Описание роевых алгоритмов, инспирированных неживой природой и бактериями, для использования в онтологической модели

О.С. Смирнова, А.В. Богорадникова, М.Ю. Блинов

Аннотация **– В статье представлены сведения по роевым алгоритмам, инспирированным неживой природой и бактериями, включающие описание биологических предпосылок и самих алгоритмов, на основе которых можно формировать ветки (листья) дерева онтологической модели бионических технологий.**

Ключевые слова **– роевые алгоритмы; бионические технологии; онтологическая модель; алгоритм гравитационного поиска; алгоритм интеллектуальных капель; стохастический диффузионный поиск; оптимизация передвижением бактерий.**

Введение

Одной из целей построения онтологической модели бионических технологий [1] является создание основы для разработки базы знаний интеллектуальной системы информационной поддержки процессов создания и развития перспективных бионических технологий [2, 3]. Указанная система должна обеспечивать накопление, организацию и использование информационных ресурсов в области бионики.

Реализация новых подходов к обеспечению проблемно-ориентированного поиска информационных ресурсов в области бионики может быть осуществлена при наличии развёрнутой онтологической модели с достаточно большой степенью детализации.

Детализация узла рассматриваемой онтологической модели «роевой интеллект» осуществляется в нескольких направлениях, одним из которых является ветка роевых алгоритмов, инспирированных неживой природой и бактериями.

Для того, чтобы правильно построить ветвление необходимо иметь большое количество сведений о биологических предпосылках и самих роевых алгоритмах. Далее представлены примеры собранных сведений для описания веток (листьев) дерева онтологической модели по алгоритмам, инспирированным неживой природой и бактериями:

– алгоритм гравитационного поиска;
– алгоритм интеллектуальных капель;
– стохастический диффузионный поиск;
– оптимизация передвижением бактерий.

Примеры описания листьев дерева онтологической модели по алгоритмам, основанным на поведении насекомых и животных, представлены в [4].

1 Алгоритм гравитационного поиска

Гравитационный поиск (англ. Gravitational Search, GS) является очень молодым алгоритмом, появившемся в 2009 году. Основу гравитационного поиска составляют законы гравитации и взаимодействия масс. Данный алгоритм похож на методы роя частиц (Particle Swarm Optimization – PSO), так как базируется на развитии многоагентной системы [5].

GS оперирует двумя законами:

1) законом тяготения (рисунок 1): каждая частица притягивает другие и сила притяжения между двумя частицами прямо пропорциональна произведению их масс и обратно пропорциональна расстоянию между ними (следует обратить внимание на то, что в отличие от всемирного закона тяготения используется не квадрат расстояния. Рашеди объясняет это тем, что во всех тестах это дает лучшие результаты);

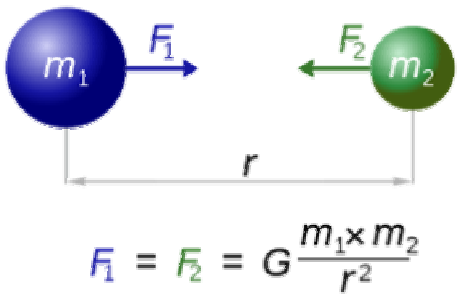

Рисунок 1 – Закон тяготения

2) законом движения: текущая скорость любой частицы равна сумме части скорости в предыдущий момент времени и изменению скорости, которое равно силе, с которой воздействует система на частицу, делённой на инерциальную массу частицы.

Статья получена 03.12.2015 г.

Исследование выполнено федеральным государственным бюджетным образовательным учреждением высшего образования «Московский государственный университет информационных технологий, радиотехники и электроники» (МИРЭА, МГУПИ) за счет гранта Российского научного фонда (проект №14-11-00854).

О.С. Смирнова, МГТУ МИРЭА (e-mail: mail.olga.smirnova@yandex.ru).

А.В. Богорадникова, МГТУ МИРЭА (e-mail: bogoradnikova@mirea.ru)

М.Ю. Блинов, МГТУ МИРЭА (e-mail: xxmaxxx2009@gmail.com)

Имея в арсенале эти два закона, метод работает по следующему плану:

1) генерация системы случайным образом;

2) определение приспособленности каждой частицы;

3) обновление значений гравитационной постоянной, лучшей и худшей частиц, а также масс;

4) подсчет результирующей силы в различных направлениях;

5) подсчет ускорений и скоростей;

6) обновление позиций частиц;

7) повторений шагов 2 – 6 до выполнения критерия окончания (либо превышение максимального количества итераций, либо слишком малое изменение позиций, либо любой другой осмысленный критерий).

$$F_{ij}(t) = G(t)\frac{M_{pi}(t) \times M_{aj}(t)}{\left\|p_i, p_j\right\| + \varepsilon}\left(p_j(t) - p_i(t)\right), \quad (1)$$

где M_{aj} – активная гравитационная масса j-й частицы;

M_{pi} – пассивная гравитационная i-й частицы;

$G(t)$ – гравитационная постоянная в соответствующий момент времени;

ε – малая константа

$\|.,.\|$– евклидово расстояние между частицами.

Предположим, у нас есть некоторая функция, которую необходимо минимизировать: $f(x): R^n \rightarrow R$

Кроме этого, есть область R, в которой генерируются начальные позиции частиц. В соответствии с планом работы гравитационного поиска, начинается все с генерации системы частиц:

$$S = \{p_i = (p_i^1, p_i^2, \dots, p_i^n) \in X\}_{i=1}^N, \quad (2)$$

где N — максимальное количество частиц в системе.

Посчитаем ускорения и скорости:

$$v_i(t+1) = (\zeta_1, \dots, \zeta_n)^T * v_i(t) + \underbrace{\frac{F_i(t)}{M_{ii}(t)}}_{a_i(t)}, \quad (3)$$

где * — операция покомпонентного умножения векторов;

ζ_i – случайная величина, равномерно распределенная от нуля до единицы;

$M_{ii}(t)$ – инертная масса i-й частицы.

Далее необходим пересчёт положения частиц:

$$p_i(t+) = p_i(t) + v_i(t+1). \quad (4)$$

Остаётся два вопроса: как изменяется гравитационная постоянная и как рассчитывать массы частиц. Значение гравитационной постоянной должно определяться монотонно убывающей функцией, зависящей от начального значений постоянной G_0 и момента времени t, т.е. $G(t)=G(G_0,t): R^+ \rightarrow R^+$.

Далее можно приступить к пересчёту масс. В простейшем случае все три массы (пассивная, активная и инерциальная) приравниваются:

$$M_{ai} = M_{pi} = M_{ii} = M_i$$

Тогда значение масс можно пересчитать по формуле:

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^N m_j(t)}, \quad (5)$$

где

$$m_i(t) = \frac{f(p_i) - \max\limits_{j \in \{1,\dots,N\}} f(p_j)}{\min\limits_{j \in \{1,\dots,N\}} f(p_j) - \max\limits_{j \in \{1,\dots,N\}} f(p_j)}, \quad (6)$$

Достоинства рассматриваемого метода:

– простота реализации;

– на практике метод точнее, чем генетические алгоритмы с вещественным кодированием и классический PSO;

– большая скорость сходимости, чем у генетических алгоритмов с вещественным кодированием и классического PSO.

К недостаткам рассматриваемого метода следует отнести следующие:

– не самая большая скорость за счет необходимости пересчёта многих параметров;

– большая часть достоинств теряется при оптимизации мультимодальных функций (особенно больших размерностей), так как метод начинает быстро сходиться к некоторому локальному оптимуму, из которого сложно выбраться, так как не предусмотрены процедуры, похожие на мутации в генетических алгоритмах.

Оригинальный алгоритм склонен к преждевременной сходимости в случае поиска экстремума сложных мультимодальных целевых функций. В связи с этим разработано значительное число модификаций алгоритма, имеющих целью преодоление данного его недостатка.

Вариантом решения данной проблемы является введение в оригинальный алгоритм гравитационного поиска операторов, так называемых знаковой пунктуации и переупорядочивающей мутации.

2 Алгоритм интеллектуальных капель

Алгоритм интеллектуальных капель (англ. IWD) основывается на алгоритме оптимизации, который использует методы естественных рек и способы, которыми они находят почти оптимальные пути к месту назначения. Алгоритм интеллектуальных капель находит оптимальные или близкие к оптимальным пути, вытекающие из реакции, протекающие между каплями воды, когда вода течёт руслом реки.

В IWD алгоритме, несколько искусственных капель воды, которые зависят друг от друга, способны менять свое окружение таким образом, что находят оптимальный путь по пути наименьшего сопротивления. IWD алгоритм – это конструктивный популяционно-ориентированный алгоритм оптимизации [6].

В природе капли воды наблюдаются в основном в реках, где они образуют огромные массы (рои капель воды). Пути, которыми текут природные реки, были созданы роями из капель воды. Для роя капель воды, реки, в которых они протекают, являются частью окружающей среды, которая была значительно изменена роем, а также будет изменена в будущем. Более того, сама среда оказывает существенное влияние на пути следования капель воды. Им оказывают сопротивление берега реки. Например, рою капель воды сопротивляются больше те части окружающей среды, из которых состоит жесткий грунт, чем части из мягкого грунта. Естественное русло реки является результатом конкуренции между каплями воды и окружающей

средой, которая оказывает сопротивление движению капель воды.

Одной из особенностей капли воды, текущей в реку, является её скорость. Предполагается, что каждая река может также нести определённое количество почвы. Таким образом, капля воды способна переносить некоторое количество почвы с одного места на другое место. Эта почва, как правило, передается от быстрых частиц к медленным частицам. Т.е. почва, схваченная рекой вместе с быстрым течением, осядет в месте с медленным течением.

Три очевидных изменения произойдет в течение этого переходного периода:

– скорость капли воды увеличивается;

– насыщенность грунтом капли воды увеличивается;

– между этими двумя точками, количество грунта в русле уменьшается [6].

Почва русла удаляется каплями воды, добавляясь в почву, которую переносит эта капля. Кроме того, скорость капли воды увеличивается в переходный период. Выше было отмечено, что каждая капля воды имеет свою скорость. Эта скорость играет важную роль в сборе почвы от русла реки.

Капля воды с высокой скоростью собирает больше грунта, чем капля с менее быстрой скоростью. Таким образом, капли воды с большей скоростью удаляют больше грунта со дна реки, чем капли воды с меньшей скоростью. Удаление грунта, таким образом, связано со скоростью капли воды.

Скорость капли воды возрастает на пути с низким уровнем почвы больше, чем на пути с высоким уровнем почвы. Таким образом, путь с небольшим количеством почвы позволяет текущей капле воды собрать больше почвы и получить большую скорость, в то время как путь с большими уровнями грунта приводит к уменьшению скорости. Ещё одно свойство природных капель воды – выбор лёгкого пути.

Основные принципы:

– капля воды предпочитает путь с меньшим количеством почвы, чем пути с большим количеством почвы;

– капля воды предпочитает более лёгкий путь, когда приходится выбирать между несколькими маршрутами, которые существуют на пути от источника к месту назначения;

– лёгкость или твёрдость пути определяется количеством почвы на этом пути. Путь с большим уровнем почвы считается трудным путем, тогда как путь с меньшим уровнем почвы считается лёгкий путем.

Алгоритм интеллектуального движения капель воды, формируется следующим образом. Каждая капля воды (IWD) обладает двумя важными свойствами:

– находящийся в ней грунт обозначают уровнем грунта (МЗ);

– скорость, которой она обладает, называется скоростью (МЗ).

Для каждого IWD, значение и свойства почвы и скорости может смениться другими IWD в его окружении. С инженерной точки зрения, среда представляет собой проблему, которая должна быть решена. Река с потоком (роем) IWDs ищет оптимальный путь для данной проблемы.

Каждый IWD предполагает движение от источника к месту назначения. В окружающей среде существует множество путей от данного источника к месту назначения. Место назначения может быть неизвестно. Если местонахождение искомого назначения известно, то для решения проблемы необходимо найти лучшие (часто кратчайшие) пути каждой капли от источника к месту назначения.

IWD алгоритм использует ряд IWDs, чтобы найти оптимальное решение данной проблемы, представляющей собой граф (N, E) с набором узлов N и множеством рёбер E. Этот граф является средой для IWDs и их потока по рёбрам графа.

Каждый IWD начинает строительство своего решения постепенно, путешествуя между узлами графа пока, наконец, не завершает её решения. Одна итерация IWD алгоритма завершается, когда все IWDs завершили свой проход по рёбрам графа. После каждой итерации, находится лучшее решение T – это лучшее решение на основе функции качества среди всех решений, полученных IWDs в текущей итерации. T используется для выполнения следующих итераций. Лучшее решение T – это лучшее решение с начала работы IWD алгоритма, которое было найдено во всех итерациях.

IWD алгоритм можно применить для решения 4х задач:

– TSP (Задача коммивояжера);

– задача H ферзей;

– MKP (многомерная задача о ранце);

– AMT (автоматический многоуровневый порог).

Рассмотрим более подробно задачу коммивояжёра.

Задача коммивояжёра [7] – одна из самых известных задач комбинаторной оптимизации, заключающаяся в отыскании самого выгодного маршрута, проходящего через указанные города хотя бы по одному разу с последующим возвратом в исходный город. В условиях задачи указываются критерий выгодности маршрута (кратчайший, самый дешёвый, совокупный критерий и тому подобное) и соответствующие матрицы расстояний, стоимости и тому подобного. Как правило, указывается, что маршрут должен проходить через каждый город только один раз – в таком случае выбор осуществляется среди гамильтоновых циклов.

Для возможности применения математического аппарата для решения проблемы, её следует представить в виде математической модели. Проблему коммивояжёра можно представить в виде модели на графе, то есть, используя вершины и рёбра между ними. Таким образом, вершины графа на рисунке 2 соответствуют городам, а рёбра (*i, j*) между вершинами *i* и *j* — пути сообщения между этими городами. Каждому ребру (*i, j*) можно сопоставить критерий выгодности маршрута $c_{ij} \geq 0$, который можно понимать как, например, расстояние между городами, время или стоимость поездки. Маршрутом (также гамильтоновым маршрутом) называется маршрут на таком графе, в который входит по одному разу каждая вершина графа. Задача заключается в отыскании кратчайшего маршрута.

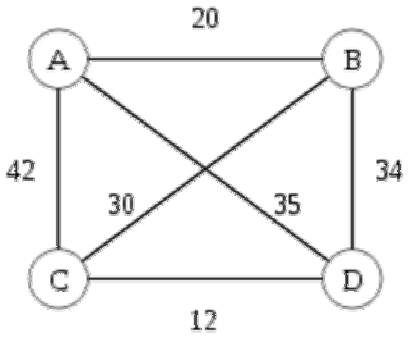

Рисунок 2 – Симметричная задача для четырёх городов

В целях упрощения задачи и гарантии существования маршрута, обычно считается, что модельный граф задачи является полностью связным, то есть, что между произвольной парой вершин существует ребро. В тех случаях, когда между отдельными городами не существует сообщения, этого можно достичь путём ввода рёбер с максимальной длиной. Из-за большой длины такое ребро никогда не попадет к оптимальному маршруту, если он существует.

В зависимости от того, какой критерий выгодности маршрута сопоставляется величине рёбер, различают различные варианты задачи, важнейшими из которых являются симметричная и метрическая задачи.

Одним из подходов к решению задачи является формулировка её в виде задачи дискретной оптимизации, при этом решения представляются в виде переменных, а связи – в виде отношений неравенства между ними. Таким образом, возможно несколько вариантов. Например, симметричную задачу можно представить в виде множества ребер V. Каждому ребру $\{i, j\}$ сопоставляется двоичная переменная $x_{ij} \in \{0,1\}$, равная 1, если ребро принадлежит маршруту, и 0 – в противном случае. Произвольный маршрут можно представить в виде значений множества переменных принадлежности, но не каждое такое множество определяет маршрут. Условием того, что значения множества переменных определяют маршрут, являются описанные далее линейные неравенства.

На рисунке 3 представлен пример графа с маршрутом между двумя точками.

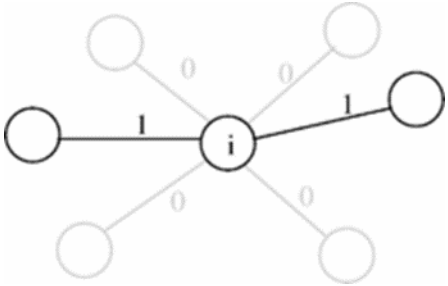

Рисунок 3 – Пример графа с маршрутом между двумя точками

Условие кратности: каждая вершина должна иметь одно входное и одно выходное ребро маршрута.

Каждая вершина должна сообщаться через пару рёбер с остальным вершинам, то есть, через входное и выходное ребро:

$$\forall i \in V, \sum_{j \in V \setminus \{i\}} x_{ij} = 2 . \qquad (7)$$

В сумме каждое слагаемое x_{ij} равно или 1 (принадлежит маршруту) или 0 (не принадлежит). То есть, полученная сумма равна количеству рёбер в маршруте, имеющих вершину i на одном из концов. Она равна 2, так как каждая вершина имеет входное и выходное ребро. В приведённом рисунке 4 вершина показана с входным и выходными ребрами, а рёбра маршрута обозначены толстыми линиями. Рядом с рёбрами указаны длины x_{ij}, прилагаемые к указанной выше сумме.

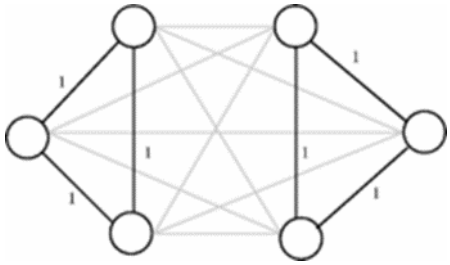

Рисунок 4 – Пример графа с замкнутыми маршрутами

Циклы: переменные удовлетворяют условию кратности, но не определяют маршрут.

Описанные ранее условия кратности выполняются не только маршрутами, но и значениями переменных, соответствующих отдельным циклам, где каждая вершина принадлежит лишь одному циклу (см. рисунок 8). Чтобы избежать подобных случаев, должны выполняться так называемые неравенства циклов (или условия устранения подмаршрутов), которые были определены Данцигом, Фалкерсоном и Джонсоном в 1954 году под названием условия петель (англ. loop conditions). Этими неравенствами определялось дополнительное условие того, что каждое множество вершин $S \subset V$ является либо пустым, либо содержит все вершины, сочетающееся с остальным вершинам через минимум два ребра:

$$\sum_{i \in S, j \notin S} x_{ij} \geq 2 \qquad (8)$$

для всех множеств вершин S, где $1 \leq |S| \leq |V|-1$. Эта сумма равна сумме длин рёбер маршрута между вершиной $i \in S$ и вершиной $j \notin S$. Чтобы устранить лишние неравенства, можно ограничиться множествами вершин S с минимум двумя и максимум $|V|-2$ вершинами. На рисунке 8 рядом рёбра $\{i, j\}$ с длинами $x_{ij}=1$ обозначены толстыми линиями, остальные рёбра имеют длину $x_{ij}=0$. Введение дополнительных условий (8) для множества вершин S, состоящего из трех левых вершин, будет гарантировать, что S сочетается через минимум два ребра маршрута с тремя вершинами справа, чтобы устранить оба цикла. Количество неравенств устранения циклов согласно Данцигу, Фалкерсону и Джонсону равняется $2^n-2(n-1)$.

В 1960 году Миллер, Такер и Землин изобрели альтернативные условия устранения подмаршрутов путём введения n новых переменных, определяющих порядок посещенных городов, требующих только n^2-n+1 дополнительных неравенств. Более того, из-за

двойственности x_{ij} в формулировках Миллера, Такера и Землина задача коммивояжёра остается NP-сложной.

Так, каждый вектор с элементами, равными 0 и 1, удовлетворяющий всем неравенствам, определяет корректный маршрут, который является решением переформулированной задачи коммивояжёра: вычислить

$$\min\left\{\sum_{i\in V}\sum_{j\in V\setminus\{i\}} c_{ij}x_{ij} \mid x \text{ valid } (1)(2), x_{ij}\in\{0,1\}\right\}. \quad (9)$$

Поскольку переменные $x_{i,j}$ имеют значения только 0 и 1, сумма равна общей длине c_{ij} рёбер $\{i, j\}$, принадлежащих маршруту.

Количество неравенств типа (8) растет экспоненциально по мере увеличения количества городов, поскольку почти каждое из $2^{|V|}$ подмножеств узлов определяет одно неравенство. Эту проблему можно решить применением метода отсечения плоскостью, благодаря которому неравенства добавляются, только когда эти неравенства действительно необходимы. С геометрической точки зрения, линейные неравенства можно представить как гиперплоскости в пространстве переменных. Множество векторов, удовлетворяющих этим неравенствам, образуют в таком пространстве политоп (многомерный многогранник), или многомерный многоугольник, точная форма определяется длинами c_{ij} и является в основном неизвестной. Однако, можно показать, что условия (7) и (8) определяют грани (фацет) политопа, то есть боковые поверхности политопа с наивысшей размерностью. Поэтому они относятся к самым сильным линейным неравенствам, которые могут описывать маршрут. Также существует много разных граней, определяемых неравенствами, известными лишь в немногих случаях. Хотя (7) и (8) вместе с ограничениями полностью моделируют проблему только для двоичных векторов, эти неравенства могут использоваться в методе ветвей и границ, чтобы отбросить решения методами линейной оптимизации с нецелыми координатами.

3 Стохастический диффузионный поиск

Алгоритм стохастического диффузионного поиска (англ. Stochastic Diffusion Search, SDS) представил в 1989 г. Бишоп (J. Bishop). В исходном виде алгоритм SDS представляет собой алгоритм дискретной оптимизации. Модификация алгоритма для задачи глобальной непрерывной оптимизации была предложена только в 2011 г.[8].

Алгоритм инспирирован игрой «Ресторан». Суть игры состоит в следующем.

Группа делегатов участвует в длительной конференции в незнакомом городе, в котором имеется большое число ресторанов. Делегаты ставят перед собой задачу найти лучший из ресторанов, т. е. такой, который понравится наибольшему числу делегатов. Проверка каждого ресторана и его ассортимента каждым делегатом займет слишком много времени. Поэтому делегаты используют следующую стратегию поиска. Каждый делегат формирует свою гипотезу относительно того, какой ресторан является лучшим в городе. Вечером он проверяет свою гипотезу, ужиная в этом ресторане и заказывая одно случайное блюдо из меню. На следующее утро те делегаты, которые не были довольны своим ужином, просят поделиться своими впечатлениями случайного коллегу. Если его впечатления положительны, то недовольный делегат принимает выбор этого коллеги. В противном случае, он выбирает новый случайный ресторан.

В терминах алгоритма SDS схема рассмотренной стратегии поиска лучшего ресторана имеет следующий вид:

1) на этапе инициализации алгоритма каждый из агентов высказывает начальную гипотезу (hypothesis) о лучшем ресторане;

2) каждый агент тестирует (test) свою гипотезу, т. е., обедая в выбранном ресторане, подтверждает или опровергает своё первоначальное мнение;

3) путём прямых контактов между агентами происходит диффузия (diffusion) или обмен гипотезами и результатами их проверки;

4) на этой основе агенты формируют новые гипотезы, тестируют их и вновь обмениваются гипотезами. И так далее до выполнения условий сходимости (convergence, halt) итераций.

Рассмотрим задачу глобальной минимизации в параллелепипеде Π.

Тестирование. Приспособленность каждого из агентов S_i, $i\in[1:|S|]$ сравниваем с приспособленностью агента S_j, $j\in[1:|S|]$, $j\neq i$. Если $\varphi_i\leq\varphi_j$, то агента S_i объявляем активным агентом (active agent), а в противном случае – неактивным агентом (inactive agent).

Диффузия. Для каждого из агентов S_i, $i\in[1:|S|]$, выполняем следующие действия:

1) агент S_i неактивен. Выбираем случайного агента S_j. $j\in[1:|S|]$, $j\neq i$. Если этот агент является активным, то агент S_i присваивает гипотезу из некоторой окрестности агента S_j, т. е. полагаем:

$X_i=d(X_j)$, $\varphi_i=\varphi(X_i)$,

где окрестность $d(X_j)$ равна:

$$d(X_j)=\{X\mid x_j^l\leq x_k\leq x_j^u,\ k\in[1:|X|]\};$$
$$x_j^l=max((x_j-0{,}5b(x_j^+-x_j^-)),\ x_j^-),$$
$$x_j^u=max((x_j+0{,}5b(x_j^+-x_j^-)),\ x_j^+). \quad (10)$$

Значение свободного параметра h лежит в интервале (0; 1) и обычно равно 0,1;

2) агент S_i активен. Выбираем для этого агента новую гипотезу. Выбор может проводиться случайно или детерминировано. Рекомендуют выбор гипотезы в центре наибольшей из текущих неисследованных областей множества Π по следующей схеме:

– сортируем текущие координаты всех агентов популяции по каждому из измерений x_k;

– находим наибольший интервал между двумя соседними точками по каждому из измерений x_k; пусть координаты этих точек равны: $x_k^{min}<x_k^{max}$, $k\in[1:|X|]$,

– помещаем агента S_i в точку X_i с координатами:

$x_{i,k}=x_k^{min}+0{,}5(x_k^{max}-x_k^{min})$, $k\in[1:|X|]$. (11)

Сходимость. Алгоритм SDS может использовать различные условия окончания итераций и, в частности, так называемые, сильный и слабый критерии.

4 Оптимизация передвижения бактерий

К настоящему времени разработано несколько алгоритмов оптимизации, объединяемых общих названием бактериальная оптимизация (англ. bacterial optimization). Принято считать, что метод бактериальной оптимизации предложен К. М. Пассино в 2002 г. [8].

Подвижные бактерии, такие как кишечная палочка или сальмонелла, продвигают себя с помощью вращающихся жгутиков [9]. Чтобы двигаться вперед, все жгутики вращаются в одном направлении. При этом бактерия совершает движение, которое называют плавание (swims). При вращении жгутиков в разные стороны бактерия разворачивается – совершает движение кувырок (tumble).

Поведение бактерий обусловлено механизмом, который называется бактериальным хемотаксисом (bacterial chemotaxis) и представляет собой двигательную реакцию микроорганизмов на химический раздражитель. Данный механизм позволяет бактерии двигаться по направлениям к аттрактантам (чаще всего, питательным веществам) и от репеллентов (потенциально вредных для бактерии веществ).

Если выбранное бактерией направление движения соответствует увеличению концентрации аттрактанта (снижению концентрации репеллента), то время до следующего кувырка увеличивается. В силу малого размера бактерии сильное влияние на её перемещения оказывает броуновское движение. В результате бактерия только в среднем движется в направлениях к полезным веществам и от вредных веществ.

В контексте задачи поисковой оптимизации бактериальный хемотаксис можно интерпретировать как механизм оптимизации использования бактерией известных пищевых ресурсов и поиска новых, потенциально более ценных областей.

Канонический алгоритм бактериальной оптимизации

Рассмотрим задачу многомерной глобальной безусловной максимизации.

Канонический алгоритм бактериальной оптимизации (англ. Bacterial Foraging Optimization, BFO) основан на использовании трёх следующих основных механизмов: хемотаксис, репродукция, ликвидация и рассеивание [9].

Пусть $X_{i,r,l}$-(|X|×1) – вектор текущего положения бактерии S_i S на итерации t (на t-м шаге хемотаксиса), r-м шаге репродукции и l-м шаге ликвидации и рассеивания. Здесь $i \in [1:|S|], t \in [1:\hat{t}], r \in [1:\hat{t}^r], l \in [1:\hat{t}^l]$,

где |S| – чётное число агентов в колонии бактерий S;

$\hat{t}$, $\hat{t}^r$, $\hat{t}^l$ – общие числа итераций (шагов хемотаксиса), шагов репродукции, а также шагов ликвидации и рассеивания. Соответствующее значение фитнесс-функции обозначим $\varphi_{i,r,l}$.

Хемотаксис. Процедура хемотаксиса реализует в алгоритме BFO локальную оптимизацию. Следующее положение $X'_{i,r,l}$ бактерии S_j определяет формула:

$$X'_{i,r,l} = X_{i,r,l} + \lambda_i \frac{V_i}{\|V_i\|_E}, \quad (12)$$

где V_i – текущий направляющий (|X| x 1) – вектор шага хемотаксиса бактерии S_j;

λ_i – текущее значение этого шага. При плавании бактерии на следующей итерации вектор V_i остается неизменным, т. е. имеет место равенство $V'_i = V_i$. При кувырке бактерии вектор V'_i представляет собой случайный вектор, компоненты которого имеют значения в интервале [-1; 1]. Другими словами, при кувырке имеет место равенство $V'_i=U_{|X|}$ (-1;1). Плавание каждой из бактерий продолжается до тех пор, пока значения фитнесс-функции увеличиваются.

Значение шага хемотаксиса в формуле (12) может меняться в процессе поиска, уменьшаясь по некоторому закону с ростом числа итераций t.

Репродукция (reproduction). Механизм репродукции имеет своей целью ускорение сходимости алгоритма (за счёт сужения области поиска). Назовём текущим состоянием здоровья (health status) h_i бактерии s_i сумму значений фитнесс-функции во всех точках её траектории от первой до текущей итерации:

$$h_i = \sum_{\tau=1}^{\hat{t}} \varphi_{i,r,l}(\tau) \quad (13)$$

Вычислим значения h_i, i [1:|S|], отсортируем все бактерии в порядке убывания состояний их здоровья и представим результат сортировки в виде линейного списка. Механизм репродукции состоит в том, что на (r+1)-м шаге репродукции вторая половина агентов (наиболее слабых) исключается из указанного списка (погибает), а каждый из агентов первой (выжившей) половины списка расщепляется на два одинаковых агента с одинаковыми координатами, равными координатам расщеплённого агента.

Пусть, например, Sj, j = [1:|S|], – один из выживших агентов, положение которого определяет вектор X_(i,r,l). После репродукции этого агента получаем агентов, $s_j, s_k, k = \frac{|S|}{2} + j,$, положения которых равны

$$X'_{j,(r+1),l}=Xi,r,l,X'_{k,(r+1),l} =X_{j,r,l} \quad (14)$$

В результате выполнения процедуры репродукции результирующее число бактерий в популяции остаётся низменным и равным |S|.

Ликвидация и рассеивание (elimination and dispersal). Рассмотренных процедур хемотаксиса и репродукции в общем случае недостаточно для отыскания глобального максимума многоэкстремальной целевой функции, поскольку эти процедуры не позволяют бактериям покидать найденные ими локальные максимумы этой функции. Процедура ликвидации и рассеивания призвана преодолеть этот недостаток.

Механизм ликвидации и рассеивания включается после выполнения определённого числа процедур репродукции и состоит в следующем. С заданной вероятностью ξ_e случайным образом выбираем n бактерий S_{i1}, S_{i2}....S_{in} и уничтожаем их. Вместо каждой из уничтоженных бактерий в случайно выбранной точке пространства поиска создаём нового агента с тем же номером. В результате выполнения операции

ликвидации и рассевания число бактерий в колонии также остаётся постоянным и равным |S|.

Более строго процедуру ликвидации и рассеивания определяет следующая последовательность шагов:

– полагаем $j = 1$;

– генерируем натуральное случайное число $i_j=U_l(1:|S|)$ и вещественное случайное число $u_j=U_l(0;1)$;

– если $u_j>\xi_e$, то новые координаты бактерии S_j определяем по формуле:

$$x_{i_j,\tau,l,k} = U_l(x_k^-; x_k^+), k \in [1:|X|], \quad (15)$$

где x_k^-, x_k^+ – константы, определяющие диапазон возможных начальных координат бактерий;

– полагаем j = j + 1 и продолжаем итерационный процесс до тех пор, пока не будет уничтожено n бактерий.

Определение векторов начальных положений бактерий $x_{j,0,0}(0), i \in [1:|S|]$ выполняется по формуле, аналогичной формуле (15).

Кооперативная бактериальная оптимизация

Предполагает разделение процедуры оптимизации колонией бактерий на несколько этапов, каждый из которых занимает некоторую часть общего числа поколений и характеризуется различными значениями параметра λ. Известны два варианта кооперативного бактериального алгоритма оптимизации (Cooperative Bacterial Foraging Optimization, CBFO) – алгоритмы CBFO-S и CBFO-H [10]. Рассмотрим вначале первый из указанных алгоритмов.

Последовательный алгоритм бактериальной оптимизации (CBFO-S) представляет собой пример гибридизации по схеме препроцессор/постпроцессор [11]. Алгоритм использует несколько алгоритмов BFO, отличающихся значениями параметра λ и выполняющихся последовательно друг за другом. Выход предыдущего алгоритма BFO (лучшие позиции, найденные каждой из бактерий) поступают на вход следующего алгоритма BFO. На начальном этапе используется алгоритм BFO с большим значением параметра λ, обеспечивающим поиск по всему пространству поиска. В процессе поиска каждая из бактерий сохраняет в своей памяти координаты посещённых точек, а также соответствующие значения фитнесс-функции. Позиция с наибольшим значением фитнесс-функции объявляется многообещающим участком (promising regions).

При переходе на следующий этап (к выполнению следующего алгоритма BFO) текущее значение параметра λ по некоторому правилу уменьшается и начинается поиск из точек, принадлежащих найденным на пре¬дыдущем этапе многообещающим позициям. Поиск с данным значением λ продолжается до выполнения условия перехода к следующему этапу. Затем процесс повторяется ещё меньшим значением параметра λ и так далее до выполнения условия окончания итераций.

Число этапов алгоритма n_P < t является свободным параметром. Числа шагов хемотаксиса в пределах каждого из этапов обычно полагают одинаковыми. Новое значение шага λ определяют, например, по правилу:

$$\lambda' = \nu\lambda, \quad (16)$$

где $\nu \in (0;1)$ – свободный коэффициент уменьшения шага алгоритма.

Развитием алгоритма CBFO-S можно считать адаптивный алгоритм бактериальной оптимизации (Adaptive BacteriaI Foraging Optimization. ABFO), в котором шаг λ меняется, вообще говоря, на каждой итерации по правилу:

$$\lambda' = \lambda(t+1) = \sum_{\tau=0}^{m} b_\tau \lambda(t-\tau), \quad (17)$$

где m – число предыдущих учитываемых шагов, b_τ – весовые коэффициенты, назначаемые лицом, принимающим решения.

Гибридный алгоритм бактериальной оптимизации (CBFO-H) [12] являет собой пример низкоуровневой гибридизации вложением [13, 14]. Алгоритм является многопопуляционным и выполняется в два этапа. На первом этане используем алгоритм BFO с большим значением параметра λ, который после заданного числа итераций обнаруживает аналогично алгоритму BFO-S многообещающие участки и передаёт их на следующий этап.

Основной особенностью второго этапа алгоритма BFO-H является разложение пространства поиска $R^{|X|}$ на $\frac{|X|}{2}$ подпространств, каждое из которых имеет размерность равную двум. Поиск в каждом из указанных двумерных подпространств выполняет своя колония бактерий, совокупность которых образует мегаколонию $S=\{S_i,\ i \in [1:|X|/2]\}$, где $\frac{|X|}{2} = |S|$ – число колоний в мегаколонии.

Алгоритм, использующий эффект роения бактерий

Эффект роения бактерий наблюдается для нескольких разновидностей бактерий, включая бактерию кишечной палочки и бактерию сальмонеллы (лат. Salmonella), в полутвердой питательной среде. Эффект обусловлен локальным уменьшением концентрации аттрактанта вследствие его потребления бактериями. В результате возникает градиент этой концентрации и в хаотическом перемещении бактерий появляется составляющая, направленная по градиенту. Кроме того, бактерии продолжают размножаться. Можно считать, что данный эффект объясняется наличием неявного канала связи между бактериями. В рассматриваемом алгоритме этот канал формализуют следующим образом.

Введем в рассмотрение величины d_{att}, w_{att} определяющие размеры области наличия аттрактанта, а также аналогичные величины d_{rep}, w_{rep}, характеризующие размеры области репеллентов (имеются в виду двумерная модель популяции бактерий). Указанную выше связь между бактериями задаем функцией [15].

$$\tilde{\varphi}(X) = \sum_{i=1}^{|S|} \left[-d_{att} \exp\left(-w_{att} \left\|X - X_i\right\|_E^2\right) + d_{rep} \exp\left(-w_{rep} \left\|X - X_i\right\|_E^2\right) \right]. \quad (18)$$

Легко видеть, что алгебраическое значение первого слагаемого в формуле (18) возрастает с увеличением евклидова расстояния от точки X до точки X_i. Наоборот, второе слагаемое в этой формуле убывает с ростом указанного расстояния. Таким образом, первая сумма в формуле (18) формализует «притяжение» бактерий областью, в которой имеются запасы аттрактанта (а значит, и сосредоточены агенты популяции). Аналогично вторая сумма в этой формуле формализует «отталкивание» бактерий этой областью. Отметим, что значение функции $\tilde{\varphi}(X)$ не зависит от значения целевой функции в точке X.

В качестве фитнесс-функции используется функция:

$$\varphi(X) := \varphi(X) + \tilde{\varphi}(X). \quad (19)$$

Варьируя значения параметров d_{att}, w_{att}, d_{rep}, w_{rep} можно управлять поведением колонии бактерий. Так, уменьшение параметра d_{att}, и увеличение параметра w_{rep}, а также увеличение d_{rep}, и уменьшение w_{rep}, увеличивает модифицированную фитнесс-функцию (19) и диверсифицирует поиск. Противоположные изменения указанных величин локализуют популяцию в небольших областях, обеспечивая высокую точность поиска.

Гибридные бактериальные алгоритмы

Известно большое число алгоритмов оптимизации, построенных на основе гибридизации бактериального алгоритма с другими популяционными алгоритмами. Рассмотрим в качестве примеров низкоуровневую гибридизацию [13, 14] бактериального алгоритма с генетическим алгоритмом (алгоритм BFO-GA) и алгоритмом роя частиц (алгоритм BFO-PSO).

Гибридный алгоритм BFO-GA [12] представляет собой расширенный генетическими операторами отбора, скрещивания и мутации алгоритм, использующий роение бактерий. В качестве оператора отбора алгоритм BFO-GA использует алгоритм на основе метода рулетки. Скрещивание реализовано с помощью оператора арифметического кроссовера, а мутация – с помощью неравномерного мутатора Михалевича [16].

Указанные модифицирующие операторы выполняются после завершения процедур хемотаксиса и репродукции бактериального алгоритма перед процедурами ликвидации и рассеивания этого алгоритма.

Основной целью гибридизации в алгоритме BFO-PSO было расширение возможностей агентов бактериального канонического алгоритма средствами обмена информацией между ними, позаимствованными в алгоритме PSO [17]. Таким образом, алгоритм BFO-PSO моделирует процессы хемотаксиса, репродукции, ликвидации и рассеивания колонии бактерий, а также процесс роения агентов.

Заключение

Построение онтологической модели бионических технологий в соответствии с представленным примером позволит создать адаптируемое и дополняемое описание предметной области, при этом глубина проработки отдельных аспектов будет определяться практической необходимостью. Представленный пример сведений для описания веток (листьев) дерева онтологической модели позволит систематизировать понятия рассматриваемой предметной области. При этом следует отметить, что при формировании базы знаний интеллектуальной системы информационной поддержки процессов создания и развития перспективных бионических технологий [18] может использоваться меньший объём сведений по сравнению с предложенным в данном примере.

БИБЛИОГРАФИЯ

[1] Сигов А.С., Нечаев В.В., Кошкарев М.И. Архитектура предметно-ориентированной базы знаний интеллектуальной системы. International Journal of Open Information Technologies ISSN: 2307-8162 vol. 2, no.12, 2014.

[2] Sigov A.S., Nechaev V.V., Baranyuk V.V., Koshkarev M.I., Smirnova O.S., Melikhov A.A., Bogoradnikova A.V. Architecture of domain-specific data warehouse for bionic information resources. Ecology, environment and conservation, №4, 2015.

[3] Баранюк В.В., Смирнова О.С. Роевой интеллект как одна из частей онтологической модели бионических технологий. International Journal of Open Information Technologies. Vol.3, no. 12, 2015.

[4] Баранюк В.В., Смирнова О.С. Детализация онтологической модели по роевым алгоритмам, основанным на поведении насекомых и животных. International Journal of Open Information Technologies. Vol. 3, no. 12, 2015.

[5] Гравитационный поиск. Эл. ресурс: http://www.pvsm.ru/algoritmy/44008/print/

[6] Shah-Hosseini H. The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. // Int. J. Bio-Inspired Comput. 1, 1/2 (January 2009). Geneva: Inderscience Publishers. – 2009. – с.71-79.

[7] Задача коммивояжёра. Эл. Ресурс: http://dic.academic.ru/dic.nsf/ruwiki/25185

[8] Карпенко А.П. Популяционные алгоритмы глобальной поисковой оптимизации. Обзор новых и малоизвестных алгоритмов. Журнал «Информационные технологии», Приложение, №7/2012 г. Изд.: «Новые технологии», 32 с.

[9] Das S., Biswas A., Dasgupta S., Abraham A. Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications// Foundations of Computational Intelligence. Publisher: Springer. 2009. V. 203. P. 23–55.

[10] Chen H., Zhu Yu., Hu K. Cooperative Bacterial Foraging Optimization // Discrete Dynamics in Nature and Society. 2009. V. 2009. P. 1–17.

[11] Eiben A. E., Michalewicz Z., Schoenauer M, Smith J. E. Parameter Control in Evolutionary Algorithms // Parameter Setting in Evolutionary Algorithms. Springer Verlag. 2007. P. 19–46.

[12] Kim D. H., Abraham A., Cho J. H. A hybrid genetic algorithm and bacterial foraging approach for global optimization // Information Sciences. 2007. V. 177. P. 3918–3937.

[13] El-Abd, Kamel M. A taxonomy of cooperative search algorithms // Hybrid Metaheuristics: Second International Workshop. Springer. 2005. V. 3636. P. 32–41.

[14] Raidl G. R. A Unified View on Hybrid Metaheuristics // Lecture Notes in Computer Science. Springer-Verlag. 2006. V. 4030. P. 1–12.

[15] Datta T., Misra I. S., Mangaraj B.B., Imtiaj S. Improved adaptive bacteria foraging algorithm in optimization of antenna array for faster convergence // Progress In Electromagnetics Research. 2008. V. 1. P. 143–157.

[16] Курейчик В. М. Генетические алгоритмы и их применение. Таганрог: Изд-во Таганрогского РТУ. 2002. С. 244.

[17] Korani W. M, Dorrah H. T, Emara H. M. Bacterial Foraging Oriented by Particle Swarm Optimization Strategy for PID Tuning// Computational Intelligence in Robotics and Automation (CIRA). IEEE International Symposium on 15–18 Dec. 2009. P. 445–450.

[18] Баранюк В.В., Смирнова О.С., Богорадникова А.В. Интеллектуальная система информационной поддержки развития перспективных бионических технологий: основные направления работ по созданию. International Journal of Open Information Technologies ISSN: 2307-8162 vol. 2, no. 12, 2014.

Describing the swarm algorithms, inspired by abiocen and bacterias, in the bionics ontology

O.S. Smirnova, A.V. Bogoradnikova, M.U. Blinov

Abstract – **The paper provides a detailed review of swarm algorithms, inspired by abiocen and bacterias, to be used in the population of bionics ontology.**

Keywords – **swarm algorithms; bionics; ontological model; gravitational search algorithm; intelligent water drops algorithms; stochastic diffusive search; bacterial forage optimisation.**