Outdoors Evacuation Routes Algorithm Using Cellular Automata and Graph Theory for Uphills and Downhills

Washington Velasquez * and Manuel S. Alvarez-Alvarado

Faculty of Electrical and Computer Engineering, Escuela Superior Politécnica del Litoral, Campus Gustavo Galindo Km 30.5 Vía Perimetral, Guayaquil P.O. Box 09-01-5863, Ecuador; manuel.alvarez.alvarado@ieee.org
* Correspondence: wavelasq@espol.edu.ec

Abstract: This paper proposes an innovative algorithm for the determination of the optimal outdoor evacuation routes in hills. The algorithm uses web services to obtain the Google Image Tile (a portion of the area to evaluate) and each area’s geographic information. Then, the routes are calculated using cellular automata in 3D and graph theory with geographic information obtained from Google Maps (latitude, longitude, and elevation). The calculation of evacuation routes per study area is performed in an adequate time, revealing the potential of the proposed approach for different applications in emergency scenarios such as wildfires, smart transport, and climbing.

Keywords: automata cells; graph theory; outdoor; routes

1. Introduction

An evacuation route is specifically designed to evacuate people from facilities and geographical places in the shortest possible time in case of an emergency [1,2]. These routes must be continuous and unobstructed, from a starting point (office, building, street, mountain, etc.) to a safe place, generally outside. They can be defined as horizontal or vertical, or a combination of both [3]. The routes’ design follows the evacuation and contingency plan of a company or government, establishing an order, program, or follow-up of actions, whose purpose is to integrate all the elements to help cases of emergencies and disasters such as earthquakes, fires, floods, landslides, pollution, etc. [4,5]. The crucial activity in these situations is to advise as soon as possible about a safe zone [1]. However, providing effective communication in such circumstances presents a significant challenge because of the great diversity of people within a building or a locality, e.g., people with disability and other vulnerable groups [6], adverse conditions such as stress [7], and visual impairment (e.g., fire causing smoke) [8]. In this context, evacuation routes are divided into indoor and outdoor. Several works study evacuation routes from an indoor perspective, i.e., buildings, schools, and laboratories. The design of these routes is highly dependent on the equipment, infrastructure, and location of buildings [9], as well as potential sources of external hazards such as gas stations, gas pipelines, advertisements, etc. The analysis is carried out with the entire building’s architectural blueprints and facilities to determine the work environments, influx, and optimal escape routes [10,11]. For example, in [12], a geographic information system implements evacuation routes based on the shortest distance to a nearby exit. Velásquez et al. [13] described the use of the body mass index (BMI) for the calculation of evacuation routes using graph theory and at the same time obtaining a risk index of the building. Finally, the authors of [14,15] presented a proposal to generate dynamic evacuation routes indoors according to obstructions caused by fire or collapse of walls to direct users to the nearest exit. The system uses a sensor network to monitor the building’s ways, and it uses signals to guide people to a safe area (outside the building). Nevertheless, the research areas in outdoor evacuation routes focus on providing aid and relief mechanisms in the face of a natural disaster. For example, in [16], routes are evaluated in areas affected by the phenomenon of subsidence, i.e., in the event of a
rupture of the coastal dyke due to a seismic shock. Pratiwi et al. [17] described a method to elaborate the cartography of evacuation routes in the event of a tsunami, where surface costs are determined and, with them, the optimal route to evacuate from the flood zone to shelters or safe areas. Finally, Jeong et al. [18] presented green infrastructure using spatial syntax and Geographical Information System (GIS) network analysis. The purpose of the study is to provide evacuation routes in the coastal profile against possible floods.

Paths, sidewalks, and highways generally are used to create outdoor evacuation routes that guide people to a safe area. These zones could be elevated spaces in floods, bunkers in the case of tornadoes, and clear zones in an earthquake. It all depends on the geographic area and the possible natural disasters that could occur. These evacuation routes are previously defined and duly marked in disaster-prone localities. Most of the inhabitants know at least the road to the safeguarded area closest to their home. For example, Baños de agua santa is a city in the province of Tungurahua in Ecuador. It is known for its access to ways in the active Tungurahua volcano and its mineral-rich hot springs. Because the city is located in an area surrounded by mountains and prone to a possible natural disaster, it has several evacuation routes that guide people to shelters and safe places in case of a volcanic eruption [19]. However, suppose the approach to creating outdoor evacuation routes is focused on mountains. In that case, the prospects for using human-defined paths vary. Certain adverse factors can weaken evacuation, e.g., the slope of a mountain, adrenaline, obstructions, and if a person goes uphill or downhill [20]. Due to this type of soil’s topographic influence, it is customary to mark the routes where people should walk; generally, they form a zigzag path that decreases the slope [21]. In this context, a study carried out in a mountainous area of China used an automated cell with topography surface in 3D to simulate pedestrians’ evacuation following predefined routes [22]. This work focuses on the terrain topography and speed variation of people, resulting in a simulation that allows observing the population’s behavior in congestion, queues, and collision prevention. Nevertheless, what would happen if the roads are blocked? Are there mountains not inhabited by human beings? Are there people trapped by a forest fire? Such emergencies need a more exhaustive analysis because the road’s slope influences the speed of evacuation. Previously defined paths may be obstructed, even in areas prone to forest fires, where the environment’s variability could play against the rangers or firefighters. Although several evacuation route calculations work in different applications and from different perspectives (indoors and outdoors), the current literature does not relate the evacuation route calculations in the mountainous area according to the meteorological changes caused by the emergency.

This paper describes a novel algorithm to calculate evacuation routes in the open considering sloping ground (hill) and whether the user moves up or down. The model is based on the evacuation routes that the animals take in cases of emergencies [23]. The algorithm uses 3D cellular automata that simulate the topography of the study area. This mechanism allows finding the lowest slope points with a spherical coordinate in Google Maps (latitude, longitude, and elevation). Besides, it makes a mapping of the area in order to find possible evacuation routes using graph theory to find the optimal route from an initial point \((X_i, Y_i)\) in the automated cell to a final position \((X_f, Y_f)\), representing the final positions as the possible safe zones. The algorithm allows easy adaptation to people’s movement, depending on whether the user decides to go uphill or downhill to find the optimal evacuation routes.

The remainder of this paper is divided into five sections. Section 2 describes the methodology to calculate optimal evacuation routes, using the Moore model to design an automated cell in 3D with geolocation values (latitude, longitude, and elevation) obtained from Google Maps and creating a graph with the Dijkstra algorithm to find the evacuation routes. Section 3 presents the case study describing the area with the location points and the work environment. Section 4 describes the simulation and the results obtained from the algorithm implementation. Finally, a discussion of the algorithm and its viability to be incorporated into various applications or emergencies is presented.
2. Methodology

This section describes the methodology for calculating evacuation routes outdoors. The different stages and the computational tools for the creation of the algorithm are described.

2.1. Geographical Information

Evacuation routes are created from a specific geographic area. The algorithm gets a Google Tile Image (GTI) of a position (latitude and longitude) using Google Maps API [24]. The GTIs are a portion of the entire analysis area with a size of 256 × 256 pixels. Additionally, a 256 × 256 pixel matrix of the same size as the GTI is created; each pixel represents a coordinate on the map with a respective latitude (lat), longitude (long), and elevation (ele), as shown in Figure 1. The Google Maps Elevation API [25] is used to obtain each cell’s elevation (ele) coordinate in the matrix. The data structure [26] that each cell has is described below.

- Latitude (lat), Longitude (long), and Elevation (ele): The geographical coordinates used refer to Google Maps with the cellular automata. A precision of eight digits is used so that the evaluation points are not so distant in the cellular automata.
- \((X_i, Y_i)\): Positions in the matrix are used for the user’s movement in the cellular automata, i.e., positions are used to create evacuation routes.
- States \(S\): This is a finite non-empty set of states used in cellular automata. Each state changes depending on the user’s location and if it has been evaluated in the route calculation.

![Figure 1. Creation of the cellular automata with geographic information.](image)

2.2. Cellular Automata

A cellular automata (CA) is a mathematical model for a dynamic system that evolves in discrete steps depending on transition states [27]. CA can be used in several applications, especially in the calculation of evacuation routes. For example, in [28], a model is presented using CA to redirect pedestrians to a closer exit in case they change direction. In [29], a simulation of a room with multiple exits is presented, where a two-dimensional model of basic cellular automata based on human behavior is proposed. This model is used to create evacuation routes and redirects people to the nearest exit.

CA is a dynamic system capable of replicating itself according to the values or states of the set of cells or cells that make it up. The states are instantaneously varied in discrete units, and the value following this depends on the current state of the cell and the current state of its neighborhood [30,31]. A CA has several properties that are useful for creating routes:
• **Network Infrastructure**: The cells in a CA are usually rectangular or square, although they can have other topologies, but always with a finite-dimensional. This CA case is $256 \times 256$.

• **State $S$**: There is a finite set of states in a CA; each cell must be in a defined state. The set of states on the different cells will help us to identify the possibility that there is a path through the cells.

• $S_N, S_{NW}, S_{NE}, S_S, S_{SW}, S_{SE}, S_W,$ and $S_E$: They are labels of the cellular automaton states to show the user the direction of their evacuation route.

• **Cell neighborhood**: Each cell will have a neighborhood, usually defined by the developer; for two-dimensional structures, Moore and von Neumann schemes are the most used. Each cell’s status and those of neighboring cells within the neighborhood will depend on the transition rules that will be applied in each case [30]. A general scheme is shown in Figure 2.

• $(X_i, Y_i)$: Cell position within the cellular automata. The position $(i)$ is considered as the initial position, and the surrounding positions as its neighborhood.

• **Transition rules**: Each cell in the CA has applied a rule, which determines its state based on its state in time $(T - 1)$ and on the state of its neighbors in time $T$.

![Figure 2. Cellular automata with Cartesian coordinates in the Moore model.](image)

The CA model for calculating evacuation routes is described in Figure 2, where the Moore model is used to analyze each of the cells that surround a cell $(X_i, Y_i)$. However, for the analysis of a route on a hill, labels have been added that simulate a direction of the cardinal system (IP: Initial Point; N, North; NW, Northwest; NE, Northeast; S, South; SW, Southwest; SE, Southeast; E, East; W, West), and, for the transition states $S$, the following labels are used:

• Empty Zone (0): Initial state of the cell.

• Evaluated Zone (1): If the cell is previously evaluated but is not considered acceptable in the final route.

• Accepted Zone (2): When the cells around a cell $(X_i, Y_i)$ are analyzed and it presents a lower slope with respect to the initial cell (considering if the person is walking downhill).

The cellular automata initially evaluate each cell in the neighborhood of $P(X_i, Y_i)$ and assign a state to the cells (1 or 2). Those points that meet the requirement (uphill or downhill) are added to the graph. This graph can have multiple ramifications due to the precision in the geographic coordinates.

2.3. Evacuation Routes

A graph is a set of objects called vertices or nodes, joined by lines known as edges. The graphs allow studying the interrelationships between units that interact with each other as
it occurs, e.g., in a LAN network [32], where the nodes would be the routers, switches, and hosts, while the edges are the cables through which the connection is established. There are different search methods in graphs; one of the best known and used is the Dijkstra Algorithm, which calculates the most optimal route from a point [12]. However, there are also the BFS (Breadth-First Search) and DFS (Depth First Search) algorithms, and both are examples of uninformed search algorithms [33]. The Dijkstra algorithm is used in this study. It consists of exploring the shortest path, which starts from an origin node to all related nodes, growing throughout the graph. The algorithm uses the value of the slope as the edge weight; i.e., if it has a point \( P_i = (X_i, Y_i) \) and a point \( P_j = (X_j, Y_j) \), the slope between both points represents the distance (edge) between the two nodes. The weights are added in each discrete step of the cellular automata. Later, a destination and the direction (up or down) are selected. The user’s direction is configured manually, i.e., a user configures whether to go uphill and downhill.

The evacuation route calculation uses the following notation:

- \( P_{start} = (X_{start}, Y_{start}) \): Initial starting point (when the user starts walking).
- \( P_{save} = (X_{save}, Y_{save}) \): These points are the final position where the user is going; they are used to know the graph’s optimal evacuation route.
- \( N, NW, NE, S, SW, SE, W, \) and \( E \): Labels on the edges to know the direction of evacuation routes.

Figure 3 shows the graph diagram to create the evacuation routes. When the cellular automata has been created and an IP \((X_i, Y_i)\) is specified in the matrix, it begins with the calculation of evacuation routes using a directed graph, \( G = (Node, Edge) \), where an edge \( (e) \) is considered directed if it goes from a path \((x, y)\) [12]. The route calculation starts from a position \( P_{start} = (X_{start}, Y_{start}) \), adding a new node \( N \) to the graph. Depending on whether the user moves uphill or downhill, assuming that you have two positions \( P_i = (lat_i, lng_i, ele_i) \) and \( P_j = (lat_j, lng_j, ele_j) \), the distance \( d \) is found using the GlobalMercator class provided by Google. This class has methods to convert the positions \((lat, lng)\) into meters \((mx, my)\) and thus obtain the distance \( d \). The calculation diagram is shown in Figure 4.

Figure 3. Graph for calculating the optimal evacuation route.
The ground elevation \( h \) between the points \((P_i, P_j)\) is defined by Equation (1):

\[
h = ele_f - ele_i
\]

The evacuation time is calculated based on the decision made by the user, such that defined by the user,

\[
T_{\text{evacuation}}(\theta) = \begin{cases} 
\text{distanceuphill/} \text{humanspeed} & \Rightarrow 0 \leq \theta < 180^\circ \\
\text{distancedownhill/} \text{humanspeed} & \Rightarrow 180^\circ \leq \theta < 360^\circ 
\end{cases}
\]

(2)

where the angle \( \theta \) is calculated based on the slope.

On the other hand, the slope is a function can be written as follows:

\[
slope = \tan \theta = \frac{h}{\sqrt{d^2 - h^2}}
\]

(3)

However, a more comprehensive way to express Equation (3) is:

\[
slope = \frac{ele}{\sqrt{lat^2 + long^2 + ele^2 - ele^2}}
\]

(4)

The model currently uses the slope to calculate possible evacuation routes. However, due to the precision of eight digits at each point in the matrix, the distance between \( P_i = (X_i, Y_i) \) and \( P_j = (X_j, Y_j) \) is minimal (approximately one step). For this reason, the graph creates several branches in different directions of the Cartesian plane, considering that the neighborhood of a point \( P_i \) is evaluated. Finally, a \( P_{\text{save}} \) is configured and the best evacuation route is obtained from a \( P_{\text{start}} \) using the Dijkstra algorithm.

The main section of the code is described in Algorithm 1, and the evacuation routes flowchart is shown in Figure A1 in Appendix A.
Algorithm 1: Evacuation Routes Algorithm

Result: Evacuation routes

Input: Location, Init point, Tile, index, image name.

Output:

- x : List with points of the path in axis X #Longitude.
- y : List with points of the path in axis Y #Latitude.

#Route Graph

The GTI of the image to be processed is obtained

Algorithm: neighbor maint SimulationCellularAutomata(Image Name, Location);

# Process each cell in the matrix while Not being in a safe area do

- **if The next step is accepted zone then**
  - initTile, tile, index = evacuation.initRoute(targetTile, tile);
  - change parameters (targetTile, tile);

- **else**
  - CA = neigh.meshPointsXlocation[i][“mesh”];
  - dgraph = evacuation.forest.G(Location, CA);
  - targetTile = evacuation.safeZoneTiles(index, neighbor.meshPointsXlocation);
  - targetNodo = evacuation.tileTograph(dgraph, targetTile);
  - routeX, routeY = evacuation.RouteGraph(dgraph, punto de inicio, targetNodo);
  - routeX.extend(routeX)
  - routeY.extend(routeY)

end

end

ax1 = Figure.add_subplot(111)

x = np.array([routeX]) #Longitude array
y = np.array([routeY]) #Latitude array

ax1.plot.wireframe(x, y)

plt.show()

3. Case Study

The analysis of the algorithm was done using the simulation tool described in [26], where a square area was created to explore evacuation routes using Google Maps, as shown in Figure 5. This tool was designed to serve as a resource for relief agencies, allowing immediate and effective intervention in an emergency.

Figure 5. Study area \( R = (A, B, C, D) \) using Google Maps.
The tool gets the locations’ elevation points and saves them in Neo4j Database (Graph Database) [34]. These points allow obtaining the study area’s image and create the different GTIs and, consequently, the cellular automata. For example, the information from Points A and B is shown in Table 1 and Figure 6. Here, the maximum elevation is 16.68 m, and the minimum is 7.73 m with a distance between (A, B) of 106.195 m. The simulation tool shows the elevations from points (A – B), (B – C), (C – D), and (D – A).

Table 1. Information of points (A, B).

<table>
<thead>
<tr>
<th>Point</th>
<th>Latitude</th>
<th>Longitude</th>
<th>Elevation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>40.13897826</td>
<td>-8.69440953</td>
<td>7.77376699</td>
</tr>
<tr>
<td>B</td>
<td>40.13802564</td>
<td>-8.69434342</td>
<td>16.68270111</td>
</tr>
</tbody>
</table>

Figure 6. Elevation of all points between (A, B) obtained from Google Maps.

When the points (A, B, C, D) have been configured, an image of the entire area is obtained that is divided into 25 GTIs of 256 × 256 pixels.

4. Results and Discussion

This section shows the case study results using a tool that was designed with Node.js and React.js to obtain evacuation routes [26]. The tool requires the user to configure the study area (Figure 5). Thus, it pulls the image from Google Maps and the algorithm uses the GoogleMercator class to divide it into 25 GTIs. Then, a GTI is extracted depending on the location of the user \(P_{start}\). Figure 7 shows the study area, and the Tile \((x, y) = (124,744, 99,110)\) is extracted to create the automated cells matrix. The study area is approximately 1.8 km × 1.5 km.

Figure 7. Google Tile Image (GTI) of the study area.
The first GTI selected represents the user’s starting location \((P_{\text{start}})\). This GTI represents a \(256 \times 256\) pixel image, where each pixel represents a possible node within the graph. Each GTI carries out the analysis; that is, each image is processed separately, and, at the border points of the GTI, they are joined with the next one. Everything depends on the user’s direction (uphill or downhill). The graph has several \(P_{\text{save}}\) points of the entire study area; these allow obtaining the evacuation route. For example, if the user is at position \(P_{\text{start}} = (X_{\text{start}}, Y_{\text{start}}) = (150, 180)\); GTI \((124,744, 99,110)\), and the Dijkstra algorithm is executed towards position \(P_{\text{save},1} = (X_{\text{save}}, Y_{\text{save}}) = (134, 192)\), GTI \((124,743, 99,111)\), as shown in Figure 8. The algorithm processes three different GTIs to create a graph with each GTI and finds a more precise evacuation route.

![Figure 8. Graph using different GTIs.](image)

The processing of the automated cells for each GTI was carried out separately to save computational resources (memory). It was more feasible to process a \(256 \times 256\) pixel matrix that took approximately 1.5 s per GTI to obtain the graph’s nodes that met the requirement (uphill or downhill) than a \(256 \times 256 \times 256\) pixel image requiring an average of 10 min to obtain the total evacuation route. It is essential to mention that each GTI produces a graph with multiple branches. Therefore, each time a GTI pixel is on the edge of the image, a new GTI is selected to follow the process. This mechanic causes several GTIs to be processed in parallel. If the user wants to process a more extensive study area, it is recommended to use a computer designed for image processing.

The application allows visualizing the user’s evacuation route in two ways: (1) in two dimensions using Google Maps; and (2) in three dimensions using the javascript library plotly.min.js ([https://codigofuente.io/crear-un-grafico-de-superficie-3d-con-javascript/](https://codigofuente.io/crear-un-grafico-de-superficie-3d-con-javascript/)) (accessed on 15 November 2020). Figure 9 shows the evacuation route from the point \(P_{\text{start}} = (X_{\text{start}}, Y_{\text{start}}) = (150, 180)\); GTI \((124,744, 99,110)\), and the Dijkstra algorithm is executed towards position \(P_{\text{save},1} = (X_{\text{save}}, Y_{\text{save}}) = (134, 192)\), GTI \((124,743, 99,111)\), as shown in Figure 8. The algorithm processes three different GTIs to create a graph with each GTI and finds a more precise evacuation route.
\((X_{start}, Y_{start}) = (150, 180)\) of the GTI \((124,743, 99,112)\) to the \(P_{save_1} = (X_{save_1}, Y_{save_1}) = (134, 192)\) of the GTI \((124,742, 99,112)\) in two dimensions.

\[\text{Figure 9. Evacuation route in the application.}\]

On the other hand, consider that points \(P_{start} = (X_{start}, Y_{start}) = (150, 180)\) and \(P_{save_1} = (X_{save_1}, Y_{save_1}) = (134, 192)\) belong to GTI \((124,740, 99,110)\), i.e., the starting point and the arrival point are in a unique GTI. The user wants to go uphill. The preliminary result is a PATH with values of the nodes that refer to the automated cell. The evacuation route in this scenario is:

**PATH:**
\[
P_{start} = (150,180) \rightarrow (149,179) \rightarrow (148,178) \rightarrow (147,179) \rightarrow (146,180) \rightarrow (145,181) \rightarrow (144,182) \rightarrow (143,183) \rightarrow (142,184) \rightarrow (141,185) \rightarrow (140,186) \rightarrow (138,187) \rightarrow (138,188) \rightarrow (137,189) \rightarrow (136,190) \rightarrow (135,191) \rightarrow P_{save_1} = (134, 192).
\]

This sequence belongs to the evacuation route obtained by the graph. The algorithm processes each node of that route to extract the automated cells’ characteristics (latitude, longitude, and elevation) belonging to GTI \((124,740, 99,110)\). It is necessary to obtain these characteristics of the cell because the graph does not save these values. The application uses them to show the way that the user should follow. Finally, Figure 10 shows a diagram of the evacuation route using the elevation of the terrain. It is observed that the user moves from position \((0,0,0)\) that belongs to the point \(P_{start} = (X_{start}, Y_{start}) = (150, 180)\) and must go uphill around 25 m to \(P_{save_1} = (X_{save_1}, Y_{save_1}) = (134, 192)\).
5. Conclusions

This paper presents an innovative algorithm based on cellular automata and graph theory to dynamically calculate the optimal evacuation route in geographic positioning (latitude, longitude, and elevation). The algorithm maps a specific study area and converts it into an automated cell in 3D; a directed graph is constructed according to the user’s direction (uphills and downhills), using the locations with the lowest slope. The calculation of evacuation routes per study area (GTI) is 1.5 s; this result ensures that the algorithm can be applied to study areas more extensively. However, the algorithm currently does not present variations in the decision for each cell; i.e., when selecting a direction (uphill or downhill) and the $P_{safe}$ points, the direction is not reconfigured. Finally, the evacuation route is calculated using Dijkstra’s algorithm.

The algorithm has the potential to be implemented in digital apps for the benefit of users in the case of emergency scenarios. For instance, the algorithm can be employed to evacuate people in emergencies or establish safe routes over hills quickly. The algorithm does not consider the obstacles hindering the progression of the user along the path. Moreover, it does not show the global optima since the routes depend on the user decision, becoming limitations of the proposed approach. However, in future research, the use of advanced metaheuristics techniques, such as grey wolf optimization and accelerated quantum particle swarm optimization, could significantly improve the algorithm’s computational efficiency, leading to the implementation of optimal routes in the presence of obstacles. Some applications can be developed with the proposed approach to improve evacuation times and rescue people by relief agencies. Besides, with the development of the application to test the algorithm, it opens the door to create a tool that users can use in different situations, e.g., escaping from a forest fire, and relief agencies for a better understanding of the terrain and prepare contingency rescue plans.

**Author Contributions:** Conceptualization, W.V.; methodology, W.V. and M.S.A.-A.; software, W.V.; validation, W.V. and M.S.A.-A.; formal analysis, W.V.; investigation, W.V. and M.S.A.-A.; resources, W.V. and M.S.A.-A.; data curation, W.V.; writing—original draft preparation, W.V. and M.S.A.-A.; writing—review and editing, W.V. and M.S.A.-A.; visualization, W.V.; and project administration, W.V. and M.S.A.-A. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Acknowledgments:** The authors would like to thank Escuela Superior Politécnica del Litoral.
Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Evacuation Routes Flowchart

Figure A1 shows the flow chart for calculating evacuation routes. The start of the process, the creation of GTI, the creation of automated cells, the change of state of the cells, and the creation of routes with the Dijkstra algorithm are detailed.

References


19. Herrera, G.; Rodríguez, G. Resiliencia y turismo: El caso de la ciudad de Baños de Agua Santa-Ecuador. HOLOS 2016, 3, 229–250. [CrossRef]


