ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Дисциплина «Гибридные интеллектуальные системы и мягкие вычисления»

**ОТЧЕТ**

**Лабораторная работа №1**

на тему:

«Меметические алгоритмы для обучения нейронных сетей: подход, основанный на алгоритме гравитационного поиска»

Выполнил:

студент группы 3540901/02001

Бараев Дамир Рашидович

«\_\_» \_\_\_\_\_\_\_\_\_\_ 2021г., \_\_\_\_\_\_\_\_\_\_

(подпись)

(подпись)

Проверила:

Бендерская Елена Николаевна

«\_\_» \_\_\_\_\_\_\_\_\_\_ 2021г., \_\_\_\_\_\_\_\_\_\_

Санкт-Петербург 2021

**Содержание**

Оглавление

[1. Постановка задачи, связанной с рассматриваемой системой 3](#_Toc72463731)

[2. Описание теоретической базы рассматриваемой системы. 3](#_Toc72463732)

[3. Задача обучения нейронной сети 4](#_Toc72463733)

[Алгоритм хаотического гравитационного поиска (CGSA) 5](#_Toc72463734)

[Меметические алгоритмы на основе GSA и CGSA 6](#_Toc72463735)

[Квазиньютоновские методы 7](#_Toc72463736)

[Моделирование алгоритма GSA+QM в среде MatLab 7](#_Toc72463737)

[Сравнение M-CGSA с другими алгоритмами. 11](#_Toc72463738)

[Вывод 12](#_Toc72463739)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 14](#_Toc72463740)

[ПРИЛОЖЕНИЕ 15](#_Toc72463741)

# Постановка задачи, связанной с рассматриваемой системой

В данной работе рассматривается задача глобальной оптимизации многоэкстремальных функций с помощью алгоритма гравитационного поиска. Классический алгоритм является стохастическим и основан на гравитационном взаимодействии совокупностей масс и законах движения. На основе сил определяются векторы скорости и ускорения каждого зонда для дальнейшего его перемещения. Данный алгоритм схож с методом роя частиц, так как базируется на развитии многоагентной системы. Также изучается применение меметических подходов (MGSA и MCGSA) к сложной задаче машинного обучения, в частности к обучению FNN (нейронной сети с прямой связью)

Основным преимуществом использования меметического подхода является более высокая скорость сходимости, чем у метаэвристических алгоритмов, и возможность найти глобальный оптимум проблемы, в отличие от BP (Алгоритм обратного распространения).

# Описание теоретической базы рассматриваемой системы.

Гравитационный поиск (GS) является молодым алгоритмом. Появился он в 2009 году и являлся логическим развитием метода центральной силы. Основу GS составляют законы гравитации и взаимодействия масс. В принципе, данный алгоритм похож на методы роя частиц (Particle Swarm Optimization — PSO), так как базируется на развитии многоагентной системы.

**Стратегия**

GS оперирует двумя законами:

1. Тяготения: 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; Автор объясняет это тем, что во всех тестах это дает лучшие результаты, оставим это на его совести)
2. 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, деленной на инерциальную массу частицы.

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

* генерация системы случайным образом,
* определение приспособленности каждой частицы,
* обновление значений гравитационной постоянной, лучшей и худшей частиц, а также масс,
* подсчет результирующей силы в различных направлениях,
* подсчет ускорений и скоростей,
* обновление позиций частиц,
* повторений шагов 2 — 6 до выполнения критерия окончания (либо превышение максимального количества итераций, либо слишком малой изменение позиций, либо что вашей душе угодно любой другой осмысленный критерий).

**Использование GSA для обучения нейронных сетей**

Метаэвристика и их гибридизация была применена к задаче обучения FNN, чтобы избежать недостатков алгоритмов на основе градиентов. Гибридные алгоритмы можно разделить на две основные категории:

* Комбинации двух алгоритмов для использования возможностей локального и глобального поиска обоих алгоритмов. Таким образом, [8] предложили меметический алгоритм, основанный на GA и BP, в то время как [7] предложил меметический PSO с алгоритмом BP для обучения FNN
* Комбинации двух метаэвристик с целью получения более надежного глобального поиска. Типичным примером этой тенденции является [6], где PSO и GA были использованы для разработки гибридного алгоритма для проектирования рекуррентных нейронных сетей.

**Использование GSA для оптимизации топологии сети**

Выбор оптимального количества скрытых нейронов и скрытых слоев является ключевым моментом при рассмотрении FNN. Это сложная проблема, поскольку ее нужно настраивать под конкретную проблему. По этой причине многие исследования предлагают разные размеры для скрытого слоя, а затем выбирают сеть, которая демонстрирует лучшую производительность.

# Задача обучения нейронной сети

Связи между нейронами из разных слоев всегда прямые, и обычно все нейроны одного слоя связаны со всеми нейронами следующего слоя.

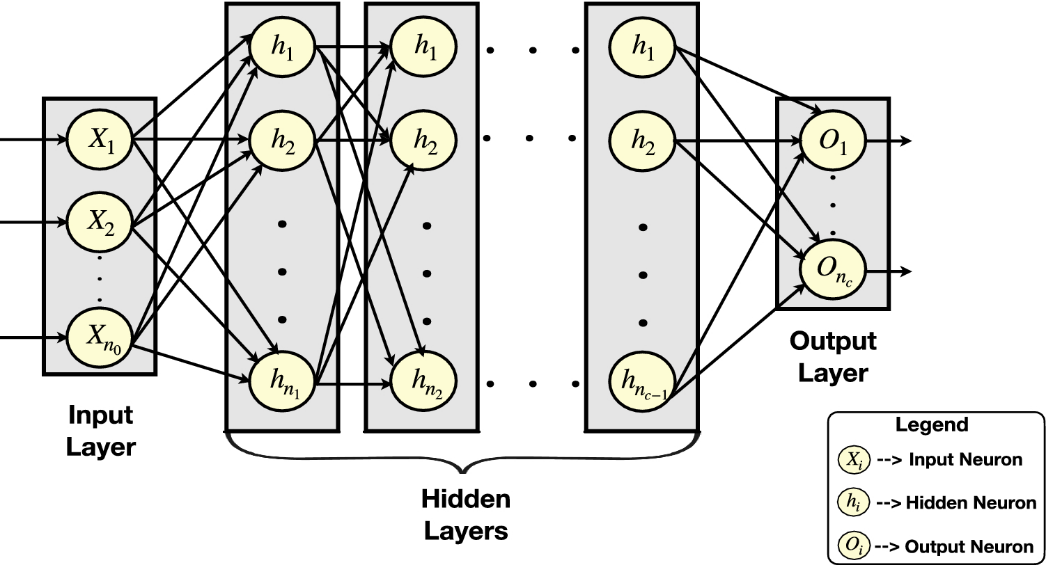


Рисунок 1 - Общая архитектура FNN

В каждой FNN связи между нейронами из разных слоев связаны с действительным числом, так называемым весом соединения.

В тренировочном процессе FNN есть два основных этапа. Сначала входы сети распространяются вперед, чтобы получить выход. Затем вычисляется ошибка между выходным сигналом сети и желаемым значением. Наконец, на втором этапе ошибки распространяются в обратном направлении, регулируя веса и пороги для каждого нейрона в топологии, чтобы минимизировать функцию ошибок или потерь (определенную разработчиком). Этот последний шаг будет выполнен с использованием метода оптимизации.

GSA — это био-вдохновленный алгоритм, предложенный Рашеди. Он основан на теории ньютоновской гравитации в физике таким образом, что каждое решение в пространстве поиска соответствует массе в пространстве по метафоре Вселенной. Затем, в соответствии с универсальным законом всемирного тяготения, более тяжелые массы (решения с лучшими значениями пригодности) притягивают меньшие (решения с худшими значениями приспособленности), обеспечивая сходимость.

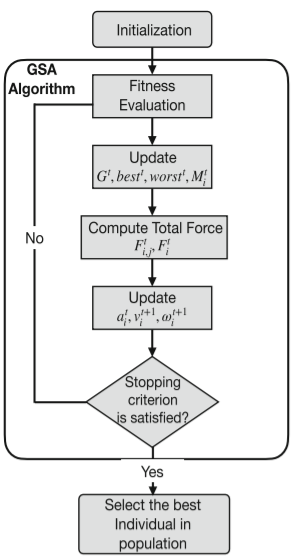


Рисунок 2 - Блок-схема GSA

# Алгоритм хаотического гравитационного поиска (CGSA)

CGSA — это вариант GSA, который появился в [3] и направлен на решение проблемы медленной конвергенции, присущей GSA, и улучшения компромисса между разведкой и разработкой.

В GSA компромисс между разведкой и разработкой контролируется двумя функциями и с одной стороны, поощряет шаг исследования алгоритма, позволяя всем массам приложить свои силы к остальным. Эта функция уменьшается с использованием линейной функции, и, следовательно, по мере увеличения количества итераций количество масс, воздействующих на остальные, уменьшается, фокусируя поиск на наиболее многообещающих средах, найденных на данный момент, таким образом, приближает стадию эксплуатации к концу.

В [3] добавлены различные хаотические отображения, чтобы нарушить поведение. В исходном GSA постоянно уменьшается на итерациях, поэтому алгоритм либо исследует, либо использует. Добавление хаотической карты к гравитационной постоянной будет хаотично изменять значение, уменьшая его во время итераций, давая возможность исследования и эксплуатации в начале и в конце алгоритма. Этот подход значительно улучшает производительность исходного GSA, показывая, что синусоидальная хаотическая карта является наиболее подходящей для GSA. Графический пример введения хаотическую синусоидальный карту показан на рисунке 3.

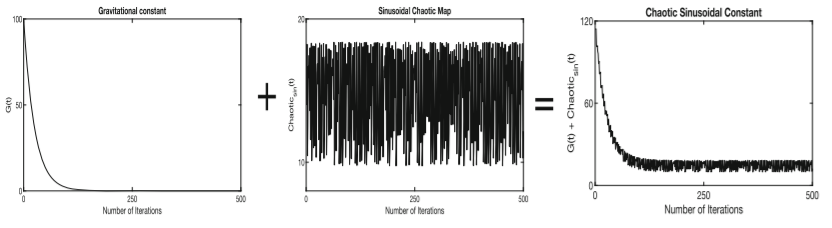


Рисунок 3 - Получение хаотической синусоидальной гравитационной постоянной

# Меметические алгоритмы на основе GSA и CGSA

Дизайн обоих меметических алгоритмов основан на описании локальных минимумов. Основа этих алгоритмов заключается в том, что они способны обнаруживать перспективную окрестность, которая запускает метод qN. Таким образом, добавление механизмов для характеристики локальных минимумов является важным этапом в разработке меметических алгоритмов. Эта статья характеризует концепцию многообещающей окрестности неисследованного локального минимума с использованием двух правил:

* он содержит решение, которое имеет лучшее значение пригодности, чем лучшее из найденных до сих пор
* оно расположено далеко от текущего лучшего решения.

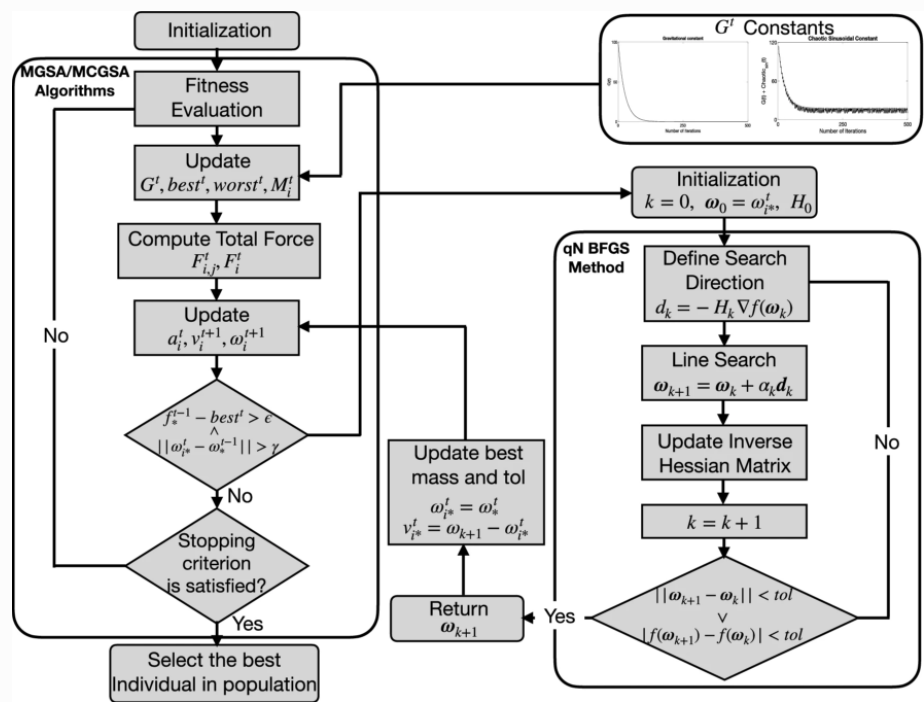


Рисунок 4 - Блок-схема алгоритмов MGSA и MCGSA

На блок-схеме можно увидеть, как выполняется метаэвристический алгоритм, и при обнаружении перспективной области (удовлетворяющей правилам 1 и 2) запускается метод qN, начиная с лучшей текущей точки. Точка, полученная методом qN, заменит лучшую точку населения, обновляя не только ее положение, но и скорость. Структура этого меметического алгоритма также позволяет расширить использование метода qN на недифференцируемые задачи. Если меметический алгоритм достигает недифференцируемой точки, то этап локального поиска, выполняемый qN, будет нулевым этапом. Однако меметический алгоритм избегает этой проблемной точки благодаря своей эволюционной структуре.

# Квазиньютоновские методы

Квазиньютоновские методы — методы оптимизации, основанные на накоплении информации о кривизне целевой функции по наблюдениям за изменением градиента, чем принципиально отличаются от ньютоновских методов. Класс квазиньютоновских методов исключает явное формирование матрицы Гессе, заменяя её некоторым приближением.

Меметический алгоритм на основе гравитационного поиска состоит из двух этапов. Этап разведки и этап эксплуатации. В этапе разведки как раз и используется алгоритм гравитационного поиска. Основные формулы для движения частиц — это гравитационная сила на массу I от массы j. Ускорение высчитывается по второму закону ньютона. То есть общая сила делится на массу объекта.

После выполнения гравитационного поиска осуществляется этап эксплуатация с помощью квазиньютоновского метода. А именно итерационный метод численной оптимизации BFGS - формула Бройдена-Флетчера–Гольдфарба–Шанно. В отличие от ньютоновских методов в квазиньютоновских не вычисляется напрямую гессиан функции, т.е. нет необходимости находить частные производные второго порядка. Вместо этого гессиан вычисляется приближенно, исходя из сделанных до этого шагов. Он аппроксимирует гессиан функции что в итоге даёт более быструю и точную оптимизацию.

Фундаментальная идея квазиньютоновского обновления состоит в том, чтобы объединить самую последнюю наблюдаемую информацию о целевой функции с существующими знаниями, вложенными в текущее обратное Гессианово приближение.

# Моделирование алгоритма GSA+QM в среде MatLab

В ходе выполнения лабораторной работы было проведено моделирование алгоритма гравитационного поиска в среде Matlab.

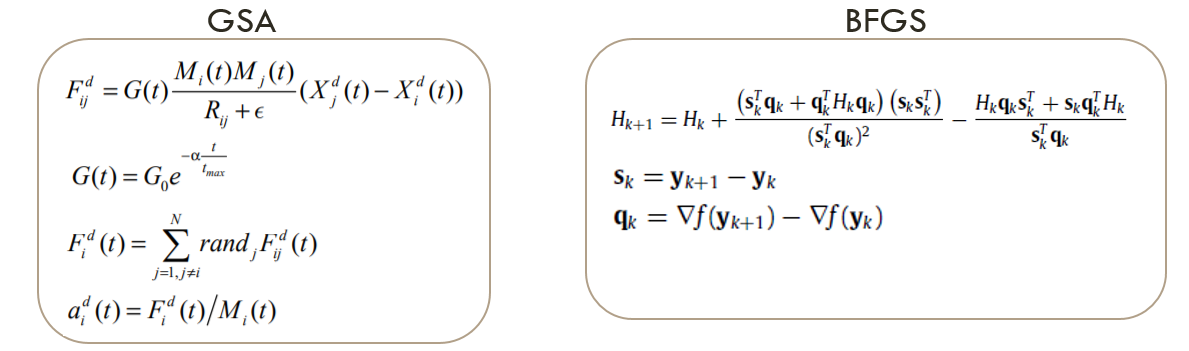


Рисунок 5 - Формулы двух этапов алгоритма

Расчёт каждого параметра был разделен на функции.

В функции chaos гравитационная постоянная может рассчитываться разными способами. В зависимости от того какая хаотичная карта выбрана. Например: Gauss iterated map, Tent map, Chebyshev map. Со всеми функциями я попробовал разные хаотические карты и лучший результат выписал в таблицу.

В статье сравниваются несколько эвристических алгоритмов, но я провел сравнение между алгоритмом гравитационного поиска и меметическим хаотическим гравитационным алгоритмом (гравитационный алгоритм + квазиньютоновский метод оптимизации). Сравнивал по такому критерию, как количество итераций нужные для полной сходимости. Гибридный алгоритм во всех задачах оптимизации показал себя лучше.

Таблица 1 - Сравнение GSA и M-GSA

|  |  |  |
| --- | --- | --- |
|  | MGSA | GSA |
| Функция 1 | 70 | 95 |
| Функция 2 | 65 | 80 |
| Функция 3 | 60 | 85 |
| Функция 4 | 70 | 85 |
| Функция 5 | 60 | 85 |
| Функция 6 | 65 | 70 |
| Функция 7 | 70 | 85 |
| Функция 8 | 75 | 90 |
| Функция 9 | 60 | 80 |
| Функция 10 | 70 | 85 |

**Этап разведки. Запуск алгоритма GSA в Matlab.**

* N (Количество агентов) – 150
* G0 (начальное значение гравитационной постоянной) – 1000
* ITER (Количество итераций) – 25

Изображение выглядит как текст, компьютер, внутренний, другой

Автоматически созданное описание

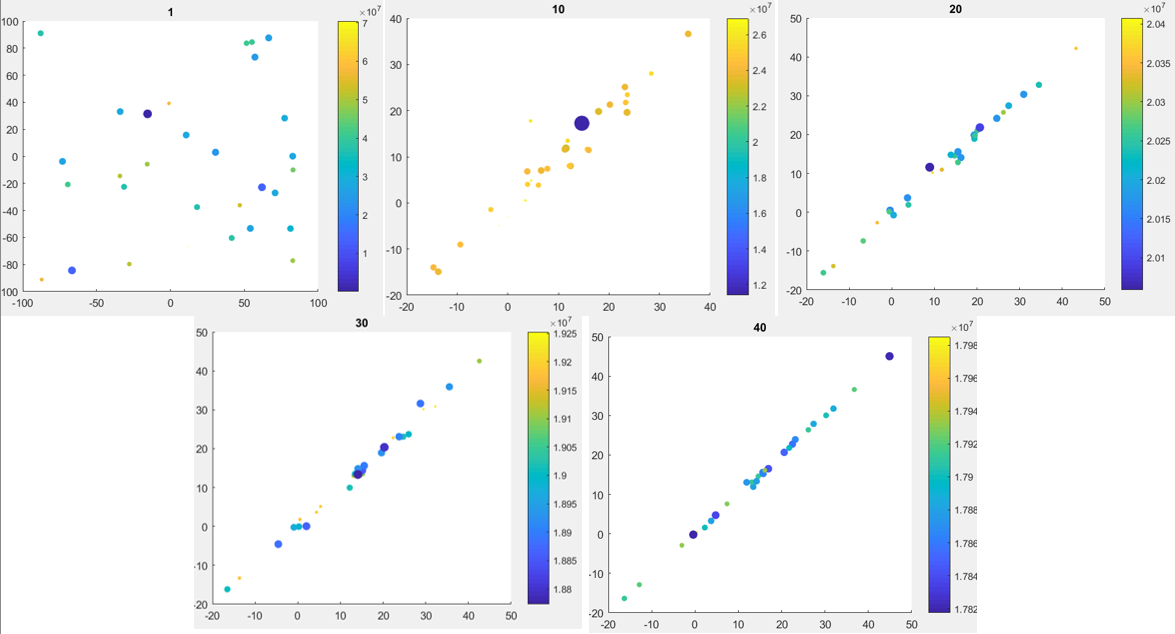
Изображение выглядит как текст, компьютер, снимок экрана

Автоматически созданное описание

Рисунок 6 - Визуализация работы алгоритма GSA в MatLab

**Этап эксплуатации (разведки+эксплуатации). Запуск алгоритма M-CGSA в Matlab.**

* N (Количество агентов) – 30
* ITER (Макс. количество итераций) – 100



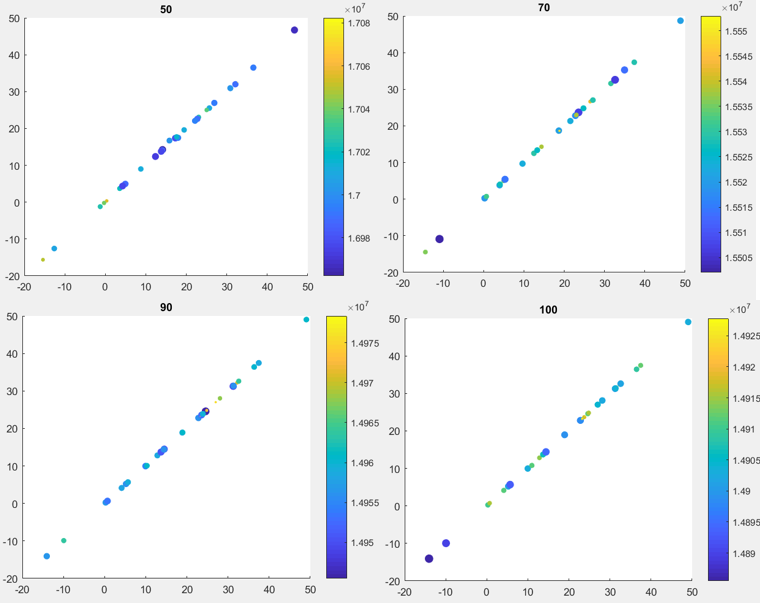


Рисунок 7 - Визуализация работы алгоритма M-CGSA в MatLab

**Этап эксплуатации (разведки+эксплуатации). Запуск алгоритма GSA в Matlab.**

Сходимость гораздо хуже, чем при запуске выше.

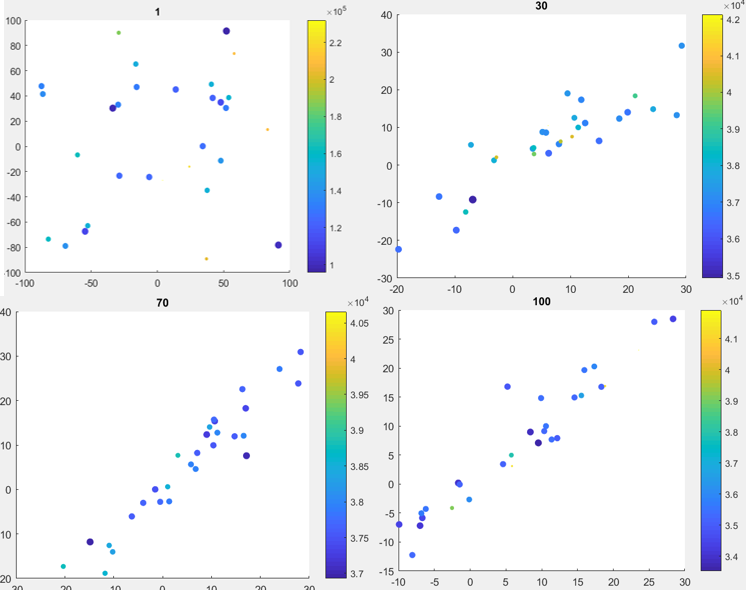


Рисунок 8 - Этап эксплуатации. Визуализация работы алгоритма GSA в MatLab

# Сравнение M-CGSA с другими алгоритмами.

Так же в статье эвристический алгоритм сравнивается с другими алгоритмами, но так как реализовать их не представилось возможности, проверить результаты на достоверность также не получилось. Однако если верить результатам авторов, то и здесь алгоритм гравитационного поиска показывает хорошие результаты.

По проведенным экспериментам автора, в статье делается вывод, что M-CGSA работает значительно лучше, чем исходный GSA, в 23 тестовых задачах. Более того, существуют 5 тестовых задач, в которых оба алгоритма показывают одинаковую производительность. Также примечательно, что не возникают проблемы, когда GSA превосходит M-CGSA.

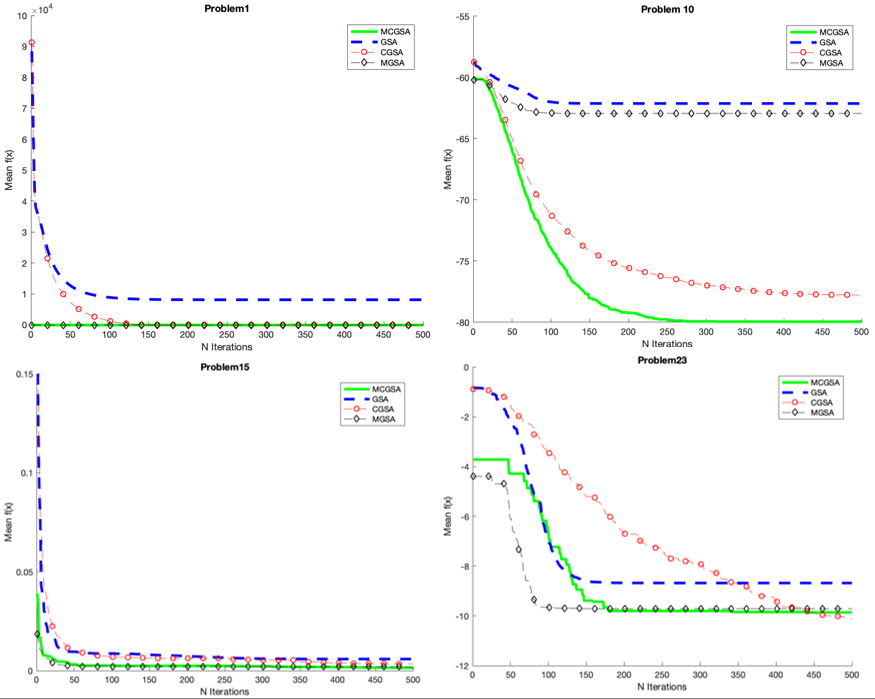


Рисунок 9 - Сравнение алгоритмов в синтетических задачах

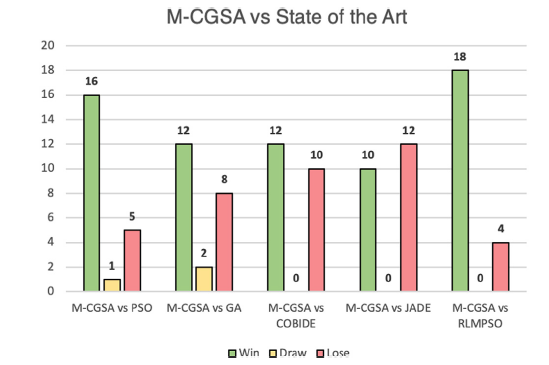


Рисунок 10 - Сравнение M-СGSA с другими эвристическими алгоритмами

Только алгоритм JADE опережает его по количеству выполненных задач.

# Вывод

В ходе данной лабораторной работы провели сравнение между двумя алгоритмами: алгоритм гравитационного поиска и гибридный алгоритм гравитационного поиска с использованием квазиньютоновского метода оптимизации. Гибридный алгоритм оказался быстрей и более точный в задачах оптимизации. Так же привел в сравнения и другие эвристические алгоритмы, но так как реализовать их я не смог то все результаты были взяты из статьи. При подборе хаотических карт для каждой функции выяснилось, что лучшей нет, для каждой задачи их нужно подбирать.

Выводы по алгоритму:

* Прост в реализации
* Метод достаточно точный
* Большая скорость сходимости
* Результаты лучше, чем у алгоритма гравитационного поиска
* В среднем результаты лучше, чем у схожих эвристических алгоритмов оптимизации

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Mazurowski MA, Habas PA, Zurada JM, Lo JY, Baker JA, Tourassi GD (2008) Training neural network classifiers for medical decision making: the effects of imbalanced datasets on classification performance. Neural Netw 21(2–3):427–436
2. Mirjalili S, Mohd Hashim S, Moradian Sardroudi H (2012) Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. Appl Math Comput 218(22):11125–11137
3. Rashedi E., Nezamabadi-Pour H., Saryazdi S. GSA: a gravitational search algorithm //Information sciences. – 2009. – Т. 179. – №. 13. – С. 2232-2248.
4. García-Ródenas R., Linares L. J., López-Gómez J. A. A Memetic Chaotic Gravitational Search Algorithm for unconstrained global optimization problems //Applied Soft Computing. – 2019. – Т. 79. – С. 14-29.
5. Zhang JR, Zhang J, Lok TM, Lyu MR (2007) A hybrid particle swarm optimization-back-propagation algorithm for feedforward neural network training. Appl Math Comput 185(2):1026–1037
6. Гравитационный поиск. Gravitational Search - [Электронный ресурс]. URL: https://habr.com/ru/post/194674/ (дата обращения: 14.02.2021).
7. Метод BFGS или один из самых эффективных методов оптимизации. Пример реализации на Python - [Электронный ресурс]. URL: https://habr.com/ru/post/333356/ (дата обращения: 15.02.2021).

# ПРИЛОЖЕНИЕ

Текст программы – главный скрипт GSA.m

|  |
| --- |
| function [BestChart,Eval\_f]=GSA(F\_index,N,max\_it,ElitistCheck,chaosIndex,chValueInitial,k)    global contador  G\_History=zeros(1,max\_it);  Rnorm=2;  Rpower=1;  min\_flag=1; % 1: minimization, 0: maximization  Tolerancia=0.01;    [low,up,dim]=test\_functions\_range(F\_index)  X =initialization(dim,N,up,low);    BestChart=[];MeanChart=[];Eval\_f=[];    V=zeros(dim,N);  wMax=chValueInitial;  wMin=1e-10;  for iteration=1:max\_it  chValue=wMax-iteration\*((wMax-wMin)/max\_it);    %  X=space\_bound(X,up,low)    fitness=evaluateF(X,F\_index);    [best best\_X]=min(fitness); %min: minimization. max: maximization.      if iteration==1  Fbest=best;Lbest=X(:,best\_X);Ibest=best\_X;  n\_c=1;  end  if best<Fbest % < : minimization. > : maximization  Fbest=best;  if norm(Lbest-X(:,best\_X)) >=1  n\_c=1;  Tolerancia=Tolerancia/10;  end  Lbest=X(:,best\_X); Ibest=best\_X;  end          % explotation stage      if (n\_c==1 & k==1)  n\_c=0;  [X,V,fitness,Fbest]=ExplotationStage(iteration,X,V,fitness,Ibest,Fbest,dim,F\_index,Tolerancia);  end    BestChart=[BestChart Fbest];  MeanChart=[MeanChart mean(fitness)];  Eval\_f=[Eval\_f contador];    %Calculation of M. eq.14-20  [M]=massCalculation(fitness,min\_flag);    %Calculation of Gravitational constant. eq.13.  G=Gconstant(iteration,max\_it);  %G=chaos(3,iteration,max\_it,10);  switch chaosIndex  case 1  G=Gconstant(iteration,max\_it);  case 2  G=G+chaos(chaosIndex-1,iteration,max\_it,chValue);  case 3  G=G+chaos(chaosIndex-1,iteration,max\_it,chValue);  case 4  G=G+chaos(chaosIndex-1,iteration,max\_it,chValue);  case 5  G=G+chaos(chaosIndex-1,iteration,max\_it,chValue);  case 6  G=G+chaos(chaosIndex-1,iteration,max\_it,chValue);  case 7  G=G+chaos(chaosIndex-1,iteration,max\_it,chValue);  case 8  G=G+chaos(chaosIndex-1,iteration,max\_it,chValue);  case 9  G=G+chaos(chaosIndex-1,iteration,max\_it,chValue);  case 10  G=G+chaos(chaosIndex-1,iteration,max\_it,chValue);  case 11  G=G+chaos(chaosIndex-1,iteration,max\_it,chValue);  end  G\_History(iteration)=G;  test\_G(iteration)=G;  if iteration==499  alisop=0;  end  %Calculation of accelaration in gravitational field. eq.7-10,21.  a=Gfield(M,X,G,Rnorm,Rpower,ElitistCheck,iteration,max\_it);    %Agent movement. eq.11-12  [X,V]=move(X,a,V);  if(iteration == 1)  scatter(X(:,1), X(:,2), M\*1000+1, fitness, 'filled');  colorbar  title(iteration);  pause(10)  elseif(iteration == 10)  scatter(X(:,1), X(:,2), M\*1000+1, fitness, 'filled');  colorbar  title(iteration);  pause(10)  elseif(iteration == 20)  scatter(X(:,1), X(:,2), M\*1000+1, fitness, 'filled');  colorbar  title(iteration);  pause(10)  elseif(iteration == 30)  scatter(X(:,1), X(:,2), M\*1000+1, fitness, 'filled');  colorbar  title(iteration);  pause(10)  elseif(iteration == 40)  scatter(X(:,1), X(:,2), M\*1000+1, fitness, 'filled');  colorbar  title(iteration);  pause(10)  elseif(iteration == 50)  scatter(X(:,1), X(:,2), M\*1000+1, fitness, 'filled');  colorbar  title(iteration);  pause(10)  elseif(iteration == 60)  scatter(X(:,1), X(:,2), M\*1000+1, fitness, 'filled');  colorbar  title(iteration);  pause(10)  elseif(iteration == 70)  scatter(X(:,1), X(:,2), M\*1000+1, fitness, 'filled');  colorbar  title(iteration);  pause(10)  elseif(iteration == 80)  scatter(X(:,1), X(:,2), M\*1000+1, fitness, 'filled');  colorbar  title(iteration);  pause(10)  elseif(iteration == 90)  scatter(X(:,1), X(:,2), M\*1000+1, fitness, 'filled');  colorbar  title(iteration);  pause(10)  elseif(iteration == 100)  scatter(X(:,1), X(:,2), M\*1000+1, fitness, 'filled');  colorbar  title(iteration);  pause(10)  end    end |

Текст программы – Расчёт гравитационной постоянной Gconstant.m

|  |
| --- |
| function G=Gconstant(iteration,max\_it)    alfa=20;G0=100;  G=G0\*exp(-alfa\*iteration/max\_it); |

Текст программы – Расчёт масс massCalculation.m

|  |
| --- |
| function [M]=massCalculation(fit,min\_flag);    Fmax=max(fit); Fmin=min(fit); Fmean=mean(fit);  [i N]=size(fit);    if Fmax==Fmin  M=ones(N,1);  else    if min\_flag==1 %for minimization  best=Fmin;worst=Fmax; %eq.17-18.  else %for maximization  best=Fmax;worst=Fmin; %eq.19-20.  end    M=(fit-worst)./(best-worst); %eq.15,    end    M=M./sum(M); %eq. 16. |

Текст программы – Расчёт движения агентов move.m

|  |
| --- |
| %This function updates the velocity and position of agents.  function [X,V]=move(X,a,V)    %movement.  [dim,N]=size(X);  V=rand(dim,N).\*V+a; %eq. 11.  X=X+V; %eq. 12. |

Текст программы – проверка границы пространства поиска для агентов space\_bound.m

|  |
| --- |
| function X=space\_bound(X,up,low);    % Agents that go out of the search space, are returned to the boundaries.  [dim,mass\_num]=size(X);  for i=1:mass\_num  Tp=X(:,i)>up;Tm=X(:,i)<low;X(:,i)=(X(:,i).\*(~(Tp+Tm)))+up.\*Tp+low.\*Tm;  end |

Текст программы – расчёт ускорения каждого агента в гравитационном поле Gfield.m

|  |
| --- |
| function a=Gfield(M,X,G,Rnorm,Rpower,ElitistCheck,iteration,max\_it);    [dim,N]=size(X);  final\_per=2;    %%%%total force calculation  if ElitistCheck==1  kbest=final\_per+(1-iteration/max\_it)\*(100-final\_per); %kbest in eq. 21.  kbest=round(N\*kbest/100);  else  kbest=N; %eq.9.  end  [Ms ds]=sort(M,'descend');    for i=1:N  E(:,i)=zeros(dim,1);  for ii=1:kbest  j=ds(ii);  if j~=i  R=norm(X(:,i)-X(:,j),Rnorm); for k=1:dim  E(k,i)=E(k,i)+rand\*(M(j))\*((X(k,j)-X(k,i))/(R^Rpower+eps));  end  end  end  end    a=E.\*G; |

Текст программы –функция придающая модели хаотичность chaos.m

|  |
| --- |
| function O=chaos(index,curr\_iter,max\_iter,Value)    x(1)=0.7;  switch index  %Chebyshev map  case 1  for i=1:max\_iter  x(i+1)=cos(i\*acos(x(i)));  G(i)=((x(i)+1)\*Value)/2;  end  case 2  %Circle map  a=0.5;  b=0.2;  for i=1:max\_iter  x(i+1)=mod(x(i)+b-(a/(2\*pi))\*sin(2\*pi\*x(i)),1);  G(i)=x(i)\*Value;  end  case 3  %Gauss/mouse map  for i=1:max\_iter  if x(i)==0  x(i+1)=0;  else  x(i+1)=mod(1/x(i),1);  end  G(i)=x(i)\*Value;  end  case 4  %Iterative map  a=0.7;  for i=1:max\_iter  x(i+1)=sin((a\*pi)/x(i));  G(i)=((x(i)+1)\*Value)/2;  end  case 5  %Logistic map  a=4;  for i=1:max\_iter  x(i+1)=a\*x(i)\*(1-x(i));  G(i)=x(i)\*Value;  end  case 6  %Piecewise map  P=0.4;  for i=1:max\_iter  if x(i)>=0 && x(i)<P  x(i+1)=x(i)/P;  end  if x(i)>=P && x(i)<0.5  x(i+1)=(x(i)-P)/(0.5-P);  end  if x(i)>=0.5 && x(i)<1-P  x(i+1)=(1-P-x(i))/(0.5-P);  end  if x(i)>=1-P && x(i)<1  x(i+1)=(1-x(i))/P;  end  G(i)=x(i)\*Value;  end    case 7  %Sine map  for i=1:max\_iter  x(i+1) = sin(pi\*x(i));  G(i)=(x(i))\*Value;  end  case 8  %Singer map  u=1.07;  for i=1:max\_iter  x(i+1) = u\*(7.86\*x(i)-23.31\*(x(i)^2)+28.75\*(x(i)^3)-13.302875\*(x(i)^4));  G(i)=(x(i))\*Value;  end  case 9  %Sinusoidal map  for i=1:max\_iter  x(i+1) = 2.3\*x(i)^2\*sin(pi\*x(i));  G(i)=(x(i))\*Value;  end    case 10  %Tent map  x(1)=0.6;  for i=1:max\_iter  if x(i)<0.7  x(i+1)=x(i)/0.7;  end  if x(i)>=0.7  x(i+1)=(10/3)\*(1-x(i));  end  G(i)=(x(i))\*Value;  end    end  O=G(curr\_iter); |