ISSN 0005-1179 (print), ISSN 1608-3032 (online), Automation and Remote Control, 2024, Vol. 85, No. 2, pp. 177–189.




# Searching for a Sub-Optimal Solution of the Dynamic Traveling Salesman Problem Using the Monte Carlo Method

**A. A. Galyaev**[*,a] **and E. A. Ryabushev**[*,b]

[*] *Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Moscow, Russia*
*e-mail:* [a] *galaev@ipu.ru,* [b] *lispandhaskell@gmail.com*

Received October 5, 2023
Revised December 4, 2023
Accepted December 21, 2023

**Abstract**—The problem of drawing up a bypass plan for targets moving rectilinearly to one point for simple movements of an interceptor (traveling salesman) is considered. A new criterion of the problem is proposed based on the initial partition of the possible intercept area, as well as an algorithm for finding a sub-optimal bypass plan based on the construction of a solution search tree by the Monte Carlo method. A numerical implementation of the algorithm has been developed, modeling has been carried out and the obtained plans for bypassing targets have been statistically analyzed.

*Keywords*: moving targets traveling salesman problem, combinatorial optimization, interception in simple motions, Monte Carlo algorithm

**DOI:** 10.31857/S0005117924020063

## 1. INTRODUCTION

This work is dedicated to the problem of defence of a specified point in space from the attack of linearly moving targets. It is proposed to guard the point using an interceptor, which is capable of freely moving in space and destroying the attacking targets. The selection of an optimal order of interception of the approaching targets plays key role in this tusk. This problem has already been discussed in [1], where concepts of danger, convenience, and difficulty of a target interception were considered, and vector quality criteria for target interception plans were proposed. In [1], an intelligent algorithm was constructed based on exhaustive search in the general case, capable of efficiently finding interception plans for a single interceptor against up to twenty attacking targets. In this work, a different algorithm is proposed, capable of finding a sub-optimal interception plan that is able to operate with acceptable efficiency in the case of the attack of more than twenty targets.

The well-known "Dynamic Traveling Salesman Problem" (DTSP) [2, 3] or Moving Targets Traveling Salesman Problem (MTTSP) [4, 5] is conceptually close to the problem investigated in this work. The DTSP generalizes the Traveling Salesman Problem (TSP), and in 1972, the NP-completeness of the Hamiltonian Cycle problem was proven, implying the NP-completeness of TSP and, consequently, DTSP [6]. At present, there are no efficient methods for exactly solving DTSP, and the problem itself is in the early stages of research [7]. This suggests that in general, constructing an exact interception plan can be quite challenging when dealing with a large number of targets, hence it is reasonable to consider the question of quickly finding a sub-optimal solution. Currently, there are approaches to constructing such DTSP solutions based on genetic algorithms, which can be further explored, for instance, in [7], or local heuristic optimization [8].

One of the methods for finding sub-optimal solutions to discrete optimization problems is the Monte Carlo Tree Search algorithm [9, 10]. This method gained wide recognition after its successful application to the game of Go as part of the AlphaGo project [9], where Monte Carlo tree search was used in conjunction with neural network learning. In addition to antagonistic games, this algorithm in various variations has been applied to single-player games, which can be interpreted as discrete optimization problems [10, 11]. Additionally, the Monte Carlo Tree Search algorithm provides a convenient basis for applying neural network approaches to discrete optimization problems. Thus, neural network learning can be used to compute heuristic utility functions and to construct probability distributions, based on which random continuations are generated. Therefore, the basic tree search algorithm allows breaking down the original problem into simpler components that can be solved by trained neural networks. A more detailed discussion of possible ways to combine neural network approaches with Monte Carlo tree search can be found in [9, 12, 13]. In this work, the fundamental ideas of the MCTS algorithm are applied to the problem of defence of a point from the attack of moving targets. To achieve this, a simple motion model is used for the interceptor, as the distances between targets during movement are significantly larger compared to the actual turning radius of the interceptor. Under these assumptions, considering maneuverability in the target interception planning does not affect the structure of the optimal plan, allowing us to transition from a discrete-continuous optimization problem to a discrete one. Another distinction of the current work from well-known literature setups is the problem criterion, which is meaningful for practical applications and whose optimal value is generally achievable on a set of plans, i.e., it is not unique. Thus, the main proposition of this work lies in applying a new approach, initially developed for solving game problems, to the problem of constructing a sub-optimal solution for a variant of the dynamic traveling salesman problem with the defense of a guarded point against attacking targets. The proposed approach allows obtaining such solutions in a manner not previously used for these purposes and may serve as an alternative to more well-known approaches based on genetic algorithms, which have been previously applied in similar problems.

## 2. MATHEMATICAL MODEL AND PROBLEM STATEMENT

The mathematical model and problem statement used in this work coincide with those proposed in [1]. However, for the sake of completeness, we provide their description in this section.

### *2.1. Model of Target and Interceptor Motion*

We assume that the interceptor needs to intercept $n$ targets that move linearly in a single plane with velocities in the range $[v_{\min}, v_{\max}]$. The protected point is located on the plane at the origin, and the targets appear on the outer boundary of a circle with radius $R$ within a layer of thickness $2\Delta R$. The interceptor can move in simple motions at a speed $v(t) < V$, where $V > v_{\max}$.

The initial positions of the targets are given by the position vectors $\mathbf{r_1^0}, \ldots, \mathbf{r_n^0}$, where $\mathbf{r_i^0} = (x_i^0, y_i^0)$. We assume that the velocities of the targets $\mathbf{v_1}, \ldots, \mathbf{v_n}$ are known, and their motion is described by linear equations

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

It is also assumed that the trajectories of all targets pass through the origin, and thus, the time of arrival of a target $i$ at the defended point is given by

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

The inverse of this time, termed the danger of the target, is denoted as

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

Thus, the initial tactical situation is determined by the initial positions of all targets together with their velocity vectors, along with the initial position of the interceptor.

The motion of the interceptor is determined by the system of differential equations

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

where $\mathbf{r_a} = (x_a, y_a)$ represents the position of the interceptor, $v(t)$ denotes the magnitude of its velocity, and $\psi(t)$ represents the control of its motion direction. The minimum time $\tau_i$ required for the interceptor to intercept target $i$ is determined from the quadratic equation

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

as its minimum positive root

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

Thus, the target interception function is the solution to the target's fastest interception problem. This model is sufficient for finding the optimal plan for the interceptor moving in the class of simple motions, as the principles of minimizing standstill and moving at maximum speed apply. This was proven in [1] for optimizing the interception plan based on the number of missed targets and interception time. In the present work, a different optimization criterion will be used, but the model of the fastest interception of a single target will still be employed.

### *2.2. Interception Plan*

The plan $\pi$ for intercepting $k$ targets is defined as an ordered list $[\pi_1, \ldots, \pi_k]$, where $\pi_i$ denotes the number of the target intercepted at position $i$. The space of all interception plans for $k$ targets coincides with the set

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

For example, for a total of $n = 2$ targets, the plan spaces are defined as $\Pi_0 = \{[\,]\}$, $\Pi_1 = \{[1], [2]\}$, $\Pi_2 = \{[1, 2], [2, 1]\}$. Finally, the space of all interception plans for $n$ targets is given by the union

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

of all plans of finite length.

However, some plans from $\Pi$ may be incorrect, as there may be targets that reach the defended point before being intercepted. To formalize this idea, let's introduce, for an arbitrary plan $\pi = [\pi_1, \ldots, \pi_k]$, the time required for its execution:

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

Now, following [1], the space of all valid plans can be defined as the set

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

of plans for which interception of the targets occurs before they reach the defended point.

For further convenience, let's introduce an additional definition. The tactical situation associated with plan $\pi$ refers to the positions of the interceptor $\mathbf{r_a}$ and all targets $\mathbf{r_i}$; $i = 1, \ldots, n,\ i \notin \pi$, that are not included in the plan $\pi$ at the moment of its completion. It should be noted that for a plan $\pi = [\pi_1, \ldots, \pi_k]$ with execution time $t_\pi = T(\pi)$, the quantities comprising the tactical situation are computed by the formulas

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

This is because after the plan is completed, the position of the interceptor and the other targets is determined by the location and time of arrival to the last target in $\pi$ respectively.

### *2.3. Interception Plan Criterion*

To formulate the optimization problem, it is necessary to define criteria for the quality of the interception plans. For this purpose, we will define a vector loss function on the set of admissible plans $\Pi_A$.

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

where

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

is the number of targets that reached the defended point at the end of the plan $\pi$ (I – indicator of a successful interception), and

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

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

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

are the numbers of targets that were intercepted in the distances from the defended point from 0 to $\frac{R}{4}$, from $\frac{R}{4}$ to $\frac{R}{2}$, from $\frac{R}{2}$ to $R$, respectively. Note that these distance ranges are parameters of the problem and can be selected and refined for specific scenarios. Also, it should be noted that comparing two interception plans according to this criterion is done according to the standard lexicographic ordering.

Minimization of the losses according to this criterion is equivalent to interception of as many targets as possible at the greatest possible distance from the defended point. Indeed, the more targets have reached the defended point, the worse the outcome. When the number of missed targets is the same, it is necessary to compare the number of targets intercepted near the defended point, and the fewer, the better. Under other equal conditions, it is necessary to compare the number of targets intercepted at medium and long distances from the defended point. Moreover, it is reasonable to take the radius $R$ of the area where targets appear as a characteristic scale of the problem.

As for every admissible plan $\pi \in \Pi_A$ it is true that

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

then the total number $n$ is the upper bound for $n_1[\pi], \ldots, n_4[\pi]$. Therefore, the loss function $J[\pi]$ can be interpreted as a description of a number in positional counting system with base $n + 1$,

which allows to introduce a numerical equivalent of the loss function:

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

These two loss functions are equivalent, because $J[\pi] < J[\pi'] \Longleftrightarrow J_n[\pi] < J_n[\pi']$. We also introduce the normalized loss function

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

that relates plans with the best losses $(0, 0, 0, n)$ to the estimate 1, and with the worst losses $(n, 0, 0, 0)$ – $-1$.

Now the optimization tusk can be formulated as the following: from the starting tactical situation find the admissible interception plan $\pi^*$ with minimal possible loss function $J_q[\pi]$:

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

Since finding the exact solution to this problem is quite challenging, it is reasonable to search for the sub-optimal plans on which the loss function is to its optimal value.

## 3. MONTE CARLO ALGORITHM FOR CONSTRUCTING A SUBOPTIMAL INTERCEPT PLAN

### *3.1. Fundamental Scheme of the Algorithm*

To search for a sub-optimal plan, we propose a Monte Carlo tree search algorithm. Its main idea is to construct a search tree through Monte Carlo simulations. A search tree, denoted as $S = (W, E)$, consists of sets of vertices $W$ and edges $E$. Vertices $w \in W$ are labeled with admissible interception plans $\pi_w \in \Pi_A$. A vertex $w_2$ is connected to a vertex $w_1$ by an edge if the plan $\pi_{w_2}$ extends the plan $\pi_{w_1}$ by exactly one target:

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

For convenience, we call the vertex $w_0 \in W$ labeled with an empty plan [ ] as the root of the tree, and the children of a vertex $w \in W$ are those $w' \in W$ for which the plan $\pi_{w'}$ extends $\pi_w$. Note that for any $w \in W$, there exists a unique path connecting $w$ to the root. Thus, $w \in W$ always lies on the path from the root to any of its descendants.

The basic cycle of the algorithm consists of three steps:

1. Descending along the tree from its root $w_0 \in W$ to a vertex $w \in W$ that has not yet been visited by the algorithm.
2. Generating a random extension of the interception plan $\pi_w$ corresponding to the given vertex $w$.
3. Evaluating the generated extension using the quality function (14) and back-propagating the result up to the root of the tree.

Furthermore, let $p_w$ denote the number of passes through vertex $w \in W$ during tree descents, and $q_w$ be the average estimate of random plans constructed for vertex $w \in W$ and its descendants. The value of $p_w$ increases by one with every pass through $w \in W$ on a descent. Additionally, $q_w$ is recalculated during the back-propagation of a random estimate from one of $w$'s descendants back to the tree root. Therefore, the number of random estimates averaged to compute $q_w$ exactly equals $p_w$. For example, for the root of the tree, $q_{w_0}$ coincides with the average estimate of all constructed random plans. Thus, each iteration of the basic cycle leads to:

1. Examination of one new vertex $w_1 \in W$ of the search tree, for which $p_{w_1} = 0$ up to the moment.

2. Generation of a random extension of the interception plan $\pi_{w_1}$ corresponding to the given vertex $w_1$.
3. Recalculation of the quantities $p_w$ and $q_w$ for all vertices $w \in W$ of the search tree located on the path from the root $w_0$ to $w_1$.

The basic cycle is repeated until the computational limit allocated for solving the problem is exhausted. Then, based on the accumulated statistics on $p_w$ and $q_w$, the final interception plan is constructed. Therefore, to fully describe the algorithm, it is necessary to specify the following features:

1. The method of tree descent and the heuristics used to construct it.
2. The approach for generating a random extension of the plan $\pi_w$ for an arbitrary vertex $w \in W$ of the search tree.
3. The mechanism for extracting the resulting plan from the search tree based on the numbers $p_w$ and $q_w$.

These will be addressed in the following subsections.

### *3.2. Descent Method on a Search Tree Based on Utility Function*

The descent along the search tree starts from the root $w_0 \in W$ and proceeds as follows.

1. If the current node $w$ has not been visited before, i.e., if $p_w = 0$, then the descent stops at $w$, and the algorithm moves on to construct a random extension of the plan $\pi_w$.
2. Otherwise, it is necessary to consider all adjacent nodes $w_i$, i.e., those $w_i \in W$ such that $\exists(w, w_i) \in E$, and move to one of them which maximizes the utility function

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

Here $\theta(w_i)$ and $P(w, w_i)$ are two heuristics that determine the direction of descent together with $p_w$ and $q_w$. The function $\theta(w)$ evaluates the expected best continuation of the plan $\pi_w$ in the range $[-1; 1]$, and $P(w, w_i)$ is the probability that this continuation is achieved in the direction $w_i$. For an arbitrary vertex $w \in W$,

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

where the sum is taken over all $w_i$ adjacent to $w$. Therefore, in equation (17), the term $\frac{\sqrt{2p_w}}{1+p_{w_i}}$, with which $\theta(w_i)$ and $P(w, w_i)$ enter $u(w_i, w)$, decreases as $p_{w_i}$ increases. Thus, when descending to a child $w_i$ with a large $p_{w_i}$, the high evaluation of $q_{w_i}$ plays the main role. In turn, when $p_{w_i}$ is small, the average $q_{w_i}$ should not play a role due to the small sample on which it is calculated. In this case, $\theta(w_i)$ and $P(w, w_i)$ in equation (17) gain more weight compared to the estimate $q_{w_i}$.

Instead of the (17), the following utility function can also be used:

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

where if $p_{w_i} = 0$ – an infinite utility is assigned to the vertex $w_i$. This type of utility function is called upper confidence bounds (UCB) and was first proposed in [14] for finding the optimal strategy in the multi-armed bandit problem. The application of this utility function to the problem of constructing a Monte Carlo tree search algorithm was discussed in [15]. However, for the task at hand, the utility function in the form of (17) turned out to be more effective, as it was used in [9] for playing Go, due to the presence of heuristic functions in it.

Now let's move on to the specific functions $\theta(w)$ and $P(w, w_i)$ that were used in the algorithm's construction. To do this, let's introduce the concept of refined estimate $\hat{J}q[\pi]$ for an arbitrary plan $\pi$,

which we will understand as the estimate $Jq[\pi]$ found based on the possible direct interception points of targets at the end of the plan $\pi$ (for targets included in the plan $\pi$, the points of their interception by the plan $\pi$ are taken into account in the calculation). Then the function $\theta(w)$ can be represented in the following form:

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

For vertices $w \in W$ with $p_w = 0$, the regular estimate $J_q[\pi_w]$ is used as $\theta(w)$, while for all others, the refined $\hat{J}_q[\pi_w]$ is used. This saves computational resources when finding the utility function (17) at vertices with small visit counts $p_w$. In turn, to find $P(w, w_i)$, a unit-normalized distribution is chosen:

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

where $\tau = \max\limits_i \tau_i$, $\tau = \min\limits_i \tau_i$, and $\tau_i$ is the duration of the direct interception of the $i$th target, which can be performed immediately after executing the plan $\pi_w$.

### *3.3. Construction of Random Extensions*

To extend the plan $\pi_w$ at vertex $w \in W$, the algorithm uses the probability distribution:

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

for targets not included in the plan $\pi_w$, where $d_i$ represents the hazards introduced according to (3). Then the plan $\pi^i$ is incrementally constructed by assigning one additional target according to this probability distribution until all available interception targets are exhausted.

Another method for extending plans $\pi_w$ is traversing unvisited targets in decreasing order of hazard. Although this extension method lacks a random element, due to the nature of this problem, it proves to be optimal for constructing the algorithm.

### *3.4. Construction of the Final Interception plan by the Monte Carlo Algorithm*

The final interception plan is constructed based on the resulting search tree after exhausting the computation limit (constraints on the number of iterations or execution time). For this, the algorithm additionally tracks the best estimate $J_q^*$ among all random estimates generated during repetitions of the basic cycle, and the interception plan $\pi q$ corresponding to this estimate. After completing the computations, the algorithm returns the found plan $\pi_q^*$ as the answer.

This approach allows for several improvements to the fundamental algorithm outlined in Section 3.1. Specifically, it becomes possible to check nodes $w \in W$ in the search tree for suboptimality. Thus, if for any $w \in W$ the estimate $J_q[\pi_w]$ of the plan $\pi_w$ turns out to be worse than the best achieved estimate $J_q$, it means that there is no optimal solution among the extensions of the plan $\pi_w$, and this search branch should be pruned. Therefore, during the descent along the tree, such nodes should be marked as suboptimal. Also, a node $w \in W$ should be marked as suboptimal if all its direct descendants are already marked as suboptimal. If during the descent process a vertex $w \in W$ is identified as suboptimal according to the described principles, the descent process moves up one vertex towards the root and then re-evaluates the descent direction, optimizing the utility function (17) over the truncated set of options. The initial value of the estimate $Jq$ is chosen as the estimate of the plan $\pi d$, obtained by ordering targets by hazards from the most hazardous to the least hazardous.

The current optimal estimate $J_q^*$ is also used to optimize the procedure for extending an arbitrary plan $\pi$. Specifically, this extension continues until the current estimate reached on this extension becomes worse than the optimal one. If this extension branch is found to be sub-optimal, the extensions are terminated, and the minimum possible estimate $-1$ is propagated up to the root of the search tree, significantly lowering the estimates of vertices for which the typical extension turned out to be sub-optimal.

Finally, if two plans $\pi_1$ and $\pi_2$ differ only in the order of targets and coincide on the last target, then the worse of the two is inherently sub-optimal, and the corresponding vertex should be removed from the search tree.

Thus, the measures described above allow for pruning inherently sub-optimal branches of the search tree, significantly enhancing the efficiency of the algorithm, as will be demonstrated with specific examples in the following section.

## 4. ALGORITHM SIMULATION

Let's compare the algorithm with full-depth depth-first search with branch-and-bound pruning. To simulate the algorithm's operation, we considered initial positions with randomly placed targets within the framework of the problem statement from the second section.

For a problem with 10 attacking targets, both algorithms found the optimal plan, as shown in Figs. 1 and 2. We also provide a summary in Table 1 regarding the quality of the constructed plans

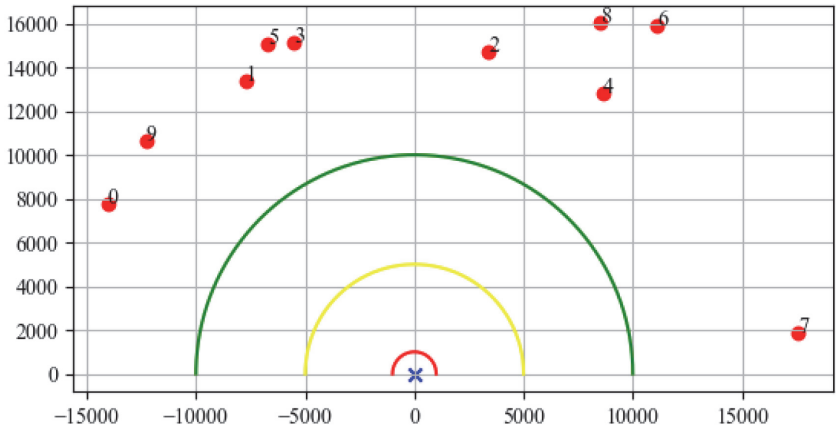


**Fig. 1.** Initial position of targets and interceptor for the problem with 10 targets.

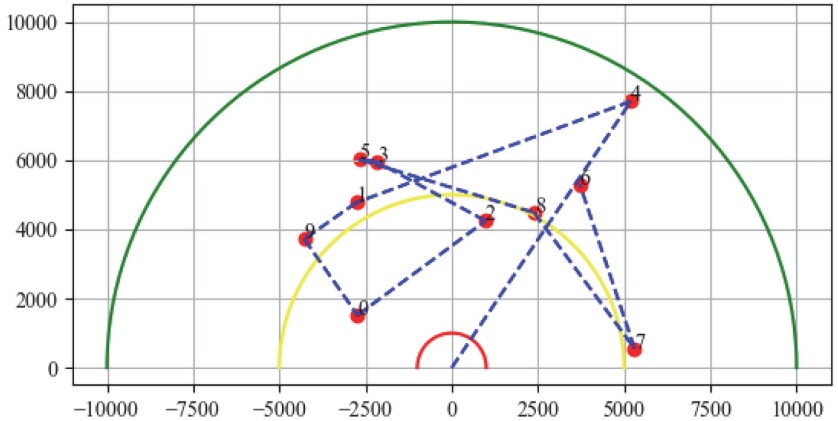


**Fig. 2.** Optimal interception plan constructed by exhaustive search and Monte Carlo algorithms.

**Table 1.** Summary of algorithm performance for 10 targets

| Algorithm type | Quality estimate | Work time |
|---|---|---|
| Exhaustive search | (0, 2, 8, 0) | 0,05 s |
| Monte Carlo | (0, 2, 8, 0) | 1 s |

and the runtime. As seen, with a small number of targets, the exhaustive algorithm is significantly faster. This is because the runtime of the tree search algorithm was set to one second regardless of the problem size.

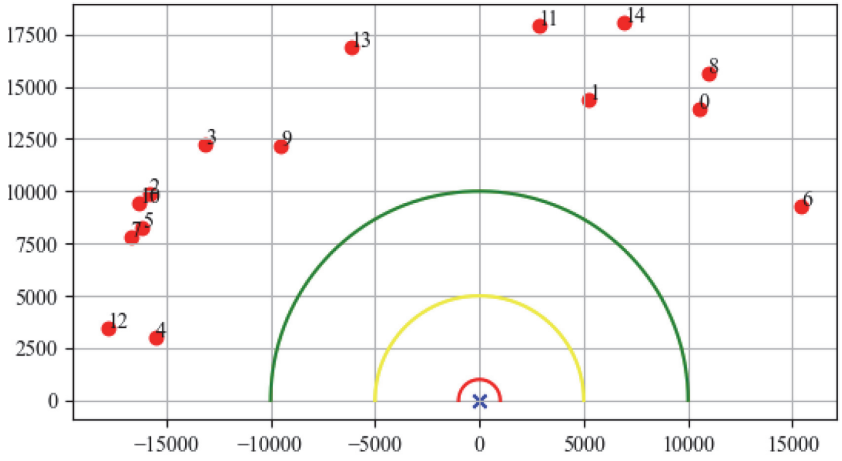


**Fig. 3.** Initial position of targets and interceptor for the problem with 15 targets.

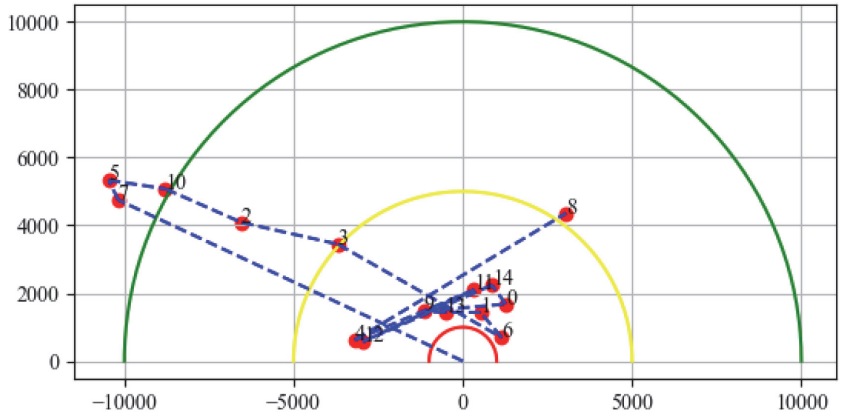


**Fig. 4.** Optimal interception plan constructed by exhaustive search and tree search for 15 targets.

Now let's consider an example with 15 targets, whose initial positions are shown in Fig. 3. In this case, both algorithms also succeeded in finding the optimal plan, as shown in Fig. 4. However, now the exhaustive algorithm outperformed the Monte Carlo algorithm by only half a second of computational time, as indicated in Table 2.

**Table 2.** Summary of algorithm performance for 15 targets

| Algorithm type | Quality estimate | Work time |
|---|---|---|
| Exhaustive search | (0, 9, 3, 3) | 0,48 s |
| MonteCarlo | (0, 9, 3, 3) | 1 s |

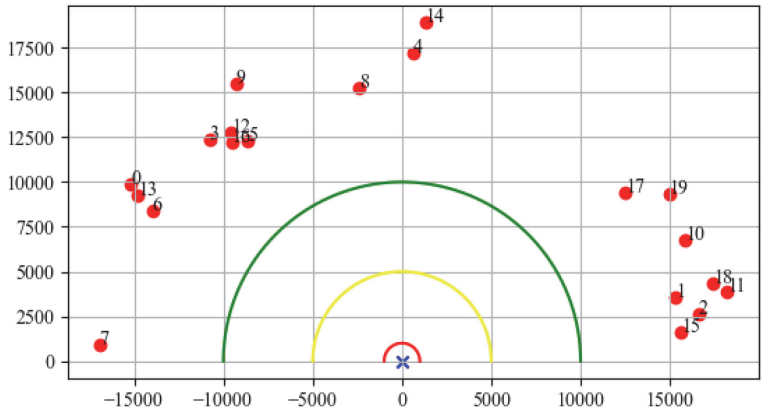

**Fig. 5.** Initial position of 20 targets.

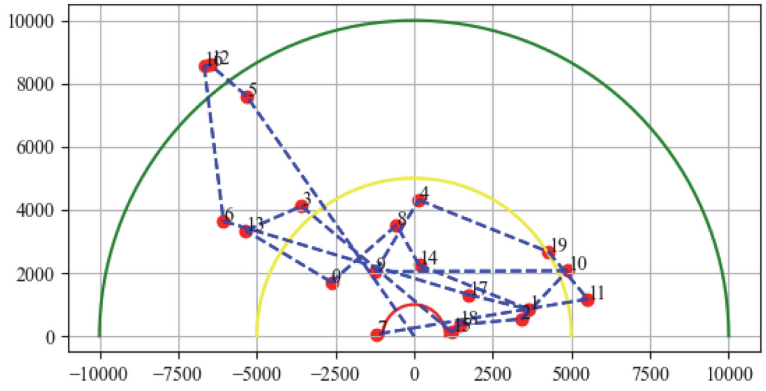

**Fig. 6.** Optimal interception plan constructed by exhaustive search for 20 targets.

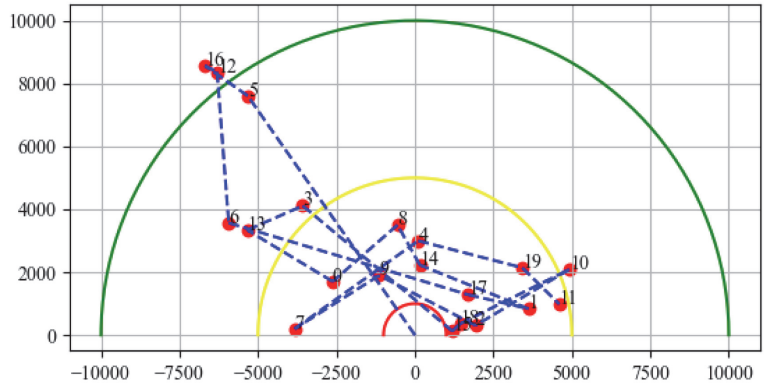

**Fig. 7.** Interception plan constructed by the Monte Carlo algorithm for 20 targets.

For a problem with 20 targets, depicted in Fig. 5, the search algorithm constructed an interception plan without skipping targets in the near zone, which, however, was sub-optimal. A comparison of the two plans is provided in Figs. 6 and 7, and in Table 3, it's noted that at this data size, the tree search was four times faster than the exhaustive search.

**Table 3.** Summary of algorithm performance for 20 targets

| Algorithm type | Quality estimate | Work time |
|---|---|---|
| Exhaustive search | (0, 11, 7, 2) | 4,07 s |
| Monte Carlo | (0, 13, 5, 2) | 1 s |

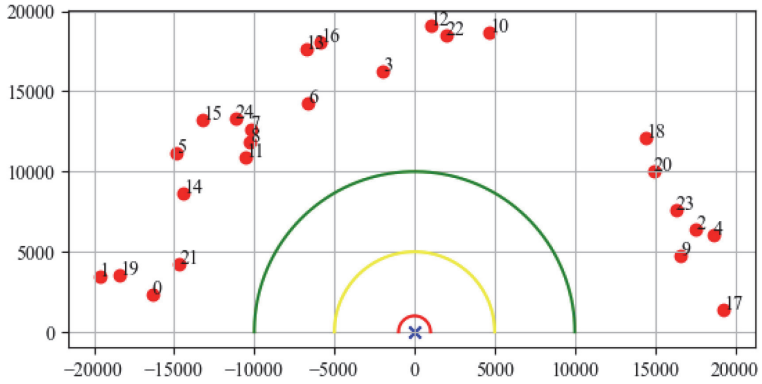

**Fig. 8.** Initial position of 25 targets.

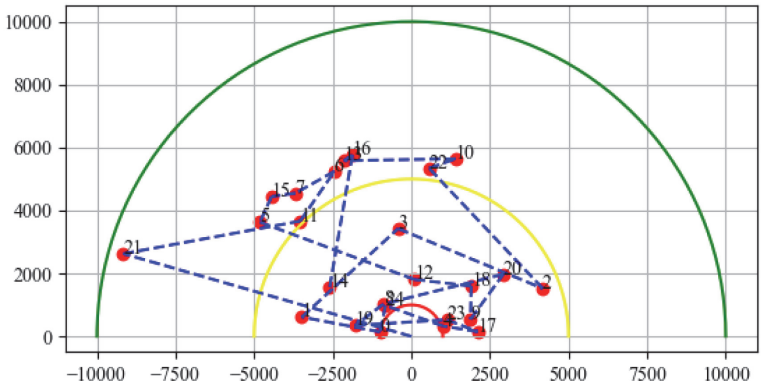

**Fig. 9.** Optimal interception plan constructed by exhaustive search for 25 targets.

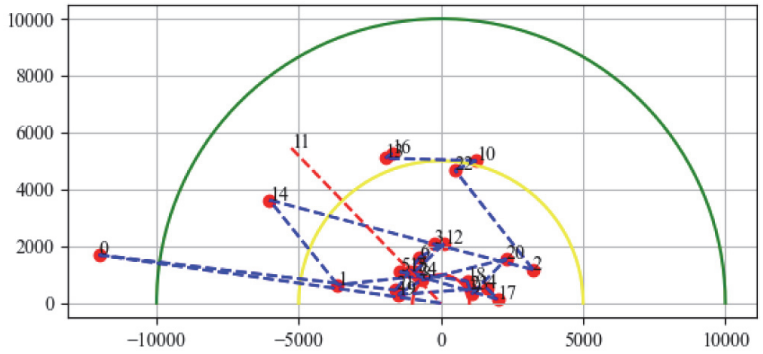

**Fig. 10.** Interception plan constructed by the Monte Carlo algorithm for 25 targets. The red line shows the trajectory of the missed target.

For the problem with 25 targets, the tree search was able to construct an interception plan with the skipping of only one target. However, its runtime was much smaller compared to the exhaustive search algorithm. The initial positions of targets for this problem are shown in Fig. 8, and the constructed interception plans are provided in Figs. 9 and 10. The runtime of the algorithms and the estimates of the obtained plans are listed in Table 4.

**Table 4.** Summary of algorithm performance for 25 targets

| Algorithm type | Quality estimate | Work time |
|---|---|---|
| Exhaustive search | (0, 15, 10, 0) | 13,63 s |
| Monte Carlo | (1, 19, 4, 1) | 1 s |

## 5. CONCLUSION

In this work, an algorithm based on Monte Carlo tree search for constructing a suboptimal interception plan for defending a protected point against linearly moving targets has been proposed. This method allows for efficiently generating acceptable solutions even for cases with a large number of attacking targets, where exhaustive search algorithms may be too slow for practical applications. However, there is still the possibility of using both algorithms simultaneously and making a choice based on their results in real-time mode. The approach proposed in this work can also be directly generalized to the case of defending a protected point by multiple interceptors against a significantly larger number of attacking targets. Furthermore, the proposed solution method is not only suitable for the problem considered in this paper but also for any other discrete optimization problem. The discussed scheme can be optimized and improved in various ways for application to a wide range of problems. A comprehensive analysis of the latest versions and applications of the Monte Carlo tree search algorithm is provided in [16]. Thus, a wide range of possible applications of the scheme described in this work for constructing approximate solutions in other discrete optimization problems is opened up. Another possible direction for further research on this topic is the application of neural network approaches in conjunction with Monte Carlo tree search. Trained neural networks can be used to compute heuristics $\theta$ and $P$, which are included in the utility function (17). The use of neural networks in computing heuristics for the Monte Carlo tree search algorithm has already proven to be extremely effective in building programs for playing Go [9]. Therefore, adopting such an approach may be equally fruitful for solving the traveling salesman problem and other similar discrete optimization problems, which could be the subject of further research on this issue.

## FUNDING

The work was supported by the Russian Science Foundation, project no. 23-19-00134.

## REFERENCES

1. Galyaev, A.A., Yakhno, V.P., Berlin, L.M., Lysenko, P.V., and Buzikov, M.E., Optimization of the Interception Plan for Linearly Moving Targets, *Autom. Remote Control*, 2023, no. 10, pp. 18–36.
2. Sikharulidze, G.G., On a Generalization of the Traveling Salesman Problem. I, *Autom. Remote Control*, 1971, no. 8. pp. 116–123.
3. Sikharulidze, G.G., On a Generalization of the Traveling Salesman Problem. II, *Autom. Remote Control*, 1971, no. 10. pp. 142–147.
4. Picard, J.C. and Queyranne, M., The Time-Dependent Traveling Salesman Problem and Its Application to the Tardiness Problem in One-Machine Scheduling, *Oper. Res.*, 1978, vol. 26, no. 1. pp. 86–110. https://doi.org/10.1287/opre.26.1.86
5. Helvig, C.S., Robins, G., and Zelikovsky, A., The Moving-Target Traveling Salesman Problem, *J. Algorithm. Comput. Technol.*, 2003, vol. 49, no. 1, pp. 153–174. https://doi.org/10.1016/S0196-6774(03)00075-0
6. Garey, M.R. and 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., and Kang, L., A New Approach to Solving Dynamic Traveling Salesman Problems, in *Simulated Evolution and Learning, Lecture Notes Comput. Sci.*, Wang, T.-D. et al., Eds., 2006, vol. 4247, Berlin, Heidelber: Springer.
8. Archetti, C., Feillet, D., Mor, A., and Speranza, M.G., Dynamic Traveling Salesman Problem with Stochastic Release Dates, *Eur. J. Oper.*, 2020, vol. 280, no. 3, pp. 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, vol. 529, pp. 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., and Uiterwijk, J.W.H.M., Single-Player Monte-Carlo Tree Search, in *Computers and Games, CG 2008, Lecture Notes in Computer Science*, vol. 5131, Berlin, Heidelberg: Springer.

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, vol. 192, pp. 2–3.

12. Cotarelo, A., Vicente, G., Edward Rolando, N., Cristian, G., Alberto, G., and Jerry, Ch., Improving Monte Carlo Tree Search with Artificial Neural Networks without Heuristics, *Appl. Sci.*, 2021, vol. 11, no. 5, pp. 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., and Fischer, P., Finite-time Analysis of the Multiarmed Bandit Problem, *Machine Learning*, 2002, vol. 47, pp. 235–256. https://doi.org/10.1023/A:1013689704352

15. Kocsis, L. and Szepesvári, C., Bandit Based Monte-Carlo Planning, in: *Machine Learning: ECML 2006, Lecture Notes Comput. Sci.*, vol. 4212, Fürnkranz, J., Scheffer, T., Spiliopoulou, M., Eds., Berlin, Heidelberg: Springer.

16. Świechowski, M., Godlewski, K., Sawicki, B., et al., Monte Carlo Tree Search: A Review of Recent Modifications and Applications, *Artif. Intell. Rev.*, 2023, vol. 56, pp. 2497–2562. https://doi.org/10.1007/s10462-022-10228-y

*This paper was recommended for publication by A.A. Lazarev, a member of the Editorial Board*