МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Крымский федеральный университет имени В. И. Вернадского»
(ФГАОУ ВО «КФУ им. В. И. Вернадского»)
Таврическая академия (структурное подразделение)
Факультет математики и информатики
Кафедра прикладной математики

Консманов Алексей Витальевич

Использование мультиагентного подхода в военном деле

Курсовая работа

Обучающегося 1 курса
Направления подготовки 01.04.04. Прикладная математика
Форма обучения очная

Научный руководитель
доцент кафедры прикладной
математики, к. ф.-м. н. Ю.Ю. Дюличева

Симферополь 2020

Оглавление

Введение

Системы противоракетной обороны играют важную роль в обеспечении защиты государства от баллистических ракет и позволяют поддерживает стратегический паритет. Математическое моделирование таких систем представляет собой сравнительную дешевую в терминах людских и материальных ресурсов и безопасную для окружающей среды и людей альтернативу проведению реальных испытаний. Как правило, системы противоракетной обороны (далее ПРО) представляют собой компле́ксную систему, имеющую нелинейный характер и состоящую из множества подсистем реального времени, работающих параллельно. Необходимость точного описания порядка взаимодействия компонентов системы и характера этих взаимодействий и их результата является ключевым важным условием оценки эффективности всей системы ПРО.

Следует заметить, что многие современные системы являются многослойными, то есть содержат не просто отдельные компоненты, выполняющие действия на отдельном этапе, а целые подсистемы, выполняющие эти действия.

Область данной работы не является принципиально новой и неисследованной и уже многие авторы внесли свой вклад в разработку построения максимально точных моделей, способных описать систему ПРО. Большинство работ, связанных с данной темой использует один из следующих подходов: традиционное детерминированное моделирование с использованием систему дифференциальных уравнений, инженерный подход к описанию системы, вычислительный эксперимент. Однако, упомянутые выше подходы имеют ряд недостатков и не могут описать внутреннюю сложность систем и связей между ними. В противоположность этим классическим методам, мультиагентное моделирование предоставляет подход к моделированию по принципу «сверху вниз», т.е. заменяет сложный подход описания всей системы с помощью одной системы уравнений описанием компонентов и связей между ними, что и формирует полное описание системы, что в свою очередь является более подходящим подходом при моделировании компле́ксных систем.

Этот интересный и новый метод не мог остаться незамеченным и уже рассмотрен в ряде работ: Все эти работы нацелены на решение за-

дач управления, но не решают задачу автономного принятия решения. Решение задачи принятия решения в данных условиях особенно интересно, т.к система ПРО представляет собой систему реального времени, и особенно важным ограничением является ограничение по времени на принятие решения – обычно, полный жизненный цикл баллистической ракеты составляет около 28 минут. Данное ограничение по времени накладывает условие на временну́ю сложность алгоритма принятия решения – он должен завершиться и завершиться намного раньше, чем это временное ограничение, т.к. помимо принятия решения необходимо также провести пуск ракеты-перехватчика и дождаться поражения или не поражения ей цели.

Таким образом, ожидается, что построение системы принятия решения улучшит характеристики моделируемой системы, ровно как и оптимизация этой системы принятия решений еще боле увеличит эффективность системы в целом. Ли в своей работе уже предложил использование модифицированного алгоритма метода роя частиц.

Целью данной работы является реализация алгоритма принятия решений для системы ПРО, предназначенной для перехвата межконтинентальных баллистических ракет, и тестирование этой системы на имитационном ПО. Для этого необходимо решить следующий комплекс задач:

- Описать систему противоракетной обороны для которой будет строиться модель;
- описать агентов;
- интегрировать описанных агентов в модель системы ПРО;
- описать систему принятия решений и накладываемые на нее ограничения;
- описать алгоритм принятия решений и провести его оптимизацию;
- провести тестирование полученного алгоритма в имитационном ПО;
- проанализировать результаты и сделать выводы.

Для решения поставленного комплекса задач использовались методы математической статистики и теории вероятности, дискретной ма-

тематики, математического анализа и теоретической механики. Разработанная модель основывается на методах агентного / мультиагентного моделирования.

Объект исследования: изучение алгоритмов оптимизации методом роя частиц.

Предмет исследования: построение и анализ модели ПРО, имеющей блок принятия решений, основанный на оптимизации методом роя частиц.

Практическая ценность работы: полученный алгоритм может быть использован как база для разработки более точных алгоритмов, специфичных для конкретных средств перехвата и их целей. Данный алгоритм способен имитировать перехват целей с заданными пространственно-скоростными ограничениями средствами с аналогичными ограничениями и описывает систему ПРО с фиксированным количеством компонентов.

В первой части данной работы рассматриваются системы принятия решения для системы ПРО; во второй анализируется предложенный алгоритм на основании метода роя частиц; в третьей части описана реализация предложенного алгоритма и оцениваются результаты его работы.

1 Мультиагентное моделирование системы поддержки принятия решений в составе системы противоракетной обороны

1.1 Описание системы ПРО

Как правило, процесс противоракетной обороны (далее ПРО) состоит из последовательности этапов, включающих ранее обнаружение, отслеживание, распознавание, принятие решения и непосредственный перехват цели, достигаемый посредством взаимодействия всех компонентов системы ПРО (далее СПРО).

Не вдаваясь в технические подробности и устройство составных компонентов СПРО, отметим только сами компоненты и функции, выполняемые этими компонентами:

- Радар раннего обнаружения фиксирует факт запуска ракет и получает приблизительные данные о положении и скорости;
- отслеживающий радар «ведёт» ракету-перехватчик до завершения перехвата или промаха, получая инструкции из командного центра и передавая их ракете-перехватчику;
- командный центр выполняет роль коммуникационного звена и центра обработки информации; производит оценку траектории ракеты, на основании данных от радара раннего обнаружения; отправляет информацию отслеживающему радару; создание (генерация) плана перехвата ракеты и отправка его ракете-перехватчику в подходящий момент времени;
- ракета-перехватчик выполняет задачу перехвата и способна к ограниченному маневрированию.

Очевидно, что для полного цикла работы СПРО необходима и ракета-цель.

1.2 Агентное моделирование и понятие «агента»

Понятие «агент» не является строго установленным, но в общем случае можно говорить об «агенте» как о сущности, имеющей активность, автономное поведение, способность к самостоятельному приему решений в соответствии с некоторой заранее заданной совокупностью правил, способность к взаимодействию с окружающей средой и другими агентами (если такие существуют в рамках модели). Мультиагентные же модели в свою очередь используют множества таких агентов для построения процессов, где поведение системы возникает как следствие взаимодействия множества агентов, а не поведение системы описывает поведение каждого агента, т.е происходит моделирование «снизу вверх».

В рамках данной работы агент будет иметь следующую структуру:

- сенсор – получает информацию из внешнего мира в рамках радиуса восприятия;
- контроллер – ядро агента, состоящее из базы априорных знаний, базы правил и системы принятия решений (СПР)
- при́вод – исполняющий компонент агента, ответственный за применение обратной связи к окружающей среде

Примерная структура агента описана схематично на рис. 1.

Согласно приведенной на рис. 1 схеме, агент способен принимать решения в соответствии с логическими высказываниями, хранящимися в его базах и / или на основании результатов вычислений. СПР является самой важной частью агента, т.к именно она отвечает за принятие решений. СПР работает по следующему алгоритму: построение модели ограничений на основании данных, полученных из сенсора; выбор наиболее подходящего поведения на основании базы правил и априорной базы знаний. В целом, СПР может имплементировать механизм дополнительного обучения, позволяющий изменять существующие правила и знания и / или добавлять новые.

В рамках рассматриваемой в данной работы СПРО, компоненты моделируются с использованием мультиагентного подхода, то есть каждый компонент системы представляет собой агент. СПР решений командного

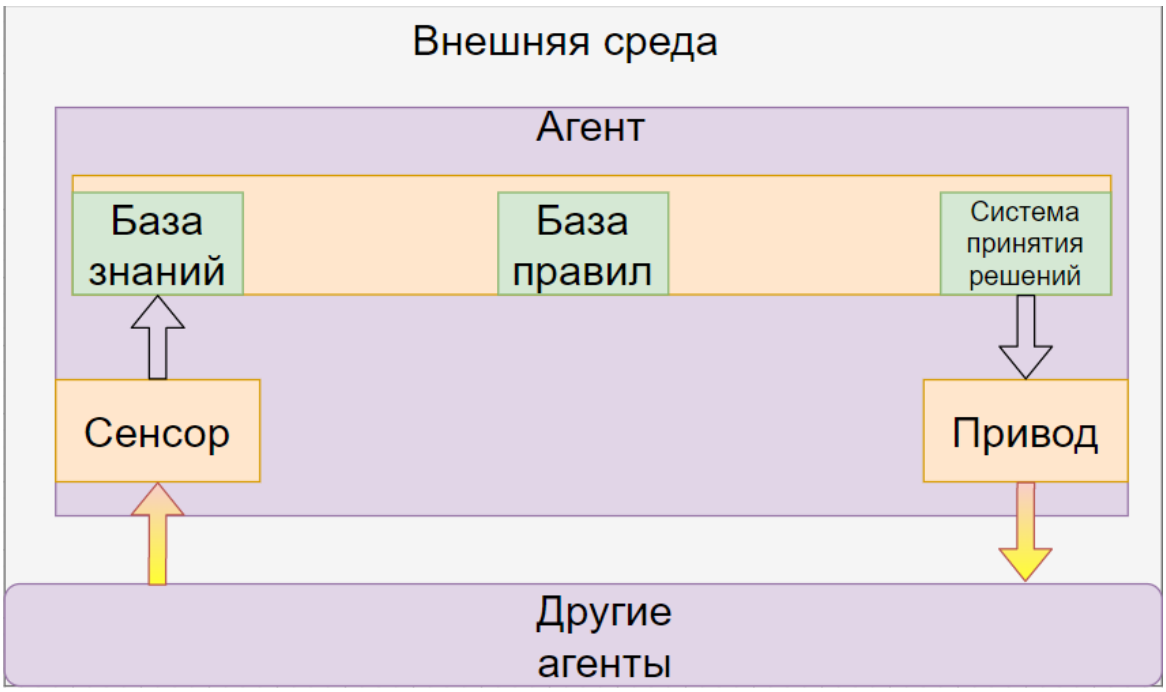

Рис. 1: Схематическое описание структуры агента и среды

.

центра СПРО будет генерировать планы перехвата цели динамически, используя для этого данные с радаров обнаружения и вычислительный эксперимент в рамках СПРО. Агенты СПРО показаны на рис. 2. Компоненты одинакового цвета имеют одинаковые возможности.

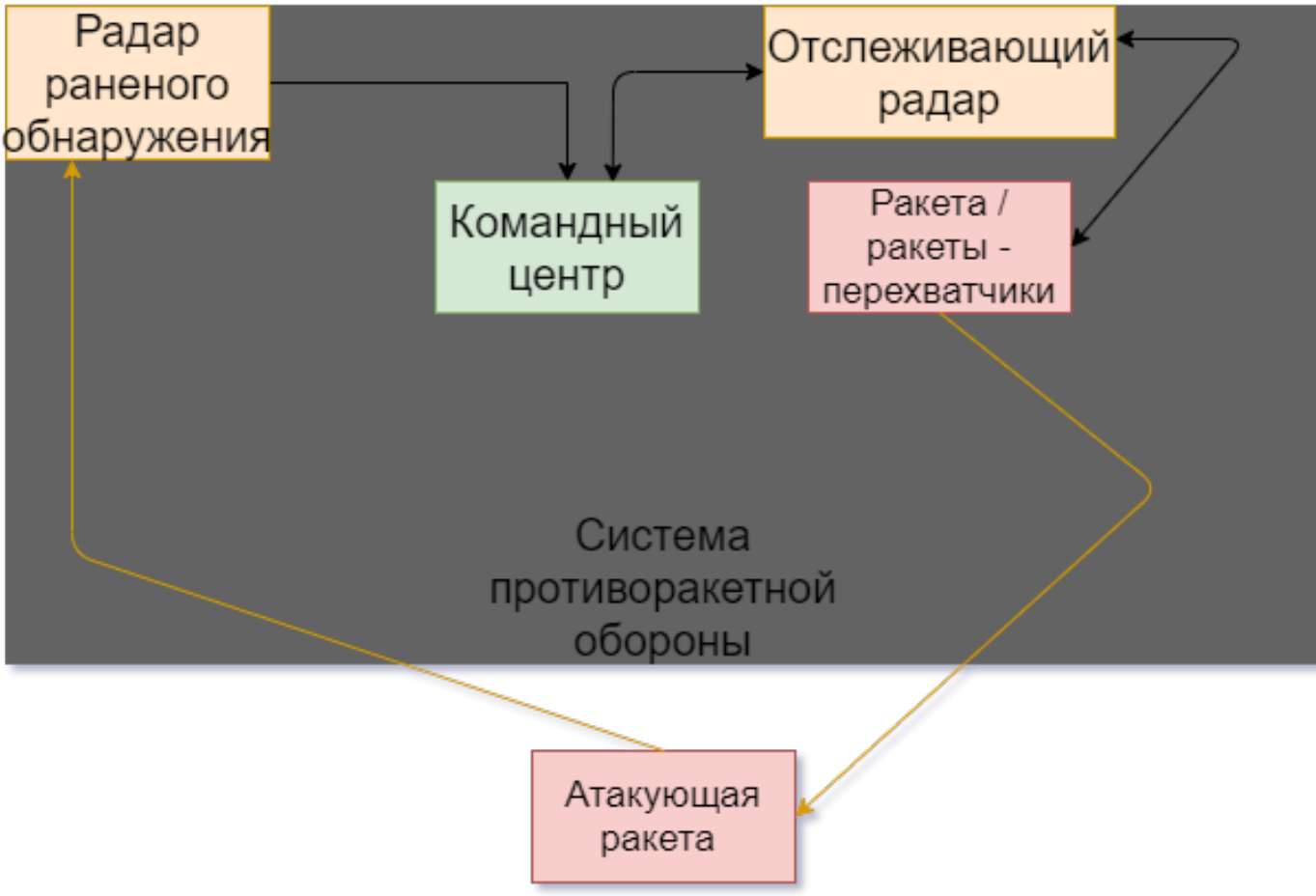

Рис. 2: Схематическое описание структуры СПРО, связей в ней и воздействия атакующей ракеты

.

Агент «командный центр»

Агент «командный центр» выполняет назначение конкретной ракеты-перехватчика на цель и посылает инструкции о запуске. Ниже представлена таблица знаний данного агента.

Свойство	Значение
Позиция командного центра (координаты)	широта, долгота, высота
Позиция ракеты-перехватчика	широта, долгота, высота
Угол наклона ракеты-перехватчика	вычисляется с помощью формул сферической геометрии
Время запуска ракеты-перехватчика	вычисляется в рамках решения задачи Ламберта

Время подлета ракеты-перехватчика к цели	вычисляется в рамках решения задачи Ламберта
Вероятность перехвата	вычисляется ракетой-перехватчиком

Таблица 1: Таблица знаний командного центра

Правила командного центра включают правила оценки угроз и правила выбора ракеты перехватчика. Эти правила поддерживаются нижеописанной моделью принятия решений.

Вычислительная модель назначения ракеты-перехватчика. Эта модель генерирует план-назначение ракет-перехватчиков. Для этого используются местоположение, тип и количество доступных ракет-перехватчиков, уровень угрозы и количество атакующих ракет и время их определения.

Вычислительная модель для точки перехвата. Точка перехвата определяется как точка, находящаяся на траектории атакующей ракеты и ближе всего к максимальной высоте ракеты-перехватчика, что позволит запускать дополнительные ракеты в случае неудачи. В свою очередь, траектория атакующей ракеты определяется посредством анализа данных, накапливаемых радаром раннего обнаружения и отслеживающим радаром. Эти данные являются массивом, где каждый элемент имеет вид «пара {координаты, время} ». Максимальная высота, достигаемая ракетой-перехватчиком вычисляется из характеристик этой ракеты, но в данной работе для упрощения вычислений является свойством самой ракеты.

Вычислительная модель определения времени запуска ракеты-перехватчика. Пусть T_L – время запуска ракеты-перехватчика; T_e – момент времени, когда цель окажется в точке перехвата; T_f – время полета ракеты-перехватчика к точке перехвата; T_p – время подготовки ракеты-перехватчика к запуску, в данной работе полагаемое константной (однако, можно предположить, что в реальности этот параметр будет вести себя как модуль квадратичной функции, зависящей от кол-ва подготовленных ракет-перехватчиков, с положительным старшим коэф-том и ну-

левым дискриминантом, возникающим при приравнивании этой функции нулю). Тогда $T_L = T_e - T_f - T_p$.

Модель вычисления угла наклона Пусть A – точка запуска ракеты-перехватчика, B – предполагаемая точка перехвата, северный полюс – C. Тогда эти три точки вместе с геоцентрической точкой o (центр Земли) формируют сферический треугольный конус; A, B, C представляют угол между двумя гранями, а a, b, c – геоцентрический угол между двумя точками. Получаем, что угол наклона $\theta_f = arccos(cos(a) \cdot cos(b) + sin(a) \cdot sin(b) \cdot cos(c))$, где $a = 90 - T_{Lat}, b = 90 - L_{Lat}, C = T_{Lon} - L_{Lon}$, где L_{Lon}, L_{Lat} и T_{Lon}, T_{Lat} представляют собой долготу и широту точки запуска и целевой точки перехвата.

Модель вычисления параметров уничтожения ракеты-цели. Установив начало координат в центр Земли и положив OX как ось, соединяющую центр Земли и точку пуска ракеты, получим полярную систему координат. Обозначим r_1, r_2, вектор точки запуска и вектор перехвата из геоцентрической точки начала координат; θ_f – угол наклона получаемый как $\theta_f = arccos(\frac{r1 \cdot r2}{|r1| \cdot |r2|})$; γ – угол наклона траектории получаемый как $\gamma = \frac{\pi - \theta_f}{4}$ при движении по траектории с минимальной энергией; μ – гравитационная константа. Согласно уравнению Ламберта, скорость перехвата ракеты V_s удовлетворяет условию $V_s = \sqrt{\frac{|r_2|(1-cos\theta_f)\mu}{|r_1|^2 \cdot cos(\gamma^2) - |r_2| \cdot cos(\theta_f + \gamma) \cdot cos\gamma}}$. Так как траектория полеты – эллипс, то время полеты ракеты рассчитывается как $t_f = \frac{|r_1|}{V \cdot cos(\gamma)(\frac{tan(\gamma)(1-cos\theta_f)+(1-\lambda sin\theta_f)}{(2-\lambda)\frac{|r_1|}{|r_2|}})} + \frac{2cos\gamma}{\lambda((2/\lambda)-1)^{1.5}} arctan \frac{((2-\lambda)-1)^{0.5}}{cos\gamma \cdot ctg(\frac{\theta_f}{2}) - sin\gamma}$, где $\lambda = |r_1| \cdot \frac{V^2}{\mu}$.

Агент «ракета»

В терминах ООП, «ракета» является абстрактным базовым классом, от которого наследуются атакующая ракета и ракета-перехватчик. База его знаний хранит данные, необходимые для вычисления траектории и включает позицию и скорость ракеты, т.е. в каждый момент времени известна широта, долгота, высота и скорость ракеты.

Рис. 3: Блок схема принятия решения командным центром

.

Агент «радар»

Как и в прошлом случае, «радар» является абстрактным базовым классом. База знаний радара содержит преимущественно данные для вычис-

Рис. 4: Блок схема принятия решения ракетой-перехватчиком

.

ления радиуса радарного покрытия и угла покрытия. Правила радара включают правила обнаружения и правила предсказания движения ракеты. Правила обнаружения вычисляют область радарного покрытия через радарное уравнение и считается, что любой объект в области радарного покрытия мгновенно обнаруживается при первом сканировании,

т.е. в рамках данной работы не рассматриваются цели с активным подавлением радаров или постановкой помех; правила предсказания отсылают информацию в командный центр если объект был замечен хотя бы трижды.

Полагая, что ракета цилиндрической формы, обозначим r – радиус ракеты, h – длина боеголовки, λ – длина волны сигнала радара, h_1, h_2, h_3 – длина двигателей этапа с соответствующим номером. Эффективный поперечник рассеяние (ЭФР) зависит также от угла между радаром и проекций движения ракеты на OY, обозначенным как θ Тогда эффективный поперечник рассеяния ракеты-цели согласно эмпирической формуле равен $R = 2\pi r(\frac{h_1^2}{\lambda} + \frac{h_2^2}{\lambda} + \frac{h_3^2}{\lambda} + \frac{4}{9\lambda h} \cdot \sqrt{(r^2 + h^2)^3}) * cos\theta$.

Рис. 5: Блок схема принятия решения радаром раннего обнаружения

.

1.3 Система обозначений и ограничений для алгоритма

Введем систему обозначений, которые в дальнейшем будут использованы для описания компонентов в алгоритме перехвата целей.

Обозначение	Определение
n	кол-во атакующих ракет-целей
j ($\in [0;n]$)	номер атакующей ракеты
T_j	«угроза» атакующей ракеты №j
R_j	время на перехват ракеты №j как расстояние от размещения перехватчиков до цели
C_j	фактор возможного вмешательства командира
D_j	расстояние от атакующей ракеты №j до цели (цель общая для всех ракет)
S_j	максимальная скорость ракеты №j
m	кол-во ракет перехватчиков
i ($\in [0;m]$)	номер ракеты-перехватчика
V_i	«цена» ракеты-перехватчика №i
w_k	массив (вектор) специальных весов
P_{ij} ($P_{ij} \in [0;1]$)	вероятность перехвата ракетой i ракеты j
x_{ij}	Булеан, указывающий назначена ли ракета i на ракету j

Таблица 2: Символы, используемые для обозначения компонентов алгоритма

Отбросим ситуации, где n или m равны 0, т.к. это автоматически приводит к бесконечному множеству решений или пустому множеству решений соответственно. Также заметим, что в рамках этой работы m $\geq$ n.

При решении задачи должны соблюдаться следующие ограничения:

1.Для обеспечения безопасности целей, атакуемых ракетами, необходимо назначить хотя бы одну ракету-перехватчик на каждую цель и приоритет по кол-ву ракет и очередности назначения должен отдаваться ракетам с максимальной «угрозой» и назначаться должны ракеты с минимальной «стоимостью», что позволит избежать быстрого расхода самых «лучших ракет-перехватчиков». Это можно выразить таким фитнесом: $\sum_{j=1}^{n}\sum_{i=1}^{n}\frac{T_j P_{ij} x_{ij}}{V_i x_{ij}}$.

2. Генерация плана перехвата зависит от оперативности и качества данных, приходящих с радара. Хорошообнаружимая ракета лучше отслеживается и должна иметь более высокий приоритет.

3. Приоритет перехвата также отдается ракетам, более близким к обороняемым целям.

4. Предпочтительно перехватывать ракеты с более высокой скоростью.

5. Вмешательство командира, основанное на эмпирическом внешнем знании играет максимальную роль.

С учетом вышеизложенного, фитнес может быть описан следующим образом $\sum_{j=1}^{n}\sum_{i=1}^{m}\frac{P_{ij}\prod_{k=1}^{3} w_k T_j S_j C_j}{\prod_{l=4}^{6} w_l V_i x_{ij} R_j D_j}$.

Для увлечения адаптируемости, положим $w_x = 1, x \in [1;6]$

2 Алгоритм роя частиц переменной окрестности с отрицательным отбором и его применение

2.1 Общая идея алгоритма

Метод роя частиц (МРЧ) использует множество частиц, каждая из которых представляет потенциальное решение и для каждой частицы можно посчитать фитнес-функцию, служащую для отбраковки худших решений. Изначально множество частиц инициализируется случайными значениями, однако важно, что каждая частица, являющая вектором, должна иметь многомерное равномерное распределение внутри своих значений. Текущая позиция частицы записывается n-мерным вектором и обозначается как $X = (X_1, ..., X_n)$. i-ый элемент частицы в поисковом пространстве S обозначается как $X_i = \begin{bmatrix} x_{i1} \\ \vdots \\ x_{iD} \end{bmatrix}$; текущая скорость частицы обозначается как $V_i = \begin{bmatrix} V_{i1} \\ \vdots \\ V_{iD} \end{bmatrix}$; локальная лучшая позиция $P_i = \begin{bmatrix} P_{i1} \\ \vdots \\ P_{iD} \end{bmatrix}$; глобальное наилучшее решение обозначается как $P_g = \begin{bmatrix} P_{g1} \\ \vdots \\ P_{gD} \end{bmatrix}$.

Обозначим через w инерционный вес, позволяющий балансировку локального и глобального поиска; k – номер итерации; V_{id} –скорость частицы i в d-ом пространстве; r_1, r_2 – случайные вещественные числа, такие, что $0 \leq r_i \leq 1, r_i \in \mathbb{R}, i \in \{1,2\}$; c_1, c_2 – неотрицательные константы, "факторы ускорения". Скорость и позиция каждой частицы обновляются итеративно по следующей формуле:

$$V_{id}^k = wV_{id}^{k-1} + c_1 r_1 (P_{id}^{k-1} - X_{id}^{k-1}) + c_2 r_2 (P_{gd}^{k-1} - X_{id}^{k-1}); X_{id}^k = X_{id}^{k-1} + V_{id}^k \tag{2.1.1}$$

. Для исключения невалидных решений, позиции и скорости частиц ограничены интервалами $[-X_{max}; X_{max}]$ и $[-V_{max}; V_{max}]$ соответственно.

Фитнес каждой частицы вычисляется на каждой итерации; новое значение фитнес-функции сравнивается с текущими локальной и глобальной лучшей позицией; если новое значение превосходит старое, то локальная лучшая позиция будет обновлена и затем среди локальных лучших позиций проводится поиск по максимум фитнес функции; если этот максимум превосходит фитнес функцию глобального лучшего решения, то глобальное лучшее решение заменяется локальным лучшим решением с указанным максимумом. Во избежание зацикливания алгоритма вводится произвольное достаточно большое число T_{max}, ограничивающее число итераций алгоритма; в рамках реализации данного алгоритма T_{max} постепенно убывает, что отражает ограниченное время на принятие решения при все более близкой ракете-цели. В конце выполнения алгоритма, наступит ли он сам по достижению заданной точности ли будет прерван T_{max}, вектор P_g содержит лучшее решение и представляет собой «выходные данные» данного алгоритма.

Общие идеи применения указанного выше алгоритма в этой работе похожи на таковые в и изложены далее. Во-первых, необходимо убедиться в назначении адекватного кол-ва перехватчиков на одну цель. Для этого сгенерированная и/или обновленная частица должна обновляться по правилу замены повторяющихся элементов в этой частице. Также необходимо избежать выхода скорости частицы за границу V_{max}. Для выполнения двух предыдущих условий можно представить формулу 2.1.1 в виде

$$V^k = \begin{cases} V_{max}, V^k > V_{max} \\ V^k, V_{min} \leq V^k \leq V_{max} \\ V_{min}, V^k < V_{min} \end{cases} ; X^k = \begin{cases} X_{max}, X^k > X_{max} \\ X^k, X_{min} \leq X^k \leq X_{max} \\ X_{min}, X^k < X_{min} \end{cases} \quad (2.1.2)$$

.

В-третьих, для обеспечения более равномерной сходимости, параметр w будет убывать линейно по формуле: $w = w_{end} + (\frac{T_{max}-k}{T_{max}})(w_0 - wend)$.

В-четвертых, c_1, c_2 можно использовать не как константы, а переменные, обозначающие фактор самообучения и группового обучения соответственно и изменяющиеся согласно формуле:

$$\begin{cases} c_1 = c_{1_{min}} + \frac{(c_{1_{max}} - c_{1_{min}})k}{T_{max}} \\ c_2 = c_{2_{min}} + \frac{(c_{2_{max}} - c_{2_{min}})k}{T_{max}} \end{cases} \quad (2.1.3)$$

Введем вспомогательную переменную K – фактор сужения, необходимый для улучшения сходимости алгоритма. Введем для этого дополнительную переменную $\phi = c_1 + c_2 : \phi \geq 2$ и дополнительные обозначения: N_r – радиус окрестности, меняющийся в ходе итераций; P_{id}^k – собственное экстремальное значение частицы; $P_{N_r d}^k$ – экстремальное значение соседа. Тогда $K = \frac{2}{|2-\phi-\sqrt{\phi^2-4\phi}|}$. С учётом этих нововведений в практической реализации работы будет использована следующая формула для обновления скорости частицы:

$$V_{id}^k = wV_{id}^k - c_1 r_1 (X_{id}^{k-1} - P_{id}^{k-1}) - c_2 r_2 (X_{id}^{k-1} - P_{N_r d}^{k-1}) \quad (2.1.4)$$

.

Теперь опишем упомянутый в начале главы процесс «отрицательный отбор». «Отрицательный отбор» позволяет оптимизировать процесс варьирования окрестности, необходимый чтобы избежать попадания алгоритма в локальный экстремум посредством обновления некоторых частиц до тех пор, пока не достигнуто условие. Основная идея процесса заключается в итеративном обновлении частиц в соразмерно параметру R_u при сходимости алгоритма. В свою очередь, сходимость алгоритма в процессе его выполнения определяется фактом того, что сходство частиц больше некоторого предела сходства T_{aff}. Обозначим сходство частицы p в d-ом измерении как A_{pd}, P_{gd} – глобальное экстремальное значение. Сходство частицы p в d-ом измерении описывается $Apd = 1 + \frac{|P_{gd} - x_{pd}|}{X_{min} - X_{max}}$

Также введем формулу для сходства p-ой частицы как среднее сходство всех размерностей: $A_{pd} = \frac{\sum_{d=1}^{D} A_{pd}}{D}$.

Теперь опишем с помощью простой блок-схемы ниже алгоритм отрицательного отбора для каждой частицы. Заметим, что согласно этой блок-схеме, механизм отрицательного отбора будет задействован только при достижении частицей состояния, в котором каждая размерность частицы меньше T_{aff}.

Рис. 6: Блок-схема процесса «отрицательного отбора»

.

После введения всех необходимых формул, изложения концепций и алгоритмов, играющих вспомогательную роль, опишем непосредственный алгоритм АРЯПОСОО.

Рис. 7: Блок-схема процесса работы алгоритма роя частиц переменной окрестности с отрицательным отбором

.

2.2 Реализация алгоритма и интеграция его в СПРО

.

Для быстроты написания, переносимости и использования готовых вспомогательных решений был выбран язык программирования Python 3.7.7.

Для агентов был введен абстрактный базовый класс «Abstract Base Class Agent» и все используемые агенты являются экземплярами классов-наследников. Компоненты СПРО коммуницируют между собой с помощью веб-сокетов. Для симуляции передвижения ракет в трехмерном пространстве ракеты хранят на каждой итерации тройку координат внутри своего экземпляра, а «сканирование» радаром ракет происходит как анализ всех существующих в данный момент ракет с проверкой нахождения координат внутри области анализа. Возможно, такой подход не является полностью точным, но значительно экономит память и упрощает описание процесса «сканирования»; альтернативой данному подходу является создание трехмерного массива, дискретно описывающего пространство.

Большой трудностью является реализация одновременно происходящих параллельных событий. Обычно, в студенческих работах для этого вводят некоторый порядок синхронных непараллельных действий и при этом предполагается, что вычисление этих действий происходит настолько быстро, что время исполнения кода ЦПУ не будет заметным. Однако, алгоритм АРЯПОСОО является затратным по времени, поэтому такой подход в данной работе невозможен.Вместо этого было решено использовать реальное параллельное выполнение кода. Проблемой Python в этой области явлвяется GIL ("Global Interpreter Lock"), однако, этого удалось избежать с использованием Thread- и ProcessPoolExecutor и запуском каждой ракеты как отдельного процесса Process.spawn() из модуля multiprocessing. Для реализации одновременности и параллельности каждый компонент выполняется в бесконечном цикле и «спит» 0.1 секунды после каждого своего «тика» (минимальная единица времени и действия); «сон» с использованием time.sleep() необходим для создания пауз в выполнении кода процессором и переключении контекста выполнения, иначе потребление ресурсов ЦПУ данным компонентом будет постоянным и вызовет задержки в переключении потока управления, когда

число компонентов превысит число логических потоков ЦПУ, что в свою очередь создаст проблемы с симуляцией одновременности. Также заметим, что для быстрого освобождения ресурсов, поток и процесс ракеты при взрыве сразу останавливается и объект ракеты удаляется вручную.

Сама СПРО состоит из генератора атакующих ракет, который с заданной интенсивностью генерирует случайное нормально распределенное на заданному интервале кол-во атакующих ракет с разными параметрами полезной нагрузки, скорости и высоты (для простоты, этот компонент сразу генерирует ракеты находящиеся на этапе прохождения тропосферы), аналогичного генератора ракет-перехватчиков, имитирующего «подвоз» ракет-перехватчиков в случайное время с интенсивностью, возрастающей по мере исчерпания запаса, трех радаров раннего обнаружения, 9 радаров отслеживания и переменного кол-ва ракет-перехватчиков, ограниченных сверху максимальной вместительностью склада, множества защищаемых объектов (программа завершает работу при полном уничтожении всех защищаемых объектов).

Для бóльшей реалистичности, радары раннего обнаружения, как и радары отслеживания, имеют различный эффективный радиус обнаружения. Для простоты предполагается, что атакующие ракеты входят в имитируемую область «слева направо» и все радары «сканируют область», радиус-вектор которой направлен «слева-направо».

Основное принятие решения и генерация плана-перехвата генерируется внутри командного центра, т.е. именно в нём выполняется АРЯПОСОО каждый раз по мере появления новых ракет-целей и ракет-перехватчиков, т.е. при появлении новой ракеты-перехватчика с характеристиками, превосходящими характеристики уже отправленной/отправленных ракеты/ракет и при отсутствии более важных целей, возможно назначение дополнительных ракет. Таким образом, генерация новых планов-перехватов происходит только после наступления события «новая ракета-цель обнаружена радаром» или «на склад поступила новая ракета-перехватчик». Также, для реализации неблокирующей работы командного центра (т.е. работы, неблокирующей прием-передачу сигналов), для генерации каждого плана-перехвата создается отдельный процесс с помощью модулю multiporcessing в режиме «spawn», где на каждой итерации проверка

каждой частицы на достижение предела сходства происходит внутри отдельного потока внутри ProcessPoolExecutor. Данный подход позволяет максимально задействовать ресурсу ЦПУ при вычислениях в АРЯПО-СОО.

2.3 Тестирование и сравнение результатов работы с аналогичными моделями

Для запуска данной модели был использован компьютер со следующими характеристиками: OS: Ubuntu 18.04.4 LTS
Разрядность: x64
CPU: Intel(R) Xeon(R) CPU E5-1650 v3 @ 3.50GHz
Оперативная память: 126 Гб

Самыми важными параметрами является предел сходства T_{aff} и параметр R_u. Для определения оптимального значения этих параметров был проведен оптимизационный эксперимент, результаты которого показаны ниже.

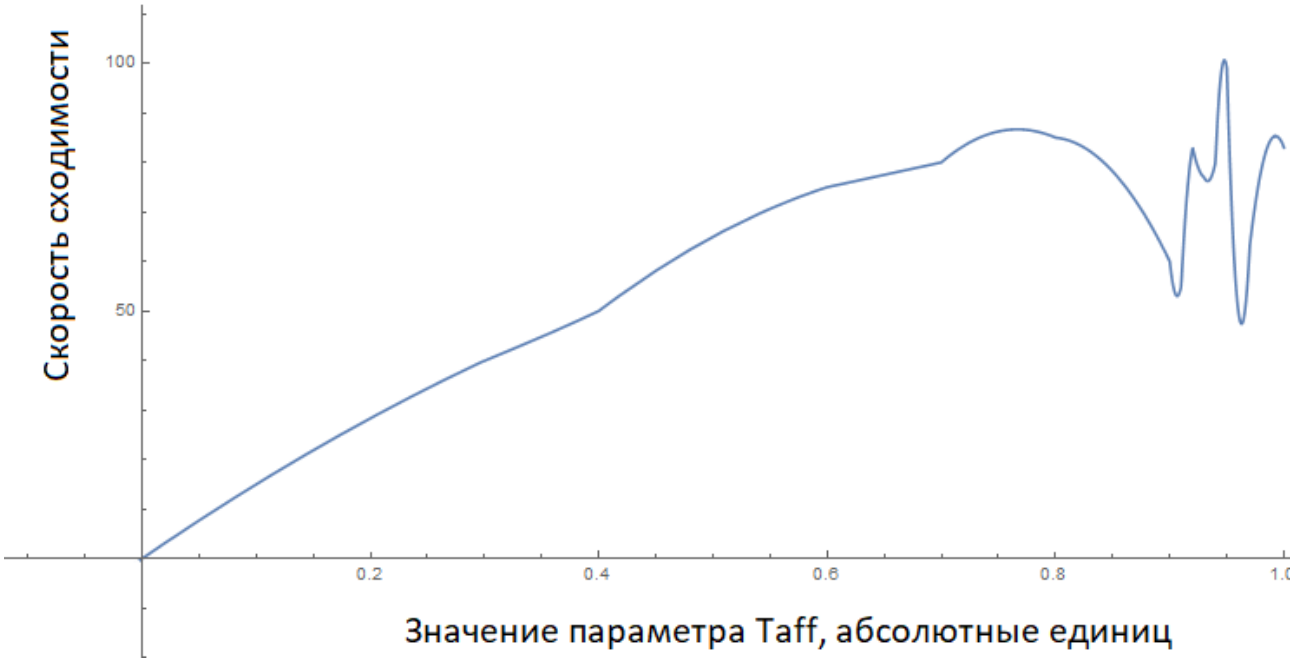

Рис. 8: Скорость сходимости алгоритма при различных значениях параметра T_{aff}

.

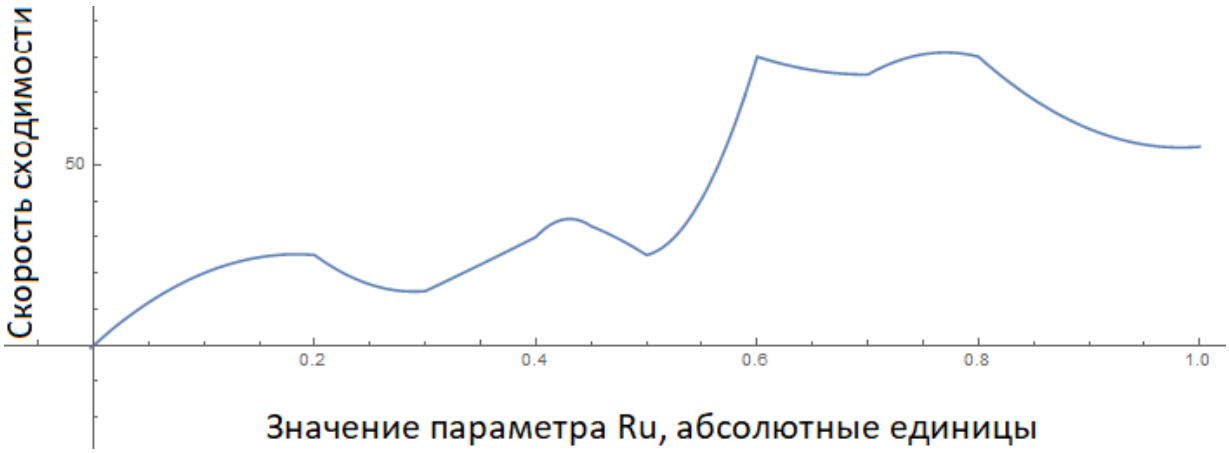

Рис. 9: Скорость сходимости алгоритма при различных значениях параметра R_u

.

Алгоритм	Средняя скорость сходимости, %	Среднее кол-во итераций
PSO	80.3	975
CFPSO	85.1	937
LDPSO	87.6	850
VNPSO	91.5	715
VNVNNSPSO	96.8	720
АРЯПОСОО	92.333	698

Таблица 3: Скорость сходимости и среднее кол-во итераций различных методов

Таким образом, оптимальная скорость сходимости (в процентах от 10^5 итераций) достигается в $T_{aff} = 0.96, R_u = 0.63$.

Будем считать, что алгоритм сходится при точности в 99.5% от оптимального решения. Выполнив 10^6 итераций каждого алгоритма ниже, имеем следующие скорости сходимости и среднее кол-во итераций.

Как видим, несмотря на не самый лучший результат в средней скорости сходимости, алгоритм АРЯПОСОО работает быстрее всех остальных, что позволяет использовать его как основной, если задача, к которой он применяется, ограничена по времени.

Разработанная программа генерирует планы-перехваты подобного вида:

1	ID ракеты-цели	Множество ID ракет-перехватчиков	Долгота	Широта	Высота	Время ожидаемого перехвата
2	timestamp(время обнаружения ракеты)	{timestamp(время запуска ракеты)_1, timestamp_2, timestamp_3}	float	float	float	timestamp
3	1591213981	{1591214071, 1591214084}	44.936543	34.135372	6273.2	1591214123

Рис. 10: Пример плана-перехвата, генерируемого программой (пост-визуализация в стороннем ПО)

.

Выходной формат .csv позволяет импортировать данные в любой табличный процессор для последующей их обработки и визуализации. Во избежание ситуации неправильного вывода, формируемого при многопоточной работе приложения, вывод данных на консоль или файл осуществляет отдельный поток.

Заключение

Разработанный и протестированный алгоритм имеет как преимущества, так и недостатки, рассмотрим их подробнее.

Преимущества:

- Полная симуляция одновременно происходящих параллельных процессов;
- Минимальное, по сравнению с другими рассмотренными алгоритмами, число итераций;
- Возможность экспортировать данные в табличные процессоры для пост-обработки;
- Низкие затраты по памяти – запущенное приложение при 10 активных ракетах-целях, 30 активных ракетах-перехватчиках и 100 ракетах-перехватчиков на складе потребляет около 500 Мб ОЗУ;
- Высокие возможности распараллеливания алгоритма на критических участках – потенциальный источник еще большей оптимизации исполнения алгоритма.

Недостатки:

- Не самая высокая скорость сходимости среди рассмотренных;
- Симуляция параллельных событий требует процессора со значительным числом ядер и логических потоков, которые часто отсутствуют на встраиваемых системах, реально используемых в подобной технике. Так, описанное выше кол-во объектов создает около 50 процессов и более 100 потоков;
- Алгоритм рассматривает упрощенную схему полёта ракеты-цели и не учитывает возможных положений, допускающих оптимизацию процесса перехвата.

Были решены следующие задачи:

- Описана СПРО для которой построена модель и агенты, участвующие в ней;
- Описана система принятия решений и накладываемые на нее ограничения, алгоритм принятия решений;
- алгоритм принятия решений был реализован в виде ПО, протестирован и сравнён с другими подобными алгоритмами;
- были решены различные задачи, возникающие в области многоточечного программирования;
- были сделаны выводы о потенциале и возможностях развития развития данного алгоритма.

К сожалению, не удалось реализовать дополнительную цель и построить полноценный интерфейс приложения, позволяющий ввод-вывод данных и, самое важное, демонстрацию работы алгоритма посредством показа назначения ракет-перехватчиков на цели и непосредственно перехватом.

Несмотря на неидеальные параметры работы алгоритма, он всё же демонстрирует высокую скорость исполнения. Отметим, что как сам алгоритм АРЯПОСОО, так и модель, имитирующая СПРО, допускают в будущем модификации как расширяющие функционал СПРО, так и модифицирующие непосредственно работу самого алгоритма АРЯПОСОО.