ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

УДК 004.421.2

Отчет об исследовательском проекте на тему:

Гиперэвристика алгоритмов роевого интеллекта

Выполнил студент:

группы #БПМИ236, 2 курса Коростелев Данил Александрович

Принял руководитель проекта:

Родригес Залепинос Рамон Антонио
Научный сотрудник
Факультет компьютерных наук НИУ ВШЭ

Москва 2025

Содержание

Аннотация **3**

1 Введение **4**

2 Эвристики, Метаэвристики, Гиперэвристики **5**

3 Обзор литературы **6**

4 World Hyper-Heuristic (WHH) **7**

4.1 Вознаграждение 8

4.2 Селекция 9

5 Анализ результатов приведенных в статье **9**

6 Анализ исходного кода авторов и воспроизведение полученных результатов **11**

7 Моя реализация кода, описанного в статье и сравнение результатов **13**

8 Заключение **14**

Список литературы **15**

Аннотация

Курсовая работа посвящена исследованию и реализации новой гиперэвристики алгоритмов роевого интллекта World Hyper-Heuristic (WHH). В рамках работы проводится анализ существующих алгоритмов роевого интеллекта для дискретных или непрерывных задач оптимизации, таких как алгоритм муравьиной колонии (ACO), оптимизация роем частиц (PSO) и других, с целью выявления их сильных и слабых сторон. Основное внимание уделяется гиперэвристикам — методам, которые позволяют автоматически комбинировать и адаптировать низкоуровневые эвристики для решения сложных оптимизационных задач.

Цель работы заключается в проверке нового алгоритма WHH, на оптимизационных задачах в многомерных пространствах и предложении модификаций, направленных на улучшение качества и скорости сходимости алгоритма. В ходе исследования планируется изучить предложенный алгоритм и провести его тестирование на стандартных наборах задач оптимизации. Результаты работы будут сравниваться с существующими алгоритмами роевого интеллекта для оценки их производительности и качества решений.

Ключевые аспекты работы включают: изучение теоретических основ роевого интеллекта и гиперэвристик, исследование реализации архитектуры нового алгоритма, предложение модификаций и оптимизаций, и экспериментальное исследование. Практическая значимость работы заключается в возможности применения разработанного алгоритма для решения задач оптимизации в различных областях, таких как машинное обучение, управление ресурсами, планирование и другие.

Ключевые слова

Обучение с подкреплением, NP-hard, Роевой интеллект

1 Введение

Всю историю человечество сталкивается с необходимостью решения различных задач как можно быстрее и экономнее в контексте ресурсов, что неизбежно привело к появлению алгоритмов различной сложности, решающих эти задачи. Многие задачи имеют решения среди алгоритмов полиномиальной сложности, например сортировка массива чисел. Такие алгоритмы в худшем случае требуют веремя пропорциональное $C \times n^k$, где n - размер входных данных, а C и k - константы.

Однако не все задачи могут быть решены при помощи алгоритма полиномиальной сложности (споры об этом до сих пор продолжаются, однако пока что существование таких алгоритмов для всех задач не доказано), к таким задачам относится например проблема комивояжера - оптимизационная задача поиска циклического маршрута с наименьшей стоимостью через все узлы взвешенного графа. Проблема коммивояжера относится к NP-сложным.

Современные задачи оптимизации, возникающие в различных областях науки и техники, зачастую характеризуются высокой сложностью, многомерностью и наличием ограничений. Традиционные методы оптимизации зачастую оказываются недостаточно эффективными для решения таких задач, что стимулирует развитие новых подходов. Одним из таких направлений являются алгоритмы роевого интеллекта, которые имитируют коллективное поведение природных систем, таких как стаи птиц, колонии муравьев или косяки рыб [9]. Эти алгоритмы позволяют с приемлимой скоростью находить субоптимальные решения.

Такие алгоритмы называются метаэвристиками, они начинают с случайного решения и постепенно улучшают его при помощи исследования, иначе говоря, поиска решений, не являющихся близкими к текущему, или эксплуатации, то есть поиска решений среди близких к текущему. Очень важно чтобы алгоритм соблюдал баланс между этими подходами.

Согласно No Free Lunch Theory [11] каждый из этих алгоритмов будет хорош лишь для определенного подмножества проблем и найти универсальный алгоритм не получится. Тут на помощь приходят Гиперэвристики [2] - комбинации метаэвристик. Такие алгоритмы переключаются между различными метаэвристиками и позволяют свести задачу к выбору подходящего решения вместо его поиска, а также позволяют лучше контролировать соотношение исследования и эксплуатации.

В 2024 году ученые из университетов Ирана и США предложили свой алгоритм, основанный на гиперэвристике алгоритмов роевого интеллекта, названный World Hyper-Heuristic [3]. Приведенные в статье данные демонстрируют значительное превосходство WHH над самыми известными метаэвристиками, что не могло не привлечь внимание к теме. В сво-

ей курсовой работе я постраюсь воспроизвести результаты полученные в статье и предложу возможные улучшения и оптимизации.

2 Эвристики, Метаэвристики, Гиперэвристики

Для понимания контекста критически важно разобраться в терминологии, а именно в том, что такое и чем ращличаются эвристики, метаэвристики и гиперэвристики.

Эвристика (эвристический алгоритм) - алгоритм, основывающийся на практике, не являющийся гарантированно точным или оптимальным, но достаточный для решения (ускорения решения) поставленной задачи[1]

Самое частое применение эвристик - неалгоритмическая оптимизация алгоритма. Эвристикой могут быть такие простые правила как "выбирать всегда первый элемент"или "всегда решать сначала более сложные задачи".

Метаэвристика гораздо более сложное понятие. Это высокоуровневые алгоритмы, которые направляют процесс поиска решений для оптимизационных задач. Они разработаны для работы с широким спектром проблем и часто вдохновлены природными процессами. В отличие от эвристик, которые обычно специфичны для задачи, метаэвристики универсальны и могут адаптироваться к различным сценариям.

Особое место занимают метаэвристики роевого интеллекта, которые имитируют коллективное поведение децентрализованных систем в природе, таких как стаи птиц, колонии муравьёв или косяки рыб. Примеры:

- Оптимизацию роем частиц (PSO) — моделирует движение частиц в пространстве поиска, вдохновлённое поведением стаи.
- Оптимизацию муравьиной колонией [4] (ACO) — основана на том, как муравьи находят кратчайший путь, используя феромоны.
- Алгоритм отжига [6] (SA) — алгоритм вдохновлен процессом отжига при обработке металла.

Эти алгоритмы работают, поддерживая популяцию потенциальных решений и улучшая их итеративно через процессы, напоминающие естественное поведение, такие как движение, коммуникация и адаптация.

[1] Согласно Википедии: https://en.wikipedia.org/wiki/Heuristic

К сожалению метаэвристики позволяют найти только субоптимальные решения задачи, однако этого достаточно для большинства реальных задач, поэтому метаэвристики получили большое распространение в промышленной разработке.

Ключевыми понятиями в метаэвристиках является исследование (exploration) и эксплуатация (exploitation) первое отвечает за поиск новых решений, не похожих на текущее найденное решение, что позволяет таким алгоритмам не застревать в локальном оптимуме. Эксплуатация в свою очередь позволяет алгоритму улучшать найденное хорошее решение. Очень важно, чтобы алгоритм соблюдал баланс между этими двумя процессами.

Гиперэвристики - следующий уровень абстракции. Эти методы работают над выбором, комбинацией различных метаэвристик для эффективного решения конкретной задачи. Гиперэвристики ищут в пространстве метаэвристик, а не в пространстве решений, что сильно отличает их от метаэвристик.

В контексте алгоритмов роевого интеллекта гиперэвристики могут быть использованы для определения, какой метаэвристический подход лучше всего подходит для данной задачи. Гиперэвристики могут быть как очень простыми, как запуск последовательности эвристик и метаэвристик [8], так и сложными, например с применением обучения с подкреплением, как гиперэвристика, которую я буду рассматривать далее.

3 Обзор литературы

Основным источником для моей работы служит статья, которую я как раз и исследую, это статья [3] представленная в 2024 году иранскими и американскими учеными, описывает новый алгоритм поиска субоптимальных решений, который базируется на гиперэвристическом подходе к оптимизации, но при этом сочетает в себе множество других популярных приемов.

Так например авторы в своем алгоритме используют адаптивный подход, основанный на "обучении с подкреплением эта сфера очень активно развивается, как и все сообщество программистов как-либо связанных с машинным обучением.

Недавние тенденции в области гиперэвристик, описанные в обзоре Докероглу и др. [**taxonmy** из 2023 года, подчеркивают интеграцию машинного обучения и оптимизации с несколькими целями, расширяя их применимость. Таксономия включает выборочные гиперэвристики, низкоуровневые эвристики и параллельные подходы, отражая зрелость области за последние 20 лет. Эти разработки предполагают, что рамки обучения с подкреплением WHH могут быть расширены для сценариев с несколькими целями, решая сложные реальные задачи,

такие как устойчивое преобразование, как отмечено в недавних сессиях IEEE CEC.

4 World Hyper-Heuristic (WHH)

В этой главе я подробно опишу принцип работы исследуемого алгоритма, а также приложу иллюстрации приведенные авторами, также опишу структуру кода реализации алгоритма, приведенную авторами статьи.

WHH или мировой алгоритм использует технику обучения с подкреплением для оптимизации выбора алгоритма на каждом шаге, метаэвристики из которых выбирает WHH определяются разработчиком, о том какие алгоритмы выбрали авторы статьи и о своих экспериментов я расскажу в секции экспериментов.

Оптимизация обучением с подкреплением должна помочь избежать ситуации, когда гиперэвристика зацикливается на одной метаэвристике.

Оптимизация состоит из двух шагов - вознаграждения и селекции.

На первом шаге алгоритм делает обоснованное предположение о том, как каждый алгоритм поведет себя в дальнейшем, а затем на этапе селекции WHH выбирает лучший алгоритм о том, как происходит награждение и выбор расскажу поподробнее далее.

Приложу также иллюстрацию из оригинальной статьи которая должа дать базовое понимание устройства WHH 4.1.

В целом работу алгоритма можно разделить на 3 фазы, который проиллюстрированы на рисунке 4.2:

1 Фаза инициализации, или первая итерация

2 Фаза "обучения"

3 Фаза вычисления функции потерь

Во время фазы инициализации генерируется начальная популяция решений, а затем метаэвристики получают свои вознаграждения в зависимости от результатов полученных на этой популяции, на следующей итерации выбор алгоритма будет основан на результатах полученных на этой фазе.

Фаза обучения это основная фаза, выбора следующей метаэвристики, подробное описание этой фазы есть в разделе селекции.

В фазу вычисления, выбранный на предыдущем этапе алгоритм запускается на текущей популяции и вычисляются функции потерь, затем, если не превышено максимальное число итераций, алгоритм возвращается к фазе "обучения".

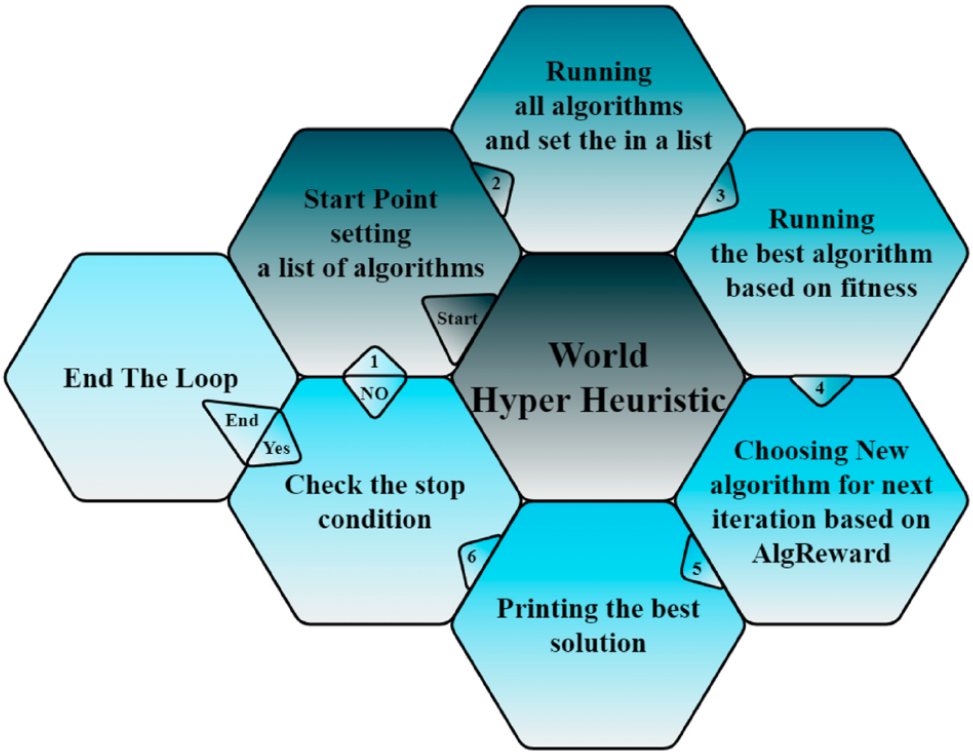

Рис. 4.1: Базовое устройство World Hyper-Heuristic

4.1 Вознаграждение

На первой итерации все алгоритмы начинают с одинаковой случайно сгенерированной популяции решений, и каждая метаэвристика находит свое оптимальное решение.

Так, пусть изначально мы сгенерировали популяцию из m начальных решений, и алгоритм i получил оптмальное значение функции потерь L_i тогда авторы вводят массив (1)

$$gpopCostA = \{L_1, L_2, \ldots, L_n\} \tag{1}$$

, где n - число метаэвристик

А затем получаем награды алгоритмов (2)

$$AlgReward = \frac{1}{gpopCostA} \tag{2}$$

При помощи этих наград наибольший приоритет в поиске решения на первой итерации получит лучший алгоритм

Далее авторы вводят переменную темпа обучения (learning rate) $RLAlpha$ который будет принимать значения в диапазоне [0, 10]

Изначально $RLAlpha = 0$, затем каждую итерацию он будет изменяться согласно (3)

$$RLAlpha = RLAlpha + \frac{1}{maxIter} \tag{3}$$

, где $maxIter$ - максимальное число итераций

Этот момент уже кажется странным, потому что такая имплементация не даст $RLAlpha$ первысить 1.

Как будет видно далее по своей сути $RLAlpha$ вообще сложно назвать темпом обучения, а реальный диапазон значений [0, 1] выбран не случайно.

Далее авторы заявляют, что если результат оптимизации не меняется две итерации подряд, то награда лучшего алгоритма уменьшается как показано в выражении (4)

$$AlgReward_i = AlgReward_i - \frac{\sum_{j=1}^{j \neq i} AlgReward_j}{n} \tag{4}$$

4.2 Селекция

На первой итерации на фазе селекции просто выбирается алгоритм с наибольшией наградой $AlgReward$

В последующих итерациях на фазе селекции генерируется случайное число chN от 0 до 1, затем если $chN > RLAlpha$ то выбор следующей метаэвристики происходит равновероятно, в ином случае для выбора используется RouletteWheelSelection, рандомный выбор, при котором вероятность выбора метаэвристики пропорциональна величине $AlgReward$

Теперь становится ясно, что $RLAlpha$ это скорее вероятность исследования (exploration), чем темп обучения.

Видно, что с каждой итерацией эта вероятность уменьшается, что логично, чем больше итераций прошло, тем мы ближе к оптимальному решению и было бы не так разумно отказываться от него, чем на первых шагах.

5 Анализ результатов приведенных в статье

Авторы статьи провели несколько экспериментов с целью измерения эффективности WHH по сравнению с классическими метаэвристиками.

Сравнения проводились на дискретных и непрерывных NP-сложных задачах.

Так, например были проведены эксперименты на таких дискретных задачах как проблема коммивояжера на 1000 искусственно сгенерированных городах, координаты которых находятся в диапазоне от 0 до 100.

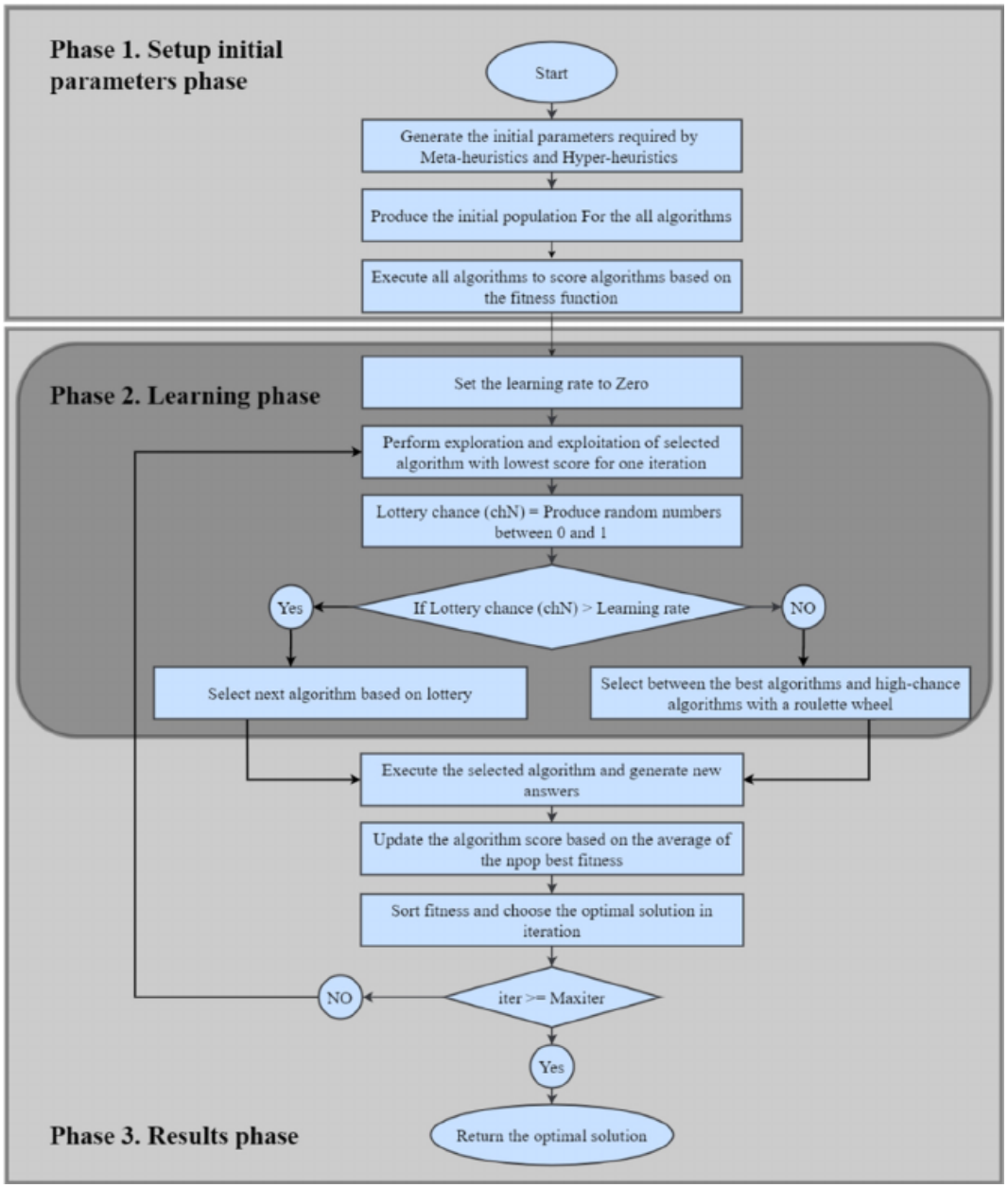

Рис. 4.2: Иллюстрация трех фаз алгоритма

Авторы также сравненивали свой алгоритм с популярными метаэвристиками на датасете из 10000 реальных городов со всего мира [12].

Третьей дискретной задачей была так называемая *COTEJA* [13] (Combinatorial-Optimizatio Based Threat Evaluation and Jamming Allocation) - задача представленная в 2019 году, она заключается в поиске сети радаров для обнаружения воздушных судн.

Непрерывные задачи включали в себя поиск глобального минимума 25 функций (см. Таблицу 8.1)

Также авторы тестировали WHH на реальных проблемах, таких как Welded Beam Design Problem [10] и Speed Reducer Problem [5]

Для дискретных задач авторы использовали набор из 6 метэвристик, их начальные параметры можно увидеть в таблице 8.2, однако в исходном коде отсутствует реализация Grey Wolf Optimization поэтому я буду считать что авторы отказались от этого алгоритма.

Видно что значения взяты одинаковые для всех алгоритмов, авторы утверждают, что это сделано для того, чтобы все метаэвристики были в равных условиях.

Гиперпараметры специфичные для каждого алгоритма выбраны в соответствии с рекомендованными в соответствующих статьях: Imperialist Competitive Algorithm [1], Simulated Annealing [6], Ant Colony Optimization [4], Grey Wolf Optimization [7]

Для непрерывных задач начальные параметры также установлены согласно рекомендациям авторов алгоритмов, значения величины популяции (Population size) и число генераций (Number of generations) для всех алгоритмов одинаковые, всего авторы используют 10 метаэвристик, которые можно увидеть в таблице 8.3

6 Анализ исходного кода авторов и воспроизведение полученных результатов

Заранее стоит сказать, что мною не затронуты возможные ошибки в логике реализации метаэвристик авторами, я сосредоточился на анализе возможных ошибок, допущенных авторами, которые могли бы повлиять на результаты, приведенные в статье.

Код, представленный в статье, вызывает массу вопросов. Отсутствие статической типизации делает его трудным для восприятия. Без явного указания типов данных приходится угадывать, с каким типом данных имеешь дело. Это замедляет процесс анализа и значительно увеличивает вероятность ошибок.

Что касается структуры, то код выглядит перегруженным. Вместо того чтобы делить

программу на логичные, самостоятельные блоки, авторы предпочли оставить все в огромных функциях, которые выполняют множество задач одновременно. Это усложняет восприятие, каждый раз приходится разбирать, что именно происходит в функции. Когда код не разделен на логические части, его становится гораздо труднее отлаживать и тестировать.

В частности авторы используют громадное число различных структур данных которые могут даже дублировать друг друга, имена переменных не следуют никакой логике, где-то камелкейс, где-то нижние подчеркивания, одни переменные с большой буквы другие с маленькой, как будто код писали два брата один на клавиатуре другой на мышке.

К примеру в начале основного скрипта авторы вводят частично определенную функцию, однако периодически забывают о ней в процессе и все равно передают зафиксированные параметры.

Вероятно такой подход привел к тому, что сами авторы допустили некоторые критические ошибки.

Так, например в коде дискретного алгоритма WHH в фазе инициализации авторы объявляют 5 используемых метаэвристик, однако запускают только 3, причем награда третьего алгоритма записывается в индекс, в которм уже лежит награда первого, это существенно влияет на всю фазу инициализации – вместо 5 алгоритмов запускаются только 2, что может сильно влиять на начальные данные для второй фазы. Что еще более забавно, в статье авторы говорят о 6 алгоритмах, в коде объявляют 5 из них, реализованы лишь 3, а записаны результаты только для двух алгоритмов.

Также в коде приложенном авторами полностью отсутствует реализация большинства метаэвристик, который они, как утверждается использовали.

Аналогично, невозможно воспроизвести эксперименты с реальными данными в виду отсутствия реализации взаимодействия я с ними.

Некоторые гиперпараметры переопределяются несколько раз, наример авторы вводят переменную *alpha* сначала как коэффициент остывания для алгоритма отжига, а затем тут же переопределяют ее как коэффициент *alpha* в ACO, где он вообще принят целым числом равным 1, что недопустимо для алгоритма отжига, однако авторы избегают ошибок связанных с этим, так как просто понижают температуру не в том месте, где это должно просиходить.

Очень странное также решение авторов в фазе "обучения"они выбирают лучший алгоритм, а затем выбирают случайный алгоритм из Tabu и ICA.

Становися очевидно, что авторами был приложен мало того, что не полный код, который использовался для проведения измерений, этот код содержит множество ошибок и

я бы назвал чудом, то что он вообще запускается – малейшее изменение гиперпараметров, например размерности входных данных приводит к ошибкам и код перестает работать, есть ощущение, что был приложен черновик, а не исходный код к статье.

В связи с этим я переписал код с матлабовских скриптов на питон, а также исправил ошибки (какие нашел) допущенные авторами, добавил статическую типизацию и улушил структуру проекта, для большей читаемости кода, а также реализовал метаэвристики, которых не было в коде авторов.

7 Моя реализация кода, описанного в статье и сравнение результатов

Как и описано в статье я использовал для дискретных задач 6 алгоритмов, их начальные параметры.

Для лушчей орагнизации кода я добавил статичную типизацию (насколько это возможно на питоне), а также разделил код на связанные по смыслу классы.

Мною был введен класс Solution, содержащий два поля: Position и Cost – первое олицетворяет решение задачи, второе значение функции потерь для этого решения.

Авторы использовали похожую структуру данных $empty_individual$, моя реализация сделала код сильно более читаемым, также в классе я объявил стандартный конструктор, использование которого сильно упростило код, а также оператор сравнения, он был необходим для легкой имплементации сортировок.

Каждая метаэвристика также получила свой класс, содержащий методы:

- Конструктор - отвечает за инициализацию гиперпараметров метаэвристики
- Метод initialize - основной метод, который принимает текущую популяциб и функцию потерь, изменяет популяцию в процессе и возвращает лучшее найденное решение (Solution) в контексте переданной функции потерь
- Вспомогательные методы, необходимые для работы алгоритма, например создание соседей, обновление позиций или проверку корректности

Такая структура проекта сильно упростила читаемость и воспринимаемость WHH, все метаэвристики мне пришлось реализовывать самому, поэтому насколько качественное это сравнение с реализациями авторов статьи не могу.

Также я полностью переписал основной файл WorldAlg.py под новую структуру проекта, теперь код выглядит последовательным и ясным.

В своей реализации я использовал скрипт run.py для запуска гиперэвристики и построения графиков, далее расскажу о них поподробне.

Я провел 10 экспериментов с сгенерированной случайной выборкой точек с целочисленными координатами в двумерном пространстве, результаты можно увидеть на графике 8.2

Также я построил график с усредненными результатами для каждой итерации 8.3 можем легко сравнить его с тем, который авторы приводят в своей статье для этой задачи 8.1, конечно они вычислительные мощности у них боли существенно больше, поэтому провести 1000 экспериментов я не смог, но и 10 достаточно, чтобы заметить схожесть вида графиков, однако алгоритм переписанный мной получил ощутимо лучшие результаты.

8 Заключение

В заключение хочется сказать, что несмотря на ужасный код, приложенный авторами под видом исходного, сам алгоритм при должном подходе код становится понятным и масштабируемым, в дальшейшем можно произвести любое число модификаций моей реализации и добиться еще более впечатляющих результатов.

В дальнейшем я не планирую забрасывать существующий проект и планирую развивать его дальше, в близжайших планах реализовать свою версию для непрерывного алгоритма. Также я хочу улучшить структуру кода, добавить больше абстракций, расширяя просторы для расширения функционала. Не помешает также создать удобный интерфейс (хотя бы скрипт) для тюнинга параметров и, например загрузки пользовательской задачи оптимизации.

Как видно, современные ученые сильно продвинулись в области поиска субоптимальных решений NP-сложных задач, что не может не радовать. Надеюсь авторы не забросят свои наработки, и продолжат развивать эту идею, например можно внедрить реальное машинное обучение, а еще надеюсь, что они все-таки опубликуют оригинальный код.

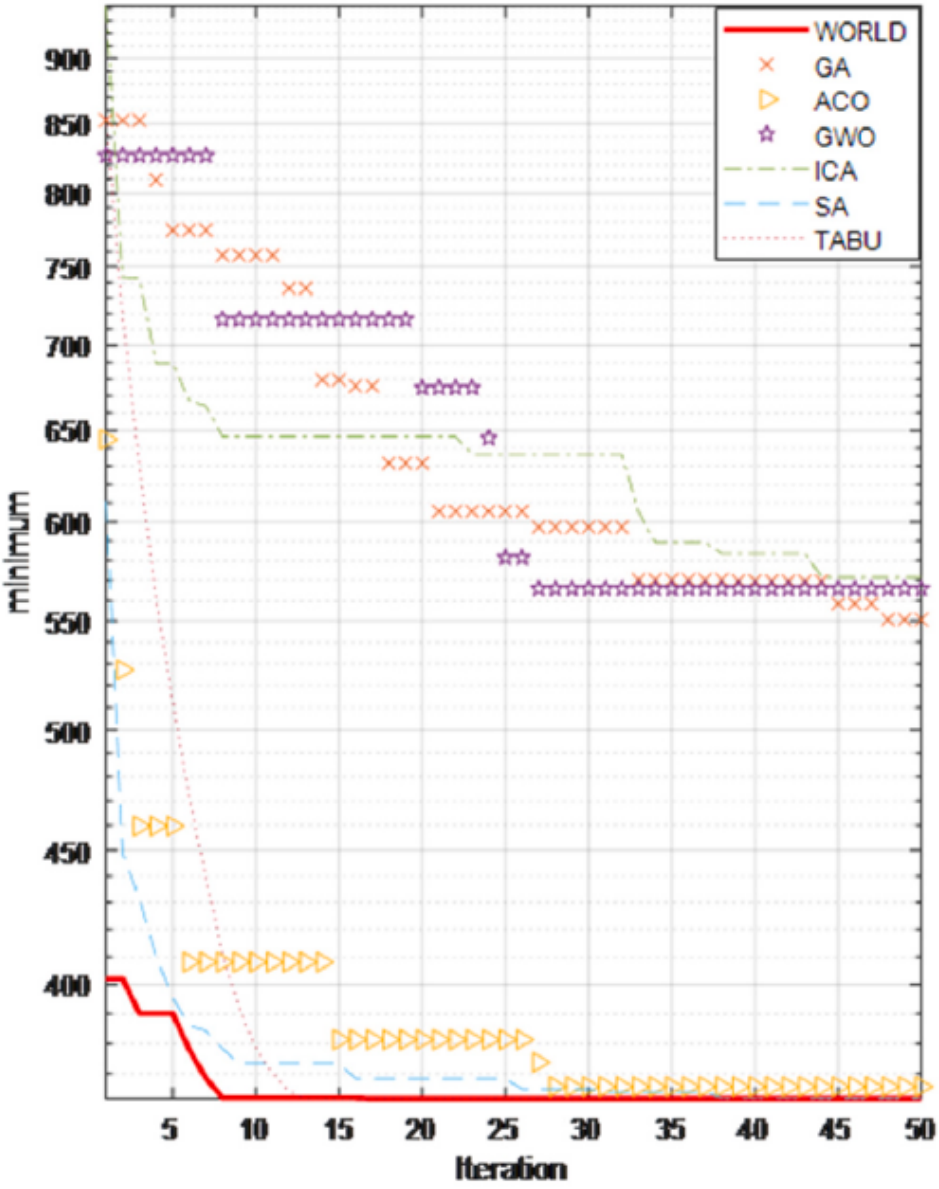

Рис. 8.1: График результатов, полученных авторами статьи

Список литературы

[1] E. Atashpaz-Gargari и C. Lucas. “An Algorithm for Optimization Inspired by Imperialistic Competition”. В: *In 2007 IEEE Congress on Evolutionary Computation* (2007), с. 4661—5467.

[2] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Özcan и J. R. Woodward. “A Classification of Hyper-Heuristic Approaches: Revisited”. В: *Handbook of Metaheuristics*. Springer, 2019, с. 453—477.

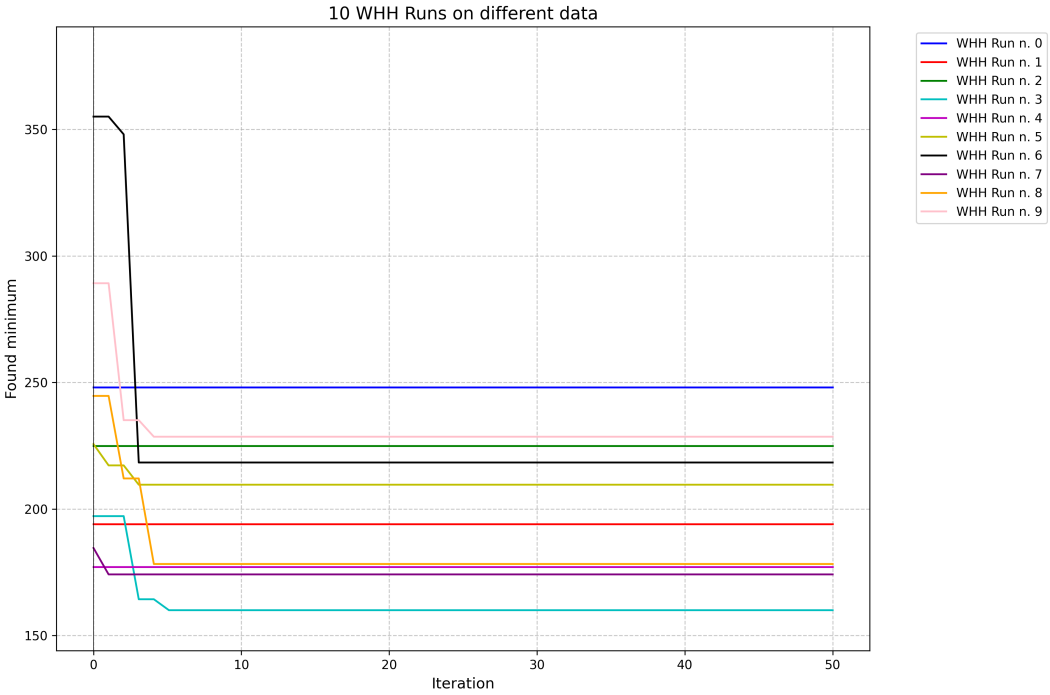

Рис. 8.2: График минимального значения, найденного алгоритмом на каждой итерации

[3] Arman Daliri, Mahmoud Alimoradi, Mahdieh Zabihimayvan и Reza Sadeghi. “World Hyper-Heuristic: A novel reinforcement learning approach for dynamic exploration and exploitation”. В: *Expert Systems with Applications* 244 (2024).

[4] Marco Dorigo, Mauro Birattari и Thomas Stützle. “Ant Colony Optimization”. В: *IEEE Computational Intelligence Magazine* 1 (2006), с. 28—39.

[5] Hassan, Rania, Babak Cohanim, Olivier De Weck и Gerhard Venter. “A Comparison of Particle Swarm Optimization and the Genetic Algorithm.” В: *AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials* 46 (2005).

[6] S. Kirkpatrick, Jr C. D. Gelatt и M. P. Vecchi. “Optimization by Simulated Annealing”. В: *Science* 220 (1983), с. 671—680.

[7] S. Mirjalili, S. M. Mirjalili и A. Lewis. “Grey Wolf Optimizer”. В: *Advances in Engineering Software* 69 (2014), с. 46—61.

[8] R. F. Thompson и W. I. Welker. “Role of auditory cortex in reflex head orientation by cats to auditory stimuli”. В: *Journal of Comparative and Physiological Psychology* 56.6 (1963), с. 996.

[9] V. Trivedi, P. Varshney и M. Ramteke. “A simplified multi-objective particle swarm optimization algorithm”. В: *Springer Nature* 14 (2020), с. 83—116.

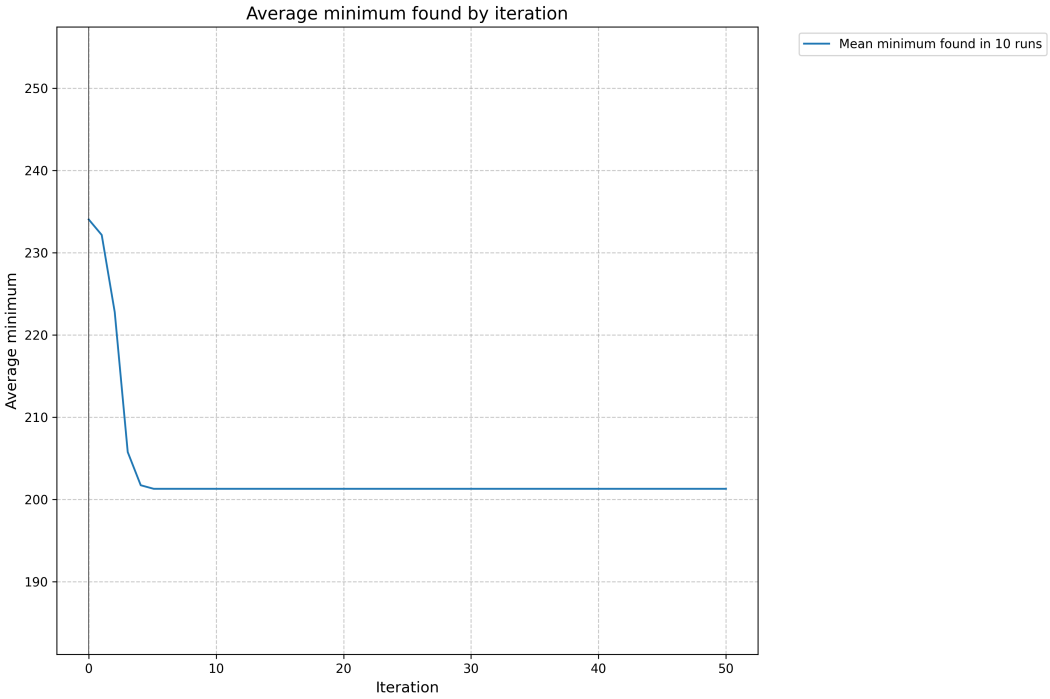

Рис. 8.3: График среднего минимального значения, найденного алгоритмом на каждой итерации

[10] Wang, Weijun, Stéphane Caro, Fouad Bennis и Oscar Brito Augusto. “Toward the Use of Pareto Performance Solutions and Pareto Robustness Solutions for Multi-Objective Robust Optimization Problems.” В: *Engineering Systems Design and Analysis* 5.44861 (2012), с. 41—50.

[11] D. H. Wolpert и W. G. Macready. “No Free Lunch Theorems for Optimization”. В: *IEEE Transactions on Evolutionary Computation* 1.1 (1997), с. 67—82.

[12] *World Traveling Salesman Problem*. URL: http://www.math.uwaterloo.ca/tsp/world/index.html (дата обр. 04.10.2020).

[13] Shixun You, Ming Diao и Lipeng Gao. “"Implementation of a combinatorial-optimisation-based threat evaluation and jamming allocation system"”. В: *"IET Radar, Sonar & Navigation"* 13.10 (2019), с. 1636—1645.

Таблица 8.1: Функции, на которых проводились эксперименты авторами статьи. В столбце $Fmin$ указан глобальный минимум

Название функции	Формулы	Fmin
Ackley	$f(x) = f(x_1, \ldots, x_n) = -a \cdot \exp\left(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(cx_i)\right) + a + \exp(1)$	0
Brown	$f(x) = \sum_{i=1}^{n-1} (x_i)^{x_{i+1}^2+1} + (x_{i+1}^2)^{x_i^2+1}$	0
Exponential	$f(x) = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right)$	-1
Griewank	$f(x) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$	0
Periodic	$f(x) = 1 + \sum_{i=1}^{n} \sin^2(x_i) - 0.1\exp\left(\sum_{i=1}^{n} x_i^2\right)$	0.9
Powell Sum	$f(x) = \sum_{i=1}^{n} \lvert x_i\rvert^{i+1}$	0
Qing	$f(x) = \sum_{i=1}^{n} (x_i^2 - i)^2$	0
Quartic	$f(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	0
Rastrigin	$f(x) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$	0
Ridge	$f(x) = x_1 + d\left(\sum_{i=2}^{n} x_i^2\right)^{\alpha}$	-10
Rosenbrock	$f(x) = \sum_{i=1}^{n-1} b(x_{i+1} - x_i^2)^2 + (a - x_i)^2$	0
Salomon	$f(x) = 1 - \cos\left(2\pi\sqrt{\sum_{i=1}^{n} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{n} x_i^2}$	0
Schwefel 2.20	$f(x) = \sum_{i=1}^{n} \lvert x_i\rvert$	0
Schwefel 2.21	$f(x) = \max \lvert x_i\rvert$	0
Schwefel 2.22	$f(x) = \sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	0
Schwefel 2.23	$f(x) = \sum_{i=1}^{n} x_i^{10}$	0
shubert3	$f(x) = \sum_{i=1}^{n} \sum_{j=1}^{5} j\sin((j+1)x_i + j)$	-101
Sphere	$f(x) = \sum_{i=1}^{n} x_i^2$	0
Styblinski-Tank	$f(x) = \frac{1}{2}\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i$	-200
Sum Squares	$f(x) = \sum_{i=1}^{n} i x_i^2$	0
Xin-She Yang	$f(x) = \sum_{i=1}^{n} \in_i \lvert x_i\rvert^i$	0
Xin-She Yang N. 2	$f(x) = (\sum_{i=1}^{n} \lvert x_i\rvert)\exp\left(\sum_{i=1}^{n} \sin(x_i^2)\right)$	0
Xin-She Yang N. 3	$f(x) = \exp\left(-\sum_{i=1}^{n}\left(\frac{x_i}{\beta}\right)^{2m} - 2\exp\left(-\sum_{i=1}^{n} x_i^2\right)\prod_{i=1}^{n}\cos^2(x_i)\right)$	0
Xin-She Yang N. 4	$f(x) = \left(\sum_{i=1}^{n}\sin^2(x_i) - \exp\left(-\sum_{i=1}^{n} x_i^2\right)\right)\exp\left(-\sum_{i=1}^{n}\sin^2\sqrt{\lvert x_i\rvert}\right)$	-1
Zakharov	$f(x) = \sum_{i=1}^{n} x_i^2 + \left(0.5\sum_{i=1}^{n} i x_i\right)^2 + \left(0.5\sum_{i=1}^{n} i x_i\right)^4$	0

Таблица 8.2: Начальные параметры метаэвристик для дискретных задач

Алгоритм	Параметры	Значения
Ant Colony Optimization (ACO)	Population Size	50
	Number of generations	100
	Conversion ratio fitness	100
Genetic Algorithm (GA)	Population Size	50
	Number of generations of 10,000 cities	1000
	Number of generations for others	100
Grey Wolf Optimization (GWO)	Control Parameter	[0, 2]
	Number of generations of 10,000 cities	1000
	Number of generations for others	100
	Number of particles	50
Imperialist Competitive Algorithm (ICA)	Number of countries	50
	Number of generations of 10,000 cities	1000
	Number of generations for other	100
	Number of nimp	10
Simulated Annealing (SA)	Population Size	50
	Number of generations of 10,000 cities	1000
	Number of generations for other	100
	Number of neigbors	10
Tabu (Taboo)	Population Size	50
	Number of generations of 10,000 cities	1000
	Number of generations for other	100

Таблица 8.3: Начальные параметры метаэвристик для непрерывных задач

Алгоритм	Параметры	Значения
Adaptive differential evolution algorithm (ADEA)	Population Size	50
	Number of generations	1000
Artificial bee colony algorithm (ABC)	Population Size	50
	N Number of generations	1000
Biogeography Optimization (BO)	Alpha	0.9
	Number of generations	1000
	Keep Rate	0.4
Differential Evolution (DE)	Beta min	0.2
	Beta max	0.8
	PCR	0.2
	Number of generations	1000
Firefly Algorithm (FA)	Gamma	1
	Beta	2
	Number of generations	1000
Genetic Algorithm (GA)	Population Size	50
	Number of generations	1000
	Gamma	0.4
Harmony Search (HS)	Number of nimp	10
	Neigboring value rate	0.3
	Discrete set and fret	17700,1
	Width	1000
	Number of generations	
Improved Invasive Weed Optimization (IWO)	Exponent	2
	Sigma initial	0.5
	Sigma final	0.001
	Number of generations	1000
Particle Swarm Optimization (PSO)	Inertia coefficient	0.75
	Cognitive & social coeff	[1.8, 2]
	Number of generations	1000
	Search agents	100
Self-adaptive Differential Evolution Algorithm (SADE)	Population size	50
	Number of generations	1000
Shuffled Frog-Leaping Algorithm (SFLA)	Memeplex	10
	Alpha	3
	Beta	5
	Number of generations	1000
Simulated Annealing (SA)	Alpha	0.99
	Mu	0.5
	Number of generations	1000
Single objective real-parameter optimization (jSO)	Population Size	50
	Number of generations	1000
Teaching-Learning Optimization Algorithm (TLBO)	Population Size	50
	Number of generations	1000