Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»

Институт Уральский энергетический

Кафедра «Электротехника»

ДОПУСТИТЬ К ЗАЩИТЕ ПЕРЕД ГЭК

Зав. кафедрой электротехники
_____ В.Э.Фризен

«_13_» ___июня_____2024 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РАЗРАБОТКА АЛГОРИТМОВ ОПТИМИЗАЦИИ РАЗМЕЩЕНИЯ КОМПЕНСИРУЮЩИХ УСТРОЙСТВ В ЭЛЕКТРИЧЕСКОЙ СЕТИ НА ОСНОВЕ АДАПТИВНЫХ АЛГОРИТМОВ РОЕВОГО ИНТЕЛЛЕКТА

Научный руководитель: Матренин Павел Викторович, _____
к.т.н., в.н.с научной лаборатории ЦДЭЭ

Нормоконтролер: Хальясмаа Александра Ильмаровна, _____
к.т.н

Студент группы ЭНМ-221201 Тронин Артем Юрьевич _____

Екатеринбург
2024

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования
«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина»

Институт *Уральский энергетический институт*
Кафедра *Электротехники*
Направление (специальность) *01.04.04 – Прикладная математика*
Образовательная программа *Искусственный интеллект в электроэнергетике*

УТВЕРЖДАЮ
Зав. кафедрой В.Э. Фризен
«15» января 2024 г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

студента *Тронина Артема Юрьевича* **группы** *ЭНМ-221201*
(фамилия, имя, отчество)

1. Тема выпускной квалификационной работы

Разработка алгоритмов оптимизации размещения компенсирующих устройств в электрической сети на основе адаптивных алгоритмов роевого интеллекта
Утверждена распоряжением по институту от «10» января 2024 г. № 33.07-03/003
2. Руководитель Матренин Павел Викторович, к.т.н, в.н.с. лаборатории ЦДЭЭ
(Ф.И.О., должность, ученое звание, ученая степень)

3. Исходные данные к работе *Схема электрической сети 220 кВ*
4. Перечень демонстрационных материалов Презентация (15 слайдов)
5. Календарный план

№ п/п	Наименование этапов выполнения работы	Срок выполнения этапов работы	Отметка о выполнении
1.	*Обзор существующих методов оптимального размещения компенсирующих устройств в электрической сети*	до 1 марта 2024 г.	
2.	*Анализ методов многокритериальной оптимизации применительно к задаче размещения компенсирующих устройств*	до 1 апреля 2024 г.	
3.	*Разработка адаптивных алгоритмов роевого интеллекта для решения задач многокритериальной оптимизации*	до 1 мая 2024 г.	

Руководитель _____ *П.В. Матренин*
(подпись) (Ф.И.О.)
Задание принял к исполнению 15.01.2024 _____
дата (подпись)

6. Выпускная квалификационная работа закончена «6» июня 2024 г. считаю возможным допустить Тронина Артема Юрьевича к защите магистерской диссертации в Государственной экзаменационной комиссии (протокол заседания кафедры № __ от «__» июня 202_ г.).
Руководитель_____ Матренин П. В
(подпись) Ф.И.О.

7. Допустить Тронина Артема Юрьевича к защите магистерской диссертации в Государственной экзаменационной комиссии (протокол заседания кафедры №___ от «___» _____ 202__г.).
Зав. кафедрой _____ В.Э Фризен
(подпись) Ф.И.О.

РЕФЕРАТ

Отчет 65 страницы, 14 рисунков, 14 таблиц, 36 источников, 2 приложения.

Ключевые слова: мощности, метод, компенсирующих устройств, оптимизации, сети, решения, потерь.

Объект исследования ВКР – метаэвристические популяционные методы решения многокритериальных задач оптимизации.

Целью данной работы является создание программного кода для решения задач размещения компенсирующих устройств в электрической сети на основе адаптивных алгоритмов роевого интеллекта.

Выполнен обзор методов оптимизации мест размещения и мощностей устройств компенсации реактивной мощности для снижения потерь активной мощности в линиях электропередачи электрической сети. Применительной к решаемой задаче разработан и реализован алгоритм оптимизации на основе модификации алгоритма роя частиц. Выполнено тестирование алгоритма и исследование его эффективности с использованием библиотеки моделирования электроэнергетических систем Pandapower.

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящей работе применяют следующие термины и определения с соответствующими определениями:

IEEE	–	Institute of Electrical and Electronics Engineers (Институт инженеров электротехники и электроники)
CIGRE	–	International Council on Large Electric Systems (Международный совет по крупным электрическим системам)

УСЛОВНЫЕ ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей работе применяют следующие обозначения и сокращения:

КУ	–	Компенсирующие устройства
ЭС	–	Электроэнергетическая система
ЕЭС России	–	Единая энергетическая система России
PSO	–	Particle Swarm Optimization (Оптимизация роя частиц)
ГА	–	Генетический алгоритм
МГС	–	Метод градиентного спуска
НЛП	–	Нелинейное программирование
УКРМ	–	Устройство компенсации реактивной мощности
МКО	–	Многокритериальная оптимизация
НКП	–	Нелинейное квадратичное программирование

ВВЕДЕНИЕ

Объект исследования – компенсация реактивной мощности в системе электроснабжения.

Предмет исследования – методы оптимизации размещения компенсирующих устройств на основе роевого интеллекта.

Цель исследования – разработка методов решения задач размещения компенсирующих устройств в электрической сети на основе адаптивных алгоритмов роевого интеллекта.

Для достижения этой цели поставлены следующие задачи:

1) анализ существующих методов оптимального размещения компенсирующих устройств в электрической сети;
2) обоснование использования метаэвристические популяционных алгоритмов в задачах многокритериальной оптимизации оптимального размещения компенсирующих устройств в электрической сети;
3) разработка адаптивных алгоритмов роевого интеллекта для решения задач многокритериальной оптимизации;

Научная новизна работы заключается в разработке оригинальной модификации алгоритма роя частиц применительно к решению задачи оптимизации размещения компенсирующих устройств в электрической сети.

1 ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ ОПТИМАЛЬНОГО РАЗМЕЩЕНИЯ КОМПЕНСИРУЮЩИХ УСТРОЙСТВ В ЭЛЕКТРИЧЕСКОЙ СЕТИ

1.1 Задача компенсации реактивной мощности

В современной энергетической индустрии вопрос оптимального размещения компенсирующих устройств (КУ) является ключевым в обеспечении надежной и эффективной работы электроэнергетических систем [1-5]. Согласно годовому отчету ПАО «Россети» – крупнейшей системообразующей электросетевой компании России, за 2022 год потери в энергосистеме составили 24 702 млн кВт*ч или 8.69% от общего объема отпущенной электроэнергии потребителям. Данный показатель является одним из ключевых при анализе надежности и качестве энергоснабжения. Снижение потерь электроэнергии является основной целью для сетевых компаний, так как от величины потерь напрямую зависят: экономические затраты на производство, передачу и распределения электроэнергии; надежность и качество энергоснабжения; тарификация цены на электроэнергию и т. д. Поэтому сетевые компании крайне заинтересованы в снижении показателя потерь в сетях.

Одним из ключевых параметров, которые обеспечивают снижение уровня потерь в сети, является уровень напряжения. Так как потери энергии в линии пропорциональны квадрату тока $P = I^2 \cdot R$, а при повышении напряжения ток уменьшается при той же мощности передачи. Изменение напряжения от номинального значительно влияет на потери в электрических сетях. Повышение напряжения, как правило, снижает ток и, следовательно, активные потери, но может увеличить реактивные потери, если не контролировать реактивную мощность. Снижение же напряжения приводит к увеличению тока и, соответственно, к увеличению активных потерь. Для минимизации потерь важно поддерживать оптимальный уровень напряжения и использовать различные меры по управлению напряжением и компенсации

реактивной мощности. Как раз для этих целей в электроэнергетике используются КУ.

Компенсирующие устройства, такие как конденсаторы, реакторы, статические компенсаторы реактивной мощности (СКРМ), батареи статических компенсаторов (БСК) и другие, используются для управления реактивной мощностью, снижения потерь электроэнергии и улучшения качества электроснабжения.

Конденсаторные батареи считаются наиболее удобными и легкими в установке среди других методов компенсации реактивной мощности, что делает их предпочтительным выбором в этом контексте.

Основными задачами компенсации реактивной мощности являются [6]:

1. Снижение потерь активной мощности в сети
2. Снижение падения напряжения
3. Обеспечение нормативного коэффициента мощности
4. Уменьшение величины полной мощности, передаваемой через электросетевые элементы
5. Снижение затрат на эксплуатацию электрической сети

Основная идея компенсации реактивной мощности заключается в следующем [7,8]. На рисунке 1 представлена простейшая схема электроснабжения, включающую в себя линию с активным сопротивлением R, связывающую источник питания напряжением U и потребитель мощностью $P + jQ$.

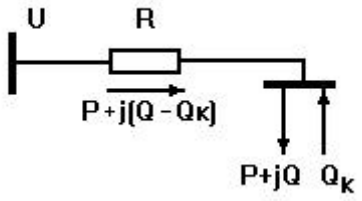

Рисунок 1 – Простейшая схема компенсации реактивной мощности

Потери активной мощности в линии при отсутствии у потребителя компенсирующего устройства ($Q_k = 0$) составляют:

$$\Delta P = \frac{(P^2 + Q^2) \cdot R}{U^2} \quad (1.1)$$

при установке у потребителя компенсирующего устройства ($Q_k \neq 0$) эти потери уменьшатся до величины:

$$\Delta P = \frac{(P^2 + (Q - Q_k)^2) \cdot R}{U^2} \quad (1.2)$$

Следовательно, компенсация реактивной мощности способствует сокращению потерь активной мощности в электроснабжении, что в итоге улучшает технико-экономические характеристики данной системы.

Однако успешность установки КУ в значительной степени зависит от правильного выбора их местоположения в распределительной сети. Этот вопрос может быть решен с помощью современных высокоэффективных алгоритмов, что подтверждается рядом исследований.

1.2 Методы оптимизации расположения и мощности компенсирующих устройств

Первые исследования по определению оптимальной размещений компенсирующих устройств начались в конце прошлого века в 50-х – 80-х годах. Для решения этой задачи применялись в основном аналитические методы, к примеру методы численного программирования [9-13]. В дальнейшем большее распространение получили другие методы.

Среди прямых методов оптимизации, применяемых для выбора компенсирующих устройств, наибольшее распространение получили несколько подходов, таких как метод покоординатного спуска [14,15], метод нелинейного квадратичного программирования [16], метод потенциалов затрат [17] и другие.

Метод покоординатного спуска, или метод чередующихся направлений, применяется для решения задач оптимизации, особенно когда функции не имеют простых аналитических решений. Данный метод зачастую

используется совместно с другими методами для выбора оптимального размещения компенсирующих устройств в распределительной сети [13-14]. Суть метода заключается в последовательном минимизировании целевой функции по одной координате, фиксируя остальные. Процесс начинается с задания начальной точки в пространстве решений и определения критерия остановки, такого как достижение малой разницы между итерациями или превышение максимального числа итераций.

На каждом шаге метода фиксируются все координаты, кроме одной, и целевая функция минимизируется по выбранной координате. Этот процесс повторяется для каждой координаты, пока не будет достигнут критерий остановки. Метод позволяет постепенно приближаться к оптимальному решению, улучшая значение целевой функции на каждом шаге.

Формулировка задачи включает определение целевой функции (например, минимизация потерь мощности) и установление ограничений (допустимые диапазоны напряжений, мощности устройств). Далее определяется начальное размещение компенсирующих устройств, и рассчитываются начальные потери мощности и напряжения в узлах [15].

На каждой итерации фиксируется размещение всех компенсирующих устройств, кроме одного, и оптимизируется размещение выбранного устройства для минимизации потерь мощности. После каждой оптимизации пересчитываются параметры сети. Этот процесс повторяется до достижения критерия остановки, такого как незначительное изменение потерь мощности или максимальное число итераций.

Таким образом, данный метод позволяет поэтапно улучшать размещение компенсирующих устройств, оптимизируя сеть для минимизации потерь мощности или других эксплуатационных показателей. Этот метод особенно эффективен для задач с большими размерами, где оптимизация по всем координатам сразу затруднена или невозможна.

Другим распространенным методом является метод нелинейного квадратичного программирования (НКП) [16] Метод нелинейного

программирования (НП) представляет собой набор математических методов и алгоритмов для решения задач оптимизации, где целевая функция или ограничения, или обе, являются нелинейными. Эти методы находят широкое применение в различных областях, включая экономику, инженерное дело, управление и другие дисциплины, где линейные модели недостаточны для описания реальных процессов [17].

Процесс решения задач нелинейного программирования начинается с формулировки задачи: определяются целевая функция и ограничения. Целевая функция $f(x)$ должна быть оптимизирована (максимизирована или минимизирована) при соблюдении ограничений $g_i(x) \leq 0 и h_j(x) = 0$, где$x$ – вектор переменных. Задачи нелинейного программирования могут включать как неравенства, так и равенства, что делает их более сложными по сравнению с линейными задачами.

Одним из основных методов нелинейного программирования является метод Лагранжа, который использует мультипликаторы Лагранжа для преобразования задачи с ограничениями в задачу без ограничений [18]. Метод Куна-Таккера (или ККТ-условия) расширяет этот подход, обеспечивая необходимые и достаточные условия оптимальности для задач с неравенствами [19]. Эти методы позволяют находить стационарные точки целевой функции, которые являются кандидатами на оптимальные решения.

Для численного решения задач НП часто используются градиентные методы, такие как метод наискорейшего спуска и метод Ньютона. Метод наискорейшего спуска заключается в итеративном движении в направлении антиградиента целевой функции для нахождения локального минимума. Метод Ньютона использует вторые производные для ускорения сходимости, что делает его более эффективным для задач, где требуется высокая точность.

В практических приложениях, таких как оптимизация размещения производства, управление ресурсами, планирование маршрутов и дизайн инженерных систем, методы нелинейного программирования позволяют учитывать сложные взаимосвязи и нелинейные эффекты. Это делает их

незаменимыми инструментами для решения задач, где линейные модели не могут адекватно описать реальность, обеспечивая более точные и эффективные решения.

Метод потенциалов затрат, также известный как метод потенциалов узлов, применяется для решения задач оптимизации в транспортных и сетевых системах. Его основная цель — минимизация затрат на перемещение ресурсов по сети при соблюдении всех ограничений. Процесс начинается с инициализации начального допустимого решения, которое обычно задается методом северо-западного угла или методом минимальных затрат. Далее вычисляются потенциалы для каждого источника и потребителя, удовлетворяющие уравнению $u_i + v_j = c_{ij}$ для базисных ячеек.

После этого для небазисных ячеек вычисляются оценки$\Delta_{ij} = c_{ij} - (u_i + v_j)$. Если все эти оценки положительны или равны нулю, решение считается оптимальным. Если найдена отрицательная оценка, решение можно улучшить путем построения цикла перемещений и перераспределения ресурсов таким образом, чтобы улучшить общую стоимость. Процесс повторяется до достижения оптимального решения.

Метод потенциалов затрат можно эффективно применять для задачи оптимального размещения компенсирующих устройств в электрической сети [20]. В этой задаче целью является минимизация потерь мощности или стоимости энергии при соблюдении ограничений на напряжение в узлах сети. Для начала необходимо смоделировать сеть, определить структуру узлов и линий, а также параметры сети, такие как мощности нагрузок и допустимые уровни напряжения.

Задача формулируется путем определения цели оптимизации (например, минимизация потерь мощности) и ограничений (допустимые диапазоны напряжений, мощности устройств). Далее определяется начальное допустимое размещение компенсирующих устройств и рассчитываются текущие потери мощности и напряжения в узлах. Затем, аналогично методу потенциалов

затрат, рассчитываются потенциалы узлов и стоимость перемещения компенсирующих устройств.

На основе разницы в потерях мощности определяется перемещение, которое приведет к наибольшему уменьшению потерь. Выполняется соответствующее перемещение компенсирующих устройств, пересчитываются параметры сети, и процесс повторяется до тех пор, пока дальнейшее уменьшение потерь или затрат невозможно. Таким образом, метод затратных потенциалов предоставляет эффективный способ оптимизации размещения компенсирующих устройств в электрических сетях, позволяя сократить потери и улучшить их эксплуатационные характеристики.

Эти методы отличаются постановкой исходной задачи и способом её решения. Тем не менее, их объединяет то, что они являются прямыми методами, основанными на итеративных процессах вычисления и сравнения значений оптимизируемых функций.

Однако, в большинстве случаев, исходная задача представляет собой задачу безусловной оптимизации, где необходимо найти абсолютный экстремум целевой функции без каких-либо ограничений или граничных условий. Кроме того, данные алгоритмы не учитывают такие факторы, как увеличение пропускной способности линий, рост нагрузки, изменение графиков потребления электроэнергии, проблемы с высоким напряжением в периоды низкой нагрузки, изменение стоимости электроэнергии и стандартные значения мощности производимого оборудования.

Кроме того, эти алгоритмы имеют ряд недостатков, связанных с упрощением моделей реальных сетей. Например, они предполагают равномерное распределение реактивной нагрузки в сети, одинаковые сечения проводов линий и ограниченное количество устанавливаемых устройств, а также не учитывают ряд других факторов.

Определение оптимального местоположения КУ является сложным из-за комбинаторной природы задачи, вызванной дискретным характером множества возможных расположений и номинальных параметров устройств

компенсации реактивной мощности (УКРМ). Кроме того, при проведении расчетов необходимо учитывать не только обычный режим работы энергетической системы (ЭС), но и различные варианты отключения отдельных ее компонентов, а также различные аварийные ситуации [21]. Эти факторы делают задачу сбалансированной и требуют использования вероятностных методов для оценки результатов.

1.3 Особенности популяционных алгоритмов

Поэтому в данное время предпочтение в решении задачи оптимального места размещения и выбора мощности компенсирующих устройств отдается эвристическим методам [22-25, 27-28], в том чисел с использованием искусственного интеллекта.

Одним из наиболее распространённых алгоритмов оптимизации является алгоритма роя частиц (PSO). Этот метод основан поведении роя насекомых или стай рыб. Частицы (потенциальные решения) перемещаются в пространстве поиска, обновляя свои позиции на основе собственного опыта и опыта соседей. Алгоритм показывает свою эффективность для решения сложных задач с большой размерностью, является довольно простым в реализации, обладает достаточной масштабируемостью, а также простота его реализации и высокая гибкость делает его весьма популярным при решении задач в разных сфера [28]. Но при этом данный метод имеет достаточно высокую степень конвергенции, то есть обладает особенность преждевременного схождения в локальный оптимум. И также он требует тщательной настройки параметров для корректно поиска оптимального решения задачи.

Другим часто встречающимся методом оптимизации является генетический алгоритм (ГА). Данный алгоритм имитирует процесс естественного отбора, используя популяцию потенциальных решений, которая эволюционирует на основе операторов мутации, скрещивания и отбора [24, 25]. К его плюсам можно отнести как простоту реализации, так и

способность находить глобальный оптимум, избегая локальных минимумов. Немаловажным плюсом также является его применимость к задачам с большим числом переменных и различных ограничений. Но при этом данный алгоритм имеет и существенные минусы, такие как: высокое время расчета из-за большого количества выселений для низкоэффективных решений, сложная настройка работы алгоритма, так как требуется довольно точный подбор его параметров (размер популяции, вероятность мутации, скрещивания, селекции и т. д.).

В литературе также описывается применение другого метода – метода роя пчел (Bee Algorithm). Данный метод является одним из метаэвристических методов оптимизации, вдохновленным поведением пчел при поиске пищи. Этот метод имитирует процесс поиска нектара пчелами и используется для решения сложных оптимизационных задач, где традиционные методы могут быть менее эффективными. Метод роя пчел сочетает в себе глобальный поиск новых областей пространства решений и локальный поиск вблизи уже найденных хороших решений.

Процесс начинается с инициализации случайной популяции пчел, каждая из которых представляет возможное решение задачи. Эти начальные решения распределяются по пространству поиска. Затем пчелы оценивают качество (фитнес) своих текущих позиций на основе заданной целевой функции. Пчелы с наилучшими решениями (разведчицы) отправляются на более тщательный поиск в своих областях, привлекая других пчел (наблюдателей) для локального поиска вокруг перспективных позиций.

После этапа локального поиска пчелы сравнивают новые найденные решения с текущими и обновляют свои позиции, если найденные решения лучше. Таким образом, метод роя пчел сочетает глобальный и локальный поиск, что позволяет ему избегать застревания в локальных минимумах и повышает вероятность нахождения глобального оптимума. Процесс повторяется до достижения заданного критерия остановки, такого как

максимальное число итераций или достижение удовлетворительного уровня фитнеса.

Метод роя пчел обладает гибкостью и адаптивностью, что делает его эффективным для решения различных задач оптимизации, включая те, где пространство поиска является дискретным, непрерывным или смешанным. Он успешно применяется в задачах оптимального проектирования, планирования маршрутов, управления ресурсами и других областях, где требуется эффективный поиск глобального оптимума.

Применение метода роя пчел к задаче оптимального размещения компенсирующих устройств в электрической сети предполагает минимизацию потерь мощности или стоимости энергии при соблюдении ограничений на напряжение в узлах сети [5]. Сначала необходимо смоделировать сеть, включая узлы (подстанции, потребители) и линии передачи, и определить параметры сети, такие как мощности нагрузок и допустимые уровни напряжения.

В этом контексте пчелы представляют возможные конфигурации размещения компенсирующих устройств. На каждой итерации оценивается фитнес каждой конфигурации, например, на основе потерь мощности в сети. Пчелы с лучшими конфигурациями отправляются на локальный поиск, привлекая других пчел для исследования близлежащих решений. Этот процесс позволяет постепенно улучшать размещение компенсирующих устройств, снижая потери мощности и обеспечивая соответствие напряжений в узлах сети заданным требованиям.

Таким образом, метод роя пчел эффективно решает задачу оптимального размещения компенсирующих устройств, обеспечивая баланс между глобальным и локальным поиском. Это позволяет минимизировать потери мощности и улучшить эксплуатационные характеристики сети, делая метод полезным инструментом для задач энергетического управления и оптимизации.

Так или иначе, данные алгоритмы не всегда позволяют найти глобальный минимум целевой функции, при котором будет найден минимум потерь активной мощности и эксплуатационных затрат.

1.4 Выводы по главе 1

Рассмотрена задача компенсации реактивной мощности в электрической сети для снижения потерь активной мощности в линиях электропередачи, определена ее актуальность. Выполнен обзор существующих методов решение данной оптимизационной задачи, показано, чир применение метаэвристических алгоритмов, таких как популяционные алгоритмы, является перспективным направлением исследований в рассматриваемой области.

2 АНАЛИЗ МЕТОДОВ МНОГОКРИТЕРИАЛЬНОЙ ОПТИМИЗАЦИИ ПРИМЕНИТЕЛЬНО К ЗАДАЧЕ РАЗМЕЩЕНИЯ КОМПЕНСИРУЮЩИХ УСТРОЙСТВ

Поиск оптимального размещение компенсирующих устройств в электрической сети — сложная задача, для решения которой может использоваться целевая функция как с одним критерия, так и с множеством критериев. В качестве одного критерия может использоваться: минимизация потерь активной мощности; минимизация потерь электроэнергии; минимизация издержек; минимизация отклонения напряжения; максимизация выгоды; максимизация выдачи мощности от РГ.

Выражения для многокритериальной оптимизации можно разделить на три группы. В первую группу входят функции, представляющие сумму отдельных критериев с различными весами. Ко второй группе относятся функции, в которых различные критерии входят в формулу для расчета многокритериального коэффициента, для которого осуществляется поиск экстремума. К третьей группе относятся функции, определяющие лучшее многокритериальное решение, как равноудаленное от лучших однокритериальных решений и находящееся посередине [26].

2.1 Метод градиентного спуска

Метод градиентного спуска (МГС) представляет собой один из фундаментальных методов оптимизации, широко используемых в различных областях науки и техники, включая машинное обучение, статистику и численные методы [14]. Его основное предназначение заключается в минимизации (или максимизации) функции путём итеративного улучшения её значений.

Основная идея метода градиентного спуска заключается в движении по направлению, противоположном градиенту функции в точке текущего значения параметров. Пусть $f(x)$ – дифференцируемая функция, которую

требуется минимизировать, где $x \in \mathbb{R}^n$ – вектор параметров. Градиент функции $\nabla f(x)$ – это вектор, содержащий частные производные функции f по каждому из параметров x, указывающий направление наибольшего возрастания функции в данной точке:

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_k}\right), \quad (2.1)$$

где $\frac{\partial f}{\partial x_k}$ – частная производная функции f по k -му параметру x

Следовательно, движение в направлении, противоположном градиенту, уменьшает значение функции:

$$x_{k+1} = x_k - \alpha \nabla f(x), \quad (2.2)$$

где α – шаг или скорость обучения, определяющая, насколько сильно изменяются параметры на каждой итерации.

Алгоритм градиентного спуска можно описать следующим образом [18]:

- Выбрать начальную точку x_0 и задать шаг α;
- Вычислить градиент функции $\nabla f(x_k)$ в текущей точке x_k ;
- Обновить параметры по формуле $x_{k+1} = x_k - \alpha \nabla f(x)$;
- Проверить критерий останова:
    - Норма градиента $\|\nabla f(x_k)\|$ меньше заданного порога ε;
    - Изменение функции $|f(x_{k+1} - f(x_k))|$ меньше заданного порога;
    - Достигнуто максимальное количество итераций.
- Когда критерий остановки выполнен, вернуть текущие значения параметров x_k.

2.2 Алгоритм роя частиц

Алгоритм роя частиц (PSO) представляет собой метод оптимизации, основанный на коллективном поведении группы агентов, называемых

частицами, которые движутся в поисковом пространстве для нахождения оптимального решения.

Каждая частица i характеризуется текущей позицией x_i и скоростью v_i, а также хранит информацию о наилучшей найденной позиции p_i. Кроме того, весь рой частиц хранит информацию о глобально наилучшей найденной позиции g [28].

Шаги обновления скорости, позиции и личных и глобальных лучших положений повторяются до тех пор, пока не будет достигнуто максимальное количество итераций или не будет удовлетворено условие остановки (например, достижение заданной точности).

Рассматривая работу алгоритма работы PSO, можно выделить несколько основных этапов:

Частицы инициализируются случайным образом в пределах допустимых границ (2.3):

$$x_i(0) \sim U(l, u), \tag{2.3}$$

где l, u — векторы нижних и верхних границ соответственно. Скорости также инициализируются случайным образом:

$$v_i(0) \sim U(-v_{max}, v_{max}), \tag{2.4}$$

На каждой итерации скорость частицы i обновляется по формуле:

$$v_i(t+1) = wv_i(t) + c_1 r_1\big(p_i(t) - x_i(t)\big) + c_2 r_2\big(g(t) - x_i(t)\big), \tag{2.5}$$

где w – вес инерции, контролирующий влияние предыдущей скорости на текущую [0.4, 0.9],

c_1 – когнитивный коэффициент, определяющий силу притяжения к личному лучшему положению [1.0, 2.0],

c_2 – социальный коэффициент, определяющий силу притяжения к глобальному лучшему положению [1.0, 2.0],

r_1 и r_2 – случайные числа, равномерно распределенные в интервале [0,1].

Позиция частицы обновляется с использованием новой скорости:

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2.6}$$

Позиции (2.5) и скорости (2.6) ограничиваются заданными границами:

$$x_i(t+1) = clip(x_i(t+1), l, u), \tag{2.7}$$

$$v_i(t+1) = clip(v_i(t+1), -v_{max}, v_{max}), \tag{2.8}$$

где $clip(x, a, b)$ — функция, ограничивающая x интервалом $[a, b]$.

После обновления позиции вычисляется значение целевой функции $f(x_i(t+1))$. Если оно лучше текущего лучшего значения для частицы, то обновляется личное лучшее положение:

если $f\big(x_i(t+1)\big) < f(p_i(t))$, то $p_i(t+1) = x_i(t+1)$

Если оно лучше глобального лучшего значения, то обновляется глобальное лучшее положение:

если $f\big(x_i(t+1)\big) < f(g(t))$, то $g(t+1) = x_i(t+1)$

2.3 Генетический алгоритм

Генетический алгоритм (ГА) — это метод оптимизации, который вдохновлён естественным отбором и генетикой. Он используется для поиска приблизительных решений сложных задач оптимизации, когда другие методы менее эффективны. Основная идея заключается в эволюции популяции потенциальных решений через поколения с использованием операций, подобных биологической эволюции [29].

Суть работы алгоритма в следующем. Сначала создаётся начальная популяция потенциальных решений, обычно случайным образом. Каждое решение называется индивидуумом или хромосомой.

$$P(0) = \{x_1, x_2, \dots, x_n\}, \tag{2.9}$$

где $P(0)$ – начальная популяция;

x_n – индивидуум (решение) в популяции, где $i = 1, 2, \dots, n$;

n – размер популяции (количество индивидуумов).

Затем каждое решение оценивается с использованием функции приспособленности (fitness function), которая определяет, насколько хорошим является решение для данной задачи. Функция приспособленности $f(x_i)$ может быть в форме:

$$f(x_i) = \frac{1}{1 + g(x_i)}, \tag{2.10}$$

где $g(x_i)$ – целевая функция.

Далее индивидуумы отбираются для участия в создании следующего поколения. Чем лучше приспособленность, тем выше шанс быть выбранным.

$$P_{selected} = \{x_i | P_{select}(x_i) > random(0,1)\}, \tag{2.11}$$

где $P_{selected}$ – целевая функция;

$P_{select}(x_i)$ – вероятность отбора индивидуума x_i, пропорциональная его приспособленности;

$random(0,1)$ – случайное число от 0 до 1.

$$P_{select}(x_i) = \frac{f(x_i)}{\sum_{i=1}^{n} f(x_i)}, \tag{2.12}$$

где $\sum_{i=1}^{n} f(x_i)$ – сумма приспособленностей всех индивидуумов в популяции.

После этого отобранные индивидуумы (родители) комбинируются для создания потомков, так называемый процесс скрещивания (кроссовер). Это обеспечивает передачу генов из одного поколения в другое. В качестве способа скрещивания выбирается метод рулетки, турнирного отбора или другой способ:

$$x_{child1}, x_{child2} = crossover(x_{parent1}, x_{parent2}) \ , \tag{2.13}$$

где x_{child1}, x_{child2} – потомки;

$x_{parent1}, x_{parent2}$ – родители;

Некоторая доля потомков подвергается случайным изменениям для поддержания генетического разнообразия в популяции:

$$x_i' = \begin{cases} random(0,1), \ random(0,1) < p_{mutation} \\ x_i, \ random(0,1) \geq p_{mutation} \end{cases}, \tag{2.14}$$

где x_i' – мутировавший индивидуум;

$p_{mutation}$ – вероятность мутации.

Затем старое поколение заменяется новым:

$$P(t+1) = replace(P(t), P_{new}), \tag{2.15}$$

где $P(t+1)$ – новое поколение;

$P(t)$ – текущее поколение;

P_{new} – новое поколение после кроссовера и мутации.

Данный процесс повторяется до тех пор, пока не будет достигнут критерий остановки (например, номер текущего поколения больше заданного количества поколений или достигнуто заданное значение целевой функции приспособленности).

2.4 Метод взвешенных сумм

Одним из наиболее популярных методов мнокритериальной оптимизации (МКО) является метод взвешенных сумм (Weighted Sum Method). Данный метод преобразует многокритериальную задачу в однокритериальную, присваивая каждому критерию весовой коэффициент и суммируя результаты [30]. Функция цели имеет вид:

$$F = \sum_{i=1}^{n} w_i \cdot f_i, \tag{2.9}$$

где w_i – весовой коэффициент для i-го критерия;

f_i – значение i-го критерия.

К достоинствам данного МКО можно отнести следующее:

- Простота реализации и понимания;
- Возможность учитывания различных приоритетов критериев.

Также у метода взвешенных сумм имеется и ряд недостатков, таких как:

- Довольно большая сложность выбора правильных весовых коэффициентов;
- Возможное игнорирование важных критериев, если весовые коэффициенты выбраны неправильно.

Применительно к задаче выбора оптимального размещения КУ в ЭС в качестве критериев можно указать: снижение потерь активной мощности в сети, экономические критерии по минимизации затрат, критерии, связанные с отклонением напряжения и др.

2.5 Метод ограничений

Другой часто используемый метод – это метод ограничений (Constraint Method). В нем один из критериев рассматривается как целевая функция, а остальные критерии включаются как ограничения задачи. К основным методам ограничений относятся фиксация граничных значений, штрафных функций, множителей Лагранжа и т. п. [31] В общем виде формула имеет вид:

$$\min f_1, \quad \text{при условии } f_i \leq b_i, \forall i \neq 1, \tag{2.10}$$

де b_i – допустимые пределы для i-го критерия.

Достоинства метода ограничений:

- Четкое соблюдение заданных ограничений;
- Возможность управления компромиссами между критериями.

Недостатки данного метода:

- Нет гарантии выбора оптимального решения задачи из множества приемлемых решений;
- Возможность игнорирования критериев, не включенных в целевую функцию или ограничения.

В качестве оптимизации одного критерия можно использовать минимизацию потерь активной мощности, при соблюдении ограничений по другим критериям (например, ограничения по затратам и качеству напряжения)

2.6 Выводы по главе 2

Определено, что необходимо учитывать многокритериальность задачи оптимизации размещения устройств компенсации реактивной мощности в электрической сети. При этом большинство методов оптимизации направлено на решение однокритериальной задачи, поэтому чаще всего выбирается один

критерий, остальные переводятся в разряд ограничений. В данной работе было принято решение использовать взвешенную сумму критериев, чтобы не усложнять алгоритмы оптимизации и при этом не выбирать один приоритетный критерий.

3 РАЗРАБОТКА АДАПТИВНЫХ АЛГОРИТМОВ РОЕВОГО ИНТЕЛЛЕКТА ДЛЯ РЕШЕНИЯ ЗАДАЧ МНОГОКРИТЕРИАЛЬНОЙ ОПТИМИЗАЦИИ

3.1 Постановка задачи для применения алгоритма роя частиц

В качестве критериев оптимизации было принято решение использовать два критерия в качестве основных: суммарные потери активно мощности в энергосистеме $\Delta P_{\text{сумм}}$ и сумма отклонений напряжения $\Delta U_{\text{сумм}}$ в каждом узле. Для нормализации критерия, отвечающего за потери активной мощности, на каждом шаге итерации было предложено находить сумму отношений текущего значения потери мощности на линии и значения потери мощности на линии без оптимизации:

$$\Delta P_{\text{сумм}} = \sum_{i=1}^{n} \frac{\Delta P_i}{\Delta P_{di}}, \tag{3.1}$$

где n – количество линий в сети;

ΔP_{di} – активные потери в i линии в сети без компенсирующих устройств, МВт.

ΔP_i – активные потери в i линии в сети с компенсирующими устройствами, МВт.

Для критерия, отвечающего за величину отклонения напряжения в узлах, производилось суммирование модулей отклонения напряжения на каждом из узлов. Также для нормализации данного критерия значение модуля отклонения напряжения домножалось на 10:

$$\Delta U_{\text{сумм}} = \sum_{i=1}^{n} |U_i| \cdot 10, \tag{3.2}$$

где n – количество узлов в сети;

U_i – напряжение в узле относительно номинального, о.е.

В качестве метода определения целевой функции был взят метод взвешенных сумм, представляющий собой сумму двух критериев с соответствующим весовым коэффициентом каждый $J(P, U)$:

$$J(P, U) = \Delta P_{\text{сумм}} \cdot k_1 + \Delta U_{\text{сумм}} \cdot k_2 \,, \tag{3.3}$$

где $k_1,\ k_2$ – весовые коэффициенты.

Были выбраны следующие весовые коэффициенты: $k_1 = 0.7$ (коэффициент, учитывающий влияние значения потерь активной мощность в целевой функции), $k_2 = 0.3$ (для критерия отклонения напряжения).

3.2 Апробация алгоритма на примере сети IEEE-case-6

Для моделирования энергосистемы была использована библиотека Pandapower языке программирования Python. В качестве основной сети была выбрана «Case 6ww» по стандарту IEEE и описана в книге A. J. Wood and B. F. Wollenberg, Power generation, operation, and control. John Wiley & Sons, 2012. Данная ЭС имеет 6 узлов напряжением 230 кВ. В узлах 2 и 3 расположены генераторы мощностью 50 и 60 МВт соответственно.

Для моделирования КУ использовалась встроенная модель статического генератора (Static generator). Для данного элемента ЭС указывалось только значение реактивной мощности в МВар, выдаваемой в сеть.

На первом этапе формируется сеть командой network = nw.case6ww(). В каждый узел сети подключается статический генератор с номинальными параметрами: $p_{mv} = 0.0$, $q_{mvar} = 0$. После этого выполняется расчет режима методом Ньютона-Рафсона для того, чтобы определить параметры сети в нормальном режиме без оптимизации. Результаты расчетов представлены в таблицах 1 и 2. А также на рисунке 2 представленная схема сети с цветовым отображением уровня напряжения в каждом узле в о.е. и загрузкой каждой линии в процентах.

Таблица 1 – Параметры узлов неоптимизированной сети

Номер узла	Отношение напряжения в узле к номинальному, о.е.	Угол напряжения, °	Потребляемая активная мощность, МВт	Потребляемая реактивная мощность, МВар	Отклонение напряжения по модулю, о. е
1	1.050	0.000	-107.875	-15.956	0.050
2	1.050	-3.671	-50.000	-74.356	0.050
3	1.070	-4.273	-60.000	-89.627	0.070
4	0.989	-4.196	70.000	70.000	0.011
5	0.985	-5.276	70.000	70.000	0.015
6	1.004	-5.947	70.000	70.000	0.004
Суммарное отклонение напряжения					0.200

Таблица 2 - Параметры линий неоптимизированной сети

Номер линии	Номер начального узла	Номер конечного узла	Активные потери в линии, МВт	Реактивные потери в линии, МВар
0	0	1	0.905	-2.600
1	0	3	1.088	0.188
2	0	4	1.074	-2.195
3	1	2	0.040	-6.541
4	1	3	1.505	0.929
5	1	4	0.498	-2.653
6	1	5	0.583	-3.612
7	2	4	1.094	-2.921
8	2	5	1.003	2.863
9	3	4	0.036	-7.727
10	4	5	0.050	-5.791
Суммарные потери			7.876	-30.060

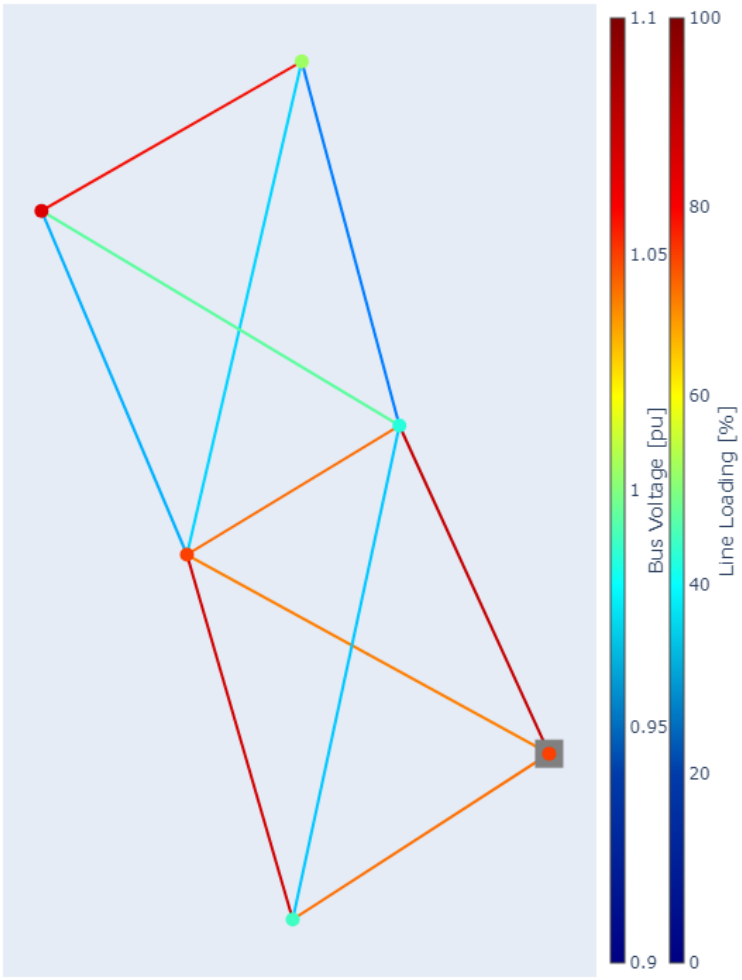

Рисунок 2 – Условная схема неоптимизированной сети с указанием напряжений в узлах и загрузки каждой линий

Далее запускается алгоритм роя частиц, в котором в роли частиц выступает список значений реактивных мощностей статических генераторов, установленных в каждый узел сети. В качестве начальных параметров метода были выбраны средние значения из соответствующих диапазонов (2.3): $w = 0.6, c_1 = 1.5, c_2 = 1.5$.

На первом шаге работы алгоритма рандомно задаются величины мощностей в диапазоне от 0 до 1. При расчете режима данные величин умножаются на 100, чтобы значение реактивной мощности в каждом узле примерно соответствовало классу напряжения и средним номинальным величинам компенсирующих устройств. Также задается количество итераций равное 20 и количество частиц – 10.

После этого производится расчет режима, и на основании результатов расчета, вычисляется целевая функция, состоящая из суммы потерь активной мощности в энергосистеме $\Delta P_{\text{сумм}}$ и отклонений напряжения $\Delta U_{\text{сумм}}$ в узлах сети (3.3). В качестве метода многокритериальной оптимизации был выбран метод взвешенных сумм. Для критерия суммы потерь активной мощности был выбран коэффициент 0.7, а для критерия отклонения напряжения от номинального – 0.3.

Получив значение целевой функции, для каждой частицы выполняется проверка: если оно лучше текущего лучшего значения для частицы, то обновляется личное лучшее положение; если оно лучше глобального лучшего значения, то обновляется глобальное лучшее положение. Для визуализации принципа работы алгоритма ниже приведена блок-схема алгоритма PSO (рисунок 3).

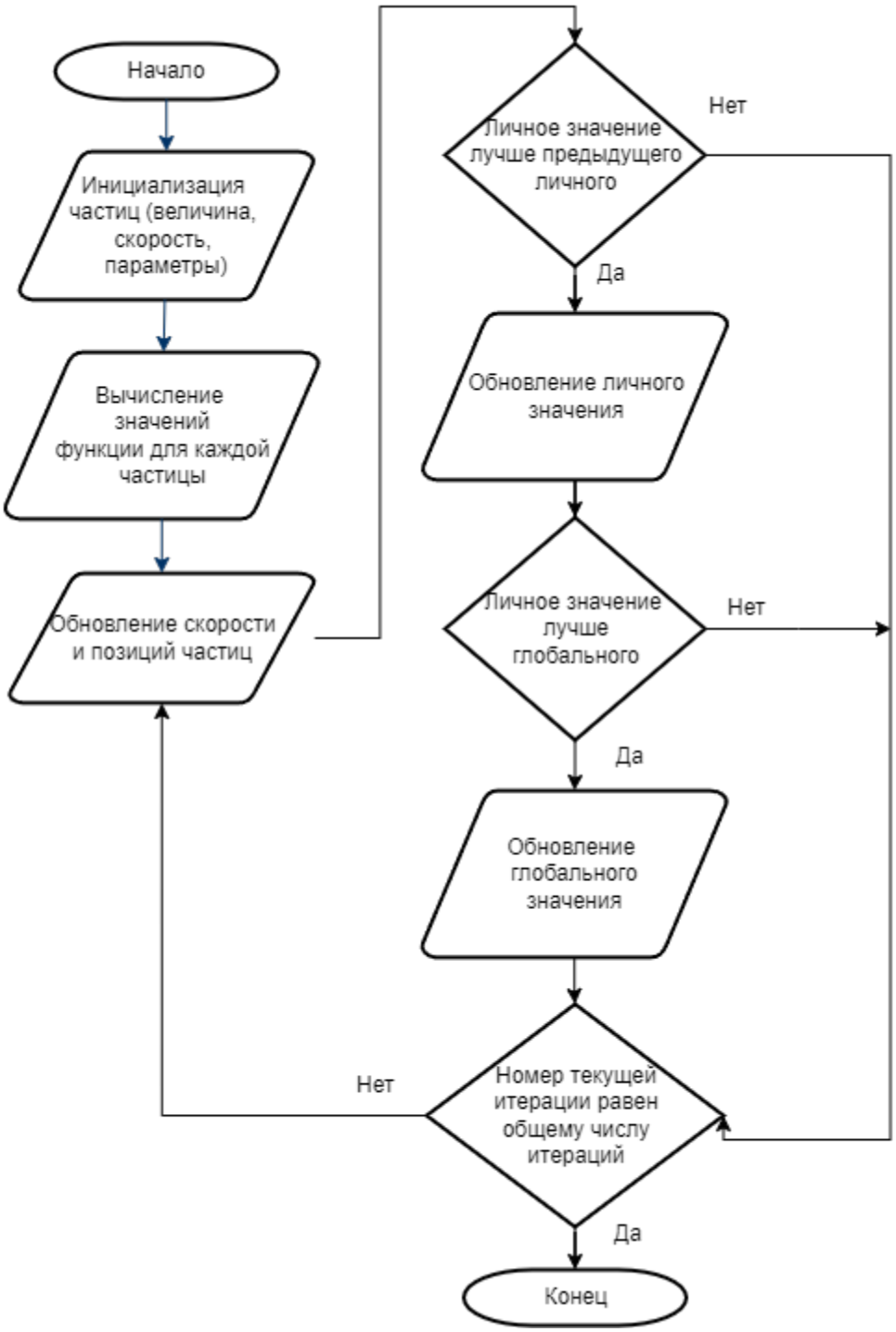

Рисунок 3 – Блок схема алгоритма PSO

По результатам работы программы были получены значения основных элементов сети: узлов, линий и КУ. Данные представлены в таблицах 3–5. Помимо этого ниже на рисунке 4 представлена схема сети с цветовым отображением уровня напряжения в каждом узле в о.е. и загрузкой каждой линии в процентах после выполнения оптимизации. На рисунке 5 изображен график изменения значения целевой функции относительно номера итерации.

Таблица 3 – Параметры узлов сети после оптимизации методом роя частиц

Номер узла	Отношение напряжения в узле к номинальному, о.е.	Угол напряжения, °	Потребляемая активная мощность, МВт	Потребляемая реактивная мощность, МВар	Отклонение напряжения по модулю, о. е
1	1.050	0.000	-104.428	27.278	0.050
2	1.050	-3.121	-50.000	16.119	0.050
3	1.070	-3.712	-60.000	-33.416	0.070
4	1.042	-4.849	70.000	-7.311	0.042
5	1.037	-5.684	70.000	12.265	0.037
6	1.038	-5.792	70.000	27.525	0.038
Суммарное отклонение напряжения					0.284

Таблица 4 - Параметры линий сети после оптимизации методом роя частиц

Номер линии	Номер начального узла	Номер конечного узла	Активные потери в линии, МВт	Реактивные потери в линии, МВар
0	0	1	0.654	-3.102
1	0	3	0.929	-0.660
2	0	4	0.902	-3.154
3	1	2	0.040	-6.542
4	1	3	0.424	-1.339
5	1	4	0.234	-3.657
6	1	5	0.392	-4.328
7	2	4	0.347	-4.801
8	2	5	0.480	0.180
9	3	4	0.025	-8.597
10	4	5	0.000	-6.459
Суммарные потери			4.428	-42.460

Таблица 5 - Параметры КУ сети после оптимизации методом роя частиц

Номер узла	Реактивная мощность КУ, МВар
2	43.571
3	91.409
4	77.311
5	57.735
6	42.475

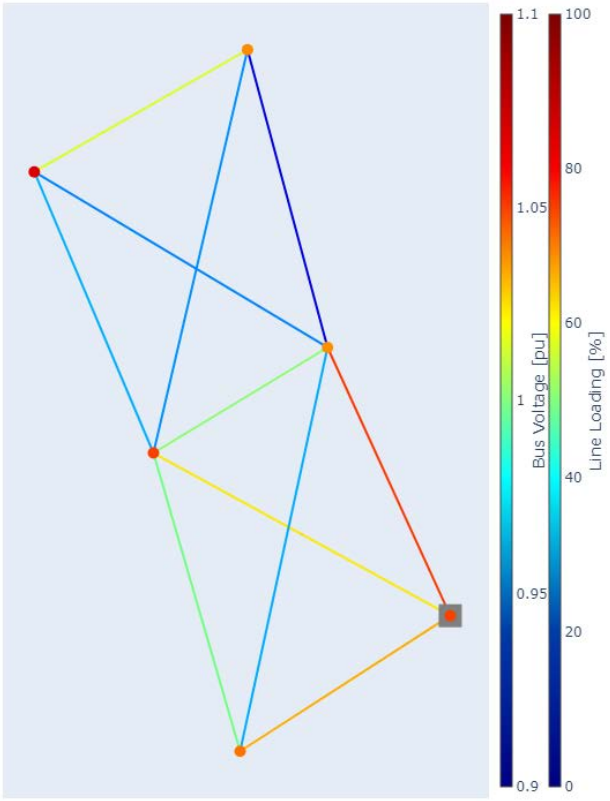

Рисунок 4 – Условная схема оптимизированной сети методом роя частиц с указанием напряжений в узлах и загрузки каждой линий

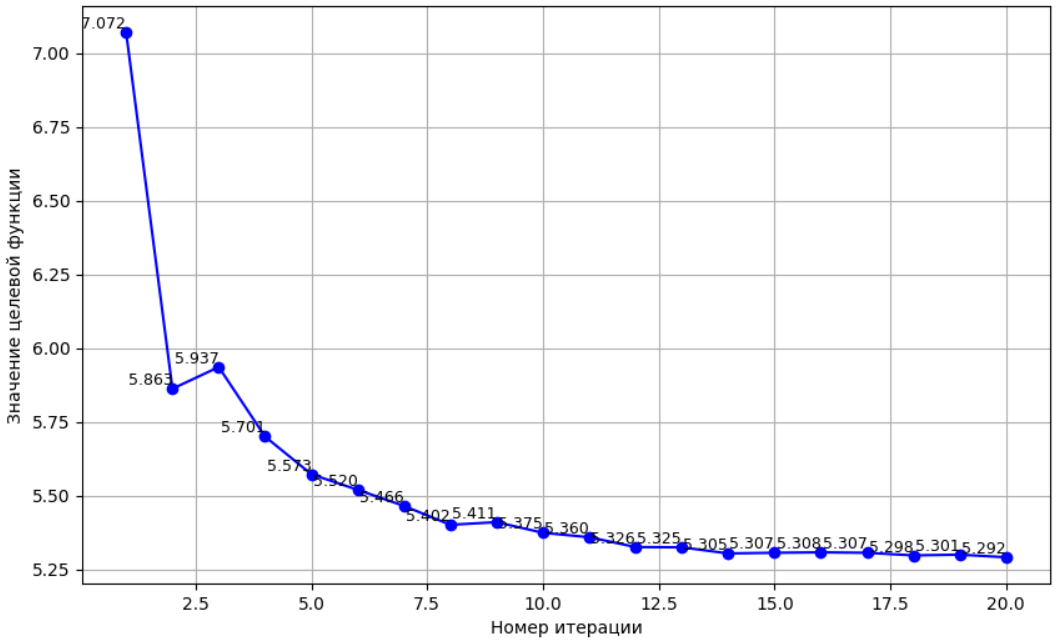

Рисунок 5 – График зависимости значения целевой функции от номера итерации в методе роя частиц (количество частиц =20; количество итераций = 20; w=0.6, c1=1.5, c2=1.5)

Из результатов работы алгоритма видно, что потери активной мощности в линиях снизились на 56.22% или на 3.448 МВт. Также можно наблюдать, что в узлах 4 и 5 отклонение напряжение от номинального стало 4.2%, 3.7% и 3.8% соответственно. Помимо этого, стоит отметить, что метод роя частиц показал высокую степень сходимости даже на малом количестве итераций и с выставлением средних значений параметров w, c_1, c_2 из общепринятого диапазона. И исходя из результатов тестирования работы метода, можно сказать, что он обладает достаточно простой настройкой параметров и стабильно высоком качестве полученных результатов расчётов.

Для сравнения результатов работы алгоритма роя частиц в задачи выбора оптимального места компенсирующих устройств в сети, был также применен метод градиентного спуска.

На первом этапе для метода градиентного спуска эмпирическим путем были получены следующие значения параметров метода: количество итераций = 200; a=0.05, delta=0.05. Также установляются начальные значения, выбранные рандомно из диапазона от 0 до 1.

После этого производится расчет режима, и на основании результатов расчета, вычисляется целевая функция, основанная также на методе взвешенных сумм критериев $\Delta P_{\text{сумм}}$ и $\Delta U_{\text{сумм}}$.

На основании полученного значения целевой функции считается градиент (2.1) и производится проверка критериев останова: меньше ли заданного порога $\varepsilon = 1 \cdot 10^{-6}$ норма градиента $\|\nabla f(x_k)\|$ меньше или число итераций достигнуто указанного значения $iter = 200$.

Для визуализации принципа работы ниже приведена блок-схема метода градиентного спуска (рисунок 6).

В таблицах 6–7 приведены результирующие значения параметров сети после оптимизации данным методом. Помимо этого, ниже на рисунке 7 представлена схема сети с цветовым отображением уровня напряжения в каждом узле в о.е. и загрузкой каждой линии в процентах после выполнения

оптимизации. На рисунке 8 изображен график изменения значения целевой функции относительно номера итерации.

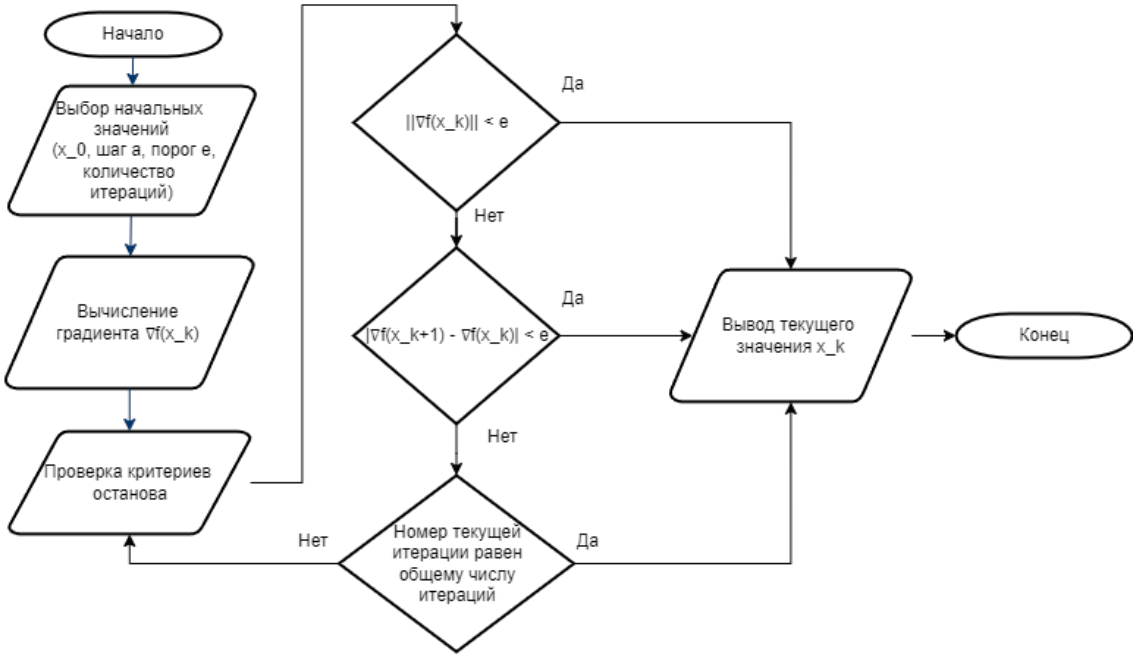

Рисунок 6 – Блок схема метода градиентного спуска

Таблица 6 – Параметры узлов сети после оптимизации методом градиентного спуска

Номер узла	Отношение напряжения в узле к номинальному, о.е.	Угол напряжения, °	Потребляемая активная мощность, МВт	Потребляемая реактивная мощность, МВар	Отклонение напряжения по модулю, о. е
1	1.050	0.000	-104.460	27.089	0.050
2	1.050	-3.124	-50.000	14.497	0.050
3	1.070	-3.708	-60.000	-36.527	0.070
4	1.042	-4.851	70.000	-7.421	0.042
5	1.037	-5.678	70.000	12.229	0.037
6	1.035	-5.757	70.000	32.398	0.035
Суммарное отклонение напряжения					0.284

Таблица 7 - Параметры линий сети после оптимизации методом градиентного спуска

Номер линии	Номер начального узла	Номер конечного узла	Активные потери в линии, МВт	Реактивные потери в линии, МВар
0	0	1	0.655	-3.100
1	0	3	0.930	-0.657
2	0	4	0.901	-3.156
3	1	2	0.040	-6.543
4	1	3	0.424	-1.340
5	1	4	0.233	-3.655
6	1	5	0.393	-4.313
7	2	4	0.352	-4.788
8	2	5	0.507	0.317
9	3	4	0.025	-8.593
10	4	5	0.001	-6.438
Суммарные потери			4.460	-42.265

Таблица 8 - Параметры КУ сети после оптимизации методом градиентного спуска

Номер узла	Реактивная мощность КУ, МВар
2	3.790
3	71.921
4	77.421
5	57.771
6	37.602

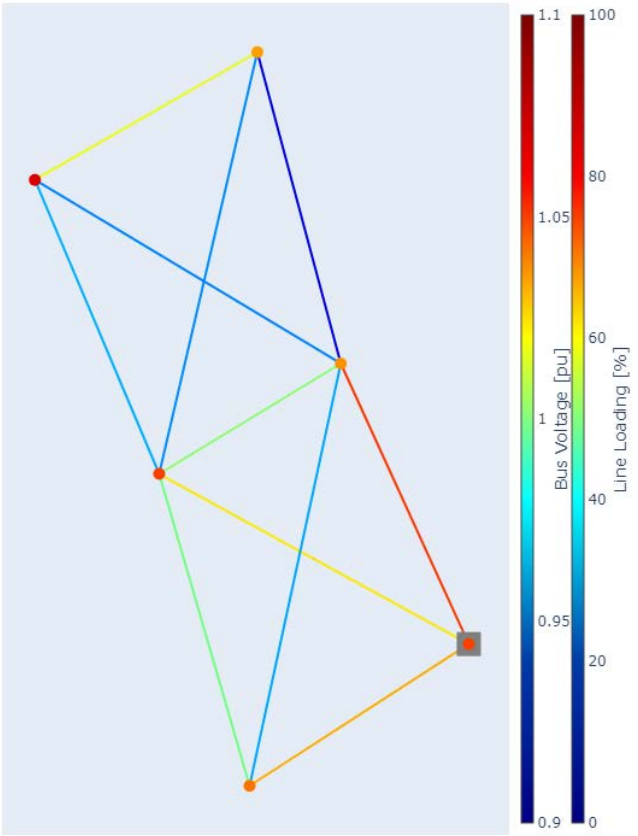

Рисунок 7 – Условная схема оптимизированной сети методом градиентного спуска с указанием напряжений в узлах и загрузки каждой линий

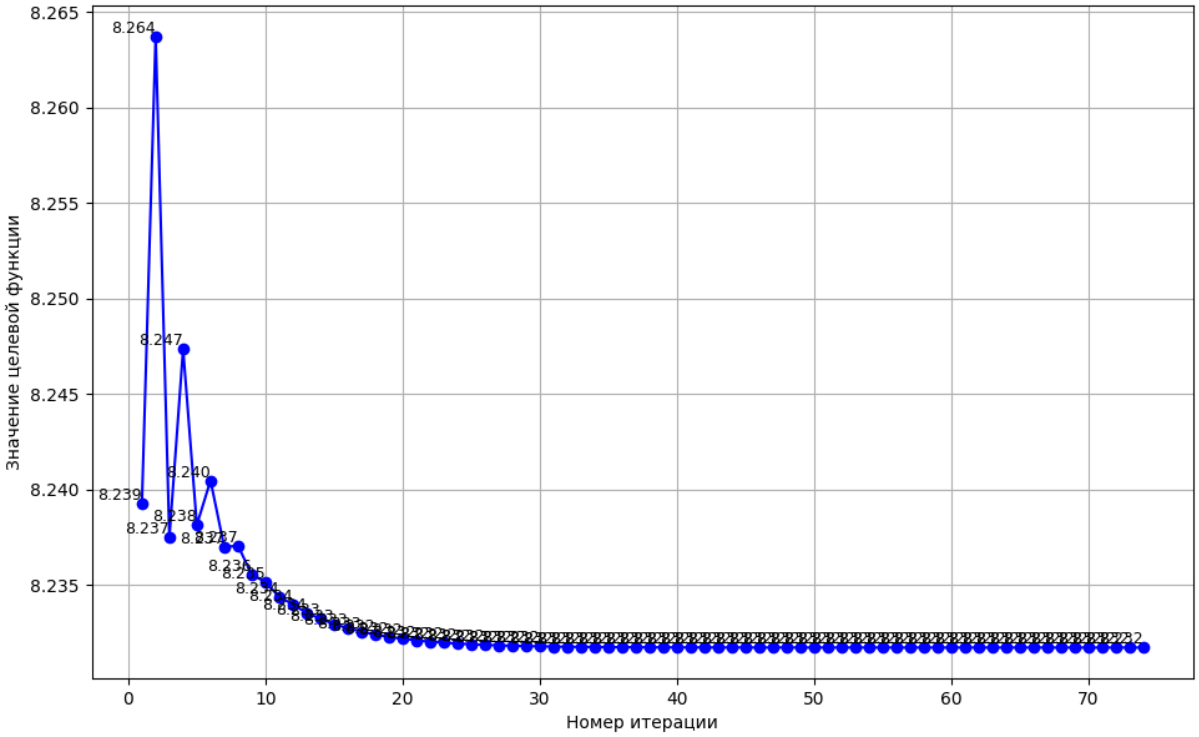

Рисунок 8 – График зависимости значения целевой функции от номера итерации в методе градиентного спуска (количество итераций = 200; a=0.05, delta=0.05)

По результатам работы МГС можно сделать вывод, что данный метод оптимизации показал близкие результаты в сравнения с методом роя частиц. Но при это стоит отметить, что данный способ оптимизации является менее универсальным, так как требует точной настройки параметров. А также в процессе тестирования его работы была подтверждена его особенность «падения» в локальный минимум. Что ещё раз доказывает важность индивидуальной настройки его параметров перед запуском работы алгоритма.

В работе также был использован другой эвристический метод, а именно – генетический алгоритм.

Сначала для работы алгоритма эмпирическим путем формируется и задается оптимальная начальная популяция особей с указанием их количества = 50, числом поколений = 30, количеством «родителей» = 20 и вероятностью мутации = 0.3.

Как и в прядущих методах, производится расчет режима, и на основании результатов расчета, вычисляется целевая функция.

На основании полученного значения целевой функции производится проверка: достигло ли значение целевой функции необходимой величины функции приспособленности = 0.1, либо номер поколения равен указанному общему числу поколений. При недостижении обоих условий частицы проходят селекцию (отбираются лучшие кандидаты), производится скрещивание и мутация для особей с вероятностью мутации < 0.3.

Ниже приведена общая блок-схема для генетического алгоритма (рисунок 9).

В таблицах 9–11 предоставлены значения параметров сети после оптимизации данным методом. А также, на рисунке 10 представлена схема сети с цветовым отображением уровня напряжения в каждом узле в о.е. и загрузкой каждой линии в процентах после выполнения оптимизации. На рисунке 11 изображен график изменения значения целевой функции относительно номера итерации.

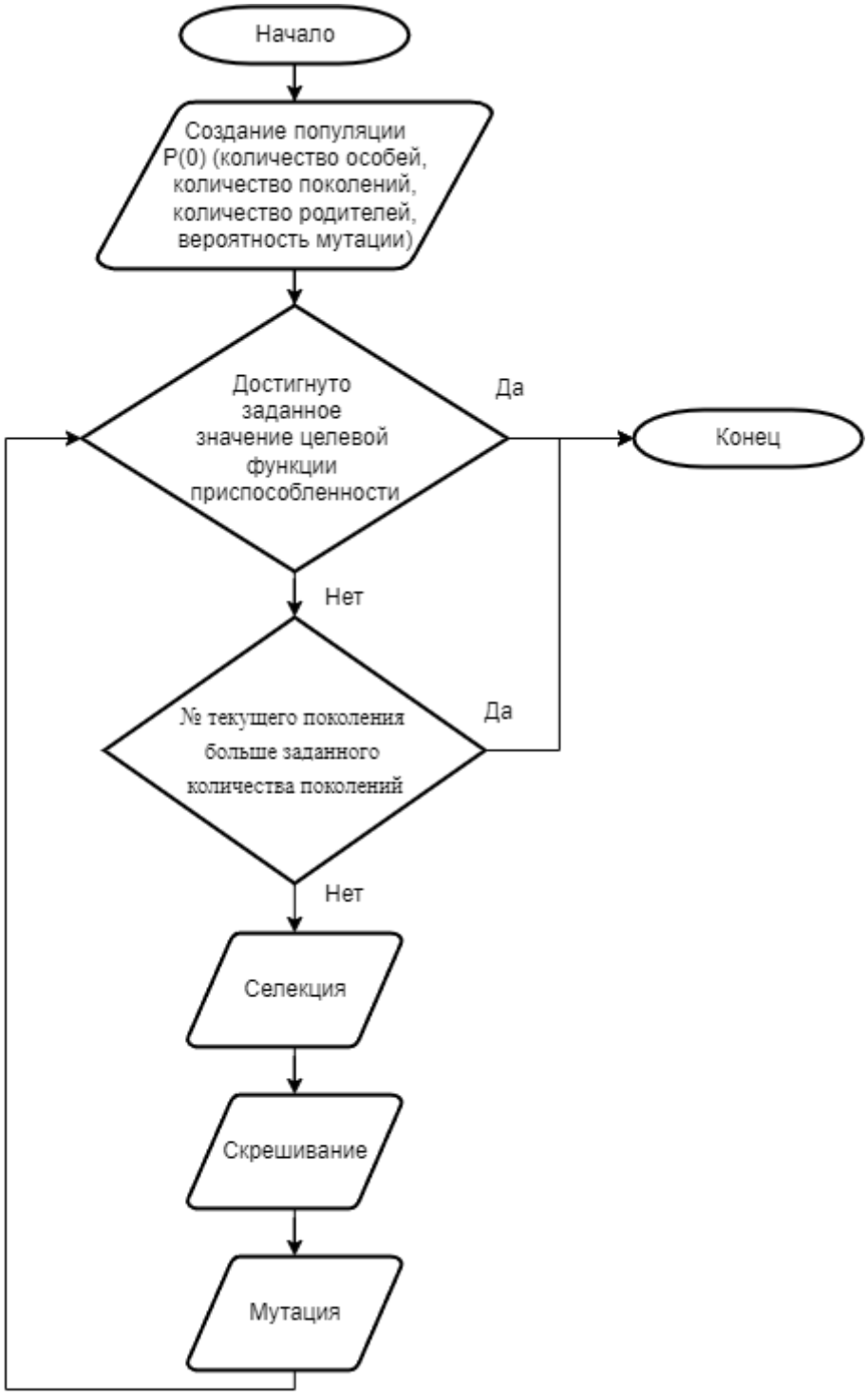

Рисунок 9 – Блок-схема работы генетического алгоритма

Таблица 9 – Параметры узлов сети после оптимизации с применением генетического алгоритма

Номер узла	Отношение напряжения в узле к номинальному, о.е.	Угол напряжения, °	Потребляемая активная мощность, МВт	Потребляемая реактивная мощность, МВар	Отклонение напряжения по модулю, о. е
1	1.050	0.000	-104.446	25.663	0.050
2	1.050	-3.136	-50.000	12.676	0.050
3	1.070	-3.726	-60.000	-35.988	0.070
4	1.040	-4.824	70.000	-4.619	0.040
5	1.035	-5.661	70.000	14.739	0.035
6	1.036	-5.786	70.000	29.817	0.036
Суммарное отклонение напряжения					0.282

Таблица 10 - Параметры линий сети после оптимизации с применением генетического алгоритма

Номер линии	Номер начального узла	Номер конечного узла	Активные потери в линии, МВт	Реактивные потери в линии, МВар
0	0	1	0.660	-3.089
1	0	3	0.922	-0.681
2	0	4	0.898	-3.156
3	1	2	0.040	-6.543
4	1	3	0.418	-1.347
5	1	4	0.233	-3.651
6	1	5	0.393	-4.319
7	2	4	0.361	-4.761
8	2	5	0.495	0.259
9	3	4	0.025	-8.564
10	4	5	0.001	-6.435
Суммарные потери			4.446	-42.288

Таблица 11 - Параметры КУ сети после оптимизации с применением генетического алгоритма

Номер узла	Реактивная мощность КУ, МВар
2	61.19281
3	29.83077
4	74.6187
5	55.26081
6	40.18315

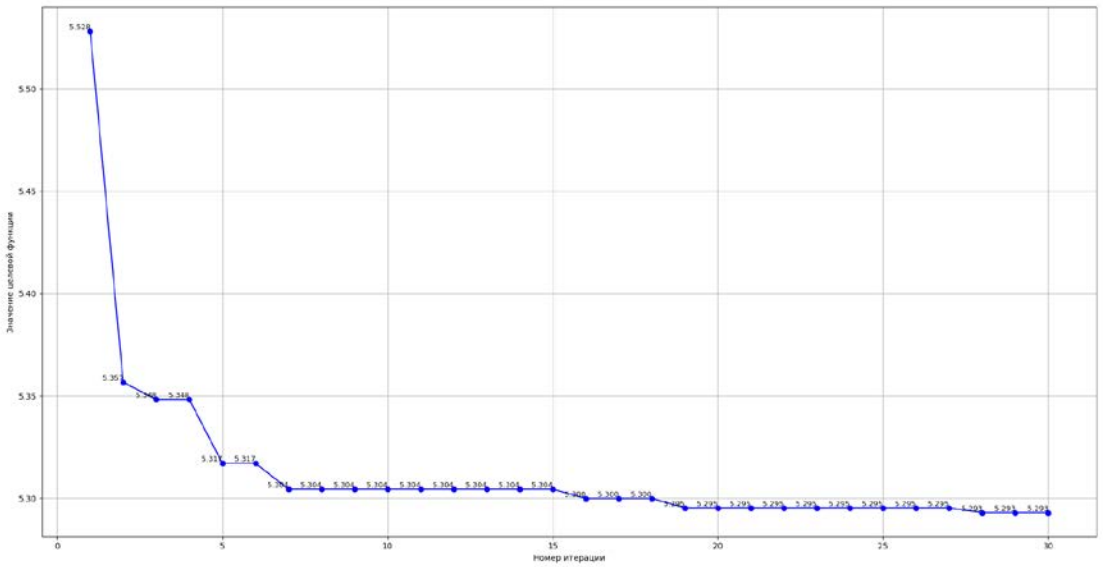

Рисунок 10 – График зависимости значения целевой функции от номера итерации в при использовании генетического алгоритма (количество особей = 50; количество поколений = 30; количество родителей = 20; вероятность мутации = 0.3)

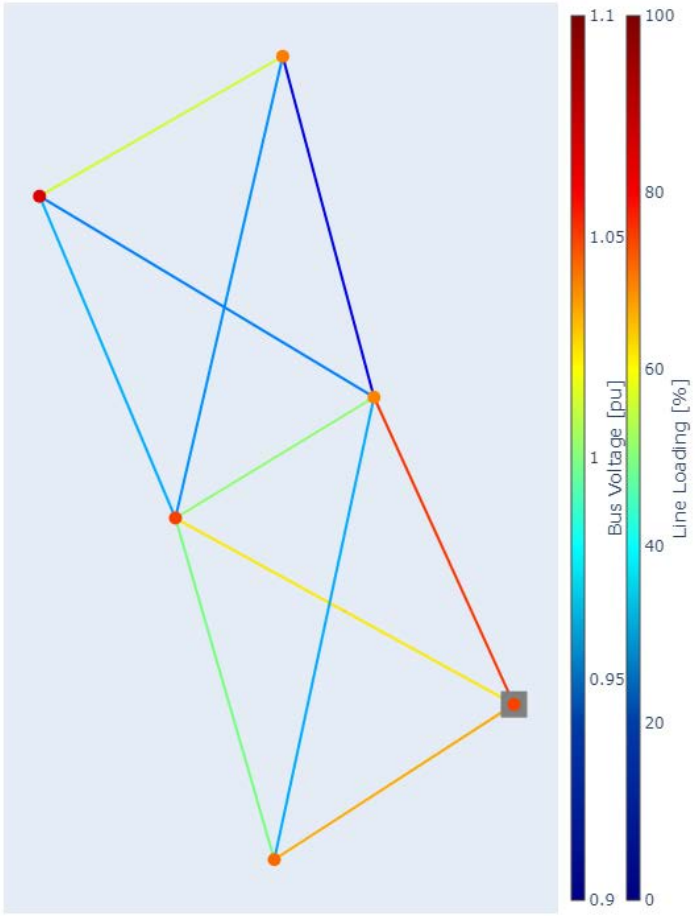

Рисунок 11 – Условная схема оптимизированной сети после применения генетического алгоритма с указанием напряжений в узлах и загрузки каждой линий

Данный алгоритм показал хорошую степень сходимости и высокую скорость вычисления. После оптимизации ЭС данным методом потери активной мощности в линиях снизились на 56.45% или на 3.43 МВт.

3.3 Апробация алгоритма на примере сети CIGRE-case-12

Для дополнительной проверки работы выбранных методов оптимизации также была выбрана другая сеть, разработанная Целевой группой CIGRE C6.04.02 [36]. Данная сеть имеет 12 узлов с напряжением 380, 220 и 22 кВ. В ней присутствуют 3 генератора в узлах 10, 11 и 12 мощность 500, 200 и 300 МВт соответственно. Схема сети представлена на рисунке 12. Также на рисунке 13 представлена условная цветовая схема сети с указанием загрузки линий и отклонением напряжения. Параметры узлов и линий сети приведены в таблицах 12–13

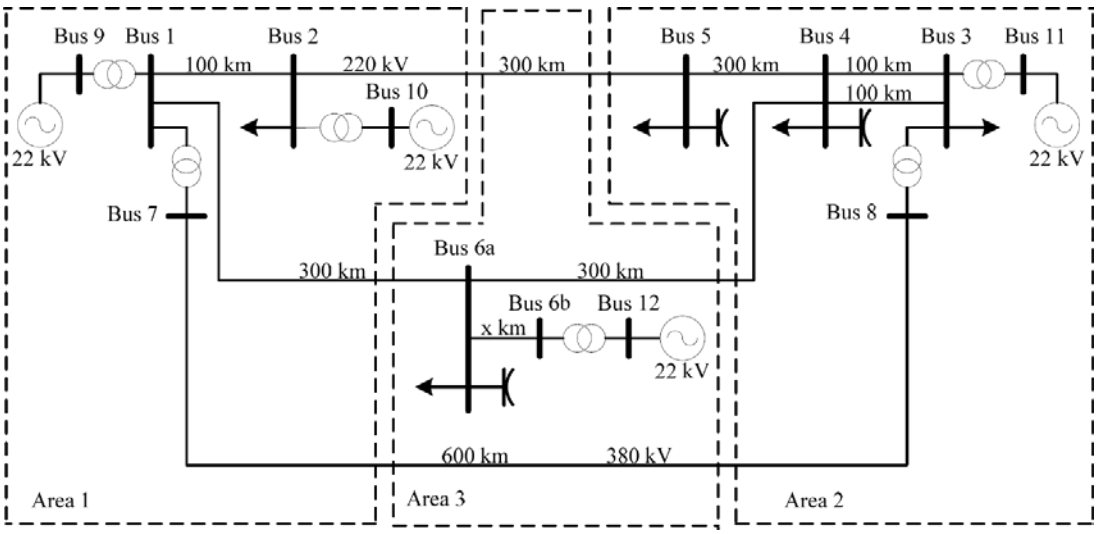

Рисунок 12 – Схема сети CIGRE-case-12

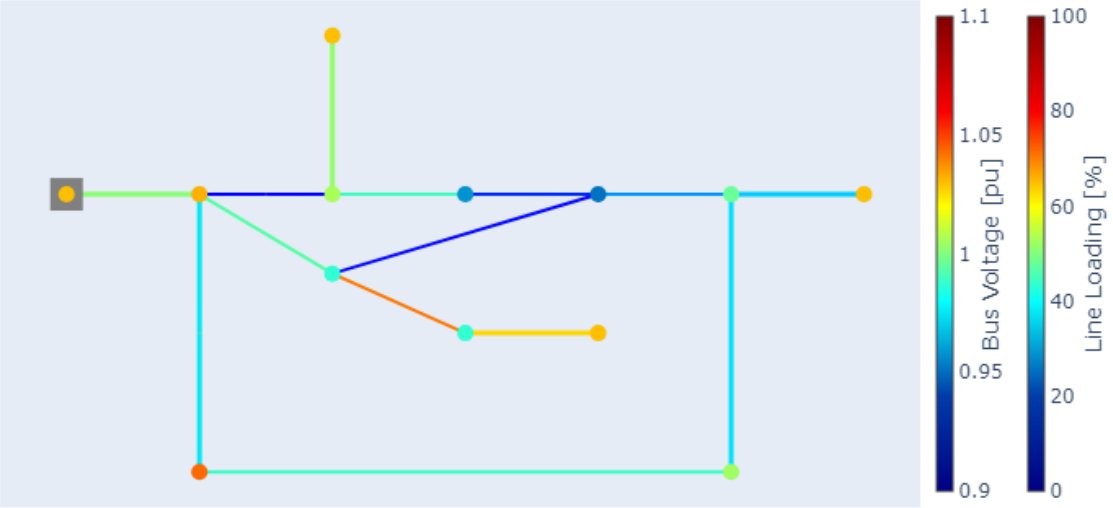

Рисунок 13 – Условная схема CIGRE-case-12 сети до оптимизации с указанием напряжений в узлах и загрузки каждой линий

Таблица 12 – Параметры узлов сети после оптимизации с применением генетического алгоритма

Номер узла	Отношение напряжения в узле к номинальному, о.е.	Угол напряжения, °	Потребляемая активная мощность, МВт	Потребляемая реактивная мощность, МВар	Отклонение напряжения по модулю, о. е
1	1.032	-33.696	0.000	0.000	0.032
2	1.006	-31.969	285.000	200.000	0.006
3	0.996	-64.347	325.000	244.000	0.004
4	0.951	-69.385	326.000	99.282	0.049
5	0.958	-59.009	103.000	-11.376	0.042
6	0.987	-61.976	435.000	120.660	0.013
7	0.987	-61.963	0.000	0.000	0.013
8	1.044	-36.170	0.000	0.000	0.044
9	1.005	-61.809	0.000	0.000	0.005
10	1.030	0.000	-527.366	2.186	0.030
11	1.030	1.627	-500.000	-203.584	0.030
12	1.030	-32.895	-200.000	-270.096	0.030
Суммарное отклонение напряжения по модулю					0.328

Таблица 13 - Параметры линий сети после оптимизации с применением генетического алгоритма

Номер линии	Номер начального узла	Номер конечного узла	Активные потери в линии, МВт	Реактивные потери в линии, МВар
1	0	1	0.316	-12.421
2	0	3	15.887	54.583
3	0	4	13.798	44.132
4	1	2	1.819	-2.009
5	1	3	1.819	-2.009
6	1	4	1.932	-25.950
7	1	5	1.098	-32.210
8	2	4	16.680	-169.739
9	2	5	0.016	0.082
Суммарные потери			53.366	-145.542

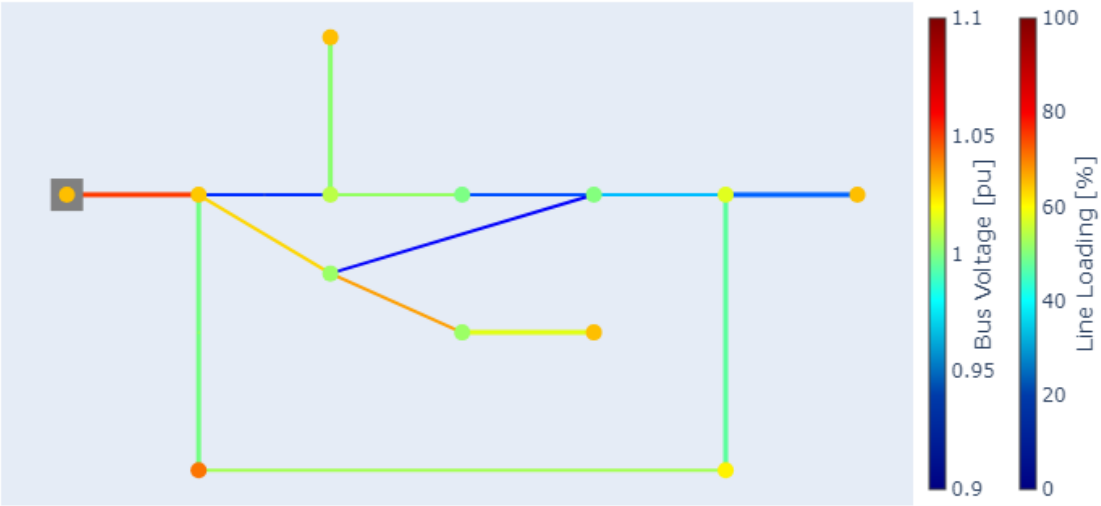

Рисунок 14 – Условная схема оптимизированной сети CIGRE-case-12 с применением генетического алгоритма с указанием напряжений в узлах и загрузки каждой линий

Как и для предыдущей модели сети для данной была выполнена оптимизация описанными ранее 3 методами. Помимо этого, для проверки работы методов были произведены расчеты нормального режима для обеих сетей, а также в режиме в режиме утяжеления, а именно с увеличением общей нагрузки. Для сети IEEE-case-6: от 5 до 50%; для CIGRE-case-12 – от 5 до 20%. Результаты данной работы представлены в таблице 14.

Таблица 14 – Общие результаты оптимизации установки КУ для схем IEEE-case-6 и CIGRE-case-12 в нормальном и утяжеленном режимах

	Процент увеличения нагрузки, %	ЭС без оптимизации		ЭС с оптимизацией методом роя частиц		ЭС с оптимизацией методом градиентного спуска		ЭС с оптимизацией генетическим алгоритмом	
		Потери активной мощности, Мвт	Отклонение напряжения, о.е.	Потери активной мощности, Мвт	Отклонение напряжения, о.е.	Потери активной мощности, Мвт	Отклонение напряжения, о.е.	Потери активной мощности, Мвт	Отклонение напряжения, о.е.
Сеть IEEE-case-6	0	7.876	0.200	4.428	0.284	4.460	0.284	4.446	0.282
	5	8.681	0.201	5.107	0.286	5.140	0.283	5.108	0.283
	10	9.570	0.202	5.858	0.285	5.893	0.282	5.850	0.287
	15	10.544	0.203	6.684	0.284	6.719	0.281	6.697	0.284
	20	11.603	0.205	7.583	0.283	7.618	0.280	7.580	0.286
	30	13.989	0.210	9.608	0.285	9.643	0.278	9.603	0.282
	40	16.743	0.217	11.940	0.279	11.977	0.276	11.947	0.275
	50	19.882	0.225	14.591	0.278	14.631	0.274	14.583	0.280
Сеть CIGRE-case-12	0	53.366	0.328	51.322	0.252	51.125	0.275	50.952	0.315
	5	64.687	0.367	61.364	0.243	61.178	0.253	61.136	0.257
	10	78.723	0.426	72.168	0.260	72.341	0.254	72.567	0.242
	15	96.767	0.506	84.862	0.285	85.494	0.253	86.054	0.227
	20	122.227	0.624	101.092	0.217	101.010	0.234	101.354	0.221

Как можно видеть из таблицы 14 при увеличении нагрузки в сети методы оптимизации увеличивают свой показатель эффективности. Если в нормальном режиме в оптимизированной сети IEEE-case-6 потери активной мощности в сети уменьшились на примерно 56%, то при увеличении нагрузки в 1.5 раза данный показатель снизился до 27% в процентном соотношении, но увеличился в абсолютном – с 3.4 до 5.3 МВт. При этом можно наблюдать, что суммарное отклонение напряжения во всех узлах оптимизированной сети остается практически неизменным, в то время как, в сети без оптимизации этот показатель возрастает с увеличением процента нагрузки.

Касательно сети CIGRE-case-12, то в её нормальном режиме в сети после оптимизации суммарные потери уменьшились на 5% или 2 МВт, в сравнении с сетью до оптимизации. И притом величина отклонения напряжения также снизилась на 23%. А при увеличении нагрузки в 1.2 раза оптимизации дала снижение показателя суммарных потерь активной мощности на 17% и на 65% для показателя отклонения напряжения.

3.4 Выводы по главе 3

Разработанный алгоритм и программное обеспечения апробированы на двух известных тестовых примерах. Для сети IEEE-case-6 потери активной мощности в линиях снизились на 56.22% или на 3.448 МВт.

Для сети CIGRE-case-12 суммарные потери уменьшились на 5% или 2 МВт, а величина отклонения напряжения также снизилась на 23%.

Тестирования методов при утяжеленном режиме работы сети показали высокую эффективность и адаптивность представленных методов.

ЗАКЛЮЧЕНИЕ

В данной на основе анализа существующих методов оптимального размещения компенсирующих устройств в электрической сети обоснован выбор метаэвристических популяционных алгоритмов с учетом многокритериальности задачи.

Выполнена разработка и программная реализация адаптивных алгоритмов роевого интеллекта для решения задачи выбора узлов размещения компенсирующих устройств и их мощностей для снижения потерь активной мощности и отклонений напряжения.

Разработан программный код на языке Python с применением библиотеки Pandapower было апробировано на двух известных тестовых примерах из наборов бенчмарков IEEE и СИГРЕ. Полученные результаты подтвердили эффективность адаптивного алгоритма роя частиц для решения рассматриваемой задачи.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Карпов Ф. Ф. Компенсация реактивной мощности в распределительных сетях / Ф. Ф. Карпов. // М. : Энергия, 1975. 185с.
2. Elsheikh A., Helmy Y., Abouelseoud Y., Elsherif A. Optimal capacitor placement and sizing in radial electric power systems. Hosted by Elsevier B. V. on behalf of faculty of engineering, Alexandria university journal 53, the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/3.0/). 2014. pp. 809-816.
3. Железко Ю. С. Потери электроэнергии; Реактивная мощность; Качество электроэнергии. М: Издательский НЦ ЭНАС. 2009. ISBN 978-5-93196-958-9.
4. Haghifam M. R., and Malik O. P, “Genetic Algorithm-based approach for fixed and switchable capacitors placement in distribution systems with uncertianty and time varying loads”, IET Generation, Transmission and Distribution, pp. 244-252, 2007.
5. Kokin S., Manusov V., Matrenin P. (2018): Optimal Placement of Reactive PowerSources in Power Supply Systems Using Particle Swarm Optimization and ArtificialBees Colony Optimization. IEEE Conference.
6. Толба,М.А.Х. Развитие методов оптимизации размещения компе нсирующих устройств и возобновляемой распределенной генерац ии в радиальных электрических сетях / дисс. на соиск. уч. ст. канд. техн. наук, МЭИ, Москва, 2018.
7. В. Н. Костин.: Оптимизационные задачи электроэнергетики: Учеб. пособие. - СПб.: СЗТУ, 2003 - 120 с.
8. Ерошенко С.А. Расчеты допустимых перетоков мощности в энергосистемах: учебное пособие / С. А. Ерошенко, А. О. Егоров, В. О. Самойленко [и др.]; научный редактор А. А. Суворов;

Министерство образования и науки Российской Федерации, Уральский федеральный университет имени первого Президента России Б.Н. Ельцина. – Екатеринбург : Издательство Уральского университета, 2017. – 86, [2] с. – ISBN 978-5-7996-1994-7.

9. Neagle N.M., Samson D.R. Loss reduction from capacitors installed on primary feeders. Power Apparatus Syst Part III Trans Am Inst Electr Eng. 1956.
10. Cook R.F. Analysis of capacitor application as affected by load cycle. Power Apparatus Syst Part III Trans Am Inst Electr Eng. 1959.
11. Grainger J.J., Lee S.H. Optimum size and location of shunt capacitors for reduction of losses on distribution feeders. IEEE Transaction on Power Apparatus and Systems. 1981. PAS-100 (3):1105–18.
12. Grainger J.J., Civanlar S., Lee S.H. Optimal design and control scheme for continuous capacitive compensation of distribution feeders. IEEE Transactions on Power Apparatus and Systems. 1983. PAS-102(10):3271–8.
13. Хохлова, Т. Е. Оценивание параметров RL-цепей электромеханических систем в режиме функционирования на основе метода покоординатного спуска [Текст] / Т. Е. Хохлова, А. С. Глазырин, В. И. Полищук // Известия Томского политехнического университета. - 2013. - Т. 322, № 2 : Математика и механика : Физика. - С. 41-46 : ил. - Библиогр.: с. 46 (9 назв.) . - ISSN 1684-8519
14. Гасников А. В. Современные численные методы оптимизации. Метод универсального градиентного спуска : учебное пособие. — М.: МФТИ, 2018. — 291 с. — ISBN 978-5-7417-0667-1.
15. Маркман, Г. З. Энергоэффективность преобразования и транспортировки электрической энергии / Г. З. Маркман. – Томск : Изд-во Томского политехнического университета, 2008. – 184 с.

16. Najafi, Arsalan & Masoudian, Ali & Mohammadi-ivatloo, Behnam. (2020). Optimal Capacitor Placement and Sizing in Distribution Networks. 10.1007/978-3-030-34050-6_4.
17. Химмельблау Д. Прикладное нелинейное программирование. М: Мир, 1982.
18. Ефременко Владимир Михайлович, Беляевский Роман Владимирович Расчет оптимального размещения компенсирующих устройств методом множителей Лагранжа // Вестник КузГТУ. 2012. №6(94).
19. Wang, Guanglei & Hijazi, Hassan. (2018). Mathematical programming methods for microgrid design and operations: a survey on deterministic and stochastic approaches. Computational Optimization and Applications. 71. 10.1007/s10589-018-0015-1.
20. Холмский, В. Г. Решение проектной задачи оптимального распределения реактивных мощностей методом потенциалов затрат / В. Г. Холмский, Ю. В. Щербина, С. В. Колесников. – В кн. : Электрические сети и системы. – Львов : Вища школа, 1968. – Вып. 4. – С. 6-9.
21. Александров, О. И. Уменьшение потерь в сложнозамкнутой электрической сети путем компенсации реактивных мощностей нагрузок / О. И. Александров, Л. П. Падалко, Н. Н. Никольская. – В кн. : Опыт планирования, анализа потерь энергии и разработка мероприятий по их снижению в энергосистеме. – Минск : Вышейшая школа, 1974. – С. 65-71.
22. Prakash K., Sydulu M. Particle Swarm Optimization Based Capacitor Placement on Radial Distribution System. IEEE Power engineering society general meeting. 2007. pp. 1-5.
23. Swarup K.S. Genetic algorithm for optimal capacitor allocation in radial distribution systems. Proceedings of the 6th WSEAS

international conference on evolutionary, Lisbon, Portugal. 2005. pp. 152-159.

24. Das D. Optimal placement of capacitors in radial distribution system using a fuzzy-GA method. Electrical Power & Energy Systems. 2008. vol 30. Issue 6-7. pp. 361-367.

25. Матренин П. В. Оптимизация распределения источников реактивной мощности в системах электроснабжения с помощью алгоритмов роевого интеллекта. / П. В. Матренин, В. З. Манусов, Е. С. Третьякова // Оперативное управление в электроэнергетике. Подготовка персонала и поддержание его квалификации. - 2016. - № 2. - С. 24-29.

26. Сысоева Ю.И. Исследование вопросов оптимального размещения устройств компенсации реактивной мощности в многокритериальной постановке задачи: статья в сборнике статей . / Сысоева, Юлия Игоревна; Поповцев, Владислав Викторович // Актуальные проблемы развития технических наук : сборник статей участников XXV Областного конкурса научно-исследовательских работ «Научный Олимп» по направлению «Технические науки»: сборник статей. Екатеринбург: Федеральное государственное автономное образовательное учреждение высшего профессионального образования "Уральский федеральный университет им. первого Президента России Б.Н. Ельцина", 2022. стр. 113-118.

27. Aman M.M., Jasmon G.B., Bakar A.H.A., Mokhlis H., Karimi M.Optimum shunt capacitor placement in distribution system—A review and comparativestudy. Elsevier. Renewable and Sustainable Energy Reviews journal. 2014. № (30). pp. 429-439.

28. J. Kennedy and R. Eberhart, "Particle swarm optimization," Proceedings of ICNN'95 - International Conference on

Neural Networks, Perth, WA, Australia, 1995, pp. 1942-1948 vol.4, doi: 10.1109/ICNN.1995.488968.

29. Иванов, И.А. Самоконфигурируемый генетический алгоритм решения задач поддержки многокритериального выбора / И.А. Иванов, Е.А. Сопов // Сибирский журнал науки и технологий. - 2013. -№1 (47). -С. 30-35.

30. Malczewski, Jacek; Rinner, Claus (2015). Multicriteria Decision Analysis in Geographic Information Science. doi:10.1007/978-3-540-74757-4. ISBN 978-3-540-86875-0. S2CID 126734355

31. Дородных, Н.О & Dorodnykh, N.O. & Юрин, А.Ю & Yurin, Alexander & Коршунов, С.А & Korshunov, S.A.. (2018). Средства поддержки моделирования логических правил в нотации RVML. Международный журнал Программные продукты и системы. 29. 667-672. 10.15827/0236-235X.124.667-672.

32. Zhu, Quanmin & Azar, Ahmad Taher. (2015). Complex System Modelling and Control Through Intelligent Soft Computations. 10.1007/978-3-319-12883-2.

33. E. V. Vinay, Optimal Placement of Compensating Devices in Distribution System by Using PSO Algorithm, B. Tech EEE, MERITS Engineering College, Udayagiri, Nellore (dist), A.P, India

34. Муратаева Г.А., Муратаев И.А. Разработка метода диагностики электроизоляционной жидкости трансформаторного электрооборудования // Известия высших учебных заведений. ПРОБЛЕМЫ ЭНЕРГЕТИКИ. 2019. Т.21. No4. С.41-47. doi:10.30724/1998-9903-2019-21-4-41-47

35. Manusov, Vadim & Matrenin, Pavel & Kirgizov, Alifbek. (2017). Swarm optimization for reactive power control in electrical grids. Energy-Safety and Energy-Economy. 3. 28-32. 10.18635/2071-2219-2017-3-28-32.

36. K. Rudion, A. Orths, Z. A. Styczynski and K. Strunz, Design of benchmark of medium voltage distribution network for investigation of DG integration 2006 IEEE Power Engineering Society General Meeting, Montreal, 2006

ПРИЛОЖЕНИЕ А

Программный код:

A.1. genetic_algorithm.py

```
import pandapower as pp
import pandapower.networks as nw
import numpy as np
import matplotlib.pyplot as plt


def create_initial_population(pop_size, num_sources):
    population = np.random.rand(pop_size, num_sources)
    return population


def evaluate_fitness(network, individual, start_loss):
    for i, bus in enumerate(network.bus.index):
        network.sgen.loc[network.sgen.bus == bus, 'q_mvar'] = individual[i] *
100

    pp.runpp(network, numba=False)
    power_loss = np.sum(network.res_line.pl_mw / start_loss)
    voltage_deviation = np.sum(abs(network.res_bus.vm_pu - 1.0)) * 10
    total_loss = 0.7 * power_loss + 0.3 * voltage_deviation

    return total_loss


def selection(population, fitness_values, num_parents):
    parents = np.empty((num_parents, population.shape[1]))
    for parent_num in range(num_parents):
        min_fitness_idx = np.argmin(fitness_values)
        parents[parent_num, :] = population[min_fitness_idx, :]
        fitness_values[min_fitness_idx] = float('inf')
    return parents


def crossover(parents, offspring_size):
    offspring = np.empty(offspring_size)
    crossover_point = np.uint8(offspring_size[1] / 2)

    for k in range(offspring_size[0]):
        parent1_idx = k % parents.shape[0]
        parent2_idx = (k + 1) % parents.shape[0]
        offspring[k, 0:crossover_point] = parents[parent1_idx,
0:crossover_point]
        offspring[k, crossover_point:] = parents[parent2_idx,
crossover_point:]

    return offspring


def mutation(offspring_crossover, mutation_rate=0.1):
    for idx in range(offspring_crossover.shape[0]):
        for gene_idx in range(offspring_crossover.shape[1]):
            if np.random.rand() < mutation_rate:
                random_value = np.random.rand()
                offspring_crossover[idx, gene_idx] = random_value
    return offspring_crossover


def genetic_algorithm(network, pop_size, num_sources, num_generations,
```

```
num_parents_mating, start_loss, mutation_rate):
    population = create_initial_population(pop_size, num_sources)
    best_output = []
    for generation in range(num_generations):
        fitness_values = np.array([evaluate_fitness(network, individual,
start_loss) for individual in population])
        best_output.append(np.min(fitness_values))
        parents = selection(population, fitness_values, num_parents_mating)
        offspring_crossover = crossover(parents, (pop_size -
parents.shape[0], num_sources))
        offspring_mutation = mutation(offspring_crossover, mutation_rate)
        population[0:parents.shape[0], :] = parents
        population[parents.shape[0]:, :] = offspring_mutation

    best_fitness_idx = np.argmin([evaluate_fitness(network, individual,
start_loss) for individual in population])
    best_solution = population[best_fitness_idx, :]

    return best_solution, best_output


def plot_graph_iterations(iterations, values):
    plt.figure(figsize=(10, 6))
    plt.plot(iterations, values, marker='o', linestyle='-', color='b')
    for i, value in enumerate(values):
        plt.text(iterations[i], value, f'{value:.3f}', fontsize=9,
ha='right', va='bottom')
    plt.xlabel('Номер итерации')
    plt.ylabel("Значение целевой функции")
    plt.grid(True)
    plt.show()


def main(network, pw):
    print(pw)
    network.load['p_mw'] = [p_mw * pw for p_mw in network.load['p_mw']]
    num_sources = len(network.bus.index)

    for i in range(num_sources):
        pp.create_sgen(network, bus=i, p_mw=0.0, q_mvar=0.0)

    pp.runpp(network, numba=False)
    print(f'Суммарные потери мощности в неоптимизированной
сети:\n{np.sum(network.res_line.pl_mw)}\n')
    print(f'Суммарные отклонение напряжения в неоптимизированной
сети:\n{np.sum(abs(network.res_bus.vm_pu - 1.0))}\n')
    pp.plotting.plotly.pf_res_plotly(network)
    start_loss = network.res_line.pl_mw
    power_loss = np.sum(start_loss)
    voltage_deviation = np.sum((network.res_bus.vm_pu - 1.0) ** 2)
    total_loss = 0.7 * power_loss + 0.3 * voltage_deviation

    losses = [total_loss]

    pop_size = 50  # количество особей в популяции, определяет разнообразие
решений
    num_generations = 30  # количество поколений, которое будет выполнено,
влияет на время работы и глубину поиска
    num_parents_mating = 20  # количество родителей, отбираемых для
кроссовера, определяет количество новых решений
    mutation_rate = 0.3  # вероятность мутации

    best_solution, list_losses = genetic_algorithm(network, pop_size,
num_sources, num_generations, num_parents_mating,
```

```
                                                    start_loss, mutation_rate)

    iterations = [int(i) for i in range(1, num_generations + 1)]
    losses = list_losses
    plot_graph_iterations(iterations, losses)

    loss = evaluate_fitness(network, best_solution, start_loss)
    print("Optimal active power sources:")
    print(f'Суммарные потери мощности в оптимизированной
сети:\n{np.sum(network.res_line.pl_mw)}\n')
    print(f'Суммарные отклонение напряжения в оптимизированной
сети:\n{np.sum(abs(network.res_bus.vm_pu - 1.0))}\n')

    # отрисовка схемы сети со степенью загрузки каждой линии и уровнем
напряжения
    pp.plotting.plotly.pf_res_plotly(network)


if __name__ == "__main__":
    # for i in range(5, 40, 5):
    net = nw.create_cigre_network_hv(length_km_6a_6b=0.1)
    main(net, 1 + 20 * 0.01)
```

A.2. pso_net.py

```
import pandapower as pp
import pandapower.networks as nw
import numpy as np
import matplotlib.pyplot as plt


def create_particle_swarm(num_particles, num_sources):
    particles = np.random.rand(num_particles, num_sources)
    return particles


def evaluate_fitness(network, particle, start_loss):
    # Расположение источников мощности в сети
    for i, bus in enumerate(network.bus.index):
        network.sgen.loc[network.sgen.bus == bus, 'q_mvar'] = particle[i] *
100

    # Расчет показателя эффективности
    pp.runpp(network, numba=False)  # запуск расчета установившегося режима
    power_loss = np.sum(network.res_line.pl_mw / start_loss)  # суммарные
потери по линиям в сети
    voltage_deviation = np.sum(abs(network.res_bus.vm_pu - 1.0)) * 10  #
отклонения напряжения (модуль отклонения)
    total_loss = 0.7 * power_loss + 0.3 * voltage_deviation

    return total_loss


def update_velocity_position(particles, velocities, best_particle,
global_best, w, c1, c2):
    for i in range(len(particles)):
        r1, r2 = np.random.rand(), np.random.rand()

        velocities[i] = w * velocities[i] + c1 * r1 * (best_particle[i] -
particles[i]) + c2 * r2 * (
                global_best - particles[i])
        particles[i] = particles[i] + velocities[i]
```

```
        # Ограничение на допустимые значения
        particles[i] = np.clip(particles[i], 0, 1)

    return particles, velocities


def particle_swarm_optimization(network, num_particles, num_sources,
max_iterations, w, c1, c2, start_loss):
    particles = create_particle_swarm(num_particles, num_sources)
    velocities = np.zeros_like(particles)
    best_particle = particles.copy()
    fitness_values = np.zeros(num_particles)

    global_best_index = np.argmin(fitness_values)
    global_best = particles[global_best_index].copy()
    losses = []
    for iteration in range(max_iterations):
        for i in range(num_particles):
            fitness_values[i] = evaluate_fitness(network, particles[i],
start_loss)

            if fitness_values[i] < evaluate_fitness(network,
best_particle[i], start_loss):
                best_particle[i] = particles[i].copy()

            if fitness_values[i] < evaluate_fitness(network, global_best,
start_loss):
                global_best = particles[i].copy()

        # print(f'Итерация № {iteration + 1}\nСтатические
генераторы:\n{network.res_sgen}\n')

        particles, velocities = update_velocity_position(particles,
velocities, best_particle, global_best, w, c1, c2)
        losses.append(evaluate_fitness(network, global_best, start_loss))

    return global_best, losses


def plot_graph_iterations(iterations, values):
    plt.figure(figsize=(10, 6))
    plt.plot(iterations, values, marker='o', linestyle='-', color='b')
    # Добавление значений около точек
    for i, value in enumerate(values):
        plt.text(iterations[i], value, f'{value:.3f}', fontsize=9,
ha='right', va='bottom')

    # plt.title(f'График оптимизации (кол-во частиц: {num_particles}, w: {w},
c1: {c1}, c2: {c2})')
    plt.xlabel('Номер итерации')
    plt.ylabel("Значение целевой функции")
    plt.grid(True)
    plt.show()


def main(network, pw):
    print(pw)
    network.load['p_mw'] = [p_mw * pw for p_mw in network.load['p_mw']]

    # Параметры алгоритма роя частиц
    num_particles = 20  # количество частиц [10, 100]
    num_sources = len(network.bus.index)  # количество узлов
    max_iterations = 30  # число итераций [100, 1000]
    # w - степень сохранения текущей скорости частицы при обновлении [0.4,
```

```
0.9]
    # c1 - коэффициент, который контролирует влияние лучшего положения
частицы на ее скорость [1, 2]
    # c2 - коэффициент, который контролирует влияние глобального лучшего
положения на скорость частицы [1, 2]
    w, c1, c2 = 0.6, 1.5, 1.5

    # добавление статических гераторов реактивной мощности
    for i in range(num_sources):
        pp.create_sgen(network, bus=i, p_mw=0.0, q_mvar=0.0)

    # простой расчет сети, без источников реактивной мощности
    res = pp.runpp(network, numba=False)

    start_loss = network.res_line.pl_mw
    power_loss = np.sum(start_loss)  # суммарные потери по линиям в сети
    voltage_deviation = np.sum((network.res_bus.vm_pu - 1.0) ** 2)  #
отклонения напряжения (сумма квадратов отклонения)
    total_loss = 0.7 * power_loss + 0.3 * voltage_deviation

    losses = [total_loss]

    print(f'First runpp:')
    # print(f'Результаты расчета для шин:\n{network.res_bus}\n')
    # print(f'Статические генераторы:\n{network.res_sgen}\n')
    # print(f'Результаты расчета для линий:\n{network.res_line}\n')
    print(f'Суммарные потери мощности в неоптимизированной
сети:\n{np.sum(network.res_line.pl_mw)}\n')
    print(f'Суммарные отклонение напряжения в неоптимизированной
сети:\n{np.sum(abs(network.res_bus.vm_pu - 1.0))}\n')

    # Запуск алгоритма роя частиц
    optimal_sources, list_losses = particle_swarm_optimization(network,
num_particles, num_sources, max_iterations, w,
                                                               c1, c2,
start_loss)

    iterations = [int(i) for i in range(1, max_iterations + 1)]
    losses = list_losses
    # plot_graph_iterations(iterations, losses)

    loss = evaluate_fitness(network, optimal_sources, start_loss)
    print("Optimal active power sources:")
    # print(f'Результаты расчета для шин:\n{network.res_bus}\n')
    # print(f'Статические генераторы:\n{network.res_sgen}\n')
    # print(f'Результаты расчета для линий:\n{network.res_line}\n')
    # print(f'Величина КУ:\n{optimal_sources}\n')
    print(f'Суммарные потери мощности в оптимизированной
сети:\n{np.sum(network.res_line.pl_mw)}\n')
    print(f'Суммарные отклонение напряжения в оптимизированной
сети:\n{np.sum(abs(network.res_bus.vm_pu - 1.0))}\n')

    # отрисовка схемы сети со степенью загрузки каждой линии и уровнем
напряжения
    # pp.plotting.plotly.pf_res_plotly(network)


if __name__ == '__main__':
    # Создание электроэнергетической сети
    for i in range(0, 40, 5):
        net = nw.create_cigre_network_hv(length_km_6a_6b=10)
        main(nw.create_cigre_network_hv(length_km_6a_6b=0.1), 1 + i * 0.01)
```

A.3. gradient_descent_method.py

```
import random

import pandapower as pp
import pandapower.networks as nw
import numpy as np
import matplotlib.pyplot as plt


# Функция для расчета целевой функции
def calculate_objective_function(net, start_loss):
    pp.runpp(net)
    power_loss = np.sum(net.res_line.pl_mw / start_loss)  # суммарные потери
по линиям в сети
    voltage_deviation = np.sum(abs(net.res_bus.vm_pu - 1.0)) * 10  #
отклонения напряжения (модуль отклонения)
    total_loss = 0.7 * power_loss + 0.3 * voltage_deviation
    return total_loss


def restriction_check(val):
    if val < 0:
        val = 0
    elif val > 1:
        val = 1

    return val


# Упрости это выражение:
# def restriction_check(val):
#     if val < 0:
#         val = 0
#     elif val > 1:
#         val = 1
#
#     return val

# Функция для вычисления градиента
def calculate_gradient(net, Q, start_loss, delta=0.05):
    gradient = []
    for i in range(len(Q)):
        Q_temp = Q.copy()
        Q_temp[i] += delta
        Q_temp[i] = restriction_check(Q_temp[i])
        net.sgen['q_mvar'] = Q_temp * 100
        f_plus = calculate_objective_function(net, start_loss)

        Q_temp[i] -= 2 * delta
        Q_temp[i] = restriction_check(Q_temp[i])
        net.sgen['q_mvar'] = Q_temp * 100
        f_minus = calculate_objective_function(net, start_loss)

        grad_i = (f_plus - f_minus) / (2 * delta)
        gradient.append(grad_i)

    return np.array(gradient)


def plot_graph_iterations(iterations, values):
    plt.figure(figsize=(10, 6))
    plt.plot(iterations, values, marker='o', linestyle='-', color='b')
    # Добавление значений около точек
```

```
    for i, value in enumerate(values):
        plt.text(iterations[i], value, f'{value:.3f}', fontsize=9,
ha='right', va='bottom')

    # plt.title(f'График оптимизации (кол-во частиц: {num_particles}, w: {w},
c1: {c1}, c2: {c2})')
    plt.xlabel('Номер итерации')
    plt.ylabel("Значение целевой функции")
    plt.grid(True)
    plt.show()


def main(net, pw):
    print(pw)
    net.load['p_mw'] = [p_mw * pw for p_mw in net.load['p_mw']]
    # Инициализация параметров
    for i in range(len(net.bus.index)):
        pp.create_sgen(net, bus=i, p_mw=0.0, q_mvar=0.0)

    pp.runpp(net)
    start_loss = net.res_line.pl_mw  # суммарные потери по линиям в сети
    print(f'Суммарные потери мощности в неоптимизированной
сети:\n{np.sum(net.res_line.pl_mw)}\n')
    print(f'Суммарные отклонение напряжения в неоптимизированной
сети:\n{np.sum(abs(net.res_bus.vm_pu - 1.0))}\n')
    objective_value = calculate_objective_function(net, start_loss)
    # print(f"Iteration {0}: Objective function = {objective_value}")

    Q_initial = net.sgen['q_mvar'].values
    Q = np.random.rand(len(net.bus.index))
    alpha = 0.05  # Шаг градиентного спуска
    max_iterations = 200
    tolerance = 1e-6

    losses = []
    iterations = [int(i) for i in range(1, max_iterations + 1)]
    # Основной цикл градиентного спуска
    for iteration in range(max_iterations):
        net.sgen['q_mvar'] = Q
        objective_value = calculate_objective_function(net, start_loss)
        losses.append(objective_value)
        gradient = calculate_gradient(net, Q, start_loss)

        Q -= alpha * gradient
        for i, q in enumerate(Q):
            Q[i] = restriction_check(q)

        # print(f"Iteration {iteration + 1}: Objective function =
{objective_value}, Q: {Q}")
        # print(f'np.linalg.norm(gradient): {np.linalg.norm(gradient)}')
        if np.linalg.norm(gradient) < tolerance:
            print("Convergence reached.")
            iterations = [int(i) for i in range(1, iteration + 2)]
            break

    # Вывод результатов
    print("Optimized reactive power of compensating devices:", Q)
    print("Final objective function value:",
calculate_objective_function(net, start_loss))
    print(f'Суммарные потери мощности в оптимизированной
сети:\n{np.sum(net.res_line.pl_mw)}\n')
    print(f'Суммарные отклонение напряжения в оптимизированной
сети:\n{np.sum(abs(net.res_bus.vm_pu - 1.0))}\n')
    # plot_graph_iterations(iterations, losses)
```

```
    # pp.plotting.plotly.pf_res_plotly(net)


if __name__ == '__main__':
    for i in range(0, 40, 5):
        network = nw.create_cigre_network_hv(length_km_6a_6b=0.1)
        main(network, 1 + i * 0.01)
```