САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики-процессов управления

Программа бакалавриата
“Большие данные и распределенная цифровая платформа”

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Алгоритмы и структуры данных»

на тему «Исследование генетического алгоритма. Изучение различных кодировок генотипа»

Вариант – 17

Студент гр. 23Б15-пу
Сериков К.Г.

Преподаватель
Дик А.Г.

Санкт-Петербург
2024 г.

Оглавление

Цель работы

Целью работы является изучение эффективности различных способов кодирования генотипа хромосом в генетическом алгоритме на примере задачи минимизации функции. В рамках данной работы была разработана программа, которая находит минимум функции при помощи генетического алгоритма и предоставляет возможность переключения между обычной и модифицированной версиями алгоритма, а также между вещественным и целочисленным типами кодирования.

Описание задачи (формализация задачи)

В данной лабораторной работе необходимо исследовать эффективность генетического алгоритма в решении задачи оптимизации, а именно — в минимизации функции: x1^3 + x2^2 - 3 * x1 - 2 * x2 + 2.

Формализация задачи включает следующие компоненты:

1. **Изучение кодирования генотипов**: Исследование различных методов кодирования генетических алгоритмов, таких как вещественное и целочисленное кодирование. Рассмотрение их преимуществ и недостатков для данной задачи оптимизации.
2. **Разработка алгоритма для минимизации функции**: Написание программы, реализующей генетический алгоритм для поиска минимума функции. В программу заложена возможность выбора способа кодирования генотипов, что позволяет адаптировать алгоритм к различным условиям.
3. **Анализ эффективности алгоритма**: Сравнительный анализ результатов работы генетического алгоритма при различных параметрах. Определение влияния типа кодирования на скорость и точность нахождения минимума, а также на общее количество итераций и вычислений, необходимых для достижения оптимального результата.

Теоретическая часть

Генетические алгоритмы

Генетические алгоритмы (ГА) являются методами оптимизации, которые используют принципы генетики и естественного отбора, чтобы находить наилучшие решения для сложных задач. Эти алгоритмы имитируют процессы, происходящие в природе, такие как мутация, кроссовер и селекция, чтобы улучшать популяцию решений с каждым поколением. Основная цель заключается в том, чтобы с каждым новым поколением повышать среднюю «приспособленность» популяции, направляя её к глобальному оптимуму или его приближению.

Генетические алгоритмы начинают с создания популяции, состоящей из множества возможных решений, представленных в форме «хромосом». Каждая хромосома содержит информацию о возможном решении задачи и оценивается по функции приспособленности, которая определяет, насколько данное решение подходит для поставленной задачи.

Методы кодирования генотипа

Кодирование генотипа играет ключевую роль в эффективности генетического алгоритма, поскольку именно оно определяет форму представления решений задачи. Существуют различные подходы к кодированию.

1. **Вещественное кодирование:** Вещественное кодирование представляет хромосомы в виде чисел с плавающей точкой, что позволяет работать с более плавными и непрерывными пространствами поиска. Это важно для задач, где требуется высокая точность, поскольку вещественные значения обеспечивают более гибкое и точное представление генов.
2. **Целочисленное кодирование:** При целочисленном кодировании каждый ген представлен целым числом. Этот подход применяется для задач, где значение гена должно быть ограничено целыми числами. Целочисленное кодирование зачастую быстрее, поскольку пространство поиска ограничено, но при этом решения могут быть менее точными по сравнению с вещественным представлением.

Оценка эффективности генетических алгоритмов

Эффективность генетического алгоритма оценивается его способностью находить оптимальные или близкие к оптимальным решения за приемлемое время. Основные параметры эффективности включают:

1. **Скорость сходимости:** Быстрота, с которой алгоритм достигает заданного уровня качества решения.
2. **Качество решения:** Насколько близко полученное решение к истинному оптимуму.

Основные шаги программы

1. **Запуск программы:** Инициализируется графический интерфейс с элементами управления.
2. **Настройка параметров:** Пользователь может задать значения параметров алгоритма в интерфейсе, такие как вероятность мутации, количество хромосом, минимальные и максимальные значения генов и количество поколений. Пользователь выбирает между стандартным и модифицированным режимами алгоритма, а также между вещественной и целочисленной кодировками.
3. **Запуск алгоритма:** При нажатии на кнопку "Рассчитать" программа запускает выполнение генетического алгоритма, используя заданные параметры.
4. **Инициализация популяции:** На основе выбранного типа кодирования (вещественный или целочисленный) создаётся начальная популяция. Каждый хромосома включает два гена (x1, x2), которые генерируются случайно в пределах заданного диапазона.
5. **Оценка приспособленности:** Для каждой хромосомы рассчитывается значение функции приспособленности с помощью функции fitness. Эта функция измеряет "приспособленность" каждой хромосомы к решению задачи.
6. **Сегментация популяции:** Популяция делится на сегменты по четыре индивида, которые используются для выполнения отбора и создания новых потомков.
7. **Скрещивание и мутация:** Для каждой пары родителей выполняется операция кроссинговера, затем происходит мутация с заданной вероятностью..
8. **Создание нового поколения:** Из потомков формируется новое поколение. Если фитнес нового поколения превышает предыдущее лучшее решение, обновляется глобальное лучшее решение.
9. **Отображение результатов:** На экране отображается количество пройденных поколений, текущее лучшее решение (значения x1, x2), значение функции и общее количество пройденных поколений, а также таблица, в которой выводятся хромосомы последнего поколения с их индексами, значениями генов и значением функции приспособленности.

Блок схема программы

Рис 1. Блок-схема основной программы

Описание программы

Программная реализация написана на языке Python 3.13.0 с использованием следующих библиотек: random [3] и tkinter [4]. Программа организована в виде графического интерфейса для решения задач оптимизации, выполняемых с помощью генетического алгоритма. В процессе разработки программы использовался main.py, включающий 11 функций, каждая из которых имеет чётко определённое назначение:

Таблица 1. main.py

Функция	Описание	Возвращаемое значение
function	Определение функции.	float
create_initial_population	Создание начальной популяции с параметрами, заданными пользователем.	list of tuples
fitness	Оценка приспособленности на основе значения функции.	float
segment_population	Разделение популяции на сегменты по четыре для применения ранжированной селекции.	list of lists
ranking_selection	Выбор лучших хромосом в сегменте для участия в создании потомков.	list
crossover	Реализация операции кроссинговера для создания новых хромосом.	tuple
mutate	Применение мутации к хромосоме с учетом вероятности мутации и пределов значений.	tuple
create_new_generation	Создание нового поколения с использованием модифицированного режима скрещивания и мутации.	list
create_new_generation_standard	Создание нового поколения с использованием стандартного режима для кроссинговера и мутации.	list
toggle_mode	Переключение режима работы и обновление отображаемого статуса режима.	None

run_algorithm	Запуск генетического алгоритма и обновление интерфейса для вывода текущих лучших решений.	None

Рекомендации пользователя

Программа позволяет запустить генетический алгоритм для оптимизации функции x1^3 + x2^2 - 3 * x1 - 2 * x2 + 2 с использованием графического интерфейса.

1. **Запуск программы**: Перед началом работы с генетическим алгоритмом откройте программу, инициализировав графический интерфейс с помощью Python 3.13.0. Откроется окно с настройками и элементами управления для ввода параметров. Задайте параметры алгоритма, такие как вероятность мутации (%), количество хромосом, минимальное и максимальное значения генов, число поколений, а также выберите между стандартным и модифицированным режимами генетического алгоритма.
2. **Запуск алгоритма**: Нажмите кнопку «Рассчитать» для выполнения генетического алгоритма. Программа начнет расчет и отобразит лучшие результаты.
3. **Просмотр результатов**: Результаты текущего поколения отображаются в таблице, где можно увидеть номер хромосомы, значение функции и значения генов x1 и x2.
4. **Оценка итогов**: Внизу интерфейса отображается лучший найденный результат, включающий значения x1, x2 и итоговое значение функции, соответствующее этим параметрам.

Рекомендации программиста

Убедитесь, что программа работает с последней стабильной версией Python и необходимых библиотек, таких как Tkinter, чтобы гарантировать совместимость с современными системами и обеспечение их функциональности. Уделяйте внимание четкому именованию переменных и функций. Регулярно проводите тестирование программы на различных входных данных, чтобы убедиться в её надежности и корректности. Убедитесь, что интерфейс программы в Tkinter легко понимается пользователями. Разделите входные данные, управление алгоритмом и отображение результатов по разным секциям.

Исходный код программы

https://github.com/romplle/spbu-algorithms-and-data-structures/

Контрольный пример

1. Запуск программы и ввод параметров

   Для запуска программы откройте main.py. Программа откроет графический интерфейс для настройки и запуска генетического алгоритма, предназначенного для минимизации функции x1^3 + x2^2 - 3 * x1 - 2 * x2 + 2.

2. Ввод начальных параметров

   После запуска программы отобразится окно с полями ввода, где можно задать параметры генетического алгоритма (Рис. 2).

   Введите значения для следующих параметров: вероятность мутации (%), Количество хромосом, минимальное и максимальное значения генов, количество поколений, тип кодировки и режим работы.

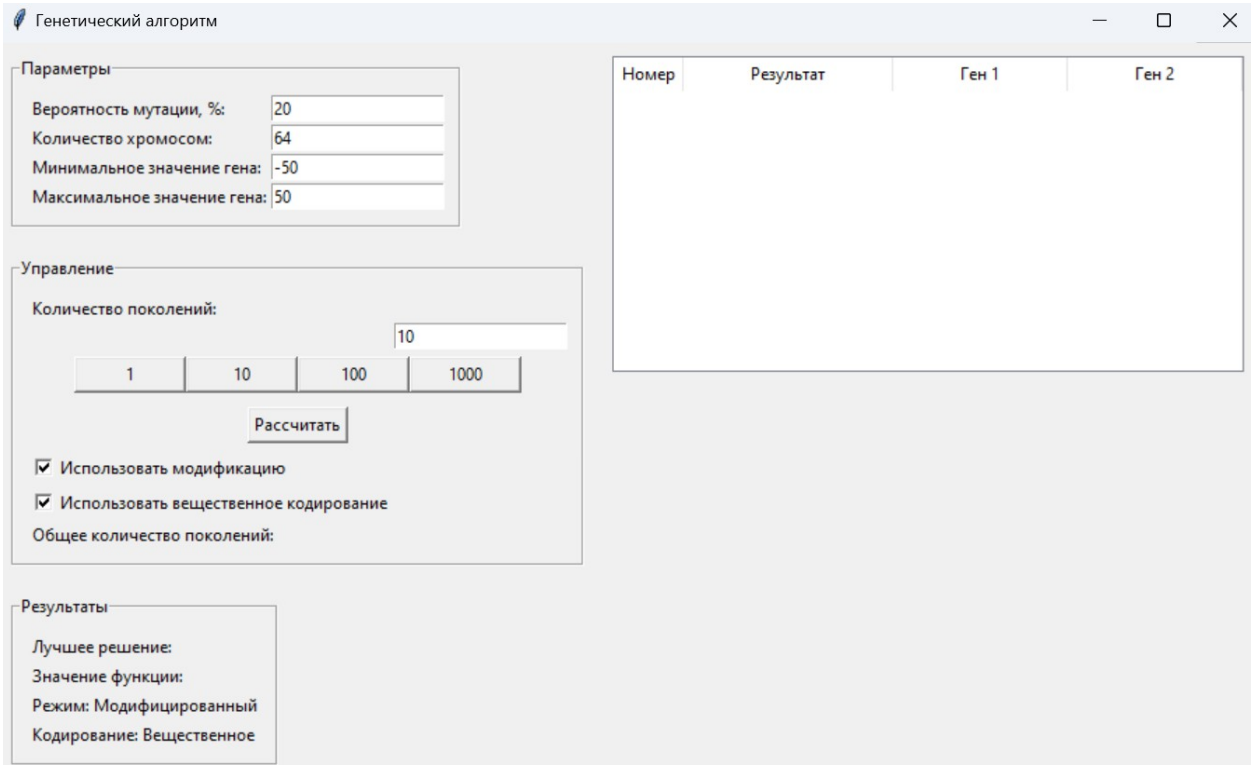

Рис. 2 Пример окна программы

3. Запуск алгоритма

   После задания всех параметров нажмите кнопку «Рассчитать», чтобы запустить алгоритм. Программа выполнит заданное количество поколений, создавая новое поколение на каждом этапе и обновляя интерфейс для отображения текущих результатов (Рис. 3).

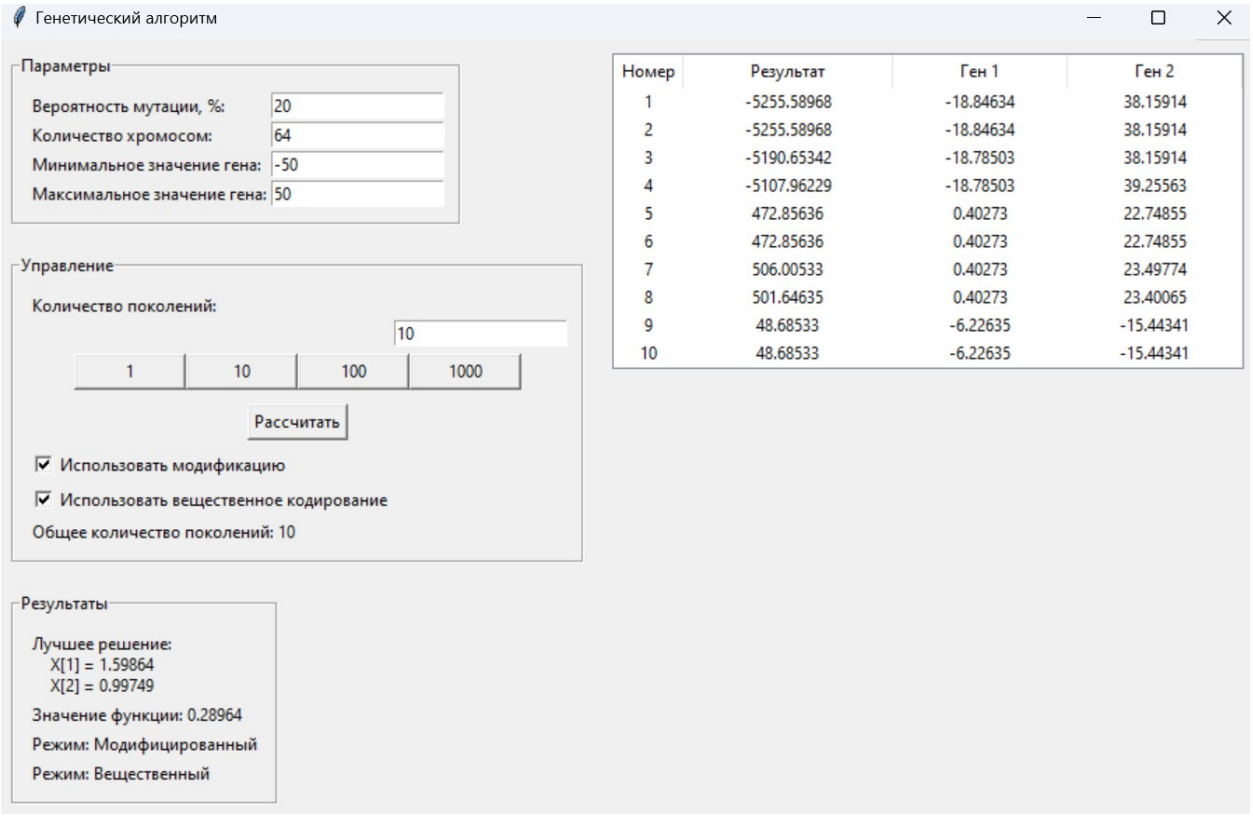

Рис. 3 Пример результатов программы

Анализ

Тесты проводились при следующих настройках: вероятность мутации 20%, минимальное значение гена -50, максимальное значение гена 50. Ниже приведены результаты для различных сочетаний параметров и режимов, с подробным анализом полученных данных.

Вещественный геном с модификацией:

Таблица 2: тесты вещественного кодирования с модификацией для 32 хромосом

Количество поколений	Количество хромосом	Лучшее решение	Значение функции
10	32	(-2.16506, -1.05824.)	1.58285
100	32	(-4.21906, -6.83680)	-0.02842
500	32	(0.99996, 1.00003)	-1.00001

Таблица 3: тесты вещественного кодирования с модификацией для 64 хромосом

Количество поколений	Количество хромосом	Лучшее решение	Значение функции
10	64	(1.23381, 0.95224)	-0.82094
100	64	(0.98584, 0.89121)	-0.98757
500	64	(1.00002, 1.00001)	-1.00001

Вещественный геном без модификации:

Таблица 4: тесты вещественный кодирования без модификации для 32 хромосом

Количество поколений	Количество хромосом	Лучшее решение	Значение функции
10	32	(-1.86453, -10.59788)	134.62237
100	32	(-6.44511, -16.17656)	47.64342
500	32	(-6.05397, -13.32659)	2.53160

Таблица 5: тесты вещественный кодирования без модификации для 64 хромосом

Количество поколений	Количество хромосом	Лучшее решение	Значение функции
10	64	(-6.24147, 16.19784)	7.55591
100	64	(5.01069, -9.49878)	0.43314
500	64	(0.97048, 0.85763)	-0.97714

Целочисленный геном с модификацией:

Таблица 6: тесты целочисленный генома с модификацией для 32 хромосом

Количество поколений	Количество хромосом	Лучшее решение	Значение функции
10	32	(0, -15)	257
100	32	(4, 1)	53
500	32	(1, 1)	-1

Таблица 7: тесты целочисленный генома с модификацией для 64 хромосом

Количество поколений	Количество хромосом	Лучшее решение	Значение функции
10	64	(-2, -4)	29
100	64	(2, -1)	10
500	64	(1, 1)	-1

Целочисленный геном без модификации:

Таблица 8: тесты целочисленный генома без модификации для 32 хромосом

Количество поколений	Количество хромосом	Лучшее решение	Значение функции
10	32	(1, -2)	13
100	32	(-4, -7)	6
500	32	(-2, 0)	0

Таблица 9: тесты целочисленный генома без модификации для 64 хромосом

Количество поколений	Количество хромосом	Лучшее решение	Значение функции
10	64	(-5, -11)	16
100	64	(-9, -25)	1
500	64	(-8, -11)	-1

Вещественный геном с модификацией

При увеличении количества поколений результаты становятся все ближе к оптимальному значению функции, стремящемуся к нулю. Использование модификации положительно сказывается на скорости сходимости к лучшему решению.

Вещественный геном без модификации

В таблицах с тестами видно, что значения функции в наилучших найденных решениях значительно выше, чем в тестах с модификацией, что указывает на недостаточную эффективность алгоритма без модификации. Алгоритм без модификации показал менее удовлетворительные результаты и большую рассеянность решений.

Целочисленный геном с модификацией

Модифицированный алгоритм с целочисленным кодированием достигает значений функции близких к оптимальным уже при меньшем числе поколений. Модифицированный алгоритм с целочисленным кодированием показывает хорошую точность и приближение к глобальному минимуму.

Целочисленный геном без модификации

Алгоритм в целочисленном режиме без модификации в среднем показал удовлетворительные результаты, но не такие стабильные, как в модифицированной версии. Без модификации целочисленный режим также способен находить неплохие решения, но для достижения наилучших результатов требуется больше поколений.

Общие наблюдения

Большее число хромосом способствует нахождению более качественных решений во всех режимах. Это связано с тем, что увеличенная популяция повышает вероятность нахождения решения, близкого к глобальному

минимуму. Алгоритм с модификацией показал заметное преимущество над алгоритмом без модификации, что свидетельствует о необходимости использования дополнительных улучшений в генетическом алгоритме для повышения его продуктивности и точности.

Вывод

В ходе выполнения работы были изучены особенности кодирования генотипа и их влияние на эффективность алгоритма. Модификация генетического алгоритма, включающая улучшенный отбор и кроссинговер, привела к более быстрому нахождению минимума функции. Программа с графическим интерфейсом позволяет наглядно видеть влияние различных параметров на результат работы алгоритма, что делает её удобной для дальнейших исследований.

Источники

1. Genetic algorithm // URL: https://en.wikipedia.org/wiki/Genetic_algorithm (дата обращения: 06.11.2024).
2. Генетический алгоритм на Python // URL: https://habr.com/en/articles/498308/ (дата обращения: 06.11.2024).
3. random— Generate pseudo-random numbers // URL: https://docs.python.org/3/library/random.html (дата обращения: 06.11.2024).
4. tkinter — Python interface to Tcl/Tk // URL: https://docs.python.org/3/library/tkinter.html (дата обращения: 07.11.2024).
5. tkinter.ttk — Tk themed widgets // URL: https://docs.python.org/3/library/tkinter.ttk.html (дата обращения: 07.11.2024).