Оптимизация, системный анализ и исследование операций

© 2024 г. А.А. ГАЛЯЕВ, чл.-корр. РАН (galaev@ipu.ru),
Е.А. РЯБУШЕВ (lispandhaskell@gmail.com)
(Институт проблем управления им. В.А. Трапезникова РАН, Москва)

ПОИСК СУБОПТИМАЛЬНОГО РЕШЕНИЯ ДИНАМИЧЕСКОЙ ЗАДАЧИ КОММИВОЯЖЕРА МЕТОДОМ МОНТЕ-КАРЛО[1]

Рассматривается задача составления плана обхода прямолинейно движущихся в одну точку целей для простых движений перехватчика (коммивояжера). Предлагаются новый критерий задачи на основе начального разбиения области возможного перехвата, а также алгоритм поиска субоптимального плана обхода на основе построения дерева поиска решения методом Монте-Карло. Разработана численная реализация алгоритма, проведено моделирование и статистически проанализированы полученные планы обхода целей.

Ключевые слова: динамическая задача коммивояжера, перехват в простых движениях, комбинаторная оптимизация, алгоритм Монте-Карло.

DOI: 10.31857/S0005231024020065, **EDN:** UCSGKV

1. Введение

Настоящая работа посвящена проблеме предотвращения достижения заданной точки пространства прямолинейно движущимися атакующими целями. Защищать точку предлагается с помощью перехватчика, который способен свободно перемещаться в пространстве и обходить атакующие цели. При этом ключевую роль играет выбор наилучшего порядка обхода множества приближающихся целей. Данная задача уже рассматривалась в [1], где обсуждались понятия опасности, удобства и сложности перехвата целей и предлагались векторные критерии качества планов обхода целей. В [1] предлагался интеллектуальный алгоритм в общем случае на основе полного перебора, способный эффективно находить планы обороны защищаемой точки одним перехватчиком от атак до двадцати целей. В настоящей работе предлагается другой алгоритм, способный находить субоптимальный план перехвата, который справляется с задачей при наличии более двадцати целей с приемлемой эффективностью.

Известная в литературе «динамическая задача коммивояжера» (ДЗК) [2, 3] или Moving Targets Traveling Salesman Problem (MTTSP) [4, 5] является близкой по смыслу к проблеме, исследуемой в настоящей работе. ДЗК обобщает задачу коммивояжера (ЗК), и в 1972 г. была доказана NP-полнота за-

[1] Работа выполнена при частичной финансовой поддержке Российского научного фонда (грант № 23-19-00134).

дачи о гамильтоновом цикле, что подразумевает NP-полноту ЗК и, как следствие, ДЗК [6]. На данный момент не существует эффективных методов точного решения ДЗК, а сама задача находится на начальных этапах исследования [7]. Из этого следует, что в общем случае точное построение плана перехвата может оказаться весьма затруднительным при большом числе целей, и поэтому разумным является вопрос о быстром поиске суб-оптимального решения. В настоящей момент существуют подходы к построению таких решений ДЗК на основе генетических алгоритмов, о которых более подробно можно прочитать, например, в [7] или локальной эвристической оптимизации [8].

Одним из методов поиска субоптимальных решений для задач дискретной оптимизации является алгоритм поиска по дереву Монте-Карло (Monte Carlo Tree Search) [9, 10]. Этот метод получил широкую известность после его успешного применения для игры в Го в рамках проекта AlphaGo [9], где поиск деревом Монте-Карло применялся совместно с нейросетевым обучением. Помимо антагонистических игр, этот алгоритм в различных вариациях применялся и к одиночным играм, которые можно трактовать как задачи дискретной оптимизации [10, 11]. Дополнительно, алгоритм поиска по дереву Монте-Карло предлагает удобную базу для применения нейросетевых подходов к задачам дискретной оптимизации. Так, нейросетевое обучение может применяться для вычисления эвристических функций полезности и для построения распределений вероятности, по которым строятся случайные продолжения. Таким образом, базовый алгоритм поиска по дереву позволяет разбить исходную задачу на более простые составляющие, которые уже могут решаться обученными нейросетями. Более подробное обсуждение возможных способов сочетания нейросетевых подходов с поиском по дереву Монте-Карло можно найти в [9, 12, 13].

В данной работе применяются основные идеи этого алгоритма к рассматриваемой задаче защиты точки от проникновения в нее движущихся целей. Для этого используется модель простых движений для перехватчика, поскольку дистанции при перемещении между целями являются достаточно великими по сравнению с реальным радиусом разворота перехватчика. В таких предположениях учет маневренности в задаче планирования обхода целей не оказывает влияния на структуру оптимального плана, что позволяет перейти от дискретно-непрерывной задачи оптимизации к дискретной. Еще одним отличием текущей работы от известных в литературе постановок является критерий задачи, который имеет смысл для приложений на практике и оптимальное значение которого в общем случае достижимо на наборе планов, т.е. является неединственным. Таким образом, основное предложение данной работы состоит в применении нового подхода, изначально разработанного для решения игровых задач, к проблеме построения субоптимального решения для варианта динамической задачи коммивояжера с защитой охраняемой точки от приближающихся целей. Предлагаемый здесь подход позволяет получать такие решения способом, который ранее для данных целей не приме-

нялся и может служить альтернативой более известным подходам на основе генетических алгоритмов, которые ранее применялись в подобных задачах.

2. Математическая модель и постановка задачи

Математическая модель и постановка задачи, используемые в настоящей работе, совпадают с предложенными в [1], однако для полноты изложения приводим их описание в данном разделе.

2.1. Модель движения целей и перехватчика

Полагаем, что перехватчику требуется перехватить n целей, которые прямолинейно движутся в одной плоскости со скоростями из диапазона $[v_{\min}, v_{\max}]$. Защищаемая точка расположена на плоскости в начале координат, а цели появляются на внешней границе круга радиуса R в слое толщиной $2\Delta R$. Перехватчик при этом может перемещаться в простых движениях со скоростью $v(t) < V$, причем $V > v_{\max}$.

Начальные положения целей задаются радиус-векторами $\mathbf{r_1^0}, \ldots, \mathbf{r_n^0}$, где $\mathbf{r_i^0} = (x_i^0, y_i^0)$. Скорости целей $\mathbf{v_1}, \ldots, \mathbf{v_n}$ полагаем известными и задаем их движение линейными соотношениями

$$\mathbf{r_i}(t) = \mathbf{r_i^0} + \mathbf{v_i}t. \tag{1}$$

Также считаем, что траектории всех целей проходят через начало координат и, таким образом, время подлета i-цели к защищаемой точке дается выражением

$$t_i(t) = \frac{|\mathbf{r_i(t)}|}{|\mathbf{v_i}|}. \tag{2}$$

Величину, обратную к времени подлета, будем называть опасностью цели, для чего введем обозначение:

$$d_i(t) = t_i^{-1}(t) = \frac{|\mathbf{v_i}|}{|\mathbf{r_i(t)}|}. \tag{3}$$

Таким образом, начальная тактическая обстановка задается начальным расположением всех целей совместно с их векторами скорости, а также указанием начального положения перехватчика.

Движение перехватчика определяется системой дифференциальных уравнений

$$\begin{cases} \dot{x_a} = v(t)\sin\psi(t) \\ \dot{y_a} = v(t)\cos\psi(t) \end{cases}, \tag{4}$$

где $\mathbf{r_a} = (x_a, y_a)$ – положение перехватчика, $v(t)$ – модуль его скорости, а $\psi(t)$ – управление направлением движения. Минимальное время τ_i, необходимое перехватчику на перехват i-цели, находится из квадратного уравнения

$$(\mathbf{r_i} - \mathbf{r_a} + \mathbf{v_i}\tau)^2 = V^2\tau^2 \tag{5}$$

как его минимальный положительный корень

$$\tau_i(\mathbf{r_a}, \mathbf{r_i}, \mathbf{v_i}) = \frac{\sqrt{(\mathbf{r_i} - \mathbf{r_a})^2(V^2 - \mathbf{v_i}^2) + ((\mathbf{r_i} - \mathbf{r_a}) \cdot \mathbf{v_i})^2} - (\mathbf{r_i} - \mathbf{r_a})\mathbf{v_i}}{V^2 - \mathbf{v_i}^2}. \tag{6}$$

Таким образом, функция перехвата цели является решением задачи наискорейшего перехвата одиночной цели. Данной модели достаточно для поиска оптимального плана для перехватчика, движущегося в классе простых движений, поскольку для нее справедливы принципы неоптимальности простоя и движения на максимальной скорости. В [1] это было доказано для оптимизации плана перехвата по числу пропущенных целей и времени перехвата. В текущей работе будет использоваться другой критерий оптимизации, однако модель наискорейшего перехвата одиночной цели все равно будет использоваться.

2.2. План перехвата

План π по перехвату k целей задается упорядоченным списком $[\pi_1, \ldots, \pi_k]$, где π_i – номер цели, которая перехватывается в i-очередь. Пространство всех планов по перехвату k целей совпадает с множеством

$$\Pi_k = \{[\pi_1, \ldots, \pi_k] : \forall i = 1, \ldots, k \to \pi_i \in \{1, \ldots, n\}, \pi_i \neq \pi_j \longleftrightarrow i \neq j\}. \tag{7}$$

Например, для общего количества из $n = 2$ целей определены пространства планов $\Pi_0 = \{[\,]\}$, $\Pi_1 = \{[1], [2]\}$, $\Pi_2 = \{[1, 2], [2, 1]\}$. Наконец, пространство всех планов перехвата для n целей дается совокупностью

$$\Pi = \bigcup_{k=0}^{n} \Pi_k \tag{8}$$

всех планов конечной длины.

Однако часть планов из Π может быть некорректна, поскольку для этих планов могут найтись цели, которые достигнут защищаемой точки, прежде чем их перехватят. Формализируем данную идею, введя для произвольного плана $\pi = [\pi_1, \ldots, \pi_k]$ время, необходимое на его выполнение:

$$T(\pi) = \begin{cases} 0, & k = 0; \\ \tau_i(\mathbf{r_a}, \mathbf{r_i}, \mathbf{v_i}), & k = 1; \\ t + \tau_i(\mathbf{r_a}, \mathbf{r_i}, \mathbf{v_i}), & k > 2,\ t = T([\pi_1, \ldots, \pi_{k-1}]). \end{cases} \tag{9}$$

Теперь, следуя [1], пространство всех допустимых планов может быть определено как множество

$$\Pi_A = \{\pi \in \Pi : \forall j \in \{1, \ldots, k\} : T([\pi_1, \ldots, \pi_j]) \leqslant t_{\pi_j}\} \tag{10}$$

таких планов, для которых перехват целей происходит, прежде чем они достигают защищаемой точки.

Для дальнейшего удобства введем дополнительное определение. Под тактической обстановкой, связанной с планом π, понимается расположение перехватчика $\mathbf{r_a}$ и всех целей $\mathbf{r_i}$; $i = 1, \ldots, n,\ i \notin \pi$, не вошедших в план π на момент его завершения. Обращаем внимание, что для плана $\pi = [\pi_1, \ldots, \pi_k]$ со временем выполнения $t_\pi = T(\pi)$ величины, составляющие тактическую обстановку, вычисляются по формулам

$$\begin{cases} \mathbf{r_a} = r_{\pi_k}(t_\pi) \\ \mathbf{r_i} = \mathbf{r_i}(t_\pi), \quad i = 1, \ldots, n,\ i \notin \pi, \end{cases} \tag{11}$$

поскольку после завершения плана положение перехватчика и остальных целей определяется местом и временем достижения последней цели в π соответственно.

2.3. Критерий качества планов перехвата

Для постановки задачи оптимизации необходимо сформулировать критерии качества планов перехвата. Для этого на множестве допустимых планов Π_A зададим векторную функцию потерь

$$J[\pi] = (n_1[\pi], n_2[\pi], n_3[\pi], n_4[\pi]), \tag{12}$$

где

$$n_1[\pi] = \sum_{j=1}^{n} \mathrm{I}(j \notin \pi)$$

есть количество целей, пропущенных по итогу плана π (I – индикатор выполнения предиката), а

$$n_2[\pi] = \sum_{j=1}^{n} \mathrm{I}\left(|\mathbf{r_j}(t_{\pi_j})| \in [0, R/4]\right),$$

$$n_3[\pi] = \sum_{j=1}^{n} \mathrm{I}\left(|\mathbf{r_j}(t_{\pi_j})| \in (R/4, R/2]\right),$$

$$n_4[\pi] = \sum_{j=1}^{n} \mathrm{I}\left(|\mathbf{r_j}(t_{\pi_j})| \in (R/2, R]\right)$$

равны числу целей, которые на момент перехвата по плану π приблизились к защищаемой точке на расстояния от 0 до $\frac{R}{4}$, от $\frac{R}{4}$ до $\frac{R}{2}$, от $\frac{R}{2}$ до R соответственно. Заметим, что данные диапазоны расстояний являются параметрами задачи и могут быть выбраны и уточнены для конкретных постановок. Также обратим внимание, что сравнение двух планов перехвата по данному критерию осуществляется согласно стандартному словарному упорядочиванию.

Минимизация потерь по данному критерию равносильна перехвату как можно большего числа целей на как можно большем удалении от защищаемой точки. Действительно, чем больше целей было пропущено к защищаемой

точке, тем хуже; при одинаковом количестве пропущенных целей необходимо сравнивать число целей, перехваченных вблизи защищаемой точки, и чем их меньше, тем лучше; при прочих равных условиях, необходимо сравнивать количество целей, перехваченных на среднем и большом удалении от защищаемой точки. Причем в качестве характерной меры масштаба задачи разумно принять радиус R области, на границе которой появляются цели.

Поскольку для любого допустимого плана $\pi \in \Pi_A$ верно, что

$$n = n_1[\pi] + n_2[\pi] + n_3[\pi] + n_4[\pi],$$

то общее число целей n является верхней границей для $n_1[\pi], \ldots, n_4[\pi]$. Поэтому значение функции потерь $J[\pi]$ можно рассматривать как запись числа в позиционной системе счисления с основанием $n+1$, что позволяет ввести числовой эквивалент функции потерь:

$$J_n[\pi] = n_1[\pi](n+1)^3 + n_2[\pi](n+1)^2 + n_3[\pi](n+1) + n_4[\pi]. \tag{13}$$

Эти две функции потерь равносильны, так как $J[\pi] < J[\pi'] \Longleftrightarrow J_n[\pi] < J_n[\pi']$. Для дальнейшего описания введем также нормированную функцию качества

$$J_q[\pi] = \frac{n(n+1)^3 + 1}{n(n+1)^3 - 1} - \frac{2J_n[\pi]}{n(n+1)^3}, \tag{14}$$

которая планам с наименьшей возможной фукнцией потерь $(0,0,0,n)$ ставит оценку 1, а с наихудшей $(n,0,0,0)$ – -1.

Теперь оптимизационную задачу можно сформулировать следующим образом: по начальной обстановке найти допустимый план перехвата π^* с минимально возможной функцией потерь $J_q[\pi]$:

$$\begin{cases} \mathbf{r_1^0}, \ldots, \mathbf{r_n^0} \\ \mathbf{v_1^0}, \ldots, \mathbf{v_n^0} \end{cases} \longrightarrow \pi^* \in \Pi_A : \pi^* = \operatorname{argmin} J_q[\pi]. \tag{15}$$

Поскольку найти точное решение вышеуказанной задачи весьма затруднительно, разумной также является задача поиска субоптимальных планов, на которых функция потерь принимает значение, близкое к оптимальному.

3. Алгоритм Монте-Карло для построения субоптимального плана перехвата

3.1. Принципиальная схема алгоритма

Для поиска субоптимального плана предлагается алгоритм поиска по дереву Монте-Карло. Его основная идея состоит в построении дерева поиска решения посредством Монте-Карло симуляций. Под деревом поиска понимается граф планов $S = (W, E)$, где W и E – множества его вершин и ребер

соответственно. Вершины $w \in W$ помечаются допустимыми планами перехвата $\pi_w \in \Pi_A$, причем вершина w_2 соединяется ребром с w_1, только если план π_{w_2} продолжает план π_{w_1} ровно на одну цель:

$$(w_1, w_2) \in E \Longleftrightarrow \pi_{w_1} = [\pi_1, \ldots, \pi_{k-1}] \wedge \pi_{w_2} = [\pi_1, \ldots, \pi_{k-1}, \pi_k]. \tag{16}$$

Для удобства назовем вершину $w_0 \in W$, которая помечена пустым планом [ ], корнем дерева, а потомками вершины $w \in W$ – такие $w' \in W$, для которых план $\pi_{w'}$ есть продолжение π_w. Заметим, что для любой $w \in W$ существует единственный путь, который соединяет w с корнем. Таким образом, $w \in W$ всегда лежит на пути от корня до любого из своих потомков.

Базовый цикл алгоритма состоит из трех шагов:

1. Спуск по дереву от корня $w_0 \in W$ до вершины $w \in W$, которая не посещалась алгоритмом на предыдущих итерациях.
2. Построение для вершины w случайного продолжения плана перехвата π_w, соответствующего данной вершине.
3. Оценка полученного продолжения по функции качества (14) и поднятие от последнего результата к корню дерева.

Далее, пусть p_w задает количество прохождений алгоритма через вершину $w \in W$ при спусках по дереву, а q_w – это средняя оценка случайных планов, построенных для вершины $w \in W$ и ее потомков. Значение p_w возрастает на единицу при каждом проходе алгоритма через $w \in W$ на очередном спуске. А величина q_w пересчитывается при поднятии случайной оценки от одного из потомков $w \in W$ обратно к корню. Поэтому количество случайных оценок, которые усредняются для вычисления q_w, в точности равняется p_w. Например, для корня дерева q_{w_0} совпадает со средней оценкой всех построенных случайных планов. Таким образом, каждая итерация базового цикла приводит

1. К просмотру одной новой вершины $w_1 \in W$ дерева поиска, для которой $p_{w_1} = 0$ до этого момента.
2. К построению для w_1 случайного продолжения плана перехвата π_{w_1}, соответствующего данной вершине.
3. К пересчету чисел p_w и q_w для всех вершин $w \in W$ дерева поиска, находящихся на пути от корня w_0 до w_1.

Базовый цикл повторяется до исчерпания вычислительного лимита, отведенного на решение задачи. Затем по результатам накопленной статистики по числам p_w и q_w строится итоговый план перехвата. Поэтому для полного описания алгоритма необходимо указать следующие его особенности:

1. Метод спуска по дереву и использованные для его построения эвристики.
2. Способ случайного продолжения плана π_w для произвольной вершины $w \in W$ дерева поиска.
3. Механизм извлечения результирующего плана из дерева поиска по числам p_w и q_w.

Это будет выполнено в следующих подразделах.

3.2. Метод спуска по дереву поиска на основе функции полезности

Спуск по дереву поиска начинается от корня $w_0 \in W$ и осуществляется следующим образом.

1. Если текущая вершина w ранее не посещалась, т.е. если $p_w = 0$, то спуск останавливается на w и алгоритм переходит к построению случайного продолжения для плана π_w .
2. Иначе необходимо рассмотреть все смежные вершины w_i, т.е. такие $w_i \in W$, что $\exists(w, w_i) \in E$, и перейти из них к доставляющей максимум функции полезности

$$u(w_i, w) = q_{w_i} + (\theta(w_i) + P(w, w_i))\frac{\sqrt{2p_w}}{1 + p_{w_i}}. \tag{17}$$

Здесь $\theta(w_i)$ и $P(w, w_i)$ – это две эвристики, задающие направление спуска совместно с p_w и q_w. Функция $\theta(w)$ оценивает предполагаемое наилучшее продолжение плана π_w в диапазоне $[-1; 1]$, а $P(w, w_i)$ – вероятность того, что данное продолжение достигается в направлении w_i.

Для произвольной вершины $w \in W$

$$p_w = 1 + \sum_{w_i} p_{w_i},$$

где сумма берется по всем w_i, смежным с w. Поэтому в (17) множитель $\frac{\sqrt{2p_w}}{1+p_{w_i}}$, с которым $\theta(w_i)$ и $P(w, w_i)$ входят в $u(w_i, w)$, убывает с ростом p_{w_i}. Таким образом, при спуске к потомку w_i с большим p_{w_i} главную роль играет высокая оценка q_{w_i}. В свою очередь, при малом p_{w_i} среднее q_{w_i} не должно играть роли из-за малости выборки, по которой оно находится. В этом случае $\theta(w_i)$ и $P(w, w_i)$ приобретают в (17) больший вес по сравнению с оценкой q_{w_i}.

Вместо функции полезности (17) также может использоваться функция

$$u(w_i, w) = q_{w_i} + \frac{\sqrt{2}\ln(p_w)}{p_{w_i}},$$

где при $p_{w_i} = 0$ вершине w_i приписывается бесконечная полезность. Данный вид функции полезности называется верхней границей уверенности (upper confidence bounds) и был впервые предложен в [14] для поиска оптимальной стратегии при игре в многорукого бандита. Приложение данной функции полезности к проблеме построения алгоритма поиска по дереву Монте-Карло обсуждалось в [15]. Однако для задачи более эффективной оказалась функция полезности в виде (17), которая использовалась в [9] для игры в Го, за счет наличия в ней эвристических функций.

Теперь перейдем к конкретным функциям $\theta(w)$ и $P(w, w_i)$, которые использовались при построении алгоритма. Для этого введем понятие уточненной оценки $\hat{J}_q[\pi]$ для произвольного плана π, под которой будем понимать

оценку $J_q[\pi]$, найденную по точкам возможного прямого перехвата целей по итогу плана π (для целей, вошедших в план π, в расчет берутся точки их перехвата по плану π). Тогда функцию $\theta(w)$ можно представить в следующем виде:

$$\theta(w) = \begin{cases} \hat{J}_q[\pi_w], & p_w > 0 \\ J_q[\pi_w], & p_w = 0. \end{cases} \tag{18}$$

Для вершин $w \in W$ с $p_w = 0$ в качестве $\theta(w)$ используется обычная оценка $J_q[\pi_w]$, а для всех остальных – уже уточненная $\hat{J}_q[\pi_w]$. Это экономит вычислительные ресурсы при нахождении функции полезности (17) на вершинах с малыми числами посещений p_w.

В свою очередь, для нахождения $P(w, w_i)$ выбрано нормированное на единицу распределение

$$P(w, w_i) = \frac{\tau^* - \tau_i + \tau_*}{\sum\limits_{i=1}^{n} (\tau^* - \tau_i + \tau_*)}, \tag{19}$$

где $\tau^* = \max\limits_i \tau_i$, $\tau_* = \min\limits_i \tau_i$, а τ_i – длительность прямого перехвата i-цели, который можно выполнить сразу после выполнения плана π_w.

3.3. Построение случайных продолжений

Для продолжения плана π_w на вершине $w \in W$ алгоритм использует распределение вероятностей

$$P(i \notin \pi_w) = \frac{d_i}{\sum\limits_{i \notin \pi_w} d_i} \tag{20}$$

на целях, не вошедших в план π_w, где d_i – это опасности, введенные согласно (3). Затем план π^i последовательно достраивается назначением по одной дополнительной цели по данному распределению вероятности до полного исчерпания доступных для перехвата целей.

Другим способом построения продолжений для планов π_w является обход непосещенных целей в порядке убывающей опасности. Хотя данный способ продолжения не содержит в себе случайного элемента, в силу специфики данной задачи, он оказывается оптимальным для построения алгоритма.

3.4. Построение итогового плана перехвата алгоритмом Монте-Карло

Итоговый план перехвата строится по результирующему дереву поиска после исчерпания лимита вычислений (ограничения на число итераций или времени выполнения). Для этого алгоритм дополнительно отслеживает наилучшую оценку J_q^* из всех случайных оценок, построенных при повторении

базового цикла, и соответствующий данной оценке план перехвата π_q^*. После завершения вычислений алгоритм в качестве ответа возвращает найденный таким образом план π_q^*.

Данный подход позволяет внести ряд улучшений в принципиальную схему алгоритма, изложенную в разделе 3.1. Действительно, для вершин $w \in W$ дерева поиска становится возможной проверка на неоптимальность. Так, если для произвольной $w \in W$ оценка $J_q[\pi_w]$ плана π_w оказывается хуже наилучшей достигнутой оценки J_q^*, то, значит, среди продолжений плана π_w нет оптимального решения и эту ветвь поиска следует отсечь. Поэтому при спуске по дереву такие вершины следует помечать как неоптимальные. Также вершину $w \in W$ следует пометить как неоптимальную в случае, если все ее прямые потомки уже помечены как неоптимальные. Если в процессе спуска достигается некоторая вершина $w \in W$, которую по описанным выше принципам алгоритм помечает как неоптимальную, то процесс спуска поднимается на одну вершину к корню и затем заново принимает решение о направлении спуска, оптимизируя функцию полезности (17) по усеченному множеству вариантов. В качестве исходного значения оценки J_q^* в силу специфики задачи выбирается оценка плана π_d, который получается путем упорядочивания целей по опасностям от самой опасной к наименее опасной.

Текущая оптимальная оценка J_q^* также используется для оптимизации процедуры продолжения произвольного плана π. А именно, данное продолжение осуществляется, пока текущая оценка, достигнутая на этом продолжении, не становится хуже оптимальной. Если данная ветвь продолжения оказывается заведомо неоптимальной, продолжения прекращаются и к корню дерева поиска поднимается минимально возможнная оценка -1, что существенно понижает оценки вершин, для которых типичное продолжение оказалось неоптимальным.

Наконец, если два плана π_1 и π_2 отличаются друг от друга лишь порядком целей и при этом совпадают по последней цели, то тогда худший их них является заведомо неоптимальным и соответствующая ему вершина должна быть удалена из дерева поиска.

Таким образом, описанные выше меры позволяют отсекать заведомо неоптимальные ветви дерева поиска, что существенно повышает эффективность алгоритма, как это будет показано на конкретных примерах в следующем разделе.

4. Моделирование работы алгоритма

Приведем сравнение алгоритма с полным перебором в глубину с отсечением вариантов по методу ветвей и границ. Для моделирования работы алгоритма рассматривались начальные позиции со случайной расстановкой целей в рамках постановки задачи из раздела 2. Для задачи с 10 атакующими целями, оба алгоритма нашли оптимальный план, как это показано на рис. 1 и 2. Также приводим сводную табл. 1 по качеству построенных планов и времени работы. Как видно, при малом числе целей полнопереборный алгоритм

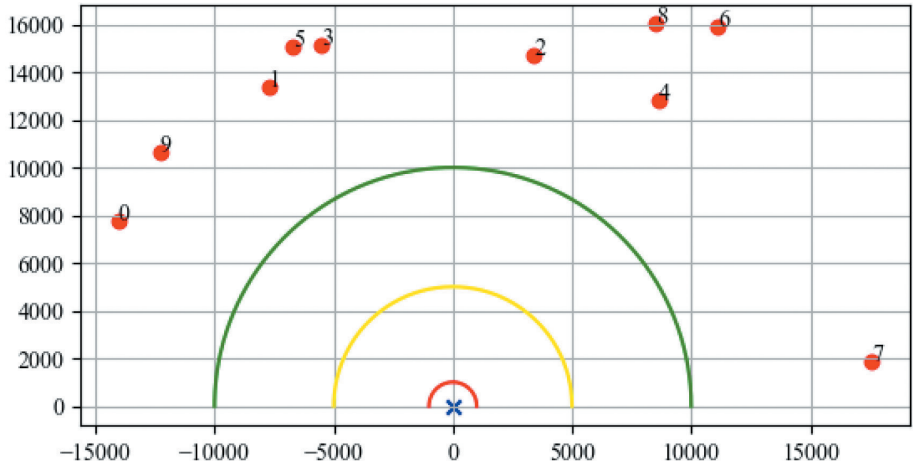

Рис. 1. Начальное положение целей и перехватчика для задачи с 10 целями.

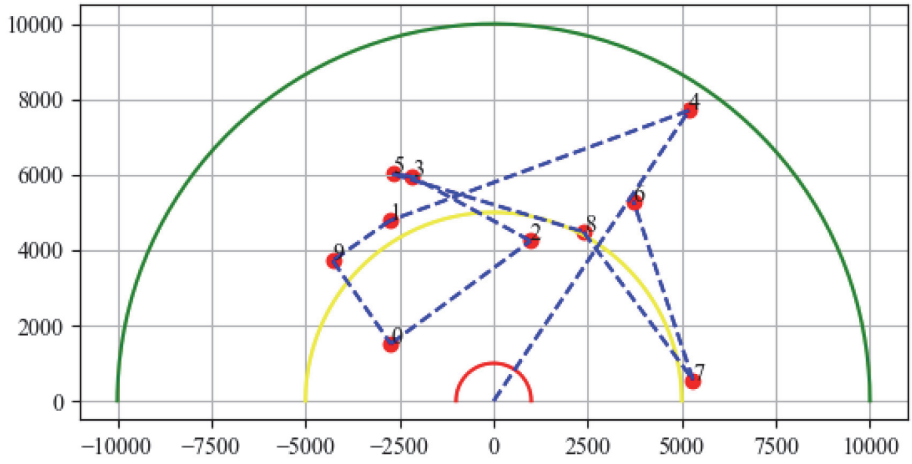

Рис. 2. Оптимальный план перехвата, построенный алгоритмами полного перебора и Монте-Карло.

оказывается значительно быстрее. Это связано с тем, что время работы алгоритма поиска деревом было задано равным одной секунде безотносительно размера задачи.

Таблица 1. Сводные данные о работе алгоритмов для задачи с 10 целями

Тип Алгоритма	Оценка качества	Время работы
Полный перебор	(0, 2, 8, 0)	0,05 с
Монте-Карло	(0, 2, 8, 0)	1 с

Теперь перейдем к примеру с 15 целями, начальное положение которых приведено на рис. 3. В этом случае оба алгоритма также справились с на-

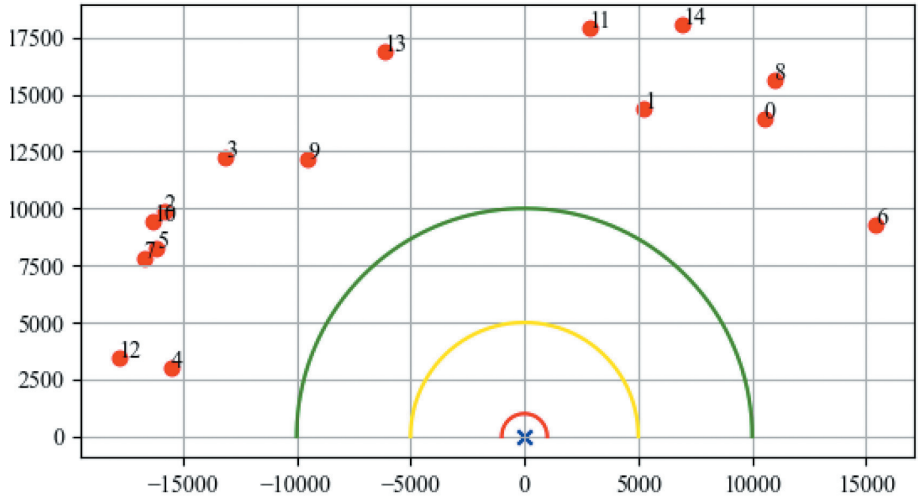

Рис. 3. Начальное положение целей и перехватчика для задачи с 15 целями.

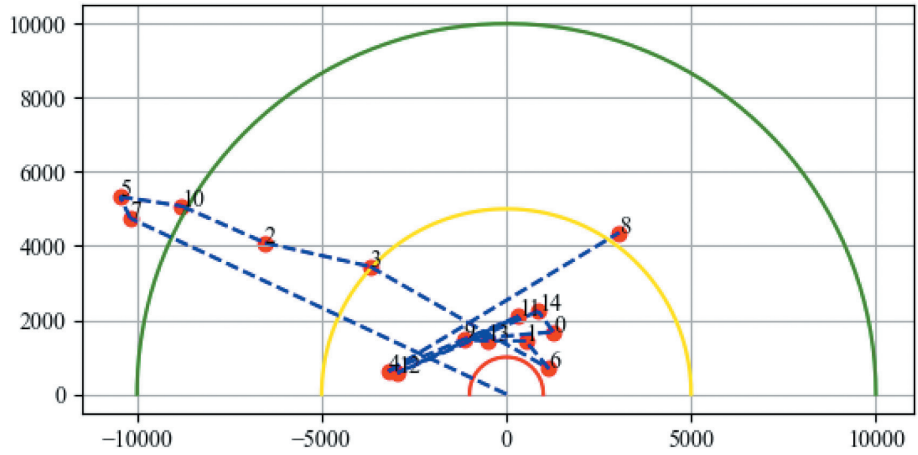

Рис. 4. Оптимальный план перехвата, построенный полным перебором и по поиску деревом для 15 целей.

хождением оптимального плана, показанного на рис. 4. Однако теперь полнопереборный алгоритм опередил алгоритм Монте-Карло лишь на полсекунды вычислительного времени, как это следует из табл. 2.

Таблица 2. Сводные данные о работе алгоритмов для задачи с 15 целями

Тип Алгоритма	Оценка качества	Время работы
Полный перебор	(0, 9, 3, 3)	0,48 с
Монте-Карло	(0, 9, 3, 3)	1 с

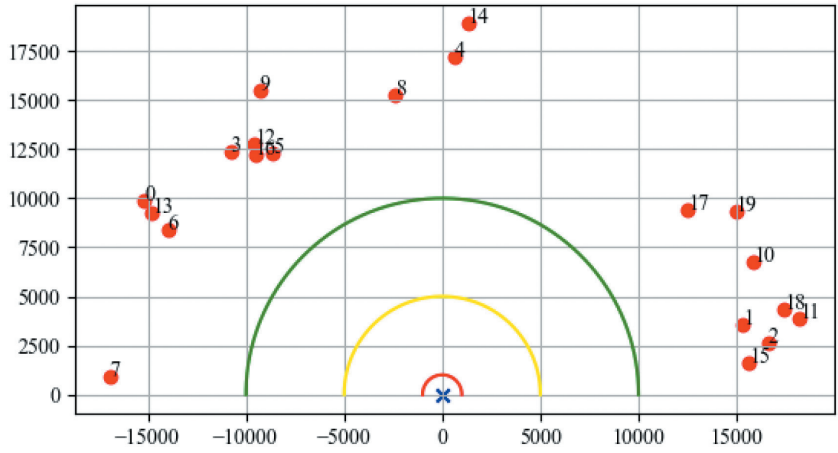

Рис. 5. Начальное положение 20 целей.

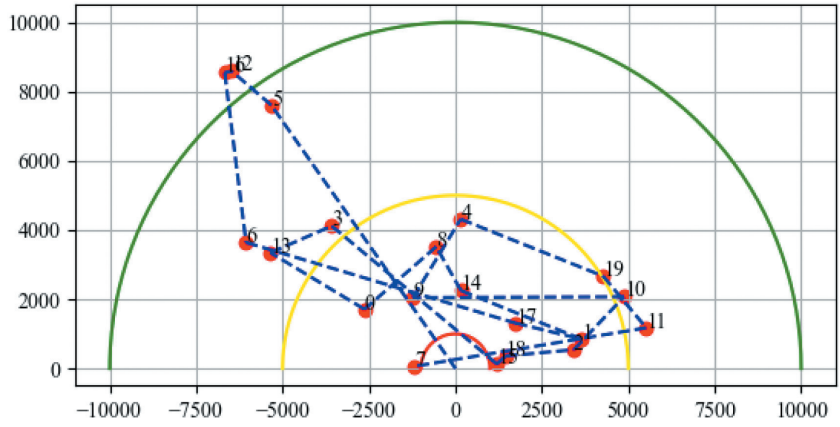

Рис. 6. Оптимальный план перехвата, построенный алгоритмом полного перебора для 20 целей.

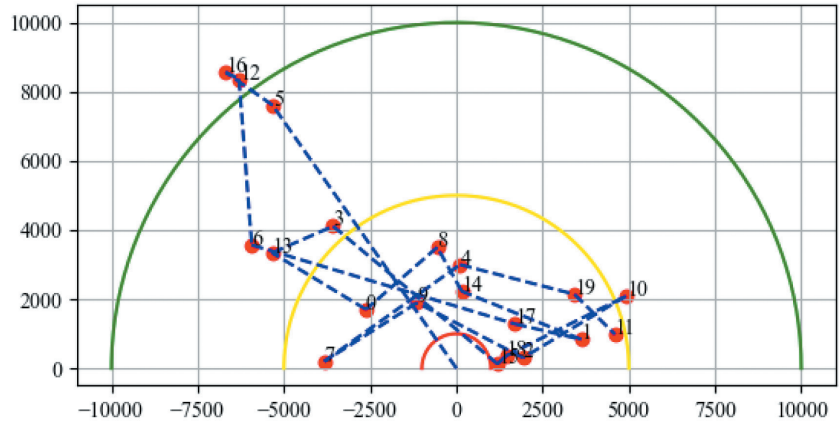

Рис. 7. План перехвата, построенный алгоритмом Монте-Карло для 20 целей.

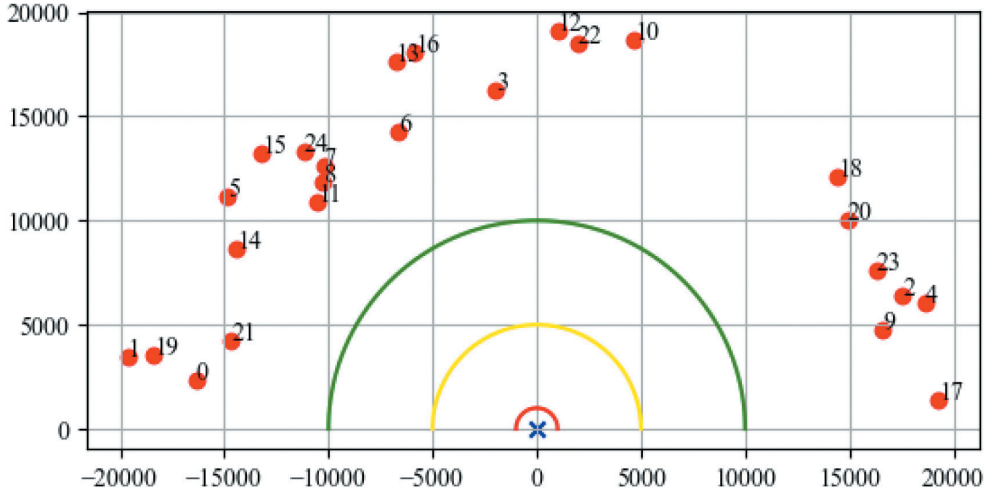

Рис. 8. Начальное положение 25 целей.

Для задачи с 20 целями, приведенной на рис. 5, алгоритм поиска построил план перехвата без пропуска целей в ближнюю зону, который, однако, был неоптимальным. Сравнение двух планов приведено на рис. 6 и 7 и в табл. 3, обращаем внимание, что на этом размере входных данных поиск деревом оказался в 4 раза быстрее полного перебора.

Таблица 3. Сводные данные о работе алгоритмов для задачи с 20 целями

Тип Алгоритма	Оценка качества	Время работы
Полный перебор	(0, 11, 7, 2)	4,07 с
Монте-Карло	(0, 13, 5, 2)	1 с

Для задачи с 25 целями поиск деревом смог построить план перехвата лишь с пропуском одной цели. Однако время его работы оказалось намного меньшим по сравнению с полнопереборным алгоритмом. Начальное положение целей для этой задачи показано на рис. 8, построенные планы перехвата приведены на рис. 9 и 10, а время работы алгоритмов и оценки полученных планов указаны в табл. 4.

Таблица 4. Сводные данные о работе алгоритмов для задачи с 25 целями

Тип Алгоритма	Оценка качества	Время работы
Полный перебор	(0, 15, 10, 0)	13,63 с
Монте-Карло	(1, 19, 4, 1)	1 с

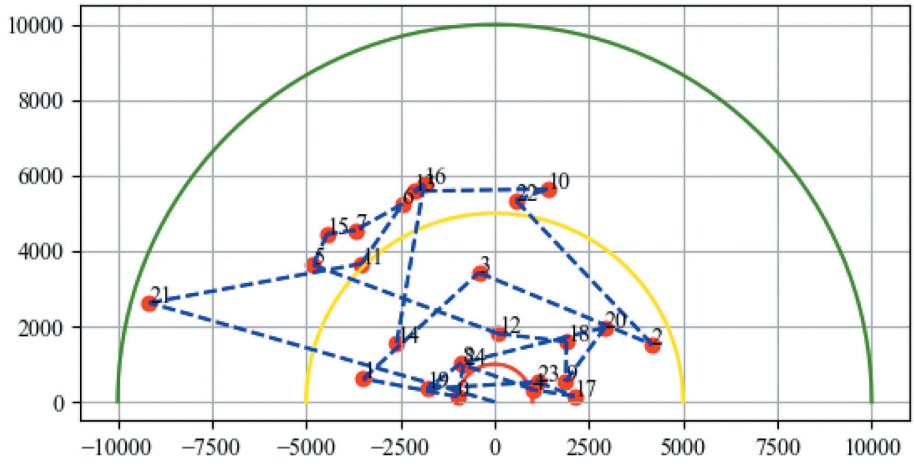

Рис. 9. Оптимальный план перехвата, построенный алгоритмом полного перебора для 25 целей.

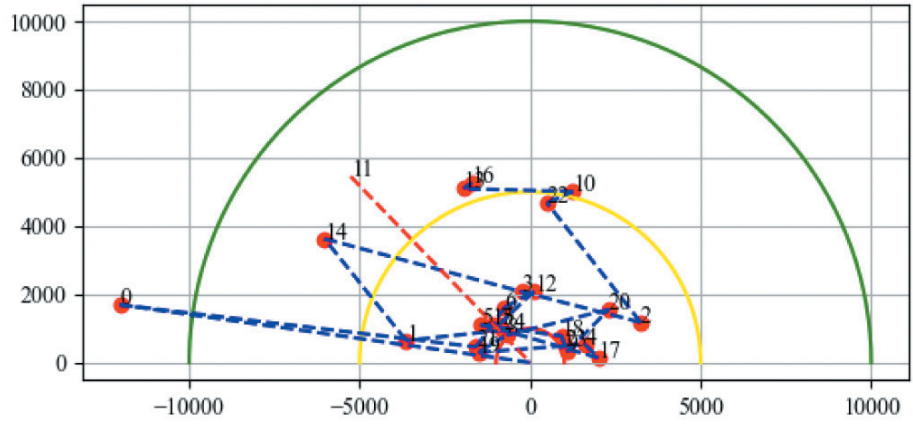

Рис. 10. План перехвата, построенный алгоритмом Монте-Карло для 25 целей. Красной линией показана траектория пропущенной цели.

5. Заключение

В данной работе предложен алгоритм поиска деревом Монте-Карло при построении субоптимального плана перехвата для обороны защищаемой точки от прямолинейно движущихся целей. Этот метод позволяет эффективно строить приемлемые решения даже для случаев с большим количеством атакующих целей, при которых полнопереборные алгоритмы могут оказаться слишком медленными для практических приложений. Однако остается возможность использования двух алгоритмов одновременно и осуществлять выбор по результатам их работы в режиме реального времени. Подход, предложенный в данной работе, также может быть прямо обобщен на случай обо-

роны защищаемой точки несколькими перехватчиками против существенно большего числа атакующих целей.

Кроме того, предложенный метод решения подходит не только для рассматривавшейся задачи, но и для любой другой проблемы дискретной оптимизации. Расмотренная схема может различным образом оптимизироваться и улучшаться для приложения к самым разнообразным задачам. Широкий разбор последних версий и приложений алгоритма поиска деревом Монте-Карло приводится в [16]. Таким образом, открывается широкий выбор возможных приложений описанной в данной работе схемы построения приближенных решений в других задачах дискретной оптимизации.

Другим возможным направлением для дальнейших исследований по данной теме является применение нейросетевых подходов совместно с поиском по дереву Монте-Карло. Так, обученные нейросети могут применяться для вычисления эвристик θ и P, которые входят в функцию полезности (17). Использование нейросетей при вычислении эвристик для алгоритма поиска по дереву Монте-Карло уже показало себя чрезвычайно эффективным при построении программ для игры в Го [9]. Поэтому адоптация такого подхода может оказаться столь же плодотворной и для решения динамической задачи коммивояжера и других подобных задач дискретной оптимизации, что может стать темой для дальнейших исследований по данному вопросу.

СПИСОК ЛИТЕРАТУРЫ

1. *Галяев А.А., Яхно В.П., Берлин Л.М., Лысенко П.В., Бузиков М.Э.* Оптимизация плана перехвата прямолинейно движущихся целей // А и Т. 2023. № 10. С. 18–36.
2. *Сихарулидзе Г.Г.* Об одном обобщении задачи коммивояжера. I // А и Т. 1971. № 8 С. 116–123.
3. *Сихарулидзе Г.Г.* Об одном обобщении задачи коммивояжера. II // А и Т. 1971. № 10. С. 142–147.
4. *Picard J.C., Queyranne M.* The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling // Oper. Res. 1978. V. 26. No. 1. P. 86–110. DOI: 10.1287/opre.26.1.86
5. *Helvig C.S., Robins G., Zelikovsky A.* The moving-target traveling salesman problem // J. Algorithm. Comput. Technol. 2003. V. 49. No. 1. P. 153–174. https://doi.org/10.1016/S0196-6774(03)00075-0
6. *Garey M.R., Johnson D.S.* Computers and intractability: A guide to the theory of NP-completeness. San Francisco, Calif.: W. H. Freeman & Co., 1979.
7. *Li C., Yang M., Kang L.* A New Approach to Solving Dynamic Traveling Salesman Problems. In: Wang, TD., et al. Simulated Evolution and Learning // Lecture Notes Comput. Sci. 2006. V. 4247. Springer, Berlin, Heidelber.
8. *Archetti C., Feillet D., Mor A., Speranza M.G.* Dynamic traveling salesman problem with stochastic release dates // Eur. J. Oper. 2020. V. 280. I. 3. P. 832–844. ISSN 0377-2217

9. *Silver D., Huang A., Maddison C. et al* Mastering the game of Go with deep neural networks and tree search // Nature. 28 January 2016. 529 (7587): P. 484–489. https://doi.org/10.1038/nature16961.
10. *Schadd M.P.D., Winands M.H.M., van den Herik H.J., Chaslot G.M.J.B., Uiterwijk J.W.H.M.* (2008). Single-Player Monte-Carlo Tree Search // Computers and Games. CG 2008. Lecture Notes in Computer Science, vol 5131. Springer, Berlin, Heidelberg.
11. *Mattia Crippa, Pier Luca Lanzi, Fabio Marocchi.* An analysis of Single-Player Monte Carlo Tree Search performance in Sokoban // Expert Syst. Appl. 15 April 2022. V. 192. P. 2–3.
12. *Cotarelo A., Vicente G., Edward Rolando N., Cristian G., Alberto G., Jerry Ch.* Improving Monte Carlo Tree Search with Artificial Neural Networks without Heuristics. Appl. Sci. 2021. V. 11, No. 5. 2056. https://doi.org/10.3390/app11052056
13. *Marco K.* Beyond Games: A Systematic Review of Neural Monte Carlo Tree Search Applications // arXiv:2303.08060, https://doi.org/10.48550
14. *Auer P., Cesa-Bianchi N., Fischer P.* Finite-time Analysis of the Multiarmed Bandit Problem // Machine Learning. 2002. V. 47. P. 235–256. https://doi.org/10.1023/A:1013689704352
15. *Kocsis L., Szepesvari C.* Bandit Based Monte-Carlo Planning. Furnkranz J., Scheffer T., Spiliopoulou M. (eds) // Machine Learning: ECML 2006. ECML 2006. Lecture Notes Comput. Sci. V. 4212. Springer, Berlin, Heidelberg.
16. *Swiechowski M., Godlewski K., Sawicki B. et al.* Monte Carlo Tree Search: a review of recent modifications and applications // Artif. Intell. Rev. 2023 V. 56. P. 2497–2562. https://doi.org/10.1007/s10462-022-10228-y

Статья представлена к публикации членом редколлегии А.А. Лазаревым.

Поступила в редакцию 05.10.2023

После доработки 04.12.2023

Принята к публикации 21.12.2023