# Rapidly Replanning A*

Nuwan Ganganath†, Chi-Tsun Cheng, and Chi K. Tse
Department of Electronic and Information Engineering
The Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong
†Email: nuwan@ganganath.lk

***Abstract*—In this paper, Rapidly Replanning A* (RRA*) algorithm is proposed for path planning and replanning in partially unknown environments. RRA* uses an effective mechanism to reuse previous search results, which considerably accelerates its replanning process compared to repetitive replanning from scratch. RRA* guarantees to find an optimal path from the current location of an agent to its target location based on the available information. Simulation results verify the optimality of the path generated by RRA* and the superior efficiency of RRA* in path replanning.**

***Index Terms*—A*, RRA*, path planning, replanning, incremental search, heuristic search.**

## I. Introduction

Mobile robots are often utilized in partially known environments. Paths generated in such environments can be suboptimal or even impractical due to incomplete information. Therefore, initially planned paths might need to be modified on the fly to obtain feasible optimal paths. Since repetitive path replanning from the scratch can be a tedious task in many situations, incremental search algorithms have been introduced for efficient path replanning as extensions to well-known A* search algorithm [1].

Stentz introduced Dynamic A* (D*) [2] algorithm which can be used for path planning with partial or no information. It allows an agent to learn the surrounding while navigating and modify the map if any changes in the surrounding are detected. The planning is carried out using the updated map information. Later, Focused D* search algorithm [3] was proposed by combining properties of incremental and heuristic searches. It improves the efficiency of both initial planning and subsequent replanning steps. Koenig *et al.* proposed an incremental version of A*, called Lifelong Planning A* (LPA*) [4]. It replans a shortest path from a source node to a target node in a given graph whenever a change in the graph is detected. It can successfully handle changes to costs of edges, node deletions, and node additions. For a given problem, LPA* finds the same paths as D* does, but using a different algorithmic procedure. LPA* is easy to implement and analyze. Consequently, a new algorithm called D* lite [5] was proposed for path replanning in goal-directed navigation as an extension to LPA*. D* lite is algorithmically shorter than LPA* as it avoids many complex conditional statements, yet all the properties of LPA* are inherited by D* lite. In contrast to LPA*, D* lite performs its search from the target node to the source node. Recently, we proposed Dynamic Z* algorithm [6], [7] as an incremental version of Z* algorithm [8] for energy-efficient path planning and replanning.

This paper presents an incremental search algorithm called *Rapidly Replanning A** (RRA*) as a generalization of Dynamic Z* [7], which can be used for optimal path planning and replanning on finite digraphs. The rest of this paper is organized as follows. Section II formulates the path replanning problem. The proposed RRA* algorithm is described in Section III. In Section IV, it is proven that RRA* always finds an optimal path from current location of an agent to a target location based on the updated map information. In Section V, the efficiency of RRA* is analyzed using simulations. Some concluding remarks are given in Section VI.

## II. Problem Formulation

Assume that a mobile agent is assigned a task of navigating from a given source location to a target location in an environment with partial or no information. Here, the path replanning problem is to find an optimal path from the current location of the agent to the target location based on the currently available map information whenever the previously planned path becomes infeasible. To facilitate the path planning process, a map of an environment is represented using a finite digraph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ in which a set of nodes $\mathcal{V}$ represents way points in the environment and a set of edges $\mathcal{E}$ represents connections between them. The traversal cost from a node $n \in \mathcal{V}$ to one of its neighboring node $n' \in \mathcal{V}$ is represented using a non-negative edge cost $c(n, n')$. Assume that $\mathcal{G}$ has been formulated based on currently available map information. Now, the path replanning problem can be transformed into a graph search problem in which an optimal path needs to be found from a node $q \in \mathcal{V}$ that represents the current location of the agent to a node $t$ that represents its target, using the updated $\mathcal{G}$.

## III. RRA* Algorithm

The proposed RRA* handles path replanning task efficiently by reusing its previous search results. Inspired by D* lite [5], the search direction of RRA* is set to be the reverse of A*, *i.e.* RRA* always starts its search form $t$ and continues until it reaches $q$. RRA* replicates node selection and expansion operations of A* with some modifications to its cost calculations. In RRA*, the selection of a node $n$ for expansion is based on the expected cost of traversing from the current location of the

agent to the target via a node $n$, which can be obtained by

$$f(n) = h(q, n) + g(n, t). \quad (1)$$

Here, $g(n, t)$ is the cost of a path $\lambda_{nt} = \langle n, \ldots, t \rangle$ from $n$ to $t$. An optimal path from $n$ to $t$ is denoted by $\lambda^*_{nt}$. In (1), $h(q, n)$ is a heuristic cost estimate of $\lambda^*_{qt}$. Note that $g(t, t) = h(q, q) = 0$.

The proposed RRA* algorithm is summarized in Algorithm 1. Initially, the agent's start location $s$ is same as the current location, *i.e.* $q \equiv s$. In initial planning, nodes with unknown traversability status are assumed to be traversable. Here, due to the reverse search direction, for any node $n_i, n_{i+1} \in \mathcal{V}$, the node $n_i$ is said to be a successor of $n_{i+1}$ if $(n_i, n_{i+1}) \in \mathcal{E}$. If $f$-cost of the node selected in Step 2a is infinitely large, there is no feasible path exists between the current location of the agent and its target, thus, the algorithm terminates. Otherwise, it jumps to Step 3 when the search process reaches $q$.

In the navigation step, the agent starts following the already found path while updating the traversability of neighboring nodes until the target is reached. If the preceding node of its current node in the search tree is unobstructed, the agent moves to the preceding node and update its position $q$. If the preceding node is obstructed, RRA* updates $\mathcal{G}$ by setting costs of all the edges that start and end at the abstracted nodes to $\infty$. Even though the current path is inaccessible beyond the obstructed node, the $g$-costs of the nodes which lie on that path in between the obstructed node and $t$ are still valid. Therefore, those costs need not to be recalculated in replanning steps. This can be identified as the main advantage of searching in reverse direction.

In order to carry out replanning efficiently, RRA* needs to make the maximum use of previous search results. On the other hand, paths resulted in replanning steps must be feasible and optimal. Based on these concepts, a new method is proposed for rearranging OPEN and CLOSED lists such that RRA* can efficiently reuse previous search results to find optimal paths in replanning steps. The proposed method rearranges nodes on OPEN and CLOSED lists using the search tree of the previous search. When an obstructed node is found on a planned path, the branches of the search tree which expands through that particular node become invalid beyond the obstructed node. However, if the nodes in those branches beyond the obstructed node are removed, the rest of the search tree can still be used for replanning of the paths. A rearrangement procedure for OPEN and CLOSED lists based on this concept is given in Step 4. Once OPEN and CLOSED lists are rearranged, $f$-costs of nodes on OPEN are updated according to the current location of the agent. Since $g$-costs of the nodes in OPEN remain the same, only $h$-costs have to be recalculated to obtain the corresponding $f$-costs. After the cost updates, RRA* recalculates the path with the updated OPEN and CLOSED lists.

Algorithm 1: RRA* algorithm

1. Initialize:
   a. Define two empty lists, OPEN and CLOSED.
   b. Set $q \leftarrow s$.
   c. Record $t$ on OPEN and define a pointer $Pred(t) \leftarrow$ NULL.
   d. Calculate initial costs, $g(t, t) \leftarrow 0$, $f(t) \leftarrow h(q, t)$.
2. Calculate a path:
   a. Remove a node $n_i$ from OPEN whose $f$-cost is minimum and record it on CLOSED. Resolve ties arbitrarily, but favor $q$.
   b. If $f(n_i) = \infty$, exit without a path.
   c. If $n_i = q$, go to Step 3.
   d. For all successors $n_{i+1}$ of $n_i$ in $\mathcal{G}$ that are not on CLOSED,
      i. Calculate $g(n_{i+1}, t) = c(n_{i+1}, n_i) + g(n_i, t)$ and $f(n_{i+1}) = h(q, n_{i+1}) + g(n_{i+1}, t)$.
      ii. If $n_{i+1}$ is on OPEN and $f(n_{i+1})$ is smaller than its previous estimate, update its $f$-cost and the pointer $Pred(n_{i+1}) \leftarrow n_i$.
      iii. If $n_{i+1}$ is not on OPEN, record $n_{i+1}$ on OPEN and define a pointer $Pred(n_{i+1}) \leftarrow n_i$.
   e. Go to Step 2a.
3. Navigate:
   a. Observe the traversability of neighboring nodes and update $\mathcal{G}$.
   b. If $Pred(q)$ is unobstructed, then
      i. Move to $Pred(q)$ and set $q \leftarrow Pred(q)$.
      ii. If $q \neq t$, go to Step 3a.
4. Rearrange lists:
   a. Remove $Pred(q)$ from CLOSED.
   b. Define a list TEMP $\leftarrow \{Pred(q)\}$.
   c. Remove an arbitrary node $n$ from TEMP.
   d. For all successors $n'$ of $n$ in $\mathcal{G}$,
      i. If $Pred(n') = n$ and $n'$ is on OPEN, then remove $n'$ from OPEN and put it on TEMP.
      ii. If $Pred(n') = n$ and $n'$ is on CLOSED, then remove $n'$ from CLOSED and put it on TEMP.
      iii. If $Pred(n') \neq n$ and $n'$ is on CLOSED, then remove $n'$ from CLOSED and put it on OPEN.
   e. If TEMP is not empty, go to Step 4c.
   f. If OPEN is not empty, update the $f$-costs of all the nodes in OPEN and go to Step 2.
   g. Go to Step 1c.

## IV. Analysis of RRA*

This section analyzes the optimality of the paths found by RRA*. Properties of A*-like search algorithms are clearly depend on their heuristics. In A* [9], an admissible heuristic provides an optimistic estimate of the goal cost whereas in RRA*, an admissible heuristic provides an optimistic estimate of the start cost.

*Definition 1:* A heuristic function $h$ used in RRA* is said to be admissible if

$$h(q, n) \leq k(q, n), \quad \forall n \in \mathcal{V}.$$

where $k(q, n)$ is the cost of $\lambda^*_{nt}$.
Similarly, the definition of consistency given for the heuristics used in A* [9], can be adapted for the heuristics used in RRA* as follows.

*Definition 2:* A heuristic function $h$ used in RRA* is said to be consistent if

$$h(q, n) \leq h(q, n') + k(n', n), \quad \forall n, n' \in \mathcal{V}.$$

By taking $n' = q$, it can be shown that all consistent heuristics are admissible. Here, all heuristics used with RRA* are assumed to be consistent.

Let $\mathcal{G}_m$ be the updated digraph used for $m^{\text{th}}$ execution of Step 2 of RRA* and $c_m^*$ be the cost of an optimal path from $q$ to $t$ in $\mathcal{G}_m$.

*Lemma 1:* If there exists a path from $q$ to $t$ in $\mathcal{G}_m$, then there exists an OPEN node $n_i$ on $\lambda_{qt}^* = \langle q, \ldots, n_i, \ldots, t\rangle$ with $f(n_i) \leq c_m^*$ while Step 2 of RRA* is being executed.

*Proof:* Initial planning using RRA* is the same as using A* except the search direction. Hence, using *Lemma 1* in [9], it can be shown that there exists an OPEN node $n_i$ on $\lambda_{st}^* = \langle s, \ldots, n_i, \ldots, t\rangle$ with $f(n_i) \leq c_1^*$ before RRA* exits from Step 2 in initial planning. Before every replanning step, RRA* rearrange OPEN and CLOSED lists in Step 4. In order to prove this lemma for replanning, we first prove that there exists an OPEN node $n_i$ on $\lambda_{qt}^*$ with $f(n_i) \leq c_m^*$ when Step 2 of RRA* is reached while replanning.

First assume that OPEN is empty after rearranging lists. Then, $t$ will be recorded on OPEN with $f(t) = h(q,t)$ in Step 1c. Using the admissibility of $h$, we have $f(t) \leq k(q,t) = c_m^*$.

Now assume that OPEN is not empty but there does not exist an OPEN node $n_i$ on $\lambda_{qt}^* = \langle q, \ldots, n_i, \ldots, t\rangle$ after rearranging lists. This can be considered under three scenarios: (a) None of the nodes on $\lambda_{qt}^*$ is on CLOSED. However, if OPEN is not empty, $t$ must be on CLOSED at any given time. (b) All of the nodes on $\lambda_{qt}^*$ are on CLOSED. However, when rearranging lists, $q$ is definitely removed from CLOSED. Therefore, there should be at least one node of $\lambda_{qt}^*$ which is not on CLOSED. (c) Some of the nodes on $\lambda_{qt}^*$ are on CLOSED. Let $n_{i-1}$ be the furthest node from $t$ on $\lambda_{qt}^*$ that is currently on CLOSED. Since the successor $n_i$ of $n_{i-1}$ on $\lambda_{qt}^*$ is currently not on CLOSED, it must have been removed from CLOSED in Step 4(d)iii. Therefore, $n_i$ must be on OPEN. However, $n_i$ might be on OPEN with $f(n_i) > c_m^*$. In Step 4f, $f(n_i)$ is calculated as $f(n_i) = h(q,n_i) + g(n_i,t)$. Using the admissibility of $h$, we have $f(n_i) \leq k(q,n_i) + g(n_i,t)$. Since $n_i$ is on an optimal path, $k(q,n_i) + g(n_i,t) = c_m^*$. Thus, there exists an OPEN node $n_i$ on $\lambda_{qt}^*$ with $f(n_i) \leq c_m^*$ when Step 2 is reached while replanning.

In order to complete the proof, it is necessary to show that there exists an OPEN node $n_i$ on $\lambda_{qt}^*$ with $f(n_i) \leq c_m^*$ while Step 2 of RRA* is being executed in replanning. It has been already showed that Step 2 is reached in replanning, there is an OPEN node $n_i$ on $\lambda_{qt}^*$ with $f(n_i) \leq c_m^*$. Whenever, $n_i$ is moved from OPEN to CLOSED, the successor $n_{i+1}$ of $n_i$ on $\lambda_{qt}^*$ will be recorded on OPEN with $f(n_{i+1}) = h(q,n_{i+1}) + c(n_{i+1},n_i) + g(n_i,t)$. Using the admissibility of $h$, we have $f(n_{i+1}) \leq k(q,n_{i+1}) + g(n_{i+1},t)$. Since, $n_{i+1}$ is on $\lambda_{qt}^*$, $k(q,n_{i+1}) + g(n_{i+1},t) = c_m^*$, and therefore, there exists an OPEN node $n_{i+1}$ on $\lambda_{qt}^*$ with $f(n_{i+1}) \leq c_m^*$ while Step 2 of RRA* is being executed. ■

*Theorem 1:* RRA* guarantees to find an optimal path $\lambda_{qt}^*$ in $\mathcal{G}_m$ if such a path exists.

*Proof:* A sequence of contradictions are used for this proof. First, assume that RRA* does not terminate on $\mathcal{G}_m$. However, any best-first search strategy that prunes cyclic paths always terminates on finite graphs [9], thus, RRA* must terminate on $\mathcal{G}_m$. Now assume that RRA* terminates without a path in Step 2b. If there exists a path from $q$ to $t$ in $\mathcal{G}_m$, according to *Lemma 1*, there exists an OPEN node $n_i$ on $\lambda_{qt}^*$ with $f(n_i) \leq c_m^*$ while Step 2 of RRA* is being executed. In Step 2a, RRA* selects a node from OPEN whose $f$-cost is minimum. Therefore, if there exists a path from $q$ to $t$ in $\mathcal{G}_m$, RRA* cannot terminate at Step 2b without a path. Even if RRA* finds a path $\lambda_{qt}$ in $\mathcal{G}_m$, assume that $\lambda_{qt}$ is sub-optimal, *i.e.* $f(t) > c_m^*$. However, RRA* always select a node with minimum $f$-cost from OPEN. Therefore, by the time that RRA* selects $t$ for expanding along $\lambda_{qt}$, $t$ should be minimum $f$-cost among the nodes in OPEN which contradict with *Lemma 1*. Thus, RRA* guarantees to find an optimal path $\lambda_{qt}^*$ in $\mathcal{G}_m$ if such a path exists. ■

Here, it is assumed that heuristics used with RRA* are consistent. Nevertheless, if the heuristics are admissible but not consistent, then RRA* may need to re-expand the nodes on CLOSED and improve their costs to find optimal paths.

## V. Simulations

A set of simulations were performed to evaluate the computational efficiency of RRA* and to verify the optimality of paths found. Benchmark results were obtained using A* [1] which guarantees to find a shortest path by exploring a minimum number of nodes when it is guided by consistent heuristics.

Simulations were performed using two terrain models described in [6]. Parameters of two simulation setups are given in Table I. A grid-based elevation map of a terrain model is transformed into a finite digraph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. A node $n \in \mathcal{V}$ represents a center of a cell in the elevation map. Each node is connected up to 8 other nodes which represent neighboring cells. For $(n,n') \in \mathcal{E}$, the edge cost is calculated using

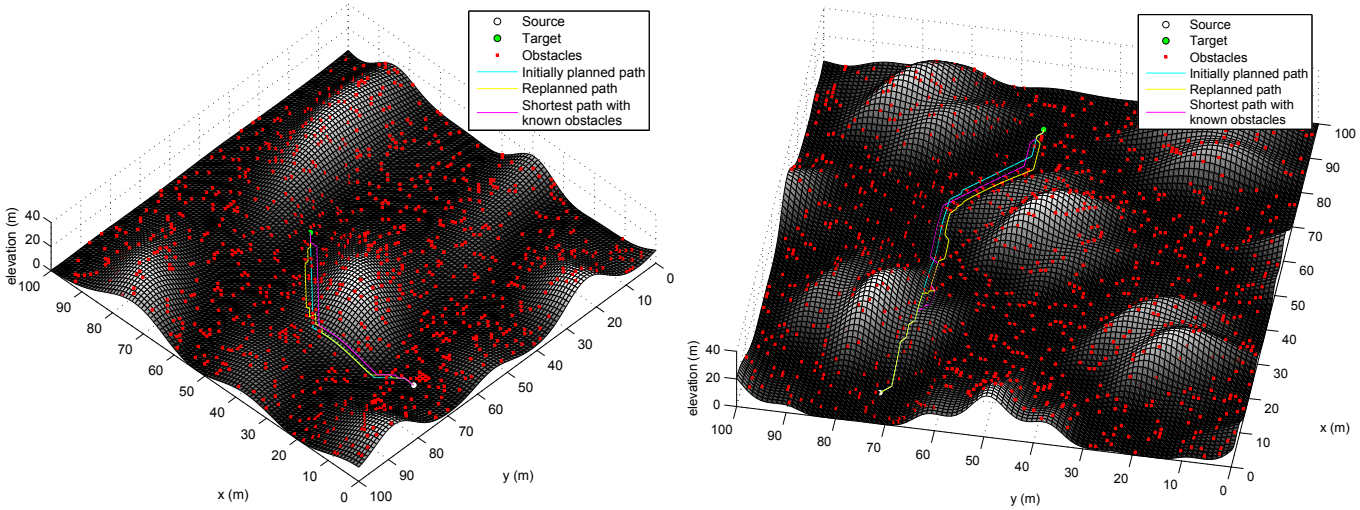
$$c(n,n') = \begin{cases} \infty, & \text{if } \phi(n,n') > \phi_m, \\ d(n,n'), & \text{otherwise,} \end{cases}$$

where $d(n,n')$ is the Euclidean distance from $n$ to $n'$, $\phi(n,n')$ is the angle of inclination from $n$ to $n'$, and $\phi_m$ is the critical impermissible angle for uphill traversal [10]. In simulations, $\phi_m = 44.71°$. The heuristic cost is calculated using the Euclidean distance, *i.e.* $h(q,n) = d(q,n)$. Obstacles are distributed uniformly at random such that 10% of cells in elevation maps of terrains are obstructed in each simulation.

The simulations were performed under two scenarios to evaluate performances of RRA* in feasible shortest path replanning on uneven terrains. In the first scenario, the algorithms under test were provided with locations of obstacles prior to initial planning and in the second scenario, they were not. According to the simulation results, RRA* is capable of finding the same shortest paths as A* does, with or without having prior knowledge of the obstacle locations. If the algorithms under test had complete information about the obstacle locations prior to initial planning, paths generated do not coincide with any obstacles, thus, no replanning takes place. If the obstacle locations are unknown prior to the initial planning, the algorithms under test assume that all the cells in

TABLE I: SIMULATION PARAMETERS AND RESULTS.

| Setup | Terrain model [6] | $s$ (m) | $t$ (m) | Path length (m) | | | Number of replans | Number of nodes expanded in replanning | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Shortest path with known obstacles | Initially planned path with unknown obstacles | Replanned path | | A* | RRA* |
| I | 1 | (13,69) | (65,50) | 70.35 | 68.88 | 72.19 | 8 | 1556 | 31 |
| II | 2 | (09,73) | (89,54) | 98.75 | 95.97 | 101.37 | 8 | 3918 | 36 |




(a) (b)

Fig. 1: Shortest paths generated by the algorithms under test using (a) simulation setup I and (b) simulation setup II.

the elevation map are traversable. Both A* and RRA* return the same paths that are optimal if the terrain is obstacle free. However, these paths coincide with several obstacles. The agent cannot reach its target by simply following these paths, thus, replanning has to be taken place when the agent discovers that the next node on the path is obstructed. Under each simulation setup, paths are replanned 8 times. Such replanning usually results in longer paths compared to the paths that are found with prior information about obstacle locations. Nevertheless, RRA* is capable of finding equally optimal paths when compare to those found by repeatedly applying A*. According to the results given in Table I, RRA* has expanded significantly fewer number of nodes for path replanning during navigation. These results verify the optimality of the path generated by RRA* and the superior efficiency of RRA* in path replanning.

## VI. Conclusion

This paper proposed RRA* algorithm which can be used for rapid path replanning while navigating in partially known environments. The search process of RRA* starts from the target node and proceeds till it reach the current location of an agent. If the traversability of a node is not available by the time of planning, it is assumed to be traversable. Once a path is obtained, the agent follows that till the goal is reached if all nodes on that path is unobstructed. However, if the path is obstructed, RRA* reuses the previous search results to efficiently replan a new path after removing the obsolete branches in the search tree. The analysis provided in this paper proves that RRA* always find an optimal path based on the updated information.

## Acknowledgment

This work is supported by the Hong Kong PhD Fellowship Scheme (Project RTKL) and the Department of Electronic and Information Engineering, the Hong Kong Polytechnic University (Project G-YBKH).

## References

[1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[2] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1994, pp. 3310–3317.

[3] A. Stentz, "The Focussed D* algorithm for real-time replanning." in *International Joint Conference on Artificial Intelligence*, vol. 95, 1995, pp. 1652–1659.

[4] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong Planning A*," *Artificial Intelligence*, vol. 155, no. 1, pp. 93–146, 2004.

[5] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Trans. on Robotics*, vol. 21, no. 3, pp. 354–363, 2005.

[6] N. Ganganath, C.-T. Cheng, and C. K. Tse, "Rapid replanning of energy-efficient paths for navigation on uneven terrains," in *IEEE International Conference on Industrial Informatics (INDIN)*. IEEE, 2015, pp. 408–413.

[7] N. Ganganath, C.-T. Cheng, and C. K. Tse, "An improved Dynamic Z* algorithm for rapid replanning of energy-efficient paths," in *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*. IEEE, 2015, pp. 395–398.

[8] N. Ganganath, C.-T. Cheng, and C. K. Tse, "A constraint-aware heuristic path planner for finding energy-efficient paths on uneven terrains," *IEEE Trans. on Industrial Informatics*, vol. 11, no. 3, pp. 601–611, 2015.

[9] J. Pearl, *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Pub. Co., Inc., Reading, MA, 1984, ch. Formal properties of heuristic methods, pp. 73–85.

[10] N. Ganganath, C.-T. Cheng, and C. K. Tse, "Multiobjective path planning on uneven terrains based on NAMOA*," in *IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2016, pp. 1846–1849.