# Application of the RRT* and RRT-connect algorithms for finding the optimal trajectory in the sense of construction cost

Kira Dmitrieva[1] and Majid Abbasov[2][0000-0003-1484-4733]

[1] St. Petersburg State University, SPbSU, 7/9
Universitetskaya nab., St. Petersburg, 199034 Russia

[2] St. Petersburg State University, SPbSU, 7/9
Universitetskaya nab., St. Petersburg, 199034 Russia
m.abbasov@spbu.ru

**Abstract.** The essence of the research work is to consider the practical solution of the variational problem using iterative approach methods such as RRT* and RRT-connect. Originally, these algorithms were proposed to find the shortest trajectory connecting two given points on the terrain. In contrast with that the modifications presented in the present work allow us to search the trajectory, which is optimal in the sense of construction cost. This cost is defined as integral functional. The algorithms work based on randomly constructed graph-trees to construct a piecewise linear approximation of the solution. Though a specific problem is considered, the resulting solution approach opens access to solving a large class of problems. By applying these algorithms in infrastructure development, civil engineering, and related fields, engineers and planners will be able to achieve a more streamlined process for creating road networks that minimize construction costs and address the unique challenges posed by different terrains.

**Keywords:** optimal path, graph tree, fast growing graph trees.

## 1 Introduction

The problem of finding the optimal in terms of construction costs path of a road connecting two given points, which naturally arises before various private organizations, government agencies and military structures, is a subject of study for many researchers. One of the most popular approaches to solving this problem is the Cost Path Analysis; it is based on the construction and analysis of a cost lattice. This paper considers the variational problem of obtaining a road trajectory that is optimal in terms of construction cost, using other computational tools. To get the solution, a modified methods of rapidly-exploring random trees RRT* and RRT-connect are applied. In contrast to the original algorithms used for finding the shortest path ([3], [4], [5]), and

other modifications ([6], [7]), this paper proposes their adaptation for obtaining the trajectory on which the minimum of the construction cost functional is achieved.

Let the points A and B be given on the terrain, acting as the start and end points, respectively. It is required to connect them by a road in such a way that the cost of its construction is minimized. Let us denote by α the cost of delivery of materials required to lay a unit length of roadbed per the unit length of the roadbed. We assume that the same amount of materials is required to build a unit of road, regardless of the terrain. Let us denote the cost of construction works per unit length of the roadbed as β: $R^2 \to R$, which is a continuous function that has continuous partial derivatives. As it was shown in [1], [2], the functional can be written in the form:

$$\mathrm{J(y)} = \frac{\alpha}{2}\left(\int_{\mathrm{x_A}}^{x_B} \sqrt{1 + y'^2(x)}\ \mathrm{dx}\right)^2 + \int_{\mathrm{x_A}}^{x_B} \beta(\mathrm{x,y}) \sqrt{1 + y'^2(x)}\ \mathrm{dx} \quad (1)$$

## 2 Main Part

### 2.1 RRT*

The algorithm works on the basis of randomly constructed graph-trees from the starting point until one node reaches the target point. We demonstrate how the RRT* algorithm can be used for solving our problem of calculus of variations. Using this approach, we obtain piecewise linear approximation of the solution. As it was mentioned in [5], the algorithm has time complexity $O(nlog(n))$ for processing, and $O(n)$ for query.

**General conditions for graph construction.** Let ρ be a given distance in the state space $U$, where $U$ is bounded, and a tree $T \subset U$ (a tree is the union of a set of vertices and a set of edges). Let $x_0$ denote the initial state, the first node of our model; $x_0 \in U$, $x_0 \in T$, and $x_*$ denote the final state to be reached; $x_* \in U$, $x_* \in T$. Let Y represent the set of obstacles to be circumvented. The only thing we can say about the obstacles is whether the chosen state lies in the obstacle region or not. In this manner, $U \backslash Y$ is the free region in which the path will lie; $T \subset U \backslash Y$. The value $\delta = const$ is the step, and $\varepsilon \geq 0$ is the radius of the neighborhood of the endpoint, once in which the algorithm stops further work.

**Algorithm at the k-th step.** We denote by $X_k$ the set of vertices of the tree $T$ at step k, $X_k \subset T$.

1. We choose a random state $x_i$ from a free region; $x_i \in U \backslash Y$.

2. We find the nearest neighbor $\bar{x}_k$ from the set $X_k$ to the state $x_i$ in the sense of distance $\rho$.

3. We add a new node $x_{k+1}$:

$$x_{k+1} = \begin{cases} \bar{x}_k + (x_i - \bar{x}_k)\delta, \ if\ \rho(\bar{x}_k;\ \bar{x}_k + (x_i - \bar{x}_k)\delta) \le \rho(\bar{x}_k; x_i) \\ \bar{x}_k + x_i, \ if\ \rho(\bar{x}_k;\ \bar{x}_k + (x_i - \bar{x}_k)\delta) > \rho(\bar{x}_k; x_i) \end{cases} (2)$$

If the segment $[\bar{x}_k; x_{k+1}]$ intersects the set of obstacles $Y$, we assume that $X_{k+1} = X_k$, without adding new edges to $T$, and proceed to the (k+1)-th iteration. If the segment does not intersect the set of obstacles, then $X_{k+1} = X_k \cup \{x_{k+1}\}$.

Let $\hat{X}_k$- the set of points of the $X_k$ that are in the r-neighborhood of the point $x_{k+1}$:

$$\hat{X}_k = X_k \cap B_r(x_{k+1}) \ , r > \boldsymbol{\delta} (3)$$

In the set $\hat{X}_k \backslash x_{k+1}$, we look for a vertex $\hat{x}_{min;\,k}$ of the tree $T$, the edge from which to $x_{k+1}$ is minimal in the sense of the functional (1):

$$\mathrm{J(y)} = \frac{\alpha}{2}\left(\int_{x_A}^{x_B} \sqrt{1 + y'^2(x)}\ \mathrm{dx}\right)^2 + \int_{x_A}^{x_B} \beta(\mathrm{x,y}) \sqrt{1 + y'^2(x)}\ \mathrm{dx}$$

Since the edges of the tree are segments, we can simplify the functional. Let node $x_{k+1}$ have coordinates $(X_1; Y_1)$, and $\hat{x}_{min;\,k}$ - $(X_2; Y_2)$. Then the equation of the tree edge can be represented as:

$$y = \frac{(x - X_1)(Y_2 - Y_1)}{X_2 - X_1} + Y_1 (4)$$

Derivative $y'(x)$ is constant, so the functional J(y) can be rewritten as:

$$\mathrm{J(y)} = \frac{\alpha}{2}((\mathrm{X}_2 - \mathrm{X}_1)^2 + (Y_2 - Y_1)^2) + \int_{X_1}^{\mathrm{X}_2} \beta\left(\mathrm{x}, \frac{(x - X_1)(Y_2 - Y_1)}{X_2 - X_1} + Y_1\right) \sqrt{1 + \left(\frac{Y_2 - Y_1}{X_2 - X_1}\right)^2}\ \mathrm{dx} (5)$$

If there are several such vertices, we take any of them as $\hat{x}_{min;\,k}$. Obtained in this way, the edge is added to the tree.

In the same set of vertices $\hat{X}_k \backslash x_{k+1}$, we check the paths from the starting point $x_0$ to $\hat{x}_k$, $\forall \hat{x}_k \in \hat{X}_k$. Let us denote the path from $x_0$ to $\hat{x}_k$ by $P(x_0; \hat{x}_k)$. Such a path can be uniquely defined, since it is drawn along the edges of a tree in which each vertex has a single 'parent' vertex. If the path constructed earlier from the starting point is more expensive, in the sense of functional J(y), than the path$[\hat{x}_k; x_{k+1}] \cup [x_{k+1}; \hat{x}_{min;\,k}] \cup P(x_0; \hat{x}_{min;\,k})$, then we remove the edge connecting vertex $\hat{x}_k$ and its 'parent' from the tree, and add the edge $[\hat{x}_k; x_{k+1}]$ to $T$. Such manipulations do not destroy the graph-tree structure and it optimizes the construction.

4. If $\rho(x_{k+1}, x_*) \leq \varepsilon$, we stop further computations and consider that the path is constructed. If not, we proceed to the (k+1)-th iteration.

This modification of the RRT* algorithm allows us to find a path that is optimal in the sense of the cost functional (1):

$$J(y) = \frac{\alpha}{2}\left(\int_{x_A}^{x_B} \sqrt{1 + y'^2(x)}\ dx\right)^2 + \int_{x_A}^{x_B} \beta(x,y)\sqrt{1 + y'^2(x)}\ dx$$

**Algorithm code description.**

```
Algorithm RRT*:
V ← {x_init}; E ← ∅;
for i = 1, . . . , n do
  x_rand ← Random_i;
  x_nearest ← Nearest(G = (V, E), x_rand);
  x_new ← Step(x_nearest, x_rand);
  if ObtacleFree(x_nearest, x_new) then
    X_near ← Neighborhood(G = (V, E), x_new,ε);
    V ← V ∪ {x_new};
    X_min ← x_nearest;
    c_min ← CostPath(x_nearest) + Functional(x_nearest,
x_new);
    foreach x_near ∈ X_near do
      if ObstacleFree(x_near, x_new)then
        if CostPath(x_near) + Functional(x_near, x_new) <
c_min then
          x_min ← x_near;
          c_min ← CostPath(x_near) + Functional(x_near,
x_new)
    E ← E ∪ {(x_min, x_new)};
    foreach x_near ∈ X_near do
      if ObstacleFree(x_new, x_near) then
        if CostPath(x_new) + Functional(x_new, x_near) <
CostPath(x_near) then
          x_parent ← Parent(x_near);
      E ← (E \ {(x_parent, x_near)}) ∪ {(x_new, x_near)}
return G = (V, E);
```

**Example of path construction**. Consider a problem with parameters:

$$\begin{gathered} \alpha = 0.1 \\ x_0 = y_0 = 0 \\ x_* = y_* = 1 \\ \beta(x,y) = 1 + sin\,5x - sin\,y \end{gathered} \qquad (6)$$

For convenience of interpretation, we can consider that the function β(x, y) defines the equation of the terrain surface, i.e. the cost of construction works is greater the higher the point lies above the $Oxy$ plane.

The following are graphical representations of the operation of the algorithm.

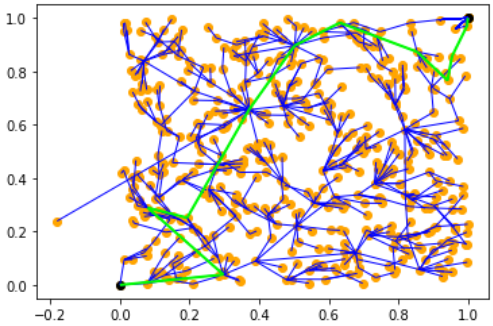

**Fig. 1.** Example of path construction using a modification of RRT* for 500 nodes; 2-dimentional representation

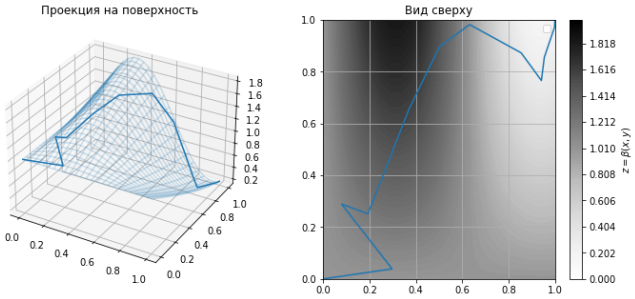


**Fig. 2.** Example of path construction using a modification of RRT* for 500 nodes; 3-dimentional representation

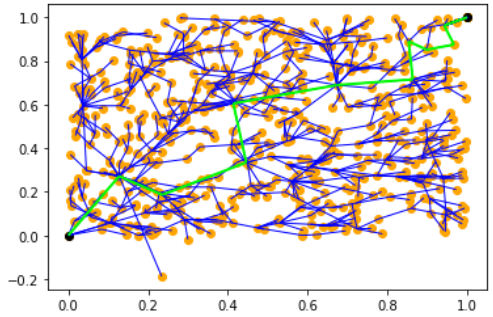

**Fig. 3.** Example of path construction using a modification of RRT* for 600 nodes; 2-dimentional representation

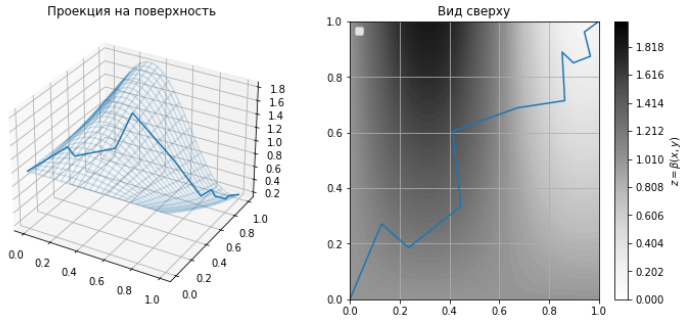


**Fig. 4.** Example of path construction using a modification of RRT* for 600 nodes; 3-dimentional representation

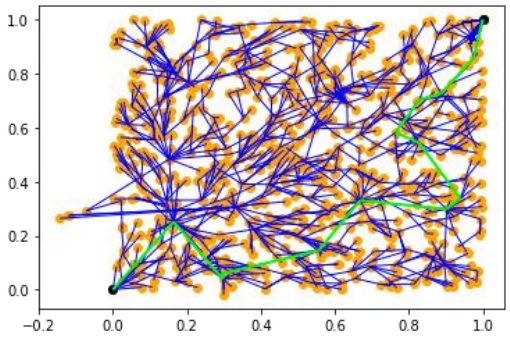

**Fig. 5.** Example of path construction using a modification of RRT* for 900 nodes; 2-dimentional representation

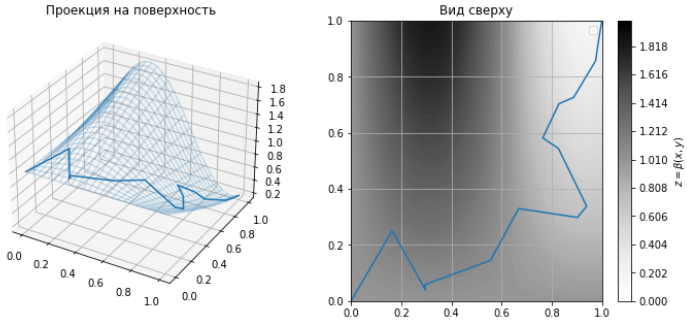


**Fig. 6.** Example of path construction using a modification of RRT* for 600 nodes; 3-dimentional representation

## 2.2 RRT-connect

The algorithm works on the basis of randomly constructed two graph-trees from the start and end points simultaneously until the trees are connected. Once trees are connected, it is claimed that the path has been built. The proposed algorithm differs from the original one in [4] or its modification [6] in the way graphs are connected. We propose the modification where the process of connection is organized in a way which provides optimality of the entire trajectory in the sense of functional (1). Using this approach, we obtain piecewise linear approximation of the solution.

**General conditions for graph construction.** Let $\rho$ be a given distance in the state space $U$, where $U$ is bounded, and trees $T_1, T_2 \subset U$ (a tree is the union of a set of vertices and a set of edges). We denote by $x_0$ the initial state, the node from which the first tree will grow; $x_0 \in U$, $x_0 \in T_1$, and by $x_*$ the final state from which the second tree will grow; $x_* \in U$, $x_* \in T_2$. Let $Y$ represent the set of obstacles to be bypassed. The only thing we can say about the obstacles is whether the chosen state lies in the obstacle region or not. Let $U \backslash Y$ be the free region in which the path will lie. The value $\delta = const$ is the step, and $\varepsilon \geq 0$ is the radius of the neighborhood of each node.

**Algorithm at the k-th step.** Let us denote by $X_{k;1}$ and $X_{k;2}$ the sets of tree nodes $T_1, T_2$ at step k respectively, $X_{k;1} \subset T_1$, $X_{k;2} \subset T_2$.

1. We choose a random state $x_i$ from a free region; $x_i \in U \backslash Y$.

2. We find the nearest neighbor $\bar{x}_{k;1}$ from the set $X_{k;1}$ and $\bar{x}_{k;2}$ from the set $X_{k;2}$ to state $x_i$ in the sense of distance $\rho$.

Consider the construction of a new node for the tree $T_1$; for $T_2$, all the construction proceeds in the same way, the optimization will be for paths from the endpoint $x_*$.

3. We add a new node $x_{k+1;1}$ to the $T_1$:

$$x_{k+1;1} = \begin{cases} \bar{x}_{k;1} + (x_i - \bar{x}_{k;1})\delta, & if\ \rho(\bar{x}_{k;1};\ \bar{x}_{k;1} + (x_i - \bar{x}_{k;1})\delta) \leq \rho(\bar{x}_{k;1}; x_i) \\ \bar{x}_{k;1} + x_i, & if\ \rho(\bar{x}_{k;1};\ \bar{x}_{k;1} + (x_i - \bar{x}_{k;1})\delta) > \rho(\bar{x}_{k;1}; x_i) \end{cases} \quad (7)$$

If the segment $[\bar{x}_{k;1}; x_{k+1;1}]$ intersects the set of obstacles $Y$, we assume that $X_{k+1;1} = X_{k;1}$, without adding new edges to $T_1$, we proceed to the (k+1)-th iteration. If the segment does not intersect the set of obstacles, then $X_{k+1;1} = X_{k;1} \cup \{x_{k+1;1}\}$.

Let $\hat{X}_{k;1}$ be the set of points of the $X_{k;1}$ that are located in the r-neighborhood of the point $x_{k+1;1}$:

$$\hat{X}_{k;1} = X_{k;1} \cap B_r(x_{k+1;1}) \text{ , } r > \boldsymbol{\delta} (8)$$

In the set $\hat{X}_{k;1} \backslash x_{k+1;1}$, we look for a vertex $\hat{x}_{min;\,k;1}$ of the tree $T_1$ whose edge from which to $x_{k+1;1}$ is minimal in the sense of the functional (1):

$$\mathrm{J(y)} = \frac{\alpha}{2}\left(\int_{x_A}^{x_B} \sqrt{1 + y'^2(x)}\ \mathrm{dx}\right)^2 + \int_{x_A}^{x_B} \beta(\mathrm{x,y}) \sqrt{1 + y'^2(x)}\ \mathrm{dx}$$

Since the edges of the tree are segments, we can simplify the functional. Let node $x_{k+1;1}$ have coordinates $(X_1; Y_1)$, and $\hat{x}_{min;\,k;1}$ - $(X_2; Y_2)$. Then the equation of the tree edge can be represented as:

$$y = \frac{(x - X_1)(Y_2 - Y_1)}{X_2 - X_1} + Y_1 (9)$$

Derivative $y'(x)$ is constant, so the functional J(y) can be rewritten as:

$$\mathrm{J(y)} = \frac{\alpha}{2}((\mathrm{X}_2 - \mathrm{X}_1)^2 + (Y_2 - Y_1)^2) + \int_{\mathrm{X}_1}^{\mathrm{X}_2} \beta\left(\mathrm{x}, \frac{(x - X_1)(Y_2 - Y_1)}{X_2 - X_1} + Y_1\right) \sqrt{1 + \left(\frac{Y_2 - Y_1}{X_2 - X_1}\right)^2}\ \mathrm{dx} (10)$$

If there are several such vertices, we take any of them as $\hat{x}_{min;\,k;1}$. Obtained in this way, we add the edge to the tree.

In the same set of vertices $\hat{X}_{k;1} \backslash x_{k+1;1}$ , we check the paths from the starting point $x_0$ to $\hat{x}_{k;1}$, $\forall \hat{x}_{k;1} \in \hat{X}_{k;1}$. Let us denote the path from $x_0$ to $\hat{x}_{k;1}$ by $P(x_0; \hat{x}_{k;1})$. Such a path can be unambiguously defined, since it traverses the edges of a tree in which each vertex has a single 'parent' vertex. If the path constructed earlier from the starting point is more expensive, in the sense of functional J(y), than the path $[\hat{x}_{k;1}; x_{k+1;1}] \cup [x_{k+1;1}; \hat{x}_{min;\,k;1}] \cup P(x_0; \hat{x}_{min;\,k;1})$, then we remove the edge connecting vertex $\hat{x}_{k;1}$ and its 'parent' from the tree $T_1$, and add the edge $[\hat{x}_{k;1}; x_{k+1;1}]$ to it. Such manipulations do not destroy the graph-tree structure and it optimizes the construction.

Simultaneous with the procedures for the tree from the initial node, we do all the same steps for the tree from the final state respectively. Thus, we add a new node $x_{k+1;2}$ to the $T_2$:

$$x_{k+1;2} = \begin{cases} \bar{x}_{k;2} + (x_i - \bar{x}_{k;2})\delta, & if\ \bar{x}_{k;2} + \rho(\bar{x}_{k;2};\ (x_i - \bar{x}_{k;2})\delta) \le \rho(\bar{x}_{k;2}; x_i) \\ \bar{x}_{k;2} + x_i, & if\ \bar{x}_{k;2} + \rho(\bar{x}_{k;2};\ (x_i - \bar{x}_{k;2})\delta) > \rho(\bar{x}_{k;2}; x_i) \end{cases} (11)$$

If the segment $[\bar{x}_{k;2}; x_{k+1;2}]$ intersects the set of obstacles $Y$, we assume that $X_{k+1;2} = X_{k;2}$, without adding new edges to $T_2$, we proceed to the (k+1)-th iteration. If the segment does not intersect the set of obstacles, then $X_{k+1;2} = X_{k;2} \cup \{x_{k+1;2}\}$.

Let $\hat{X}_{k;2}$ be the set of point of the $X_{k;2}$ that are located in the r-neighborhood of the point $x_{k+1;2}$:

$$\hat{X}_{k;2} = X_{k;2} \cap B_r(x_{k+1;2}) \, , r > \boldsymbol{\delta} (12)$$

In the set $\hat{X}_{k;2} \backslash x_{k+1;2}$, we look for a vertex $\hat{x}_{min;\, k;2}$ of the tree $T_2$ whose edge from which to $x_{k+1;2}$ is minimal in the sense of the functional (1):

$$\mathrm{J(y)} = \frac{\alpha}{2}\left(\int_{\mathrm{x_A}}^{x_B} \sqrt{1 + y'^2(x)}\ \mathrm{dx}\right)^2 + \int_{\mathrm{x_A}}^{x_B} \beta(\mathrm{x,y}) \sqrt{1 + y'^2(x)}\ \mathrm{dx}$$

Since the edges of the tree are segments, we can simplify the functional. Let node $x_{k+1;2}$ have coordinates $(X_1; Y_1)$, and $\hat{x}_{min;\, k;2}$ - $(X_2; Y_2)$. Then the equation of the tree edge can be represented as:

$$y = \frac{(x - X_1)(Y_2 - Y_1)}{X_2 - X_1} + Y_1 (13)$$

Derivative $y'(x)$ is constant, so the functional J(y) can be rewritten as:

$$\mathrm{J(y)} = \frac{\alpha}{2}((\mathrm{X_2 - X_1})^2 + (Y_2 - Y_1)^2) + \int_{\mathrm{X_1}}^{\mathrm{X_2}} \beta\left(\mathrm{x}, \frac{(x - X_1)(Y_2 - Y_1)}{X_2 - X_1} + Y_1\right) \sqrt{1 + \left(\frac{Y_2 - Y_1}{X_2 - X_1}\right)^2}\ \mathrm{dx} (14)$$

If there are several such vertices, we take any of them as $\hat{x}_{min;\, k;2}$. Obtained in this way, we add the edge to the tree.

In the same set of vertices $\hat{X}_{k;2} \backslash x_{k+1;2}$, we check the paths from the starting point $x_0$ to $\hat{x}_{k;2}$, $\forall \hat{x}_{k;2} \in \hat{X}_{k;2}$. Let us denote the path from $x_0$ to $\hat{x}_{k;2}$ by $P(x_0; \hat{x}_{k;2})$. Such a path can be unambiguously defined, since it traverses the edges of a tree in which each vertex has a single 'parent' vertex. If the path constructed earlier from the starting point is more expensive, in the sense of functional J(y), than the path $[\hat{x}_{k;2}; x_{k+1;2}] \cup [x_{k+1;2}; \hat{x}_{min;\, k;2}] \cup P(x_0; \hat{x}_{min;\, k;2})$, then we remove the edge connecting vertex $\hat{x}_{k;1}$ and its 'parent' from the tree $T_2$, and add the edge $[\hat{x}_{k;2}; x_{k+1;2}]$ to it. Such manipulations do not destroy the graph-tree structure and it optimizes the construction.

4. If $\rho(x_{T_1}, x_{T_2}) \leq \varepsilon$, i.e. ε-neighborhoods of nodes from trees $T_1, T_2$ intersect, we connect the nodes by an edge $[x_{T_1}; x_{T_2}]$, consider that the optimal path is constructed

and stop further computations. If there are several such intersections, then the construction of an edge connecting two trees will be done, as described below.

Let there be m intersections of the neighborhoods of the nodes of two trees:

$$\begin{cases} \rho\left(x_{1;T_1}, x_{1;T_2}\right) \le \varepsilon \\ \rho\left(x_{2;T_1}, x_{2;T_2}\right) \le \varepsilon \\ \dots \\ \rho\left(x_{m-1;T_1}, x_{m-1;T_2}\right) \le \varepsilon \\ \rho\left(x_{m;T_1}, x_{m;T_2}\right) \le \varepsilon \end{cases} \quad (9)$$

The paths $P(x_0; x_{1;T_1})$,…, $P(x_0; x_{m;T_1})$ and $P(x_*; x_{1;T_2})$,…, $P(x_*; x_{m;T_2})$ are uniquely defined due to the graph-tree structure. The edge that will connect trees $T_1$ and $T_2$ is the edge that provides the minimum path, in the sense of a functional J(y), of the following possible paths:

$$\begin{gathered} P\left(x_0; x_{1;T_1}\right) \cup [x_{1;T_1}; x_{1;T_2}] \cup P(x_*; x_{1;T_2}) \\ \dots \\ P\left(x_0; x_{1;T_1}\right) \cup [x_{1;T_1}; x_{m;T_2}] \cup P(x_*; x_{m;T_2}) \\ P\left(x_0; x_{2;T_1}\right) \cup [x_{2;T_1}; x_{1;T_2}] \cup P(x_*; x_{1;T_2}) \\ \dots \\ P\left(x_0; x_{2;T_1}\right) \cup \left[x_{2;T_1}; x_{m;T_2}\right] \cup P\left(x_*; x_{m;T_2}\right) \\ \dots \\ P\left(x_0; x_{m;T_1}\right) \cup \left[x_{m;T_1}; x_{1;T_2}\right] \cup P\left(x_*; x_{1;T_2}\right) \\ \dots \\ P\left(x_0; x_{m;T_1}\right) \cup [x_{m;T_1}; x_{m;T_2}] \cup P(x_*; x_{m;T_2}) \end{gathered} \quad (15)$$

If there are no nodes in trees $T_1, T_2$ whose ε-neighborhoods intersect, then we proceed to the (k+1)-th iteration.

**Algorithm code description.**

```
Algorithm RRT-connect:
V_0 ← {x_0}; E_0 ← ∅;
V_* ← {x_*}; E_* ← ∅;
for i = 1, . . . , n do
  x_rand ← Random_i;
  x_nearest_0 ← Nearest(G = (V_0, E_0), x_rand);
  x_nearest_* ← Nearest(G = (V_*, E_*), x_rand);
  x_new_0 ← Step(x_nearest_0, x_rand);
  x_new_* ← Step(x_nearest_*, x_rand);

  if ObtacleFree(x_nearest_0, x_new_0) && Obta-
cleFree(x_nearest_*, x_new_*) then
```

```
    X_near_0 ← Neighborhood(G = (V_0, E_0), x_new_0,ε);
    X_near_* ← Neighborhood(G = (V_*, E_*), x_new_*,ε);
    V_0 ← V_0 ∪ {x_new_0};
    V_* ← V_* ∪ {x_new_*};
    X_min_0 ← x_nearest_0;
    X_min_* ← x_nearest_*;
    c_min_0 ← CostPath(x_nearest_0) + Function-
al(x_nearest_0, x_new_0);
    c_min_* ← CostPath(x_nearest_*) + Function-
al(x_nearest_*, x_new_*);

    foreach x_near_0 ∈ X_near_0 do
      if ObstacleFree(x_near_0, x_new_0)then
        if CostPath(x_near_0) + Functional(x_near_0,
x_new_0) < c_min_0 then
          x_min_0 ← x_near_0;
          c_min_0 ← CostPath(x_near_0) + Function-
al(x_near_0, x_new_0)
    E_0 ← E_0 ∪ {(x_min_0, x_new_0)};
    foreach x_near_* ∈ X_near_* do
      if ObstacleFree(x_near_*, x_new_*)then
        if CostPath(x_near_*) + Functional(x_near_*,
x_new_*) < c_min_* then
          x_min_* ← x_near_*;
          c_min_* ← CostPath(x_near_*) + Function-
al(x_near_*, x_new_*)
    E_* ← E_* ∪ {(x_min_*, x_new_*)};

    foreach x_near_0 ∈ X_near_0 do
      if ObstacleFree(x_new_0, x_near_0) then
        if CostPath(x_new_0) + Functional(x_new_0,
x_near_0) < CostPath(x_near_0) then
          x_parent_0 ← Parent(x_near_0);
      E_0 ← (E_0 \ {(x_parent_0, x_near_0)}) ∪ {(x_new_0,
x_near_0)}
    foreach x_near_* ∈ X_near_* do
      if ObstacleFree(x_new_*, x_near_*) then
        if CostPath(x_new_*) + Functional(x_new_*,
x_near_*) < CostPath(x_near_*) then
          x_parent_* ← Parent(x_near_*);
      E_* ← (E_* \ {(x_parent_*, x_near_*)}) ∪ {(x_new_*,
x_near_*)}

    if Neighborhood(G = (V_0, E_0), x_new_*,ε) not ∅then
      X_last ← Nearest(G = (V_0, E_0), x_new_*)
```

```
      V_0 ← V_0 ∪ {x_new_*};
      x_min_last ← x_last;
      c_min_last ← CostPath(x_last) + Functional(x_last,
x_new_*) + CostPath(x_new_*);
      foreach x_last ∈ Neighborhood(G = (V_0, E_0),
x_new_*,ε)do
        if ObstacleFree(x_last, x_new_*)then
          if CostPath(x_last) + Functional(x_last,
x_new_*) + CostPath(x_new_*) < c_min_last then
            x_min_last ← x_last;
            c_min_last ← CostPath(x_last) + Function-
al(x_last, x_new_0) + CostPath(x_new_*)
      E_0 ← E_0 ∪ {(x_min_last, x_new_*)} ∪ Path(x_new_*,
x_*);
      G ← (V_0, E_0)
      break
    if Neighborhood(G = (V_*, E_*), x_new_0,ε) not ∅ then
      x_last ← Nearest(G = (V_*, E_*), x_new_0)
      V_* ← V_* ∪ {x_new_0};
      x_min_last ← x_last;
      c_min_last ← CostPath(x_last) + Functional(x_last,
x_new_0) + CostPath(x_new_0);
      foreach x_last ∈ Neighborhood(G = (V_*, E_*),
x_new_0,ε)do
        if ObstacleFree(x_last, x_new_0)then
          if CostPath(x_last) + Functional(x_last,
x_new_0) + CostPath(x_new_0) < c_min_last then
            x_min_last ← x_last;
            c_min_last ← CostPath(x_last) + Function-
al(x_last, x_new_0) + CostPath(x_new_0)
      E_* ← E_* ∪ {(x_min_last, x_new_0)} ∪ Path(x_new_0,
x_0);
      G ← (V_*, E_*)
      break

return G;
```

## 3 Conclusion

This paper describes the RRT* and RRT-connect used for seeking the most efficient road trajectory in terms of construction costs while considering the constraints posed by the terrain. These algorithms offer a systematic approach to optimizing the road layout given the various constraints imposed by the topography and other environmental factors. Via the RRT* and RRT-connect algorithms, engineers and planners

can achieve a more streamlined process for developing road networks that minimize construction costs and adhere to the unique challenges presented by diverse terrains. This paper serves as a guide to understanding and implementing these algorithms in infrastructure development, in civil engineering and related fields.

## 4 Acknowledgments

This research was supported by the Russian Science Foundation (RSF), project No 23-21-00027, https://rscf.ru/project/23-21-00027/.

## References

1. Abbasov, M. E., & Sharlay, A. S. Searching for the cost-optimal road trajectory on the relief of the terrain. Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes *17*(1), 4–12. (2021).
2. M. E. Abbasov, A. S. Sharlay, “Variational approach to finding the cost-optimal trajectory”, Matem. Mod., 35:12 (2023), 89–100
3. S. M. LaValle. Rapidly-exploring random trees:A new tool for path planning.TR 98-11,Computer Science Dept., Iowa State University.http://janowiec.cs.iastate.edu/papers/rrt.ps, Oct.1998.
4. J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, San Francisco, CA, USA, 2000, pp. 995-1001 vol.2, doi: 10.1109/ROBOT.2000.844730.
5. S. Karaman and E. Frazzoli. Sampling-based Algorithms for Optimal Motion Planning. International Journal of Robotics Research, 30(7):846-894, June 2011.
6. Klemm, Sebastian & Oberländer, Jan & Hermann, Andreas & Roennau, Arne & Schamm, Thomas & Zöllner, J. & Dillmann, Rüdiger. (2015). RRT*-Connect: Faster, Asymptotically Optimal Motion Planning. 10.1109/ROBIO.2015.7419012.
7. Kang, J.-G.; Lim, D.-W.; Choi, Y.-S.; Jang, W.-J.; Jung, J.-W. Improved RRT-Connect Algorithm Based on Triangular Inequality for Robot Path Planning. Sensors 2021, 21, 333.