Оглавление

Введение

В настоящее время большое распространение во всем мире получают программные средства, основанные на технологии и методах искусственного интеллекта. Экспертные системы, нейронные сети и другие программные средства значительно увеличивают число практически значимых задач, которые можно моделировать и решать на компьютере.

В последнее время возрос интерес к решению задач математического программирования с помощью методов интеллектуальной оптимизации, все более востребованным становится решение задач возможностного программирования, коэффициенты которых являются нечеткими числами. В работе [4] была исследована одна из моделей задачи возможностного программирования (модель уровневой оптимизации при ограничении по возможности) и была доказан теорема, позволяющая построить для нее эквивалентный детерминированный аналог. Было установлено, что в случае, когда параметры задачи не являются минисвязанными, получаемый детерминированный эквивалент является задачей негладкой и невыпуклой. В [2] был описан подход числового решения эквивалентного детерминированного аналога с помощью генетических алгоритмов. В настоящей работе продолжаются исследования, начатые в этой области, и проводится сравнительный анализ эффективности ряда методов интеллектуальной оптимизации для решения полученного детерминированного эквивалента.

Таким образом, цель данной работы: провести качественный анализ эффективности работы методов интеллектуальной оптимизации для решения эквивалентного детерминированного аналога задачи возможностного программирования (модель уровневой оптимизации при ограничении по возможности).

Для выполнения поставленных целей необходимо решить следующие задачи:

изучить математический аппарат, необходимый для решения задач классического и возможностного математического программирования;

изучить метод оптимизации на основе генетических алгоритмов;

изучить методы интеллектуальной оптимизации;

построить выборку для проведения сравнительного анализа: совокупности параметров для 3-х, 4-х и 5-ти мерной модели соответствующей задачи возможностной оптимизации – удовлетворяющую условию невырожденности области определения детерминированного эквивалента; для каждой задачи получить эквивалентный детерминированный аналог;

разработать программный комплекс, реализующий выбранные методы интеллектуальной оптимизации;

провести сравнительный анализ эффективности работы выбранных методов и сделать соответствующие выводы.

Искусственный интеллект

Искусственный интеллект – наука и технология создания интеллектуальных машин, особенно интеллектуальных компьютерных программ. Он связан со сходной задачей использования компьютеров для понимания человеческого интеллекта, но не обязательно ограничивается биологически правдоподобными методами. [16]

Одно из частных определений интеллекта, общее для человека и «машины», можно сформулировать так: «Интеллект – способность системы создавать в ходе самообучения программы для решения задач определённого класса сложности и решать эти задачи». [16]

Существует несколько направлений среди моделей и методов исследования искусственного интеллекта:

моделирование рассуждений: необходимо создать символьную систему, которой будет дана некая задача, а в результате ее работы требуется получить решение этой задачи;

обработка естественного языка: анализ того, как можно понять, обработать и сгенерировать текст на человеческом языке;

машинное обучение: в процессе работы интеллектуальная система сама получает некоторые знания;

 генетический подход: пусть есть несколько алгоритмов, которые мы назовем «родители». Тогда если взять у этих алгоритмов наилучшие характеристики и присвоить их какому-то новому алгоритму («потомку»), то этот «потомок» будет более эффективным.

 роевой интеллект.

Роевой интеллект

Общее описание

Роевой интеллект описывает коллективное поведение децентрализованной самоорганизующейся системы. Рассматривается в теории искусственного интеллекта как метод оптимизации. [16]

Системы роевого интеллекта – это множество агентов, которые взаимодействуют между собой и с окружающей средой. При локальном взаимодействии агенты, которые поодиночке довольно просты, создают так называемый роевой интеллект. Примером могут быть рой пчел, птиц, колония муравьев, стая рыб и т.д.

Насекомые, живущие колониями, выполняют свою работу совместными усилиями. При этом в отличие от людей, к примеру, у них нет никакого руководителя. Групповой интеллект животных часто превосходит умственные способности одной особи. Например, рой пчел может решать сложнейшие задачи, немыслимые для единичной особи: находить путь к источникам пищи, защищать свою территорию и т.д. То есть если отдельно взятая пчела – довольно примитивное создание, то целый рой пчел способен эффективно решать свои задачи. Все это становится возможным благодаря роевому интеллекту. Таким образом, можно сказать, что рой (или рой пчел, муравейник, стая рыб и т.д. в каждом из случаев) – это отдельная особь со своими целями, стремлениями, проблемами, т.е. это – единый интеллект.

Разгадка данного природного феномена может принести большую пользу людям различных профессий, а также науке и технике. Создающиеся компьютерные модели, описывающие поведение живой природы, уже применяются для создания различных математических алгоритмов, позволяющих решать сложнейшие задачи (составление расписания авиарейсов, оптимальных маршрутов и т.д.).

Примеры алгоритмов роевого интеллекта

Ниже представлены изученные аппроксимационные (**Аппроксимация - научный метод, состоящий в замене одних объектов другими, в том или ином смысле близкими к исходным, но более простыми. Аппроксимация позволяет исследовать числовые характеристики и качественные свойства объекта, сводя задачу к изучению более простых или более удобных объектов (например, таких, характеристики которых легко вычисляются, или свойства которых уже известны). [16]**) алгоритмы роевого интеллекта, различные варианты которых существуют как для задач одно-, так и многокритериальной оптимизации. Алгоритмы, вдохновленные поведением живой природы:

муравьиный алгоритм [16];

пчелиный алгоритм [5];

светлячковый алгоритм [6];

кукушкин поиск [12];

сорняковый алгоритм$ $[12];

обезьяний поиск$ $[12];

тасующий алгоритм прыгающих лягушек [12];

алгоритм летучей мыши [12];

оптимизация передвижением бактерий [8];

метод роя частиц [16].

Алгоритмы, основанные на поведении неживой природы и человеческого общества:

гармонический поиск [12];

алгоритм гравитационного поиска [12];

стохастический диффузионный поиск [9].

Для решения поставленной задачи были выбраны:

муравьиный алгоритм;

метод роя частиц;

пчелиный алгоритм;

светлячковый алгоритм,

т.к. данные алгоритмы основаны на поведении живой природы, сходны по своему строению.

Муравьиный алгоритм

Отдельный муравей не может принять ни единого решения, т.к. он устроен довольно примитивно: все его действия – это элементарные реакции на окружающую среду и других особей. Муравей не может самостоятельно принимать решения, делать какие-либо выводы, анализировать.

Однако, муравьи очень успешны как вид: существуя на планете более 100 миллионов лет, они строят огромные муравейники, обеспечивают себя и свои жилища всем необходимы. Если сравнивать достижения целой колонии муравьев и отдельной особи, то первые кажутся просто невообразимыми.

 Добиться таких успехов муравьи могут благодаря тому, что они живут в коллективе – колонии, и, таким образом, формируют роевой интеллект. Эффективность деятельности колонии зависит лишь от качества взаимодействия особей, отдельный муравей не обязательно должен быть «умным».

 Колония муравьев – это самоорганизующаяся единица, где нет доминирующих особей. Координацию действий, дачу и выполнение указаний совершают все муравьи колонии, а не отдельная особь.

 Муравей может знать только об окружающей его локальной обстановке, но не обо всей ситуации в целом. Причем знания он может получать либо сам, либо от своих сородичей, явно или неявно. На неявных взаимодействиях муравьев основаны механизмы поиска кратчайшего пути от муравейника до источника пищи.

 Когда муравей совершает путь до источника пищи и обратно, он оставляет за собой след – дорожку феромонов. Другие муравьи, почувствовав такие следы на земле, будут инстинктивно устремляться к своему сородичу, оставившему данный след. При этом, каждый муравей, проходящий по этому пути, также будет оставлять за собой феромоны, что будет в итоге определять его (пути) привлекательность. Соответственно, чем больше муравьев пройдет по одной дорожке, тем привлекательнее для других будет этот путь. Кроме того, чем короче данный путь, тем меньше времени требуется муравьям, чтобы его пройти, и, соответственно, феромоны, оставленные особями, будут заметнее.

Описанный природный механизм был заимствован для решения задач оптимизации, вследствие чего уже существуют различные муравьиные алгоритмы, основанные на поведении колонии муравьев в природе.

Здесь используются многоагентные системы (**Многоагентная система - система, образованная несколькими взаимодействующими интеллектуальными агентами (разумными сущностями, наблюдающими за окружающей средой и действующими в ней, при этом их поведение рационально в том смысле, что они способны к пониманию и их действия всегда направлены на достижение какой-либо цели). [16]**), агенты которых функционируют по крайне простым правилам. [19] Эти правила довольно эффективны при использовании в решении сложных комбинаторных задач (например, задача коммивояжера (**Задача коммивояжёра - 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. [16]**)).

Далее в разделе под понятием агента будет подразумеваться муравей.

Агенты, как и настоящие муравьи, достаточно просто устроены: каждый шаг работы – это выполнение простого вычисления. Также одному агенту требуется совсем немного памяти, чтобы выполнить свои обязанности.

В памяти каждого муравья хранится список тех узлов, где он уже побывал. Когда агенту нужно сделать следующий шаг, он «помнит» о тех узлах, в которых уже побывал, и не рассматривает их при принятии решения. Таким образом, список уже пройденных узлов на каждом шаге пополняется новым. Перед тем, как муравей снова будет проходить свой путь, т.е. перед новой итерацией алгоритма, список будет опустошен.

 При выборе нового узла агент руководствуется как вышеописанным списком уже пройденных узлов, так и привлекательностью ребер, которые он может пройти. Она зависит как от веса данного ребра (т.е. расстояния между узлами), так и от следов феромонов, которые оставили на нем муравьи, которые прошли его раньше. Вес ребра является константой, а величина, характеризующая след феромона, обновляется при каждой итерации алгоритма. Это происходит из-за того, что след со временем начинает испаряться, а муравьи, которые проходят по данному ребру,наоборот усиливают его.

Описание муравьиного алгоритма:

пусть муравей находится в узле $i$, $S\_{i}$ – множество узлов, доступных для перехода, узел $j\in S\_{i}$. Обозначим вес ребра, соединяющего узлы $i$ и $j$, как $w\_{ij}$, а интенсивность феромона на нем – $t\_{ij}$. Тогда вероятность перехода муравья из $i$ в $j$ будет равна:

$$p\_{ij}=\frac{t\_{ij}^{α}+\frac{1}{w\_{ij}^{β}}}{\sum\_{}^{}l\in S\_{i}(t\_{ij}^{α}+\frac{1}{w\_{ij}^{β}})},$$

где $α$ и $β$ – регулируемые параметры, определяющие важность составляющих (веса ребра и уровня феромонов) при выборе пути (выбор правильного соотношения параметров является предметом исследований, и в общем случае производится на основании опыта). [19]

После того, как муравей успешно проходит маршрут, он оставляет на всех пройденных ребрах след, обратно пропорциональный длине пройденного пути:

$$∆\_{ij}=\left\{\begin{array}{c}\frac{k}{L}, \left(ij\right)\in P,\\0, \left(ij\right)!\in P \end{array}\right.,$$

где $L$ – длина пути, а $k$ – регулируемый параметр. [19]

Кроме этого, следы феромона испаряются, то есть интенсивность феромона на всех ребрах уменьшается на каждой итерации алгоритма.

Таким образом, в конце каждой итерации необходимо обновить значения интенсивностей:

$$t\_{ij}=\left(1-e\right)\*t\_{ij}+∆t\_{ij},$$

где $e $– скорость испарения феромона. [19]

Существуют различные, более усовершенствованные вариации данного алгоритма. Они отличаются тем, что исследование областей около уже найденных наилучших решений проводится более тщательно, а также используется история поиска.

Более полное описание муравьиного алгоритма для решения задачи содержится в работе Карпенко А.П. «Популяционные алгоритмы глобальной поисковой оптимизации». [12]

Метод роя частиц

Примером коллективного поведения животных может служить стая птиц. На самом деле, они летают большими группами, плавно и скоординировано. Очень редко происходят случаи столкновения птиц друг с другом во время полета. Наверное, каждый в своей жизни хоть раз видел летящий на юг клин журавлей, и стоит заметить, что у этого клина нет постоянного вожака, птицы управляют стаей попеременно. Или же кормушка, повешенная во дворе – через некоторое время о ней будут знать все птицы в округе.

 Cтая представляет собой роевой интеллект. Птицы ведут себя по определенным правилам, взаимодействуя друг с другом. Каждая особь наблюдает за своими сородичами, определяет их положение и в соответствии с этим координирует свое движение. Если птица находит какой-либо источник пищи, она сообщает об этом всей стае или отдельным особям, которые также передают эту информацию другим.

Из-за того, что источники пищи обычно располагаются случайным образом, отдельная птица может в течение долгого времени не найти ни один из них и погибнуть. Шансы каждой особи выжить повысятся, если все птицы будут делиться своей информацией о найденной пище с другими сородичами. Поэтому такая стратегия поведения является эффективной для выживания стаи и всего вида, при этом будучи неэффективной для отдельно взятой птицы.

На основе такого поведения птиц был создан алгоритм, названный алгоритмом роя частиц. Он моделирует многоагентную систему, которая представляет собой стаю. Каждый агент здесь – это частица (или птица), которая обменивается своей информацией и знаниями с соседями и двигается к оптимальному решению.

 У каждой частицы есть две характеристики – координаты в пространстве решений и вектор скорости перемещения. При инициализации эти параметры задаются случайным образом. Помимо этого, для каждой частицы хранятся координаты лучшего из найденных ею решений, а также лучшее из пройденных всеми частицами решений – с помощью этого происходит имитация обмена информацией между птицами.

Описание метода роя частиц:

пусть $f$ – целевая функция, $S$ – количество частиц в рое, каждой из которых сопоставлена координата $x\_{i}$ в пространстве решений и скорость $v\_{i}$. Пусть $p\_{i}$ – лучшее из известных положений частицы $i$, $g$ – наилучшее известное состояние роя в целом. [19]

Тогда общий вид метода роя частиц:

для каждой частицы $i=1, …, S$:

сгенерировать начальное положение частицы с помощью случайного вектора $x\_{i}\in \left[a, b\right]$;

присвоить лучшему известному положению частицы его начальное значение: $p\_{i}=x\_{i}$;

если $f(p\_{i})<f(g)$ , обновить наилучшее известное состояние роя: $g=p\_{i}$.

присвоить значение скорости частицы: $v\_{i}\in \left[\left(a-b\right),(b-a) \right]$.

пока не выполнен критерий остановки:

 для каждой частицы $i=1, …, S$:

сгенерировать случайные векторы $r\_{p}, r\_{g}\in \left[0, 1\right]$;

обновить скорость частицы: $v\_{i}=ωv\_{i}+φ\_{p}r\_{p}\*\left(p\_{i}-x\_{i}\right)+φ\_{g}r\_{g}\*\left(g-x\_{i}\right)$, где операция \* – покомпонентное умножение;

обновить положение частицы переносом $x\_{i}$ на вектор скорости: $x\_{i}= x\_{i}+v\_{i}$;

если $f(x\_{i})<f(p\_{i})$, то:

обновить лучшее известное положение частицы: $p\_{i}= x\_{i}$;

если $f(p\_{i})<f(g)$ , обновить наилучшее известное состояние роя: $g=p\_{i}$.[19]

 Результат работы, т.е. лучшее из полученных решений, будет содержаться в $g$.

 Параметр $ω$ определяет поведение, параметры $φ\_{p}$, $φ\_{g}$ – эффективность метода, они выбираются произвольно вычислителем.

Более полное описание метода роя частиц для решения задачи содержится в работе Карпенко А.П. «Популяционные алгоритмы глобальной поисковой оптимизации». [12]

Светлячковый алгоритм

В мире насчитывается около двух тысяч видов светлячков, большинство из которых обладают способностью светиться, производя короткие и ритмичные вспышки. Считается, что основной функцией таких вспышек является привлечение особей противоположного пола и потенциальных жертв. Кроме того, сигнальные вспышки могут служить защитным механизмом предупреждения потенциальных хищников о том, что светлячок горек на вкус.

Существуют три правила, которыми описывается поведение светлячков:

светлячки не делятся по половой принадлежности, поэтому каждый светлячок может быть привлекательным для любого другого.

привлекательность светлячков пропорциональна их яркости, и для любых двух светлячков менее яркий привлечет более яркого, однако яркость может уменьшиться, когда расстояние между светлячками увеличивается.

если нет светлячков, более ярких, чем данный, то все остальные будут перемещаться беспорядочно.

Описание светлячкового алгоритма:

целевая функция $f\left(X\right), X=(x\_{1}, ..., x\_{d})$ – результирующая популяция светлячков;

генерируем исходное число светлячков в популяции $X\_{i}, (i=1, 2, …, n)$;

интенсивность света $I\_{i}$ популяции $X\_{i}$ определяется функцией $f(X\_{i})$;

определяем коэффициент поглощения света $γ$;

пока $(t<MaxGeneration)$

по всем $i=1,…,n$

 по всем $j=1,…,n$

если $(I\_{i}<I\_{j})$, то двигаем светлячка $i$ по направлению к светлячку $j$ в -измерении;

вычисляем привлекательность с помощью расстояния $r$ через $exp⁡(-γ\*r)$;

оцениваем новое решение и изменяем интенсивность света;

 классифицируем светляков и ищем лучшего среди всех.

 Здесь $t$ – счетчик, $MaxGeneration$ – размер самой большой популяции.

Обозначим привлекательность светлячка через $β$. Она определяется интенсивностью света, но с увеличением расстояния интенсивность уменьшается, а соответственно, уменьшается и привлекательность. Значит, $β$ зависит от расстояния между светлячками.

Привлекательность светлячка $i$ светлячку $j$ определяется по формуле: $β\left(r\right)= β\_{0}\*e^{-γr^{2}},$ где:

$r$ – расстояние между светлячком $i$ и светлячком $j$;

$β\_{0} $– объективное функциональное значение $I$;

$γ$ – коэффициент поглощения.

 Передвижение светлячка $i$ к более привлекательному светлячку $j$ определяется по формуле: $X\_{i}= X\_{i}+β\_{0}\*e^{-γr^{2}}\left(X\_{j}-X\_{i}\right)+α\left(rand-\frac{1}{2}\right),$ где:

$α$ – произвольный параметр;

$rand$ – случайное число из интервала $\left[0, 1\right]$.

Пчелиный алгоритм

Принцип работы пчелиного алгоритма основан на поведении роя пчел.

Основная цель роя – найти на поле ту область, где плотность цветов будет наибольшей. Изначально пчелы не имеют никакого представления о данном поле, они начинают поиск цветов со случайных позиций, при этом у каждой пчелы свой вектор скорости (но он может совпадать с вектором скорости любой другой пчелы). Каждая пчела запоминает те позиции, где она нашла самое большое количество цветов. При этом она может каким-либо образом знать, где другие пчелы нашли области с наибольшей плотностью цветов.

Далее перед пчелой стоит выбор: вернуться туда, где она нашла наибольшее количество цветов, либо лететь к тому месту, где другие пчелы нашли идентичную область. Выбор пчелы в данном случае будет зависеть о того, что окажет большее влияние на ее решение – ее собственное воспоминание или социальный рефлекс.

Во время того, как пчела будет лететь на выбранное ею место, она может найти область с еще более высокой концентрацией цветов, чем было найдено ранее. Это место может быть выделено как новое, найденное пчелой, у которого плотность цветов наибольшая, а также как найденное всем роем.

Однако пчела может случайно пролететь мимо того места, где плотность цветов выше, чем у области, найденной другой особью. Тогда весь рой в итоге будет лететь в направлении этого места, которое добавится к наблюдению каждой пчелы.

Получается, что пчелы постоянно проверяют и контролируют те области, в которых они побывали. Пчела сравнивает каждое новое место с тем, в котором она уже побывала, оценивая концентрацию цветов. В итоге она находит ту область, где концентрация наибольшая, и там заканчивает свое движение. Через некоторое время весь рой пчел сосредотачивается в окрестностях выбранного места. Т.к. возможность найти область с еще большей концентрацией цветов отсутствует, они роятся в том месте, которое было получено в итоге.

Описание пчелиного алгоритма:

На первом шаге алгоритма выбираются параметры, которые нужно оптимизировать, а также определяется допустимый интервал для поиска оптимальных значений. Далее случайным образом происходит распределение пчел по допустимой области, задается вектор движения и скорость. Далее необходимо сделать так, чтобы каждая частица двигалась согласно пространству решений, будто бы она находится в рое.

На каждую пчелу алгоритм действует отдельно: она циклично двигается через весь рой, при этом перемещение происходит на небольшую величину.

Для каждой частицы выполняются шаги:

оценка пригодности для частицы, сравнение с персональной наилучшей позицией (ПНП) и глобальной наилучшей позицией (ГНП). [19] Для вычисления функции пригодности используются координаты частицы в пространстве решений. В результате возвращается значение пригодности для текущей позиции. В том случае, когда полученное значение превышает ПНП, которое соответствует данной частице, или ГНП, соответствующие позиции заменяются текущей.

корректировка скорости частицы. [19] Действия со скоростью частицы – это основополагающий элемент всей оптимизации. Точность понимания уравнения, используемого для определения скорости, является основополагающей для осмысления всего процесса оптимизации в целом.

 Скорость частицы меняется в соответствии с взаимным расположением позиций ПНП и ГНП. Она стремится в направлении этих позиций наибольшей пригодности в соответствии со следующим уравнением:

$v\_{n}^{i+1}=w\*v\_{n}^{i}+c\_{1}\*rand\left(\right)\*\left(p\_{n}-x\_{n}\right)+c\_{2}\*rand\left(\right)\*\left(g\_{n}-x\_{n}\right)$, где:

$v\_{n}^{i}$ – скорость частицы в -м измерении на предыдущем шаге,

$x\_{n}$ – координата частицы в -м измерении,

 $p\_{n}$ – ПНП,

 $g\_{n}$ – ГНП. [19]

Значения рассчитываются для каждого из $n$. Вышеприведенное уравнение показывает, что значение новой скорости есть результат масштабирования старой скорости на $w$. Также к этому определенному направлению прибавляются направления ГНП и ПНП.

В уравнении $c\_{1}$ и $c\_{2}$ – масштабные коэффициенты: $c\_{1}$ определяет степень влияния памяти частицы о ПНП на саму частицу, $c\_{2}$ определяет степень влияния на частицу остальных пчел из роя.

Если увеличивается параметр $c\_{1}$, то исследование пространства решений производится следующим образом: каждая частица движется в направлении своего ПНП. Если увеличивается параметр $c\_{2}$, то исследуется предполагаемый глобальный максимум.

$rand\left(\right)$ – функция, возвращающая случайное число в интервале между -1 и 1. Если в реализации данная функция встречается два и более раз, это означает, что происходит два или более различных вызова этой функции (в модификациях алгоритма количество вызовов может быть разным).

Параметр $ω$ – «инерционный вес». Он определяется в интервале от 0 до 1 и показывает, насколько частица остается верной своему изначальному курсу (это значит, что данный курс не подвергся влиянию ГНП и ПНП).

Таким образом, общее описание алгоритма выглядит так:

инициализируем популяцию со случайными характеристиками;

оцениваем характеристики;

пока не достигнут критерий остановки, формируем новую популяцию;

выбираем места для поиска окрестностей;

отправляем пчел для поиска окрестностей, оцениваем характеристики;

выбираем самую продуктивную пчелу на каждом участке;

отправляем остальных пчел искать окрестности без определенного порядка, оцениваем характеристики. [19]

Более полное описание пчелиного алгоритма для решения задачи содержится в работе Карпенко А.П. «Популяционные алгоритмы глобальной поисковой оптимизации». [12]

Оптимизация с помощью генетических алгоритмов

Генетический алгоритм – это эвристический (**Эвристический алгоритм – алгоритм решения задачи, не имеющий строгого обоснования, но, тем не менее, дающий приемлемое решение задачи в большинстве практически значимых случаев. [16]**) алгоритм поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, напоминающих биологическую эволюцию. [16]

Задача формализуется таким образом, чтобы её решение могло быть закодировано в виде вектора («генотипа (**Генотип – совокупность генов данного организма, характеризующая особь. [16]**)») генов, где каждый ген может быть битом, числом или неким другим объектом. В классических реализациях данного алгоритма предполагается, что генотип имеет фиксированную длину. Однако существуют вариации, свободные от этого ограничения.

Сначала случайным образом (в большинстве случаев) создается множество генотипов начальной популяции, которые в дальнейшем будут оцениваться с использованием функции приспособленности. В результате эта приспособленность соотносится с каждым генотипом (т.е. одному генотипу ставится в соответствие одно значение приспособленности). Значение приспособленности определяет качество решения поставленной задачи описываемым им фенотипом (**Фенотип – совокупность характеристик, присущих индивиду на определённой стадии развития. [16]**).

Результат, т.е. полученное множество решений, будем называть поколением. Тогда, учитывая значение приспособленности, из поколения берутся те решения, к которым применяются генетические операторы (в большинстве случаев «скрещивание» – crossover и «мутация» – mutation). В результате получаются новые решения, для которых снова вычисляется значение приспособленности. Далее происходит отбор (т.е. селекция) лучших решений в для следующего поколения.

Этот набор действий повторяется итеративно, так моделируется «эволюционный процесс», продолжающийся несколько жизненных циклов (поколений), пока не будет выполнен критерий остановки алгоритма. Таким критерием может быть:

нахождение глобального, либо субоптимального решения;

исчерпание числа поколений, отпущенных на эволюцию;

исчерпание времени, отпущенного на эволюцию. [16]

Таким образом, общее описание модели генетического алгоритма выглядит так:

задать целевую функцию для особей популяции;

создать начальную популяцию;

пока не достигнут критерий остановки:

размножение (скрещивание);

мутирование;

вычислить значение целевой функции для всех особей;

формирование нового поколения (селекция). [16]

Сначала случайным образом создаем исходную популяцию. Может случиться так, что полученная популяция будет совершенно неконкурентоспособной, однако есть вероятность того, что за небольшой промежуток времени генетический алгоритм сделает из нее вполне жизнеспособную. Из этого следует вывод, что для первого шага достаточно того, чтобы особи исходной популяции соответствовали формату, а также с их помощью можно было рассчитать функцию приспособленности. В итоге получаем популяцию $H$, состоящую из $N$ особей.

Далее следует этап размножения. Т.к. оно половое (обычно в генетических алгоритмах используется именно половое размножение), для произведения потомка нужны родители (их обычно двое).

Размножение в генетических алгоритмах зависит от представления данных. Однако, необходимо выполнение следующего условия: потомок, или потомки, должны иметь возможность «смешивать» черты обоих родителей, в результате чего они смогут унаследовать их.

Этап мутации чем-то напоминает этап размножения. Для некоторой доли мутантов $m$ (параметр алгоритма) необходимо выбрать $mN$ особей. После этого происходит их изменение согласно операциям мутации, которые были определены заранее.

Этап отбора заключается в следующем. Необходимо выбрать определенную долю из всей популяции так, чтобы она осталась жизнеспособной на данном этапе эволюции. Для проведения отбора существуют различные методы.

Необходимо помнить о зависимости вероятности $h$ того, что особь выживет, и значения функции приспособленности. Долю оставшихся в живых особей $s$ задают заранее, т.к. она является параметром алгоритма.

В результате формируется популяция $H'$, куда войдут $sN$ особей из $N$ особей популяции $H$. При этом все остальные особи в процессе погибают.

Более полное описание генетического алгоритма для решения задачи содержится в работе Солдатенко И.С. «О генетическом алгоритме решения задачи возможностной оптимизации при взаимно Tw-связанных параметрах». [2]

Теоретические сведения

Основные понятия

Для выполнения данной работы было необходимо определить нелинейную негладкую задачу математического программирования.

Математическое программирование – это раздел математики, исследующий математические модели и методы решения многоэкстремальных задач с ограничениями. [16]

В данной работе определяется задача возможностного программирования.

В данном случае под математическим программированием понимается оптимизация – в математике, информатике и исследовании операций задача нахождения экстремума (минимума или максимума) целевой функции (**Целевая функция - функция, связывающая цель (оптимизируемую переменную) с управляемыми переменными в задаче оптимизации. [16]**) в некоторой области конечномерного векторного пространства, ограниченной набором линейных и/или нелинейных равенств и/или неравенств. [16]

Задача называется нелинейной, негладкой, если в ее условии содержатся нелинейные, негладкие функции.

Линейная функция – функция вида $y=kx+b$ (для функций одной переменной). Тогда нелинейная функция (**В ряде случаев этот термин может применяться и к зависимостям**$ y=kx+b$**, где** $b\ne 0$**, то есть к неоднородным линейным функциям, поскольку они не обладают свойством линейности, а именно в этом случае** $f\left(x\_{1}+x\_{2}\right)\ne f\left(x\_{1}\right)+f(x\_{2})$ **и** $f(cx)\ne cf(x)$**. [16]**) – любая произвольная функция, не являющаяся линейной.

Негладкая функция – функция, у которой первая производная не является непрерывной функцией на всей области определения. [18]

Необходимый математический аппарат

Введем необходимые определения из понятия теории возможностей. Пусть множество $Г$ есть модельное множество, $γ\in Г$ – его элементы, $Р(Г)$ – множество всех подмножеств множества $Г$. [3]

Возможностной (нечеткой) величиной называется вещественная функция $A\left(∙\right):Г\rightarrow E^{1}$, возможные значения которой характеризуются ее распределением возможностей $μ\_{А}\left(x\right)$ – возможность того, что $А$ может принять значение $x$:

$μ\_{А}\left(x\right)=π\left\{γ\in Г:А\left(γ\right)=x\right\}, ∀x\in E^{1}$. [3]

Носителем возможностной величины $А$ называется четкое подмножество $supp\left(A\right)=\left\{x | μ\_{А}\left(x\right)>0\right\}, x\in E^{1}$. [3]

Для любой возможностной переменной $А $и любого $α\in \left[0, 1\right]$ $α$-уровневым множеством $A^{α}$ называется:

$$A^{α}=\left\{x\in E^{1} | μ\_{А}\left(x\right)\geq α\right\}, для ∀α\in \left(0, \left.1\right],\right.$$

$$A^{α}=cl\left(supp\left(A\right)\right), для α=0,$$

где $cl\left(supp\left(A\right)\right)$ – замыкание носителя возможностной переменной $А$. [3]

Возможностной мерой $π:Г\rightarrow E^{1}$ называется функция множества, обладающая следующими свойствами:$ π\left\{∅\right\}=0, π\left\{Г\right\}=1;$ $π\left\{\bigcup\_{i\in I}^{}A\_{i}\right\}=sup\_{i\in I}π\left\{A\_{i}\right\}, ∀A\_{i}\in P\left(Г\right), ∀I$. [3]

Тройка $(Г, Р\left(Г\right), π)$ называется возможностным пространством. [3]

Возможностная переменная $А$ называется выпуклой, если ее функция распределения является квазивогнутой:

$$μ\_{А}\left(λx\_{1}+\left(1-λ\right)x\_{2}\right)\geq min\left\{μ\_{А}\left(x\_{1}\right), μ\_{А}\left(x\_{2}\right)\right\}, $$

$λ\in \left[0, 1\right], x\_{1}, x\_{2}\in E^{1}$. [3]

Как правило, возможностные переменные на числовой прямой, характеризующиеся строго унимодальными (**Пусть в интервале** $\left[a,b\right]$ **имеется единственное значение** $x^{\*}$**такое, что** $f\left(x^{\*}\right)=minf(x)$ **на** $\left[a,b\right]$**;** $f(x)$ **строго убывает для** $x\leq x^{\*}$ **и строго возрастает для** $x\geq x^{\*}$**. Такая функция называется унимодальной. [3]**), квазивогнутыми, полунепрерывными сверху функциями распределения и ограниченными носителями, называются нечеткими числами. Если функция распределения возможностей при этом не является строго унимодальной, то возможностная переменная называется нечетким интервалом. [3]

Для моделирования нечетких чисел и нечетких интервалов в данной работе используются распределения $(L, R)$-типа. [3]

Функцией распределения $(L, R)$-типа возможностной величины Х называется функция, определяемая следующим образом:

$$μ\_{X}\left(x\right)=\left\{\begin{array}{c}L\left(\frac{m-x}{d}\right), x\leq m,\\1, m<x<M,\\R\left(\frac{x-M}{D}\right), x\geq M,\end{array}\right.$$

где $m$ и $M$ – границы интервалов толерантности (**Толерантный интервал - случайный интервал, построенный по независимым одинаково распределенным случайным величинам, функция распределения которых неизвестна, и содержащий с заданной вероятностью по крайней мере долю Р (0<P<1) заданной вероятностной меры. [3]**), $d$ и $D$ – левый и правый коэффициент нечеткости (**В данной работе возможностная величина будет иметь триангулярное распределение:** $x\in T\_{r}(a,b)$**. [3]**). [3]

В данной работе определена задача возможностной оптимизации с взаимодействующими параметрами. Зависимость параметров моделируется на основе аппарата $t$ -норм – двуместная функция $T\left(∙, ∙\right) : \left[0, 1\right]×\left[0, 1\right]\rightarrow \left[0, 1\right]$. Используется слабая $t$-норма $T\_{w}$:

$$T\_{w}=\left\{\begin{array}{c}min\left\{x, y\right\}, если max\left\{x, y\right\}=1, \\0, иначе. \end{array}\right.$$

[3]

Пусть даны возможностное пространство $(Г, Р\left(Г\right), π)$ и $t$-норма $T$. Множества $X\_{1}, …, X\_{n}\in , Р\left(Г\right)$ называются взаимно $T$-связанными, если для любого подмножества $\left\{i\_{1}, …,i\_{k}\right\}$ множества $\left\{1, …, n\right\}$, $1\leq k\leq n$:

$π\left\{X\_{i\_{1}}∩…∩X\_{i\_{k}}\right\}=T\left(π\left\{X\_{i\_{1}}\right\}, …, π\left\{X\_{i\_{k}}\right\}\right),$ где

$T\left(π\left\{X\_{i\_{1}}\right\}, …, π\left\{X\_{i\_{k}}\right\}\right)=T(T(…T(T\left(π\left\{X\_{i\_{1}}\right\}, π\left\{X\_{i\_{2}}\right\}\right), π\left\{X\_{i\_{3}}\right\}),…,π\left\{X\_{i\_{k-1}}\right\}),π\left\{X\_{i\_{k}}\right\})$. [3]

Допустим, что $А\_{1}, …, А\_{n}$ – возможностные величины, заданные на возможностном пространстве $(Г, Р\left(Г\right), π)$, тогда они называются взаимно $T$-связанными, если для любого подмножества $\left\{i\_{1}, …,i\_{k}\right\}$ множества $\left\{1, …, n\right\}$, $1\leq k\leq n$:

$μA\_{i\_{1}},…, A\_{i\_{k}}\left(x\_{i\_{1}},…, x\_{i\_{k}}\right)= π\left\{γ\in Г | A\_{i\_{1}}\left(γ\right)= x\_{i\_{1}}, …, A\_{i\_{k}}\left(γ\right)= x\_{i\_{k}}\right\}=π\left\{A\_{i\_{1}}^{-1}\left\{x\_{i\_{1}}\right\}∩…∩A\_{i\_{k}}^{-1}\left\{x\_{i\_{k}}\right\}\right\}=T\left(π\left\{A\_{i\_{1}}^{-1}\left\{x\_{i\_{1}}\right\}∩…∩A\_{i\_{k}}^{-1}\left\{x\_{i\_{k}}\right\}\right\}\right), x\_{ij}\in E^{1}$. [3]

Модель задачи уровневой оптимизации

Определим задачу возможностного программирования (или задачу уровневой оптимизации):

$$k\rightarrow max,$$

$$π\left\{f\_{0}\left(x,γ\right)=k\right\}\geq α\_{0},$$

$$\left\{\begin{array}{c}π\left\{f\_{i}\left(x,γ\right)=0\right\}\geq α\_{i}, i=1,…,m,\\x\in E\_{+}^{n}.\end{array}\right.$$

 Здесь $f\_{0}\left(x,γ\right)=\sum\_{j=1}^{n}a\_{0j}(γ)x\_{j}$, $f\_{i}\left(x,γ\right)=\sum\_{j=1}^{n}a\_{ij}(γ)x\_{j}-b\_{i}(γ)$,$ i=1,…,m$. [4]

$E\_{+}^{n}$ – неотрицательное подмножество -мерного евклидова пространства. [4]

Предполагается, что $a\_{0j}(γ)$, $a\_{ij}(γ)$ и $b\_{i}(γ)$ – взаимно $T\_{w}$-связанные нечеткие величины $(L, R)$-типа:

$$a\_{0j}\left(γ\right)=\left(a\_{0j}^{'}, a\_{0j}^{''},ϑ\_{0j}^{},β\_{0j}^{}\right)\_{LR}, a\_{ij}\left(γ\right)=\left(a\_{ij}^{'}, a\_{ij}^{''},ϑ\_{ij}^{},β\_{ij}^{}\right)\_{LR, }$$

$b\_{i}\left(γ\right)= \left(b\_{i}^{'}, b\_{i}^{''},ϑ\_{i}^{},β\_{i}^{}\right), x\in E\_{+}^{n}$.

Для данного случая взаимно $T\_{w}$-связанных параметров и идентичных функций представления формы, моделирующих распределение нечетких параметров, существует эквивалентный детерминированный аналог (при условии существования обратных функций представления форм $L^{-1}и R^{-1} $). Теорема о построении эквивалентной детерминированной системы, которая имеет следующий вид:

$$f^{''}\left(x\right)\rightarrow max,$$

$$\left\{\begin{array}{c}f^{''}\left(x\right)-f^{'}\left(x\right)\geq 0,\\\sum\_{j=1}^{n}a\_{ij}^{'}x\_{j}-max\_{j=1,…,n}\left\{x\_{j}ϑ\_{ij}\right\}L^{-1}\left(α\_{i}\right)\leq b\_{i}^{''}+β\_{i}R^{-1}\left(α\_{i}\right), i=1,…,m;\\\sum\_{j=1}^{n}a\_{ij}^{''}x\_{j}-max\_{j=1,…,n}\left\{x\_{j}β\_{ij}\right\}R^{-1}\left(α\_{i}\right)\geq b\_{i}^{'}-ϑ\_{i}L^{-1}\left(α\_{i}\right), i=1,…,m;\\x\in X,\end{array}\right.$$

доказана в статье [4].

Числовой эксперимент

Задача возможностной оптимизации и ее детерминированный эквивалент

В п. 4.3 была определена модель задачи уровневой оптимизации, согласно которой определялись задачи для исследования.

Рассмотрим решение задачи с тремя параметрами на следующем примере.

Дана задача уровневой оптимизации с $T\_{w}$-связанными параметрами при $n=m=3$ и параметрах $α\_{0}=0.5, α\_{1}=0.3, α\_{2}=0.4, α\_{3}=0.6$ :

$$k\rightarrow max;$$

$$π\left\{α\_{01}\left(γ\right)x\_{1}+α\_{02}\left(γ\right)x\_{2}+α\_{03}\left(γ\right)x\_{3}=k\right\}\geq 0.5;$$

$$\left\{\begin{array}{c}π\left\{α\_{11}\left(γ\right)x\_{1}+α\_{12}\left(γ\right)x\_{2}+α\_{13}\left(γ\right)x\_{3}-b\_{1}(γ)=k\right\}\geq 0.3;\\π\left\{α\_{21}\left(γ\right)x\_{1}+α\_{22}\left(γ\right)x\_{2}+α\_{23}\left(γ\right)x\_{3}-b\_{2}(γ)=k\right\}\geq 0.4;\\π\left\{α\_{31}\left(γ\right)x\_{1}+α\_{32}\left(γ\right)x\_{2}+α\_{33}\left(γ\right)x\_{3}-b\_{3}(γ)=k\right\}\geq 0.6;\\x\in R\_{+}^{2}.\end{array}\right.$$

Здесь $α\_{ij}\left(γ\right), b\_{j}\left(γ\right)$ – триангулярные взаимно $T\_{w}$-связанные нечеткие величины $(i=0,…,3;j=1,…,3)$:

$$α\_{01}\left(γ\right)=T\_{r}\left(1, 2\right), α\_{02}\left(γ\right)=T\_{r}\left(5,3\right), α\_{03}\left(γ\right)=T\_{r}\left(1,4\right),$$

$$α\_{11}\left(γ\right)=T\_{r}\left(8, 3\right), α\_{12}\left(γ\right)=T\_{r}\left(-2,1\right), α\_{13}\left(γ\right)=T\_{r}\left(8,4\right),$$

$$α\_{21}\left(γ\right)=T\_{r}\left(1, 2\right), α\_{22}\left(γ\right)=T\_{r}\left(-5,4\right), α\_{23}\left(γ\right)=T\_{r}\left(1,-5\right),$$

$$α\_{31}\left(γ\right)=T\_{r}\left(1, 2\right), α\_{32}\left(γ\right)=T\_{r}\left(-2,1\right), α\_{33}\left(γ\right)=T\_{r}\left(3,1\right),$$

$$b\_{1}\left(γ\right)=T\_{r}\left(14,7\right), b\_{2}\left(γ\right)=T\_{r}\left(0,5\right), b\_{3}\left(γ\right)=T\_{r}\left(0,3\right).$$

Представим эти нечеткие величины в $(L, R)$-форме. Для триангулярных нечетких величин функции представления форм будут следующими:

$$L\left(t\right)=R\left(t\right)=max\left\{0,1-t\right\}, t\in R\_{+}^{1}.$$

Получаем величины $(L, R)$-типа:

$$α\_{01}\left(γ\right)=(1,1,1,1)\_{LR}, α\_{02}\left(γ\right)=(5,5,\frac{3}{2},\frac{3}{2})\_{LR}, α\_{03}\left(γ\right)=(1,1,2,2)\_{LR},$$

$$α\_{11}\left(γ\right)=(8,8,\frac{3}{2},\frac{3}{2})\_{LR}, α\_{12}\left(γ\right)=(-2,-2,\frac{1}{2},\frac{1}{2})\_{LR}, α\_{13}\left(γ\right)=(8,8,2,2)\_{LR},$$

$$α\_{21}\left(γ\right)=(1,1,1,1)\_{LR}, α\_{22}\left(γ\right)=(-5,-5,2,2)\_{LR}, α\_{23}\left(γ\right)=(1,1,-\frac{5}{2},-\frac{5}{2})\_{LR},$$

$$α\_{31}\left(γ\right)=(1,1,1,1)\_{LR}, α\_{32}\left(γ\right)=(-2,-2,\frac{1}{2},\frac{1}{2})\_{LR}, α\_{33}\left(γ\right)=(3,3,\frac{1}{2},\frac{1}{2})\_{LR},$$

$$b\_{1}\left(γ\right)=(14,14,\frac{7}{2},\frac{7}{2})\_{LR}, b\_{2}\left(γ\right)=(0,0,\frac{5}{2},\frac{5}{2})\_{LR}, b\_{3}\left(γ\right)=(0,0,\frac{3}{2},\frac{3}{2})\_{LR}.$$

Построим эквивалентный детерминированный аналог для данной задачи:

$$x\_{1}+5x\_{2}+x\_{3}+\frac{5}{2}max\left\{x\_{1},\frac{3}{2}x\_{2},2x\_{3}\right\}\rightarrow max,$$

$$\left\{\begin{array}{c}max\left\{x\_{1},\frac{3}{2}x\_{2},2x\_{3}\right\}\geq 0,\\8x\_{1}-2x\_{2}+8x\_{3}-max\left\{\frac{3}{2}x\_{1},\frac{1}{2}x\_{2},2x\_{3}\right\}\*0.7\leq 16.45,\\8x\_{1}-2x\_{2}+8x\_{3}+max\left\{\frac{3}{2}x\_{1},\frac{1}{2}x\_{2},2x\_{3}\right\}\*0.7\geq 11.55,\\x\_{1}-5x\_{2}+x\_{3}-max\left\{x\_{1},2x\_{2},-\frac{5}{2}x\_{3}\right\}\*0.6\leq 1.5,\\x\_{1}-5x\_{2}+x\_{3}+max\left\{x\_{1},2x\_{2},-\frac{5}{2}x\_{3}\right\}\*0.6\geq -1.5,\\x\_{1}-2x\_{2}+3x\_{3}-max\left\{x\_{1},\frac{1}{2}x\_{2},\frac{1}{2}x\_{3}\right\}\*0.4\leq 0.6,\\x\_{1}-2x\_{2}+3x\_{3}+max\left\{x\_{1},\frac{1}{2}x\_{2},\frac{1}{2}x\_{3}\right\}\*0.4\geq -0.6,\\x\_{1},x\_{2},x\_{3}\geq 0.\end{array}\right.$$

С помощью выбранных алгоритмов получаем решение данной системы: $x\_{1}=2,50807$, $x\_{2}=1,72345$, $x\_{3}=4,12693$, $F\_{w}\left(x\_{1}, x\_{2}, x\_{3}\right)=9,24678$.

Аналогичным образом осуществляется определение задач с четырьмя и пятью параметрами, а также построение эквивалентной детерминированной системы.

Решение задачи с помощью выбранных методов

На следующих графиках представлен ход решения поставленной задачи.

Муравьиный алгоритм:



Рис 1. Решение задачи с применением муравьиного алгоритма

Метод роя частиц:



Рис 2. Решение задачи с применением метода роя частиц

Светлячковый алгоритм:



Рис 3. Решение задачи с применением светлячкового алгоритма

Пчелиный алгоритм:



Рис 4. Решение задачи с применением пчелиного алгоритма

Генетический алгоритм:



Рис 5. Решение задачи с применением генетического алгоритма

По данным графикам можно сделать следующий вывод: оптимальное решение задачи муравьиный алгоритм находит на 85-й итерации, метод роя частиц – на 136-й, светлячковый алгоритм – на 108-й, пчелиный алгоритм – на 146-й, генетический алгоритм – на 86-й.

Таким образом, при проведении первого эксперимента наилучший результат показал муравьиный алгоритм. Однако генетический алгоритм совершил всего лишь на одну итерацию больше. Дольше всех решение искал пчелиный алгоритм.

Как видно на графиках, существуют некоторые монотонные области, т.е. на данных итерациях алгоритм находил одно и то же решение. К примеру, рассмотрим график решения задачи генетическим алгоритмом, на котором видно как минимум две такие области:



Рис 6. Итерации алгоритма с одинаковым значением решения

Для того чтобы определить, найдено ли оптимальное решение задачи, использовался специальный параметр porog\_end (в программной реализации может задаваться произвольно). Данный параметр используется в том случае, если алгоритм несколько итераций подряд находит одно и то же решение задачи. Тогда заданное количество раз (равное данному параметру) алгоритм продолжает выполнять итерации. Если на какой-либо из них будет найдено новое решение – алгоритм продолжает свою работу, иначе считаем, что оптимальное решение найдено, находим значение той итерации, когда решение было найдено впервые.

При проведении исследования параметр porog\_end для каждого эксперимента был одинаковым – 100 итераций.

Анализ результатов

Описание эксперимента

Для анализа применения алгоритмов роевого интеллекта и генетического алгоритма для решения определенной в п. 4.3. модели задачи уровневой оптимизации проводились следующие действия.

Для каждого вида задач (т.е. с тремя, четырьмя и пятью параметрами) генерировались сто примеров. Далее строились их детерминированные аналоги, для которых с помощью выбранных алгоритмов происходил поиск оптимальных решений. Свободные параметры алгоритмов, описанные в п. 2.2.1, 2.2.2, 2.2.3, 2.2.4, 3, для каждого эксперимента задавались одинаково, чтобы избежать погрешности вычисления.

При проведении экспериментов учитывалось процессорное время, затраченное на решение каждой задачи. На основании полученных данных проводился анализ эффективности поиска оптимальных решений методами интеллектуальной оптимизации.

Рассмотрим анализ полученных данных и сравним результаты.

Анализ решения задач с тремя параметрами

Для анализа решения задач с тремя параметрами было проведено 100 экспериментов. На графике представлены результаты исследования:

Рис 7. Результаты проведения экспериментов для случая трехмерных задач

Проведем анализ полученных результатов.

Ось абсцисс описывает номер проводимого эксперимента, ось ординат – время выполнения итерации. Время измеряется в минутах с точностью до секунд. Цена одного деления – 1 минута.

При использовании муравьиного алгоритма среднее время, потраченное на нахождение оптимальных решений, составляет 15,21 мин. При этом наименьшее количество затраченного времени – 12,06 мин (эксперимент № 9), наибольшее – 18,06 мин (эксперимент № 2).

При использовании метода роя частиц среднее время, потраченное на нахождение оптимальных решений, составляет 22,15 мин. При этом наименьшее количество затраченного времени – 19,04 мин (эксперимент № 33), наибольшее – 26,03 мин (эксперимент № 12).

При использовании светлячкового алгоритма среднее время, потраченное на нахождение оптимальных решений, составляет 21,08 мин. При этом наименьшее количество затраченного времени – 17,04 мин (эксперимент № 18), наибольшее – 25,53 мин (эксперимент № 8).

При использовании пчелиного алгоритма среднее время, потраченное на нахождение оптимальных решений, составляет 33,12 мин. При этом наименьшее количество затраченного времени – 29,01 мин (эксперимент № 29), наибольшее – 36,07 мин (эксперимент № 28).

При использовании генетического алгоритма среднее время, потраченное на нахождение оптимальных решений, составляет 11,35 мин. При этом наименьшее количество затраченного времени – 7,49 мин (эксперимент № 5), наибольшее – 15,58 мин (эксперимент № 87).

Таким образом, можно сделать следующие выводы.

Как видно на графике, муравьиный и генетический алгоритм ищут оптимальное решение задачи быстрее остальных методов, однако результаты генетического алгоритма значительно лучше. В некоторых случаях муравьиный алгоритм решает поставленную задачу быстрее, но такие случаи единичные. Сравним среднее время поиска оптимального решения: муравьиному алгоритму необходимо 15,21 мин, генетическому – 11,35 мин. Разница составляет более четырех минут.

Метод роя частиц и светлячковый алгоритм имеют приблизительно равные результаты. Среднее время поиска, необходимое методу роя частиц – 22,15 мин, светлячковому алгоритму – 21,08 мин. Разница составляет примерно полторы минуты.

Пчелиный алгоритм, как видно на графике, ищет оптимальное решение дольше всех.

Таким образом, быстрее всех решение задачи ищет генетический алгоритм. Вторым по скорости будет муравьиный алгоритм, третьим – светлячковый, четвертым – метод роя частиц. Самым медленным оказался пчелиный алгоритм. Для получения результата исходя из средних значений ему нужно практически на 22 минуты больше, чем генетическому алгоритму.

Анализ решения задач с четырьмя параметрами

Для анализа решения задач с четырьмя параметрами было проведено 100 экспериментов. На графике представлены результаты исследования:

Рис 8. Результаты проведения экспериментов для случая четырехмерных задач

Проведем анализ полученных результатов.

Ось абсцисс описывает номер проводимого эксперимента, ось ординат – время выполнения итерации. Время измеряется в минутах с точностью до секунд. Цена одного деления – 1 минута.

При использовании муравьиного алгоритма среднее время, потраченное на нахождение оптимальных решений, составляет 19,01 мин. При этом наименьшее количество затраченного времени – 15,01 мин (эксперимент № 58), наибольшее – 22,27 мин (эксперимент № 29).

При использовании метода роя частиц среднее время, потраченное на нахождение оптимальных решений, составляет 25,35 мин. При этом наименьшее количество затраченного времени – 22,01 мин (эксперимент № 39), наибольшее – 29,12 мин (эксперимент № 48).

При использовании светлячкового алгоритма среднее время, потраченное на нахождение оптимальных решений, составляет 24,16 мин. При этом наименьшее количество затраченного времени – 21,04 мин (эксперимент № 53), наибольшее – 27,24 мин (эксперимент № 7).

При использовании пчелиного алгоритма среднее время, потраченное на нахождение оптимальных решений, составляет 35,46 мин. При этом наименьшее количество затраченного времени – 31,49 мин (эксперимент № 74), наибольшее – 39,11 мин (эксперимент № 10).

При использовании генетического алгоритма среднее время, потраченное на нахождение оптимальных решений, составляет 15,09 мин. При этом наименьшее количество затраченного времени – 12,01 мин (эксперимент № 59), наибольшее – 18,12 мин (эксперимент № 94).

Таким образом, можно сделать следующие выводы.

Как видно на графике, в случае четырех измерений генетический и муравьиный алгоритмы ищут оптимальное решение задачи быстрее других алгоритмов. Разница в скорости, по сравнению с трехмерным случаем, изменилась не существенно. При сравнении средних значений времени, потраченного методами на нахождение оптимума, снова получаем разницу примерно в четыре минуты.

Метод роя частиц и светлячковый алгоритм имеют приблизительно равные результаты. Среднее время поиска, необходимое методу роя частиц – 25,35 мин, светлячковому алгоритму – 24,16 мин. Разница в среднем времени чуть больше минуты.

Пчелиный алгоритм, как видно на графике, ищет оптимальное решение дольше всех.

Таким образом, быстрее всех решение задачи, как и в случае трех измерений, ищет генетический алгоритм. Вторым по скорости будет муравьиный алгоритм, третьим – светлячковый, четвертым – метод роя частиц. Самым медленным снова оказался пчелиный алгоритм. Для получения результата исходя из средних значений ему нужно практически на 20 минут больше, чем генетическому алгоритму.

Анализ решения задач с пятью параметрами

Для анализа решения задач с пятью параметрами было проведено 100 экспериментов. На графике представлены результаты исследования:

Рис. 9. Результаты проведения экспериментов для случая пятимерных задач

Проведем анализ полученных результатов.

Ось абсцисс описывает номер проводимого эксперимента, ось ординат – время выполнения итерации. Время измеряется в минутах с точностью до секунд. Цена одного деления – 1 минута.

При использовании муравьиного алгоритма среднее время, потраченное на нахождение оптимальных решений, составляет 20,24 мин. При этом наименьшее количество затраченного времени – 17,02 мин (эксперимент № 47), наибольшее – 23,31 мин (эксперимент № 53).

При использовании метода роя частиц среднее время, потраченное на нахождение оптимальных решений, составляет 28,03 мин. При этом наименьшее количество затраченного времени – 23,49 мин (эксперимент № 64), наибольшее – 31,12 мин (эксперимент № 59).

При использовании светлячкового алгоритма среднее время, потраченное на нахождение оптимальных решений, составляет 26,33 мин. При этом наименьшее количество затраченного времени – 23,14 мин (эксперимент № 88), наибольшее – 29,17 мин (эксперимент № 28).

При использовании пчелиного алгоритма среднее время, потраченное на нахождение оптимальных решений, составляет 38,10 мин. При этом наименьшее количество затраченного времени – 34,05 мин (эксперимент № 35), наибольшее – 40,59 мин (эксперимент № 13).

При использовании генетического алгоритма среднее время, потраченное на нахождение оптимальных решений, составляет 17,03 мин. При этом наименьшее количество затраченного времени – 13,52 мин (эксперимент № 53), наибольшее – 20,24 мин (эксперимент № 34).

Таким образом, можно сделать следующие выводы.

Как видно на графике, в случае пяти измерений генетический и муравьиный алгоритмы ищут оптимальное решение задачи быстрее других алгоритмов. Разница в скорости, по сравнению с четырехмерным случаем, изменилась не существенно. При сравнении средних значений времени, потраченного методами на нахождение оптимума, снова получаем разницу примерно в четыре минуты.

Метод роя частиц и светлячковый алгоритм снова имеют приблизительно равные результаты. Среднее время поиска, необходимое методу роя частиц – 28,03 мин, светлячковому алгоритму – 26,33 мин. Разница в среднем времени примерно полторы минуты.

Пчелиный алгоритм, как видно на графике, ищет оптимальное решение дольше всех.

Таким образом, быстрее всех решение задачи, как и в случае трех и четырех измерений, ищет генетический алгоритм. Вторым по скорости будет муравьиный алгоритм, третьим – светлячковый, четвертым – метод роя частиц. Самым медленным снова оказался пчелиный алгоритм. Для получения результата исходя из средних значений ему нужно практически на 21 минуту больше, чем генетическому алгоритму.

Сравнительный анализ полученных результатов

В следующей таблице приведен анализ полученных результатов (3, 4, 5 – количество параметров в задачах):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|   | Муравьиныйалгоритм | Метод роя частиц | Светлячковыйалгоритм | Пчелиныйалгоритм | Генетическийалгоритм |
| min | max | среднее | min | max | среднее | min | max | среднее | min | max | среднее | min | max | среднее |
| 3 | 12,06 | 18,06 | 15,21 | 19,04 | 26,03 | 22,15 | 17,04 | 25,53 | 21,08 | 29,01 | 36,07 | 33,12 | 7,49 | 15,58 | 11,35 |
| 4 | 15,01 | 22,27 | 19,01 | 22,01 | 29,12 | 25,35 | 21,04 | 27,24 | 24,16 | 31,49 | 39,11 | 35,46 | 12,01 | 18,12 | 15,09 |
| 5 | 17,02 | 23,31 | 20,24 | 23,49 | 31,12 | 28,03 | 23,14 | 29,17 | 26,33 | 34,05 | 40,59 | 38,10 | 13,52 | 20,24 | 17,03 |

Таблица 4. Результаты проведения экспериментов

На основании полученных данных можно сделать следующие выводы.

С увеличением числа параметров в задачах количество времени, потраченное на нахождение их оптимальных решений, изменялось равномерно в пределах двух-пяти минут, т.е. для каждого метода среднее время решения задачи изменялось в данных пределах.

На графиках в п. 6.2-6.4 видно, что муравьиный и генетический алгоритмы, светлячковый алгоритм и метод роя частиц в некоторых случаях имеют сходное время для поиска оптимума, однако оценка эффективности учитывает среднее значение времени для решения задач.

В целом, можно сделать следующий вывод. Для решения всех типов задач самым оптимальным является генетический алгоритм, самым медленным – пчелиный. Муравьиный алгоритм занимает второе место, однако в некотрых случаях его результаты немного лучше, чем у генетического алгоритма. Светлячковый алгоритм и метод роя частиц делят третье и четвертое места соответственно. Время их работы отличается несущественно, однако при оценке среднего времени, потраченного на поиск оптимального решения, светлячковый алгоритм выигрывает у метода роя частиц.

Заключение

В ходе написания магистерской диссертации был изучен математический аппарат возможностной оптимизации, метод оптимизации с помощью генетических алгоритмов, а также алгоритмы интеллектуальной оптимизации.

Для исследований была выбрана модель уровневой оптимизации с ограничениями по возможности. Сравнительный анализ выполнен для задач с тремя, четырьмя и пятью параметрами. Для проведения анализа была сгенерирована выборка из 300 задач (по 100 на каждую размерность), удовлетворяющих условиям невырожденности области определения детерминированного эквивалента.

Разработан программный комплекс, реализующий алгоритмы интеллектуальной оптимизации, с помощью которых были проведены эксперименты.

После решения задач выбранными алгоритмами был проведен сравнительный анализ эффективности работы исследуемых алгоритмов интеллектуальной оптимизации.

 Таким образом, все задачи магистерской диссертации выполнены, цели достигнуты.

Список литературы

1. Солдатенко И.С. О методе решения одной задачи возможностного программирования в случае Tw-связанных параметров // Сб. научн. тр. IV-й Междунар. научно-практической конф. «Интегрированные модели и мягкие вычисления в искусственном интеллекте». Т.1, М.: издат. Физматлит, 2007.
2. Солдатенко И.С. О генетическом алгоритме решения задачи возможностной оптимизации при взаимно Tw-связанных параметрах // Вестник Тверского гос. ун-та. Сер. Прикладная математика. №4(64), вып. 8, 2008.
3. Язенин А.В. Возможностное и интервальное линейное программирование // Изв. РАН. Техн. кибернетика. 1993. N5.
4. Солдатенко И.С., Язенин А.В. Задача возможностной оптимизации с взаимно t-связанными параметрами: сравнительное изучение // Известия РАН. Теория и системы управления, 2008, № 5.
5. D.T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri , S. Rahim , M. Zaidi. The Bees Algorithm – A Novel Tool for Complex Optimisation Problems // Manufacturing Engineering Centre, Cardiff University, UK.
6. Firefly and Glowworm Algorithms // LIACS Natural Computing Group Leiden University.
7. Гладков Л. А., Курейчик В. В., Курейчик В. М. Генетические алгоритмы: Учебное пособие. – 2-е изд. – М: Физматлит, 2006.
8. Swagatam Das, Arijit Biswas, Sambarta Dasgupta, Ajith Abraham. Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications // Dept. of Electronics and Telecommunication Engg, Jadavpur University, Kolkata, India; Norwegian University of Science and Technology, Norway.
9. Kris De Meyer. Foundations of Stochastic Diffusion Search // Philosophy Department of Cybernetics, Norwegian University of Science and Technology, Norway – 2004.
10. Esmat Rashedi, Hossein Nezamabadi-pour, Saeid Saryazdi. A Gravitational Search Algorithm // Department of Electrical Engineering, Shahid Bahonar University of Kerman, Iran.
11. Hamed Shah-Hosseini. Optimization with the Nature-Inspired Intelligent Water Drops Algorithm // Faculty of Electrical and Computer Engineering, Shahid Beheshti University, G.C.Tehran, Iran.
12. Карпенко А.П. Популяционные алгоритмы глобальной поисковой оптимизации // Новые технологии. Приложение к журналу «Информационные технологии». 2012. N7.
13. Магомедова М.Н. Когнитивное моделирование в системах искусственного интеллекта // Конкурс «Молодежь и наука»: печатные материалы. СПбГИЭУ, г. Кизляр, 2012 г.
14. Ильясов Ф.Н. Разум искусственный и естественный // Известия АН Туркменской ССР, серия общественных наук. 1986. №6.
15. <http://www.intuit.ru/> – Интернет-Университет Информационных Технологий
16. <http://ru.wikipedia.org/> – свободная Интернет-энциклопедия
17. <http://www.chtivo.ru/> – объединенный каталог печатных изданий
18. <http://sknowledge.ru/> – «Источник знаний» (сайт об учебно-справочных материалах по различным областям знаний)
19. <http://habrahabr.ru/> – сайт с публикациями статей, связанных с информационными технологиями