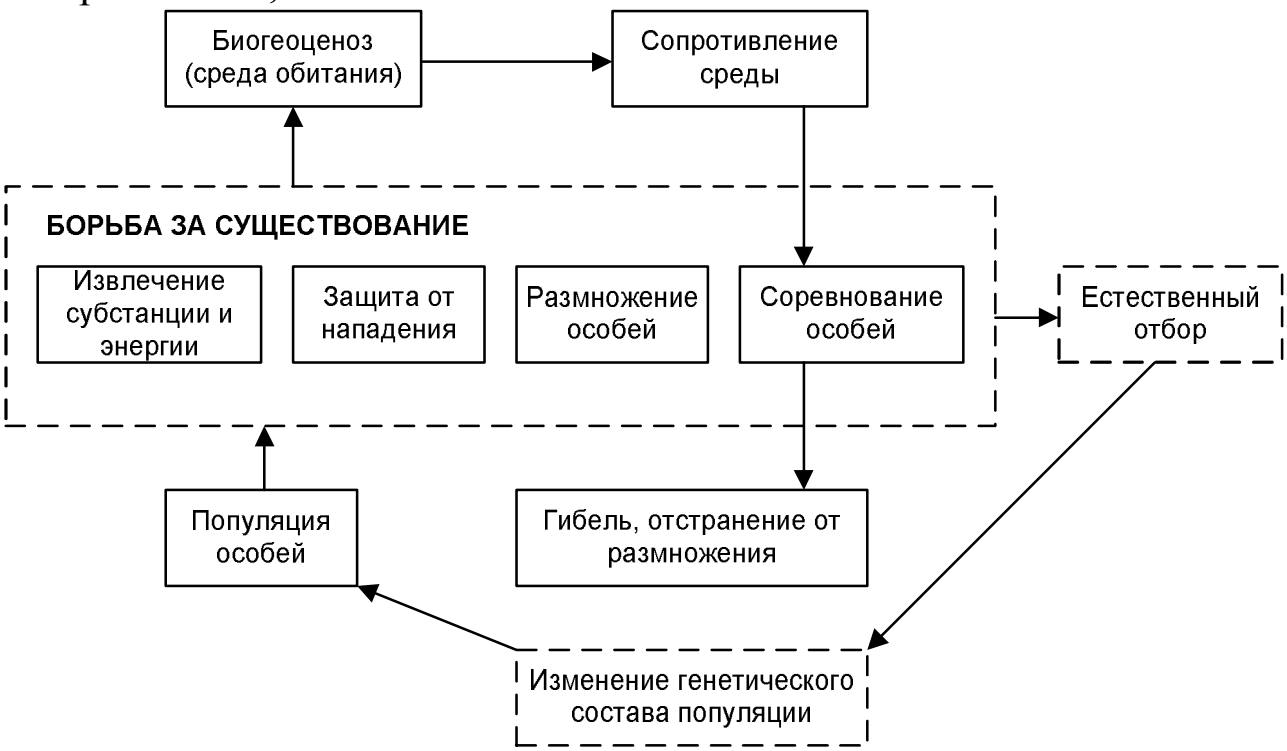


Рис. 1.7. Взаимодействие факторов эволюции

Конкуренция и отбор являются необходимыми, но не достаточными факторами для начала эволюционного процесса. Для эволюции системы ключевым является наличие мутации – случайного наследственного изменения свойств системы. При этом эффект от мутации может быть разным: вредным либо полезным. В первом случае возможно разрушение важных с точки зрения оптимальности системы свойств. Во втором – у системы случайно возникает выгодный с точки зрения конкурентоспособности признак, который в процессе эволюции закрепляется и усиливает доминирование систем с этим

признаком. В [15] такие возможности в эволюции систем называются инновациями, а способность системы к их порождению с последующим отбором и распространением ценных вариантов является решающей предпосылкой ее способностью эволюционировать.

С другой стороны, процесс эволюции нельзя свести к простому суммированию или комбинированию мутаций. Для полноценного эволюционного процесса необходимо сочетание отмеченных ранее факторов (с различной степенью их доминирования). Взаимодействие мутации и отбора в процессах эволюции можно описать следующим образом [15].

Каждая мутация означает возмущение в установившемся на некоторое время равновесии. Мутация вызывает проверку системы на устойчивость. Если мутация не дает никаких преимуществ по сравнению с существующими объектами, то мутированные объекты исчезают в результате процесса отбора. Как следствие система оказывается устойчивой относительно возмущения и возвращается в исходное состояние. При наличии преимуществ от мутации система оказывается неустойчивой и переходит в новое состояние, соответствующее более высокому уровню эволюции.

Рассмотренные биологические принципы развития и являются главной составляющей методов ЭМ, работа которых есть не что иное, как эволюция, совершенствование решений той или иной задачи.

Несмотря на то, что ЭМ находится на стыке биологии и информатики, именно последняя играет доминирующую роль в его применении. Биология здесь выступает скорее словарем основных понятий, источником заимствования идей, законов, попытка моделирования, которых образует основу ЭМ. Обобщенно можно выделить следующие аспекты, которые ЭМ заимствует из биологии:

- понятийный аппарат;
- идею коллективного поиска экстремума при помощи популяции особей;
- способы представления генетической информации;
- способы передачи генетической информации в череде поколений (генетические операторы);

- идею о преимущественном размножении наиболее приспособленных особей.

Учитывая частично заимствованный ЭМ биологический словарь, приведем правила интерпретации соответствующих терминов для эволюционной и математической моделей (табл.).

Таблица

Соответствие терминов эволюционной и математической моделей

Эволюционная модель	**Математическая модель**
Хромосома	Решение, объект, строка, последовательность
Ген	Переменная, параметр, характеристика, признак
Аллель	Значение фрагмента закодированного параметра
Локус	Номер фрагмента закодированного параметра
Генотип	Множество закодированных решений задачи, пространство поиска
Фенотип	Множество решений задачи, пространство решений
Особь, индивидуум	Объект, система
Пригодность, приспособленность	Качество, оптимальность
Fitness-функция	Целевая функция
Популяция	Множество решений
Поколение	Итерация работы ЭА

Пусть имеется некоторое пространство решений, заполненное всеми возможными решения задачи, процесс поиска заключается в исследовании точек этого пространства с целью постоянного улучшения значений параметров задачи (рис. 1.8).

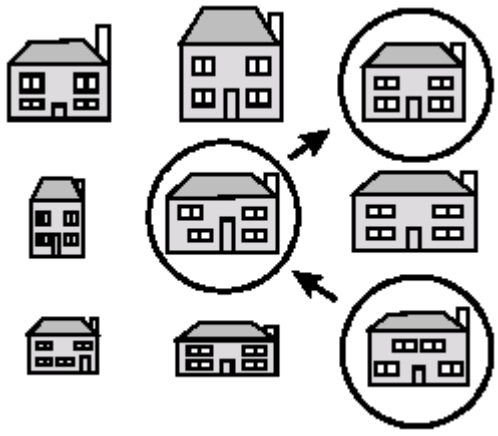


Рис. 1.8. Поиск лучшего решения в пространстве решений

Впрочем, такая постановка является общей для классических поисковых алгоритмов, однако здесь проявляется одно из главных особенностей ЭА: вместо работы с одним решением в каждый момент времени подобные алгоритмы оперируют с большой коллекцией или популяцией решений сразу. В связи с этим можно отметить следующие специфические особенности эволюционного поиска:

- каждая новая популяция состоит только из «жизнеспособных» объектов;

- каждая новая популяция лучше (в смысле приспособленности, оптимальности) предыдущей;

- в процессе эволюции последующая популяция зависит только от предыдущей.

Жизнеспособность столь необычного по отношению к рассмотренным подхода к оптимизации заложенного в ЭА подтверждается практикой его применения к самым различным областям человеческой деятельности с возможностью получения лучших результатов по сравнению с другими методами решения подобных задач [48]. Этот факт подтверждается применением методов ЭМ в таких областях, как проектирование технических систем, диагностика надежности систем, управление робото-техническими комплексами, оптимизация размещения микросхем, планирование работ, анализ временных рядов, машинное обучение и др. Приведенные данные объясняются тем, что в настоящее время методы информационных технологий применяются в самых различных наукоемких отраслях и с точки зрения исследователя вместо создания, поиска узкоспециализированных подходов решения поставленной задачи предпочтительнее и удобнее использовать универсальные алгоритмы, испытанные своим применением на устойчивость и эффективность [48].

Главное же обоснование этого видится в том, что сама идея оптимальности пришла в науку из биологии и многие методические приемы оптимального проектирования имеют корни в селекционной практике и являются примером не всегда осознанного подражания природе [11]. Более того, само мышление человека, а точнее его творческая составляющая, является эволюционным процессом,

направленным на комбинаторную оптимизацию эвристик, решений, идей, возникающих на подсознательном уровне.

Методологическая основа ЭМ базируется на гипотезе селекции, которая может быть сформулирована так: чем выше приспособленность особи, тем выше вероятность того, что в потомстве, полученном с ее участием, признаки, определяющие приспособленность, будут выражены еще сильнее. Поэтому механизмы, заложенные в ЭМ, в общем виде реализуют набор инструкций, позволяющих компьютеру управлять популяциями решений, а именно разрешить лучшим решениям выступать в качестве родительских и «иметь потомство» и соответственно худшим – «погибнуть» (они отбрасываются на текущем шаге эволюции). Решения – потомки наследуют характеристики их родителей с небольшой вариацией, определяемой некоторой вероятностью и затем лучшие из этих решений получают возможность «размножения» путем «скрещивания» с другими индивидуумами в популяции, наименее пригодные члены популяции в силу этого «вымирают». Полученная новая популяция возможных решений образует, таким образом, новое «поколение», сохраняющее в значительно большей пропорции те качества (признаки), которые были присущи лучшим представителям предыдущего поколения. Эта процедура является основой эволюции, и после некоторого числа поколений, последовательно улучшая популяцию путем сохранения и преумножения в ней наиболее сильных сторон индивидуумов, компьютер имеет возможность выделить решения, которые в значительной степени превосходят своих предков в самом начале работы ЭМ (рис. 1.9).

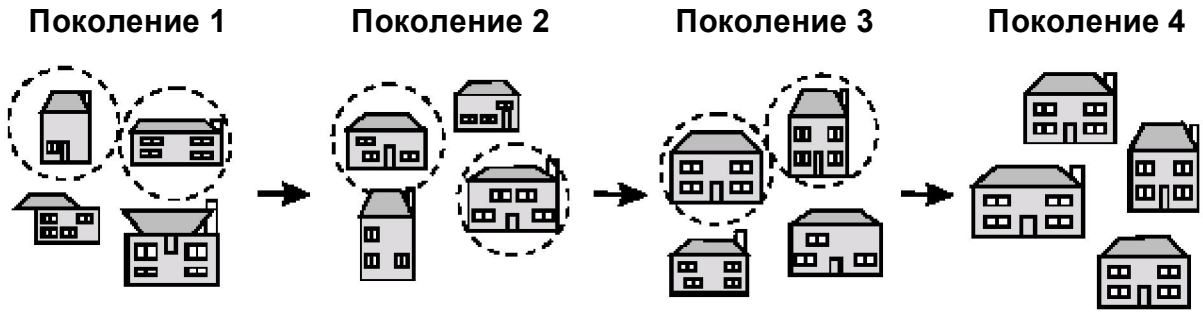


Рис. 1.9. Четыре эволюции в задаче поиска оптимального решения с размером популяции – четыре. Родители следующего поколения обведены линией

В соответствии с областью поиска рассмотрим, как в процессе эволюции находятся оптимальные решения. На рис. 1.10 представлено пространство поиска для примера, изображенного на рис. 1.9. Из рис. 1.10 видно, что эволюционный поиск исследует пространство параллельно, то есть для приведенного примера одновременно рассматриваются четыре объекта, данная особенность позволяет локализовать лучшую с точки зрения оптимальности область в пространстве решений, в результате чего получено несколько хороших решений всего за четыре поколения. Кроме этого, данный факт можно объяснить еще тем, что различные параметры в реальных системах в большинстве своём слабосвязаны [31], то есть постепенное изменение одного из них не требует радикального изменения второго для восстановления оптимальности, что позволяет исследовать пространство. Кроме этого, функции оптимальности, как правило, имеют спуски и подъёмы и не являются сплошным изрезанным полем, что создаёт почву для постепенного приспособления.

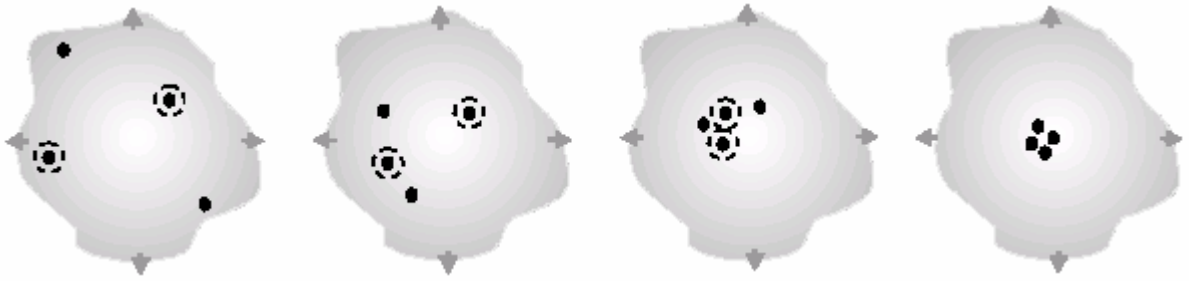


Рис. 1.10. Размещение выделенных решений в каждом поколении. Лучшие решения найдены в центре пространства поиска

Процесс ЭМ неразрывно связан с его реализацией на ЭВМ, поэтому одним из ключевых вопросов является представление информации о задаче в эволюционных алгоритмах.

Подобно тому, как в задаче оптимального проектирования вектор переменных описывает объект, в терминологии, принятой в генетике, признаки индивидуума определяются с помощью его генотипа – набора генов, структурированных в виде хромосомы. В частности, можно утверждать, что категория «вектор переменных проектирования» играет в технике ту же роль, что и категория «генотип» в биологии. При этом с помощью генов могут быть описаны как количественные, так и качественные переменные простых типов, а также

сложных структур данных. В свою очередь, внешнее представление и проявление этих признаков объекта образуют его фенотип.

Заметим, что только в одном из ЭА – генетическом алгоритме (ГА) - наблюдается четкое разделение между этими понятиями. В то же время ГА является наиболее часто используемым среди методов ЭМ, поэтому понимание роли генотипа и фенотипа в ГА можно назвать принципиальным.

Таким образом, ЭА в процессе своей работы используют два разделенных пространства: пространство поиска (генотипы) и пространство решений (фенотипы). Пространство поиска представляет собой область двоичных наборов закодированных решений задачи, область решений – множество конкретных решений. Подобно тому, как в природе эволюция происходит на генетическом уровне, процесс поиска решений в ЭА осуществляется на уровне генотипа, то есть в процедуре оптимизации закодированные координаты новых точек пространства поиска получаются как результат манипулирования координатами старых. Причем оцениваются они на уровне фенотипа, поэтому закодированные решения или генотипы должны иметь средства однозначного отображения в область конкретных решений задачи или фенотипов (рис. 1.11).

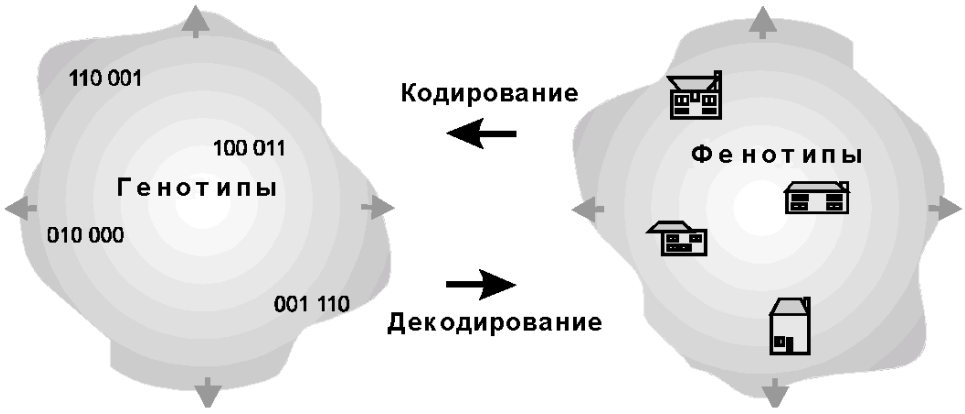


Рис. 1.11. Отображение генотипов пространства поиска в фенотипы пространства решений

Таким образом, при решении задачи поиска ЭА оперирует популяцией или набором объектов, описывающих решение с тем или иным соответствием оптимальности. Каждый индивидуум состоит из генотипа и соответствующего фенотипа. Фенотипы обычно содержат

наборы параметров (в соответствии с рис.1.7 подобные параметры могут определяться количеством и позицией окон, расположением крыши, шириной, высотой дома и т.п.). Генотип, в свою очередь, содержит закодированные версии этих параметров.

Все ЭА требуют для своей работы специальные средства, направляющие процесс эволюции в лучшие области пространства решений. Они получают их посредством вычисления для каждого решения в популяции индекса пригодности и определяется он как «fitness» (приспособленность, пригодность) или в контексте задачи оптимизации – оптимальность, качество, эффективность решения. Оптимальность решения основывается на рейтинговой оценке и показывает, насколько хорошо решение удовлетворяет цели и вычисляется на основе специализированной fitness-функции. Такая функция играет роль направляющей в эволюционном процессе, и ее улучшение соответствует совершенствованию живого организма с позиций приспособленности к реакциям внешнего мира. Продолжая аналогию с природной эволюцией, достижение глобального оптимума есть некоторая идеальная особь, вершина эволюции в определенном природном сообществе. Так, известный палеонтолог, эволюционист и философ Пьер Тейяр де Шарден полагал, что повышение уровня организации живых существ, неуклонно происходящее в ходе эволюции, не может быть объяснено отбором случайных, ненаправленный мутаций и служит доказательством присутствия какой-то особой направляющей силы. Шарден называл ее радиальной энергией, потому что, по его мнению, она движет эволюцию к некоему абсолютному сосредоточию или центру – «точке Омега» [26].

Обычно значения fitness-функции принимают вещественные неотрицательные значения, где нулевое значение соответствует самому лучшему решению. Задача ЭА - оптимизировать данную функцию посредством предоставления лучшим решениям иметь тем больше потомков, чем меньше значение функции. Применительно к рассматриваемому примеру целью проблемы является определение внешнего облика дома, который имел бы четыре равномерно расположенных окна, дверь в центре, дымоход и т.п. Значение fitness-функции рассчитывается на основе полученных решений и показывает степень

соответствия решения цели оптимизации, например при вычислении числа окон функция будет просто увеличиваться (4 – отсутствие окон в решении).

Значения fitness-функции, нанесенные на координатную сетку пространства поиска, образуют «рельефный» fitness-ландшафт, где самому высокому пику и глубокой впадине соответствуют решения, определяющие экстремумы задачи.

Рассмотренные понятия и принципы ЭМ являются основополагающими и свойственными всем ЭА, среди которых в настоящее время выделяют четыре основных типа: генетический алгоритм, генетическое программирование, эволюционное программирование и эволюционные стратегии. Кроме этого, каждый из перечисленных методов эволюционного моделирования имеет от нескольких до множества различных модификаций.

1.2.2. Некоторые прикладные аспекты эволюционного моделирования

Говоря о прикладных свойствах ЭМ, можно утверждать, что все они формируются вокруг моделирования по определенным законам коллективного поведения объектов. Типы этих объектов могут быть совершенно произвольными от оптимизируемых переменных до социально-экономических структур, технических систем и даже биологических сообществ. Главное, чтобы взаимодействие между этими объектами происходило по известным законам эволюции. Наряду с отмеченной ранее высокой эффективностью применения ЭМ в качестве технологии оптимизации, которому уделяется основное внимание в данной книге, существуют и другие классы задач для ЭМ: самоорганизация, адаптивное поведение, «искусственная жизнь» и эволюционное проектирование. Остановимся на них подробнее.

Самоорганизация, или (упрощенно) способность систем улучшать свою организацию является важным свойством как биологических, так и искусственных систем, способных к адаптации в условиях существования (эксплуатации). Но если для первого типа систем способность к самоорганизации является, по сути, врожденной,

передаваемой и совершенствуемой от поколения к поколению, то искусственные системы наделяет этими свойствами специальные алгоритмы. Их особенностью является необходимость работы в условиях постоянной неопределенности, вызываемой периодическими, чаще спонтанными, изменениями внешней среды.

Математической мерой организации может служить величина, обратная энтропии, или любая другая мера избыточности информации. Понятие «организация» может быть связана с понятием «зависимость», считая, что теория организации частично совпадает с функцией многих переменных. О «хорошей» или «плохой» организации можно говорить лишь по отношению к конкретной внешней среде. Поэтому более «хорошей» организацией следует считать ту, которая обеспечивает лучшее приспособление системы к внешним условиям существования. Таким образом, можно определить самоорганизующуюся систему как систему, «живущую» в некоторой среде и изменяющую свою организацию с целью улучшения условий «выживания» [8].

Наиболее сложные из существующих систем, возникших в процессе самоорганизации, безусловно, являются биологическими. Появляясь со случайной, далеко от оптимальной организацией, природные системы под воздействием эволюционных факторов достигли устойчивой формы существования. Моделирование подобных способностей в ЭА изначально привлекало исследователей в самоорганизации систем. Классическим примером здесь является исследование эволюционных механизмов в самоорганизации коллективного поведения автоматов. Теоретическая основа, а также принципы такого моделирования подробно рассмотрены в [8], основная же идея состоит в следующем.

В среду, в которой функционирует, через сеть питания непрерывным потоком поступает «пища». Перед автоматами, входящими в систему, стоит задача выжить в этой среде. При достаточном «питании» автомат активируется, что проявляется в обмене сигналами с другими автоматами, то есть в создании и функционировании каналов передачи информации. Каждой стадии активности автоматов соответствует некоторая структура сигнальной связи, определяемая и

поддерживаемая расходом «пищи». Если количество «пищи», накопленной в каком-либо узле, достигает критической величины, то в этом месте возникает автомат. При недостаточном «питании» автомат погибает. При моделировании среда рассматривалась как двумерная решетка, а автоматы разделялись на два класса: автоматы первого класса могли перемещаться только в продольном (вертикальном) направлении, автоматы второго класса - соответственно только в поперечном (горизонтальном).

Каждый из примитивных автоматов может совершать только движения вдоль своего направления: сдвинуться на шаг вперед, назад или остаться на месте. Необходимость в перемещении связана с тем, что, оставаясь долго на месте, автомат поглощает «пищу» и погибает. Если в каком-то месте соединяются два примитивных автомата одного и того же класса, то получаются автоматы, способные совершать уже пять движений вдоль продольного или поперечного направлений: один и на два шага вперед и назад, а также оставаться на месте. Если же соединяются автоматы разных классов, то получающиеся автоматы могут совершать уже девять движений в двумерном поле. Автоматы могут соединяться до бесконечности, и каждый новый вид совершает все движения своих предшественников.

В процессе функционирования этой системы, чтобы выжить автомат движется в сторону большей концентрации «пищи», оставляя за собой «волну» истощения запасов пищи. Стремясь к местам большей концентрации «пищи», автоматы собираются в одной области среды. При этом они сближаются друг с другом и отражаются друг от друга, создавая цепи колеблющихся автоматов или так называемые когерентные структуры в системе.

Таким образом, управляющим фактором эволюции в этом примере является доступ к потоку «пищи». Процесс самоорганизации состоит в построении коллективом автоматов такой внешней среды, которая будет способствовать выживанию только этой группы, характеризующейся некоторым, в своем роде оптимальным набором движений.

В начале 90-х годов XX в. возникли два необычных тесно связанных между собой направления исследований «Искусственная

жизнь» (Artificial Life) и «Адаптивное поведение» (Adaptive Behavior), которые в системе друг с другом стали качественным развитием исследований по самоорганизации систем, образуя при этом такую дисциплину, как бионика (греч. bion – элемент жизни).

Основной мотивацией исследований **искусственной жизни** послужило желание понять и смоделировать формальные принципы организации биологической жизни, а также синтетически воспроизвести биологическое поведение в различных средах. Практическим приложением этих исследований в настоящее время являются робототехнические системы, медицина, нанотехнологии, моделирование «жизни» социальных и экономических систем.

В качестве методов исследований в этой области обычно используются искусственные нейронные сети и эволюционное моделирование, значительно расширяющие возможности знаменитой игры Дж. Конвея «Жизнь».

Примером компьютерной реализации искусственной жизни может служить система ПолиМир (PolyWorld) [33]. Рассмотрим структуру, а также модель эволюции в этой системе.

Предположим, что есть некое ограниченное пространство (для простоты это может быть большой стол, ограниченный по периметру барьерами), на котором могут жить искусственные организмы (агенты). На столе могут появляться участки с пищей. Организмы могут двигаться прямолинейно, поворачиваться, распознавать соседние объекты (обладают зрением), а также поглощать «пищу», увеличивая тем самым свою энергию. Также организмы могут скрещиваться и бороться друг с другом, при этом каркас побежденного организма становится «пищей». Поведением организмов управляет нейронная сеть, а развитие их популяции осуществляется методами эволюционного моделирования. Поедая «пищу» или проявляя активность, организмы расходуют энергию, если ее ресурс становится ниже определенного уровня, то организм умирает, превращаясь в «пищу». Популяция организмов постоянно эволюционирует, при этом потомки наследуют гены родителей, возможно, со случайными изменениями под воздействием незначительных мутаций. Гены представляют собой закодированные свойства организмов (размер, скорость движения, сила, цвет и т.п.), а также структуру нейронной сети.

В процессе экспериментов с системой ПолиМир был получен целый ряд нетривиальных стратегий поведения организмов. При этом поведение принимало активный, пассивный или осторожный характер. В последнем случае организмы концентрировались по периметру стола, циркулируя по или против часовой стрелке вдоль барьеров. Это приводило к определенным преимуществам, связанным с возможностью исследования всего «мира» без угрозы быть побежденным более сильным соперником.

Современной областью исследований, связанной с моделированием искусственной жизни является так называемая синтетическая (эволюционная) химия, возникшая для реализации потребностей нанотехнологий. Она занимается вопросами молекулярной сборки самоорганизующихся, самособирающихся, самовоспроизводящихся и самовосстанавливающихся устройств, построением биологических ДНК-машин, исследованием биохимии клеток, стыковкой наноустройств с цифровой аппаратурой. К этой же области относится теория аморфных вычислений, состоящая в использовании сетей химических реакторов как вычислителей, где аналоговый результат формируется в ходе взаимодействия химических процессов. Перспективы подобных исследований видятся в конструировании с помощью нанотехнологий молекулярных компьютеров [6].

Естественным дополнением направления «Искусственная жизнь» является **«Адаптивное поведение»**, основной подход которого состоит в конструировании и исследовании искусственных (в виде компьютерной программы или робота) организмов, способных приспосабливаться к внешней среде. Такие организмы стали называться «АНИМАТАМИ» (ANIMAl + roboT)[33]. При этом акцент здесь делается именно на конструировании (изобретении) исследователем структуры аниматов, обеспечивающей им «интеллектуальное» поведение и возможность выживания, развития в динамически изменяющейся внешней среде. Важно, что специфика внутренней организации анимата должна повторять обычно с некоторыми изменениями способности реальных животных. Так, одним из способов реализации эффекта восприятия окружающего мира является моделирование принципов эхолокации, используемых такими животными, как дельфин, летучая мышь.

Для того чтобы поведение анимата было не случайным, а носило направленный и в то же время естественный без навязывания воли автора модели характер, используется понятие мотивации. В узком смысле оно может восприниматься как fitness-функция в ЭА, пропорциональная отклонению от оптимума, соответствующего условиям гомеостаза (сочетание оптимальных параметров для жизни организма). В широком понимании мотивация выступает активной движущей силой, которая стимулирует нахождение, которое адекватно потребностям животного в данной ситуации.

Типичным примером моделирования адаптивного поведения является использование анимата для решения навигационной задачи, согласно которой он должен выработать стратегию обхода препятствий в некотором пространстве и достичь заданной цели. При этом анимат не обладает априорной информацией о расположении препятствий и план своих действий вырабатывает динамически на основе поступающей из внешней среды информации.

Общность рассмотренных направлений «Искусственная жизнь» и «Адаптивное поведение» позволяет воспринимать их в качестве одного из главных инструментов моделирования и исследования бионических систем. По этим направлениям регулярно проводятся конференции («International Conference Artificial Life», «European Conference Artificial Life», «Simulation of Adaptive Behavior»), издаются журналы («Artificial Life», «Adaptive Behavior»).

Идеи применения законов развития биологических систем в качестве эвристик для проектирования технических систем (ТС) появились относительно недавно и связаны как с получением оптимальных форм конструкций, так и с синтезом принципиально новых решений [31]. Действительно, если проанализировать структуру, а также особенности функционирования окружающих нас биологических организмов, то нетрудно увидеть оптимальность их форм, возникшую за долгий период эволюции. Кроме того, многие изобретения человека копируют природные аналоги, например насос, клапан, система теплообмена, оптическая линза, сонар и др.

В настоящее время возможности эволюционного моделирования определили направление в области автоматизации проектирования и разработки интеллектуальных САПР, известное как **эволюционное проектирование** [48] (рис. 1.12). Эволюционное проектирование основывается на применении методов ЭМ к задачам САПР и преследует две основные цели. Во-первых, это оптимизация технических решений средствами ЭА, во-вторых, синтез принципиально новых концепций систем вследствие способности ЭА к генерации различных вариантов конструкций ТС. В последнем случае речь идет об автоматизации структурного синтеза, в качестве исходных данных для ЭА здесь используются возможные прототипы, компоненты будущей конструкции, описание которых формализуется в иерархической форме или в виде морфологических таблиц, а также набор эвристик.

Рис. 1.12. Место эволюционного проектирования среди научных дисциплин

Для примера ЭП здесь можно упомянуть исследования ученых университетского колледжа Лондона, которые успешно применяли ЭА для совершенствования конструкции гоночных машин «Формула 1» [6]. В процессе проектирования компонентов машин их компьютерные модели развивались в процессе искусственной эволюции и постоянно проходили испытания на двух виртуальных маршрутах, пока лучшая модель не побила рекорды трасс.

На рис. 1.13 показан другой пример эволюционного проектирования ТС, в качестве которой выступает стол. Причем эволюционное моделирование используется здесь скорее не для оптимизации параметров стола, а для генерации различных вариантов его прототипов, которая состоит в комбинировании эвристик различных сочетаний компонентов стола. Здесь используется главный принцип эволюционного проектирования: вместо моделирования технической системы в уже готовом виде следует заниматься исследованием эволюции образующих ее простых объектов.

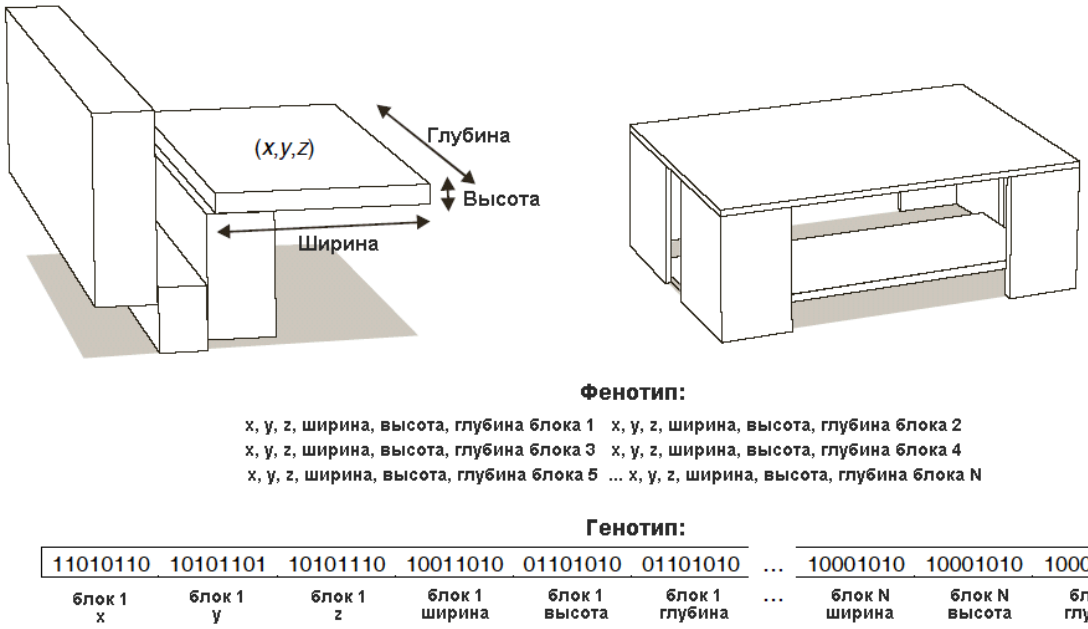


Рис. 1.13. Пример эволюционного проектирования

В качестве fitness-функции выступают требования удовлетворения заданным геометрическим критериям, а также максимизация устойчивости стола. Что касается объектов, из которых в ходе эволюционного проектирования формируется ТС, то для их описания могут использоваться не отдельные числовые параметры, а целые структуры данных, которые особым образом кодируются в генотипе. При этом динамика процесса поиска структуры ТС визуализируется и пользователь сам, играя роль fitness-функции, может оценивать и интерактивно назначать ее значения членам популяции, например на основе эстетических соображений (рис. 1.14). Конечный результат эволюционного проектирования приведен на рис. 1.13.

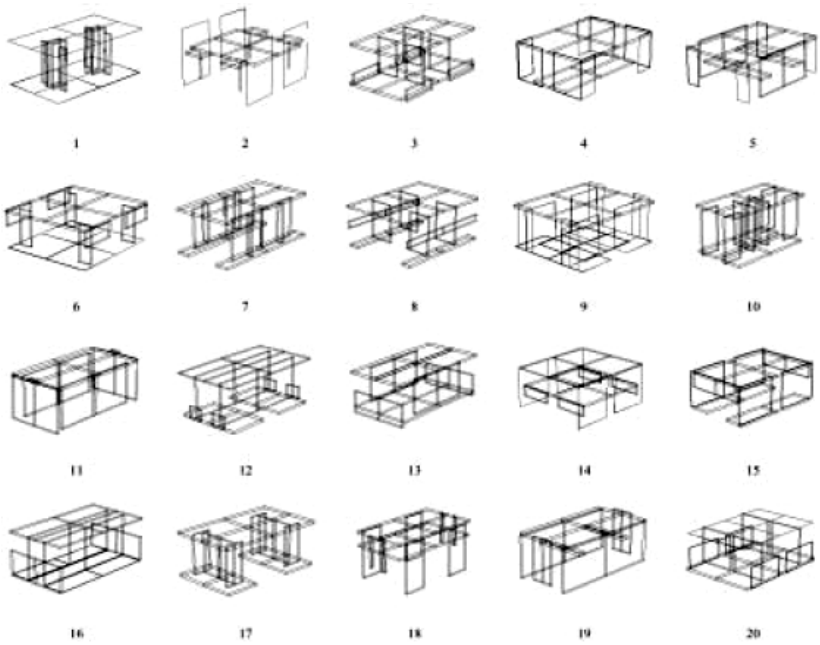


Рис. 1.14. Промежуточные варианты конструкции ТС при ЭП

В целом, ЭП позволяет во многом компьютеризировать решение задач инженерного творчества и как следствие известную тайну «Всеобщего искусства изобретения» Лейбница, согласно которой одной из двух частей этого искусства является комбинаторика – перебор постепенно усложняющихся комбинаций исходных данных. Второй частью является эвристика – свойство догадки человека.

В связи с этим особый интерес представляет разработка и исследование модификаций ЭА для решений подобных задач самоконструирования. В частности, распространение получил генетический алгоритм для комбинирования эвристик [29, 30], позволяющий эффективно решать задачи структурного синтеза проектных решений.

Таким образом, ЭП в сочетании со средствами САПР позволит увеличить производительность и качество проектирования путем исследования множества креативных (творчески полученных) решений задачи посредством генерации этих решений для проектировщика, в том числе на основе знаний, идей и их сочетаний. При этом могут быть сгенерированы принципиально новые технические решения, для чего ранее требовалось участие человека. Все это позволяет говорить о переходе от проектирования автоматизированного к автоматическому.

ГЛАВА 2. МЕТОДЫ ЭВОЛЮЦИОННОГО МОДЕЛИРОВАНИЯ

Глава посвящена описанию эволюционных алгоритмов. Основное внимание уделяется главному методу эволюционного моделирования – генетическому алгоритму. Рассматриваются его свойства, структура, операторы, модификации, а также особенности настройки и мониторинга функционирования. Представлены особенности и принципы реализации других эволюционных алгоритмов – генетического программирования, эволюционных стратегий и эволюционного программирования. В конце главы приводится и описывается структура обобщенного эволюционного алгоритма.

2.1. Генетический алгоритм

Генетический алгоритм (ГА) является наиболее известным среди других ЭА, благодаря широкому спектру областей его применения, а также доступностью его компьютерной реализации. Началом истории его создания принято считать опубликованную в 1975 г. работу Дж. Холланда (J. Holland) «Адаптация в природных и искусственных системах» [59], в которой ГА изначально были названы репродуктивными планами. Почти одновременно в своей диссертационной работе его аспирант К. Де Джонг (K. De Jong) [52] провел исследования теории Холланда применительно к различным математическим моделям, где доказал возможность применения законов естественной эволюции к решению оптимизационных задач. Свое настоящее название простой или стандартный генетический алгоритм получил позже в работе Д. Голдберга (D. Goldberg) [56], в которой была наглядно продемонстрирована высокая эффективность применения ГА при решении разноплановых оптимизационных задач для различных прикладных областей.

Несмотря на то, что к настоящему времени разработан целый ряд модификаций ГА, в основе их лежит принцип работы стандартного ГА.

2.1.1. Представление информации генетического алгоритма

Математически стандартный генетический алгоритм можно определить как метод стохастической оптимизации для задач дискретной оптимизации вида: минимизировать *f(C)* при условии, что $C \in \Omega = \{0,1\}^N$, где $f : \Omega \to R$ представляет fitness-функцию;

$C : \Omega \to R$ - N - мерный двоичный вектор из дискретного множества Ω, который называется хромосомой длины N;

множество $\Omega = \{0,1\}^N$ представляет собой множество вершин N – мерного гиперкуба с ребром, равным 1;

$R = (-\infty; +\infty)$ - множество действительных чисел.

Как отмечалось ранее, ГА для поиска решений использует закодированное в виде хромосом представление значений параметров – генотип. Наиболее простым способом реализовать это является использование прямого двоичного кода. В этом случае хромосому C можно рассматривать как бинарную последовательность длиной $L(C)$, описывающую одно решение и состоящую из n числа генов g, каждый из которых представляет двоичный код длиной $L(g)$ соответствующего параметра задачи. На рис. 2.1 показано преобразование координат точки $x = \{x_1, x_2\}$ пространства решений в хромосомное представление ГА.

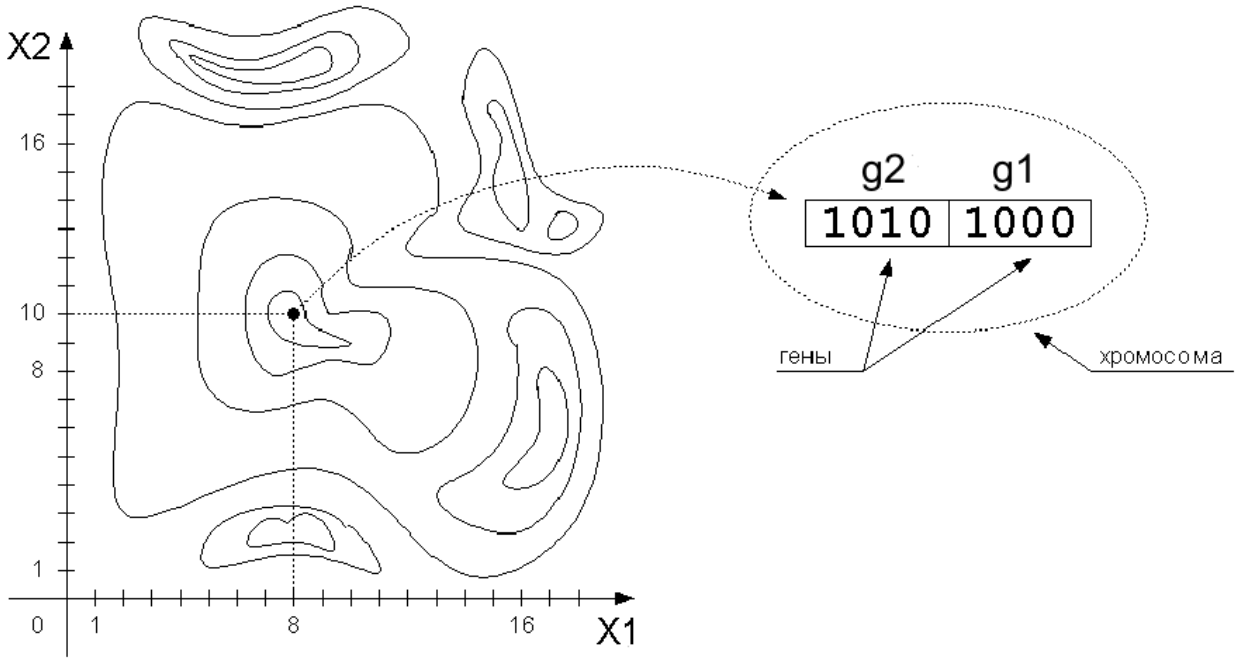


Рис. 2.1. Представление точки пространства решений в виде хромосомы

В отличие от кодирования целого числа, для бинарного представления вещественного значения используется другой подход. Он состоит в том, что интервал, в пределах которого лежит значение параметра, разбивается на $2^{L(g)}$ отрезков, каждому из которых соответствует определенное значение гена. Для его декодирования из бинарного в вещественное десятичное значение используется следующее выражение:

$$x_j^{(r)} = a + \frac{x_j^{(r)}}{2^{L(g)} - 1}(b - a),$$

где a, b – границы интервала [a, b], включающего в себя значения декодируемого параметра;

$x_j^{(r)}$ - вещественное значение параметра x_j;

$x_j^{(d)}$ - десятичное представление закодированного параметра x_j.

Для примера рассмотрим процесс восстановления вещественного значения параметра по его бинарному коду размером 8 бит на интервале [0, 1]. При двоичном представлении гена 00100101 его десятичное и вещественное декодированные значения соответственно составят $x_j^{(d)} = 37$, $x_j^{(r)} = 0{,}145$.

Размер генов определяет число возможных значений целочисленный параметров и точность представления вещественный параметров и должен удовлетворять условию

$$L(g) \geq \log_2\left(\frac{x_{\max} - x_{\min}}{\Delta}\right),$$

где $x_{\max}$ и $x_{\min}$ - максимальное и минимальное значения параметра x;

Δ- заданная погрешность определения оптимального значения x.

На рис. 2.2 приведен пример кодирования параметров задачи для случая оптимизации функции трех переменных $f = f(x_1, x_2, x_3)$. Здесь каждый из трех параметров кодируется в виде гена с $L(g) = 5$, в результате чего общая длина хромосомы составит 15 бит.

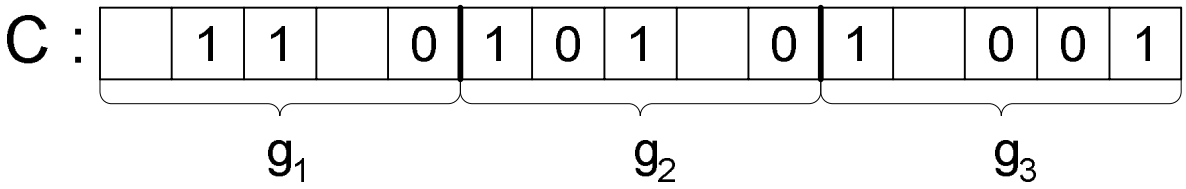


Рис. 2.2. Двоичное кодирование вектора параметров

Для повышения эффективности работы ГА, кроме обычной двоичной кодировки, для кодового представления параметров может использоваться код Грея (первые 16 слов этого кода для $N = 4$ приведены в табл. 2.1).

Таблица 2.1

Сравнение двоичного кода и кода Грея

Целое	**Двоичный код**	**Код Грея**
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Математически взаимное преобразование между бинарным и кодом Грея длины N может быть записано следующим образом:

$$\begin{cases} \Gamma(C[k]) = C[k+1] \oplus C[k]; \\ C[k] = C[k+1] \oplus \Gamma(C[k]); \\ \Gamma(C[N-1]) = C[N-1]). \end{cases}$$

Здесь $k \in [0..N-1]$ - индекс бита в хромосоме C;

$\Gamma(C[k])$ – преобразование k-го бита хромосомы к коду Грея.

Отличительной чертой кода Грея является его высокая помехозащищенность, благодаря непрерывности бинарной комбинации - изменение кодируемого числа на единицу соответствует изменению кодовой комбинации только в одном разряде. Следовательно, в отличие от двоичного код Грея гарантирует, что две соседние кодовые комбинации всегда могут быть декодированы в две ближайшие точки пространства целых (вещественных) чисел.

В ряде задач, наряду с двоичной кодировкой, могут применяться и альтернативные ей целочисленная или символьная и вещественная [46], в соответствии с которыми генотип представляется в естественном виде как набор десятичных чисел (целых или вещественных) или символов (рис. 2.3).

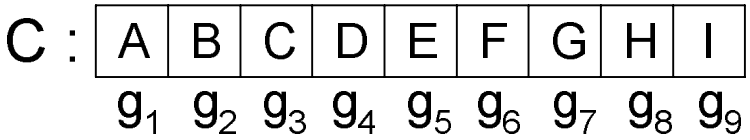


Рис. 2.3. Символьное представление хромосомы

Целочисленная или символьная форма в основном используются при решении комбинаторных задач оптимизации (коммивояжера, составления расписаний и т.д.), а хромосома здесь соответствует одному из вариантов перестановок элементов. Вещественное представление генов больше свойственно другим эволюционным алгоритмам (эволюционные стратегии, эволюционное программирование). В ГА попытка использовать подобную форму кодирования связана с необходимостью решения оптимизационных задач в непрерывных пространствах, характеризующихся большой размерностью, областью определения переменных, а также потребностью в достижении высокой точности. В последствии ГА, в которых каждый ген содержит вещественное значение, получили название непрерывных.

Основным достоинством альтернативных способов кодирования хромосом является совпадение пространств поиска и решений и, следовательно, отсутствие потребности в операции кодирования/декодирования для оценки решений, что повышает скорость работы алгоритма. Экспериментальные исследования подтверждают эффективность их обоснованного использования при решении соответствующих задач. Для этого также были созданы различные модификации генетических операторов [16, 46]. С другой стороны, в [49, 56] приводятся контраргументы целесообразности применения не бинарного кодирования, связанные со снижением эффекта параллелизма при увеличении мощности алфавита кодирования вследствие снижения числа возможных шаблонов, а также с необходимостью существенной корректировки математических основ ГА, включая отказ от гипотезы строительных блоков [45].

Для того чтобы начать процедуру поиска, создается начальная популяция из Np двоичных хромосом $\{C_1, C_2, ..., C_{Np}\} \subset \Omega$, каждая из которых содержит $L(C)$ битов. Эта популяция определяет начальное поколение решений *G(0)*. Обычно такая популяция создается случайным образом, так как заранее не известно, где именно в Ω находится глобально оптимальное решение. Однако если такая информация имеется, то она используется при формировании начальной популяции.

В качестве примера формирования начальной популяции на рис. 2.4 приведена графическая интерпретация задачи минимизации функции *f(x)*, где начальная популяция *G(0)* состоит из $Np = 7$ хромосом. Каждая хромосома состоит из одного гена по числу параметров задачи и содержит одно закодированное решение, физический смысл которого определяется особенностями конкретной решаемой задачи.

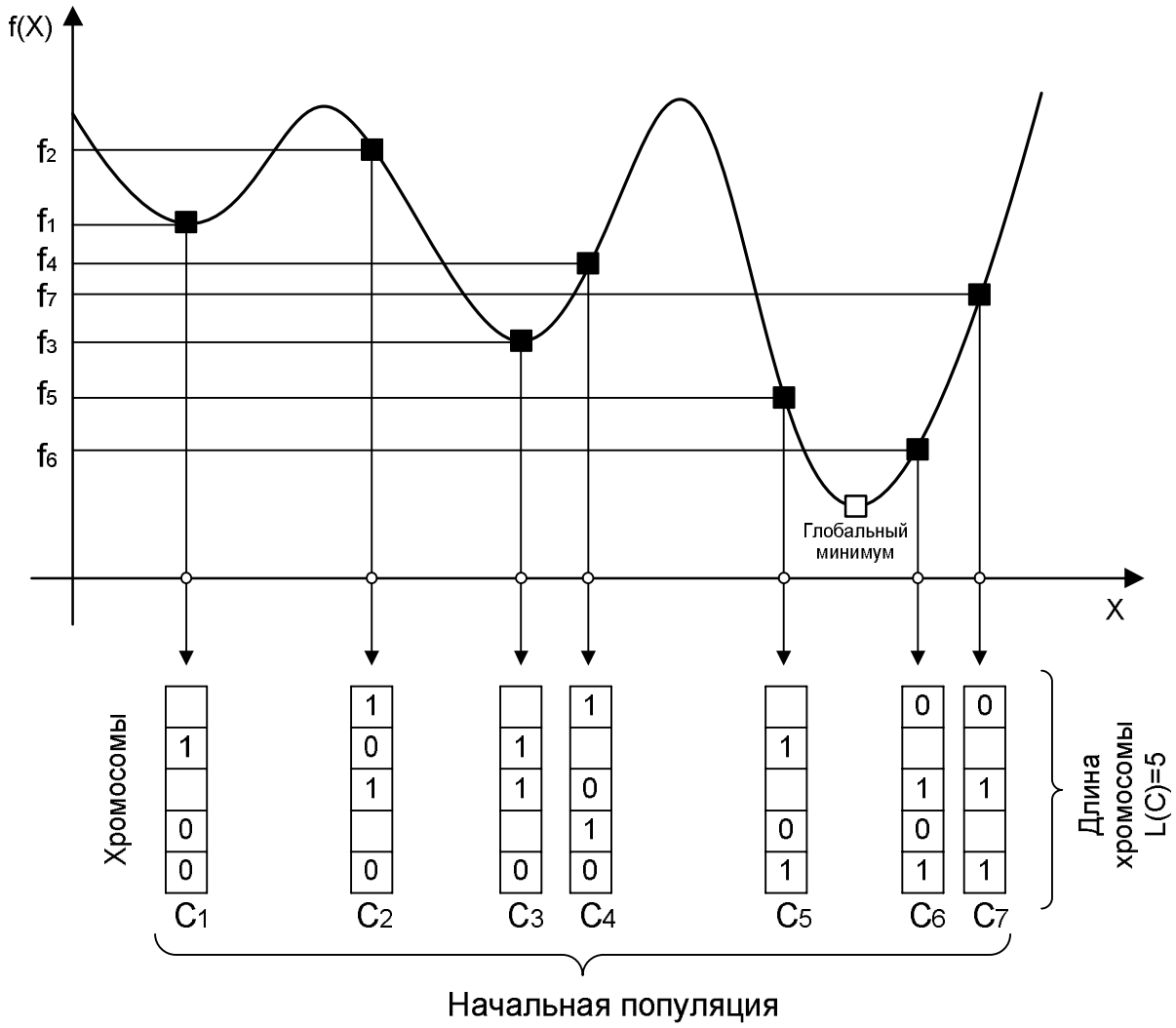


Рис. 2.4. Формирование начальной популяции решений

В соответствии с результатами экспериментальных исследований у ГА (инициализированных случайными значениями) в начале работы наблюдается быстрый прогресс в поиске оптимального решения путем комбинации и сочетания различных элементов начальной популяции. На рис. 2.5 представлена типичная кривая, получающаяся при графическом отображении лучших решений в каждой популяции текущего поколения. Из рисунка видно, что после получения ряда поколений процесс поиска сходится к популяциям, в которых наблюдается увеличение числа похожих решений (одинаковых генотипов), то есть происходит сужение области поиска с уменьшением каких – либо изменений в генотипах в ходе эволюции, пока популяция окончательно не сойдется к единому решению.

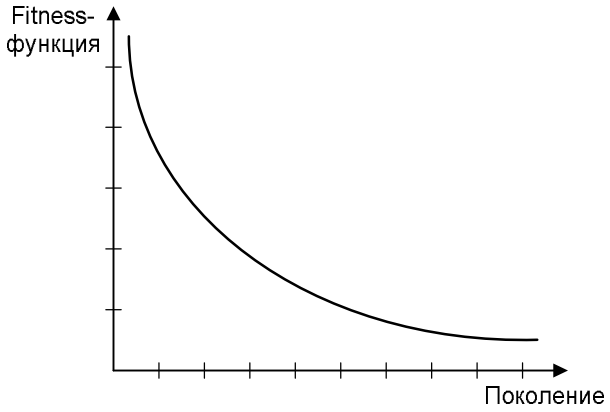


Рис. 2.5. Типовая кривая изменения fitness-функции в процессе ЭМ

Скорость сходимости ГА будет отличаться в зависимости от решаемой задачи, значений параметров ГА, используемых генетических операторов и может быть оценена аналитически с помощью выражения [49] $V = \ln \sqrt{\frac{f(C^*, t=0)}{f(C^*, t=Ng)}}$.

2.1.2. Операторы генетического алгоритма

Генетический алгоритм представляет собой итерационный алгоритм последовательного выполнения специальных операций (генетических операторов) по преобразованию данных (популяции хромосом). В общем виде порядок выполнения этих операций показан на схеме (рис. 2.6). Далее каждый элемент схемы подробно рассматривается, анализируются его особенности и роль в работе генетического алгоритма.

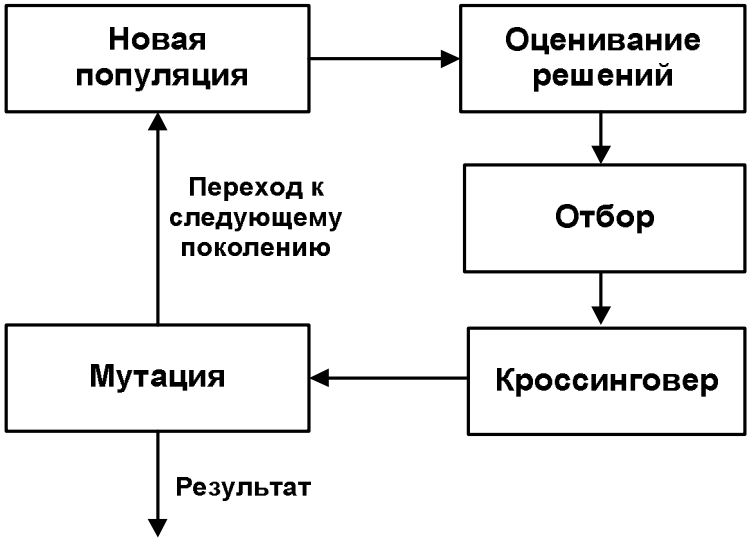


Рис. 2.6. Обобщенная схема работы стандартного ГА

Для вычисления последующих популяций к предыдущим применяются специальные генетические операторы, главными из которых являются отбор или селекция (selection), кроссинговер (crossing-over) или скрещивание и мутация (mutation).

Оператор отбора. Применяется для определения на основе значений fitness-функции хромосом-кандидатов в следующее поколение. Для реализации этого в ГА применяются различные схемы селекции, наиболее простая из которых – пропорциональный отбор, при котором число копий хромосомы в следующем поколении определяется как $Np \cdot \overline{f}(C_i)$, где $\overline{f}(C_i)$ - относительная оптимальность хромосомы C_i в популяции.

$$\overline{f}(C_i) = \frac{f(C_i)}{\sum_{j=1}^{Np} f(C_j)}.$$

При анализе оператора отбора на основе пропорциональной схемы можно сделать вывод, что только хромосомы с оптимальностью выше средней будут претендовать на «выживание» в следующих популяциях. Однако такой жестко детерминированный подход к выбору хромосом нельзя назвать полностью оправданным, потому что даже хромосомы с низкой оптимальностью могут содержать полезную информацию для поиска. В связи с этим вместо пропорционального отбора обычно используют так называемый метод «колесо рулетки», который имеет случайную составляющую, благодаря которой все хромосомы имеют шанс попасть в следующее поколение.

В этом случае хромосомы–кандидаты из *G(t)* - го поколения выбираются для выживания в следующем - *G(t+1)* – м поколении путем использования колеса рулетки. Каждая хромосома $C_i(t)$ в популяции представлена на колесе в виде сектора, ширина которого пропорциональна соответствующему значению fitness-функции. Таким образом, те хромосомы, которые имеют большую оптимальность, соответствуют большему сектору на колесе, а хромосомы с меньшей оптимальностью – наименьшему сектору колеса рулетки. Процедура отбора сводится к вращению колеса рулетки *Np* раз и принятием

в качестве кандидатов в следующем поколении тех хромосом $C_1, C_2, C_3, ..., C_{Np}$, которые будут выделены по завершении вращения. Величина $\bar{f}(C_i)$ в этом случае играет роль вероятности отбора хромосомы в следующее поколение.

В качестве иллюстрации работы оператора отбора рассмотрим пример [23] поиска максимума функции $f(x) = x^2$ на интервале [0..31]. Список членов начальной популяции, число их копий в следующем поколении, а также соответствующие им значения fitness-функции приведены в табл. 2.2. Колесо рулетки для этого примера – на рис. 2.7 (цифры на колесе указывают номера хромосом). Значения параметра *x* для упрощения закодированы в двоичном коде.

Таблица 2.2

Пример работы оператора отбора

Номер хромосомы	**Хромосома**	**Значение *x***	**Значение *f*(*x*)**	**Относительная оптимальность**	**Ожидаемое число копий в следующем поколении**	**Реальное число копий хромосомы**
1	01101	13	169	0,14	0,58	1
2	11000	24	576	0,49	1,97	2
3	01000	8	64	0,06	0,22	0
4	10011	19	361	0,31	1,23	1
Суммарное значение *f(x)*			1170	1,0	4,0	4
Среднее значение *f(x)*			293	0,25	1,00	1
Max значение *f(x)*			576	0,49	1,97	2

Чтобы вычислить следующую популяцию хромосом, колесо рулетки вращают четыре раза, что соответствует мощности начальной популяции.

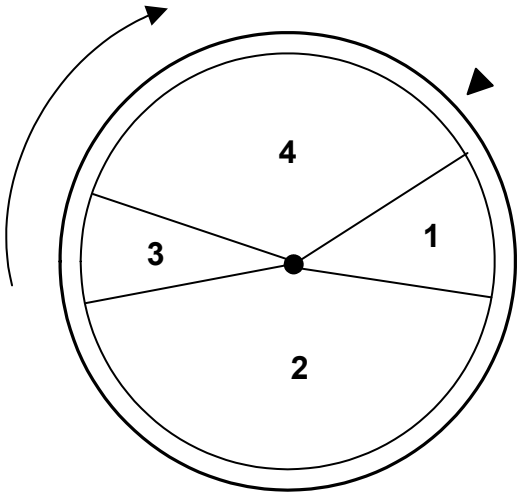


Рис. 2.7. Колесо рулетки для оператора отбора

В результате хромосомы *C1*, *C4* получают одну копию, хромосома *C2* – две копии, а хромосома *C3* не получает копий. Таким образом, «лучшие» хромосомы получают большее число копий, «средние» остаются, а плохие исключаются из следующего поколения оператором селекции. Другими словами, блокируя слабо приспособленным особям возможность стать родителем и дать потомство, увеличивается или по крайней мере не уменьшается средняя по популяции приспособленность.

Наряду с рассмотренными, часто находят применение схемы отбора на основе ранжирования членов популяции (Ranking Selection), а также турнирный отбор (Tournament Selection).

В общем случае отбор по ранжированию проводится на основании вероятностей $P_i(C_i)$, вычисленных для каждого объекта популяции. Если считать, что все хромосомы проранжированы в соответствии со значениями fitness-функции, то есть расположены в порядке убывания $f(C_i)$: $f(C_1) \geq f(C_2) \geq ... \geq f(C_{Np})$, тогда i – это ранг (rank) хромосомы в популяции.

Линейное ранжирование (linear ranking)

$$P(C_i) = \frac{1}{Np}\left[\psi_{\max} - (\psi_{\max} - \psi_{\min}) \cdot \frac{i-1}{Np-1}\right],$$

где $\psi_{\min} = 2 - \psi_{\max}$ и $1 \leq \psi_{\max} \leq 2$.

Равномерное ранжирование (uniform ranking)

$$P(C_i) = \begin{cases} 1/c, \ 1 \le i \le c, \\ 0, \ c \le i \le Np; \end{cases}$$

где c - некоторое фиксированное число первых членов популяции.

Согласно методам отбора, основанным на ранжировании степени приспособленности, популяция представляет собой отсортированную последовательность, в которой хромосома с лучшим значением fitness-функции имеет самый высокий ранг. В табл. 2.3 приведена популяция хромосом, проранжированных в процентном соотношении от размера популяции.

Таблица 2.3

Выполнение оператора отбора
на основе схемы ранжирования

Ранг	1	2	…	Np-1	Np
% от Np	20%	10%	…	0,1%	0%

При турнирном отборе популяция перетасовывается и затем разделяется на группы по N_T элементов, из которых лучшая хромосома выбирается кандидатом в следующую популяцию. Этот процесс повторяется N_T раз, потому что на каждой итерации отбирается только Np/N_T потенциальных родителей. Особенностью такой схемы отбора является гарантированное наличие копий лучших хромосом среди родителей следующего поколения.

Чаще всего в ГА применяется именно рулеточный отбор, однако с его использованием связана проблема зависимости от больших положительных значений, когда простое добавление очень большой константы к целевой функции может уничтожить отбор, приводя к простому случайному выбору.

Для ее устранения используется либо масштабирование оценок оптимальности родителей относительно наименьшего значения оптимальности в популяции, либо используется ранжирование в пропорциональной схеме отбора. Масштабирование предназначено для предотвращения доминирования какого-либо элемента популяции над остальными на ранних стадиях эволюции и для расширения диапазона значений оптимальности на финальных стадиях эволюции,

когда разнообразие в популяции существенно снижается. Среди данного класса методов можно выделить линейное динамическое масштабирование, логарифмическое масштабирование, экспоненциальное масштабирование, а также динамическое параметрическое перекодирование, которое реализуется следующим образом: когда наблюдается сходимость популяции, максимальные и минимальные значения диапазона пересчитываются для меньшего интервала, и процесс повторяется далее. Этим достигается динамическое изменение диапазона решений при приближении к оптимуму.

Перечисленные схемы отбора обладают одним общим недостатком - наилучшая хромосома в популяции может быть потеряна в любом поколении и нет гарантии, что результаты эволюции, достигнутые в ряде поколений, не будут утрачены.

Одним из способов преодоления данной проблемы является использование элитного отбора или так называемой стратегии элитизма, которая гарантирует сохранение лучшей хромосомы популяции в следующем поколении.

Выбранные таким образом хромосомы являются только кандидатами в следующую популяцию, в которую впоследствии копируются их потомки в виде точной копии родителей, либо с отличиями от них вследствие применения генетических операторов.

Оператор кроссинговера. Применяется по отношению к паре хромосом из *G(t)*, прошедших отбор и предназначен для обмена с заданной вероятностью генетическим материалом между хромосомами–родителями, полученными в результате отбора, с целью генерации хромосом-потомков для следующего поколения.

Для этого назначается вероятность выполнения кроссинговера *Pc*. Затем для случайно выбранной пары хромосом определяется случайное число $k \in \{1,2,...,L(C)-1\}$, называемое местом (сайтом) кроссинговера, и затем биты из двух выбранных хромосом меняются местами после *k* – го бита с вероятностью *Pc*. Этот процесс повторяется для остальных хромосом до тех пор, пока не будет сформирована следующая популяция *G(t+1)*. На рис. 2.8 графически представлены действия оператора кроссинговера для двух 5 – ти битовых хромосом.

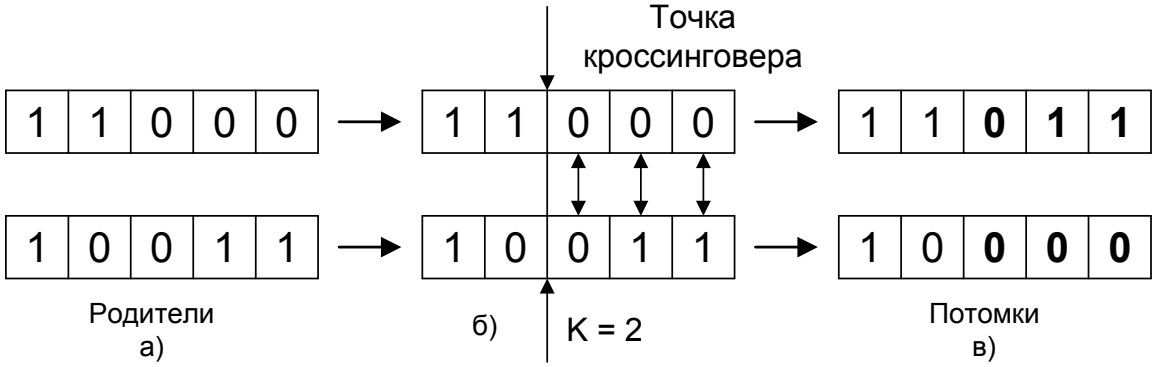


Рис. 2.8. Графическая интерпретация оператора кроссинговера

Как видно из рисунка, две родительские хромосомы (а) вступают в кроссинговер для выбранного сайта *k=3* (б) так, что хромосомы потомков (в) отличаются от родительских хромосом значениями 3 – го - 5 – го битов.

В продолжение рассматриваемого примера оптимизации квадратичной функции, приведена табл. 2.4, где показаны результаты выполнения кроссинговера.

Таблица 2.4

Пример работы оператора кроссинговера

Номер хромосомы	**Хромосомы после отбора**	**Номера случайно выбранных пар**	**Сайт кроссинговера**	**Хромосомы после кроссинговера**	**Значение *x***	**Значение *f(x)***
1	0110|1	1-2	4	01100	12	144
2	1100|0	1-2	4	11001	25	625
3	11|000	3-4	2	11011	27	729
4	10|011	3-4	2	10000	16	256
Суммарное значение *f(x)*			1754			
Среднее значение *f(x)*			439			
Max значение *f(x)*			729			

Как видно из таблицы, в столбцах три и четыре соответственно приведены номера выбранных пар хромосом и точка приложения кроссинговера. Результаты его применения расположены в столбце пять. В целом результат оператора кроссинговера привел

к улучшению всех показателей – среднего и максимального значения fitness-функции.

Естественным развитием одноточечного кроссинговера являются схемы с несколькими точками обмена информацией между членами популяции (рис. 2.9). Предельным случаем является равномерный кроссинговер, когда каждая пара битов (генов) внутри двух хромосом обмениваются битами в соответствии с определенной вероятностью. Существуют и другие модификации стандартного кроссинговера: на основе метода «золотого сечения», чисел Фибоначчи, дихотомии, а также специальные его версии для символьного и вещественного представления значений оптимизируемых параметров.

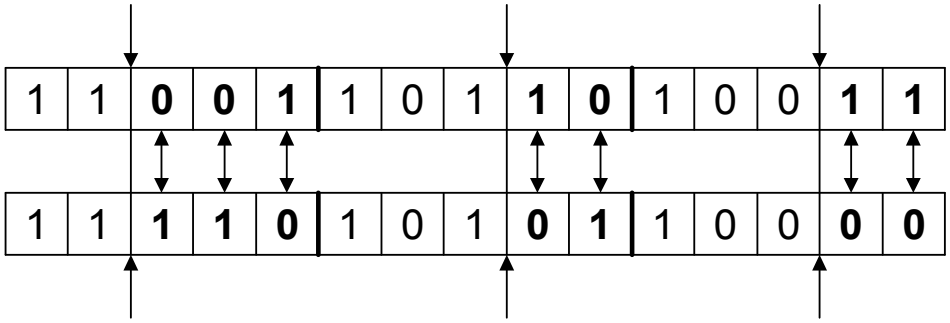


Рис. 2.9. Многоточечный кроссинговер

Так, для символьного кодирования применяются упорядоченный, частично-отображаемый, циклический и др. виды кроссинговера [16, 46], например правила выполнения упорядоченного кроссинговера заключаются в следующем [46]. Выбираются два родителя *C1*, *C2* для рекомбинации, а также две точки в них для выполнения оператора. Гены *C1*, расположенные между этими двумя точками, копируются в хромосому потомок *C1'*, занимая соответствующие позиции, совпадающие с *C1*. Затем, начиная со второй точки кроссинговера, находится следующий ген из *C2*, значение которого отсутствует в потомке. Этот ген копируется в следующий свободный слот потомка (рис. 2.10). Этот процесс повторяется до тех пор, пока *C1'* не будет полностью заполнен. При достижении конца хромосомы *C2* происходит перевод поиска к ее началу. Как следует из рисунка, особенностью этого типа кроссинговера является то, что он полностью сохраняет в потомке соседство отдельных генов первого родителя и относительное расположение генов второго родителя.

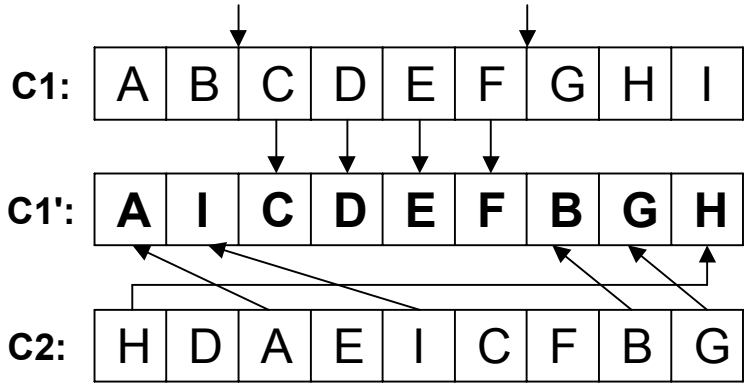


Рис. 2.10. Пример упорядоченного кроссинговера

При непрерывном ГА применяются следующие модификации кроссинговера: арифметический, геометрический, смешанный, имитирующий двоичный и др. [46]. Так при выполнении первого из них создаются два потомка С1’, С2’, значения генов которых определяются по следующим формулам:

$$g_j^{(1)\prime} = \alpha \cdot g_j^{(1)} + (1-\alpha) g_j^{(2)};$$

$$g_j^{(2)\prime} = \alpha \cdot g_j^{(2)} + (1-\alpha) g_j^{(1)},$$

где α - число в интервале [0, 1], которое может быть константой (равномерный арифметический кроссинговер) или переменной на протяжении поколений (неравномерный арифметический кроссинговер) (рис. 2.11).

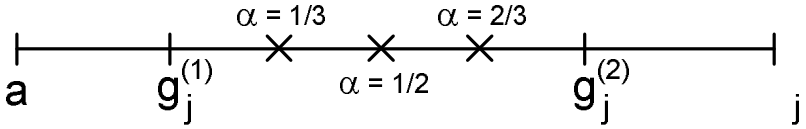


Рис. 2.11. Графическая интерпретация арифметического кроссинговера

Различные типы кроссинговера обладают общим положительным свойством – контроль баланса между дальнейшим использованием уже найденных хороших подобластей пространства поиска и исследованием новых областей. Это достигается путем сохранения общих блоков внутри хромосом - родителей и одновременного исследования новый областей в результате обмена фрагментами хромосом. С другой стороны, этот оператор можно охарактеризовать с точки зрения разрушающей способности родительских хромосом,

которая пропорциональна расстоянию по Хеммингу между ними и получившимися дочерними хромосомами. Так, одноточечный кроссинговер считается слаборазрушающим, а равномерный в большинстве случаев сильноразрушающим.

Совместное использование операторов отбора и кроссинговера приводит к тому, что области пространства, обладающие лучшей в среднем оптимальностью, содержат больше членов популяции, чем другие. Таким образом, эволюция популяции направляется к областям, содержащим оптимум с большей вероятностью, чем другие. Кроме этого, обмен генетическим материалом между двумя хромосомами придает поиску коллективный характер.

Оператор мутации. После кроссинговера к хромосомам новой популяции применяется мутация, которая в общем случае может рассматриваться как процесс перехода между различными состояниями пространства поиска. Оператор мутации состоит в случайном изменении (на противоположное) значения каждого бита с вероятностью *Pm*, то есть для каждого бита хромосомы его значение меняется с 0 на 1 или с 1 на 0 с вероятностью *Pm*.

Следуя рассматриваемому примеру, предположим, что *Pm* = 0,1 и хромосома *C= (11001)* должна подвергнуться мутации. Для определения битов в гене, которые необходимо изменить выбирается независимое случайное число $\widetilde{P}_j$ для каждого бита хромосомы. Если окажется, что *j* – й бит следует изменить, иначе значение *j* – го бита сохраняется. Пусть для рассматриваемого вектора *x* были получены следующие случайные числа: $P_1 = 0{,}91$, $P_2 = 0{,}43$, $P_3 = 0{,}03$, $P_4 = 0{,}67$, $P_5 = 0{,}29$. Тогда результатом мутации является изменение *3* – го бита хромосомы (рис. 2.12).

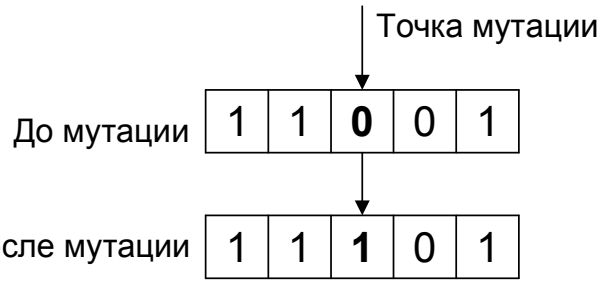


Рис. 2.12. Графическая интерпретация оператора мутации

Полученная в результате мутации хромосома имеет значение fitness-функции, равное 29^2=841, улучшив тем самым результат кроссинговера.

Что качается альтернативных форм генетической информации, то при вещественной распространение получили случайные и неравномерные мутации, при которых выбранный ген принимает случайное значение из области определения связанной с ним переменной оптимизации или приращение из интервала своего изменения. При целочисленном или символьном кодировании используются мутации на основе 2, 3, *k* – позиционных перестановок. Если оператор мутации выполняется применительно к двум случайно выбранным генам хромосомы, в результате чего они обмениваются друг с другом значениями (рис. 2.13). Если мутация применяется более, чем к двум аллелям, то значения, расположенные в данных позициях произвольно, попарно меняются, что позволяет получить *L(C)!-1* претендентов на перестановку. Порядок выбора варианта хромосомы из этого набора определяется по специальному правилу или случайно.

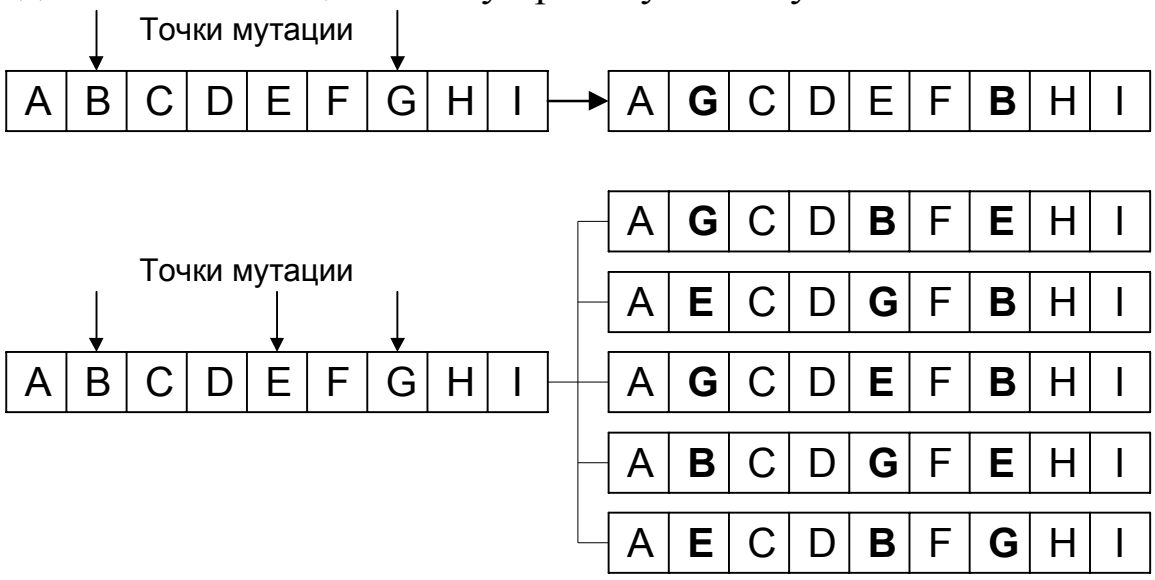


Рис. 2.13. Мутация для символьного кодирования

Оператор мутации, как и кроссинговер, вносит дополнительное разнообразие в популяцию, накладывая стохастический шум на процесс эволюции, тем самым предотвращая невосстановимую потерю в определенных позициях хромосомы ее аллелей, которые не могут быть восстановлены кроссинговером. Данное свойство мутации позволяет ограничить преждевременное сжатие пространства поиска и снизить в конечном итоге риск сходимости к локальным оптимумам.

Наряду с рассмотренными тремя основными генетическими операторами, входящими в стандартный ГА, иногда используется **оператор инверсии**. При его выполнении участок хромосомы покидает свое место в цепи хромосомы и, развернувшись на 180 градусов, вновь занимает прежнее положение (рис. 2.14). Было найдено, что для того, чтобы инверсия обеспечивала сходимость к оптимуму, вероятность ее применения должна быть очень низкой, как правило, она составляет $Pinv = 0{,}001$.

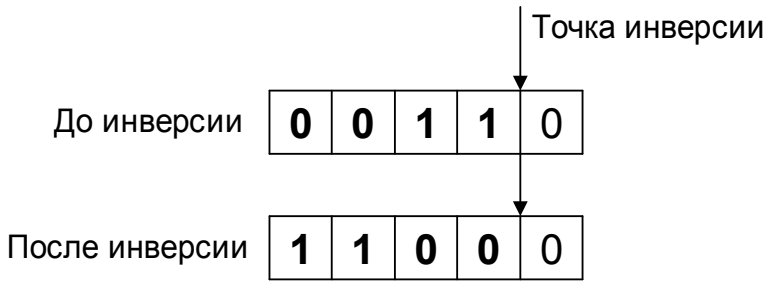


Рис. 2.14. Графическая интерпретация оператора инверсии

Таким образом, обобщая изложенное, приведем формальное описание, характеристики, а также последовательность функционирования стандартного ГА.

Формализованно ГА может быть представлен в виде следующего множества $GA = <Form, Init, Fitness, OSet, PSet>$,
где *Form* – форма представления решений задачи;
Init – процедура генерации начальной популяции решений;
Fitness – функция, позволяющая определить качество решений;
OSet – группа операторов для получения новых решений из существующих;
PSet – множество значений параметров, управляющих процессом работы ГА (вероятности выполнения генетических операторов, размер популяции, число поколений, критерии останова алгоритма и др).

Для общего случая модель стандартного генетического алгоритма имеет следующие характеристики:

- фиксированный размер популяции;
- фиксированная разрядность генов;
- пропорциональный, рулеточный или турнирный отбор;
- случайный выбор хромосом для репродукции;
- одноточечные операторы кроссинговера и мутации;
- использование или нет стратегии «элитизма».

Последовательность этапов работы ГА можно представить в виде нижеследующего псевдокода, который может быть запрограммирован на любом алгоритмическом языке.

Шаг 1: ***Инициализация начальной популяции G(t)***

```
t = 0, G(t) = ∅;
Для каждой C[i], i=1..Np
  Каждому биту C[i] присвоить случайное значение
  {0, 1};
  G(t) = G(t) + {C[i]};
```

Шаг 2: ***Вычисление fitness-функции хромосом популяции***

```
Для каждой C[i], i=1..Np
  2.1. Декодировать код хромосомы в значения
       переменных оптимизации;
  2.2. f(C) = f(X)
```

Шаг 3: ***Выполнение оператора отбора***

```
Создать промежуточную популяцию G'(t), G(t) = ∅;
Пока G'(t) < Np
  3.1. Среди подмножества C определить «лучшую» C*
  относительно выбранной схемы отбора;
  3.2. G'(t) = G'(t) + {C*};
```

Шаг 4: ***Выполнение оператора кроссинговера***

```
Сформировать популяцию G(t+1), G(t+1)= ∅;
Пока G(t+1) < Np
  Выбрать из G'(t) две хромосомы C[i], C[j];
  Если  Pc > Random(0..1)
    То получить два потомка C[k], C[l];
       G(t+1) = G(t+1) + {C[k], C[l]};
    Иначе  G(t+1) = G(t+1) + {C[i], C[j]};
```

Шаг 5: ***Выполнение оператора мутации***

```
Для каждого C[i], i=1..Np, C[i]∈G(t+1)
  Изменить с вероятностью Pm значение каждого бита
  C[i] на противоположное;
```

Шаг 6: ***Выполнение оператора инверсии***

```
Для каждого C[i], i=1..Np, C[i]∈G(t+1)
```

```
Если  Pinv > Random(0..1)
То определить точку инверсии k;
   изменить порядок следования бит с позиций
   1 до к на противоположный;
```

Шаг 7: ***Окончание работы алгоритма***

```
Если  (t > Ng) или
      (достигнут другой критерий окончания)
То конец работы алгоритма;
Иначе перейти к шагу 2;
```

Таким образом, шаги ГА можно сгруппировать в виде нескольких этапов.

1. На первом шаге случайным образом генерируется начальная популяция $G(t, t=0)$ двоичных хромосом. Каждая хромосома на основе математической модели декодируется для последующей оценки ее приспособленности.

2. К членам популяции применяется оператор отбора, на основе которого выбираются родительские хромосомы для формирования новой популяции, причем с большой вероятностью в нее переходят наиболее перспективные (с точки зрения fitness-функции) элементы. При этом для каждого родителя есть две возможности – либо быть просто скопированными в следующее поколение $G(t+1)$, либо подвергнуться воздействию генетических операторов.

3. С заданными вероятностями к родительским хромосомам применяются генетические операторы (кроссинговер, мутация, инверсия), полученные в процессе их выполнения хромосомы-потомки копируются в следующее поколение $G(t+1)$. Данный процесс повторяется до заполнения популяции следующего поколения.

4. Работа алгоритма прекращается, если выполнены условия останова эволюции. С помощью декодирования лучшей хромосомы в популяции может быть определено оптимальное решение задачи. При невыполнении условий окончания ГА повторяются этапы оценки и формирования новой популяции $G(t, t=t+1)$.

Критерий остановки эволюции может быть определен произвольным образом, например получение удовлетворительного решения, достижение определенного числа поколений, количество затраченного времени, низкая скорость сходимости и т.п.

Завершающая стадия работы ГА может быть определена по стягиванию ядра популяции сначала в плотное облачко, а затем – в точку. Кроссинговер в таких условиях не оказывает влияния на разнообразие популяции, поскольку при скрещивании идентичных родителей потомок ничем не будет от них отличаться. Мутация и инверсия будут по-прежнему модифицировать потомство, порождая новые точки поискового пространства, но наличие уже найденного лучшего решения не позволит потомкам закрепиться в следующем поколении.

Здесь важно учитывать тот факт, что эволюционные методы вычислений относятся к так называемым гибким алгоритмам [68] (anytime-algorithms), качество работы которых может улучшаться с увеличением времени работы алгоритма.

2.1.3. Фундаментальная теорема генетического алгоритма

Является основной теоремой эволюционного моделирования и связана с математическим обоснованием принципов генетического поиска оптимальных решений и его эффективности как альтернативной технологии оптимизации.

Для анализа процесса функционирования ГА используется понятие шаблона или схемы [23]. Согласно Д. Холланду шаблон описывает подмножество хромосом, имеющих одинаковые значения в некоторых позициях и определяется с помощью элементов множества

$$H \in \{0,1,*\}^{L(C)},$$

где $L(C)$ – длина хромосомы в битах;

* – означает, что в данной позиции может быть любой бит.

В общем случае шаблоны – это гиперплоскости различной размерности в *L(C)* – мерном пространстве (рис. 2.15). Для количественной оценки шаблонов используются такие характеристики, как порядок $\delta(H)$ шаблона - определяется числом зафиксированных позиций (которые равны 0 или 1) и длина $L(H)$ шаблона – расстояние между первой и последней зафиксированной позицией.

В качестве примера [23] рассмотрим шаблон (*111*), который описывает подмножество с 4 членами {01110, 11110, 01111, 11111}. Шаблон (0*1**) будет иметь 8 двоичных хромосом длины 5, то есть в общем случае в хромосоме с $L(C) = 5$ можно определить $3^5 = 243$ шаблона, однако на самом деле число различных хромосом, которые можно получить, составляет только $2^5 = 32$. Это следует из того, что шаблон (**) определяет в общем $3^2 = 9$ двоичных строк {**, *1, *0, 1*, 0*, 00, 01, 10, 11} и из этих девяти строк используется в хромосоме только четыре {00, 01, 10, 11}.

Возвращаясь к рассматриваемому примеру максимизации функции, для нахождения оптимального значения в популяции должны быть две хромосомы с шаблонами 11*** и **111. Тогда, применяя оператор кроссинговера к ним, можно получить хромосому 11111 с наилучшим значением ЦФ.

Таким образом, в процессе своего функционирования ГА обрабатывает не двоичные строки, а шаблоны, и может производить выборку значительного числа гиперплоскостей из областей с высокой оптимальностью. В течение одного поколения популяции оценивается $O(Np^3)$ структур, хотя популяция содержит только *Np* объектов. Этот эффект называется неявным параллелизмом и его следствием является то, что простое увеличение числа объектов популяции экспоненциально ускоряет решение задачи. Эта особенность генетического алгоритма выгодно выделяет его среди детерминированных, стохастических, эвристических методов оптимизации, в которых единственное решение развивается само по себе, не используя информации (генетической памяти) о предыдущих вариантах решений.

В работе [23] указывается, что число шаблонов, которое может быть получено из популяции размером *Np* с длиной хромосомы *L(C)* лежит, в интервале $[2^{L(C)}, Np \cdot 2^{L(C)}]$. Также отмечается, что число эффективных шаблонов (порождающих потенциально «хорошие» решения) составляет ориентировочную величину Np^3.

На основе шаблонов механизм работы ГА может быть проиллюстрирован следующим примером. Пусть необходимо найти оптимум некоторой функции, топология которой представлена на рис. 2.15.

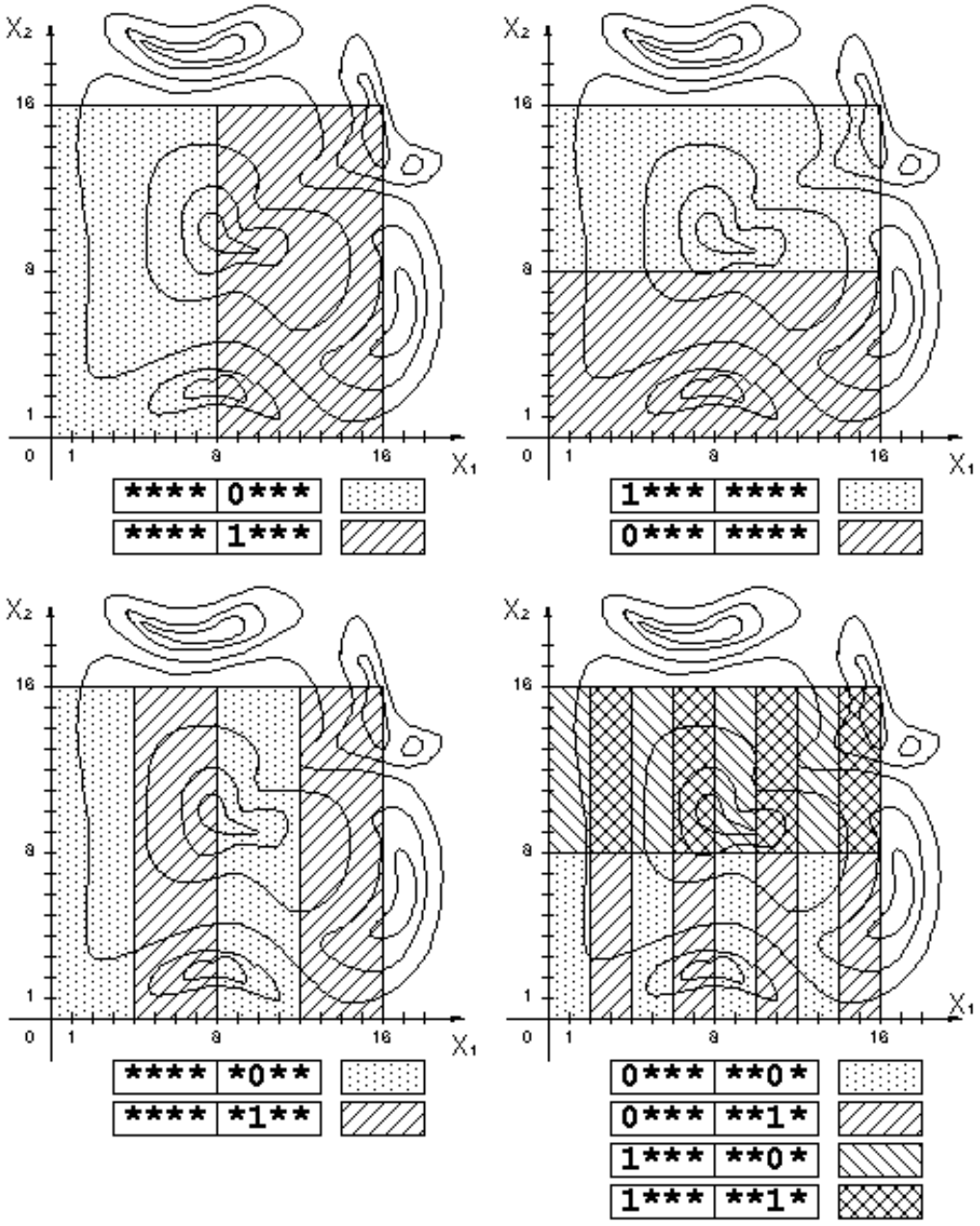


Рис. 2.15. Разделение шаблонами пространства поиска на гиперплоскости

Гиперплоскость (****0***) покрывает первую вертикальную половину пространства поиска, гиперплоскость (****1***) – вторую половину. В свою очередь, гиперплоскости (1*******) и (0*******) разделяют горизонтальные половины области поиска. Как видно из рисунка, число потенциальных оптимумов, принадлежащих области (****1***), в среднем выше, чем у других покрытий, поэтому целесообразно исследовать данную гиперплоскость более детально, однако с учетом особенностей рельефа функции глобальный оптимум может и не принадлежать этой области.

Поэтому для локализации оптимума ГА использует свойство неявного параллелизма, что позволяет исследовать все фрагменты пространства поиска путем выделения подобластей, содержащих потенциально «хорошие» решения (третий, четвертый фрагменты рис. 2.15). Таким образом, как видно из рисунка, благодаря использованию механизма шаблонов генетический алгоритм имеет возможность исследовать все пространство поиска, динамически локализуя и уточняя подобласти с более высокой по сравнению с другими оптимальностью, то есть хромосомы, содержащие шаблон, описывающий части пространства поиска с потенциальными оптимумами, будут увеличивать число своих копий в следующих поколениях. В результате можно сделать вывод о том, что с помощью ГА всегда можно найти решение, которое будет относительно эффективней по сравнению с альтернативными. Для локализации же генетическим алгоритмом глобального оптимума необходимо выполнение следующих условий:

- последовательность популяций *G(0), G(1),…,G(Ng)*, генерируемая алгоритмом, монотонна, то есть

$$\forall t \in Ng : \min(f(C) \mid C \in G(t+1)) \le \min(f(C) \mid C \in G(t));$$

- $\forall C$ C^* достижима из C посредством преобразований через применение генетических операторов.

Тогда глобальный оптимум функции f отыскивается с вероятностью

$$\lim_{t\to\infty} G(C^* \in G(t)) = 1.$$

Математической основой теоретического анализа процесса функционирования ГА являются зависимости, полученные Д. Холландом. Пусть *H* обозначает определенный шаблон, который присутствует в популяции *G(t)* и *N(H(t))* - число хромосом популяции *G(t)*, которые являются элементами *H*. Тогда может быть записано следующее выражение:

$$N(H(t+1)) = N(H(t)) \cdot \frac{\overline{f}(H(t))}{\overline{f}(t)}, \qquad (2.1)$$

где $\overline{f}(t)$ - среднее значение оптимальности в поколении *t*;

$\overline{f}(H(t))$ – средняя оптимальность хромосом, содержащих шаблон *H(t)* в поколении *t*.

$$\overline{f}(H(t)) = \frac{1}{N(H(t))} \sum_{i=1; C_i \in H(t)}^{N(H(t))} f(C_i).$$

Из (2.1) следует, что число хромосом, содержащих шаблон *H(t)* в следующем поколении, увеличивается как отношение целевой функции шаблона к средней целевой функции популяции. Шаблон с ЦФ выше средней будет получать больше возможности для копирования и наоборот. Правило репродукции Д. Холланда : «схема с целевой функцией выше средней живет, копируется, а с ниже средней целевой функцией умирает» [23].

Если предположить, что средняя оптимальность *H(t)* выше средней по популяции, то есть

$$\overline{f}(H(t)) = \overline{f}(t) + c \cdot \overline{f}(t),$$

где $c > 0$ является некоторой константой, тогда выражение (2.1) можно модифицировать к виду следующих равенств:

$$N(H(t+1)) = N(H(t)) \frac{\overline{f}(t) + c \cdot \overline{f}(t)}{\overline{f}(t)} = N(H(t))(1+c);$$

после *t* поколений, начиная с $t = 0$:

$$N(H(t)) = N(H(0))(1+c)^t$$

Копирование в новую популяцию без изменения шаблонов хромосом недостаточно для эффективного генетического поиска.

Поэтому, добавив в анализ генетические операторы, рассмотрим для примера хромосому с $L(C) = 7$ и два шаблона, представляющих эту хромосому [23]:

$C =$ 011|1000;

$H1 =$ *1*|***0;

$H2 =$ ***|10**.

Здесь символ «|» обозначает точку кроссинговера. Шаблон *H1* после выполнения оператора кроссинговера вероятней всего будет разрушен, так как значения 1 в позиции 2 и 0 в позиции 7 расположатся в различных дочерних хромосомах. Напротив, шаблон *H2* будет сохранен, поскольку после кроссинговера значение 1 в позиции 4 и 0 в позиции 5 останутся на прежнем месте в новых хромосомах. Несмотря на то, что сайт кроссинговера выбран случайно, шаблон *H1* все равно останется менее приспособленным к выживанию, чем *H2*. Действительно, если точка кроссинговера выбирается случайно среди $L(C)$-1 = 7-1 = 6 возможных позиций, то шаблон *H1* уничтожается с вероятностью $P(H1, d) = L(H1)/(L(C)-1) = 5/6$ и выживает с вероятностью $P(H1, s) = L(H1)/(L(C)-1) = 1/6$. Аналогично шаблон *H2* имеет длину $L(H2) = 1$ и вероятность его разрушения $P(H1, d) = 1/6$, а сохранения - $P(H1, s) = 5/6$.

Таким образом, при выполнении оператора кроссинговера с вероятностью *Pc*, шаблон сохраняется в следующем поколении с вероятностью

$$P(H,s) \geq 1 - \frac{Pc \cdot L(H)}{\mathrm{L(C)} - 1}. \qquad (2.2)$$

Из (2.2) следует, что вероятность выживания шаблона уменьшается с $Pc \to 1$.

С учетом полученных зависимостей (2.1, 2.2) можно оценить число шаблонов в новом поколении с учетом независимого выполнения операторов отбора и кроссинговера:

$$N(H(t+1)) \geq N(H(t)) \frac{\overline{f}(H(t))}{\overline{f}(t)} \left(1 - Pc \frac{L(H)}{\mathrm{L(C)} - 1}\right). \qquad (2.3)$$

Как видно из выражения (2.3), число шаблонов зависит от значения целевой функции и их длины, а именно шаблоны с ЦФ выше

средней и короткой длинной имеют возможность экспоненциального роста их количества в следующем поколении. Наличие знака неравенства в выражении связано с тем, что генетические операторы могут не только разрушать шаблон, но и создавать его из других шаблонов.

Для появления шаблона в новом поколении необходимо, чтобы при выполнении генетических операторов были сохранены все его специфические позиции. Рассмотрим влияние на этот процесс оператора мутации.

Вероятность сохранения хромосомы после оператора мутации определяется как (1 - *Pm*), следовательно, шаблон сохраняется, когда не изменяется каждая из его $\delta(H)$ закрепленных позиций. Вероятность этого может быть вычислена как $(1-Pm)^{\delta(H)}$, для малых *Pm* (*Pm*<<1) определение вероятности сохранения шаблона после оператора мутации может быть сведено к выражению

$$Pm(H,s) = 1 - \delta(H)Pm \ . \qquad (2.4)$$

Тогда с учетом выражения (2.4) может быть определено ожидаемое число копий шаблона *H* в следующем поколении (*t*+1) после операторов отбора, кроссинговера и мутации:

$$N(H(t+1)) \geq N(H(t))\frac{\overline{f}(H(t))}{\overline{f}(t)}\left(1 - Pc\frac{L(H)}{\mathrm{L(C)}-1} - \delta(H)\cdot Pm\right). \qquad (2.5)$$

Выражение (2.5) называется **фундаментальной теоремой о шаблонах** [56, 59] Дж. Холланда и означает, что короткие (по отношению к кроссинговеру), низкого порядка (по отношению к мутациям), имеющие оптимальность выше средней по популяции шаблоны (хромосомы) увеличивают свое представление в последовательности поколений экспоненциально.

Для проверки правильности фундаментальной теоремы при аналитической оценке числа шаблонов в следующей популяции воспользуемся результатами рассмотренного ранее примера максимизации квадратной функции.

Кроме исходной информации данной задачи, пусть имеются три шаблона: *H1* = 1****, *H2* = *10** и *H3* = 1***0, описанные в табл. 2.5.

Таблица 2.5

Описание шаблонов перед выполнением генетических операторов

Название шаблона	Шаблон	Хромосомы, содержащие шаблон	Средняя оптимальность шаблона
H1	1****	2, 4	469
H2	*10**	2, 3	320
H3	1***0	2	576

Как следует из табл. 2.5, изначально представителями шаблона *H1* являются хромосомы 2 и 4. После оператора отбора число копий этого шаблона в популяции увеличилось до трех, так как согласно табл. 2.4 две копии хромосомы 2 из табл. 2.2 попали в репродукционную группу.

В табл. 2.6 представлено измененное число шаблонов в новой популяции вследствие выполнения генетических операторов.

Таблица 2.6

Описание шаблонов после выполнения генетических операторов

Название шаблона	После отбора			После генетических операторов		
	Расчетное количество	Действительное количество	Хромосомы, содержащие шаблон	Расчетное количество	Действительное количество	Хромосомы, содержащие шаблон
H1	3,20	3	2, 3, 4	3,20	3	2, 3, 4
H2	1,18	2	2, 3	1,64	2	2, 3
H3	1,97	2	2, 3	0,0	1	4

Проверим, как согласуется расчетное (на основе теоремы) число шаблонов с реальным. Согласно (2.1) после отбора у шаблона *H1* ожидается (2*(576+361)/2)/293 = 3,2 копии, что соответствует реальному значению (три копии). На основе выражения (2.2) можно определить, что шаблон *H1* не может быть разрушен в процессе кроссинговера. Аналогично применение оператора мутации с низкой вероятностью (*Pm* = 0,01) имеет очень малые шансы разрушить этот

шаблон. Таким образом, согласно (2.5) шаблон *H1* экспоненциально увеличивает свое представительство в следующем поколении, что полностью соответствует ранее полученным результатам.

Подобным образом могут быть проанализированы шаблоны *H2* и *H3*. Отметим, что шаблон *H2* имеет больше шансов перейти в следующее поколение по сравнению с *H3*. Это связано с тем, что вероятность его сохранения после оператора кроссинговера превышает 0,75 (2.2), в отличие от шаблона *H3*, для которого это значение равно нулю. Однако шаблон *H3* все равно будет иметь представителей в следующем поколении (хромосома 4 после выполнения генетических операторов), так как он является подмножеством шаблона *H1*, а хромосома 4 содержит этот шаблон.

Таким образом, шаблоны порождают увеличивающееся или уменьшающееся число представителей в соответствии с их оптимальностью. Фактор роста γ выражается следующим образом:

$$\gamma = \frac{\overline{f}(H(t))}{\overline{f}(t)}\left(1 - Pc\frac{L(H)}{\mathrm{L(C)} - 1} - \delta(H)Pm\right).$$

Согласно данной теореме, гарантируется рост числа шаблонов в последовательности поколений при соблюдении трех условий:

- оптимальность шаблона выше средней по популяции;
- его длина относительно мала;
- он низкого порядка.

Когда все три условия выполняются, то данный шаблон называется строительным блоком. Таким образом, значение фактора роста (большее или меньшее 1) указывает на то, является ли данный шаблон строительным блоком. Когда $\gamma > 1$, последующие поколения будут содержать увеличивающееся число строк, содержащих данный шаблон. Новые двоичные строки будут создаваться посредством рекомбинации данного строительного блока и других строительных блоков. Как следует из анализа нижних фрагментов рис. 2.15 строительным блоком, согласно требованиям, может быть любой из представленных шаблонов, если его оптимальность выше средней по популяции. Следует отметить, что вид строительного блока должен выбираться из знаний о решаемой задаче так, чтобы в процессе работы ГА вероятность их разрушения была минимальна.

Однако возникает вопрос о необходимости в мутации и кроссинговере, если, согласно теореме, циклическое повторение отбора приводит к сходимости к оптимуму. При ответе на этот вопрос необходимо учесть, что при выводе теоремы использовалось предположение о популяции очень большого размера. Это подразумевает, что все пространство поиска покрыто элементами популяции с ненулевой вероятностью. Таким образом, оптимальное решение уже содержится в популяции и поэтому достаточно одного отбора для отыскания оптимальной точки или точек. В условиях популяции ограниченного размера мутация и кроссинговер являются теми операторами, которые позволяют исследовать области пространства поиска, не охваченные популяцией, и которые могут содержать более «хорошие» решения. При этом поиск может принимать различный характер в зависимости от применяемого в генетических алгоритмах соотношения исследование-использование.

Быстрый поиск субоптимального решения, как правило, достигается использованием при формировании следующих поколений преимущественно оператора кроссинговера. Такой поиск носит направленный характер и акцентируется на использовании информации об уже найденных точках внутри некоторой перспективной (относительно возможности обнаружения экстремума) области пространства поиска, что свойственно большинству детерминированных методов оптимизации. В свою очередь, исследование пространства поиска подразумевает динамическое обнаружение и переход между новыми областями. Такой режим поиска характерен для стохастических численных методов и в ГА он достигается посредством оператора мутации. Действительно, достаточно случайно изменить один бит в шаблоне, и он уже будет определять множество хромосом из других областей пространства поиска. В общем случае при решении практических задач с помощью генетических алгоритмов используют направленно-случайный поиск, характеризующийся сбалансированным, в зависимости от задачи, применением операторов кроссинговера и мутации.

Выбор значений этих, а также некоторых других управляющих параметров (число поколений, размер популяции) оказывает большое

влияние на эффективность поиска решения. Настройка генетического алгоритма является одной из главных проблем, возникающих при его использовании, так как существенно зависит от математической модели задачи оптимизации. Существует ряд рекомендаций, построенных на результатах исследования взаимовлияния управляющих параметров и направленных на выбор пропорций между их значениями. Однако в большинстве случаев настройка ГА проводится опытным путем посредством его запусков с варьированием значений этих параметров и последующим анализом полученных результатов.

2.1.4. Настройка генетического алгоритма

Исследованию влияния значений управляющих параметров на эффективность генетического поиска часто уделяется внимание в работах, посвященных ГА [46, 49, 51, 56]. Как правило, подобный анализ заключается в определении зависимости точности решения разноплановых тестовых задач оптимизации от использования параметров ГА независимо либо совместно друг с другом при варьировании их значениями. Главным достоинством таких задач является наличие априорной информации об оптимальном решении, что значительно упрощает процесс исследования эффективности ГА. В качестве результатов подобных исследований выступает ряд рекомендаций по настройке ГА для решения задач, схожих с тестовыми. Реальные же задачи в большинстве случаев такой информации не имеют, поэтому изучение взаимного влияния всевозможных параметров ГА, включающих размер популяции, используемые генетические операторы, вероятности их применения, способ кодирования переменных оптимизации и др., представляется чрезвычайно сложным.

Принимая во внимание тот факт, что размер популяции выбирается много меньшим по сравнению с общим числом ($2^{L(C)}$) возможных хромосом в пространстве поиска, а применение традиционных операторов ГА в основном достаточно для охвата большинства точек пространства решений, в работе [51] предлагается ограничиться исследованием при настройке ГА только тремя основными параметрами: размером популяции, вероятностями применения кроссинговера

и мутации. Ориентируясь на работу [51], рассмотрим более подробно условия проведения, а также ключевые результаты исследований по определению зависимости эффективности ГА от значений выбранных параметров.

Известно, что при работе ГА основные временные трудозатраты пропорциональны числу вычислений целевой функции при определении оптимальности хромосом, поэтому для сравнения эффективности ГА перед проведением экспериментов число вычислений функции ограничивалось некоторым фиксированным значением *Nf,* достаточным для нахождения экстремума в некоторой малой ε - окрестности. На основе этого значения может быть оценено максимальное число поколений $Np^* = Nf/Np$, за которые ГА с размером популяции Np может найти оптимальное решение. При этом в зависимости от сложности функции и настройки ГА значение Np^* будет разным, то есть чем больше размер популяции, тем меньше число поколений ГА и наоборот.

Здесь возникает вопрос, как количественно оценить понятия «большая» или «малая» популяция. Подобно другим нечетким понятиям точно охарактеризовать их достаточно сложно, поэтому для рассматриваемых здесь и далее задач будем предполагать, что малая популяция – $Np \le 100$; средняя популяция – $100 < Np \le 500$; большая популяция $Np > 500$. Кроме этого, нижняя и верхняя границы варьирования численностью популяции могут быть определены аналитически с помощью зависимости $[L(C), 2^* L(C)]$, приведенной в [15].

Для уменьшения влияния на результат начальной популяции, которая формируется случайным образом, ГА запускался 50 раз для каждой комбинации значений управляющих параметров. Тогда эффективность ГА может быть определена как $\eta = Nq/50$, где Nq – число случаев, когда ГА нашел оптимальное решение. Работа ГА прекращалась, когда выполнялось любое из следующих условий:

- находилось решение в некоторой малой ε – окрестности от глобального оптимума;

- достигался лимит числа поколений Ng^*.

Несмотря на изначально заданное число допустимых вычислений функции Nf, в некоторых случаях ГА было достаточно для нахождения оптимума за меньшее число вычислений функции. Поэтому, наряду с общим показателем эффективности η, использовался критерий для оценки фактора скорости локализации оптимума $\nu = 1 - \frac{N_f^*}{N_f}$, где N_f^* - действительное число вычислений функции. Экспериментальная оценка этого критерия позволяет более точно определить значение Nf для решаемой задачи.

В качестве основы для проведения экспериментов использовались известные тестовые функции и стандартный ГА с турнирным отбором.

Рассмотрим вначале процесс настройки ГА для оптимизации унимодальной функции вида:

$$f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 .$$

Эта функция (рис. 2.16) предложена Химмельблау (Himmelblau's unimodal function) и имеет глобальный минимум, равный нулю в точке с координатами [3, 2].

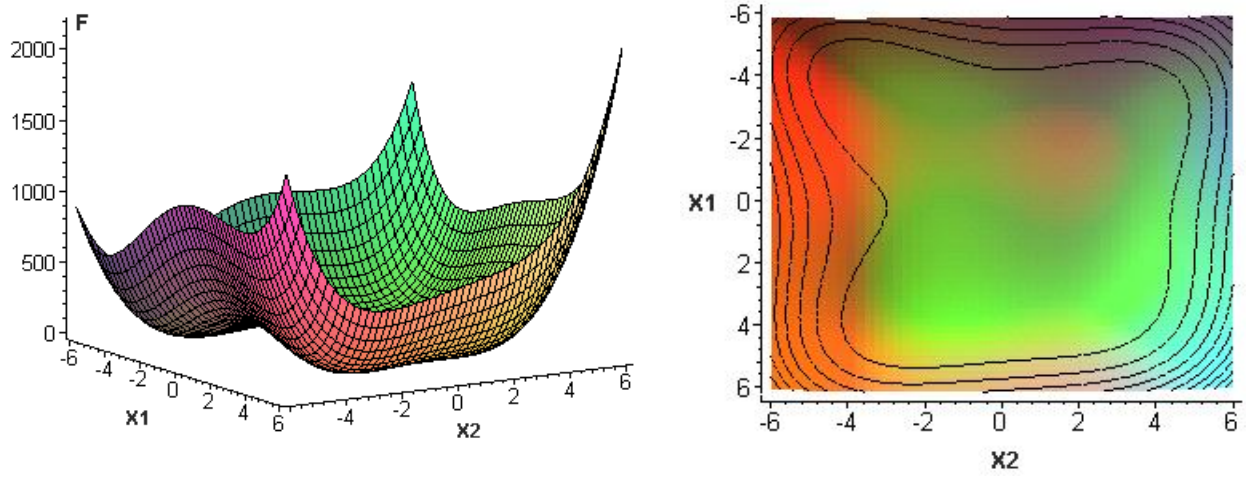


Рис. 2.16. График и топология функции Химмельблау

При оптимизации данной функции использовалось 12-битное кодирование каждой переменной, следовательно, длина хромосомы составила 24 бита. Таким образом, общее число точек пространства поиска составляет 2^{24}, или 16,8 миллионов. Тогда для нахождения оптимума в окрестности $\varepsilon = 0{,}01$ случайным поиском, учитывая область

определения функции, число вычислений приблизительно равно $\frac{6-0}{0{,}02}\cdot\frac{6-0}{0{,}02}\approx 90000$. Для ГА было определено только 3,33% от этого количества, в результате $Nf \approx 3000$, что позволяет охватить всего лишь 0,018% пространства поиска.

На рис. 2.17 показаны результаты исследования зависимости эффективности ГА для выбранной функции от значений его управляющих параметров. Как следует из рисунка, ГА в большинстве случаев находил оптимальное решение за исключением случаев с низкой вероятностью мутации, а также большим размером популяции.

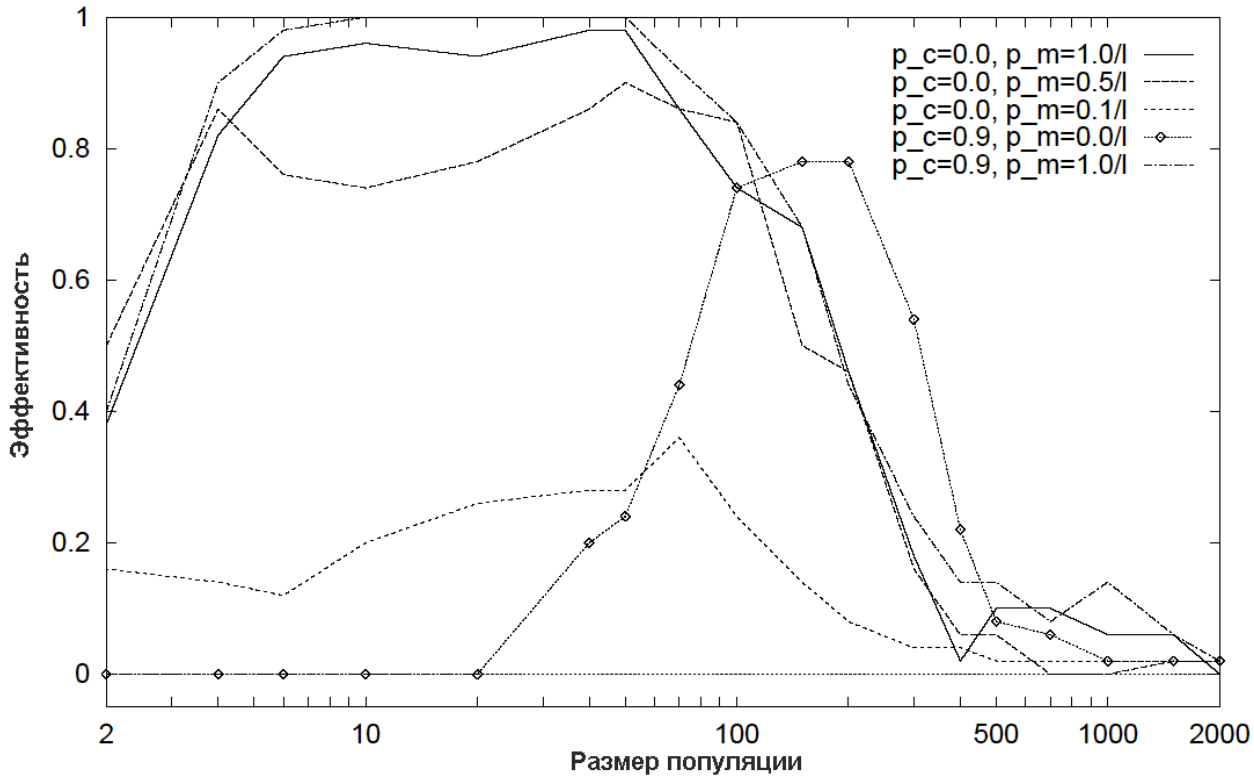


Рис. 2.17. Эффективность ГА для различных вариантов его настройки (функция Химмельблау)

На рис. 2.18 показан график зависимости скорости поиска от значений управляющих параметров ГА. Согласно ему, ГА в большинстве случаев для локализации оптимума требуется меньшее по сравнению с заданным число вычислений функции.

Таким образом, для данной функции наиболее оправдано применение популяции малого размера с преимущественным использованием оператора мутации. Это может быть обосновано с тех

позиций, что малый размер популяции является недостаточным для направленного поиска оператором кроссинговера, в то время как случайное порождение решений оператором мутации дает положительный эффект, близкий по характеру к поиску решения с помощью перебора.

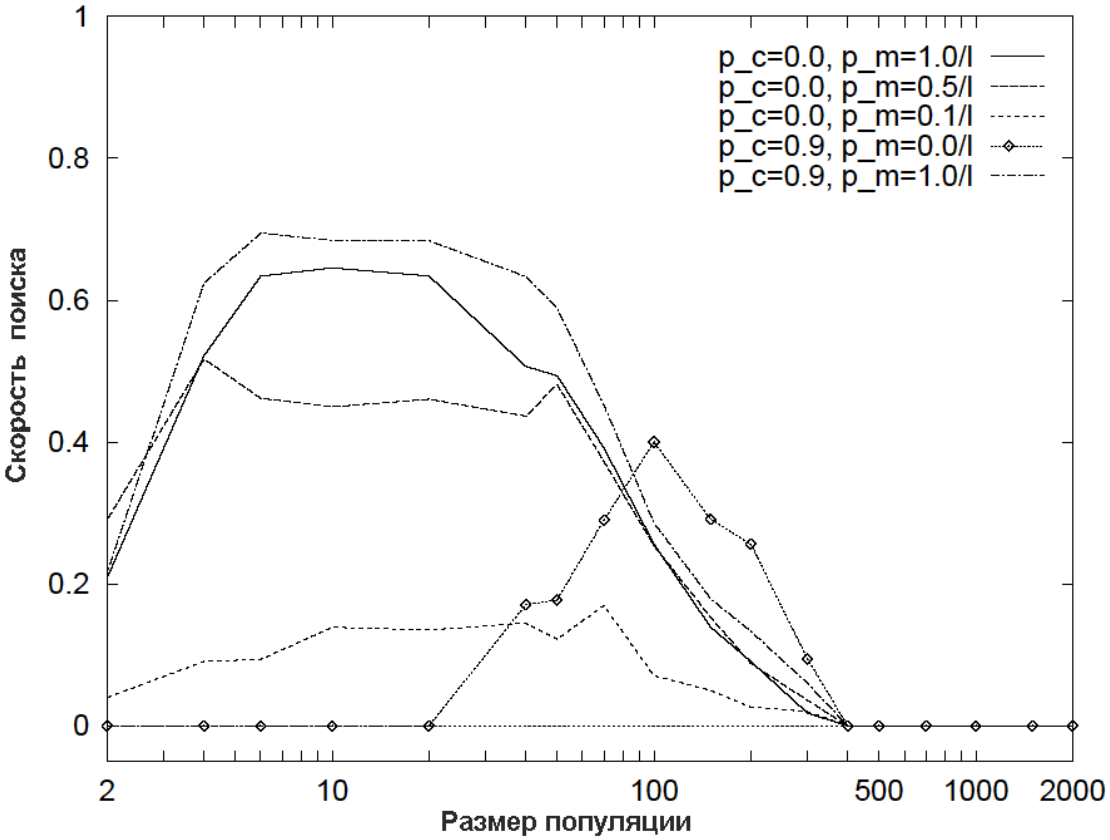


Рис. 2.18. Изменение скорости поиска ГА
для различных вариантов его настройки (функция Химмельблау)

Рассмотренная функция, с точки зрения ее аналитического задания и оптимизации, может быть отнесена к простым. Поэтому для объективности рассмотрим процесс настройки ГА для оптимизации сложной многоэкстремальной функции следующего вида:

$$f(x_1,...,x_{10}) = 200 + \sum_{i=1}^{10} x_i^2 - 10\cos(2\pi x_i) \ . \qquad (2.6)$$

Эта функция известна как функция Расстригина (Rastrigin's massively multi-modal function) и имеет область определения [-6, 6] и минимальное значение, равное нулю в точке с координатами $x_i = 0$, а также порядка 13^{10} локальных оптимумов. Для визуальной оценки

сложности этой функции на рис. 2.19 приведен ее график для двух переменных.

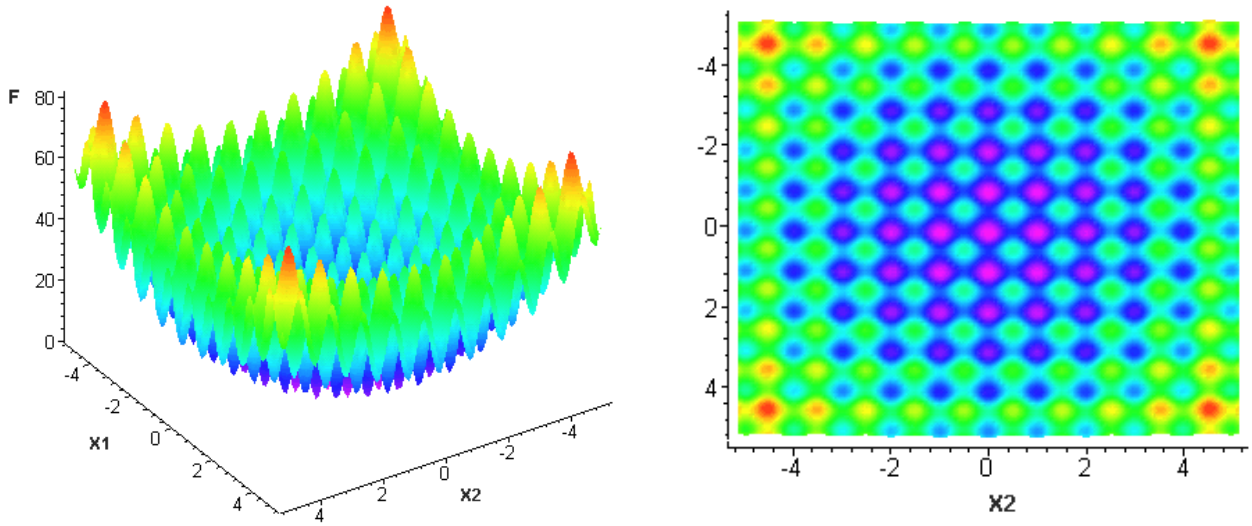


Рис. 2.19. График и топология функции Расстригина для двух переменных

При оптимизации этой функции с помощью ГА использовалось 13-битное кодирование каждой переменной, в результате размер хромосомы составил 130 бит. Погрешность в координатах оптимума принималась равной 0,1. Следовательно, для локализации решения случайным поиском требуется $[(6-(-6)/0,2]^{10}$ или около $6\cdot(10^{17})$ вычислений функции. Для ГА максимальное число вычислений функции устанавливалось равным $Nf = 45000$.

На рис. 2.20 представлено влияние комбинирования значений управляющих параметров на эффективность ГА. В отличие от результатов исследований в предыдущем случае, здесь четко проявляется разница при независимом использовании операторов мутации и кроссинговера.

Из анализа рисунка видно, что при использовании одного оператора мутации ГА не может найти решение, если же применяется один оператор кроссинговера, то ГА в 90% случаев находит его. Кроме этого, если добавляется оператор мутации даже с низкой вероятностью эффективность генетического поиска снижается. Как отмечалось ранее, тестовая функция имеет наряду с глобальным множество локальных экстремумов, поэтому оператор мутации в большинстве случаев разрушает шаблоны уже найденных перспективных решений. С другой стороны, оператор кроссинговера, комбинируя частями этих «хороших» решений, как бы уточняет их до глобального оптимума.

Объяснить такое влияние кроссинговера можно попытаться со следующих позиций.

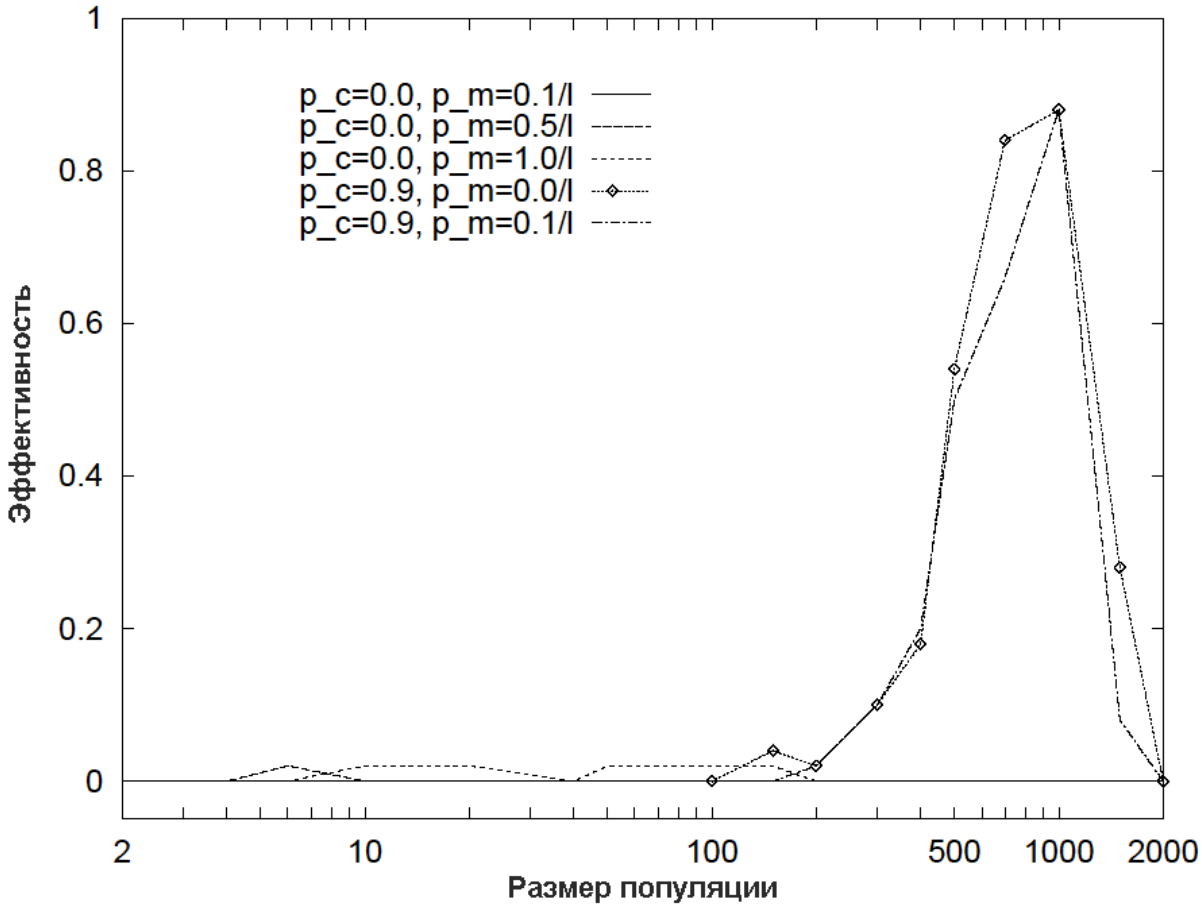


Рис. 2.20. Эффективность ГА для различных вариантов его настройки (функция Расстригина)

При использовании популяции большого размера вероятность того, что уже на первой генерации ГА направит поиск в направлении оптимума выше, чем при малой популяции. Это связано с тем, что случайно инициализированная популяция большого размера может содержать хромосомы, близкие с точки зрения расстояния Хемминга до оптимума, что особенно важно при оптимизации сложных функций. При этом ГА не сможет найти оптимальное решение, если заданное число поколений меньше минимального расстояния Хемминга до оптимума в начальной популяции (принимаем, что вероятность мутации меньше, чем $1/L(C)$ на локус). Другими словами, наличие даже приблизительной априорной информации об оптимальном решении позволяет оценить число поколений, необходимых для его локализации. При неправильно выбранном направлении ГА с

большой популяцией имеет больше шансов скорректировать его в сторону оптимального решения. Действительно, использование одного кроссинговера в сочетании с большой популяцией дает больше возможностей ГА для генерации строительных блоков, нежели преобладание оператора мутации.

На рис. 2.21 показана динамика изменения скорости поиска для разных вариантов настройки ГА. Из рисунка видно, что для данной задачи оптимизации, а также подобных ей ключевым фактором является использование только оператора кроссинговера в сочетании с популяцией большого размера.

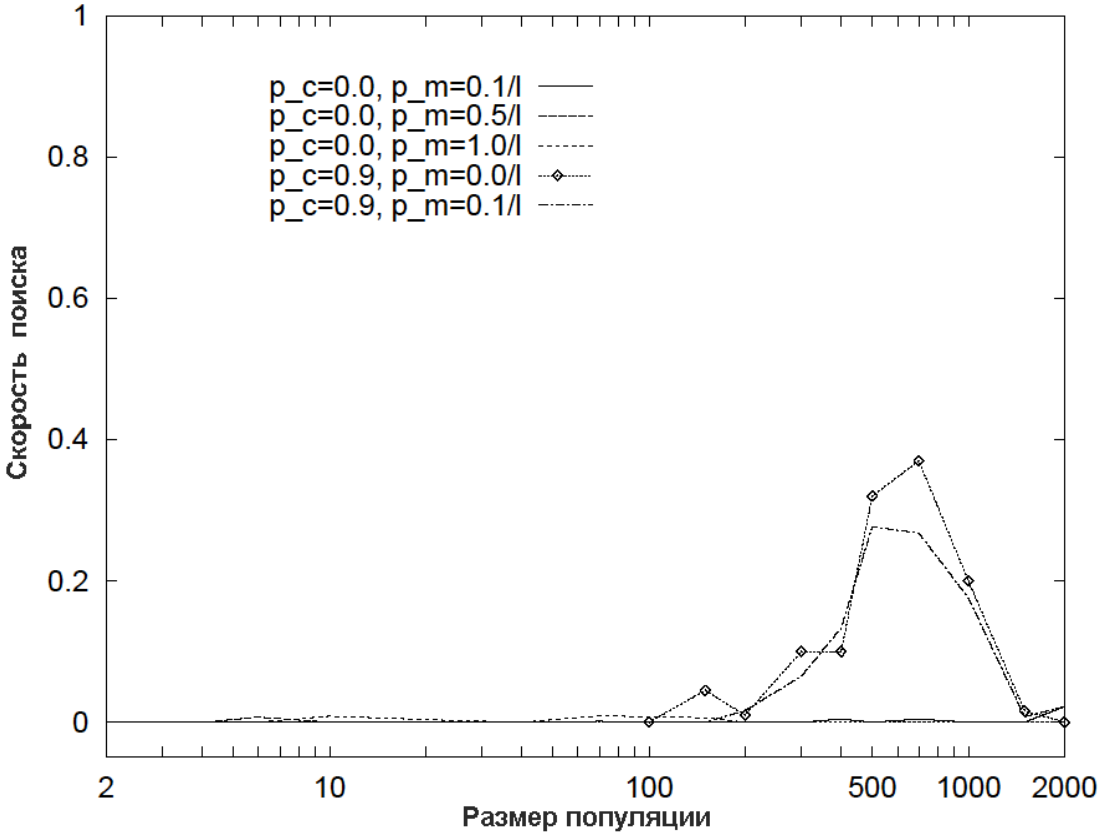


Рис. 2.21. Изменение скорости поиска ГА
для различных вариантов его настройки (функция Расстригина)

Таким образом, относительно рассмотренных тестовых задач, в частности, можно сделать следующие выводы:

- применение популяции малого размера с преобладанием оператора мутации целесообразно при оптимизации простых (с точки зрения имеющейся априорной информации) функций;

- при оптимизации сложных функций, а также математических моделей, уровень сложности которых по той или иной причине

трудно оценить, рекомендуется использовать популяцию большого размера в сочетании с применением только оператора кроссинговера;

- для большинства задач ГА работает более эффективно при использовании совместно операторов кроссинговера и мутации с соответствующим размером популяции.

К сожалению, трудно дать конкретные и в то же время универсальные рекомендации по настройке ГА для общего случая, например в [43] настройку предлагается выполнять в соответствии с проявлением тех или иных проблем, возникших при работе ГА. Для этого приводится набор мер, применение которых позволит уменьшить или разрешить проблему (табл. 2.7).

Таблица 2.7

Потенциальные проблемы при работе ГА
и меры для их разрешения

Проблема	**Меры для ее разрешения**
Низкая оптимальность решений	- увеличение числа поколений; - увеличение численности популяции; - изменение критерия оценки решений; - изменение стратегии выбора родительских хромосом для кроссинговера; - изменение схемы кроссинговера и стратегии формирования нового поколения
Преждевременная сходимость	- изменение стратегии выбора родительских хромосом для кроссинговера; - отслеживание тенденции к появлению в популяции групп одинаковых хромосом и их удаление; - использование кроссинговера с сильно разрушающей способностью; - увеличение вероятности мутации
Значительные скачки значений оптимальности решений в поколениях	- применение стратегии элитизма; - использование кроссинговера со слабой разрушающей способностью; - уменьшение вероятности мутации
Преобладание удовлетворительных результатов над хорошими	- изменение стратегии выбора родительских хромосом для кроссинговера; - изменение операторов кроссинговера и/или мутации; - использование версий ГА с несколькими популяциями

Что касается вероятностей применения генетических операторов, то во многих источниках предлагаются следующие эмпирически полученные значения $0,6 \le Pc \le 0,9$ и $0,001 \le Pm \le 0,1$ (на 1 бит). Этот вариант настройки соответствует случайно-направленному характеру поиска ГА и может быть в той или иной степени эффективно использован для поиска решения большинства задач оптимизации без априорной информации. С другой стороны, имея в этом случае целый диапазон варьирования возможных значений управляющий параметров ГА, лучшая их комбинация может быть найдена на основе их экспериментального исследования, возможно, при новом сочетании случайных факторов, найденное решение окажется более привлекательным.

2.1.5. Мониторинг процесса эволюционного моделирования

Качество работы ГА возможно оценить не только по полученным результатам, но и визуально по динамике процесса эволюционного моделирования [20]. Различные способы визуализации процесса функционирования ГА являются мощными когнитивными средствами его мониторинга, позволяющими оценить не только начало стагнации в поиске решения, но и понять, что называется изнутри, степень влияния и взаимодействия между генетическими операторами, особенно в многомерных поисковых пространствах.

В работах [47, 58], посвященных применению и исследованию различных подходов к визуализации, отмечается зависимость эффективности их применения от решаемой задачи ГА. Так, при оптимизации расписания и маршрута коммивояжера форма кодирования параметров, а также используемых генетических операторов может отличаться, что приводит к необходимости выбора специальных средств графического мониторинга работы ГА. Однако для общего случая

основные способы визуализации могут быть классифицированы в соответствии с типом отображаемой информации [47]:

- информация о наследственных отношениях между родителями и потомками в популяции;

- информация о сходимости генетического поиска;

- информация о рельефе fitness-функции.

Далее в качестве источника такой информации будем рассматривать функционирование ГА при решении задачи оптимизации функции (2.6). При этом предполагается, что перед отображением хромосомы в популяции упорядочиваются по значению fitness-функции, таким образом, что лучшей хромосоме будет соответствовать минимальный индекс.

Визуализация информации о наследственных отношениях между родителями и потомками в популяции. Позволяет оценить не только взаимоотношения хромосом в поколениях, что важно для некоторых задач моделирования искусственных жизненных форм, но и главное - определить в каждом из поколений родителей потенциальных кандидатов в лучшее решение. Это, в свою очередь, дает знания о характере поискового пространства, поскольку, если лучшее решение из поколения в поколение порождается от «лучших» родителей, то можно предположить, что такое поведение ГА сигнализирует об унимодальности оптимизируемой функции или недостаточном исследовании поискового пространства.

На рис. 2.22 представлено дерево «родственных» отношений между первыми десятью четными поколениями. Линия между j-м решением в поколении t+2 и i-м решением в поколении t символизирует наследственную связь между i и j. При этом одна линия, ведущая к потомку означает клонирование родительской хромосомы, а две линии соединяют потомка с родителями, участвующими в кроссинговере.

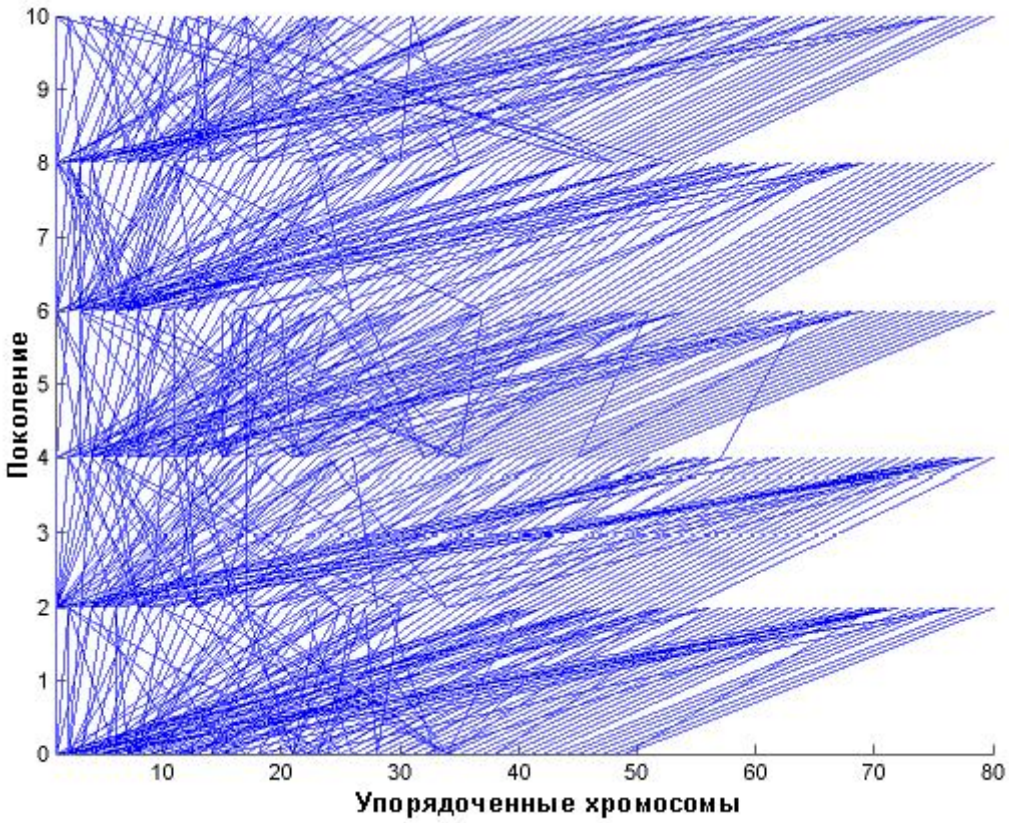


Рис. 2.22. Представление наследственных отношений в поколениях

Как следует из рисунка, в ряде случаев решения с высокой fitness-функцией способствуют появлению потомков с низкой приспособленностью. Также наблюдается и обратная ситуация, что мотивируется совместным применением генетических операторов и свидетельствует о широком использовании поискового пространства.

В продолжение на рис. 2.23 представлен рисунок, показывающий роль членов начальной популяции в локализации оптимального решения. На графике упорядоченные хромосомы нулевого поколения (ось - x) изображены напротив значений их fitness-функции (ось - y).

Также представленные на графике бары определяют изменение оптимальности потомков в ходе эволюции. Анализ графика позволяет сделать вывод, что окончательное решение генерируется путем смешивания половины из лучших сорока хромосом начального поколения. Исходя их этого можно предположить об ограниченности исследования пространства решений и о возможности сокращения численности популяции. В свою очередь, на рис. 2.24 приведена динамика изменения оптимальности лучших потомков нулевого поколения на протяжении двенадцати эпох.

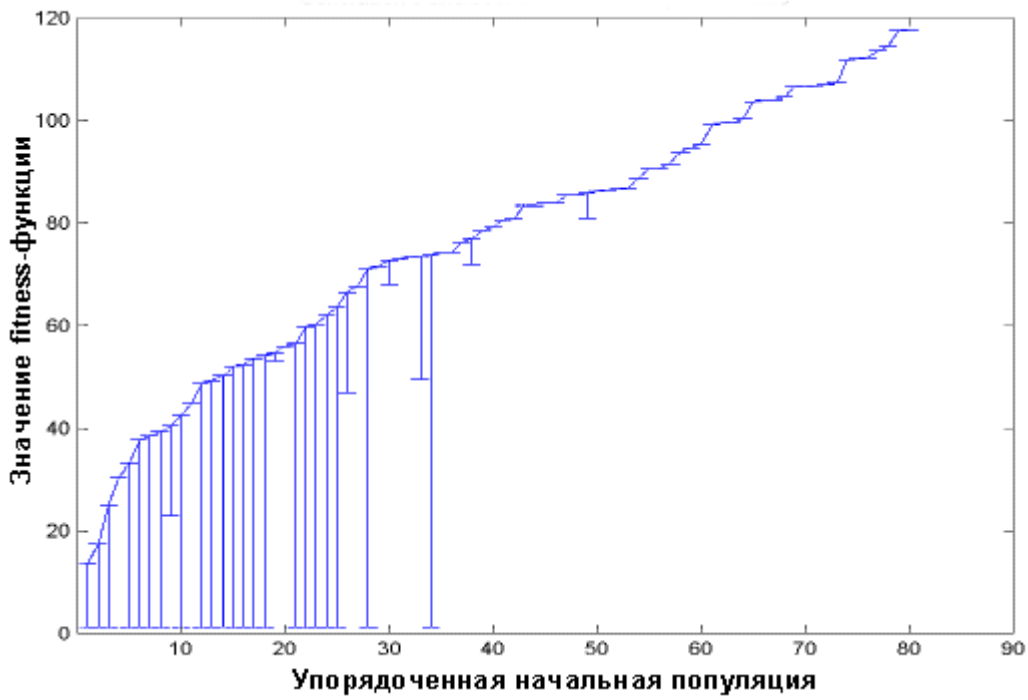


Рис. 2.23. Влияние начальной популяции на локализацию оптимального решения

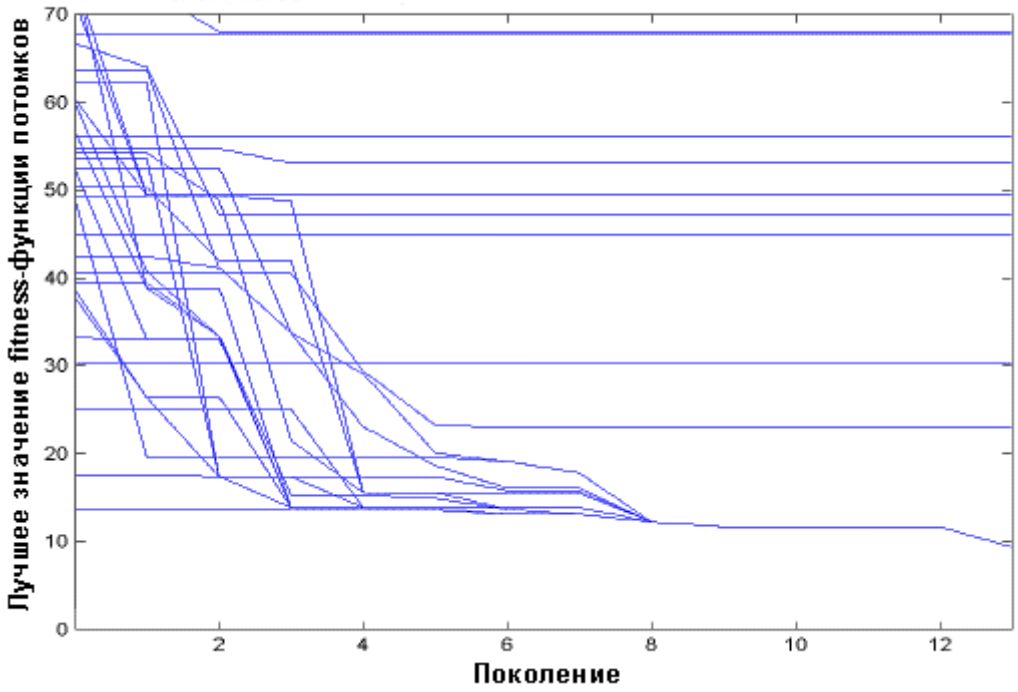


Рис. 2.24. Динамика изменения оптимальности лучших потомков

Из графика видно, что часть потомков обеспечивает переход в следующее поколение (горизонтальные линии на графике) либо через оператора отбора, либо из-за того, что оптимальность потомков не лучше родителей. В то же время другие хромосомы увеличивают свою оптимальность, вырождаясь при этом к восьмому поколению в одинаковые хромосомы. Все это свидетельствует о раннем (восьмое поколение) формировании окончательного пространства решений, что, в общем, является вполне допустимым, если речь идет о факте локализации оптимального решения. Однако в большинстве случаев подобная ситуация может потребовать внесения разнообразия в популяцию или детерминированного формирования начальной популяции.

Информация о сходимости генетического поиска. Она позволяет определить динамику изменения значения fitness-функции лучшего, среднего по популяции и худшего решения от поколения к поколению. На рис. 2.25 представлены типичные графики для данного вида мониторинга рассматриваемой информации. На рисунке четко прослеживается эффект от применения оператора мутации в виде больших отклонений значений, которые принимает худшее решение. Несмотря на его применение, популяция достаточно быстро теряет разнообразие, что видно из сближения графиков средних и лучших значений. Все это свидетельствует о вырождении популяции в одну точку пространства решений. Дальнейшее «оживление» процесса эволюционного моделирования может быть достигнуто путем увеличения доли случайного поиска или изменения стратегии отбора.

Другим эффективным, но малоизвестным способом анализа динамики изменения приспособленности популяции является использование так называемых цветных карт значений переменных оптимизации относительно лучшего, среднего и худшего значения fitness-функции в поколениях. На рис. 2.26 показана такая цветная карта для первого случая.

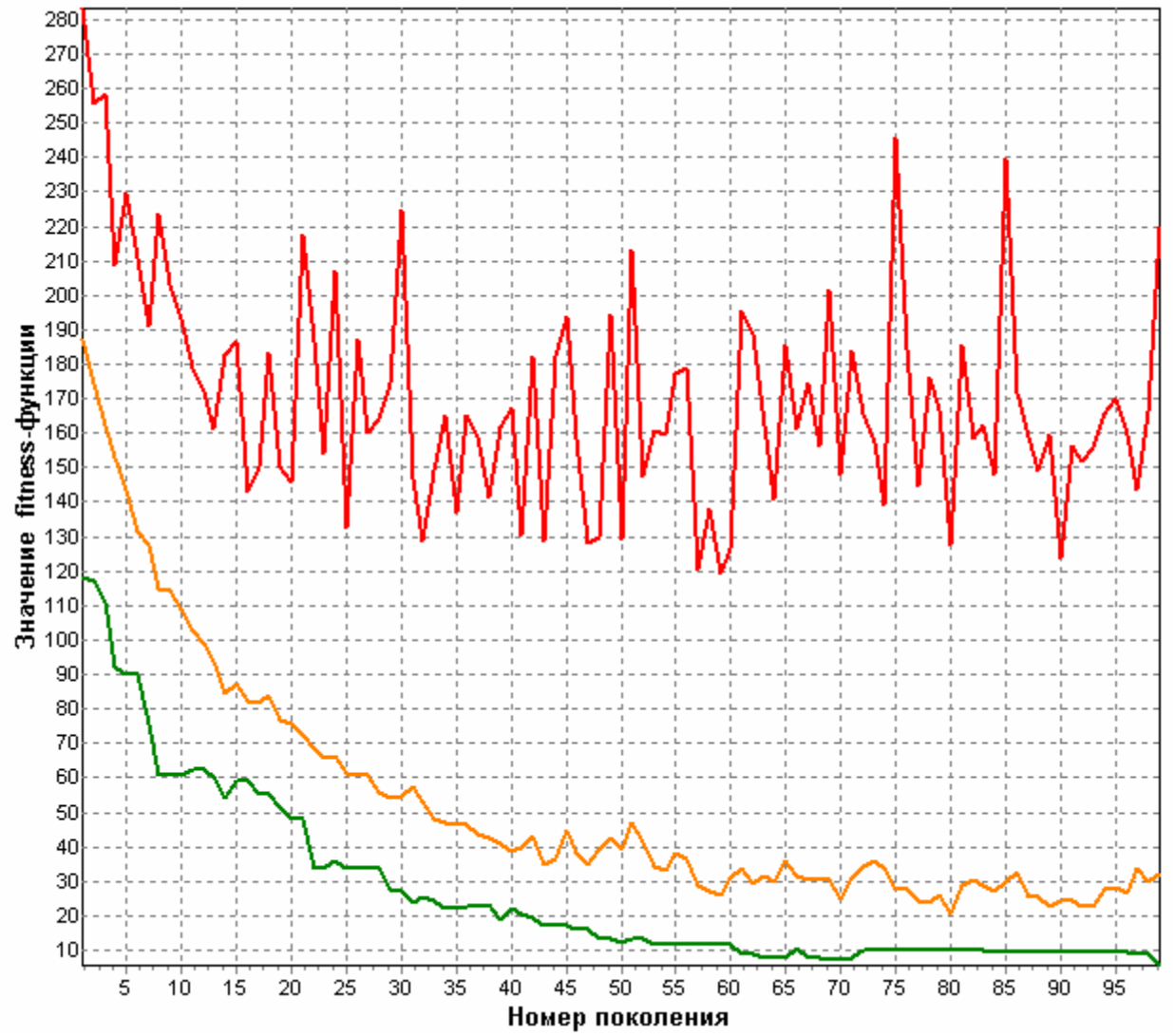


Рис. 2.25. Изменение лучшего, среднего и худшего значений fitness-функции между поколениями:

Анализ представленной карты позволяет сделать следующие выводы:

- в начальной популяции только четвертая и девятая переменные имеют значения, близкие к оптимальным, однако в процессе работы ГА они их теряют и восстанавливают только к 15 (переменная № 4), 50 (переменная №9) поколениям;

- первая переменная имеет отрицательные значения вплоть до 90-го поколения, после которого она принимает оптимальное нулевое значение;

- седьмая переменная остается в области отрицательных значений на протяжении ста поколений;

- часть переменных, например вторая и пятая, изменяют свои значения достаточно динамично в процессе эволюции, в то время как другие (четвертая, шестая и десятая) остаются почти константами.

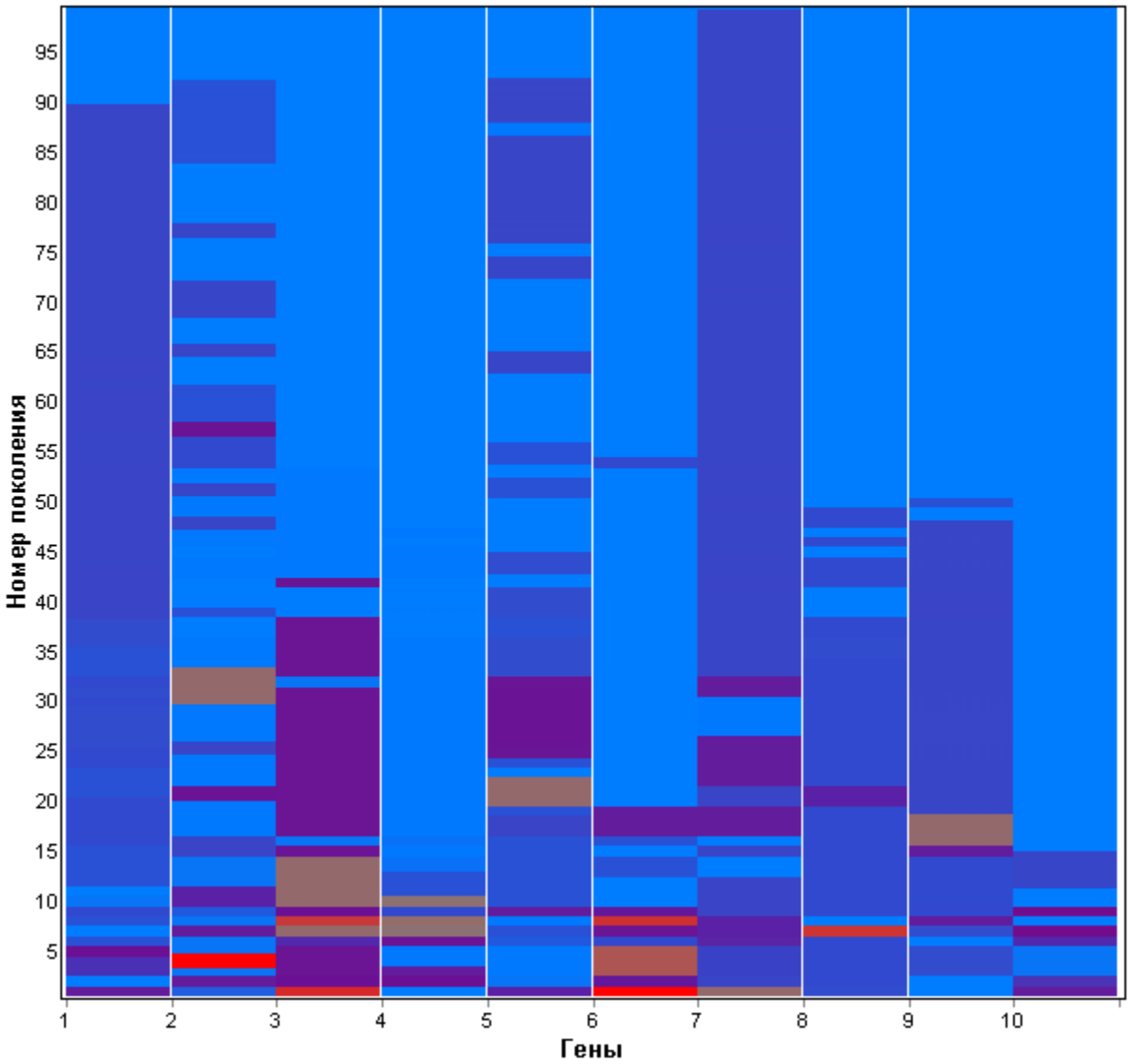


Рис. 2.26. Динамика изменения значений переменных оптимизации, соответствующих лучшему решению

Кроме этого, информация, представленная на рис. 2.26, позволяет сделать важное заключение о том, что ГА при поиске оптимального решения изменяет значения не всех возможных переменных, а только части из них, что на несколько порядков снижает трудоемкость локализации решения по сравнению с другими методами

оптимизации. Это подчеркивает тот факт, что ГА использует одно из свойств реального мира – независимость различных подсистем объектов. Другими словами, работа ГА есть проявления интеллектуального поведения, так как подобно опытному мастеру при поиске неисправностей соблюдается эвристическое правило – «никогда не трогать все сразу, только по очереди».

Информация о рельефе fitness-функции. Наличие такой априорной информации позволяет существенно упростить процесс поиска оптимума, однако ее наличие свойственно при решении одномерных и двумерных оптимизационных задач. В других случаях поиск ведется вслепую и для оценки рельефа fitness-функции, в частности, могут быть использованы методы спектрального анализа [17]. Также в [47] для этих целей предлагается анализировать частоту появления различных значений fitness-функции на протяжении всего эволюционного моделирования. На рис. 2.27 представлена такая гистограмма.

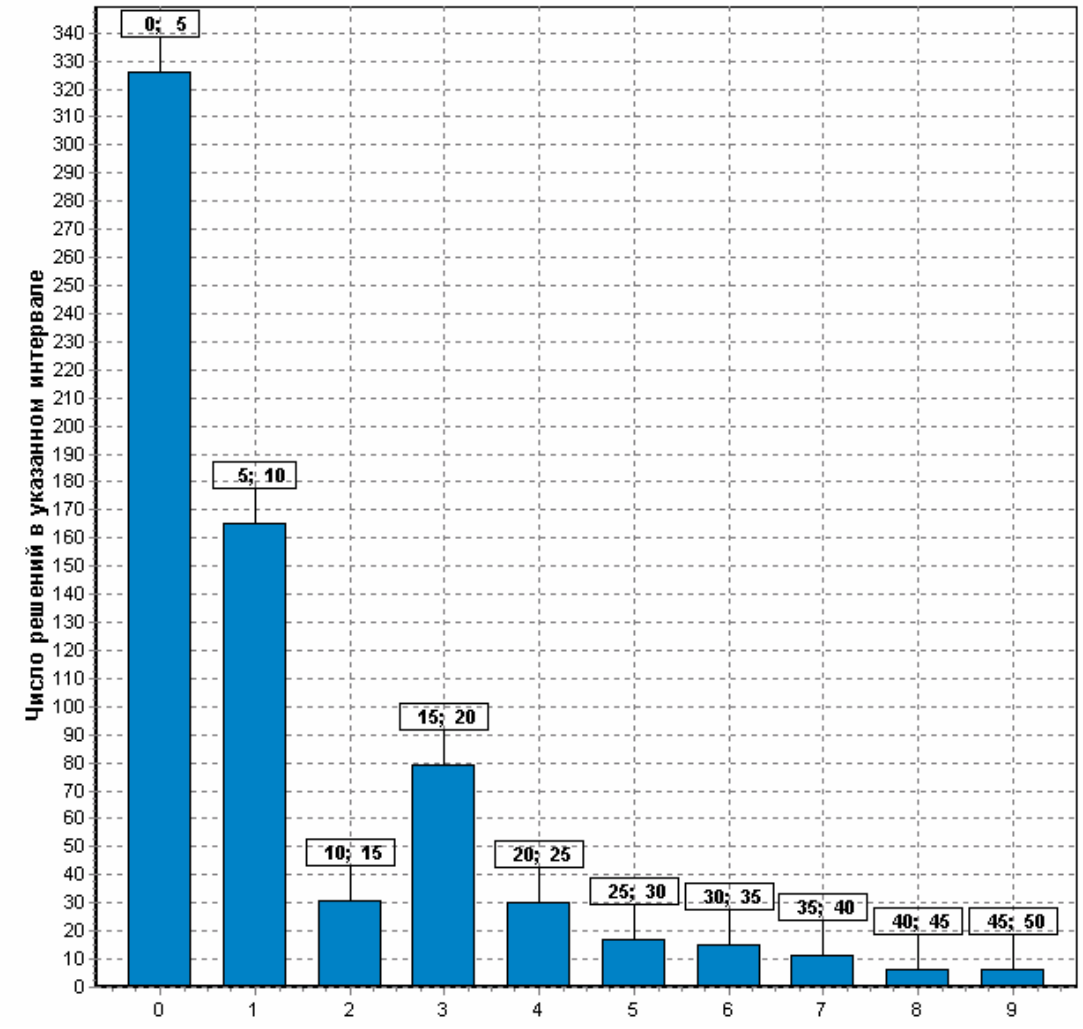


Рис. 2.27. Гистограмма fitness-функции

Из графика видно, что меньшие значения fitness-функции генерируются гораздо чаще, что позволяет предположить наличие связи между числом различных ее значений и их близостью к оптимальному решению.

Рассмотренные способы мониторинга процесса эволюционного моделирования являются важным источником информации о динамике поиска решения. Они позволяют не только определить необходимые корректировки в настройке ГА для повышения качества его работы, но и получить представление о характере рельефа оптимизируемой многомерной функции. Как следствие, это может стать одним из определяющих критериев выбора или конструирования более эффективной fitness-функции, обладающей низкой вероятностью возникновения явления преждевременной сходимости.

2.1.6. Модификации генетического алгоритма

Эффективность работы ГА во многом определяется его настройкой, а также особенностями решаемой им задачи оптимизации. Наряду с рассмотренным стандартным вариантом ГА, позволяющим эффективно решать достаточно широкий спектр таких задач, существуют и постоянно разрабатываются, совершенствуются его модификации. Это связано с постоянным стремлением повысить качество эволюционного моделирования и достигается путем создания новых технологий генетического поиска или изменением работы известных, а также созданием новых генетических операторов. В результате появляются как новые универсальные, так и версии ГА, адаптированные специально для решения некоторой конкретной оптимизационной задачи.

Среди подходов к созданию новых технологий генетического поиска можно выделить распараллеливание ГА, структурирование популяции, миграцию хромосом и популяций и т.п. Эти идеи, расширяющие возможности эволюционного моделирования, лежат в основе таких ГА, как [15]:

- параллельный ГА контролируется большим числом параметров и использует в процессе своего функционирования многопроцессорную систему;

- структурный ГА, в котором применяется идея группировки индивидуумов на основе специализированного контролируемого гена;

- ГА с моделью островов, согласно которой популяция разбивается на подпопуляции с последующей взаимной миграцией лучших решений;

- гибридный ГА основывается на комбинации с другими технологиями поиска – локальный поиск (local search), поиск с запретами (tabu search), моделирование отжига (simulated annealing);

- и др.

Среди модификаций ГА, во многом использующих классическую схему функционирования, особое место занимают ГА с динамически изменяемыми в процессе эволюции структурными параметрами. Примерами таких алгоритмов являются мобильный ГА, оперирующий хромосомами, генами переменной длинны и поколенческий ГА, характеризующийся варьируемым размером популяции. Рассмотрим особенности их функционирования подробнее [15, 37].

Большинство типов генетических алгоритмов в настоящее время используют заранее определенные схемы кодирования строк и структуры генетических операторов. Независимо от характера решаемой проблемы обычно используются как хромосомы, так и образующие их гены фиксированной длины и одно- или двухточечная схема рекомбинации. Имеющиеся исключения в большинстве случаев также ограничены специальными условиями, накладываемыми на операции обработки строк. В противоположность этому в природе эволюция от простого к сложному использует значительно больше разнообразных механизмов регуляции на уровне генотипа, например расщепление и воссоединение генома биологической клетки организмов, что не только увеличивает число копий генов, но и порождает новые взаимодействия между существующими генами, модифицируя их активность. Новые функции, которые приобретают гены в результате подобных перестроек, приводят к образованию новых функциональных направлений.

Для воссоздания свойств подобных генетических механизмов в ГА с целью увеличения его эффективности была разработана концепция особого типа ГА - **мобильного генетического алгоритма**. Свое название данный тип алгоритма получил из-за использования хромосом переменной длины. Также такой ГА известен как messy Genetic Algorithm (mGA) [56].

Этот алгоритм основан на комбинации относительно коротких, протестированных блоков генов, направленной на создание больших генотипов, кодирующих все нужные характеристики решаемой задачи. Введение в структуру ГА генов и хромосом переменной длины и новых генетических операторов часто приводит к улучшению качества решений по сравнению со схемами классического типа [37]. Характерной особенностью мобильного ГА является отсутствие ограничений на позиционирование генов внутри хромосом, допуская их различные перестановки внутри строк, вследствие чего не требуется знание априорной информации о наиболее оптимальном расположении битов внутри строки, что очень важно в приложениях.

Мобильный генетический алгоритм имеет следующие основные отличия от стандартного ГА:

- использование хромосом переменной длины, которые могут быть как пере-, так и недоопределены по отношению к решаемой задаче;

- введение правил чтения или экспрессии генов;

- использование операторов CUT (разрезание) и SPLICE (сцепление) вместо традиционной схемы кроссинговера, оперирующего над генами фиксированной длины;

- конкуренция между строительными блоками генотипа для отбора наиболее оптимальных.

- разделение эволюции на две фазы: предварительная фаза и фаза процессинга.

Кроме того, алгоритм снимает ограничение, связанное с фиксированным положением гена внутри хромосомы, свойственного стандартному ГА. Это достигается путем определения гена как упорядоченной пары, идентифицирующей имя гена, его значение и определения хромосомы как объединения таких генов. В качестве примера рассмотрим случай с длиной $L(C) = 3$ и соответствующую ему

хромосому из трех битов (011) стандартного ГА. Данная хромосома будет представлена в новом алгоритме (используя LISP-подобный синтаксис) как {(1 0), (2 1), (3 1)}, где каждый бит идентифицируется его именем и значением. В этой строке первым объектом является ген «1» со значением 0, вторым - ген «2» со значением 1 и третьим - ген «3» со значением 1. Так как и имя, и значение гена определены, то может быть достигнута любая необходимая перестройка внутри хромосомы для сцепления определенных генов.

Такая же схема идентификации гена применяется и во многих других алгоритмах, использующих перемещаемые аллели внутри хромосомы. Но мобильный алгоритм имеет отличающую его от стандартной схемы особенность: не накладывается никаких требований на наличие избыточного количества генов внутри хромосомы и на наличие одних и тех же генов со взаимно исключающими значениями. Подобные ситуации разрешаются путем использования механизма экспрессии.

Переопределение устраняется вследствие процедуры экспрессии генов, использующей правило «слева направо», согласно которому строка сканируется слева направо, и ген, расположенный левее, считается активным в данной хромосоме. Так, строка {(1 0), (2 1), (1 1)} из-за экспрессии перейдет в строку {(1 0), (2 1)} потому, что вторая версия первого гена не будет использована согласно применяемому правилу. Данное правило было выбрано вместо различных схем, использующих понятие эффективности генов, так как при применении таких схем гены, получившие большое преимущество в начале эволюции (большие значения оптимальности), могут заблокировать проявление генов, которые на данной стадии менее эффективны, но являются строительными блоками искомого оптимума.

Для устранения проблемы недоопределения используется другой подход, согласно которому мобильный ГА заполняет недоопределенные позиции хромосомы участком наиболее конкурентноспособной хромосомы – предшественницы, которая является оптимальной на предшествующем этапе эволюции. Такая генетическая структура, получаемая на одном уровне эволюции, будет представлять лучший исходный материал для формирования оптимальных

решений на следующем этапе эволюции, а оптимальный элемент этой структуры будет использоваться как строительный блок генотипа, заполняя соответствующие недоопределенные гены.

В цикле работы мобильного генетического алгоритма можно выделить три основных этапа:

- инициализация;
- предварительная фаза;
- фаза процессинга.

Инициализация. Инициализация выполняется посредством создания популяции, содержащей по одной копии всех подстрок длиной K. Этим гарантируется наличие в популяции всех необходимых строительных блоков генотипа. Для выбора наиболее подходящего блока необходимо оценить каждый член популяции. Размер популяции при такой схеме инициализации может быть определен как

$$Np = 2^K (\frac{\mathrm{L(C)}}{K}).$$

Так как размерность равна $L(C)$, то существует $K^{L(C)}$ комбинаций генов размера K, и для каждой комбинации существует 2^K различных бинарных комбинаций аллелей.

Предварительная фаза. Данная фаза предназначена для увеличения пропорции оптимальных элементов в популяции. Выполняется только отбор, другие генетические операторы (разрезание, сцепление, мутация) не используются. Оценка производится только один раз во время предварительной фазы, так как вследствие неизменяемости хромосом, значение их оптимальности сохраняется. Затем происходит отбор наиболее хороших хромосом, чтобы увеличить их процент в популяции. Одновременно через определенные интервалы времени количество элементов популяции сокращается, чтобы достичь уровня, который будет сохраняться постоянным на следующей стадии эволюции.

Фаза процессинга. После обогащения популяции конкурентоспособными элементами на предварительном этапе производится ее обработка, которая напоминает стандартный ГА. В течение этой фазы отбор используется совместно с модифицированным кроссинговером и возможной мутацией строк.

Для рекомбинации строк переменной длины применяются два новых оператора CUT и SPLICE вместо обычного кроссинговера с фиксированным положением одной или двух точек рекомбинации (рис. 2.28). CUT применяется к хромосоме с вероятностью

$$P_{cut} = (L(C) - 1)\, c_{cut},$$

где c_{cut} - некоторая константа, например 0,1.

В мобильном алгоритме гены могут находиться в произвольном месте хромосомы, и это не влияет на их интерпретацию и эффективность, что требует изменения истолкования понятия шаблона. В данном случае шаблон представляет собой кластер определенного подмножества генов, содержащих определенные аллели с длиной менее некоторого специфицированного значения.

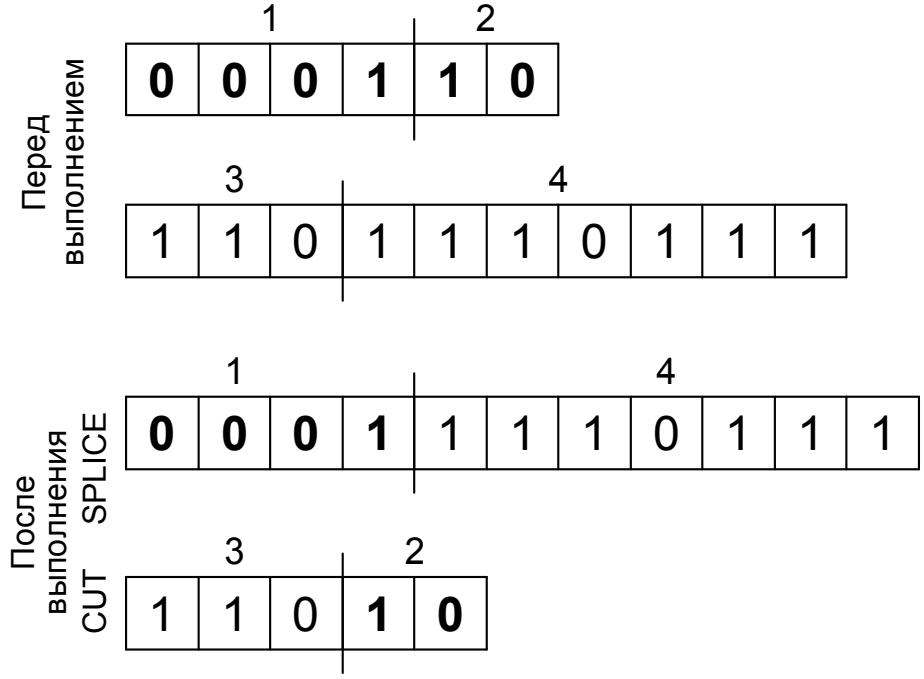


Рис. 2.28. Функционирование операторов CUT и SPLICE

SPLICE объединяет две строки вместе с фиксированной вероятностью P_{spl}. CUT и SPLICE имеют два предельных типа поведения. Первоначально, когда строки короткие, преобладает сцепление хромосом. Позднее, когда строки становятся достаточно длинными, преобладает их деление. Применяемые вместе, CUT и SPLICE производят эффект подобный кроссинговеру стандартного генетического алгоритма.

Одной из важных характеристик мобильного алгоритма является количество времени, затрачиваемого на вычисления, поэтому одним из главных направлений его применения является реализация на параллельных архитектурах. Среди других задач можно выделить следующие: проведение более строго анализа алгоритма, разработка обучаемых систем классификации на основе нового алгоритма, исследование типов функций, к которым данный алгоритм может быть применен с максимальной эффективностью.

Поколенческие генетические алгоритмы. Выделяются наличием в них логистических моделей управления жизненным циклом популяции и включают такие модификации генетического алгоритма, как ГА с переменным временем жизни, ГА с динамическим размером популяции, ГА с логистической моделью. Эти алгоритмы расширяют стандартный ГА прежде всего путем использования такой аналогии с живой природой, как зависимость времени жизни особи от степени ее приспособленности. При этом к стадии репродукции допускаются элементы с более высокой целевой функцией и «прожившие» заданное число поколений.

Обобщением таких модификаций ГА является поколенческий алгоритм с логистическими моделями [15, 24]. Последовательность его работы включает следующие стадии.

Шаг 1. Ввод исходных данных (целевая функция, вероятности применения генетических операторов) и значений параметров управления популяцией (возраст особи для репродукции, уровень репродукции, размер начальной популяции, шаг для сокращения популяции).

Шаг 2. Формирование начальной популяции.

Шаг 3. Пока не достигнут возраст для репродукции выполняется стандартный ГА.

3.1. Вычисление целевой функции, ранжирование, отбор особи для репродукции.

3.2. Применение генетических операторов кроссинговера и мутации.

3.3. Вычисление возраста особи.

3.4. Формирование следующей популяции.

3.5. Если не достигнут критерий останова, переход к п.3.

Шаг 4. Применение генетических операторов к особям, достигшим возраста репродукции.

Шаг 5. Вычисление размера и формирование следующей популяции.

Размер новой популяции определяется как

$$P_{t+1} = P_t(1 + R(t+1)) - N_D(t+1),$$

где N_D - число удаляемых элементов на итерации *t+1*;

R(t+1) – уровень репродукции на шаге *t+1*, задается пользователем.

Использование динамического размера популяции позволяет повысить ее разнообразие и роль конкуренции в процессе эволюции.

$$N_D = trunc(R(t+1)(1 - \frac{t-h}{M}) Np(t)),$$

где *M* – размер поискового пространства;

Np(t) – размер популяции в поколении *t;*

h – время от момента рождения до момента репродукции особи.

Значение *h* может быть определено как $max(LT_i(t))*\xi$ (ξ=[0.2 , 0.3]) – максимальное время жизни хромосомы *i* на итерации *t*.

$$LT_i = \min(LT(t+1)) + c(t+1) \cdot \tilde{f}(x_i),$$

где *min(LT(t+1))* – минимальное время жизни на итерации *(t+1)*;

c(t+1) – константа большая или равная нулю.

Шаг6. Если не достигнут критерий останова, переход к шагу 3.

Шаг7. Если количество итераций больше заданного числа, то популяция сокращается до начальных размеров и выполняется переход к шагу 3.

На каждой итерации, кратной заданному числу, происходит ее усечение до начального размера *Np* посредством удаления элементов, имеющих минимальное время жизни.

Адаптация фундаментальной теоремы ГА к этой модификации позволяет сделать вывод [15], что число шаблонов, выживших на *t+1*

шаге, определяется как сумма шаблонов, появившихся на этом шаге после операторов отбора, кроссинговера, мутации и шаблонов, время жизни которых не истекло. Как следствие повышается вероятность выживания шаблонов с более высокой целевой функцией.

В настоящее время попытки применения ГА при решении прикладных проблем охватывают не только класс традиционных задач оптимизации, но и распространяются на другие направления искусственного интеллекта, например управление сложными динамическими объектами в условиях неопределенности, принятие решений на основе нечеткого многокритериального выбора. Подобные исследования подразумевают под собой создание теоретических и практических основ для модификаций ГА, подавляющее большинство среди которых концентрируется вокруг следующих направлений:

- разработка и исследование новых модификаций генетических алгоритмов и операторов, а также создание математического обеспечения для анализа и прогнозирования эффективности их практического применения;

- интеграция генетических алгоритмов как с существующими численными методами оптимизации, так и с другими направлениями «мягких» вычислений (нейронные сети, нечеткое моделирование) для создания гибридных интеллектуальных систем [35];

- применение генетических алгоритмов в задачах многоэкстремальной и многокритериальной оптимизации для автоматизации поиска альтернативных оптимальных решений;

- применение генетических алгоритмов в самоорганизующихся, адаптивных системах, а также для моделирования систем искусственной жизни;

- использование генетических алгоритмов для решения практических задач эволюционного моделирования в социальных, экономических и технических системах;

- реализация генетических алгоритмов на параллельных вычислительных архитектурах.

2.2. Генетическое программирование

Генетическое программирование (ГП) является развитием ГА для решения оптимизационных задач в пространстве компьютерных программ. В данном случае объектами эволюции, составляющих популяцию, являются не бинарные хромосомы, а компьютерные программы. Цель генетического программирования заключается в синтезе программы для задачи с неизвестным алгоритмом решения.

Концепция ГП была предложена Дж. Коза (J. Koza) [61] в попытке создания средств автоматического программирования на основе эволюционных технологий. Предпосылкой этому послужила одна из главных проблем искусственного интеллекта – синтез программы по описанию задачи без непосредственного программирования.

Препятствием к тому, чтобы научить компьютер решать задачи, не будучи соответствующим образом для этого запрограммированным, является то, что существующие сегодня методы машинного обучения, построения адаптивных и самоорганизующихся систем, нейронных сетей, нечеткой логики и т.д. не ориентированы на нахождение решения в виде компьютерных программ. Вместо этого они имеют дело с узкоспециализированными структурами, имеющими отношение к конкретному классу задач и определяемыми с помощью формальных грамматик, деревьев решений, матриц синаптических весов нейронных сетей, фреймов, продукционных правил или хромосом при использовании обычного ГА. Это, в свою очередь, существенно ограничивает возможности программирования ЭВМ, поскольку перечисленные структуры не обладают необходимой гибкостью, присущей такому уникальному объекту, как компьютерная программа [10].

Особенностью ГП является отсутствие различий между пространствами поиска и решений, все операции выполняются над самим фенотипом, который формируется из набора функций (functional set) и терминальных символов (terminal set) из предметной области задачи.

Множество функций может включать в себя:

- арифметические операции (+, - , * и т.д.);
- элементарные математические функции (sin, cos, exp, log и т.д.);
- логические операции (И, ИЛИ, НЕ);
- условные операторы (ЕСЛИ – ТО - ИНАЧЕ);
- функции, связанные с реализацией циклов (ВЫПОЛНЯТЬ ДО ТЕХ ПОР, ПОКА);
- другие функции в зависимости от решаемой задачи.

Множество терминальных символов обычно включает в себя переменные, а также числовые, символьные или логические константы.

Предположим, что необходимо реализовать функцию вида *F=A+B*abs(A)*, тогда указанное множество операций можно определить в виде {*+, *, abs*}, а множество терминальных символов - как {*A, B*}. На рис. 2.29 приведено графическое изображение данного выражения в виде дерева – иерархического представления ГП. Здесь внутренние вершины имеют смысл названных функций, а внешние (оконечные) вершины представляют собой терминальные символы.

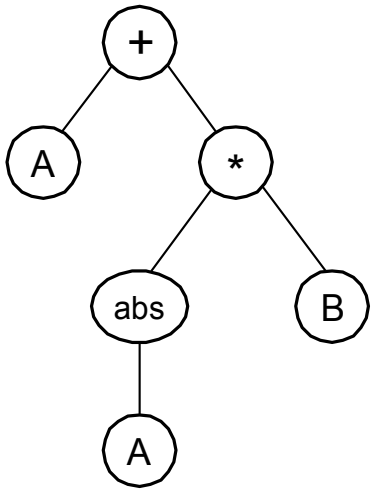


Рис. 2.29. Представление компьютерной программы с позиций ГП

Пространство решений при ГП представляет собой множество выражений, которые могут быть созданы при сочетании различных функций и различных терминальных символов. Наряду с этим пространством решений, может рассматриваться эквивалентное ему множество деревьев с упорядоченными ветвями, внутренние вершины которых соответствуют функциям , а внешние - терминальным

символам. Таким образом, ГП ищет оптимальное решение среди иерархически разветвленных программ переменной величины и сложности.

Строгая потребность в использовании подобного иерархического представления была вызвана необходимостью применения кроссинговера к хромосомам переменной длины. В подтверждение корректности данного подхода рассмотрим две хромосомы «*A+B/C*» и «*abs(A)/B+C*». В случае применения простого кроссинговера ГА с сайтом *k=1* будет получено две дочерние хромосомы: «*abs+B/C*» и «*AA/B+C*», что соответствует недопустимым и неверным выражениям. В свою очередь, иерархическое представление не разрушает синтаксические конструкции программы.

Подобно хромосомам ГА каждая компьютерная программа в популяции оценивается с помощью некоторого численного показателя, определяемого с помощью fitness-функции, показывающей качество работы данной программы с точки зрения поставленной задачи. Обычно качество может быть измерено с помощью ошибки, с точностью до которой происходит решение задачи, так как часто множество исходных значений, а также желаемый результат известны. Чем меньше эта ошибка, тем лучше компьютерная программа. Далее, следуя основным принципам эволюционного моделирования, программы с лучшими показателями работы отбираются и копируются в следующее поколение для применения к ним генетических операторов.

При этом предполагается, что отобранные таким образом программы имеют «правильно» функционирующие алгоритмические фрагменты и, следовательно, путем рекомбинации выбранных частей этих программ могут быть сформированы новые улучшенные их версии. В качестве примера реализации ГП рассмотрим задачу из класса «симуляторов искусственной жизни», идея которой заключается в следующем [3]. На тороидальной сетке размером 32x32 в 89 ячейках помещается «пища». Существуют некие препятствия , мешающие «насекомым» добраться до «пищи». «Насекомые» попадают на сетку из одной точки, и каждое движется согласно командам своей программы. В начальной популяции эти программы формируются

случайным образом из операторов, которые проверяют наличие препятствий и предписывают направление движения прямо, влево или вправо. Задается время жизни популяции (400 поколений). Качество каждой программы определяется числом перемещений, которые необходимо совершить, чтобы обойти все ячейки с «пищей». Каждая следующая популяция формируется традиционным для ГА способом. В итоге решение для популяции из 4000 «насекомых» было найдено за 20 итераций.

Рассмотрим основные стадии процедуры генетического программирования [25].

Инициализация. На этом этапе определяется множество терминальных символов, используемых функций, критерии оценки программ. Случайно генерируется начальная популяция древовидных программ. При этом в качестве корня дерева задается функция со случайными аргументами, листьев - переменные или константы. Для ограничения структуры дерева определяется его глубина, часто динамически (6 – для начальной популяции, 17 – для более поздних популяций).

Оценка решений. Результат выполнения каждой сгенерированной программы оценивается на основе выбранной fitness-функции, имеющей одну из нижеприведенных форм[10].

- «исходная» fitness-функция, определяемая в терминах решаемой задачи, например среднеквадратичная ошибка численного решения по отношению к эталонному значению предполагаемого математического выражения;

- «стандартизованная» fitness-функция

$$f_c(C_i) = \max(f) - f(C_i),$$

где *max(f)* – максимальное значение исходной функции оптимальности; в данном случае чем меньше величина $f_c(C_i)$, тем выше качество данной программы;

- модифицированная fitness-функция

$$f_м(C_i) = \frac{1}{1 + f_c(C_i)}.$$

Значения данной функции лежат в интервале [0, 1], она принимает большие значения для лучших программ в популяции;

- «нормализованная» функция пригодности.

$$f_н(C_i) = \frac{f_м(C_i)}{\sum_{i=1}^{Np} f_м(C_i)}.$$

Формирование новой популяции. Происходит с применением схем, аналогичных ГА, с учетом специфики ГП. Кроссинговер создает новые программы, составленные из частей, взятых у каждого из родителей.

Реализация кроссинговера начинается со случайного выбора по одному из узлов у каждого родителя, которые представляют собой точки кроссинговера для этих родителей. Первый из потомков образуется при этом путем удаления фрагмента дерева первого родителя ниже его точки кроссинговера и включения в эту точку фрагмента дерева второго родителя, также располагающегося ниже соответствующей точки (рис. 2.30).

Рассмотрим две родительские программы, составленные из функций {*+, /, abs*} на множестве терминальных символов {*A, B, C*}. Как видно из рис. 2.30, эти программы эквивалентны следующим выражениям: *F1=A+B/C*, *F2=abs(A)/B+C*. Предположим, что выполняется оператор кроссинговера к случайно выбранным узлам программ – родителей. Два потомка, полученные в результате работы кроссинговера (обмена участками программ), будут иметь форму следующих выражений: F1=*A+abs(A)*, *F2=(B/C)/B+C*.

Если точка кроссинговера одного из родителей совпадает с его терминальным символом, то фрагмент дерева другого родителя включается в место расположения этого терминального, а указанный символ – в точку кроссинговера второго родителя. Если же обе выбранные точки кроссинговера приходятся на терминальные символы, то операция кроссинговера выполняет перестановку этих символов из одного дерева в другое. В ситуации, когда в качестве точки кроссинговера выбрана корневая вершина одного родительского дерева, операция кроссинговера включит целиком дерево первого родителя в

дерево второго родителя в точке его кроссинговера. В этом случае первый родитель целиком станет фрагментом дерева (подпрограммой) внутри второго родителя. Фрагмент дерева второго родителя после его позиции кроссинговера станет при этом другим потомком.

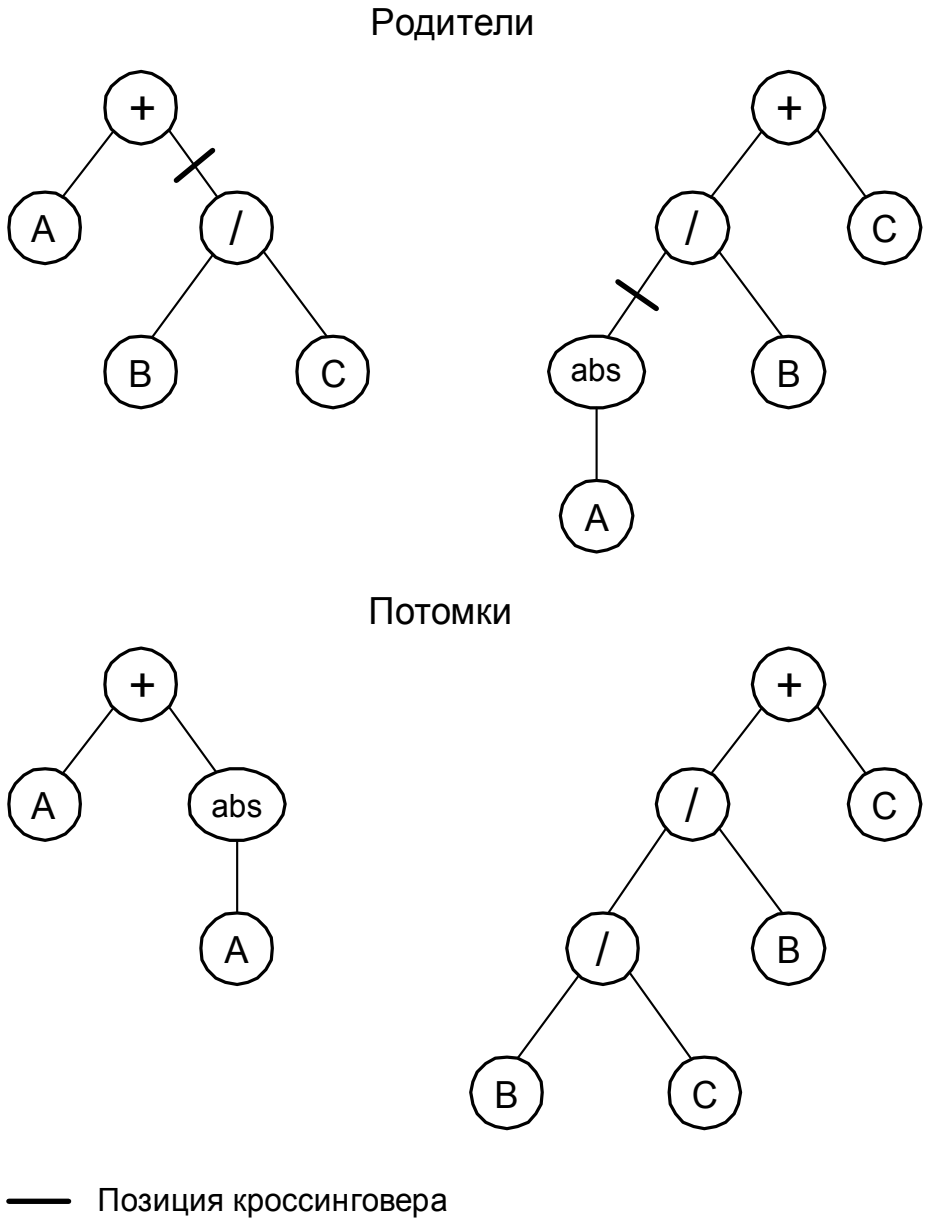


Рис. 2.30. Оператор кроссинговера ГП

В ряде случаев в ГП также может применяться модифицированный оператор мутации. В ходе своего выполнения он выбирает случайный узел дерева программы и удаляет все ветви, расположенные ниже него, заменяя при этом их случайно сгенерированным поддеревом (рис. 2.31). Однако по причине схожести выполняемых функций

операторов кроссинговера и мутации применение последнего часто является необязательным [10].

В процессе своего функционирования ГП оперирует символьными выражениями (S - выражениями), имеющими внутреннюю синтаксическую реализацию, сходную с языком LISP. Так, программы, изображенные на рис. 2.29 на языке LISP, будут записаны следующим образом: *(+A(/B C))*, *(+(/(abs(A)B)C))*.

Подобное представление дает возможность применять операторы языков программирования высокого уровня, включая операторы условия. Наличие условных операторов, с одной стороны, позволяет ввести доминантные и рецессивные аллели при выполнение кроссинговера, что предоставляет возможность выделить перспективные направления поиска, с другой - порождает проблему появления избыточного кода, который никогда не будет выполнен. Например, S – выражение *(IF (5>0) (+A B) (-A (/B C)))* всегда возвращает результат *A+B*, в то время как выражение *(-A (/B C))* никогда не будет обработано.

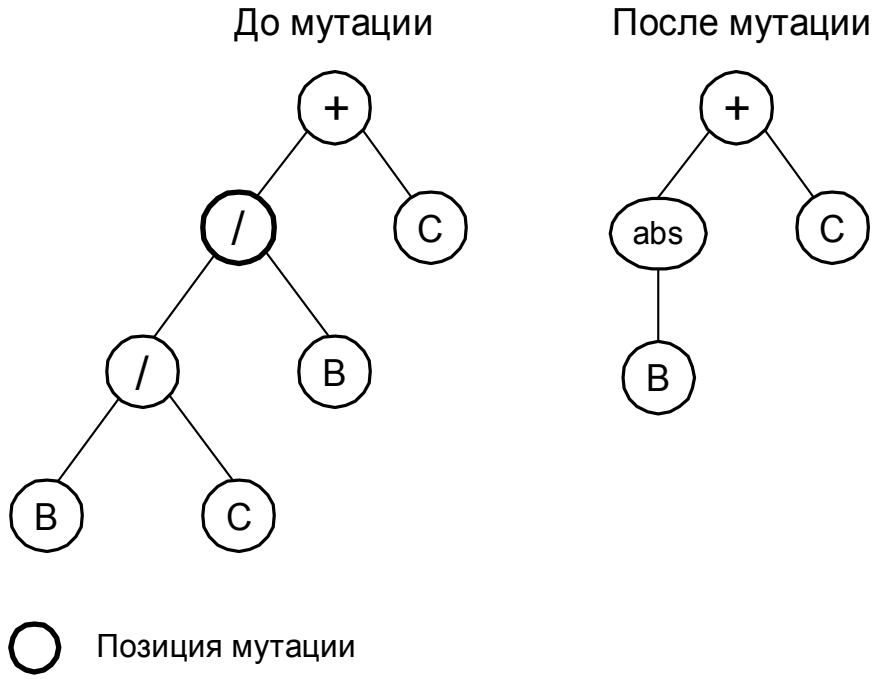


Рис. 2.31. Оператор мутации ГП

Решения с избыточным кодом могут привести к эффекту разрастания программ и снижению скорости процесса поиска в целом. Подобный эффект может быть уменьшен с помощью введения

специализированных штрафных функций применительно к размеру программ популяции.

Проверка условия останова ГП. Аналогично ГА, критериями прекращения работы ГП является стагнация поиска или достижение заданного числа поколений.

Рассмотрим возможности применения приведенной процедуры ГП для синтеза программы (выражения) для аппроксимации произвольной кривой, заданной в двумерной системе координат (рис. 2.32 (t = 0)) [19].

На рис. 2.32 показана последовательность поиска такого выражения, причем в качестве множества терминальных символов использовалась одна переменная { *x* }, а множества функций - только арифметические операции {+, -, *, \}. Размер популяции, вероятности применения генетических операторов кроссинговера и мутации были равны 100, 0,8, и 0,2 соответственно. Максимальная глубина древовидного представления выражений задавалась равной шести. Качество генерируемых выражений определялось как мера близости их значений и фактических значений в точках кривой. При этом от поколения к поколению наблюдалась тенденция постоянного уменьшения их различия. Условие окончания соответствует достижению максимального числа поколений *Ng* = 100 или правильностью работы программы в 10-ти случаях для различных значений заданных переменных при погрешности результата менее 0,01.

Процесс генетического программирования продолжался на протяжении *t* = 70 поколений, в течение которых графики генерируемых выражений и, следовательно, само выражение постепенно эволюционировали от совершенно не похожих на заданную кривую до относительно близких к ней. Факт постепенного приближения результата выполнения генерируемых программ к правильному свидетельствует о корректности работы ГП.

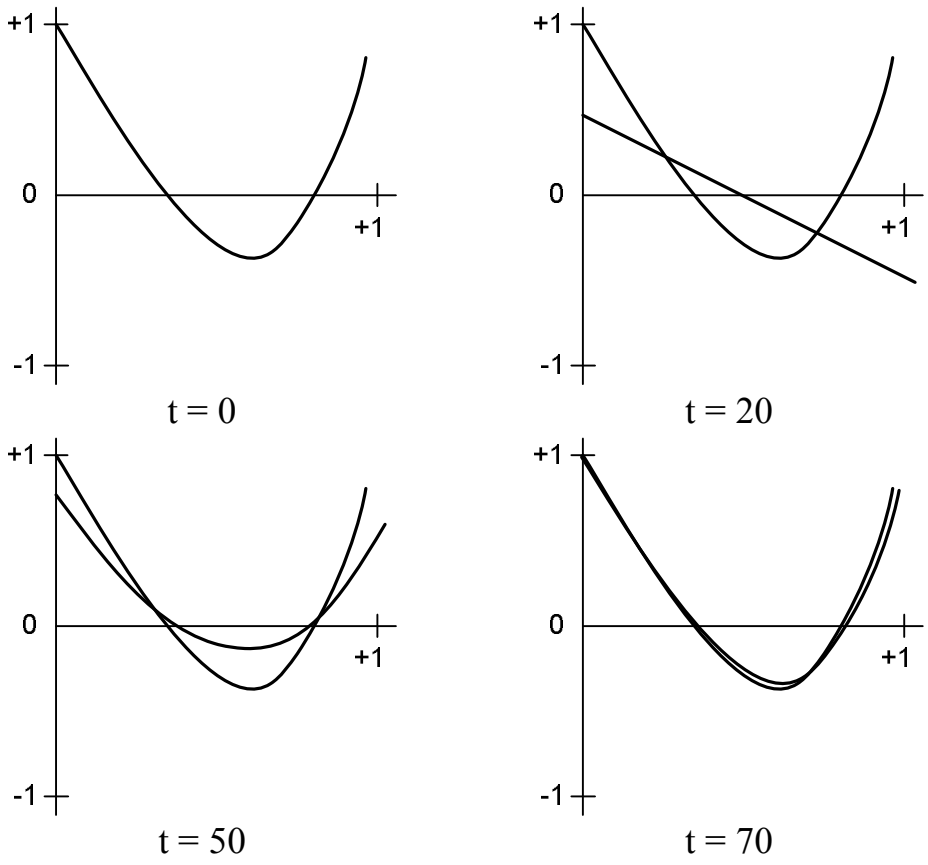


Рис. 2.32. Изменение графика аналитического выражения в процессе генетического программирования

На последнем поколении была получена формула вида (*(-1(* 4 x)) (+ (- (* (* x x) 0.4375) (* (* (* x x) x) 2)) 1)) или $(1-4*x)*(0.4375*x^2-2*x^3+1)$, достаточно точно описывающая заданную кривую. Более того, расширив выбранное функциональное множество дополнительными функциями, например тригонометрическими, могут быть синтезированы наряду с полученной и другие формы аналитических выражений этой кривой.

Заметим, что принципы ГП оказываются эффективными лишь для относительно простых задач синтеза программ, которым можно отнести и изложенную. При решении задач с более высокой вычислительной и структурной сложностью процесс ГП существенно усложняется. И связано это прежде всего с разрушающей способностью оператора кроссинговера, а также увеличением размера программ без изменения качестве их работы. Разрешением такой ситуации является генерация большого числа потомков с последующим выбором из них наилучших, использование автоматически определяемых функций,

внедрение фрагментов программ, не чувствительных к кроссинговеру. В последнем случае подразумеваются выражения вида *A =A**1, которые не влияют на функциональность программы и значение ЦФ, но повышают шансы сохранения полезных фрагментов программы от разрушения кроссинговером для будущих поколений. Примечательно, что в генетике также существует аналогичное явление, связанное с понятием интрона, который соответствует участку хромосомы, не хранящей полезной информации [33].

Кроме этого, существует проблема, связанная с математическим обоснованием ГП с позиции теоремы о шаблонах. Попытка использовать в качестве строительных блоков поддеревья привела к отрицательным свойствам кроссинговера в ГП: если в ГА, согласно теореме о шаблонах, данный оператор порождает потомков с функцией пригодности не хуже, чем у родителей, то в ГП ситуация может быть диаметрально противоположной. Данный факт послужил стимулом для создания новых операторов ГП, среди которых можно выделить операторы: перестановки (обмен двух символов в дереве), редактирования (оптимизация и сокращение S - выражения) и инкапсуляции (преобразование поддерева в узел) [60].

Расширением инкапсуляции является введение в ГП автоматически определяемых функций (АОФ) [61], позволяющих добавить эффект модульности в генерируемые программы. Как правило, система ГП настраивается на обработку предопределенного набора функций. Каждая функция может быть вызвана в программе, позволяя многократно использовать результаты вычисления АОФ без необходимости каждый раз выполнять их код, тем самым минимизируя опасность разрушения операторами ГП и повышая эффективность ГП.

На рис. 2.33 приведен пример решения с применением двух АОФ. Первая АОФ (*АОФ0*) принимает один аргумент и возвращает его куб, вторая (*АОФ1*) принимает два аргумента и возвращает выражение *1/(ARG0*ARG1)*. Основная программа выполняет сложение результатов вызова обеих функций с параметрами *A*, *B*. При преобразовании данной структуры в S – выражение будет получена следующая программа: *(+(*(*A A)A) (/ (1(*A B))))*. При необходимости в АОФ могут быть использованы вложенные вызовы других АОФ, что

позволяет моделировать такие программные конструкции, как итераторы, циклы, рекурсию, а также различные типы организации данных в памяти.

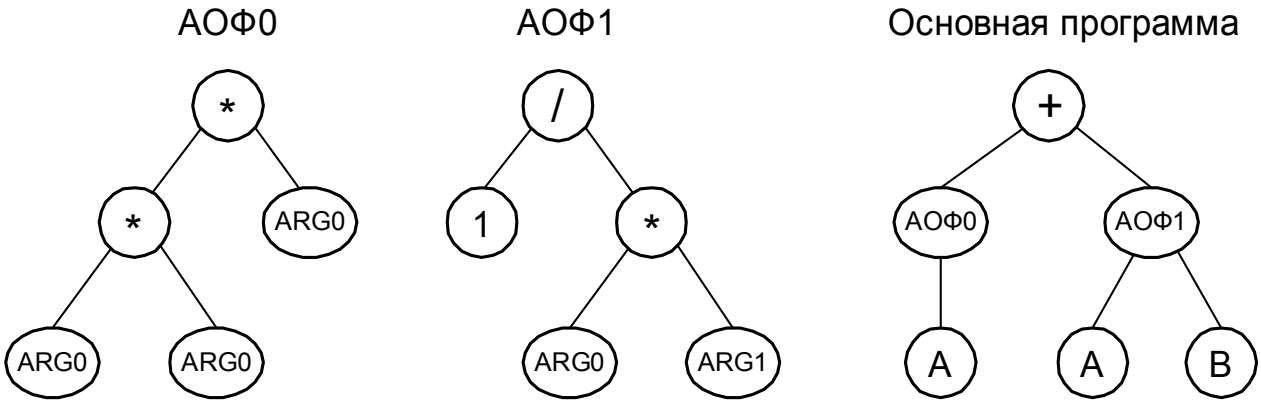


Рис. 2.33. Дерево программы с использованием АОФ

Исследования применения АОФ при ГП подтверждают повышение качества эволюционного моделирования для целого ряда задач повышенной структурной сложности [60].

2.3. Эволюционные стратегии

Эволюционные стратегии (ЭС) были разработаны в Германии в 1960 г. Одними из первых приложений этого направления эволюционного моделирования стали задачи оптимального проектирования формы колена трубопровода для минимизации сопротивления в местах стыковки, а также структурной оптимизации сопла двигателя. При этом первоначально подобные исследования проводились без применения компьютеров, то есть создавалась реальная физическая модель, которая затем исследовалась, модифицировалась за счет изменения позиций, добавления или удаления сегментов конструкции [48]. Также известно, что концепция ЭС применялась в одной из технологий промышленного менеджмента, которая называлась эволюционной операцией. Ее выполнение заключалось в систематическом тестировании альтернативной технологии производства продукции за счет незначительных модификаций стандартных параметров производственного процесса [46].

Первый компьютерный алгоритм ЭС был предложен в 1965 г. и затем усовершенствован в 1973 И. Риченбергом (I. Rechenberg) [65]. Изначальное представление ЭС имело достаточно простую форму, известную как бинарная ЭС. В ней использовалось всего два объекта – родитель и потомок. Подобно ГА в эволюционных стратегиях все манипуляции с объектами происходят напрямую с их фенотипом при четком разделении поколений родителей и потомков.

Главным источником синтеза решений в ЭС является оператор мутации, который применяется к значениям параметров фенотипа родителя. При этом случайная величина такой модификации определяется по нормальному закону распределения с нулевым математическим ожиданием и моделирует переход между точками пространства решений. Мутация применяется независимо к каждому члену популяции родителей и если получившийся потомок имеет лучшее значение fitness-функции, то он переходит в следующее поколение уже в качестве родителя. В противном случае процедура мутации родителя повторяется. Такая схема с одним родителем и потомком известна как (1+1) – ЭС и обладает рядом недостатков, вызванных бинарной схемой работы, например медленную скорость поиска решений.

Дальнейшее совершенствование технологий эволюционных стратегий связано с разработкой в начале восьмидесятых годов прошлого столетия ЭС, использующих идею популяции решений [46]. Они получили название $(\mu + \lambda)$ – ЭС и (μ, λ) – ЭС, где μ определяет число родителей, а λ - число потомков.

В $(\mu + \lambda)$ – ЭС μ родителей порождает λ потомков, среди общего количества которых происходит отбор μ лучших особей в следующее поколение. В (μ, λ) – ЭС в отборе участвуют только сгенерированные потоки, поэтому для сохранения численности популяции и выполнения условия $\lambda > \mu$ рекомендуется использовать соотношение родитель : потомки как 1:7. Теоретические исследования этих модификаций ЭС позволили установить, что при их использовании скорость поиска увеличивается логарифмически по сравнению с классическим вариантом.

Наряду с оператором мутации, в этих схемах ЭС применяется оператор рекомбинации, аналогичный кроссинговеру в ГА. При этом

компоненты вектора «потомка» создаются из компонент векторов двух родителей, например, следующими способами [3]:

- компоненты вектора потомка выбираются случайным образом из векторов родителей;

- компоненты вектора потомка формируются как средние арифметические значения компонент обоих родителей.

Ниже приводятся основные этапы работы эволюционных стратегий на основе использования популяций.

Шаг 1. ***Инициализация.*** Создается и случайно заполняется начальная популяция, в том числе посредством мутации единственного решения, введенного пользователем.

Шаг 2. ***Отбор родителей.*** Для генерации потомков случайно выбираются родители из популяции родителей.

Шаг 3. ***Применение генетических операторов.*** До достижения заданного размера популяции потомков λ выполняется оператор рекомбинации, к результатам которого применяется оператор мутации.

Шаг 4. ***Оценка решений.*** Определяется приспособленность полученных особей для определения кандидатов в следующую популяцию.

Шаг 5. ***Формирование новой популяции.*** Формируется новая популяция из потомков (в случае (μ, λ) – ЭС) или из популяций родителей и потомков (в случае $(\mu + \lambda)$ – ЭС).

Шаг 6. ***Проверка критерия останова ЭС.*** Процесс останавливается, если выполнены условия останова или продолжается с шага 2 – в противном случае.

Для придания мутации направленного характера в ЭС могут быть использованы различные эвристики, а также такие параметры, как стандартное отклонение σ и угол поворота α в пространстве решений, позволяющие адаптивно изменять направление поиска. Модифицируя эти параметры, операторы ЭС получают возможность вести поиск внутри выделенного пространства решений и достаточно быстро находить оптимальное решение путем сужения поискового пространства (рис. 2.34) [48].

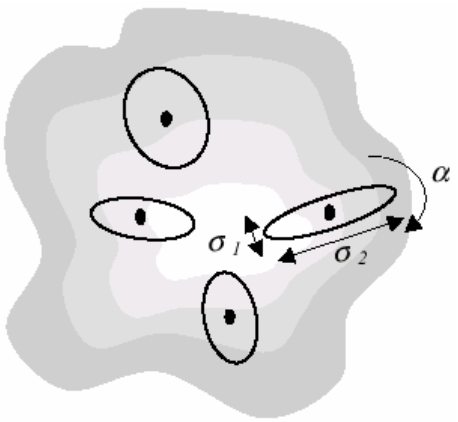


Рис. 2.34. Гиперэллипсоиды мутации, определяющие одинаковую плотность вероятности вокруг каждого решения

В [65] выполнен анализ степени сходимости для бинарной ЭС применительно к двум целевым функциям, на основе результатов которого определен оптимальный диапазон изменения σ и значения вероятности для успешной мутации. Также было сформулировано «правило успеха»: отношение успешных мутаций к общему числу мутаций должно быть равно 0,25. При превышении этого предела увеличивается диапазон изменения, при уменьшении диапазон изменения σ уменьшается.

Для применения данного правила ЭС используют накопленную информацию об отношении успешных мутаций к общему числу мутаций в течение $10 \cdot m$ эволюций. Согласно этому правилу, диапазон изменения параметра σ может увеличиваться, уменьшаться или оставаться неизменным. На основе применения цепей Маркова было доказано, что (1+1) – ЭС сходятся с вероятностью, равной единице при условии $\sigma > 0$ [46].

2.4. Эволюционное программирование

Принципы эволюционного программирования (ЭП) похожи на ЭС, однако ЭП было разработано независимо от них в 1960-х годах Л. Фогелем (L. Fogel) [39]. Изначально целью ЭП было исследование работы конечных автоматов, качество работы которых

определялось степенью соответствия выходной последовательности символов заданной.

Считается, что цель ЭП как эволюционной технологии заключалась в организации процедуры создания машинного интеллекта. Под интеллектуальным поведением подразумевалась возможность прогнозирования диаграммы конечных состояний, переводящих автомат к поставленной цели наиболее эффективным способом. В процессе эволюции использовалось единичная популяция решений, а основным оператором репродукции была мутация.

В конце восьмидесятых годов в работах [54, 55] основные идеи ЭП были пересмотрены и развиты. Главное отличие расширенного ЭП заключалось в возможности применения для решения задач непрерывной оптимизации с фиксированной длиной вектора вещественных параметров. Также важной особенностью ЭП стала возможность самоадаптации средствами направленной мутации подобно ЭС.

В настоящее время можно выделить три основных типа ЭП:

- стандартное ЭП;
- мета – ЭП (в настоящее время, называемое просто ЭП);
- Рмета – ЭП.

Их отличие друг от друга состоит в использовании различных механизмов организации направленной мутации с целью повышения уровня самоадаптации и имитации эффекта от применения оператора кроссинговера.

Подобно ЭС и ГП эволюционное программирование не разделяет генотип и фенотип, а все манипуляции выполняет на уровне последнего. Рассмотрим порядок работы процедуры эволюционного программирования.

Шаг 1. ***Инициализация.*** Задаются исходные данные решаемой задачи, посредством определения входного словаря, множества состояний, условий переходов между ними. Соответствующие параметры инициализируются равномерными значениями из предопределенных диапазонов их области определения. Случайным образом формируется начальная популяция конечных автоматов.

Шаг 2. ***Оценка решений.*** Объекты текущей популяции оцениваются с помощью выбранной fitness-функции.

Шаг 3. ***Формирование новой популяции.*** Заключается в итерационном выполнении следующей последовательности действий до увеличения размера популяции в два раза:

- с помощью турнирной схемы отбора выбираются родители из случайно сгенерированной группы;
- генерируется потомок посредством мутации копии родителя.

Шаг 4. ***Усечение популяции.*** Оценивается качество работы каждого из потомков, после чего удаляется половина популяции с наименьшими значениями fitness-функции.

Шаг 5. ***Проверка критерия останова.*** Если условия завершения эволюции выполнены, то работа ЭП прекращается, в противном случае осуществляется переход к шагу 2.

Из приведенного алгоритма работы ЭП видно, что единственным источником эволюционных преобразований является оператор мутации. Он играет роль механизма самоадаптации в пространстве конечных автоматов и может иметь следующие способы реализации [3]:

- изменение заключительного состояния;
- изменение условия перехода из одного состояния в другое;
- добавление нового состояния;
- удаление состояния;
- изменение начального состояния.

Для придания оператору мутации свойств кроссинговера и стимулирования направленного поиска вместо нормального распределения в процессе мутации может быть использовано распределение Коши [48]. Это мотивируется анализом, основанным на изучении окрестности решений и размера шага в пространстве поиска в процессе сравнения двух операторов мутации с разными распределениями. Теоретические и практические результаты исследований показали, что Коши-мутация эффективней при нахождении ближайшего к оптимуму решения, когда расстояние между текущей поисковой позицией и самим оптимумом велика, применение же Гауссовой мутации целесообразно в условиях малых дистанций.

Несмотря на то, что теоретические основы ЭС и ЭП близки, в работе [45] был проведен независимый анализ различных форм ЭП, включая случай, когда популяция состоит из одной особи индивидуума. Было выявлено, что операция мутации увеличивает или уменьшает значение переменной решения на величину, не превосходящую квадратный корень из оценки качества решения. Также было доказано, что ЭП имеет вероятность нахождения решения, равную единице, но при этом время сходимости и некоторые другие особенности ЭП остаются открытым вопросом [15].

Рассмотренные формы эволюционных алгоритмов являются главными инструментами моделирования эволюционных процессов с помощью компьютерных программ. Несмотря на ряд свойственных им различий в представлении информации, функциональности и спектре решаемых задач (табл. 2.8) [34], но учитывая единство законов их функционирования, можно говорить о них как о проявлении возможностей некоторого обобщенного эволюционного алгоритма.

Таблица 2.8

Сравнительный анализ эволюционных алгоритмов

Критерии сравнения / виды ЭА	**Представление решений**	**Целевая функция**	**Отбор**	**Рекомбинация**	**Мутация**
ГА	Двоичное, десятичное	Скаляр	Стохастический	Основной оператор	Вспомогательный оператор
ГП	Программное	Скаляр	Стохастический	Основной оператор	Факультативный оператор
ЭС	Вещественное	Вектор	Детерминированный	Факультативный оператор	Основой оператор
ЭП	Вещественное	Вектор	Детерминированный	Не используется	Единственный оператор

Этот алгоритм позволяет не только воспринимать эволюционные вычисления как единое целое, но и порождать на его основе новые технологии эволюционного моделирования. Далее рассматривается архитектура такого эволюционного алгоритма.

2.5. Общая архитектура эволюционных алгоритмов

Согласно фундаментальной теории биологического развития [50] для начала эволюционного процесса необходимо наличие следующих факторов: репродукция, наследование, изменение, селекция. Другими словами, эволюция продолжается до тех пор, пока особи воссоздают себя в потомках, которые наследуют их родительские характеристики с некоторыми изменениями, пока некоторая предпочтительная в данных условиях форма селекции отбирает особи для репродукции. Алгоритмы эволюционного поиска не являются здесь исключением.

Все ЭА выполняют репродукцию членов популяции либо прямым копированием родителей, либо используя операторы рекомбинации и мутации, тем самым обеспечивая наследование с изменением свойств. Данные операторы могут решать различные задачи от простой инверсии случайно выбранного бита до полноценного алгоритма локального поиска. Также все ЭА применяют некоторый вариант селекции для определения, какие решения будут иметь возможность репродукции, а какие нет. Здесь ключевым свойством селекции является влияние на концентрацию эволюционного процесса в направлениях к особым областям поискового пространства. Тем самым должны быть определены особи, обладающие большей вероятностью иметь потомство по сравнению с другими членами популяции.

В отличие от естественной биологической эволюции, ЭА для своего выполнения требуют наличия таких существенных признаков, как инициализация, оценивание, завершение.

Если в естественной эволюции процесс продолжается без остановки, переходя из одной формы в другую, то ЭА, как правило, начинают свою работу с инициализации начальных решений, имеющих фиксированную структуру и интерпретацию, случайными значениями. Оценивание в ЭА отвечает за обеспечение направленности эволюции к лучшим решениям. Также в сравнении с биологической эволюцией ЭА не обладают реальной средой, в которой могут быть проверены выживаемость полученных решений, вместо этого они полагаются исключительно на аналитическое исследование новых вариантов решений.

В естественной эволюции вымирание является гарантированным путем ее завершения, в то время как в ЭА не может быть применена данная особенность, поскольку в этом случае все полученные решения будут утеряны. Вместо этого применяется детерминированный критерий окончания поиска, как правило, он представляет собой факт нахождения «хорошего» решения или генерацию предопределенного числа поколений.

Можно выделить две других важных операции ЭА, которые, хотя и не являются обязательными для инициации или управления ходом эволюции, позволяют существенно улучшить возможности эволюционного поиска. Данные операции включают отображение и миграцию.

В настоящее время среди ЭА только ГА отделяет пространство поиска (содержит генотипы) от пространства решений (содержит фенотипы), а также применяет операцию неявного отображения между этими пространствами. Данная особенность позволяет достаточно эффективно определять сложные решения на основе компактного набора инструкций, включающих обработку ограничений и коррекцию ошибок в ходе работы ГА. В ряде ЭА может применяться модификация, оперирующая более чем одной популяцией, в этом случае наблюдается так называемая миграция (внедрение) особей из одной популяции в другую в ходе эволюционного процесса, тем самым предоставляя возможность относительно независимой, параллельной модификации членов популяции с частичным разделением используемой генетической информации. Подобный генетический аппарат позволяет не только реализовать ЭА на параллельных вычислительных архитектурах, что приводит к повышению скорости поиска путем снижения числа генерируемых поколений, но и повысить качество получаемых решений.

На рис. 2.35 представлена общая структурная схема архитектуры ЭА, которая может рассматриваться как единая основа эволюционных алгоритмов, но не как сам алгоритм, поскольку большинство ЭА использует только подмножество представленных на схеме этапов (неиспользуемый этап отмечен как факультативный). Так, в эволюционном программировании существуют следующие этапы:

инициализация, оценивание, отбор, репродукция, замещение и завершение. С другой стороны, ГА использует инициализацию, отображение, оценивание, отбор, определение мощности репродукции (сколько потомков может иметь каждый родитель), завершение. Ниже приводится анализ каждого из этапов общей архитектуры ЭА.

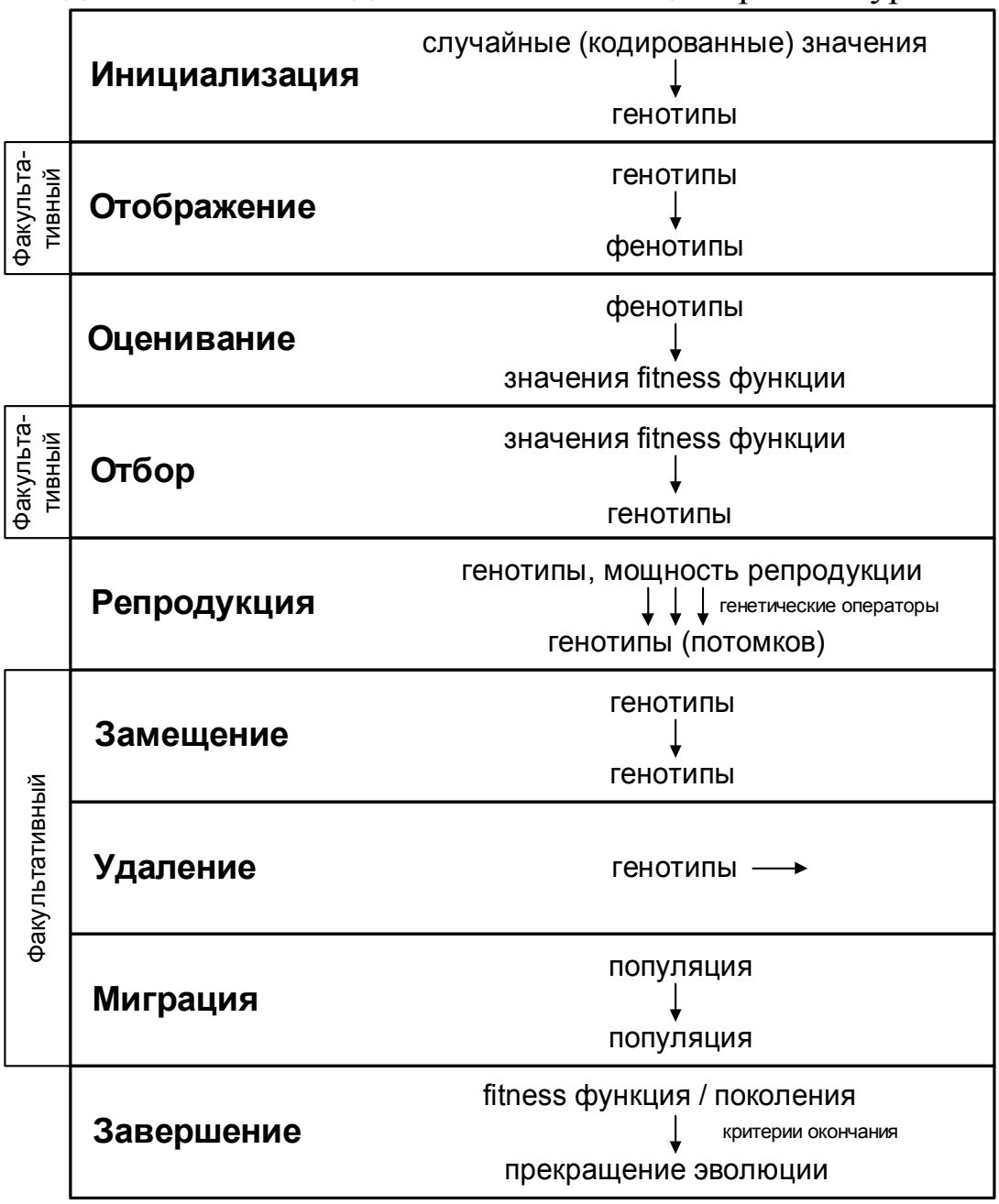


Рис. 2.35. Общая архитектура ЭА

Инициализация. Как уже отмечалось, работа ЭА начинается с инициализации случайными значениями исходной популяции, в случае с ГА это разделение пространства поиска и решений, поэтому генотип каждого индивидуума инициализируется случайными

аллелями. Иногда начальная популяция формируется из случайных мутаций решения, введенного пользователем. Часто случайные значения генерируются в пределах заданного диапазона, что представляет собой одну из форм обработки ограничений. В этом случае в ходе инициализации любые сгенерированные решения, не удовлетворяющие ограничениям, удаляются и заменяются новыми.

Многие сложные задачи часто требуют альтернативных методов инициализации. Некоторые исследователи, в частности Дж. Коза, предлагают использовать в ЭА так называемые эмбриогены (низкоуровневые строительные блоки) – упрощенные детерминированные формы решений, которые затем используются в качестве стартовой точки для получения более сложных решений. Несмотря на то, что в большинстве оптимизационных задач используются решения с фиксированной структурой (фиксированное число параметров решения), в задачах, решаемых средствами ГП, могут также модифицироваться количество и организация параметров в дополнение к изменению их значений. Для подобных алгоритмов инициализация может включать как генерацию случайных значений, так и структур.

Отображение. Среди эволюционных алгоритмов только ГА требует выполнения этапа отображения для преобразования генотипа в фенотип вследствие разделения пространств поиска и решений. Как правило, процесс отображения заключается в конвертировании двоичных аллелей генотипа в десятичные значения параметров фенотипа.

Для других ЭА нет потребности в модификации и отображении генотипов, то есть достаточно, чтобы все операции выполнялись напрямую с фенотипом, при этом не нарушая принципов эволюции. В то же время в [50] отмечается, что, проводя аналогию с естественной эволюцией, где все изменения происходят на уровне гена, качество эволюционного поиска может быть улучшено при манипулировании непосредственно генотипом, например, в таком ориентированном исключительно на фенотип ЭА, как генетическое программирование. Можно выделить следующие основные преимущества использования этапа отображения в ЭА: сокращение размерности пространства поиска, представление сложно структурированных решений, обработка ограничений на начальных стадиях функционирования ЭА.

Оценивание. Для определения качества полученных решений к каждому фенотипу должна быть применена операция оценивания. Часто один запуск ЭА может включать выполнение тысячи подобных операций, что позволяет утверждать о большой вычислительной нагрузке, приходящейся на операцию оценивания, в связи с этим для большинства ЭА время вычислений является одним из критериев эффективности.

В основе выполнения операции оценивания лежит вычисление значения fitness-функции каждого решения. Fitness-функция может принимать произвольную форму представления (многопараметрическую, мультимодальную, дискретную, нечеткую и т.д.), при этом данная особенность не влияет на процесс функционирования ЭА. В ГА, использующих разделяемую генетическую информацию (в случае параллельной обработки нескольких популяций), fitness-функция может выполнять функцию масштабирования, то есть выявления групп (кластеров) индивидуумов в популяции, обладающих схожими характеристиками. Кроме этого, результаты оценивания не всегда определяются только fitness-функцией, в ряде случаев может применяться человеческий фактор для принятия окончательного решения.

Отбор родителей. Родительские решения являются основой для генерации новых решений – потомков. Однако применение специализированных процедур отбора родителей не существенно для эволюции, то есть в общем случае родители могут быть выбраны равновероятно либо случайно. Вообще, для эволюции считается достаточным наличие таких явлений, как замещение особей и их вымирание, но в этом случае она будет носить исключительно случайный характер. Отбор лучших решений в родители следующего поколения представляет собой самый прямой путь к последовательному улучшению популяций. Здесь могут быть применены ранее отмеченные методы отбора.

При анализе некоторых ЭА, наряду с процедурой отбора родителей, применяется такое понятие, как мощность репродукции. Мощность репродукции определяет число потомков, которое может иметь родитель. Индивидуумы в популяции могут обладать низкой или даже нулевой мощностью репродукции, например при выборе

родителей из разных популяций. В большинстве ЭА мощность репродукции фиксирована – один родитель порождает одного потомка.

При решении прикладных задач в ЭА, помимо функции пригодности, могут использоваться ограничения, накладываемые на генерируемые решения. Наличие ограничений в ЭА может привести к ситуации необходимости выбора решения, скорее, удовлетворяющего всем требованиям (ограничениям), чем обладающего лучшей оптимальностью, при этом ограничения могут быть не связаны со спецификой задачи, а определяться особенностями ЭА. Также степень соответствия ограничениям может влиять на мощность репродукции, так, при использовании ГА с кодированием генотипа на основе перестановок количество потомков будет зависеть от удовлетворения соответствующим ограничениям.

Репродукция. Этап репродукции отвечает за генерацию решений - потомков на основе родительских решений, здесь ключевым моментом является тот факт, что потомки должны наследовать характеристики родителей с некоторой долей вариации. Это достигается путем применения таких генетических операций, как рекомбинация и мутация. Процесс рекомбинации заключается во взаимной перестановке элементов двух или более родительских решений. Большинство ЭА (кроме ЭП) осуществляют рекомбинацию посредством операторов кроссинговера.

В отличие от оператора кроссинговера, мутации в каждый момент времени подвергается только одно решение. У разных ЭА процедура мутации отличается, если у ГА мутации подвергаются новые решения после выполнения операции рекомбинации, то у ЭП мутация – первичный и основной способ порождения новых решений. В настоящее время существуют и разрабатываются различные варианты операторов кроссинговера и мутации, их особенности в основном зависят от области применения ЭА.

Следует отметить, что важным свойством обеих операций является «неразрушение» генетического материала. Несмотря на то, что всегда существует различие между родителем и потомком, эти изменения не вызывают существенной модификации фенотипов в целом, то есть в пространстве решений дочерние решения всегда будут

располагаться близко к их родителям. В противном случае, когда наблюдается существенное различие в характеристиках родителей и потомков, возрастает вероятность случайной локации последних в пространстве решений. Эффективный эволюционный поиск подразумевает сохранение и передачу полезной генетической информации между поколениями, избыточное разрушение данной информации превращает ЭА в алгоритм случайного (слепого) поиска.

В то же время проблема разрушения полезной генетической информации остается актуальной для некоторых ЭА, ее преодоление достигается путем использования специальных методов, так, в ГП применяются специализированные операторы кроссинговера, мутации; в ЭС и ЭП для уменьшения последствий мутации ее параметры определяются законом нормального распределения.

Замещение. Каждый раз после генерации потомка он должен быть размещен в популяции. Как правило, ЭА используют популяции фиксированных размеров, то есть добавление нового объекта сопровождается исключением из популяции какого-либо существующего. Часто процедура замещения выполняется неявно, но в ряде случаев в ЭА может использоваться оператор замещения, задачей которого является не только размещение особей в новой популяции, но и определение членов популяции, замена которых наиболее целесообразна. Подобная операция может быть выполнена на основе fitness-функции или степени подобия генотипов, а также соответствия ограничениям.

Замещение может рассматриваться как операция, обратная селекции, то есть если последняя отбирает индивидуумы для репродукции, то процедура замещения определяет кандидатов на вымирание, с позиций естественной эволюции замещение полностью подтверждает тезис «выживает сильнейший».

Кроме замещения, процедура исключения из популяции неэффективных особей может быть выполнена операцией удаления. В отличие от замещения, которое выполняется на основе сравнения нескольких решений, операция удаления манипулирует одним решением, если оно не соответствует критерию допустимости. Таким образом, задача операции удаления заключается не в стимулировании процесса генерации более лучших решений посредством эволюции, а

в моментальной «очистке» популяции от недопустимых решений. Так, в ЭП после генерации потомков операцией удаления исключается худшая половина всей популяции. В общем случае простое удаление представляет собой достаточно грубый способ поддержки эволюционного процесса и может привести к сокращению разнообразия и увеличению сложности эволюционного поиска.

Миграция. Одной из важных особенностей любого ЭА является параллельный характер работы эволюционного поиска. Это гарантирует получение оптимального решения после определенного числа поколений. Данный результат является вполне приемлемым в случае унимодальной целевой функции. Если же целевая функция мультимодальна, то часто возникает потребность в нахождении всех возможных оптимальных решений. Данная задача может быть решена средствами параллельных или распределенных ЭА. В подобных ЭА наблюдается разделение особей на множество отдельных непересекающихся подпопуляций (групп) с последующим независимым выполнением эволюционного поиска в каждой из групп. Таким образом, может быть достигнута ситуация, когда в каждой из подпопуляций, ЭА сойдется к одному из потенциальных оптимумов.

При этом, если между подпопуляциями отсутствует взаимодействие, процесс функционирования эквивалентен одновременной работе множества ЭА. Однако для повышения эффективности в параллельных ЭА рекомендуется выполнять операцию миграции для обмена генетической информацией между группами особей, кандидаты для миграции могут быть выбраны случайно либо на основе значений fitness-функции. Данная операция основывается на таких параметрах, как уровень миграции, расписание миграций, топология связи между подпопуляциями и определяет качество поиска и эффективность алгоритма. Частая миграция приводит к обмену потенциально полезным генетическим материалом, но в этом случае отрицательным эффектом является увеличение затрат на обмен информацией, поэтому необходим баланс между уровнем миграции и качеством решений.

Завершение. Процесс работы ЭА завершается при выполнении критерия окончания, который обычно основывается на достижении требуемого качества полученных решений либо на истечении заданного периода времени (числа поколений).

Выбор первого критерия окончания обусловлен при необходимости получения требуемых решений независимо от временных затрат на эволюционный поиск, при этом некоторые ЭА при достижении тупиковой ситуации (например, сходимость к локальному оптимуму) автоматически осуществляют реинициализацию параметров ЭА и рестарт эволюционного процесса.

Для алгоритмов, которые используют трудно вычисляемые fitness-функции или задачей которых является быстрая генерация решений, первичным критерием выступает временной фактор. Подобный критерий может включать в себя допустимое число поколений, число выполнения определенных операций или временные характеристики.

С целью уменьшения числа лишних поколений в ЭА может быть предусмотрена оценка сходимости в процессе эволюции и в случае подтверждения сходимости, то есть когда генотип, фенотип или fitness-функция не изменяются в течение ряда поколений, эволюционный процесс досрочно прекращается. Кроме этого, останов поиска может быть инициирован пользователем, например, при необходимости интерактивного выбора решений при отсутствии явного критерия окончания.

Таким образом, ЭА являются очень гибкими и нетребовательными алгоритмами с точки зрения корректности постановки задачи для его применения – будучи плохо реализованным или снабженным исходными данными, ЭА даже в этом случае позволяет получить приемлемые результаты [48]. Что же касается нахождения глобального оптимума, то здесь все зависит от таких факторов, как соотношение размера популяции к числу оптимизируемых параметров, вида пространства поиска, выбора способа кодирования переменных и формирования из них популяции, формы представления управляющих параметров. При этом у разработчика ЭА всегда остается выбор между максимальной скоростью сходимости, с одной стороны, и глобальностью результатов, с другой.

В заключение на основе приведенного анализа методов эволюционного моделирования сформулируем основные достоинства и недостатки их применения.

К достоинствам эволюционных алгоритмов принято относить:

- широкую область применения, особенно для задач, где отсутствуют классические методы их решения;
- эффективное решение комбинаторных и смешанных задач оптимизации без ограничений на математическую модель оптимизации;
- возможность синтеза алгоритмов поведения искусственных систем;
- доступность алгоритмизации и интеграции с другими технологиями искусственного интеллекта;
- возможность распараллеливания вычислительного процесса, а также его аппаратной реализации;
- и др.

Недостатки ЭА во многом пересекаются с другими оптимизационными процедурами (отсутствие гарантии нахождения глобального оптимума, вычислительная трудоемкость, необходимость корректировки целевой функции (конструирование fitness-функции)), однако они нивелируются отмеченными достоинствами эволюционных вычислений. Подтверждением этого может служить постоянное расширение коммерческих областей применения ЭА, а также их постоянное совершенствование.

ГЛАВА 3. ПРИМЕНЕНИЕ ЭВОЛЮЦИОННОГО МОДЕЛИРОВАНИЯ ДЛЯ РЕШЕНИЯ ЗАДАЧ ОПТИМИЗАЦИИ

В настоящее время эволюционные алгоритмы успешно применяются при решении различных задач математического моделирования и оптимизации в системах технического проектирования, автоматического управления и регулирования, коммуникационных и транспортных системах, при управлении в социально-экономических системах и др. [45, 48, 49], эффективное решение которых может служить наглядным свидетельством возможностей технологии ЭМ. Однако многие из них достаточно специфичны, и их понимание требует наличия знаний в соответствующей предметной области. В связи с этим в данной главе для демонстрации ЭМ выбраны известные задачи, позволяющие, с одной стороны, продемонстрировать возможности эволюционного моделирования относительно традиционных методов их решения, с другой – доступные для понимания широкому кругу потенциальных пользователей этой технологии искусственного интеллекта.

При решении задач для оценки временной эффективности ЭМ использовался компьютер следующей конфигурации Intel Pentium 4, 2,26 ГГц, ОЗУ 512 Мб.

3.1. Этапы решения задачи оптимизации на основе эволюционного моделирования

Необходимым условием применения эволюционных алгоритмов является наличие эволюционной модели решаемой задачи. В случае использования ГА процесс создания эволюционной модели включает в себя следующую последовательность действий.

1. Создание математической модели задачи с учетом ее последующего преобразования в эволюционную.

2. Определение количества, типа оптимизируемых переменных, а также формы их представления в хромосоме.

3. Конструирование fitness-функции для оценки решений.

4. Выбор или создание генетического алгоритма.

5. Настройка управляющих параметров ГА.

6. Анализ результатов работы ГА.

На рис. 3.1 схематично представлена последовательность выполнения этих этапов.

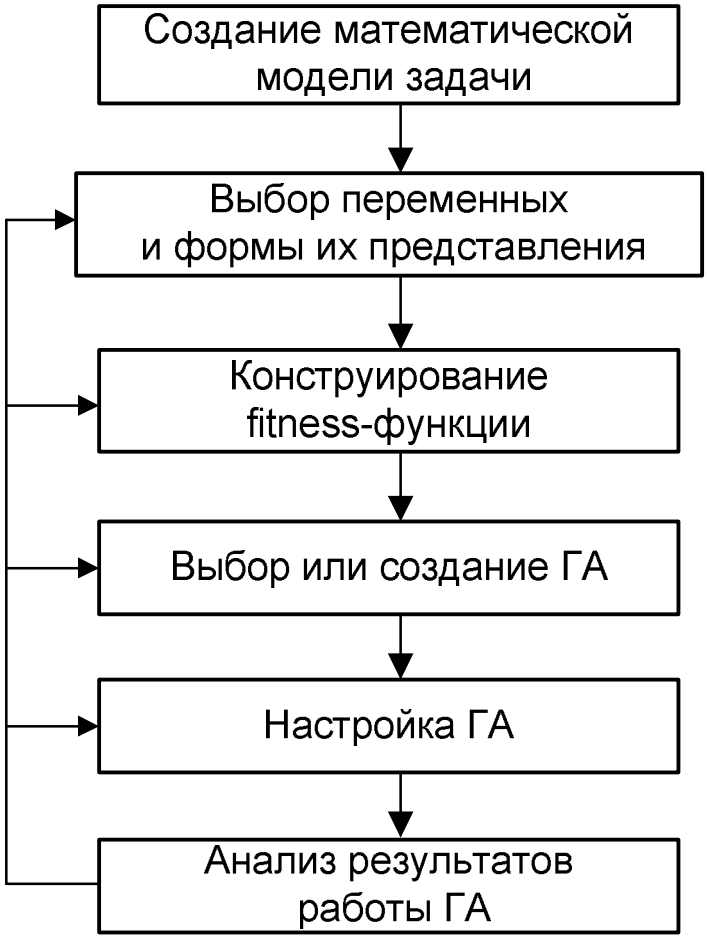


Рис. 3.1. Этапы решения задачи на основе ЭМ

Наличие на рисунке линий обратной связи означает, что не всегда созданная эволюционная модель является оптимальной для данной задачи и, следовательно, некоторые стадии ее формирования требуют внесения корректировок. Также, учитывая многообразие решаемых задач и форм реализации ГА, в приведенной схеме могут присутствовать дополнительные этапы, например, связанные с модификацией (разработкой) основных (новых) генетических операторов.

3.2. Решение задачи оптимизации функции многих переменных

Рассмотрим применение ГА к решению задач оптимизации нелинейной функции многих переменных вида

$$f(x) = \sum_{i=1}^{n} - x_i \sin(\sqrt{|x_i|}),$$

где $-500 \le x_i \le 500$, $n = 10$.

Данная функция относится к тестовым и предложена Ш. Швефелем (Schwefel's function 6)[67]. Она имеет более 10 миллионов локальных экстремумов, глобальный минимум расположен в точке x_i= 420,9687, $f(x^*)$ = - n*418,9829. Для визуальной оценки сложности этой функции на рис. 3.2 приведены её график и топология в случае двух переменных.

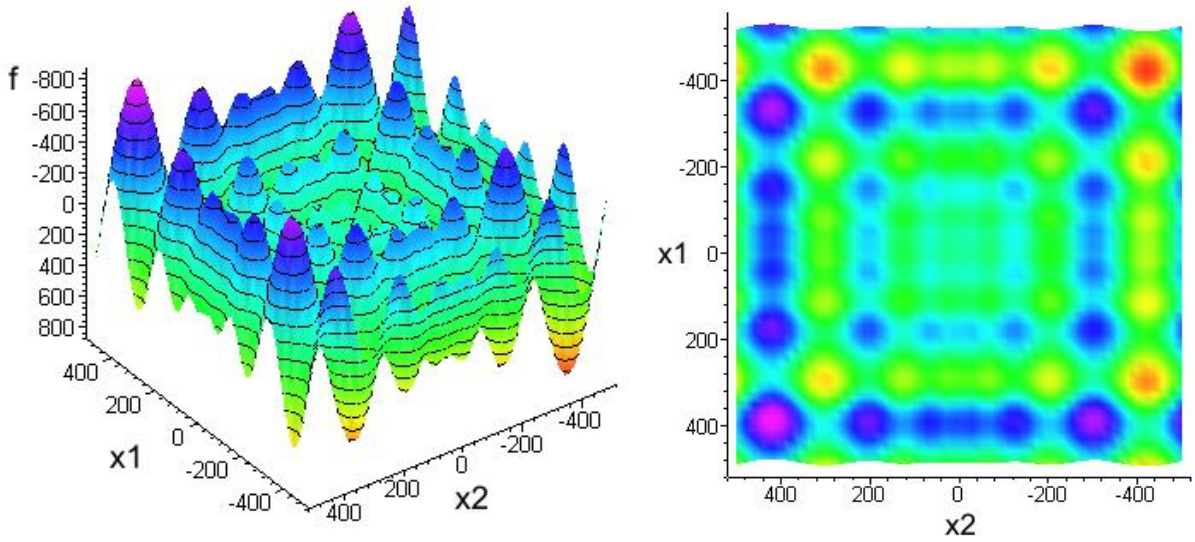


Рис. 3.2. Представление функции Ш. Швефеля для двумерного случая

Формирование математической модели. Для данного случая математическая модель задана изначально в виде оптимизируемой функции и ограничений на значения переменных.

Выбор переменных и формы их представления. В задаче имеется 10 переменных оптимизации, каждой из которых в эволюционной модели будет соответствовать ген в хромосоме. Для представления хромосом использовалось 16-ти битное кодирование переменных. В результате размер хромосомы составил 160 бит, а максимальная точность решения 0,01.

Конструирование fitness-функции. Для данной задачи fitness-функция эволюционной модели полностью повторяет ЦФ математической модели.

Выбор или создание генетического алгоритма. Для оптимизации использовался стандартный ГА с турнирной схемой отбора.

Настройка генетического алгоритма. Размер популяции и число поколений устанавливалось равным 100 и 300 соответственно. Для настройки ГА использовались различные комбинации значений управляющих параметров с целью оценить результаты направленного, случайно-направленного и случайного поиска. Для каждого случая генетический алгоритм запускался по пять раз. В качестве результата принималась комбинация значений *Pc*, *Pm*, для которых преобладало минимальное значение функции в большинстве случаев. При этом значений вероятности инверсии оставалось постоянным *Pinv* = 0,001. Результаты эксперимента приведены в табл. 3.1.

Таблица 3.1

Результаты эксперимента

Вероятность кросинговера	**Вероятность мутации**	**1**	**2**	**3**	**4**	**5**
0,9	0,001	**-4189.83**	**-4189.83**	**-4189.83**	**-4189.83**	**-4189.83**
0,8	0,005	-4185.76	**-4189.83**	-4188.81	**-4189.83**	-4177.84
0,7	0,01	**-4189.83**	-4189.39	-4136.50	-4185.14	-4188.67

Как видно из таблицы, ГА сумел найти точку глобального оптимума с точностью до сотых. При этом значение вероятностей применения ГА составило *Pc* = 0,9, *Pm* = 0,001. Время поиска в среднем составило менее 3 с. Также производилось варьирование численностью популяции: ее уменьшение приводило к ухудшению результата, а увеличение – к возрастанию времени поиска (без потери качества). На рис. 3.3, 3.4 представлены динамика изменения fitness-функции в процессе работы ГА, а также ее гистограмма (для случая оптимальных значений управляющих параметров). Анализ графиков позволяет сделать вывод, что уже в первой половине общего числа поколений ГА сумел локализовать область пространства решений, содержащую глобальный оптимум, а вторая половина поколений потребовалась для его уточнения.

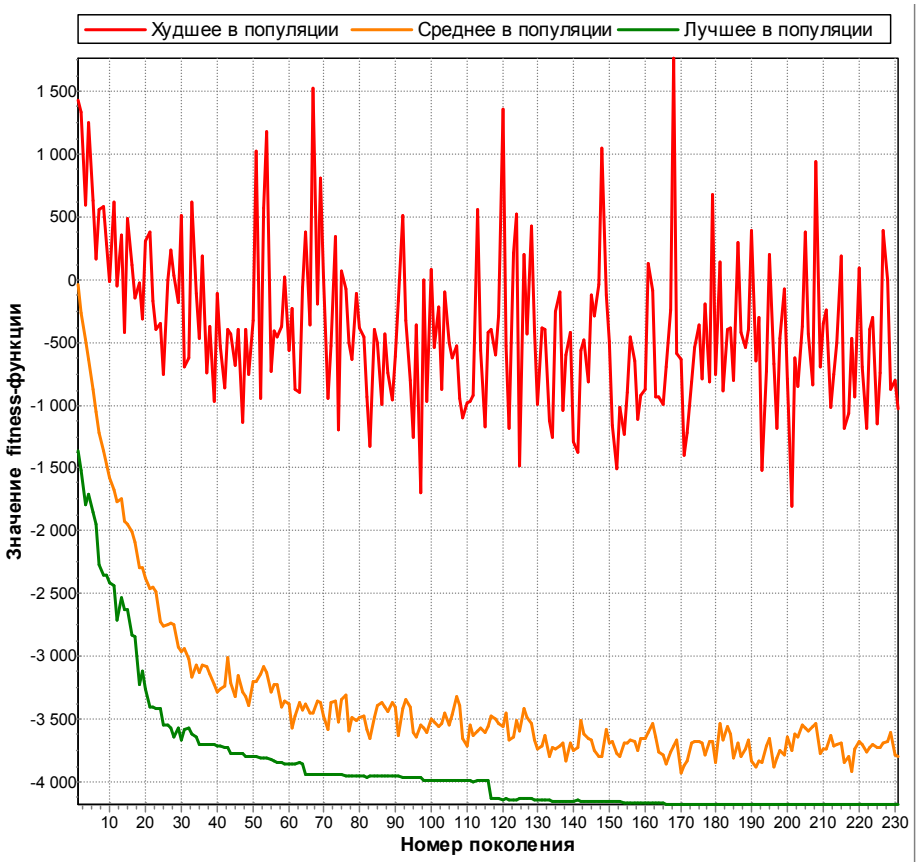


Рис. 3.3. Динамика изменения fitness-функции

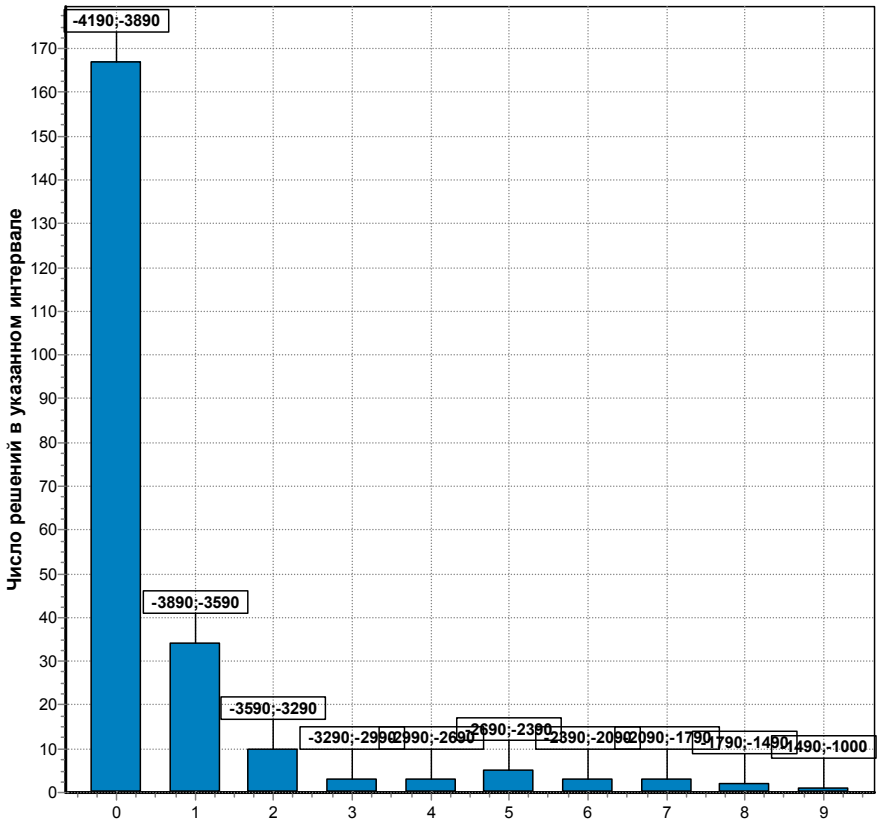


Рис. 3.4. Гистограмма fitness-функции

На рис. 3.5 показана динамика исследования пространства решений для двумерного случая на фоне топологии функции [19].

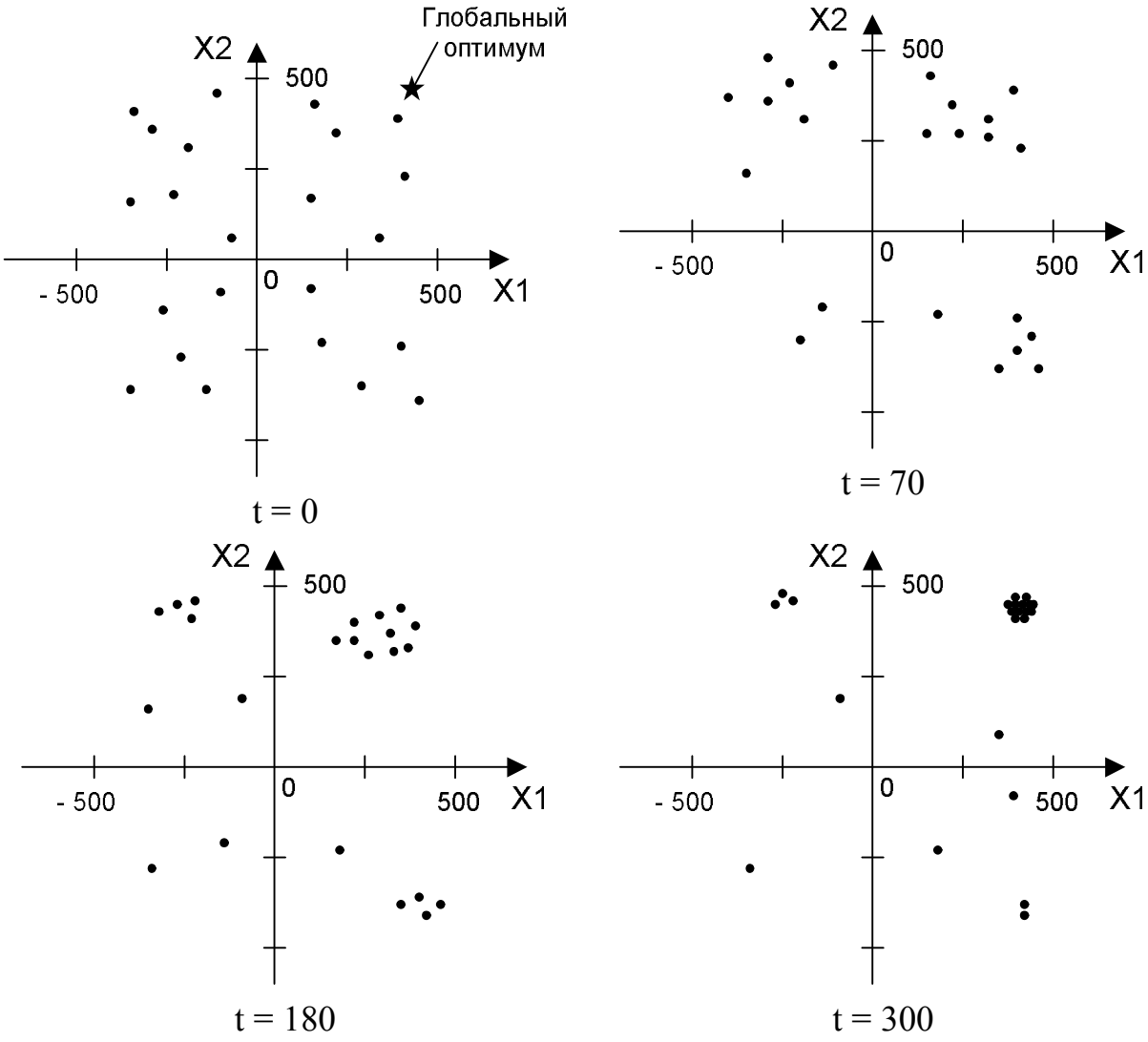


Рис. 3.5. Динамика исследования ГА пространства решений

В начальный момент времени (t = 0) популяция заполнена случайными решениями, распределенными почти по всему полю поиска. Далее под действием законов эволюционного моделирования происходит постепенное перемещение популяции в области с наилучшими (наименьшими) значениями целевой функции. При этом случайные изменения решений в популяции, вызываемые оператором мутации, приводят к перемещению между областями пространства решений, а применение кроссинговера – к обследованию этих областей. К поколению t = 180 в популяции теряется разнообразие и начинает проявляться эффект насыщения, то есть среди хромосом можно выделить

группу, которая, благодаря высокой по сравнению с другими решениями оптимальностью, тиражирует свои копии в следующие поколения. На последних поколениях популяция полностью вырождается в «облачко» и в основном представляет собой копии самой лучшей на протяжении всей эволюции хромосомы, которая и определяет оптимальное решения задачи.

3.3. Решение задачи параметрического синтеза технического объекта

Параметрический синтез является одной из проектных процедур, входящих в начальные стадии проектирования, и заключается в определении наилучших значений параметров для выбранной структуры объекта с учетом требований технического задания на его проектирование.

В качестве объекта будем рассматривать пружину на сжатие. Выбор такого технического объекта связан с тем, что, с одной стороны, пружина является важным элементом конструкции многих систем, с другой – ее структура достаточно проста для понимания и не требует дополнительного описания. Математическая модель задачи состоит в следующем [2].

Требуется минимизировать объем (вес) стальной проволоки, необходимый для изготовления одной пружины с заданными свойствами (рис. 3.6).

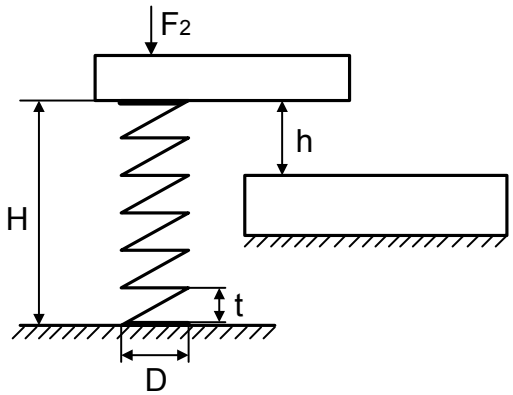


Рис. 3.6. Пример расчетной схемы пружины на сжатие

В процессе решения этой задачи использовались следующие проектировочные параметры:

F2 – заданное рабочее усилие, Н;

F3 = *F2*/0,75 – максимально допустимая нагрузка на пружину;

$G = 7{,}85 \cdot 10^4$ – модуль сдвига, МПа;

$\tau_{кр} = 960$ – допустимое напряжение кручения, МПа;

τ_3 – наибольшие касательные напряжения кручения, МПа;

H – размер (длина) пружины в свободном состоянии, м;

h – величина статической деформации пружины, м;

$\delta = 1 - F2/F3 = [0{,}05;\ 0{,}25]$ – относительный инерционный зазор пружины в наихудших по прочности условиях нагружения;

c = *F2*/*h* - жесткость пружины, Н/м

D – наружный диаметр пружины, мм;

d – диаметр проволоки пружины, мм;

m = *D*/*d* – модуль пружины;

n – число рабочих витков пружины.

Обозначения параметров выбраны в соответствии с методикой расчета пружины [4]. Значения констант зависят от материала проволоки пружины и для данной задачи соответствуют проволоке по ГОСТ 14963-78.

Множество параметров оптимизации: $x = (x_1, x_2) = (D, d)$. Целевая функция оптимизации имеет вид:

$$f(x) = \frac{\pi x_2^2}{4}(x_1 - x_2)(n+2) \to \min,$$

где x_1 - внешний диаметр пружины (*D*);

x_2 - диаметр проволоки (*d*);

n – число рабочих витков пружины.

На пространстве поиска решений накладываются следующие ограничения:

φ1 – не превышение допустимого напряжения кручения

$$\tau_3 - \tau_{кр} \le 0,$$

где $\tau_3 = \left(\frac{4m-1}{4m-4} + \frac{0{,}615}{m}\right)\frac{8F_3 D}{\pi d^3}10^{-6}$;

φ2 – наибольшее перемещение пружины без соприкосновения витков

$$h_3 + x_2\,(n+2) - H \le 0,$$

где $n = \dfrac{G d^4}{8 D^3 c} \cdot 10^6$;

φ3 – предельные отношения внешнего и внутреннего диаметров пружины

$$12 \ge m \ge 4;$$

φ4 – ограничения на число витков

$$3 - n \le 0.$$

Принимая во внимание тот факт, что значения параметра x_1 являются непрерывными величинами, а x_2 - дискретными, в результате получаем задачу смешанной оптимизации с ограничениями. Эта особенность не позволяет эффективно применить для ее решения соответствующие традиционные методы оптимизации.

Наличие четко структурированной математической модели дает возможность перейти к процессу эволюционного моделирования на базе генетического алгоритма, в качестве которого использовался стандартный вариант с турнирным отбором.

При разработке эволюционной модели задачи определялись структура генотипа, а также fitness-функция. В качестве фенотипа выступает множество *x*. Генотип образуется двоичным кодированием соответствующего параметра. Первый ген (*D*) представляет собой множество нормализованных в единичный интервал значений внешнего диаметра: $(D_{\min} \le D \le D_{\max}) \to [0,1]$. Второй ген (*d*) кодирует множество возможных дискретных значений диаметра проволоки в соответствии с требованием ГОСТ. Fitness-функция строится с применением метода штрафных функций на основе [22] и имеет вид

$$F(x,c) = f(x) + c \cdot R(x),$$

где *f(x)* – целевая функция;

R(x) – функция штрафа;

c – коэффициент штрафа.

В качестве штрафной использовалась функция вида

$$R(x) = \sum_{i=1}^{n} \left[\frac{\varphi_i(x) + |\varphi_i(x)|}{2} \right]^2,$$

где $\varphi_i(x) \le 0$ - ограничения, $i = 1,2,4$.

Ограничение *φ3* автоматически учитывается при восстановлении фенотипа из генотипа.

Анализ полученной fitness-функции показал, что функция *f(x)* и ограничения $\varphi_i(x)$ имеют различные размерности и области значений, что, безусловно, будет негативно влиять на процесс поиска решений. Основным выходом из такой ситуации является нормирование функции и ограничений в безразмерные величины на основе их физического смысла. С другой стороны, учитывая то, что наиболее значимым при оптимизации является факт соответствия ограничениям, можно предложить изменить размерности таким образом, чтобы получаемые значения штрафов ограничений оказывали значительно большее влияние на приспособленность особей, чем сама целевая функция, например это можно реализовать, если значения ограничений рассчитывать в граммах, метрах и т.д., а целевую функцию соответственно в килограммах, километрах. В зависимости от выбранной размерности определяется коэффициент штрафа - *c*.

Практическое использование подобного подхода позволило получить положительные результаты. Это может быть объяснено тем, что генетический алгоритм в динамике своей работы направляет поиск именно в те участки пространства решений, где удовлетворяются все ограничения (рис. 3.6(б)), т.е. компонента штрафа в функции пригодности равна нулю. В результате оптимизируется именно целевая функция *f(x)*.

Аналогично предыдущему примеру настройка ГА проводилась экспериментально посредством пяти запусков для каждой комбинации значений управляющих параметров *Pc*, *Pm* и постоянным значением инверсии *Pinv* = 0,001 (табл. 3.2).

Таблица 3.2

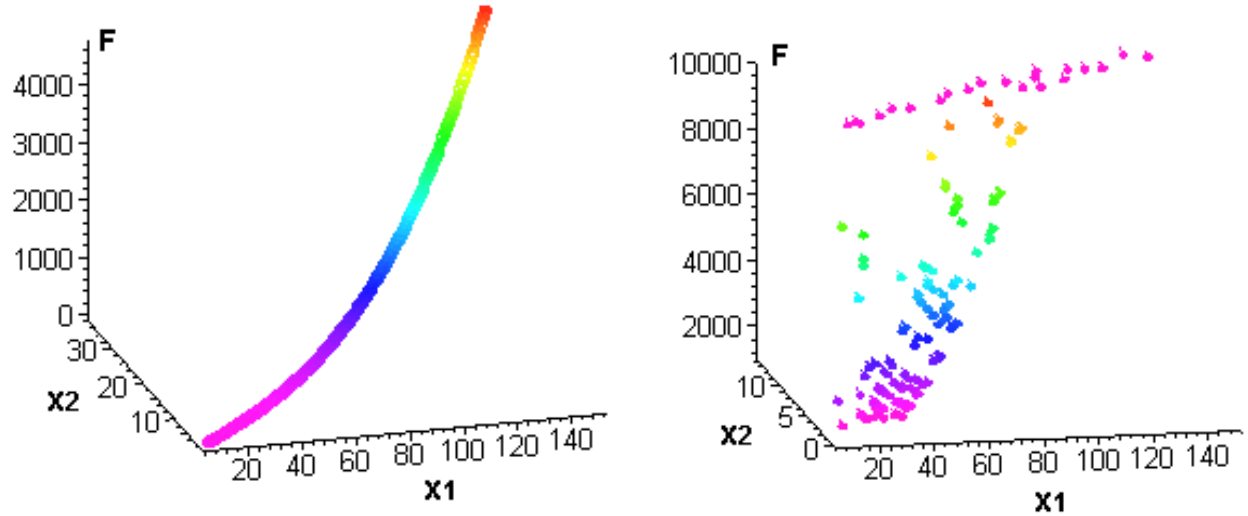
Результаты поиска значений управляющий параметров ГА

Вероятность кроссинговера	Вероятность мутации	1	2	3	4	5
0,9	0,001	6087	5936	6087	**5794**	**5794**
0,7	0,005	**5794**	**5794**	**5794**	**5794**	**5794**
0,5	0,01	**5794**	**5794**	5936	**5794**	**5794**

Как следует из результатов эксперимента, лучшее значение целевой функции при соответствии всем ограничениям равно 5794. Данное решение хотя бы один раз было получено для всех случаев изменения управляющих параметров ГА. Среднее время поиска составило 5 с. Лучший результат показал ГА со случайно-направленным характером поиска (Pc = 0,7, Pm = 0,005). Это вполне закономерно и объясняется тем, что ГА при решении данной задачи оперирует как непрерывными, так и дискретными значениями параметров. В результате пространство поиска разбито на дискретные области и именно выполнение оператора мутации позволяет перемещаться между ними.

Известно, что специфика работы ГА заключается в исследовании пространства решений в направлении улучшения fitness-функции и ГА тем эффективней, чем детальней он это пространство исследует. Поэтому представляет интерес возможность сравнить графические представления оптимизируемой функции и множества промежуточных решений, которые были получены ГА. Для рассматриваемой задачи график оптимизируемой функции может быть построен, но с рядом допущений, вызванных смешанным характером оптимизации: множество значений внешнего диаметра дискретно и ограниченно, диаметр проволоки принимается равным 1/4D, одному значению внешнего диаметра соответствует ровно одно значение диаметра проволоки, ограничения не учитываются, и число витков пружины фиксировано. На рис. 3.7(а) представлен данный график. Реальное же графическое представление рельефа оптимизируемой функции без отмеченных допущений можно оценить на основе множества решений, полученных в ходе работы ГА (рис. 3.7(б)). Здесь решениям со штрафом соответствуют максимальные значения целевой

функции. Анализ рис. 3.7 позволяет отметить схожесть обоих графиков и тем самым сделать вывод об исследовании ГА всего пространства решений, а также о глобальном характере найденного экстремального решения.



а) б)

Рис. 3.7. Рельеф оптимизируемой функции: а - на основе упрощенного аналитического представления; б - как результат процесса работы ГА

Компьютерная реализация решения задачи параметрического синтеза пружина с помощью генетического алгоритма была выполнена в виде программного модуля, главная экранная форма которого представлена на рис. 3.8.

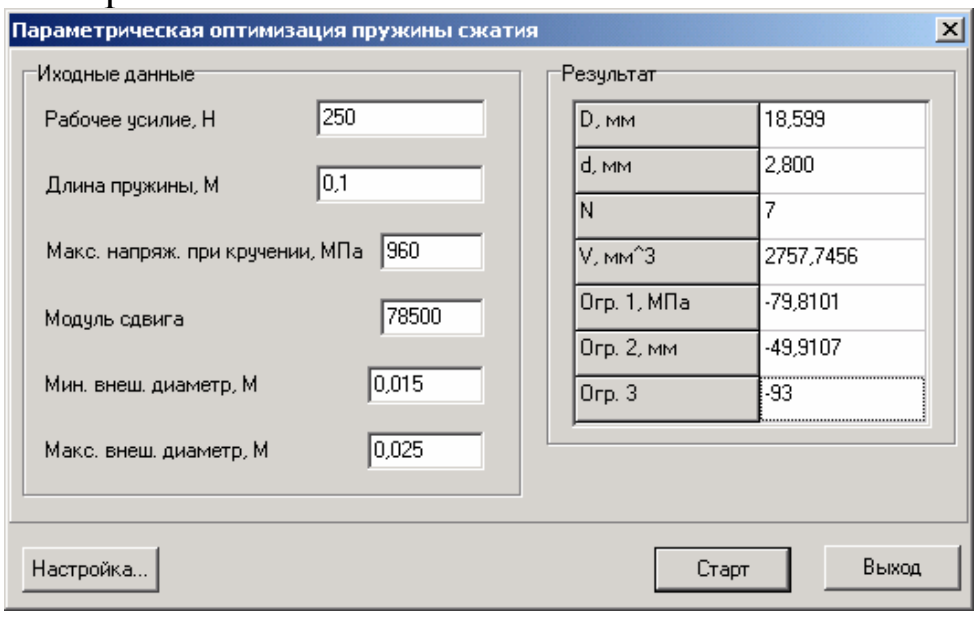


Рис. 3.8. Главное окно программы оптимизации параметров пружины

Интерфейс программного модуля предусматривает два режима работы: ввод исходных данных для оптимизации пружины и настройка параметров ГА, значения которых были выбраны в соответствии с результатами выполненного ранее эксперимента.

В качестве данных для оценки эффективности программного модуля использовались значения параметров расчета пружины из [4]. При необходимости программный модуль позволяет в динамике визуализировать процедуру оптимизации в виде графика зависимости значения оптимизируемой функции от номера поколения (рис. 3.9).

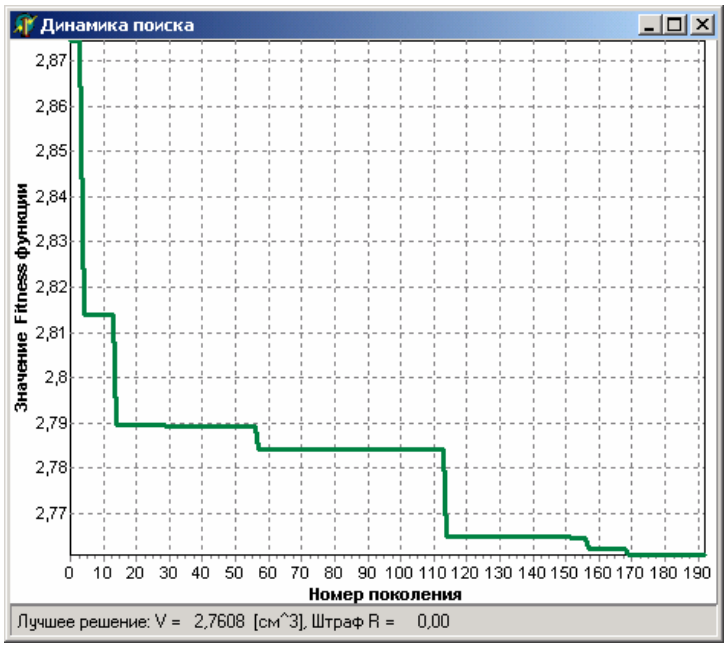


Рис. 3.9. Визуализация процесса оптимизации с помощью ГА

Анализ результатов оптимизации параметров пружины на основе ГА показал, что объем стальной проволоки, необходимый для изготовления пружины, на 17% меньше, чем при стандартном расчете пружины с аналогичными исходными данными. Таким образом, применение принципов эволюционного моделирования дало возможность преобразовать расчетную задачу в задачу оптимизации, что в итоге позволило, наряду с количественными показателями (параметры пружины), получить качественный результат (ее минимальный вес).

3.4. Решение задачи комбинаторной оптимизации

Среди задач оптимизации есть класс задач, глобально оптимальное решение которых невозможно найти в принципе. Это так называемые NP задачи (недетерминированной полиномиальной сложности), их сложность решения имеет порядок факториала $O(N!)$ или экспоненты $O(x^N, x \geq 2)$ и для них неизвестен алгоритм решения за разумное время даже путем увеличения вычислительных мощностей [27]. Примерами таких задач являются планирование расписаний, поиск маршрута, раскладка по ящикам, упаковка рюкзака, раскраска графа и др. При этом такие задачи имеют важное значение, так как являются своеобразными шаблонами для формализации ряда практических задач, таких как оптимальное распределение инвестиций, оптимальное распределение заказов, оптимизация плана работ, поиск оптимальной топологии прокладки трубопроводов, дорог, телефонных сетей и т.п.

Перечисленные задачи относятся к комбинаторным и традиционно их решением занимаются соответствующие методы дискретной оптимизации. Однако достижение оптимального результата с их помощью связано с высокой временной трудоемкостью. Поэтому поиск новых способов решения подобных задач остается актуальным, особенно учитывая, что исторически они трансформировались в своеобразные полигоны для испытания новых методик оптимизации. Эволюционное моделирование не стало здесь исключением, благодаря параллельному характеру работы ГА, была доказана их высокая эффективность решения комбинаторных задач оптимизации за линейное время [64]. В качестве примера рассмотрим применение ГА для решения задачи коммивояжера (ЗКВ).

Содержательно ЗКВ можно описать следующим образом. Задан набор городов и расстояние между любыми двумя из них. Нужно определить такой порядок, в котором следует посетить все города (по одному разу) и вернуться в исходный город, чтобы общая длина пути оказалась минимальной. Глобальный оптимум этой задачи, то есть наикратчайший путь, может быть найден только полным перебором

вариантов, что неприемлемо и возможно лишь для малого числа городов. Общее число возможных маршрутов оценивается как $Zn!$, где Zn – число городов. Так, для 15 городов это значение составляет 15! = 1307674368000, а время поиска при наличии 400 компьютеров, обрабатывающих 100 вариантов в секунду, займет около года. В общем случае увеличение численности городов на единицу увеличивает время поиска в 31 раз.

Математическая модель этой задачи может быть описана следующим образом [36].

Требуется минимизировать функцию вида

$$f = \sum_{i=0}^{Zn} \sum_{j=0}^{Zn} Z_{i,j} b_{i,j} \to \min,$$

где $Z_{i,j}$ – расстояние между городами i, j, $Z_{i,j} \geq 0$;

$b_{i,j} \in \{0,1\}$ - переменная, равная 1, если коммивояжер непосредственно переезжает из города i в город j, 0 – в противном случае.

$i, j = 0,1,2,\ldots,Zn$, $i \neq j$.

Условия однократного посещения каждого из городов имеют соответственно вид

$$\sum_{i=0}^{Zn} b_{i,j} = 1,\ j = 0,1,2,\ldots,Zn;$$

$$\sum_{j=0}^{Zn} b_{i,j} = 1, i\ = 0,1,2,\ldots,Zn.$$

С точки зрения теории графов эта задача заключается в нахождении гамильтонова цикла кратчайшей длины.

Для эволюционная моделирования при оптимизации данной математической модели необходимо определить структуру хромосомы для хранения решения (маршрутов), а также адаптировать под ЗКВ генетические операторы.

В данной реализации ГА хромосома будет описывать закодированный маршрут, в ней каждый ген будет определять посещенный город. Так, для Zn = 17 в хромосоме с разрядностью гена 4 бита будет храниться 16 городов, начиная с города с кодом 0000 и заканчивая 1111. Расположение городов задается их координатами в двумерном пространстве.

Говоря о генетических операторах, следует отметить, что применение стандартного кроссинговера, а также его модификаций (циклический, частично отображаемый, упорядоченный) в этой задаче невозможно, потому что в результате обмена фрагментами родительских хромосом потомок с большой вероятностью может содержать недопустимый маршрут, в котором некоторые города могут отсутствовать или повторяться (табл. 3.3).

Таблица 3.3

Случай появления недопустимого маршрута

Родитель1, C_1	1, 2, 4, 5, 12, 14, | 3, 9, 10, 16, 15, 7, 8, 6, 11, 13
Родитель2, C_2	13, 12, 14, 1, 5, 7, | 8, 10, 9, 2, 3, 11, 4, 16, 15, 6
Потомок, C_{12}	1, 2, 4, 5, 12, 14, 8, 10, 9, 2, 3, 11, 4, 16, 15, 6

Здесь после кроссинговера города 2, 4 повторяются, а 7, 13 отсутствуют в маршруте. Разрешением этой проблемы является разработка специального кроссинговера, лишенного отмеченных недостатков. В данном ГА это реализуется следующем образом.

Каждой хромосоме популяции ставится в соответствие характеристический вектор V, где $V[l] = 1$ ($l = 1..16$), если город с номером l уже есть в маршруте, 0 – в противном случае. Вектор V формируется для каждого потомка и заполняется при просмотре значений получившихся генов. Если обнаруживается повторяющийся город, то он заменяется первым по порядку или случайным индексом l вектора V, где $V[l] = 0$. После этого $V[l]$ устанавливается равным 1. Учитывая разнообразие возможных комбинаций таких замен, можно утверждать о сохранении разнообразия в популяции после их выполнения. В табл. 3.4 приведены результаты применения изложенного принципа для «исправления» хромосомы C_{12}, содержащей некорректные фрагменты.

Таблица 3.4

Приведение хромосомы к корректному виду

Хромосома, C_{12}	1, 2, 4, 5, 12, 14, 8, 10, 9, 2, 3, 11, 4, 16, 15, 6
Характеристич. вектор	1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1
Хромосома, C'_{12}	1, 2, 4, 5, 12, 14, 8, 10, 9, 7, 3, 11, 13, 16, 15, 6

В этой версии кроссинговера потомок во многих случаях сохраняет относительное соседство элементов родителей, что важно при решении задачи типа ЗКВ.

Оператор мутации также требует модификации под ЗКВ, так как при обычном его варианте случайное изменение бит в хромосоме может привести к появлению повторяющихся городов. В данное реализации ГА оператор мутации работает по следующему принципу. Выбирается случайным образом два гена (города) в хромосоме и они меняются местами. Для примера пусть C'_{12} = 1, 2, 4, 5, **12**, 14, 8, 10, 9, 7, 3, 11, 13, 16, **15**, 6 и выбраны гены №5 и №15, тогда после мутации C'_{12} примет следующий вид: 1, 2, 4, 5, **15**, 14, 8, 10, 9, 7, 3, 11, 13, 16, **12**, 6.

Для настройки ГА использовались две топологии расположения городов: в первой города располагаются по кругу, оптимальное решение соответствует порядку посещения городов по периметру по часовой стрелке, а длина маршрута равна длине окружности, во второй топологии города располагаются случайно и оптимальный маршрут не известен. В качестве fitness-функции используется выражение для определения общей длины пути на основе формулы Евклида.

$$f = \sum_{i=1}^{Z(n-1)} \sqrt{(Zx_{(i+1)} - Zx_{(i)})^2 + (Zy_{(i+1)} - Zy_{(i)})^2} + \sqrt{(Zx_{(n)} - Zx_{(1)})^2 + (Zy_{(n)} - Zy_{(1)})^2},$$

где Zx_i, Zy_i - координаты города с номером i.

Меньшее значение соответствует лучшему решению. Размер популяции, число поколений и схема отбора принимались аналогичными предыдущим задачам. Ниже в табл. 3.5, 3.6 приведены результаты экспериментального исследования влияния комбинаций значений управляющих параметров ГА на оптимальность поиска для двух топологий. Для топологии 1 глобально оптимальное решение выделено полужирным.

Таблица 3.5

Исследование работы ГА для топологии 1

Вероятность кроссинговера	Вероятность мутации	1	2	3	4	5
0,9	0,001	**312,36**	346,01	**312,36**	346,01	**312,36**
0,7	0,005	**312,36**	**312,36**	**312,36**	**312,36**	**312,36**
0,5	0,01	**312,36**	346,01	346,01	346,01	450,1

Таблица 3.6

Исследование работы ГА для топологии 2

Вероятность кроссинговера	Вероятность мутации	1	2	3	4	5
0,9	0,001	485	456,25	439,05	459,48	495,29
0,7	0,005	**409,78**	**409,78**	**409,78**	410,3	**409,78**
0,5	0,01	480,21	441	463,84	586,47	450,71

Далее, принимая в качестве значений Pc = 0,7, Pm = 0,005, рассмотрим эффективность работы ГА относительно других методов решения ЗКВ: комбинаторного, «жадного» и «ветвей и границ». Первый способ состоит в переборе всех возможных вариантов и выборе лучшего. Он является неэффективным с точки зрения временной сложности, но позволяет гарантированно найти глобально оптимальное решение, при этом количество перебранных решений для его достижения составляет $Zn!$. Метод «жадного» поиска относится к эвристическим и формулируется в виде следующих правил.

1. Выбирается произвольный город, который становится одновременно начальным и конечным пунктом маршрута.

2. Пока список городов не исчерпан, выполняется следующее:

- находится город, ближайший либо к начальному, либо к конечному маршруту;

- новый город добавляется в конец маршрута, если он ближайший к конечному пункту, в начало, если он ближайший к начальному пункту.

3. Маршрут найден.

Данный метод является наиболее быстрым, однако он в очень редких случаях позволяет говорить, что найденный маршрут самый лучший.

Метод «ветвей и границ», а также его модификации есть наиболее эффективные традиционные средства решения комбинаторных задач, в том числе и ЗКВ. Идея метода «ветвей и границ» состоит в последовательном разбиении пространства решений, имеющего древовидную структуру, на множества в сочетании с поиском в них

наиболее перспективных для последующего раскрытия вершин. Для этого могут использоваться различные стратегии организации ветвления дерева, а также эвристические функции.

Для оценки результативности методов поиска задавалась константа R, определяющая длину пути, достижение которой было сигналом для прекращения поиска. В табл. 3.7, 3.8 приведены результаты работы четырех методов решения ЗКВ.

Таблица 3.7

Результаты решения ЗКВ для топологии 1 (R = 312,36)

Метод решения	Длина пути	Время поиска
Комбинаторный	527,7	1 ч.
«Жадный»	312,36	0,1 с.
«Ветвей и границ»	312,36	1 с.
Генетический алгоритм	312,36	2 с.

На рис. 3.10, 3.11, 3.12 представлены маршруты на топологии 1, найденные на разных стадиях работы ГА.

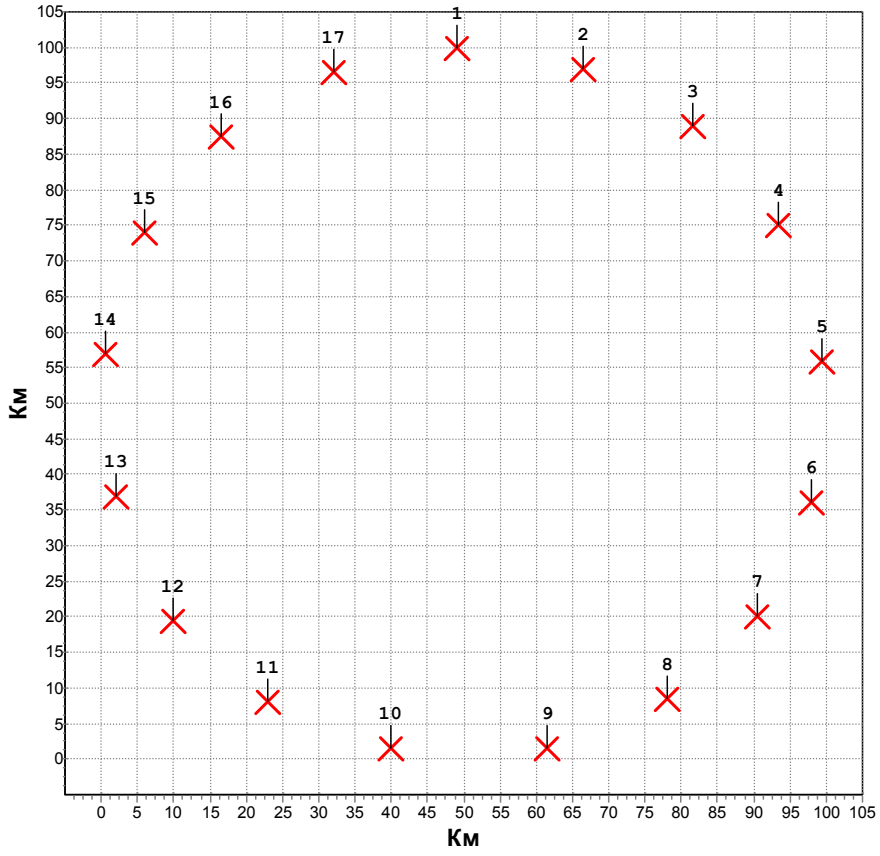


Рис. 3.10. Поиск маршрута, $t = 0$

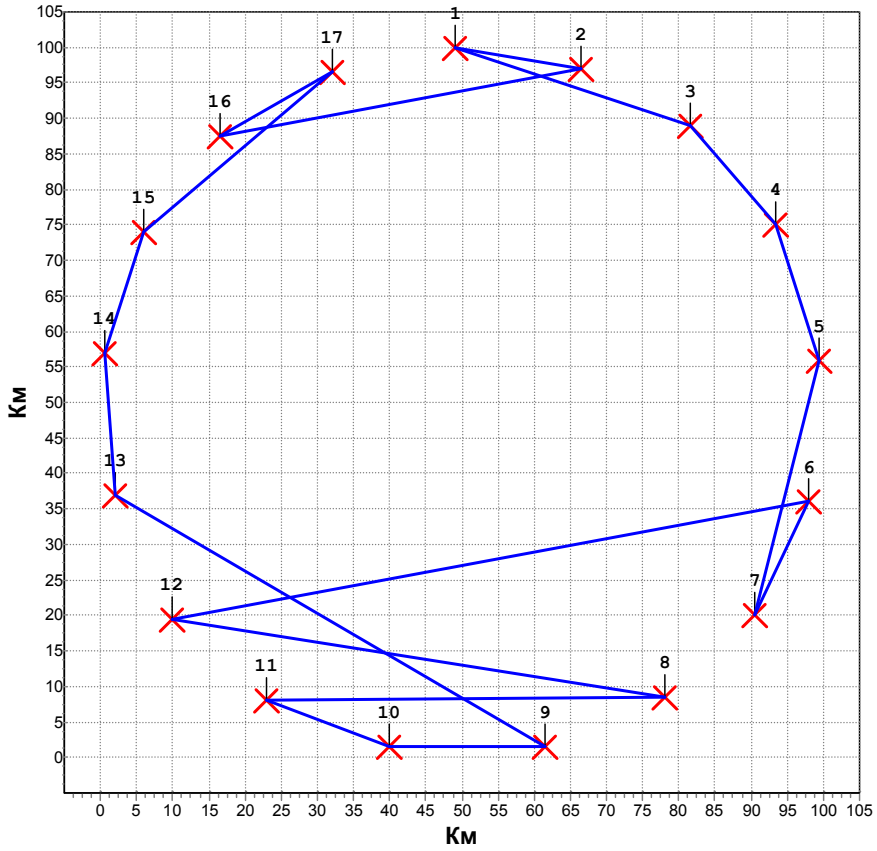


Рис. 3.11. Поиск маршрута, t = 15

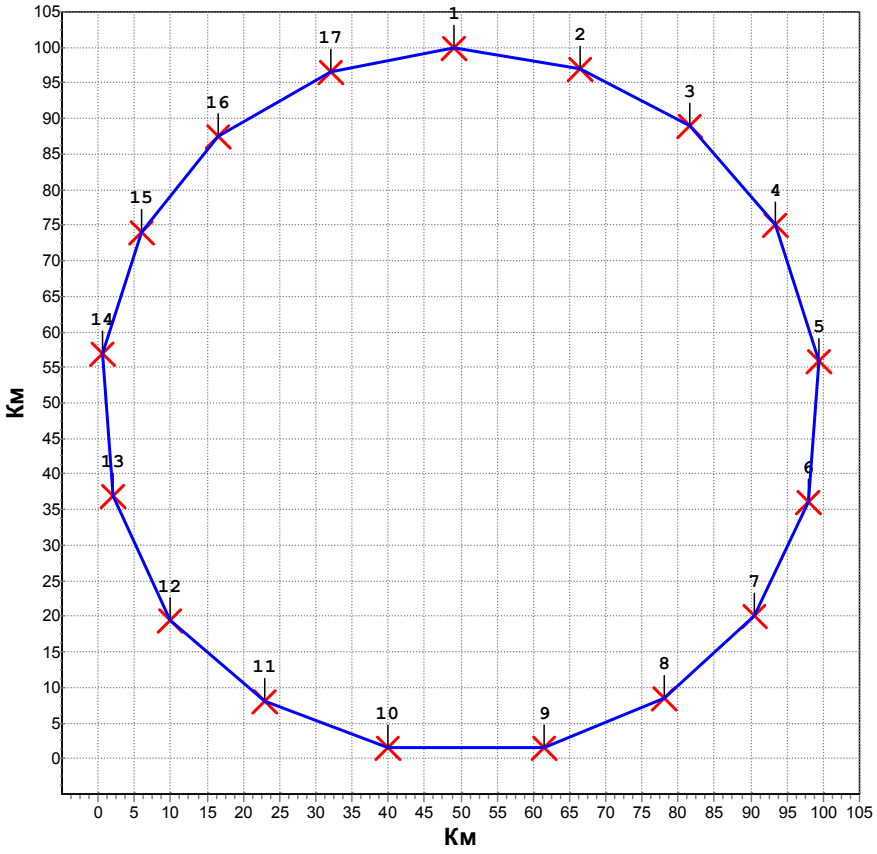


Рис. 3.12. Поиск маршрута, t = 40

Таблица 3.8

Результаты решения ЗКВ для топологии 2 (R < 430)

Метод решения	Длина пути	Время поиска
Комбинаторный	572,3	1 ч
«Жадный»	586,7	0,1 с
«Ветвей и границ»	407,6	3 мин
Генетический алгоритм	409,78	7 с

Из таблиц видно, что ГА оказался результативней методов «жадного» поиска и полного перебора. Близкий к ГА результат показал метод «ветвей и границ», однако его временная эффективность снижается при увеличении размерности поиска.

Важно сказать, что здесь использовался стандартный ГА с незначительной модификацией под ЗКВ. В то же время существуют версии ГА, специально адаптированные исключительно под ЗКВ, они позволяют находить ещё более лучшее решение, а на параллельных архитектурах – близкое к глобально оптимальному. Также существуют примеры аппаратной реализации ГА для решения задач типа ЗКВ, в частности для маршрутизации пакетов в компьютерных сетях.

3.5. Решение задачи оптимизации распределения инвестиций

Одним из ключевых факторов успешного увеличения доходности юридических и физических лиц является эффективное инвестирование средств в ценные бумаги. В сложившейся мировой практике фондового рынка эта задача связана с управлением портфелем ценных бумаг или инвестиционным портфелем (ИП) [7]. Его целью является повышение качества инвестирования в виде надежного сбережения капитала или получения максимального дохода. При этом предполагается наличие у инвестора определенной суммы денег, которую необходимо распределить по различным альтернативным вложениям (акциям, облигациям, иностранной валюте и др.), вместе обладающим такими инвестиционными характеристиками, которые недостижимы с позиций отдельно взятой ценной бумаги.

Распределение средств следует произвести таким образом, чтобы доходность портфеля была максимальна, а риск убытков минимален. В зависимости от инвестиционных свойств ценных бумаг могут быть сформированы различные инвестиционные портфели, в каждом из которых будет собственный баланс между существующим риском, приемлемым для владельца портфеля и ожидаемой доходностью в определенный период времени.

Доходы по инвестиционному портфелю являются валовой прибылью по всей совокупности бумаг, включенных в тот или иной портфель с учетом риска. Принимая во внимание постоянное динамичное и часто плохо предсказуемое изменение финансового рынка, особенно в условиях международного экономического кризиса, возникает потребность в адекватной корректировке инвестиционного портфеля с целью нахождения той грани между ликвидностью, доходностью и риском, которая позволит найти оптимальную структуру портфеля. В связи с этим автоматизация решения данной задачи является всегда актуальной.

3.5.1. Особенности формирования инвестиционного портфеля

Сформированный инвестиционный портфель представляет собой товар, который может продаваться частями – продаются доли портфеля для каждого инвестора, либо целиком – менеджер управляет портфелем ценных бумаг клиента. Относительно процедуры формирования портфеля существуют различные правила и рекомендации. Так, рекомендуется полностью пересматривать содержимое ИП, каждые 3-5 лет [28]. Добавление в портфель различных, слабокоррелирующих ценных бумаг снижает риск инвестиций. Портфель, содержащий различные виды ценных бумаг, имеющих низкий коэффициент корреляции, называется диверсифицированным. Важным является определение количества различных ценных бумаг в таком портфеле, а именно степень его диверсификации. Установлено, что максимальное снижение риска достигается, если в портфель отобраны

от 10 до 15 различных ценных бумаг [28]. Излишняя диверсификация может привести к следующим негативным результатам [28]:

- невозможность качественного управления портфелем;
- покупка недостаточно надежных, доходных, ликвидных ценных бумаг;
- рост издержек, связанных с поиском ценных бумаг (расходы на предварительный анализ и т.д.);
- высокие издержки при покупке небольших партий ценных бумаг и т.д.

Для управления ИП менеджер использует средства определения доходности активов портфеля ценных бумаг, на основе которых также оценивает риск формирования портфеля. Информация для этого может быть получена, в частности, на специализированных Интернет-сайтах [38], где регулярно проводится мониторинг изменения стоимости различных ценных бумаг, также формулируются принципы и рекомендации по их использованию. Наличие такой информации, а также соответствующих математических методов позволяет сформировать так называемый оптимальный инвестиционный портфель. Оптимизация ИП состоит в определении его состава с различными вариациями пропорций входящих в него активов, которые бы обеспечили максимальную доходность при минимальном риске. При этом часто бывает, что чем выше предполагаемая доходность активов, тем выше степень риска ее получения, и наоборот, к примеру для государственных облигаций.

3.5.2. Математическая модель задачи оптимизации инвестиционного портфеля

Применение математического моделирования при оптимизации инвестиционного портфеля позволяет в необходимые сроки корректировать содержимое ИП для улучшения его характеристик при изменениях на рынке ценных бумаг. Традиционная постановка такой задачи описывается в виде следующей математической модели [28].

Пусть некоторое юридическое или физическое лицо, заинтересованное в получении оптимального ИП, обладает финансовыми средствами в объеме U и может их использовать для приобретения n видов ценных бумаг в объемах $V_1,...,V_n$. Известны начальная стоимость α_i одной единицы ценных бумаг вида i, а также ее прогнозируемая стоимость β_i к моменту времени t. Предполагается, что $\beta_i > \alpha_i$ $(i = 1,\ldots,n)$. Необходимо определить такой состав ценных бумаг, чтобы полученная прибыль после их продажи в момент времени t была максимальна. Данная задача является задачей целочисленного программирования с булевыми переменными и записывается следующим образом:

$$f = \sum_{i=1}^{n} V_i x_i \beta_i + (U - \sum_{i=1}^{n} V_i x_i \alpha_i) \to \max,$$

$$\sum_{i=1}^{n} V_i x_i \alpha_i \le U,$$

где $x_i \in \{0,1\}$ - определяет присутствие или отсутствие ценной бумаги вида i в ИП.

Первое слагаемое функции f представляет собой доход от продажи ценных бумаг, а вторая – остаток денежных средств после формирования ИП. Учитывая, что константа U не влияет на оптимальное решение, функция f может быть упрощена и приведена к виду $f = \max \sum_{i=1}^{n} V_i x_i (\beta_i - \alpha_i)$.

Решением описанной задачи оптимизации ИП является бинарный вектор X размером n и содержащий 1 в номерах позиций, совпадающих с номером содержащейся в ИП ценной бумаги. Согласно[27] данная задача оптимизации относится к классу задач о рюкзаке с NP-сложностью.

Для решения задачи оптимизации ИП в данной постановке предлагается [28] использование разных методов поиска: «жадный», «ветвей и границ», перебора. Однако, принимая во внимание тот факт, что рекомендацией к объему ИП является не превышение 15 активов, появляется возможность точного определения f*, используя последний из отмеченных методов для упорядочения по предполагаемому доходу всевозможных комбинаций наличия/отсутствия

активов в ИП (всего $2^{15}-1$ вариант). Скорость решения при этом на современных ЭВМ составляет порядка 1с. Таким образом, для указанной постановки отсутствует потребность в поиске новых методов автоматизации решения задачи оптимизации ИП.

В связи с этим предлагается изменить математическую модель задачи оптимизации ИП, сделать ее более гибкой (отказаться от жесткого задания объема приобретаемых активов) для потенциального пользователя ИП. Для этого в качестве переменных оптимизации в математической модели используются объемы x_i каждого из активов. Кроме этого, для повышения «практичности» задачи с каждым активом связывается не прогнозируемая прибыль, а вероятностная оценка доходности, рассчитываемая на основе анализа данных фондового рынка ценных бумаг[38]. Исходная математическая модель задачи ИП при этом изменяется следующим образом:

$$f = \max \sum_{i=1}^{n} \gamma_i \frac{\alpha_i x_i}{\sum_{i=1}^{n} \alpha_i x_i},$$

$$\sum_{i=1}^{n} x_i \alpha_i \le U,$$

где γ_i - ожидаемая доходность по i-му активу;

$\frac{\alpha_i x_i}{\sum_{i=1}^{n} \alpha_i x_i}$ - удельный вес стоимости i-го актива;

$x_i \in [x_i^-, x_i^+]$ - нижняя и верхняя границы объемов ценной бумаги вида i.

Значения γ_i рассчитываются на основе возможного диапазона колебаний доходности акций за k периодов наблюдений (например, по факту проведения торгов на фондовой бирже) от минимального γ_i^- до максимального γ_i^+ с некоторыми вероятностями $P(\gamma_i)$ ($\sum_{i=1}^{n} P(\gamma_i) = 1$) и определяется как взвешенное произведение $\overline{\gamma}_i = \sum_{j=1}^{k} \gamma_{jk} P(\gamma_{jk})$.

В качестве тестового примера далее будут использоваться следующие входные данные математической модели оптимизации ИП (табл. 3.9).

Таблица 3.9

Параметры инвестиционного портфеля

№	Название актива	Начальная стоимость (руб.)	Ожидаемая доходность (%)	Минимальное число ценных бумаг в портфеле	Максимальное число ценных бумаг в портфеле
1	Газпром	104	7	100	1000
2	Роснефть	108	8	200	600
3	Аэрофлот	24	3	100	500
4	Объединенные машиностроительные заводы	150	12	100	500
5	Заволжский моторный завод	62	15	100	500
6	Новолипецкий металлургический комбинат	30	5	200	700
7	Приволжское морское пароходство	5	15	100	500
8	Брянская сбытовая компания	12	10	300	700
9	Калужская сбытовая компания	13	10	200	600
10	МобильныеТелеСистемы	114	5	200	500
11	РБК информационные системы	13	12	300	800
12	ДИКСИ	56	11	100	500
13	Сбербанк	18	5	100	600
14	М.Видео	26	6	100	200
15	Иностранная валюта (€)	39	7	500	2000

Предварительно, на основе анализа динамики за определенный период индексов и котировок ценных бумаг, выбирались активы-кандидаты в инвестиционный портфель. В данном случае проверялось выполнение условия наличия положительной доходности активов в течение одного месяца. Представленные в табл. 3.9 значения параметров ИП являются фактическими на определенный период мониторинга рынка ценных бумаг [38] и при необходимости могут быть скорректированы под новые инвестиционные требования.

3.5.3. Эволюционная модель задачи оптимизации инвестиционного портфеля

При создании эволюционной модели вначале необходимо выбрать форму кодирования значений переменных математической модели задачи, а также сконструировать fitness-функцию.

Каждый вариант инвестиционного портфеля кодируется отдельной хромосомой из 15-ти двоичных генов фиксированной разрядности (рис. 3.13). Декодированное значения гена представляет собой нормализованное в интервале [0, 1] вещественное число, поэтому при определении реального числа акций актива i выполняется масштабирование в соответствующий этому активу интервал $[x_i^-, x_i^+]$ с последующим округлением до целого.

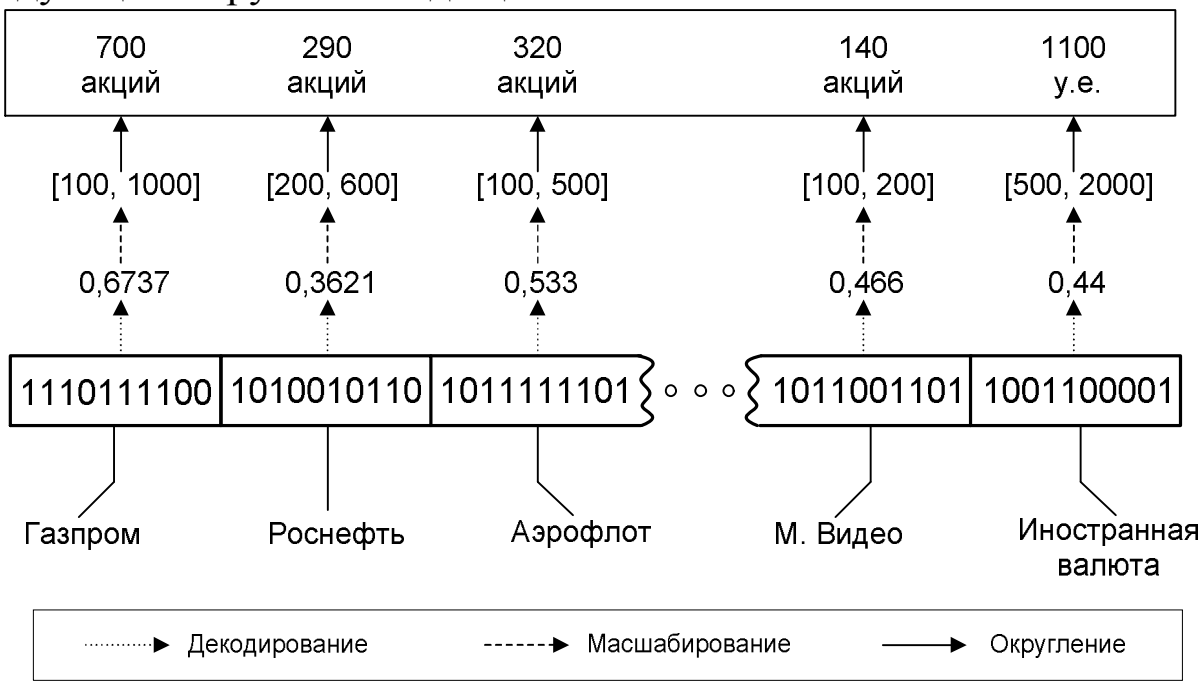


Рис. 3.13. Закодированный в виде хромосомы вариант инвестиционного портфеля

Также учитывается и возможность получения некоторого пограничного значения числа акций (например, $x_i < 10$). Для этого случая в зависимости от предпочтений ЛПР данный актив может быть как добавлен в портфель, так и не учитываться при его формировании.

Для определения оптимальности хромосомы была сконструирована fitness-функция, учитывающая специфику решаемой задачи, а именно: получение максимальной итоговой доходности по ИП, желание ЛПР «вложить» в ИП всю сумму или получить некоторый остаток денежных средств после формирования ИП, а также строгий или нет контроль объемов выделенных средств для формирования ИП. Последнее предполагает возможность увеличения начальных размеров капиталовложений для приобретения дополнительного числа акций по перспективному с точки зрения доходности активу. В результате fitness-функция получила следующий вид fitness-function(FF) = $f + R$. Здесь первое слагаемое представляет собой исходную целевую функцию математической модели задачи, второе функцию штрафа за нарушении ранее описанных ограничений. Для представления штрафной составляющей может быть использована компенсирующая сумма вида [64]

$$R = -\frac{2|U(Ci - U)| + (U(Ci) - U)}{2},$$

где $U(Ci)$ – стоимость инвестиционного портфеля, соответствующего хромосоме Ci.

При $U(Ci) = U$ значение $R = 0$, если $U(Ci) > U$, то назначается двойной штраф, так как нарушение этого ограничения является несовместимым с исходной математической моделью задачи. Если же $U(Ci) < U$, то R присваивается значение штрафа за неполное расходование средств на формирование ИП. Следствием этого может стать невозможность получения самого лучшего ИП, хотя и с сохранением для потребителя ИП некоторых исходных денежных средств.

В итоге окончательная форма fitness-функции записывается следующим образом:

$$FF = \max\left(\sum_{i=1}^{n} \gamma_i \frac{\alpha_i Ci}{\sum_{i=1}^{n} \alpha_i Ci} - \frac{2\,|U(Ci) - U| + (U(Ci) - U)}{2} \right).$$

В качестве основы эволюционной модели для данной задачи применялся стандартный ГА с турнирным отбором. С целью получения наилучших результатов при поиске решения в сочетании в высоким быстродействием выполнялась настройка ГА. Для этого проводилась серия экспериментальных запусков с варьированием значений управляющих параметров (табл. 3.10). Для каждой конфигурации задавалось максимальное число поколений Ng = 300 и объем капиталовложений 700 000 руб.

Таблица 3.10

Настройка генетического алгоритма

Np	**Pc, %**	**Pm, %**	$\overline{f^*}$**, %**	$\overline{Nf}$
200	0,9	0,01	18,6	60145
		0,02	18,45	56732
		0,03	17,5	58243
	0,8	0,01	18,2	57454
		0,02	17,3	51235
		0,03	16,8	51326
	0,7	0,01	18,3	60137
		0,02	17,2	57482
		0,03	16,7	49200
140	**0,9**	**0,01**	**18,6**	42137
		0,02	17,55	41314
		0,03	17,1	41225
	0,8	0,01	18,2	42143
		0,02	17,15	39214
		0,03	16,7	39365
	0,7	0,01	17,8	39876
		0,02	17,1	37622
		0,03	16,5	37432

Окончание табл. 3.10

Np	**Pc, %**	**Pm, %**	$\overline{f}$ ***, %**	$\overline{Nf}$
80	0,9	0,01	18,1	24219
		0,02	17,2	23311
		0,03	16,6	22822
	0,8	0,01	17,6	23133
		0,02	17,2	21314
		0,03	16,5	20122
	0,7	0,01	17,5	22353
		0,02	17,2	21254
		0,03	16,3	19563

В табл. 3.10 представлены различные тестовые конфигурации ГА. Выбор наборов значений управляющих параметров ГА мотивируется случаями наиболее различного поведения ГА (кривые лучшее, среднее, худшее решения) в тестовых запусках. Характеристики $\overline{f}$ * и $\overline{Nf}$ соответствуют усредненным по 10-ти циклам стартов ГА на каждую конфигурацию значениям оптимального решения и числа вычислений fitness-функции, достаточного для локализации оптимального решения соответственно. Из результатов испытаний видно, что для двух конфигураций ГА были получены лучшие решения, но с разным числом вычислений fitness-функции. В целом анализ результатов проведенных экспериментов показал негативное влияние оператора мутации на эффективность поиска. Это может быть связано с вынужденной дискретизацией координат решения при их декодировании и, следовательно, возникновением сильной зависимости качества решения от случайного изменения генов.

В качестве окончательной конфигурации была выбрана следующая (Ng =300 , Np = 140, Pc = 0,9, Pm = 0,01). На рис. 3.13 показаны результаты генетического поиска для решения тестовой задачи оптимизации ИП.

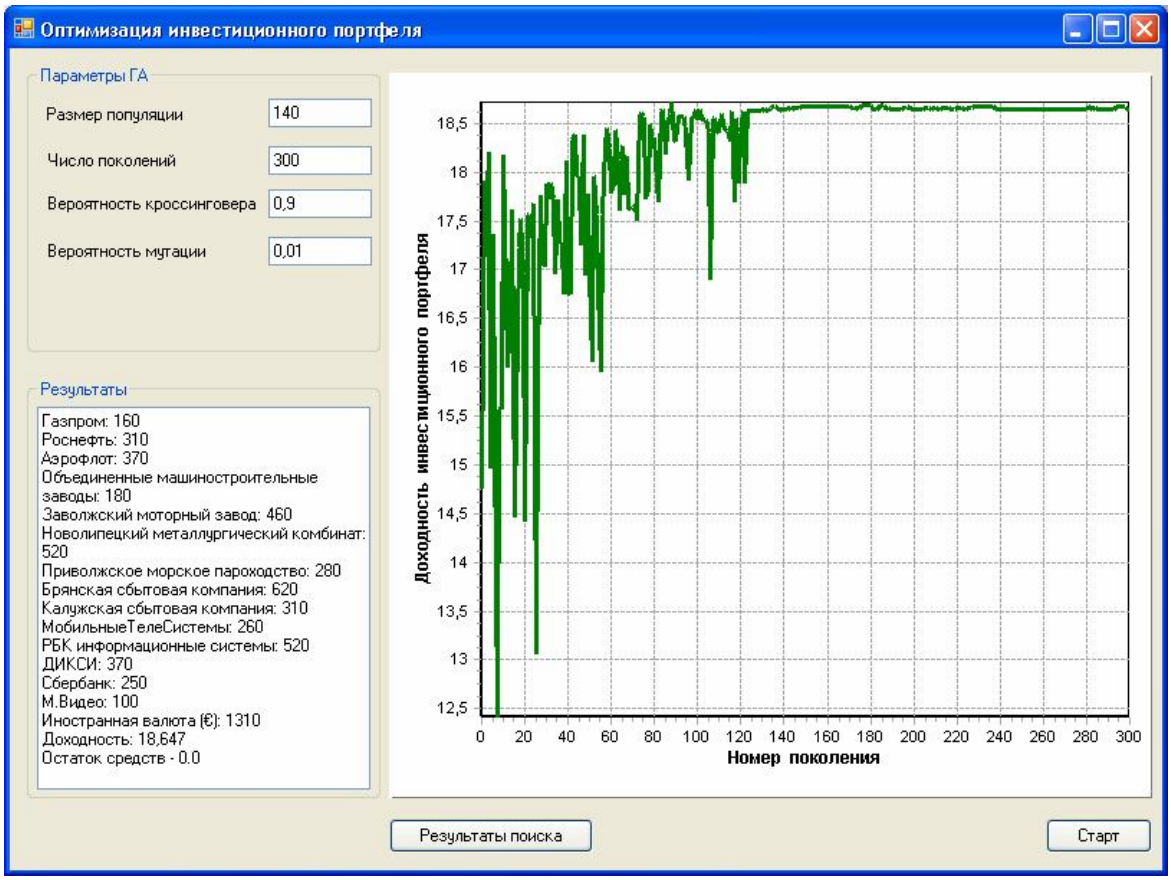


Рис. 3.14. Результаты работы генетического алгоритма оптимизации ИП

Применение генетического алгоритма для решения задачи оптимизации инвестиционного портфеля позволило автоматизировать процесс планирования инвестиционных средств, не требуя от клиента ИП изначального задания объемов приобретаемых активов. Однако созданная для этого математическая модель «усиливает» комбинаторные свойства этой задачи и предполагает возможность получения более одного варианта оптимального ИП. Для их полноценного поиска требуется применение соответствующих модификаций ГА [21]. Кроме этого, дальнейшее совершенствование информационной системы синтеза и выбора оптимального ИП может быть связано с привлечением соответствующих методов экспертного оценивания результатов работы генетического алгоритма.

3.6. Эволюционное моделирование адаптивного поведения

Общим свойством рассмотренных задач является статичность во времени целевой функции. Но возможны ситуации, обычно свойственные системам управления, экономическим системам, когда среда, в которой функционирует объект, претерпевает случайные, реже, закономерные изменения по некоторым параметрам, например это может быть поиск оптимального маршрута до цели, динамично изменяющей свою траекторию движения или расположение. Соответственно возникает потребность, чтобы поведение системы в таких динамично изменяющихся условиях как минимум оставалось адекватным, как максимум – оптимальным по определенным критериям. Благодаря независимости от математической модели, ГА позволяют решать такого типа задачи и тем самым наделять адаптивными способностями различные самонастраивающиеся системы управления [11]. Для демонстрации адаптивного поведения с помощью ГА рассмотрим достаточно простую, но наглядную задачу в следующей постановке.

Пусть есть некоторый объект с набором свойств, значения которых он может произвольно менять в зависимости от внешних условий. Визуально внешние условия, а также степень адаптации под них объекта представляются цветом: чем больше цвета похожи, тем лучше настроился под внешнюю среду объект. Обычно цвет характеризуется тремя составляющими (красная, зеленая, синяя), поэтому будем считать, что каждая из них и определяет соответствующее свойство объекта. Таким образом, задача ГА максимально быстро «окрасить» объект в цвет, наиболее близкий к заданному. При этом цвет внешних внешний условий меняется динамично, в том числе непосредственно в процессе работы ГА.

Пусть внешние условиям соответствует белый цвет, а текущему состоянию объекта – черный, тогда процесс адаптации объекта под новые условия будет иметь следующий вид (рис. 3.14).

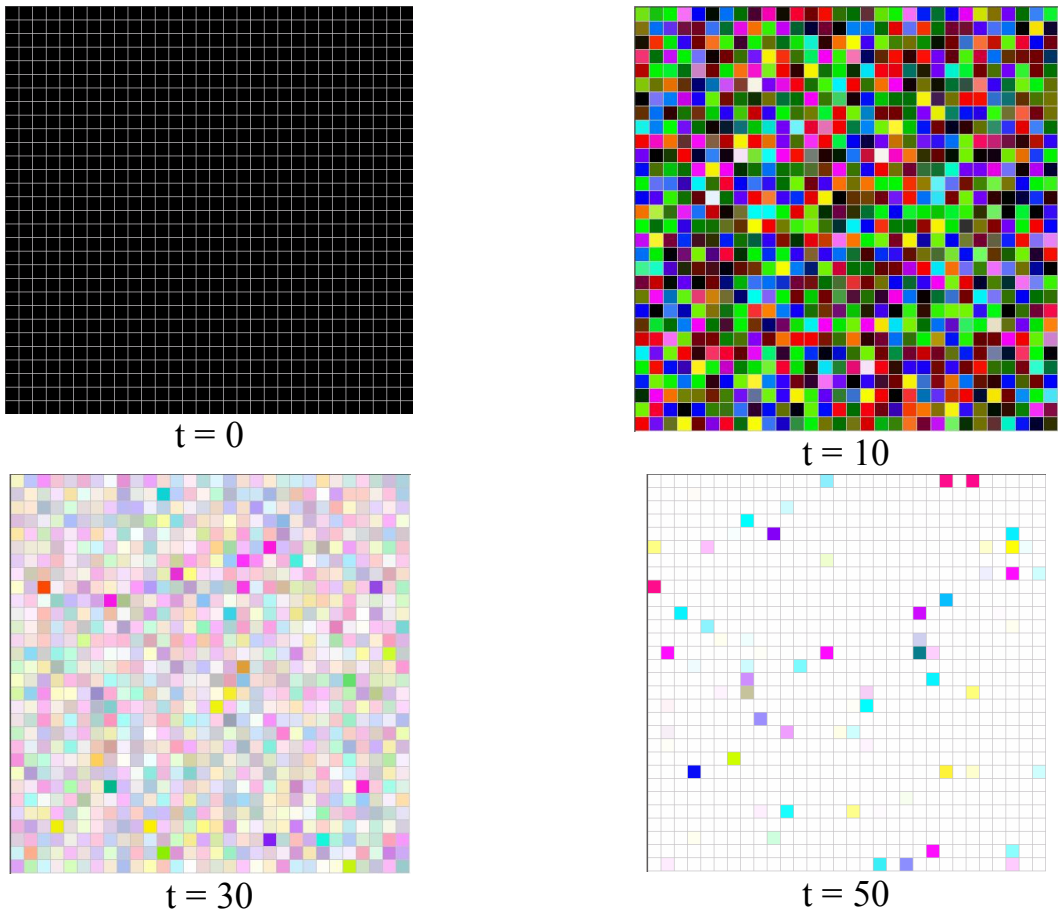


Рис. 3.14. Эволюционное моделирование адаптации объекта при изменении внешних условий

Теперь, не дожидаясь окончания работы ГА, инвертируем цвет внешних условий (рис. 3.15).

Как следует из рис. 3.14, к поколению t = 50 свойства объекта практически достигли оптимального состояния и может сложиться впечатление, что популяция окажется не подготовленной к резкому изменению внешних условий, однако это не так. Из рис. 3.15 видно первое, что происходит с популяцией после изменения, по сути, целевой функции, - это резкое увеличение ее разнообразия за счет распределения потомков по поисковому пространству. Причина этого в том, что те лучшие хромосомы, которые раньше к началу стагнации ГА составляли подавляющее большинство в популяции, при изменении условий оптимальности превращаются не более, чем в строительный материал для новых хромосом. Они не похожи друг на друга, и, следовательно, ГА получает все необходимое для полноценного исследования нового пространства поиска.

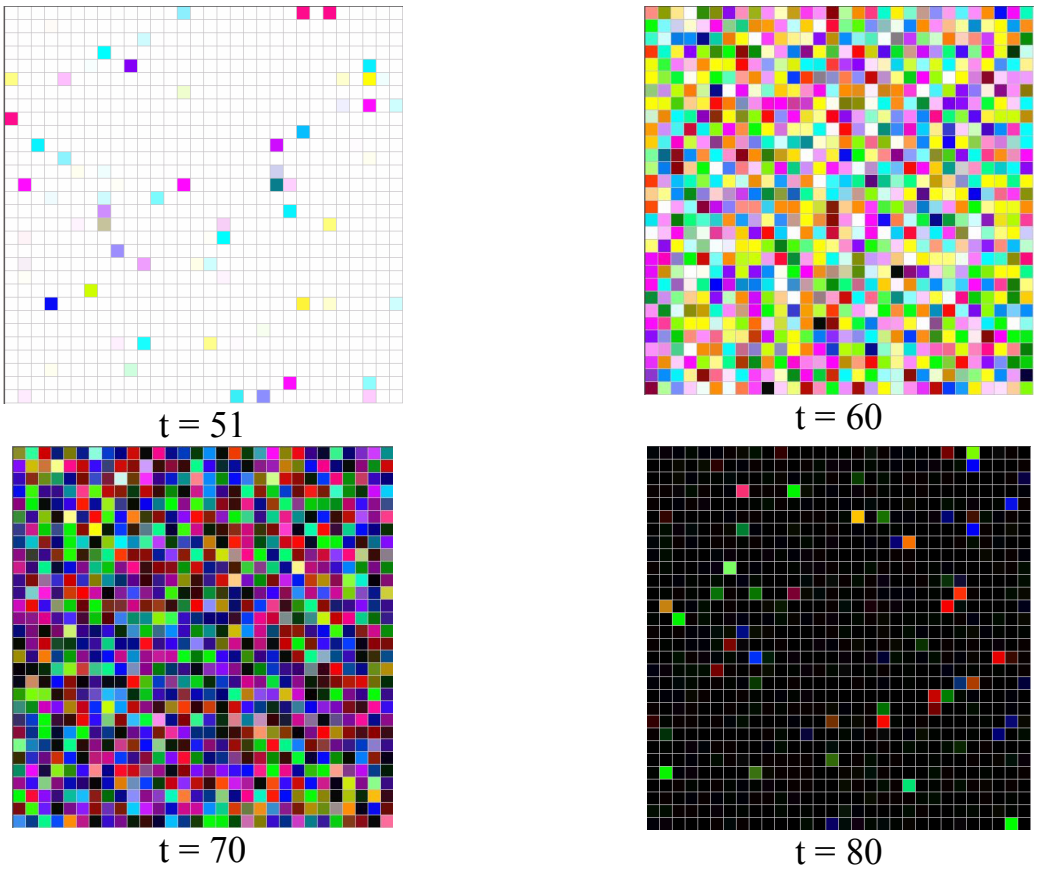


Рис. 3.15. Реакция ГА на динамическое изменение условий внешней среды

Другие эксперименты с динамическим изменением более сложных целевых функций в процессе работы ГА подтвердили его эффективность при моделировании адаптивного поведения, однако узким местом при реализации этого в реальных системах может стать быстродействие генетического поиска. Решение здесь видится в преобразовании программной составляющей ГА в аппаратную.

3.7. Программное обеспечение эволюционного моделирования

В настоящее время существует целый ряд программных средств, предназначенных для эволюционного моделирования. Условно их можно разделить на три группы: универсальные среды ЭМ, предметно-ориентированные прикладные пакеты и библиотеки программирования. Большинство из них представляет собой интерфейс

пользователя для работы с ГА, причем особенности реализации последнего обычно от пользователя скрыты. Ниже кратко рассматриваются основные программные разработки в области ЭМ [53].

GeneHunter (Ward Systems Group) является расширением системы электронных таблиц Microsoft Excel и позволяет решать оптимизационные задачи с помощью ГА в различных прикладных областях. Также с этой программой поставляется библиотека GALIB.dll, содержащая множество функций для работы с ГА и позволяющая создавать программистом собственные, независимые от MS Excel программы для ЭМ. GeneHunter имеет русифицированную версию , предназначенную для коммерческого распространения на территории РФ.

Splicer (German National Research Center NASA Johnson & Metre Corp) – пакет прикладных программ, созданный по заказу центра космических исследований NASA. Является универсальной средой ЭМ и содержит как интерфейс пользователя для управления созданием и настройкой ГА, так и библиотеку программиста для разработки собственных приложений.

EnGENer (Logica) является универсальной системой ЭМ, позволяющей адаптировать возможности ГА к различным задачам, прежде всего за счет специального языка описания эволюционной модели.

GAME (University College London) – объектно-ориентированная среда, содержащая макроязык для формирования ЭМ с использованием нескольких ГА.

GEATbx (Genetic and Evolutionary Algorithm Toolbox for use with MATLAB) – модуль оптимизации на основе ГА, используемый в среде математического моделирования MATLAB. В сочетании с функциональностью этого математического пакета GEATbx позволяет оптимизировать математические модели любой сложности, а также визуально контролировать процесс оптимизации.

SUGAL (Andrew Hunter) – инструментальная среда, предназначенная для разработки и исследования генетических алгоритмов. Характеризуется наличием различных модификаций генетических операторов, поддержкой разнообразных типов данных при кодировании хромосом, а также встроенными средствами мониторинга и генерации статистической информации о работе ГА.

ALGON (INRIA) – инструментальная среда разработки ГА на встроенном макроязыке программирования. Поддерживает различные классические генетические операторы, формы кодирования информации, имеет графический интерфейс пользователя для настройки и визуализации работы ГА.

Evolver (Axcelis) представляет собой надстройку системы электронных таблиц Microsoft Excel и предназначен главным образом для решения оптимизационных задач в финансовой сфере.

Trading Solutions (NeuroDimension) – бизнес-система, основанная на технологиях нейронных сетей и генетических алгоритмов, предназначенная для решения задач прогнозирования и моделирования в реальном времени работы биржевых рынков различного типа.

Neuro Solutions (NeuroDimension) – среда разработки нейронных сетей с применением ГА, позволяющая решать задачи, свойственные нейронным сетям (прогнозирование, кластеризация информации) научно-исследовательского и практического характера.

EVO (TransferTech) – программный комплекс, основанный на ГА и предназначенный для решения задач оптимизации в системах автоматического управления и регулирования.

GAlib* (Matthew Wall) – наиболее известная объектно-ориентированная библиотека на языке C++ для работы с ГА. Содержит множество встроенных функций, а также средства для ее расширения под различные прикладные задачи.

Genetic Server/Genetic Library (NeuroDimension) – библиотека API функций для интеграции возможностей ГА в разрабатываемое программное обеспечение. Genetic Server реализован как ActiveX компонент, позволяющий встраивать его в различные среды разработки. Genetic Library представляет собой библиотеку ГА для создания приложений в среде Visual C++.

Mendel (К.В. Махотило, Харьковский государственный политехнический университет). Система позволяет работать с произвольными пользовательскими задачами оптимизации. Целевая функция задачи оптимизации описывается во внешней dll-библиотеке. В системе доступна единственная диплоидная версия генетического алгоритма, для которой имеется возможность настройки как основных, так и специфических для этой версии ГА параметров.

В целом анализ статей, прямо или косвенно связанных с эволюционным моделированием, показывает, что обычно при компьютерной реализации эволюционных алгоритмов разрабатывается собственное программное обеспечение (прил. 1), а не используются универсальные среды. Возможно, это связано, во-первых, с тем, что одни из отмеченных выше сред и библиотек ЭМ являются достаточно дорогостоящими и на отечественном рынке программного обеспечения практически не представлены, другие – имеют недостаточное сопровождение и развитие со стороны разработчика. Во-вторых, учитывая особенности ЭА, максимальный эффект от их применения может быть достигнут лишь от непосредственного участия в их разработке и настройке. Кроме этого, наличие собственных реализаций эволюционных алгоритмов позволяет интегрировать их в уже существующие разработки, наделив их тем самым новыми полезными свойствами.

ЗАКЛЮЧЕНИЕ

Постоянное совершенствование вычислительной техники и программного обеспечения предъявляет новые требования к их использованию в виде интеллектуальной поддержки и оптимизации решений в слабоформализованных ситуациях при ограниченных ресурсах. Вместе с тем реализация этого требует поиска или разработки новых методов, объединяющих в себе возможности гибкой настройки на решаемую задачу и доступность компьютеризации. Согласно Норберту Винеру, наиболее перспективные научные исследования находятся в так называемых пограничных областях, которые нельзя точно отнести к той или иной дисциплине. Именно такое направление в искусственном интеллекте образует эволюционное моделирование. Его методы позволяют моделировать эволюционные процессы при решении задач оптимизации, самоорганизации и адаптивного поведения в производственных, технических и социально-экономических системах.

Считается, что одной из главных составляющих интеллектуальной системы является способность автоматической адаптации к изменяющимся условиям функционирования. Эволюционное моделирование позволяет создавать подобные системы благодаря наличию компьютерных алгоритмов, реализующих такие универсальные и фундаментальные принципы естественных систем, как «выживание сильнейшего», «наследование приобретенных полезных признаков», «самоорганизация под воздействием внешней среды» и их комбинации. В итоге искусственная система наделяется такими важными свойствами, как самоисправление, устойчивость к внешним условиям ее применения, гибкость по отношению к решаемым задачам и др.

Благодаря наличию удобных и доступных средств алгоритмизации, методы эволюционного моделирования могут быть отдельно или в сочетании с другими технологиями искусственного интеллекта интегрированы в существующие программные системы, тем самым качественно повышаются результаты их работы и расширяются области применения. Следствием этого является появление принципиально новых интеллектуальных информационных систем.

Теория и практика эволюционного моделирования постоянно развиваются, появляются новые области его применения. В общем случае это любые искусственные системы (производственные, технические, экономические, социальные), которым свойственны эволюционные процессы развития. Большинство таких принято относить к сложным. Они характеризуются большим количеством взаимосвязанных объектов, взаимодействующих разными способами. Решение задач оптимизации и управления в таких системах с помощью эволюционных алгоритмов позволяет использовать любые математическую модель и формы описания свойств участвующих объектов в качестве основы эволюционной модели. Необходимым и достаточным условием начала эволюционного моделирования являются правильный выбор исходного множества решений (формирование начальной популяции) и точная постановка цели.

Перспективы развития ЭМ прежде всего связаны с совершенствованием существующих и разработкой новых эволюционных алгоритмов. Прежде всего подобные исследования подразумевают создание теоретических и практических основ для модификаций генетического алгоритма. В частности, они включают:

- разработку и исследование новых модификаций генетических алгоритмов и операторов, а также создание математического обеспечения для анализа и прогнозирования эффективности их практического применения;

- интеграцию генетических алгоритмов как с существующими численными методами оптимизации, так и с другими направлениями «мягких» вычислений (нейронные сети, нечеткое моделирование) для создания гибридных интеллектуальных систем;

использование генетических алгоритмов для решения практических задач эволюционного моделирования в социальных, экономических и технических системах;

- и др.

Теорию эволюционного моделирования нельзя назвать завершенной. Все большее число исследователей обращаются к решению задач с помощью методов ЭМ. Подтверждением этого служит неослабляющий интерес к конференциям, посвященным этому разделу искусственного интеллекта и проводимым как в нашей стране, так и зарубежом.

СПИСОК ЛИТЕРАТУРЫ

1. Аверченков, В.И. Основы математического моделирования технических систем / В.И. Аверченков, В.П. Федоров, М.Л. Хейфец. - Брянск: БГТУ, 2004. - 269 с.

2. Аверченков, В.И. Применение методов эволюционного моделирования при параметрическом синтезе технических систем / В.И. Аверченков, В.С. Казаков, П.В. Казаков // Техника машиностроения. - 2006. - №4. - С. 37-44.

3. Андрейчиков, А.В. Интеллектуальные информационные системы: учеб. для вузов / А.В. Андрейчиков, О.Н. Андрейчикова. – М.: Финансы и статистика, 2004. – 424 с.

4. Анурьев, В.И. Справочник конструктора-машиностроителя / В.И. Анурьев. – 7-е изд., перераб. и доп. – М.: Машиностроение, 1992. – 720 с. – (Справочник конструктора-машиностроителя: в 3 т./ В.И. Анурьев; т.3).

5. Батищев, Д.И. Поисковые методы оптимального проектирования / Д.И. Батищев. - М.: Сов. радио, 1975. – 216 с.

6. Бобровский, С. Эволюция и искусственная жизнь / С. Бобровский // PC Week / RE. - 2005. - № 3 - 5.

7. Боди, З. Принципы инвестиций / З. Боди, А. Кейн, А. Маркус. – 4-е изд. – М.: Издательский дом «Вильямс», 2002. – 984 с.

8. Букатова, И.Л. Эволюционное моделирование и его приложения / И. Л. Букатова. - М.: Наука, 1979. - 232 с.

9. Букатова, И.Л. Эвоинформатика. Теория и практика эволюционного моделирования / И. Л. Букатова, Ю.И. Михасев, А.М. Шаров. - М.: Наука, 1991. - 206 с.

10. Васильев, В.И. Интеллектуальные системы управления с использованием генетических алгоритмов / В.И. Васильев, Б.Г. Ильясов // Информационные технологии (Приложение). - 2000. - №12. - 24 с.

11. Вороновский, Г.К. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности / Г.К. Вороновский, К.В. Махотило, С.Н. Петрашев, С.А. Сергеев. – Х.: ОСНОВА, 1997. – 112 с.

12. Генетические алгоритмы и не только [Электронный ресурс]. – Режим доступа http://qai.narod.ru.

13. Гилл, Ф. Практическая оптимизация: [пер. с англ.] / Ф. Гилл, У. Мюррей., М. Райт. – М.: Мир, 1985. – 509 с.

14. Гладков, Л.А. Генетические алгоритмы / Л.А. Гладков, В.В. Курейчик, В.М. Курейчик. - М.: ФИЗМАТЛИТ, 2006. – 320 с

15. Гладков, Л.А. Методы генетического поиска / Л.А. Гладков, Л.А. Зинченко, В.В. Курейчик, В.М. Курейчик, Б.К. Лебедев, Е.В. Нужнов, С.Н. Сорокин. - Таганрог: Изд-во ТРТУ, 2002. – 122 с.

16. Емельянов, В.В. Теория и практика эволюционного моделирования / В.В. Емельянов, В.В., Курейчик, В.М. Курейчик – М.: ФИЗМАТЛИТ, 2003. – 431 с.

17. Зинченко, Л.А. Повышение эффективности эволюционного проектирования на основе анализа спектральных свойств поверхности функции пригодности / Л.А. Зинченко, А.В. Каляда // Новости искусственного интеллекта. - 2003. - №5.

18. Казаков, П.В. Объектно-ориентированное программирование: учеб. пособие / П.В. Казаков. - Брянск: БГТУ, 2005. - 115с.

19. Казаков, П.В. Основы искусственного интеллекта: учеб. пособие для вузов/ П.В. Казаков, В.А. Шкаберин. - Брянск: БГТУ, 2007. – 196 с.

20. Казаков, П.В. Способы мониторинга работы генетического алгоритма / П.В. Казаков // Вестник компьютерных и информационных технологий. - 2008. - №9. - С. 7-11.

21. Казаков, П.В. Оптимизация многоэкстремальных функций на основе кластерной модификации генетического алгоритма / П.В. Казаков // Одиннадцатая национальная конференция по искусственному интеллекту КИИ-08: труды конференции: В 3 т. – М.: ЛЕНАНД, 2008. – Т.3. - С. 26-32.

22. Корячко, В.П. Теоретические основы САПР / В.П. Корячко, В.М. Курейчик, И.П. Норенков. - М.: Энергоатомиздат, 1987. - 400 с.

23. Курейчик, В.М. Генетические алгоритмы и их применение / В.М. Курейчик. – 2-е изд. перераб. и доп. – Таганрог: Изд-во ТРТУ, 2002. – 242 с.

24. Курейчик, В.М. Исследование динамических операторов в эволюционном моделировании / В.М. Курейчик, Л.А. Зинченко, И.В. Хабарова // Перспективные информационные технологии и интеллектуальные системы. - 2001. - №3(7). - С. 65-70.

25. Курейчик, В.М. Эволюционные вычисления: Генетическое и эволюционное программирование / В.М. Курейчик, С.И. Родзин // Новости искусственного интеллекта. - 2003. - №5.

26. Макаров, А. От Ламарка к Дарвину … и обратно к Ламарку? / А. Макаров // Компьютерра. - 2005. - № 10.

27. Макконнелл, Дж. Основы современных алгоритмов: [пер. с англ.] / Дж. Макконелл. – М.: Техносфера, 2004. – 368 с.

28. Мищенко, А.В. Попов, А.А. Некоторые подходы к оптимизации инвестиционного портфеля / А.В. Мищенко, А.А. Попов // Менеджмент в России и за рубежом. – 2002. - №2.

29. Норенков, И.П. Генетические методы структурного синтеза проектных решений / И.П. Норенков // Информационные технологии. - 1998. - №1. - С. 9-13.

30. Норенков, И.П. Основы автоматизированного проектирования / И.П. Норенков. - М.: Изд-во МГТУ им. Н.Э. Баумана, 2000. - 360 с.

31. Половинкин, А.И. Основы инженерного творчества / А.И. Половинкин. - М.: Машиностроение, 1988. - 368 с.

32. Расстригин, Л.А. Случайный поиск в задачах оптимизации многопараметрических систем / Л.А. Расстригин. - Рига: Зинатне, 1965. - 212С.

33. Редько, В.Г. Эволюционная кибернетика / В.Г. Редько. - М.: Наука, 2003. - 156 с.

34. Родзин, С.И. Формы реализации и границы применения эволюционных алгоритмов / С.И. Родзин // Перспективные информационные технологии и интеллектуальные системы. - 2002. - №1(9). - С. 36-41.

35. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы: [пер. с польск.] / Д. Рутковская, М. Пилиньский, Л. Рутковский. – М.: Горячая линия – Телеком, 2008. – 452 с.

36. Сигал, И.Х. Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы: учеб. пособие / И.Х. Сигал, А.П. Иванова. - М.: ФИЗМАТЛИТ, 2003. - 240 с.

37. Скурихин, А.Н. Генетические алгоритмы / А.Н Скурихин // Новости искусственного интеллекта. - 1995. - №5. С. 6-46.

38. Социальная сеть инвесторов [Электронный ресурс]. – Режим доступа http://tikr.ru.

39. Фогель, Л. Искусственный интеллект и эволюционное моделирование: [пер. с англ.] / Л. Фогель, А. Уолш. – М.: Мир, 1969. - 228с.

40. Химмельблау, Д. Прикладное нелинейное программирование: [пер. с англ.] / Д. Химмельблау. - М.: Мир, 1975. - 534 с.

41. Хог, Э. Прикладное оптимальное проектирование: [пер. с англ.] / Э. Хог, Я. Арора. – М.: Мир, 1983. – 483 с.

42. Холланд, Дж. Генетические алгоритмы / Дж. Холланд // В мире науки. - 1992. - № 9 - 10.

43. Цой, Ю.Р. Генетический алгоритм. Представление знаний в информационных системах: учеб. пособие / В.Г. Спицын, Ю.Р. Цой. – Томск: ТПУ, 2006. – 146 с.

44. Шуп, Т. Решение инженерных задач на ЭВМ. Практическое руководство: [пер. с англ.] / Т. Шуп. – М.:Мир, 1982. – 238 с.

45. Bäck, T. Evolutionary Algorithms in Theory and Practice / T. Bäck. - Oxford University Press, New York, 1996.

46. Bäck, T., Fogel D., Michalewicz Z. Evolutionary computations 1. Basic algorithms and operators / T. Bäck, D. Fogel, Z. Michalewicz. - IOP Publishing, 2000. - 378 p.

47. Barlow, M. Mining evolution through visualization / M. Barlow, J. Galloway // In Proceeding of ALife VIII Conference //MIT Press. - 2002. – pp. 103-111.

48. Bentley, P. Evolutionary Design by Computers / P. Bentley. - Morgan Kaufman Pub., 1999. – 464 p.

49. Coley, D. An Introduction to Genetic Algorithms for Scientists and Engineers / D. Coley. - World Scientific Publishing. 1999. - 227 p.

50. Dawkins, R. Universal Darwinism. Evolution from Molecules to Men / R. Dawkins. - Cambridge University Press, 1983.

51. Deb, K. Understanding interactions among genetic algorithm parameters / K. Deb, S. Agrawal // Foundations of Genetic Algorithms-5. - 1998. - pp. 265-286.

52. De Jong, K. Analysis of behavior of class of genetic adaptive systems: Ph D. Thesis / K. De Jong. - University of Michigan, 1975.

53. EvoWeb [Электронный ресурс]. – Режим доступа http://evonet.lri.fr.

54. Fogel, D. System Identification through Simulated Evolution: A Machine Learning Approach to Modeling / D. Fogel. - Ginn Press, 1991.

55. Fogel, D. Evolutionary Computation : Toward a New Philosophy of Machine Intelligence / D. Fogel. - IEEE Press, 1995.

56. Goldberd, D. Genetic Algorithms in Search, Optimization and Machine Learning / D. Goldberd. - N.Y.: Addison-Wesley Publishing Company, Inc. 1989 - 412 p.

57. Haque, M. Optimal frame design with discrete members using the complex method / M. Haque // Computers & Structures. - 1996, vol. 59. - pp. 847-858.

58. Hart, E. Enhancing the performance of a ga through visualization / E. Hart, P. Ross // In Proceedings of Genetic and Evolutionary Computation Conference // Morgan Kaufman. - 2000. – pp. 347-354.

59. Holand, J. Adaptation in Natural and Artificial Systems. An Introductory Analysis with Application to Biology, Control and Artificial Intelligence / J. Holand. - The University of Michigan Press, 1975. - 210 p.

60. Kinnear, Jr. Advances In Genetic Programming / Jr. Kinnear. - MIT Press, 1994. - 518 p.

61. Koza, J. Genetic Programming: On the Programming of Computer By Means of Natural Selection / J. Koza. - The MIT Press, Cambridge, Massachusetts, London, England, 1992. - 819 p.

62. Manetsch, T. Toward efficient global optimization in large dynamic systems - The adaptive complex method / T. Manetsch // IEEE Transactions on Systems, Man & Cybernetics. - 1990, vol. 20. - pp. 257-261.

63. Manetsch, T. Use of concurrent processing with the adaptive complex method for global optimization of large dynamic systems / T. Manetsch, A. Cabrera // IEEE Transactions on Systems, Man & Cybernetics. - 1991, vol. 21. - pp. 442-445.

64. Michalewicz, Z. Genetic Algorithms + Data Structures = Evolution Programs / Z. Michalewicz. - Springer, 1996. - 387 p.

65. Rechenberg, I. Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution / I. Rechenberg. - Stuttgart: Fromman-Holzboog, 1973.

66. Sahoo, N. Modified complex method for constrained design and optimization of optical multilayer thin-film devices / N. Sahoo, K. Apparao // Applied Physics A-Solids & Surfaces. - 1994, vol. 59. - pp. 317-326.

67. Schwefel, H. Evolution and optimum searching / H. Schwefel. - Wiley Interscience, New York, 1995.

68. Zilberstein S. Operational rationality through compilation of anytime algorithms // Ph.D dissertation, Computer Science Division, University of California at Berkley, 1993.

П Р И Л О Ж Е Н И Я

ПРИЛОЖЕНИЕ 1

ПРИМЕР ПРОГРАММИРОВАНИЯ ГЕНЕТИЧЕСКОГО АЛГОРИТМА

Программная реализация стандартного генетического алгоритма может быть выполнена на любом языке программирования. Ниже приводится пример описания пространства имен GeneticAl, содержащего реализацию ГА на языке C# на основе технологии объектно-ориентированного программирования [18].

Пространство имен GeneticAl содержит три класса Gene, Chromosome и GeneticAlgorithm, определяющих функциональность генетического алгоритма. Каждый класс содержит связанные с ним поля, методы и свойства. В табл. 1.1 – 1.3 приведено описание тех из них, которые имеют спецификатор доступа public.

Таблица 1.1

Основные свойства и методы класса Gene

Название	**Назначение**
BegPos	Начальная позиция гена в хромосоме
Size	Размер гена в битах
IntValue	Возвращает декодированное значение гена в формате целого числа
FloatValue	Возвращает декодированное значение гена в формате нормализованного вещественного числа

Таблица 1.2

Основные свойства и методы класса Chromosome

Название	**Назначение**
Fitness	Оптимальность хромосомы
GeneCount	Возвращает число генов в хромосоме
gene(i)	Возвращает ген номер i
Length	Размер хромосомы в битах
Clone(C)	Копирует содержимое хромосомы C
SetGeneSizes	Формирует структуру хромосомы

Таблица 1.3

Основные свойства и методы класса GeneticAlgorithm

Название	Назначение
Init	Инициализация ГА
SetGeneSizes	Устанавливает размеры и количество генов хромосомы
OneEpoch	Выполнение одной итерации ГА
BestChromosome	Возвращает ссылку на лучшую хромосому в популяции
WorstChromosome	Возвращает ссылку на худшую хромосому в популяции
Epoch	Возвращает номер поколения
BestFitness	Лучшее значение fitness-функции в популяции
WorstFitness	Худшее значение fitness-функции в популяции
TotalFitness	Возвращает суммарную оптимальность в популяции
AverageFitness	Возвращает среднюю оптимальность в популяции
GeneCount	Возвращает число генов в хромосоме
this[i]	Возвращает ссылку на хромосому с номером i
Fitness	Определяет виртуальный метод для вычисления fitness-функции
ChromosomeCount	Возвращает число хромосом в популяции
OptimizeMethod	Определяет характер оптимизации (минимизации, максимизация)
PCrossover	Определяет вероятность выполнения оператора кроссинговера
PMutation	Определяет вероятность выполнения оператора мутации
PInversion	Определяет вероятность выполнения оператора инверсии
UseElita	Включает или отключает стратегию элитизма

```
using System;
using System.Collections.Generic;
using System.Text;

namespace GeneticAl
{
   //Класс исключения для пространства имен Genetical
   public class GAException : Exception
   {
      public GAException(String ErrMessage):
```

```
                        base(ErrMessage)
        {}
    }

/*Класс описывает структуру гена и методы работы с
  данными гена */
   public class Gene
   {
      //Хромосома, содержащая ген
      private Chromosome _Chromosome;
      private byte[] GrayToDec ={ 0, 1, 3, 2, 7, 6, 4, 5
                        , 15, 14, 12, 13, 8, 9, 11, 10 };
      private byte[] DecToGray ={ 0, 1, 3, 2, 6, 7, 5, 4
                        , 12, 13, 15, 14, 10, 11, 9, 8 };

  /*Кодирование целочисленного представления гена в
    двоичный код Грея*/
      private void EncodeBinary(uint Value)
      {
         // выделяем тетрады
         int xTCount = Size >> 2;
         for (int ix = xTCount-1; ix >= 0; ix--)
         {
            //кодируем тетраду
            int xVal =DecToGray[Value & 15];
            Value = Value >> 4;
            //читаем тетраду
            int xMask = 1;
            for (int jx = 3; jx >= 0; jx--)
            {
               _Chromosome[BegPos + ix * 4 + jx]
                                 = ((xVal & xMask) > 0);
               xMask = xMask << 1;
            }
         }
      }
      /*Декодирование гена в двоичном коде Грея
        в целочисленное представление*/
      private uint DecodeBinary()
      {
         uint res=0;
         // выделяем тетрады
         int xTCount = Size >> 2;
         for (int ix = 0;ix < xTCount; ix++)
         {
```

```
            //читаем тетраду
            int xVal=0;
            int xMask=8;
            for (int jx = 0; jx <= 3; jx++)
            {
               if (_Chromosome[BegPos+ix*4+jx])
                  xVal=xVal+xMask;
               xMask = xMask >> 1;
            }
            //Декодирование
            res = res << 4;
            //Декодирование тетрады
            res = res | GrayToDec[xVal];
         }
         return res;
      }

   //Начальная позиция гена в хромосоме
      public int BegPos;
      public int Size;   //Размер гена в битах
      public double FlatValue
      {
         set
         {
            switch (Size)
            {
               case 4:
                  EncodeBinary((uint)Math.Round
                              (value * 15));
                  break;
               case 8:
                  EncodeBinary((uint)Math.Round
                              (value * 255));
                  break;
               case 12:
                  EncodeBinary((uint)Math.Round
                              (value * 4095));
                  break;
               case 16:
                  EncodeBinary((uint)Math.Round
                              (value * 65535));
                  break;
               case 20:
                  EncodeBinary((uint)Math.Round
                              (value * 1048575));
```

```
            break;
         case 24:
            EncodeBinary((uint)Math.Round
                         (value * 16777215));
            break;
         case 28:
            EncodeBinary((uint)Math.Round
                         (value * 268435455));
            break;
         case 32:
            EncodeBinary((uint)Math.Round
                         (value * 4294967295));
            break;
      }
   } //end set

   get
   {
      double gValue = DecodeBinary();
      double res=gValue;
      switch (Size)
      {
        case 4:
           res = gValue / 15;
           break;
        case 8:
           res = gValue / 255;
           break;
        case 12:
           res = gValue / 4095;
           break;
        case 16:
           res = gValue / 65535;
           break;
        case 20:
           res = gValue / 1048575;
           break;
        case 24:
           res = gValue / 16777215;
           break;
        case 28:
           res = gValue / 268435455;
           break;
        case 32:
           res = gValue / 4294967295;
```

```
                break;
            }
            return res;
        } //end get
    }
    public uint IntValue
    {
        set
        {
            EncodeBinary(value);
        } //end set

        get
        {
            return DecodeBinary();
        } //end get
    }
    public Gene(Chromosome C)
    {
        BegPos = 0;
        Size = 32;
        _Chromosome = C;
    }
};


/* Класс описывает хромосому и методы работы с данными
   хромосомы */
public class Chromosome
{
    private bool[] Alleles;

    private Gene[] _Genes;

    private int _GeneCount;

    public int GeneCount
    {
        get
        {
            return _GeneCount;
        }
    }
    public bool this[int index]
    {
```

```
            set
            {
               if((index >= 0)&(index < Alleles.Length))
                 Alleles[index] = value;
               else throw new GAException(String.Format
               ("Allele bit [{0}] is out of range",index));
            }
            get
            {
               return Alleles[index];
            }
         }

         public int Length
         {
            get
            {
               return Alleles.Length;
            }
         }
         public Gene gene (int index)
         {
            if ((index >= 0) & (index < _GeneCount))
               return _Genes[index];
            else throw new GAException(String.Format
           ("Gene index [{0}] is out of range", index));

         }
         public void SetGeneSizes(params byte[] gSizes)
         {
            _GeneCount = gSizes.Length;
            int ChLength = 0;
            Alleles = null;
            //Проверка кратности разрядности гена 4
            for (int gx = 0; gx < _GeneCount; gx++)
            {
               if ((gSizes[gx]%4 != 0)| (gSizes[gx] > 32))
                  throw new GAException(String.Format
                ("Gene [{0}] has not supported size {1}",
                  gx, gSizes[gx]));
            }


            /*Создание структуры хромосомы из _GeneCount
числа генов длиной gSizes[Индекс_Гена] */
```

```
        _Genes = new Gene[_GeneCount];
        for (int gx = 0; gx < _GeneCount; gx++)
        {
            _Genes[gx] = new Gene(this);
            _Genes[gx].Size=gSizes[gx];
            _Genes[gx].BegPos=ChLength;
            ChLength = ChLength + _Genes[gx].Size;
        }
        Alleles = new bool[ChLength];
    }

    public void Clone(Chromosome srcChromosome)
    {
        for (int ix = 0; ix < this.Length; ix++)
          this[ix] = srcChromosome[ix];

    }

    public Chromosome()
    {
        //По умолчанию создается хромосома с одним геном
        _GeneCount = 1;
        _Genes = new Gene[1];
        _Genes[0] = new Gene(this);
        //Длина хромосомы равна длине гена
        Alleles = new bool[_Genes[0].Size];

    }
    public double Fitness; //оптимальность хромосомы

};//End class TChromosome

//Класс описывает стандартный генетический алгоритм
public class GeneticAlgorithm
{
    private Random _rnd = new Random();
   //Старая и новая популяции

    private Chromosome[,] Populations;
    private int _ChromosomeCount;

    public enum OptimizeMethod {omMinimize,omMaximize};

    public double PCrossover;
    public double PMutation;
```

```
public double PInversion;

public int Epoch;
public double BestFitness
{
   get
   {
      return BestChromosome.Fitness;
   }
}
public double WorstFitness
{
   get
   {
      return WorstChromosome.Fitness;
   }
}
public double AverageFitness
{
   get
   {
      return TotalFitness / _ChromosomeCount;
   }
}

public Chromosome BestChromosome;
public Chromosome WorstChromosome;
public OptimizeMethod optimizeMethod;
public bool UseElita;

public int ChromosomeCount
{
  set
  {
     //Реинициализация популяции
     _ChromosomeCount = value;
     Populations = new
                Chromosome[2,_ChromosomeCount];
     for (int ix = 0; ix < _ChromosomeCount; ix++)
     {
        Populations[0,ix]=new Chromosome();
        Populations[1,ix]=new Chromosome();
     }
     BestChromosome = Populations[0, 0];
     WorstChromosome = Populations[0, 0];
```

```
        }
        get
        {
           return _ChromosomeCount;
        }

      }
      public int GeneCount
      {
         get
         {
            return Populations[0, 0].GeneCount;
         }
      }

      public Chromosome this [int index]
      {
         get
         {
            if ((index >= 0) &
                        (index < _ChromosomeCount))
               return Populations[(Epoch) % 2,index];
            else throw new GAException(String.Format
("Chromosome index [{0}] is out of range", index));
         }

      }
      public double TotalFitness;
      /*Виртуальная fitness-функция. Должна быть
        переопределена в дочерних классах*/
      public virtual double Fitness(Chromosome C)
      {
         return 0;
      }

      public GeneticAlgorithm()
      {
         Epoch = 0;
         TotalFitness = 0;
         ChromosomeCount = 1;
         optimizeMethod = OptimizeMethod.omMinimize;
         //Инициализация популяции из одной хромосомы
         Init();
         UseElita = false;
         PCrossover=0.95;
```

```
            PMutation = 0.001;
            PInversion = 0.01;
        }

        public void Init()
        {
            Epoch = 0;
           //Определяем длину хромосомы
            int xLen= Populations[0,0].Length;
            //Инициализация популяции случайными значениями
            for(int ix=0;ix<_ChromosomeCount;ix++)
                for (int jx = 0; jx < xLen; jx++)
                {
                    Populations[0, ix][jx] = _rnd.Next(2)>0;
                    Populations[1, ix][jx] = _rnd.Next(2)>0;
                }

        }

        public void SetGeneSizes(params byte[] gSizes)
        {
            for (int ix = 0; ix < _ChromosomeCount; ix++)
            {
                Populations[0, ix].SetGeneSizes(gSizes);
                Populations[1, ix].SetGeneSizes(gSizes);
            };
            Init();
        }

        //Оценивание хромосом новой популяции
        public /*protected*/virtual void
FitnessAssignment()
        {
            Chromosome C;
            double MinFitness, MaxFitness;
            MinFitness = MaxFitness = TotalFitness = 0;
            for (int ix = 0; ix < _ChromosomeCount; ix++)
            {
                C = Populations[Epoch % 2, ix];
                double CS = Fitness(C);
                C.Fitness = CS;
                if (ix == 0)
                {
                    MinFitness = CS;
                    MaxFitness = CS;
```

```
        }
        if (CS <= MinFitness)
        {
           MinFitness = CS;
           if (optimizeMethod ==
                     OptimizeMethod.omMinimize)
              BestChromosome = C;
           else WorstChromosome = C;
        }
        if (CS >= MaxFitness)
        {
           MaxFitness = CS;
           if (optimizeMethod ==
                     OptimizeMethod.omMaximize)
              BestChromosome = C;
           else WorstChromosome = C;
        }
        TotalFitness = TotalFitness + C.Fitness;
     }
  }

  //Генетические операторы
  //Отбор хромосомы
  protected Chromosome SelectChromosome()
  {
     Chromosome C1,C2,Res;
     Res = null;
     //Используется турнирный отбор
     do
     {
        C1 = Populations[Epoch % 2,
             _rnd.Next(_ChromosomeCount)];
        C2 = Populations[Epoch % 2,
             _rnd.Next(_ChromosomeCount)];
     }
     //Выбираются две разные хромосомы
     while (C1 == C2);
     /*В зависимости от характера оптимизации берем
       лучшую из них*/
     if (optimizeMethod == OptimizeMethod.omMinimize)
     {
        if (C1.Fitness < C2.Fitness)
           Res=C1;
        else Res=C2;
     }
```

```
    if (optimizeMethod == OptimizeMethod.omMaximize)
    {
       if (C1.Fitness > C2.Fitness)
          Res=C1;
       else Res=C2;
    }
    return Res;
}

protected void Crossover(Chromosome C1,
                  Chromosome C2, Chromosome ResC)
{
    //Определяется точка кроссинговера
    int xPos = _rnd.Next(C1.Length - 2) + 2;
    for (int ix = 0; ix < xPos; ix++)
      ResC[ix] = C1[ix];

    for (int ix = xPos; ix < C1.Length; ix++)
      ResC[ix] = C2[ix];

}
protected void Mutation(Chromosome C,
                                  Chromosome ResC)
{
    for (int ix = 0; ix < C.Length; ix++)
    {
       if (_rnd.NextDouble()<PMutation)
          ResC[ix] = !(C[ix]);
       else ResC[ix]=C[ix];
    }
}
public void Inversion(Chromosome C,
                                Chromosome ResC)
{
    //Определяется точка инверсии
    int xPos = _rnd.Next(C.Length - 2) + 2;
    for (int ix = xPos; ix < C.Length; ix++)
       ResC[ix-xPos] = C[ix];
    for (int ix = 0; ix < xPos; ix++)
       ResC[ix + C.Length-xPos] = C[ix];

}

//Основной цикл ГА
public void OneEpoch()
```

```
      {
         Chromosome C1, C2, C3;
         if (Epoch == 0)
            FitnessAssignment();
         //Использование стратегии элитизма
         if (UseElita)
            Populations[(Epoch + 1) % 2, 0].Clone
                              (BestChromosome);

         //Формирование следующего поколения
         for (int ix = Convert.ToInt32(UseElita);
                        ix < _ChromosomeCount; ix++)
         {
            C3 = Populations[(Epoch + 1) % 2, ix];
            C1 = SelectChromosome();
            C3.Clone(C1);
            double xP = _rnd.NextDouble();
            if (xP < PCrossover)
            {
               do
               {
                  C2 = SelectChromosome();
               }
               while (C1 == C2);
               Crossover(C1, C2, C3);
            }
            Mutation(C3,C3);
            xP = _rnd.NextDouble();
            if (xP < PInversion)
            {
               Inversion(C3,C3);
            }
         }
         Epoch++;
         FitnessAssignment();
      }
   }//end class GeneticAlgorithm
}
```

Далее пространство имен GeneticAl может использоваться в программах, где требуются возможности эволюционного моделирования. Для примера рассмотрим программную реализацию решения задачи из п.3.2. На рис. показан результат работы программы оптимизации с помощью ГА.

```
using System;
using System.Collections.Generic;
using System.Text;
using GeneticAl;

namespace Genetical_Test
{
   class Program
   {
      public class TestGA : GeneticAlgorithm
      {
   //Определение алгоритма вычисления fitness-функции
         public override double Fitness(Chromosome C)
         {
            double[] Params=new double[10];
            for(int idx=0;idx<=9;idx++)
            {
            //Декодирование значений параметров из генов
               Params[idx]=
                         500-C.gene(idx).FlatValue*1000;
            }
            double S=0;
            for(int idx=0;idx<=9;idx++)
            {
               S=S+(-Params[idx]*
             Math.Sin(Math.Sqrt(Math.Abs(Params[idx]))));
            }
            return S;
        }
      }
      static void Main(string[] args)
      {
         TestGA GA = new TestGA();
         //Установка значений управляющих параметров
         GA.ChromosomeCount =100;
         GA.SetGeneSizes(16, 16, 16, 16, 16,
                         16, 16, 16, 16, 16);
         GA.PCrossover = 0.9;
         GA.PMutation = 0.001;
         GA.Init();
         GA.FitnessAssignment();
         //Цикл генетического алгоритма из 300 поколений
         for (int ix = 0; ix < 300; ix++)
```

```
            {
               GA.OneEpoch();
            }
            Console.WriteLine("Параметры генетического
                               алгоритма:");
            Console.WriteLine("Ng = 300; Np = {0};
                  Pc = {1:F2}; Pm = {2:F4}; UseElita = {3}",
                        GA.ChromosomeCount,GA.PCrossover
                        ,GA.PMutation,GA.UseElita);
            //Вывод результата оптимизации
            Console.WriteLine("Оптимальное решение F =
                         {0:F3}",GA.BestChromosome.Fitness);
            for (int ix = 0; ix < GA.GeneCount; ix++)
            {
               Console.WriteLine("X{0} = {1:F3}", ix+1,
             500-GA.BestChromosome.gene(ix).FlatValue*1000);
            }
            Console.ReadLine();
         }
      }
}
```

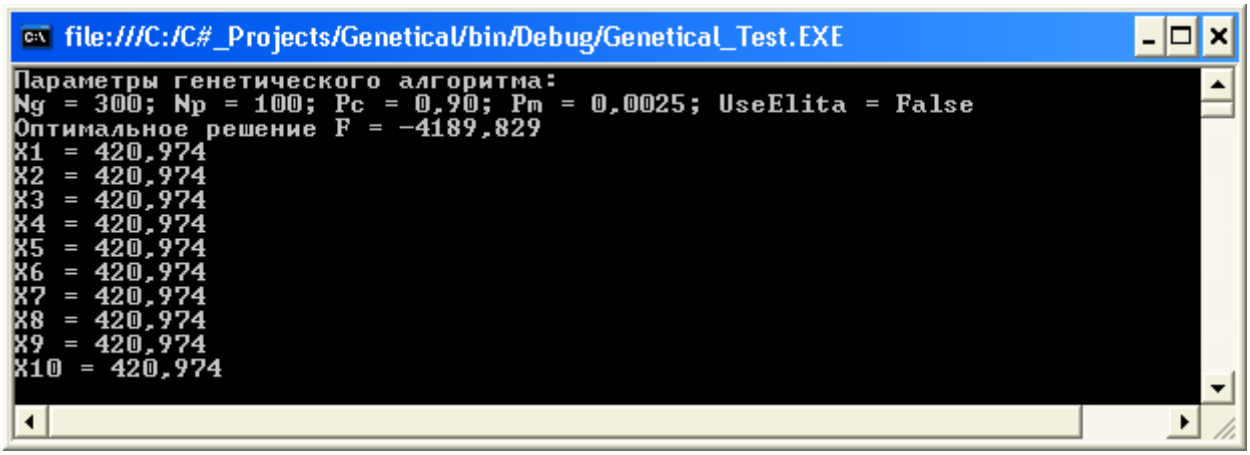


Рис. 1. Результат оптимизации многопараметрической функции с помощью ГА

Приведенная реализация ГА может служить основой для его совершенствования и создания модификаций. Для этого, учитывая объектно-ориентированную модель ГА, могут эффективно использоваться такие ее принципы как наследование и полиморфизм.

ПРИЛОЖЕНИЕ 2

ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ ЭВОЛЮЦИОННОГО МОДЕЛИРОВАНИЯ

Адаптация (адаптивное поведение) – процесс накопления и использования информации в системе, направленный на достижение ее (системы) оптимального состояния, при первоначальной неопределенности и изменяющихся внешних условиях.

Аллель – элементарный участок хромосомы, образующий ген.

Аутбридинг (outbreeding) – подход к выбору родительских хромосом, основанный на скрещивании хромосом, обладающих дальним родством (определяется на основе расстояния Хэмминга между ними).

Бионика – наука пограничная между биологией, генетикой и техникой, решающая инженерно-технические задачи на основе генетики и анализа структуры жизнедеятельности организмов.

Генетический алгоритм (ГА) (genetic algorithm) - метод эволюционного моделирования, основанный на использовании аналогий с природными процессами естественного отбора и генетических преобразований, предназначенный для решения задач оптимизации.

Генетическое программирование (genetic programming)– метод эволюционного моделирования, основанный на принципах генетического алгоритма и предназначенный для решения задач автоматического программирования.

Генетические операторы – преобразования, которым подвергаются хромосомы в процессе эволюции.

Ген – элемент, из которых состоит хромосома, как правило, соответствует закодированному значению одного параметра задачи.

Генотип – представление пространства поиска задачи в закодированном виде комплекса генов, содержащихся в наборе хромосом.

Длина шаблона – расстояние между первой и последней числовой позицией в шаблоне.

Искусственная жизнь (artificial life) – направление в эволюционном моделировании, которое занимается исследованием интеллектуального поведения искусственных систем в рамках адаптации, выживания, самоорганизации, построения децентрализованных систем.

Инбридинг (inbreeding) - подход к выбору родительских хромосом, основанный на скрещивании хромосом, обладающих близким родством.

Инверсия - оператор ГА, являющийся развитием оператора мутации и предназначенный для изменения с заданной вероятностью значений аллелей в случайно выбранном фрагменте хромосомы на противоположные.

Искусственный интеллект(artificial intelligence) – научное направление информатики, занимающееся разработкой и исследованием методов для автоматизации решения задач, требующих использования интеллектуальных функций человека.

Использование (exploitation) – характер поиска, при котором его направление определяется исключительно только на основе использования полученных точек. В ГА реализуется посредством оператора кроссинговера.

Исследование (exploration) - характер поиска, ориентированный на исследование новых областей поискового пространства. В ГА реализуется посредством оператора мутации.

Код Грея (Gray’s code) – двоичный код, при котором два соседних значения отличаются только в одном бите.

Коэволюционные системы – системы, использующие эволюционную модель с несколько развивающимися популяциями, причем оценка качества членов одной популяции может зависеть от состояния эволюционного процесса в других популяциях.

Кроссинговер - оператор ГА, предназначенный для обмена с заданной вероятностью генетическим материалом между хромосомами–родителями, полученными в результате отбора, с целью генерации хромосом-потомков для следующего поколения.

Локус – местоположение аллеля в хромосоме.

Метаэволюционный алгоритм – эволюционный алгоритм для решения задачи оптимизации значений управляющих параметров другого эволюционного алгоритма с целью наилучшего его применения к различным целевым функциям.

Мутация - оператор ГА, предназначенный для изменения с заданной вероятностью значения случайно выбранного аллеля в хромосоме на противоположное.

Неявный параллелизм (implicit parallelism) – свойство генетического алгоритма за одну итерацию исследовать число решений, превышающее размер популяции. Достигается это за счет того, что каждая хромосома содержит в себе фрагменты (шаблоны) новых хромосом, получаемых в ходе выполнения генетических преобразований.

Ниша (niche) – группа объектов популяции, которые имеют близкие значения fitness-функции.

Отбор – оператор ГА, предназначенный для выбора на основе заданного механизма его выполнения (пропорциональный отбор, ранжирование, турнирный отбор) наиболее «перспективных» хромосом-кандидатов в следующее поколение.

Поколение – новая популяция, возникающая после одной итерации работы эволюционного алгоритма.

Приспособленность хромосомы – значение fitness-функции, позволяющее оценить хромосомы относительно друг друга.

Порядок шаблона – число зафиксированных позиций (в двоичном кодировании – число нулей и единиц), представленных в шаблоне.

Популяционная генетика (population genetics) – раздел генетики, изучающий генетическое строение и динамику изменения в пространстве и во времени генетического состава популяций.

Популяция – набор хромосом, взаимодействующих друг с другом и обменивающихся генетическим материалом в процессе эволюционных преобразований.

Репродуктивные планы – первоначальное авторское название генетических алгоритмов, предложенное Дж. Холландом.

Репродукционная группа (mating pool) – подмножество хромосом популяции, прошедших отбор для применения к ним генетических операторов с целью формирования следующего поколения.

Скорость сходимости эволюционного алгоритма – определяется либо как математическое ожидание изменения различия между двумя соседними поколениями, либо изменения значения fitness-функции у лучших членов популяции в ряде последовательных поколений.

Стандартный генетический алгоритм – модель ГА, предложенная Дж. Холландом и включающая в себя базовые (по отношению к разработанным модификациям ГА) форму кодирования параметров задачи и генетические операторы.

Строительный блок (building block) – шаблон малой длины и низкого порядка, хромосомы, содержащие его, имеют оптимальность выше средней по популяции. Считается, что ГА ищут решения посредством синтеза как можно большего количества строительных блоков и комбинирования их друг с другом для получения решений с высокой оптимальностью.

Схема (schema) – шаблон, описывающий подмножество хромосом, имеющих совпадающие в некоторых позициях значения.

Управляющие параметры эволюционного алгоритма - предназначены для настройки качества и скорости работы метода эволюционного моделирования. В частности, для стандартного генетического алгоритма управляющими параметрами являются размер популяции, число поколений, вероятности применения генетических операторов.

Фенотип - представление пространства решений задачи в привычном для восприятия пользователем виде.

Фундаментальная теорема генетического алгоритма – теоретическое обоснование возможностей ГА за фиксированное число поколений находить субоптимальное решение задачи.

Хромосома – структура данных, содержащая закодированные в виде генов параметры задачи и описывающая ее решение в виде точки пространства поиска.

Эволюция (*биол.*) – необратимое историческое развитие естественный и искусственных систем.

Эволюция (*филос.***)** – потребность в реализации неиспользованных возможностей.

Эволюция – процесс чередования поколений, в которых хромосомы изменяют свои признаки, чтобы каждая новая популяция приближалась к оптимальному решению задачи.

Эволюционное моделирование (evolutionary computations) - направление в искусственном интеллекте, в основе которого лежат принципы и понятийный аппарат, заимствованные из популяционной генетики и объединяющее компьютерные методы (генетические

алгоритмы, генетическое программирование, эволюционное программирование и эволюционные стратегии) моделирования естественных эволюционных процессов.

Эволюционное проектирование (evolutionary design) – направление эволюционного моделирования, которое занимается исследованием возможностей применения эволюционных алгоритмов для решения задач САПР с целью перехода от автоматизированного к автоматическому проектированию технических систем.

Эволюционные стратегии (evolution strategy)- метод эволюционного моделирования, предложенный независимо от ГА и использующий в процессе генерации нового поколения преимущественно оператор мутации.

Эволюционное программирование (evolutionary programming) - метод эволюционного моделирования, разработанный независимо от ГА с целью моделирования интеллектуального поведения систем на основе конечных автоматов.

Элитный отбор – операция, когда хромосома с лучшей приспособленностью сохраняется в следующей популяции.

Deception – ситуация, при которой комбинация строительными блоками приводит к ухудшению fitness-функции. Является одной из проблем, приводящих к ограничению использования ГА в некоторых задачах.

Epistasis –ситуация, при которой имеет место взаимное влияния значений генов, следствием чего является возникновение проблемы Deception.

Fitness-ландшафт – гиперповерхность, получаемая вычислением значения fitness-функции в каждой точке пространства поиска.

Fitness-функция – аналитически или алгоритмически заданная функция, моделирующая собой условия для адаптации биологических особей во внешней среде.

Survival of the fittest – принцип «выживает наиболее приспособленный» предложенный Ч. Дарвином в качестве основы эволюционных преобразований.

ОГЛАВЛЕНИЕ

Научное издание

АВЕРЧЕНКОВ ВЛАДИМИР ИВАНОВИЧ
КАЗАКОВ ПАВЕЛ ВАЛЕРЬЕВИЧ

ЭВОЛЮЦИОННОЕ МОДЕЛИРОВАНИЕ И ЕГО ПРИМЕНЕНИЕ

Монография

Подписано в печать 30.11.2016.

Электронное издание для распространения через Интернет.

ООО «ФЛИНТА», 117342, г. Москва, ул. Бутлерова, д. 17-Б, комн. 324.

Тел./факс: (495) 334-82-65; тел. (495) 336-03-11.

E-mail: flinta@mail.ru; WebSite: www.flinta.ru