Лекция 11. Алгоритмы роевой оптимизации

Под роевой оптимизацией понимается класс алгоритмов, направленных на решение сложных оптимизационных задач (дискретная оптимизация, многомерная оптимизация и многокритериальная оптимизация), работа которых основана на моделировании коллективного поведения различных колоний живых организмов.

0.1 Роевой интеллект

Системы роевого интеллекта (swarm intelligence) представляют собой модели разнообразных типов колллективного поведения в колониях различных живых организмов — стаях птиц (bird flocks), косяках рыб (fish schools), колониях муравьев и т. п. Отличительной особенностью таких моделей является то, что каждый отдельный организм в этой модели выполняет очень простые действия, подчиненные своей локальной цели, взаимодействуя с ограниченным числом других организмов в рассматриваемой колонии.

0.1.1 Модель Рейнолдса

Работы в области роевого интеллекта были инспирированы исследованиями разнообразных реальных колоний. Например, метод роя частиц основан на исследовании и моделировании коллективного поведения стай птиц, проведенного Рейнолдсом. Целью этих исследований было построение реалистических анимаций полета стаи птиц. Особенностью поведения птичьих стай (например, стай ворон) является то, что в них нет никакого единого центра управления, тем не менее вся стая демонстрирует весьма цельное поведение. Рейнолдсом были сформулированы три простых принципа (рисунок 0.1), которых (в данной модели) должна придерживаться каждая отдельная птица:

1) каждая птица старается не приближаться к другим птицам (или другим объектам, например, препятствиям) на расстояние меньше некоторой заданной величины;

2) каждая птица старается выбрать свой вектор скорости наиболее близким к среднему вектору скорости среди всех птиц в своей локальной окрестности;

3) каждая птица старается расположиться в геометрическом центре масс своей локальной окрестности.

Первый принцип предназначен для того, чтобы избежать столкновений среди птиц, второй — координирует скорость и направление полета, третий — заставляет каждую птицу не отрываться от стаи (а в идеале — расположится внутри стаи). Иногда эти принципы входят в противоречие друг с другом, в этих случаях целесообразно ввести приоритет отдельных правил и следовать правилам с наивысшим приоритетом.

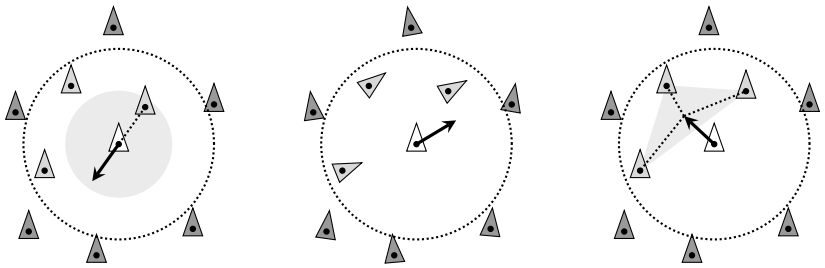

Рис. 0.1 Правила поведения птиц в модели Рейнолдса

Моделирование, основанное на перечисленных принципах, показывает весьма реалистическое поведение стай, способных, в частности, разделяться на несколько групп при встрече с препятствиями (домами, деревьями) и затем, спустя некоторое время, вновь соединяться в общую стаю.

0.1.2 Коллективная оптимизация

Переход от моделирования коллективного поведения к коллективной оптимизации основан на следующей биологической идее: организмы объединяются в колонии для улучшения своих условий существования — каждый организм в колонии в среднем имеет больше шансов на выживание в борьбе с хищниками, колония может более эффективно производить поиск, обработку и хранение пищи по сравнению с отдельными особями и т. д. Другими словами любая колония организмов в течение всего времени своего существования с той или иной степенью эффективности решает различные оптимизационные задачи, чаще всего — многокритериальные (например, максимизация количества пищи с одновременной минимизацией потерь от хищников).

Указанные соображения и легли в основу построения разнообразных математических методов оптмизации, один из которых — муравьиные алгоритмы — был подробно рассмотрен нами в предыдущей главе. В некоторых работах по роевой оптимизации принято относить к системам роевого интеллекта и генетические алгоритмы, которые при некоторой интерпретации могут рассматриваться, как колонии неких простых организмов, взаимодействие между которыми осуществляется с помощью операторов скрещивания и отбора.

В настоящей главе мы рассмотрим еще три подхода к построению алгоритмов роевой оптимизации — метод роя частиц (уже классический ал-

горитм оптимизации), алгоритмы бактериального поиска и пчелиные алгоритмы.

0.2 Метод роя частиц

Метод роя частиц был разработан Джеймсом Кеннеди и Расселом Эберхартом в 1995 году. Модель, положенная в основу этого метода, была получена упрощением модели Рейнолдса, в результате чего отдельные особи популяции стали представляться отдельными объектами, не имеющими размера, но обладающими некоторой скоростью. В силу крайней схожести с материальными частицами получившиеся простые объекты стали называться *частицами*, а их популяция — *роем*. Метод роя частиц предназначен для решения задач многомерной непрерывной оптимизации и основан на моделировании социального поведения колоний животных, выполняющих коллективный поиск мест с наилучшими условиями существования.

0.2.1 Базовый алгоритм

Пусть задана функция $f : R^n \to R$, требуется найти глобальный максимум этой функции, т. е. точку x_0 такую, что $f(x_0) \geq f(x)$ для любого $x \in R^n$. Суть подхода, на котором основан метод роя частиц, заключается в том, что глобальный максимум функции f ищется с помощью системы (роя), состоящей из m частиц. Частицы выполняют поиск, перемещаясь по пространству решений R^n. Положение i-ой частицы задается вектором $x_i \in R^n$, значение $f(x_i)$ определяет функцию качества этой частицы в текущий момент времени.

Каждая частица в рое обладает своей собственной скоростью $v_i \in R^n$, которая определяет как изменяются координаты частицы со временем:

$$x_i \leftarrow x_i + \tau v_i,$$

где τ — некоторая единица измерения скорости (продолжительность одного такта работы алгоритма, например, можно положить $\tau = 1$). Последовательность (дискретная) положений i-ой частицы будем называть ее *траекторией*.

Ключевая особенность метода роя частиц заключается в способе обновления скорости отдельных частиц, которое выполняется по формуле

$$v_i \leftarrow v_i + \alpha(p_i - x_i) + \beta(g - x_i). \tag{1}$$

Первое слагаемое в этой формуле представляет собой инерцию частицы — ее скорость на следующем временном шаге получается изменением текущей скорости.

Вектор p_i, фигурирующий во втором слагаемом, служит простейшей моделью *индивидуальной памяти* — он равен лучшей точке траектории i-ой частицы за все время ее существования (от начала работы алгоритма до текущего момента времени). Говорят, что второе слагаемое реализует *принцип простой ностальгии* — каждая частица «хочет» вернуться в ту точку, где ею было достигнуто лучшее значение функции f.

Вектор g, участвующий в вычислении третьего слагаемого, представляет собой лучшую точку, обнаруженной за время своего существования всем роем, т. е. представляет собою некую *коллективную память*. Следовательно, само третье слагаемое определяет некоторую простую схему социального взаимодействия между отдельными частицами роя.

Другими словами, изменение скорости каждой частицы (т. е. ее ускорение) определяется как некая взвешенная сумма двух векторов, первый из которых направлен на лучшую точку, обнаруженную данной частицей, а второй — на лучшую точку, обнаруженную всем роем. Коэффициенты α и β могут выбираться из разных соображений. Численные эксперименты показали, что лучшей является вероятностная схема — либо оба коэффициента выбираются случайным образом из диапазона $[0, 1]$, либо значение α выбирается случайным образом из этого диапазона, а значение β полагается равным $1 - \alpha$.

Таким образом, метод роя частиц формально может быть описан следующим образом:

ParticleSwarmOptimization(f, n, m)

Случайное распределение m частиц по пространству решений R^n
Случайная инициализация скоростей частиц
$g \leftarrow x_1$
for $i \in [1 \dots m]$ **do**
  $p_i \leftarrow f(x_i)$
  if $f(p_i) > f(g)$ **then**
    $g \leftarrow p_i$
while Не выполнен критерий останова **do**
  for $i \in [1 \dots m]$ **do**
    $\alpha \leftarrow$ Random$(0, 1)$
    $\beta \leftarrow$ Random$(0, 1)$
    $v_i \leftarrow v_i + \alpha(p_i - x_i) + \beta(g - x_i)$
    $x_i \leftarrow x_i + \tau v_i$
    if $f(x_i) > f(p_i)$ **then**
      $p_i \leftarrow x_i$
      if $f(x_i) > f(g)$ **then**
        $g \leftarrow x_i$
return g

Метод роя частиц является, по сути, метаэвристикой, хорошо зарекомендовавшей себя при решении различных оптимизационных задач. Его отличительной особенностью от многих других методов является то, что для метода роя частиц необходимо уметь вычислять только значение оптимизируемой функции, но не ее градиент. Т. е. функция, подлежащая оптимизации, не обязана быть дифференцируемой, более того, она может быть разрывной, зашумленной и т. п. С другой стороны, в силу того, что метод оперирует понятием скорости частиц, необходимым условием его применимости является непрерывность области определения функции. В частности, это означает, что данный метод неприменим напрямую к задачам дискретной оптимизации.

0.2.2 Вариации метода

Большая часть предложенных вариаций базового алгоритма Кеннеди и Эберхарта касается модификации формулы (1). Основная проблема, связанная с применением этой формулы заключается в том, что возникающие при этом скорости частиц мог быть сколь угодно большими, это привносит некоторую неустойчивость в работу метода. Простейший способ ограничить скорости частиц заключается в прямом запрете на превышение некоторой максимальной скорости $v_{\max}$: если модуль скорости, вычисленной по формуле (1), оказывается больше величины $v_{\max}$, то уменьшить скорость по величине (но не по направлению) до $v_{\max}$. Другой, более естественный, способ ограничения модуля скорости заключается в введении некоторого «сопротивления среды», задаваемого положительным коэффициентом $\gamma < 1$:

$$v_i \leftarrow \gamma \left(v_i + \alpha(p_i - x_i) + \beta(g - x_i) \right).$$

Другие, менее значительные, изменения базового алгоритма касаются следующих моментов:

- способ начального распределения частиц по пространству решений;
- способ выбора начальной скорости частиц;
- схема вычисления коэффициентов α и β;
- обновление вектора g только после вычисления новых координат всеми частицами роя;
- использование в формуле (1) вектора g, вычисленного не для всего роя (глобальная операция), а только для некоторого подмножества i-ой частицы (ее локальной окрестности в некоторой метрике, не связанной с пространством решений).

Также были предложены и исследованы вариации метода роя частиц, применимые к задачам дискретной оптимизации. Проблемой оригинального алгоритма является то, что в дискретном пространстве решений (например, на множестве двоичных последовательностей) понятие скорости частицы полностью теряет свой смысл. Один из вариантов решения этой проблемы, предложенный самим Эберхартом, заключается в том, что k-ая компонента скорости частицы, ограниченная интервалом $[0, 1]$ трактуется, как вероятность изменения k-ой координаты этой частицы — чем больше скорость, тем чаще меняется координата (например, 0 на 1 и наоборот). Этот прием позволяет отделить пространство решений от пространства скоростей и, в частности, эффективно решать задачи оптимизации в дискретных пространствах решений.

0.2.3 Параллельная реализация

Имеется две базовые схемы распараллеливания метода роя частиц — с централизованным и локальным управлением. В любом случае частицы помещаются на различные процессоры вычислительной системы, либо по одной (мелкозернистый параллелизм), либо группами (крупнозернистый параллелизм).

В случае централизованного управления используется базовый вариант метода с глобальным вычислением вектора g. Каждый рабочий процесс независимо от остальных вычисляет координаты x_i, вектор скорости v_i и вектор p_i. После этого выполняется стадия синхронизации, на которой процесс обменивается информацией с мастер-процессор: рабочий процесс посылает мастер-процессу вычисленное значение вектора p_i, которое используется для обновления вектора g, после этого мастер-процесс возвращает рабочему процессу текущее значение вектора g. Т. к. каждый рабочий процесс обменивается информацией с мастер-процессом в порядке некоторой очереди, то при увеличении числа рабочих процессов будет возрастать и время ожидания в такой очереди, что приведет к снижению общей эффективности распараллеливания. Снизить относительное время ожидания можно, если разрешить каждой частице делать несколько несколько перемещений, а не одно, между двумя последующими стадиями синхронизации.

В варианте с локальным управлением мастер-процесс вообще не используется. Вместо этого вводится понятие локальной окрестности частицы, под которой понимается некоторое подмножество частиц, распределенных на процессоры, соседние с данным. Работа параллельного алгоритма производится по той же схеме, что и для первого варианта, с тем отличием, что вектор g вычисляется теперь не глобально, а локально внутри локальной окрестности каждой частицы: каждый процесс опрашивает всех своих соседей и выбирает лучший из имеющихся у этих процессов векторов g. Этот вариант является более предпочтительным при реализации на массивно-параллельных вычислительных системах, т. к. он хорошо и легко масштабируется.

0.3 Бактериальный поиск

Алгоритм бактериального поиска, разработанный Кевином Пассино, основан на моделировании процесса поиска питательных веществ бактериями наиболее изученного вида *E. coli*, хотя поведение таких бактерий свойственно и бактериям многих других видов.

0.3.1 Хемотаксис бактерий

Целью движения бактерий вида *E. coli* является поиск наиболее благоприятных условий для их существования (много пищи и отсутствие ядовитых веществ). Клетка каждой бактерии снабжена специальными жгутиками, которые могут вращаться в двух направлениях (по и против часовой стрелки). При вращении в одну сторону (против часовой стрелки, если смотреть ходу движения), жгутики действуют согласовано, что приводит клетку в движение, которое в отсутствии помех будет равномерным и прямолинейным. Если же жгутики начинают вращаться в противоположную сторону, то согласованность исчезает и клетка начинает беспорядочно кувыркаться на месте, что фактически приводит к выбору нового случайного направления ее движения. Эффективность поиска питательных веществ бактерией определяется выбором частоты смены этих двух режимов движения.

Каждая клетка *E. coli* во время своего движения фактически отслеживает градиент количества питательных веществ. Если клетка движется в

направлении градиента (количество питательных веществ постепенно возрастает), то частота кувырканий сокращается — участки прямолинейного движения становятся более длинными. Если же количество питательных веществ начинает снижаться (движение против градиента), то эта частота увеличивается, что приводит к частой смене направления движения, пока не будет найдено правильное направление. Такой способ движения в микробиологии называется *хемотаксисом*.

Особенностью колоний бактерий, которой не обладают колонии высших животных (птиц, рыб и т. п.), является то, что бактерии могут очень быстро размножаться. Например, бактерии *E. coli* делятся примерно раз в 12 минут, т. е. за одни сутки одна бактерия теоретически может организовать колонию, состоящую примерно из 2^{120} клеток. Это свойство позволяет бактериям эффективно обживать области с высоким содержанием питательных веществ.

Собственная скорость движения бактерий *E. coli* является достаточно низкой (приблизительно 0.03 милимметра в секунду), что не позволяет бактериям самостоятельно передвигаться на большие расстояния. Широкое распространение бактерий в природе обеспечивается их переносом некоторыми носителями, например, водой или какими-нибудь животными. Явление переноса также используется в алгоритме бактериального поиска.

0.3.2 Алгоритм бактериального поиска

Пусть требуется найти глобальный максимум функции $f(x)$ (функция качества), определенной в некотором пространстве решений (для простоты рассмотрим случай n-мерного пространства R^n). Алгоритм, предложенный Кевином Пассино, заключается в том, что решение оптимизационной задачи ищется с помощью колонии искусственных бактерий, количество которых s поддерживается на одном и том же фиксированном уровне. Сам алгоритм состоит из нескольких вложенных друг в друга циклов.

Самый внутренний цикл алгоритма моделирует движение (хемотаксис) бактерий, каждая из которых выполняет несколько шагов заданной длины δ в текущем направлении, задаваемом вектором e_i единичной длины. Этот цикл завершается либо в том случае, когда значение функции качества в новой точке окажется меньше текущего значения, либо после выполнения максимально допустимого числа шагов. После завершения цикла, каждая бактерия случайным образом выбирает новое направление движения.

После того, как будет выполнено заданное число циклов движения, производится отбор лучших бактерий и их деление. Для этого все бактерии упорядочиваются по возрастанию функции качества f, усредненной по всему времени их жизни. Худшая часть всех бактерий удаляется, а соответствующая часть бактерий, оставшихся в живых, делится, причем дочерние бактерии остаются в тех же точках, где располагались их родители.

После завершения нескольких итераций цикла воспроизведения выполняется перенос (рассеяние) бактерий. Для этого каждая бактерия с некоторой небольшой вероятностью переносится в случайную точку пространства решений. Этот цикл является внешним и он завершается при выполнении некоторого условия останова (достигнута макисмально возможная итерация, получено решение с заданной степенью точности и т. д.).

Как следует из описания алгоритма, основная область его применения явялется такой же, как и для метода роя частиц — оптимизация функций с непрерывной областью определения.

0.3.3 Роевое поведение бактерий

Описанный выше базовый вариант алгоритма бактериального поиска использует только косвенную схему коммуникации между отдельными бактериями колонии, реализованную через механизм отбора лучших бактерий в текущей популяции. Тем не менее, имеется вариант алгоритма, в котором коммуникация осуществляется в явной форме. Этот вариант имеет биологическое обоснование, которое заключается в том, что каждая бактерия способна выделять некоторые вещества, сигнализирующие о наличии либо питательных, либо ядовитых веществ в том месте, где она находится. Эта информация «считывается» другими бактериями и может повлиять на их поведение.

Чтобы учесть такой способ коммуникации, функция качества модифицируется добавлением специального слагаемого:

$$\tilde{f(x)} = f(x) + \mu \sum_{i=1}^{s} h(\|x_i - x\|),$$

где функция $h(r)$, определяющая «сигнал» i-ой бактерии, дошедший от нее до точки x, имеет следующую форму.

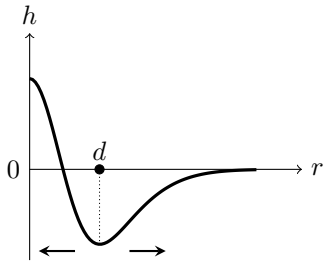

Рис. 0.2 Функция $h(r)$

Смысл такой динамической добавки к функции качества заключается в том, что две бактерии, оказавшиеся на расстоянии меньшем d друг от друга, испытывают взаимное притяжение (напомним, что бактерии движутся в направлении градиента функции качества), а бактерии, расстояние между которыми больше d — небольшое отталкивание. В результате это приводит к образованию отдельных кластеров из бактерий, каждый из которых «обживает» свою область в пространстве решений. В некоторых задачах такое поведение обеспечивает более эффективный поиск глобального максимума.

0.4 Алгоритм пчелиного поиска

Алгоритм оптимизации, моделирующий процесс поиска пчелами мест с высоким содержанием нектара, разработан в 2005 году Фамом. Базовый вариант этого алгоритма является комбинацией алгоритмов локального и случайного поиска.

0.4.1 Поведение пчел в природе

Пчелы могут выполнять поиск пищи одновременно во многих направлениях на расстоянии более чем 10 километров от улья.

На первом этапе небольшое количество разведчиков выполняет случайный поиск мест с высоким содержанием нектара. При возвращении в улей пчелы исполняют специальный «танец» (являющийся, по сути, специфической формой коммуникацией между пчелами), в котором оказывается закодированной информация о расстоянии до найденного источника пищи, о направлении к этому источнику, и о качестве и количестве найденного там нектара. Чем ближе окажется источник и чем выше его качество, тем больше пчел последует за данным разведчиком. При возвращении в улей пчелы вновь исполняют свой танец, набирая новых последователей и т. д. Такая, в общем несложная, процедура позволяет пчелиной колонии достаточно эффективно выполнять на больших площадях поиск и сбор нектара.

Поведение пчелиной колонии является адаптивным. В то время как другие пчелы собирают нектар, разведчики продолжают искать новые перспективные места, что позволяет вести колонии мониторинг общей ситуации — как только некоторый источник нектара начинает истощаться, пчелы перестраиваются на другие источники, которые были за это время обнаружены разведчиками.

0.4.2 Описание алгоритма

Пусть решается стандартная задача оптимизации (максимизации) функции $f(x)$ в пространстве решений R^n. Алгоритм пчелиного поиска использует для своей работы следующие числовые параметры:

- s — число пчел-разведчиков;
- p — число выбранных точек (решений) для дальнейшего исследования ($p < s$);
- e — число лучших (элитных) точек ($e < p$);
- s_e — число пчел для более полного исследования e элитных решений;
- s_p — число пчел для более полного исследования оставшихся $(p - e)$ выбранных решений;
- δ — размер окрестности, в которой пчелы выполняют более тщательный поиск.

Сам алгоритм тогда заключается в выполнении следующих шагов. В начале случайным образом выбирается m решений, каждое из которых представляет собой одну пчелу-разведчика. Лучшие m решений исследуются более тщательно (локальный поиск) — для этого в их δ-окрестности просматривается r случайных точек ($r = s_e$ или $r = s_p$ в зависимости от элитности данной точки), из которых выбирается лучшая. Оставшиеся $(s - m)$ решений заменяются на случайные точки из пространства решений (случайный поиск). Алгоритм завершается, когда оказывается выполненным некоторое условие останова (достигнута необходимая точность, исчерпано число итераций и т. п.).

Таким образом, алгоритм пчелиного поиска может быть формально записан следующим образом.

BeesAlgorithm(f, n)

Выбираем s случайных решений $x_i \in R^n$
while Не выполнено условие останова **do**
  Упорядочиваем решения по *убыванию* функции качества
  ▷ *Локальный поиск*
  for $i \in [1 \dots m]$ **do**
    if $i > e$ **then**
      $r \leftarrow s_e$ ▷ *число пчел для исследования элитных мест*
    else
      $r \leftarrow s_p$ ▷ *число пчел для исследования других выбранных мест*
    ▷ *поиск в локальной окрестности точки* x_i
    $\tilde{x} \leftarrow x_i$
    for $j \in [1 \dots r]$ **do**
      $x \leftarrow$ случайное решение в δ-окрестности точки x_i
      if $f(x) > f(\tilde{x})$ **then**
        $\tilde{x} \leftarrow x$
    $x_i \leftarrow \tilde{x}$
  ▷ *Случайный поиск*
  for $i \in [m+1 \dots s]$ **do**
    $x_i \leftarrow$ случайное решение в R^n
return Лучшее найденное решение

Отличительным свойством данного алгоритма от рассмотренных выше метода роя частиц и алгоритма бактериального поиска является то, что он одинаково хорошо применим для решения как задач с непрерывной функцией качества, так и задач дискретной оптимизации.

Литература

[1] S. Achasova, O. Bandman, V. Markova, et al., *Parallel Substitution Algorithm. Theory and Application.*, Singapore: World Scientific, 1994.

[2] L. M. Adleman, *Molecular Computation Of Solutions To Combinatorial Problems*, Science, 266, 11, pp. 1021–1024, 1994.

[3] E. Alba, B. Dorronsoro, *Cellular Genetic Algorithms*, Springer, 2008.

[4] D. Boneh, C. Dunworth, R. J. Lipton, J. Sgall, *On the Computational Power of DNA*, DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science, 71, 1996.

[5] C. Detrain, J. L. Deneubourg, *Self-Organized Structures in a Superorganism: Do Ants Behave Like Molecules?*, Physics of Life Reviews 3, no. 3, pp. 162-187, 2006.

[6] M. Dorigo, M. Birattari, T. Stutzle, *Ant Colony Optimization*, Technical Report No. TR/IRIDIA/2006-023, September 2006.

[7] S. Goss, S. Aron, J.-L. Deneubourg, J. M. Pasteels, *Self-organized shortcuts in the Argentine ant*, Naturwissenschaften, vol. 76, pp. 579–581, 1989.

[8] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, 1975.

[9] J. Kennedy, R. C. Eberhart, *Particle swarm optimization*, Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, pp. 1942–1948, 1995.

[10] K. M. Passino, *Biomimicry of bacterial foraging for distributed optimization and control*, IEEE Control Systems Magazine, 22, pp. 52–67, 2002.

[11] Gh. Paun, *Computing with Membranes*, Journal of Computer and System Sciences, 61, 1 (2000), 108-143.

[12] Gh. Paun, *P Systems with Active Membranes: Attacking NP Complete Problems*, CDMTCS Research Report Series, CDMTCS-102, May 1999.

[13] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, M. Zaidi, *The Bees Algorithm — A Novel Tool for Complex Optimisation Problems*, Proceedings of IPROMS 2006 Conference, pp. 454–461, 2006.

[14] P. Prusinkiewicz, A. Lindenmayer, *The Algorithmic Beauty of Plants*, Springer-Verlag, 1996 (http://algorithmicbotany.org/papers/abop)

[15] C. W. Reynolds, *Flocks, herds and schools: a distributed behavioral model*, Computer Graphics, 21, 4, pp. 25-34.

[16] G. Rozenberg, A. Salomaa, *The Mathematical Theory of L Systems*, Academic Press, 1980

[17] T. Stutzle, H. Hoos, *MAX–MIN Ant System*, Future Generation Computer Systems, vol. 16, no. 8, pp. 889–914, 2000.

[18] *The Oxford handbook of membrane computing*, edited by G. Paun, G. Rozenberg and A. Salomaa, Oxford University Press, 2010.

[19] T. Weise, *Global Optimization Algorithms – Theory and Application*, http://www.it-weise.de/.

[20] D. Whitley, *A Genetic Algorithm Tutorial*, Statistics and Computing (4): 65-85, 1994.

[21] S. Wolfram, *A New Kind of Science*, Wolfram Media, 2002.

[22] Hsu-Chun Yen, *Introduction to Petri Net Theory*, Recent Advances in Formal Languages and Applications, 2006, pp. 343-373.

[23] М. Гарднер, *Крестики–нолики*, М.: Мир, 1988

[24] В. Котов, *Сети Петри*, М.: Наука, 1984.

[25] А. А. Марков, Н. М. Нагорный, *Теория алгорифмов*, М.: Наука, 1984.

[26] М. Минский, С. Пейперт, *Персептроны*, М.: Мир, 1971.

[27] Дж. фон Нейман, *Теория самовоспроизводящихся автоматов*, М.: Мир, 1971

[28] Г. Паун, Г. Розенберг, А. Саломаа, *ДНК-компьютер. Новая парадигма вычислений*, М.: Мир, 2003.

[29] Дж. Питерсон, *Теория сетей Петри и моделирование систем*, М. : Мир, 1984.

[30] Ф. Розенблатт, *Принципы нейродинамики: Перцептроны и теория механизмов мозга*, М.: Мир, 1965.

[31] Т. Тоффоли, Н. Марголус, *Машины клеточных автоматов*, М.: Мир, 1991

[32] Ф. Уоссермен, *Нейрокомпьютерная техника: Теория и практика*, М.: Мир, 1992.

[33] С. Хайкин, *Нейронные сети: Полный курс*, М.: «Вильямс», 2006.