This is a pre-print of a contribution published in "Robot 2019: Fourth Iberian Robotics Conference. Advances in Intelligent Systems and Computing, Vol 1093. Silva M., Luís Lima J., Reis L., Sanfeliu A., Tardioli D. (eds)" published by Springer, Cham. The final authenticated version is avalaible online at: https://doi.org/10.1007.987-3-030-36150-1_10

# Velocity-Based Heuristic Evaluation for Path Planning and Vehicle Routing for Victim Assistance in Disaster Scenarios

Manuel Toscano-Moreno ✉, Anthony Mandow, María Alcázar Martínez, and Alfonso García-Cerezo

Universidad de Málaga, Systems Engineering & Automation Dept., Málaga, Spain, m.toscano@uma.es,
WWW home page: https://www.uma.es/robotics-and-mechatronics

**Abstract.** Natural and human-made disasters require effective victim assistance and last-mile relief supply operations with teams of ground vehicles. In these applications, digital elevation models (DEM) can provide accurate knowledge for safe vehicle motion planning but grid representation results in very large search graphs. Furthermore, travel time, which becomes a crucial cost optimization criterion, may be affected by inclination and other challenging terrain characteristics. In this paper, our goal is to evaluate a search heuristic function based on anisotropic vehicle velocity restrictions for building the cost matrix required for multi-vehicle routing on natural terrain and disaster sites. The heuristic is applied to compute the fastest travel times between every pair of matrix elements by means of a path planning algorithm. The analysis is based on a case study on the ortophotographic-based DEM of natural terrain with different target points, where the proposed heuristic is compared against an exhaustive search solution.

**Keywords:** multi-robot team, heuristics, search and rescue, path planning, vehicle routing problem

## 1 Introduction

Efficient management of paths and routes for teams of autonomous off-road vehicles is crucial for challenging robotic applications such as planetary exploration [11], agriculture [4], and search and rescue (SAR) in post-disaster situations [6]. In off-road environments, the absence of pre-defined roads results in a much larger search space, and terrain characteristics, such as gradient, affect vehicle mobility in aspects such as navigability, travel time or energy consumption. In this sense, grid representations such as digital elevation models (DEM) can capture terrain knowledge for each cell that is useful for considering terrain slope navigability [9][14], fuel consumption estimation [13], the presence of victims at risk in rescue operations [12], or weed infestation in farming environments [4].

There is a growing interest in the vehicle routing problem (VRP) for fleets of unmanned aerial vehicles (UAV) [10]. However, few works have addressed

the vehicle routing problems by considering terrain elevation. In [3] a digital elevation map is considered for a team of UAVs in a terrain mapping application. In the case of ground vehicles, [11], considers the functionality constraints of an unmanned ground vehicle (UGV) where the goal is to reach a several target points of a planetary surface in combination with an UAV by minimizing the travelling distance.

Even if the VRP is intrinsically a spatial problem, some applications impose relevant temporal aspects [5]. This paper focuses on SAR operations, where selecting reliable paths is necessary to actually provide timely attention to victims. In fact, one major difference between the objective function in emergency response routing and other routing problems is that the arrival time at victims' locations is more important than the total travel time [2].

Planning node-to-node paths can be the first step in a VRP solution to build the input cost matrix. A time-aware planning can be achieved by considering the traversal times of the edges in topological search graphs [1][8]. Cells in grid representations can be used as nodes in graph-search methods derived from the Dijkstra algorithm [7], but high resolution maps may result in very large search graphs.

In this paper, our goal is to evaluate a search heuristic function based on anisotropic vehicle velocity restrictions for building the cost matrix required for multi-vehicle routing on natural terrain and disaster sites. The heuristic is applied to compute the fastest travel times between every pair of matrix elements by means of a path planning algorithm. The proposed analysis is based on a case study on the ortophotographic-based DEM with a set of target points, where the proposed heuristic is compared against an exhaustive search solution. Thus, the contribution of the paper is not focused on path planning or multi-vehicle routing, but on the construction of the prior cost matrices required to address these problems. In particular, the proposed algorithms allow defining a cost matrix for each UGV.

The remaining of the paper is organized as follows. Section 2.1 offers definitions and the problem formulation. Section 2 introduces the computation of travel time and cost matrices. Section 3 discusses experimental results from a case study for a distribution of victims on a DEM. Section 4 closes the paper with the conclusions.

## 2 Travel Time Matrices and Cost Matrices

This section proposes the computation of a set of travel time matrices $\mathbf{T}$ that can be used to build cost matrices $\mathbf{C}$ for a number of UGVs ($n_{ugv}$). The outline of the approach proposed in this work is given in figure 1. The cost matrix for each vehicle indicates the times required to travel between any two elements of a set of $n_v$ victims. The purpose of these $\mathbf{C}$ matrices is to be suitable for future multi-vehicle routing solutions. First, we describe an exhaustive topological search to find $\mathbf{T}$ by considering anisotropic behavior resulting from terrain relief and

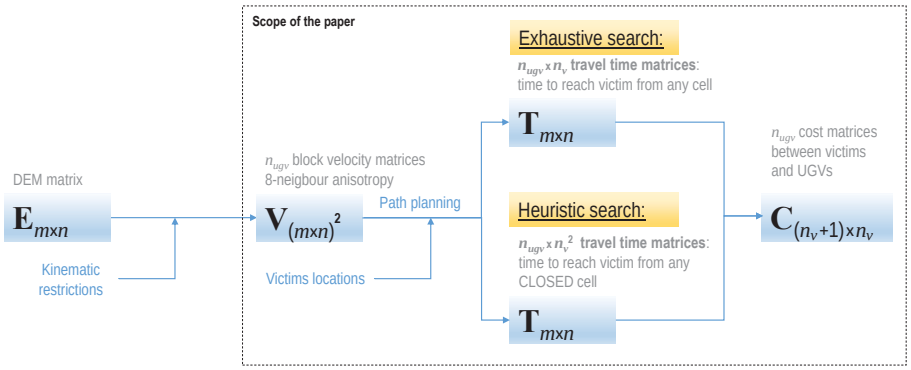



**Fig. 1.** Outline of exhaustive and heuristic computation processes of cost matrices

kinematic restrictions. Then, we introduce a heuristic to efficiently cope with the large search space provided by the grid representation of the DEM.

### 2.1 Problem Formulation

The environment is represented by an $n \times m$ matrix $\mathbf{E}$ with elevation values obtained from its DEM and the resolution $\delta$ is the distance between two contiguous cells.

Traversal velocity matrices $\mathbf{V}$ can be built for each vehicle by considering kinematic restrictions for each cell in $\mathbf{E}$. Thus, $\mathbf{V}$ is a block matrix formed by $m \times n$ sparse submatrices of the same order.

$$\mathbf{V} \in \mathcal{M}_{m \times n} \left( \mathcal{M}_{m \times n} \left( \mathbb{R} \right) \right) \tag{1}$$

Consequently, there is a submatrix for each cell in the DEM. Each submatrix represents the $XY$ traversal velocity from the corresponding cell in the map to its 8-neighbor cells, so the rest of elements of the submatrices is always zero. Each element $v_{\mathbf{c}_1 \mathbf{c}_2}$ represents the $XY$ traversal velocity from the cell $\mathbf{c}_1$ to its 8-neighbor cell $\mathbf{c}_2$ and, therefore, null value expresses the kinematic inadmissibility of UGV for a given traversal displacement.

Travel time matrices $\mathbf{T}$ for a given UGV are built as $n \times m$ matrices where each element $t_{ij}$ is the fastest (as optimized by the search algorithm) travel time from cell $\mathbf{c} = (i, j)$ to the goal cell $\mathbf{c}_g = (i_g, j_g)$.

The cost matrix $\mathbf{C}$, for a given UGV, is an $(n_v + 1) \times n_v$ matrix where the upper $n_v \times n_v$ square submatrix contains the fastest travel times between victims and the last row contains the fastest travel times to each victim from the initial UGV's location.

### 2.2 Exhaustive Search Approach for Computation of Cost Matrix

Algorithm 1 describes the procedure to compute the cost matrix $\mathbf{C}$ for a given UGV using an exhaustive search. This iterative method computes a travel time

**Algorithm 1:** Cost matrix $\mathbf{C}$ through an exhaustive search

**Data:** $\delta$ ... the resolution of environment discretization
**Data:** $\mathbf{V}$ ... the matrix with $XY$ traversal velocities for a given UGV
**Data:** $[\mathbf{c}_{g_1} \dots \mathbf{c}_{g_{n_v}}]$ ... the list of $n_v$ goal cells
**Data:** $\mathbf{c}_s$ ... the start cell for a given UGV
**Result:** $\mathbf{C}$ ... the cost matrix for applying to VRP

1 **foreach** $\mathbf{c}_{g_{k_2}} \in [\mathbf{c}_{g_1} \dots \mathbf{c}_{g_{n_v}}]$ **do**
2 $\quad \mathbf{T} \leftarrow \text{ExhaustiveTravelTimesMatrix}\left(\delta, \mathbf{V}, \mathbf{c}_{g_{k_2}}\right)$
3 $\quad$ **foreach** $\mathbf{c}_{g_{k_1}} \in [\mathbf{c}_{g_1} \dots \mathbf{c}_{g_{n_v}}]$ **do** $\mathbf{C}\big(k_1, k_2\big) \leftarrow t_{\mathbf{c}_{g_{k_1}}}$
4 $\quad \mathbf{C}\big(n_v + 1, k_2\big) \leftarrow t_{\mathbf{c}_s}$

**Algorithm 2:** Exhaustive computation of the travel time matrix $\mathbf{T}$

**Data:** $\delta$ ... the resolution of environment discretization
**Data:** $\mathbf{V}$ ... the matrix with $XY$ traversal velocities for a given UGV
**Data:** $\mathbf{c}_g$ ... the goal cell
**Result:** $\mathbf{T}$ ... the travel time matrix

**Function** `ExhaustiveTravelTimesMatrix` ($\delta$, $\mathbf{V}$, $\mathbf{c}_g$)**:**
1 $\quad$ **forall** $\mathbf{V_c} \in \mathbf{V}$ **do** $t_{\mathbf{c}} \leftarrow \infty$
2 $\quad$ **CLOSED** $\leftarrow \emptyset$ , **OPEN** $\leftarrow \left\{\mathbf{c}_g\right\}$ , $t_{\mathbf{c}_g} \leftarrow 0$
3 $\quad$ **repeat**
4 $\quad\quad \mathbf{c} \leftarrow \underset{\mathbf{c} \in \mathbf{OPEN}}{\operatorname{argmin}} \left\{t_{\mathbf{c}}\right\}$
5 $\quad\quad$ **OPEN** $\leftarrow$ **OPEN** $\setminus \left\{\mathbf{c}\right\}$ , **CLOSED** $\leftarrow$ **CLOSED** $\cup \left\{\mathbf{c}\right\}$
6 $\quad\quad$ **foreach** $\mathbf{c}_k$ **so that** $v_{\mathbf{c}_k\mathbf{c}} \not= 0$ **and** $\mathbf{c}_k \notin$ **CLOSED do**
7 $\quad\quad\quad t_{\mathbf{c}_k} \leftarrow \min\left\{t_{\mathbf{c}_k}\,,\, t_{\mathbf{c}} + \dfrac{\delta \cdot \mathrm{L}_2\big(\mathbf{c}_k, \mathbf{c}\big)}{v_{\mathbf{c}_k\mathbf{c}}}\right\}$
8 $\quad\quad\quad$ **OPEN** $\leftarrow$ **OPEN** $\cup \left\{\mathbf{c}_k\right\}$
9 $\quad$ **until** **OPEN** $= \emptyset$

matrix $\mathbf{T}$ (algorithm 2) for every goal cell and consults in $\mathbf{T}$ the travel time associated with the locations of each other goal cells for assigning to an appropriate element of $\mathbf{C}$ matrix.

### 2.3 Exhaustive Search for Computation of Travel Time Matrix

Algorithm 2, based on the strategy of well-known methods like Dijkstra, its variations (A* or D*) or Fast Marching, describes the procedure to compute the travel time matrix $\mathbf{T}$ for a given UGV. This algorithm does not incorporate heuristics, so the search for fastest travel time is uninformed and exhaustive, which implies a high computational cost, especially for the case of a topological search graph based on a dense grid discretization.

In each iteration, the cells in the environment can be assigned to two sets: **OPEN** and **CLOSED**. The set **OPEN** contains the cells in the environment which have been evaluated. On the other hand, the set **CLOSED** contains the cells $\mathbf{c}$ that have already explored by the algorithm and, therefore, whose travel times $t_{\mathbf{c}}$ have already been computed.

The algorithm initializes: 1) the $\mathbf{T}$ matrix to infinite values, 2) the set **OPEN** containing the goal cell $\mathbf{c}_g = (i_g, j_g)$, and 3) the travel time $t_{\mathbf{c}_g}$ associated with the aforementioned goal cell to zero (lines 1 to 2).

The iterative method explores the cell $\mathbf{c} = (i, j)$ belonging to set **OPEN** with the shortest value of the evaluation function, i.e. shortest travel time $t_{\mathbf{c}}$ (line 4), estimating the travel time from kinematic admissible cells $\mathbf{c}_k$ - i.e. those that have a non-null $XY$ traversal velocity $v_{\mathbf{c}_k\mathbf{c}}$ - which have not yet been explored (line 6). In this way, the travel time is accumulated in the successive iterations by updating $\mathbf{T}$ with the travel time that the vehicle takes from every cell until reaching the goal cell $\mathbf{c}_g$. For computation of cumulative travel time (line 7), the algorithm considers the distance $\delta$ between contiguous cells in the environment, the $XY$ traversal velocity $v_{\mathbf{c}_k\mathbf{c}}$ associated with this displacement, and the Euclidean distance $\mathrm{L}_2$ between the kinematic admissible cell $\mathbf{c}_k = (i_k, j_k)$ and the cell $\mathbf{c} = (i, j)$ extracted from the set **OPEN** in each iteration.

When the evaluation cannot continue (line 9), the travel time matrix $\mathbf{T}$ has been computed for all cells in the environment.

### 2.4 Heuristic Search Approach for Computation of Cost Matrix

Algorithm 3 describes the procedure to compute the cost matrix $\mathbf{C}$ for a given UGV using a heuristic search. This iterative method computes a travel time matrix $\mathbf{T}$ (algorithm 4) for each pair of cells whose represent each pair of victim's locations and consults in each $\mathbf{T}$ the travel time associated with the location of respective start cell for assigning to an appropriate element of $\mathbf{C}$ matrix.

### 2.5 Heuristic Search for Computation of Travel Time Matrix

In order to increase the computational efficiency, a variant of the initially proposed algorithm 2 is presented, which uses of a heuristic to estimate the travel time matrix $\mathbf{T}$. The use of a heuristic aims to reducing the search space by the environment, directing it towards the cell from where the navigation of UGV starts in each case. To do that, algorithm 4 presents modifications of the algorithm 2.

Algorithm 4 establishes two reference cells: 1) location $\mathbf{c}_g$ of the victim to assist, and 2) initial location $\mathbf{c}_0$ of the UGV from where the navigation starts. While the first proposal, presented in the subsection 2.3, only requires cell $\mathbf{c}_g$ associated with the location of each victim, this second one requires a new reference cell $\mathbf{c}_0$. The cell $\mathbf{c}_0$ will be different depending on the element $\mathbf{C}\big(k_1, k_2\big)$ of cost matrix $\mathbf{C}$ that is being estimated, representing the new location from where the UGV navigation will start after assisting each victim.

**Algorithm 3:** Cost matrix $\mathbf{C}$ through a heuristic search

**Data:** $\delta$ ... the resolution of environment discretization
**Data:** $\mathbf{V}$ ... the matrix with $XY$ traversal velocities for a given UGV
**Data:** $[\mathbf{c}_{g_1} \dots \mathbf{c}_{g_{n_v}}]$ ... the list of $n_v$ goal cells
**Data:** $\mathbf{c}_s$ ... the start cell for a given UGV
**Result:** $\mathbf{C}$ ... the VRP cost matrix

1 **foreach** $\mathbf{c}_{g_{k_1}} \in [\mathbf{c}_{g_1} \dots \mathbf{c}_{g_{n_v}}]$ **do**
2  **foreach** $\mathbf{c}_{g_{k_2}} \in [\mathbf{c}_{g_1} \dots \mathbf{c}_{g_{n_v}}]$ **do**
3  $\mathbf{T} \leftarrow \text{HeuristicTravelTimeMatrix}\left(\delta, \mathbf{V}, \mathbf{c}_{g_{k_1}}, \mathbf{c}_{g_{k_2}}\right)$ , $\mathbf{C}\big(k_1, k_2\big) \leftarrow t_{\mathbf{c}_{g_{k_1}}}$
4  $\mathbf{T} \leftarrow \text{HeuristicTravelTimeMatrix}\left(\delta, \mathbf{V}, \mathbf{c}_s, \mathbf{c}_{g_{k_1}}\right)$ , $\mathbf{C}\big(n_v + 1, k_1\big) \leftarrow t_{\mathbf{c}_s}$

**Algorithm 4:** Heuristic computation of the travel time matrix $\mathbf{T}$

**Data:** $\delta$ ... the resolution of environment discretization
**Data:** $\mathbf{V}$ ... the matrix with $XY$ traversal velocities for a given UGV
**Data:** $\mathbf{c}_0$ ... the cell from where the displacement starts
**Data:** $\mathbf{c}_g$ ... the goal cell
**Result:** $\mathbf{T}$ ... the travel time matrix

**Function** `HeuristicTravelTimesMatrix` ($\delta$, $\mathbf{V}$, $\mathbf{c}_0$, $\mathbf{c}_g$)**:**

1  **forall** $\mathbf{V_c} \in \mathbf{V}$ **do** $t_{\mathbf{c}} \leftarrow \infty$ , $\widetilde{t}_{\mathbf{c}} \leftarrow \infty$
2  **CLOSED** $\leftarrow \emptyset$ , **OPEN** $\leftarrow \left\{\mathbf{c}_g\right\}$ , $\widetilde{t}_{\mathbf{c}_g} \leftarrow 0$ , $h_{\mathbf{c}_g} \leftarrow 0$
3  **while OPEN** $\neq \emptyset$ **and** $\mathbf{c}_0 \notin$ **CLOSED do**
4  $\mathbf{c} \leftarrow \underset{\mathbf{c} \in \mathbf{OPEN}}{\operatorname{argmin}} \left\{\widetilde{t}_{\mathbf{c}}\right\}$
5  $t_{\mathbf{c}} \leftarrow \widetilde{t}_{\mathbf{c}} - h_{\mathbf{c}}$
6  **OPEN** $\leftarrow$ **OPEN** $\setminus \left\{\mathbf{c}\right\}$ , **CLOSED** $\leftarrow$ **CLOSED** $\cup \left\{\mathbf{c}\right\}$
7  **foreach** $\mathbf{c}_k$ **so that** $v_{\mathbf{c}_k\mathbf{c}} \neq 0$ **and** $\mathbf{c}_k \notin$ **CLOSED do**
8  $h_{\mathbf{c}_k} \leftarrow \dfrac{\delta \cdot \mathrm{L}_2\big(\mathbf{c}_k, \mathbf{c}_0\big)}{\max\left\{\mathbf{V}\right\}}$
9  $\widetilde{t}_{\mathbf{c}_k} \leftarrow \min\left\{\widetilde{t}_{\mathbf{c}_k}\,,\, t_{\mathbf{c}} + \dfrac{\delta \cdot \mathrm{L}_2\big(\mathbf{c}_k, \mathbf{c}\big)}{v_{\mathbf{c}_k\mathbf{c}}} + h_{\mathbf{c}_k}\right\}$
10  **OPEN** $\leftarrow$ **OPEN** $\cup \left\{\mathbf{c}_k\right\}$

Algorithm 4 describes the heuristic procedure to compute the travel time matrix $\mathbf{T}$ of a given UGV from a initial cell $\mathbf{c}_0$ to a goal cell $\mathbf{c}_g$ in the environment. As in algorithm 2, in each iteration, the cells in the environment can be assigned to two sets: **OPEN** and **CLOSED**.

During the iterations, the method updates values $t_c$ of T (line 5) using two auxiliary matrices $\widetilde{\mathbf{T}}$ and $\mathbf{H}$. For each cell $\mathbf{c}$ belonging to the set **OPEN**, the

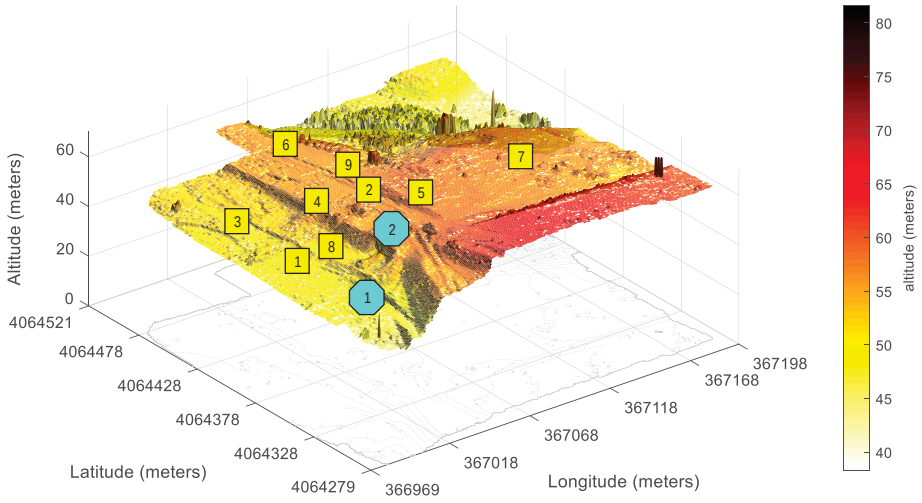



**Fig. 2.** DEM of a real environment. The elevation of the terrain is represented with different red tonalities, where darker tone indicates higher elevation value. The irregular shape in the graph's border represents unmodeled area of the environment.

matrix $\widetilde{\mathbf{T}}$ contains the estimated travel time $\widetilde{t}_{\mathbf{c}}$ from the initial cell $\mathbf{c}_0$ to the goal cell $\mathbf{c}_g$ as it passes through the cell $\mathbf{c}$ (line 9). On the other hand, the matrix $\mathbf{H}$ contains the estimated time $h_{\mathbf{c}}$ from the initial cell $\mathbf{c}_0$ to the cell $\mathbf{c}$ (line 8). The matrix $\mathbf{H}$ is used as a heuristic to accelerate the convergence of the algorithm.

When the search cannot continue or the initial cell $\mathbf{c}_0$ has been explored (line 3), the travel time matrix $\mathbf{T}$ has been computed for a subset of cells in the environment (CLOSED set).

## 3 Experimental Analysis

In this section we compare the performance of the heuristic search algorithm against the uninformed exhaustive search method. The computation has been carried out using Matlab code on a 6-core Intel i7-8750H CPU 2.21 GHz. In particular, we consider a case study with two omnidirectional UGVs ($n_{ugv} = 2$) and nine victims ($n_v = 9$). Figure 2 shows a digital elevation model (DEM) of a real SAR experimentation environment [6] where UGVs and victims are represented as numbered hexagons and squares, respectively. The dimensions of the DEM are 230 $\times$ 243 meters with a resolution of one meter.

The anisotropic traversal velocity matrices, $\mathbf{V}_1$ and $\mathbf{V}_2$, have been defined beforehand for each UGV. These velocity matrices contain the $XY$ traversal velocities for each cell in the environment towards each of its 8-neighbors. The algorithms proposed in this paper are independent of the number of kinematic restrictions considered to build $\mathbf{V}$ for a given application. On an illustrative level, this case study considers a simple set of restrictions, which are given in table 1.

**Table 1.** UGVs' kinematic restrictions

| | UGV-1 | UGV-2 |
|---|---|---|
| Safety radius | 1.4 m | 1.4 m |
| Nominal speed | 0.3 m/s | 0.2 m/s |
| Maximum navigable slope | 20° | 35° |

**Table 2.** Resulting cost matrices $\mathbf{C}$ for the case study (in seconds)

| Victim # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 391 | 147 | 497 | 408 | 686 | 514 | 90 | 541 |
| 2 | 391 | 0 | 523 | 115 | 110 | 300 | 215 | 343 | 155 |
| 3 | 147 | 523 | 0 | 629 | 540 | 819 | 646 | 198 | 674 |
| 4 | 497 | 115 | 629 | 0 | 223 | 259 | 320 | 449 | 147 |
| 5 | 408 | 110 | 541 | 223 | 0 | 401 | 111 | 361 | 226 |
| 6 | 686 | 300 | 819 | 259 | 401 | 0 | 443 | 466 | 268 |
| 7 | 514 | 215 | 646 | 320 | 111 | 443 | 0 | 466 | 268 |
| 8 | 90 | 343 | 198 | 449 | 361 | 639 | 466 | 0 | 494 |
| 9 | 541 | 155 | 674 | 147 | 226 | 175 | 268 | 494 | 0 |
| UGV-1 | 222 | 517 | 367 | 623 | 535 | 813 | 640 | 237 | 668 |

**a)** $\mathbf{C}_1$ for UGV-1

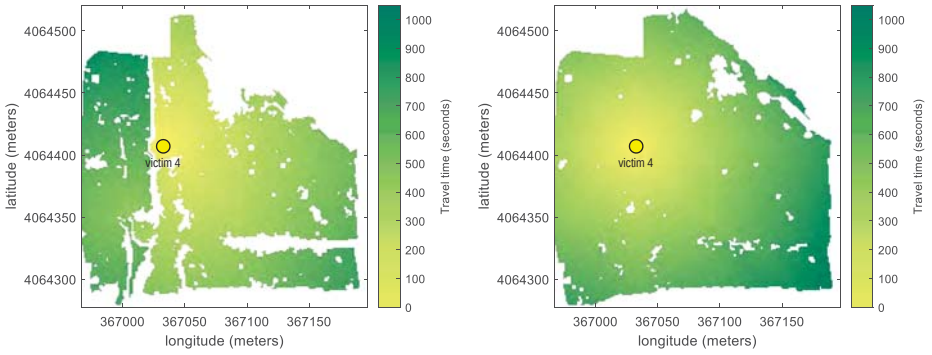
| Victim # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 360 | 217 | 303 | 453 | 678 | 616 | 135 | 511 |
| 2 | 360 | 0 | 353 | 151 | 162 | 450 | 322 | 267 | 232 |
| 3 | 217 | 353 | 0 | 207 | 510 | 462 | 673 | 297 | 407 |
| 4 | 303 | 151 | 207 | 0 | 313 | 388 | 471 | 288 | 213 |
| 5 | 453 | 162 | 510 | 313 | 0 | 602 | 167 | 340 | 339 |
| 6 | 678 | 450 | 462 | 388 | 602 | 0 | 665 | 675 | 263 |
| 7 | 616 | 322 | 673 | 471 | 167 | 665 | 0 | 504 | 402 |
| 8 | 135 | 267 | 297 | 288 | 340 | 675 | 504 | 0 | 462 |
| 9 | 511 | 232 | 407 | 213 | 339 | 263 | 402 | 462 | 0 |
| UGV-2 | 433 | 215 | 492 | 364 | 96 | 657 | 248 | 305 | 399 |

**b)** $\mathbf{C}_2$ for UGV-2

### 3.1 Cost Matrices

Both search methods reach the same values for the cost matrices $\mathbf{C}_1$ and $\mathbf{C}_2$ computed for both UGVs (see table 2). The upper squared submatrices of $\mathbf{C}_1$ and $\mathbf{C}_2$ provide the fastest travel time between the corresponding pair of victim locations whereas the last row indicates the fastest time between the initial position of the UGV and the victims.

The costs between victims depend on the relative positions between them, the terrain relief and the kinematic restrictions contained in the $\mathbf{V}$ matrices. For example, while the cost for UGV-1 between victims #3 and #4 is 629 s, for UGV-2 it is 207 s. This difference can be explained by the limitations of UGV-1 to surpass the steep slope between victim positions (see figure 2), which provokes a longer around path. Conversely, the faster nominal speed of UGV-1 results in a lower cost when travelling between #5 and #7 (i.e., 111 s against 167 s), where both vehicles could travel in a straight line.




**a)** UGV-1 **b)** UGV-2

**Fig. 3.** Examples of graphical representation of travel time matrix **T** to reach victim #4 with exhaustive search

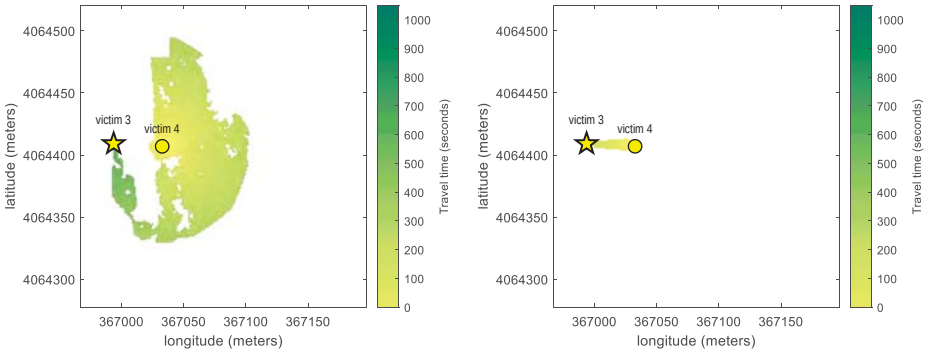
### 3.2 Exhaustive Search Approach

With exhaustive search, **T** matrices need to be computed only once for each victim/UGV pair (see figure 1). For each UGV, $(n_v + 1) \times n_v$ elements of **C** matrix are assigned looking the corresponding values up in the set of **T** matrices. The computation times for obtaining each one of $n_{ugv} \times n_v$ **T** matrices (18 for the study case) range between 57.5 ms and 84.8 ms. The total computational time to obtain the cost matrices for both UGVs has been 1249 ms.

Figure 3 illustrates two examples of travel time matrices **T** computed with the exhaustive search approach. These examples correspond to the computation of **T** to reach victim #4 with each UGV. The victim's location is represented with a yellow circle. Each cell represents the fastest travel time to reach the victim starting from that cell. Travel times are represented with different shades of green. The white cells are those from where the victim cannot be accessed due to UGV's kinematic restrictions (i.e., the search has not produced a solution path).

This difference can be explained by the limitations of UGV-1 to surpass the steep slope between victim positions (see figure 2), which requires a longer around path.

Furthermore, the darkest green cells for UGV-1 (see figure 3.a) represent a low area of the natural terrain from where the vehicle needs to travel around an unsurpassable slope to assist victim #4 (see figure 2). However, UGV-2 has no limitations to surmount most terrain slopes in these environment (with the exception of those in white cells) so travel times are more related to Euclidean distance (see figure 3.b).




**a)** UGV 1 **b)** UGV 2

**Fig. 4.** Examples of graphical representation of travel time matrix **T** from victim #3 to reach victim #4 with heuristic search

### 3.3 Heuristic Search Approach

With heuristic search, **T** matrices need to be computed once for each victim pairs and UGV. The computation times for obtaining each one of $n_{ugv} \times n_v^2$ **T** matrices (162 for the study case) range between 7.9 ms and 26.9 ms. The total computational time to obtain the cost matrices for both UGVs has been 1956 ms.

Figure 4 illustrates two examples of travel time matrices **T** computed with the heuristic search approach. These examples correspond to the computation of **T** to reach victim #4 from #3 with each UGV. The victim's location is represented with a yellow circle and the initial location of displacement is represented with a yellow star. Only the CLOSED cells in algorithm 4 have a green shape. Thus, the white area represents the cells in the environment where the search has not been done and, therefore, the fastest travel time has not been computed. The search space is reduced and the computational time to obtain each individual **T** matrix is decreased. Victim #4 is accessible for UGV-2 with shorter travel time (207 seconds) than UGV-1 (629 seconds), which requires a longer around path due to their kinematic restrictions. Whereas, UGV-2 could travel in a straight line (see figure 4.b).

Figure 5 addresses scalability by presenting the average computational cost (in seconds) to build the cost matrices against the number of victims. In relation to the case study presented, the graph represents the total computation time to construct the cost matrices for both UGVs and a variable number from 1 to 9 victims. For a number of victims less than 6, the heuristic search approach has a total computation time less than the exhaustive search approach because the search space reduction - i.e. reduction of the computation time of the **T** matrices -, in algorithm 4, compensates for the increase in the number of **T** matrices. Conversely, for a 6 or more victims, increasing the number of **T** matrices penalizes the total computation time.

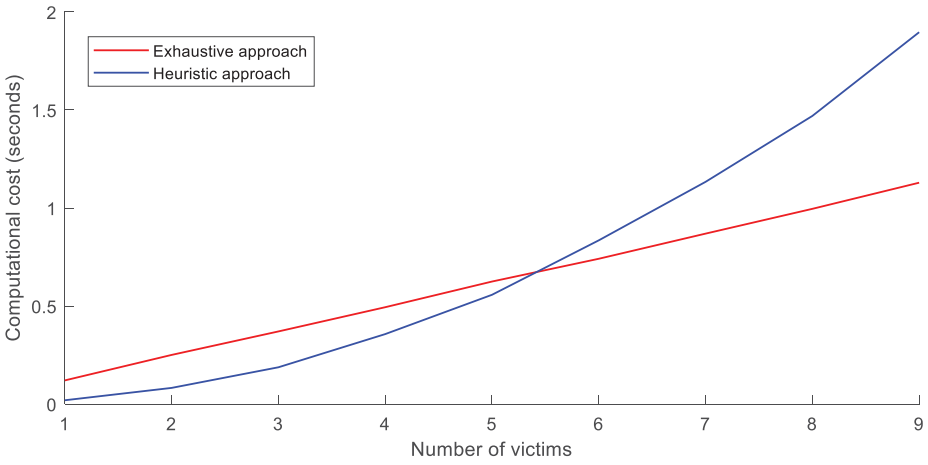



**Fig. 5.** Average computational cost (in seconds) of exhaustive against heuristic approaches for obtaining cost matrices **C** with two UGVs and different numbers of victims

## 4 Conclusions

In multi-robot disaster response on natural terrain, travel time is a crucial cost optimization criterion that may be affected by inclination and other challenging terrain characteristics. In this paper, we have evaluated a search heuristic function based on anisotropic vehicle velocity restrictions with the purpose of building the cost matrices required for multi-vehicle routing on natural terrain and disaster sites. The analysis has been based on a case study on the ortophotographic-based digital elevation model of natural terrain with different target points. The heuristic has been applied to compute the fastest travel times between every pair of matrix elements by means of a path planning algorithm.

Two alternative approaches, i.e., uninformed versus heuristic-based, have been analyzed to compute cost matrices **C** that represent the estimated times reaching every victim from another victim as well as from the UGVs' initial location. Although use of the velocity-based heuristic function aims to reducing the computational cost, the increasing of number of matrices to calculate causes the total computation time to compute the cost matrices to increase as the number of victims grows.

Future work will consider the definition of a VRP solution for multiple heterogeneous vehicles that will use the velocity-based cost matrices.

## Acknowledgments

This work has received funding from the national project RTI2018-093421-B-I00 (Spanish Government), the University of Malaga (Andalucía Tech) and the grant BES-2016-077022 of the European Social Fund.

## References


1. Bae, J., Chung, W.: Heuristics for two depot heterogeneous unmanned vehicle path planning to minimize maximum travel cost. Sensors **19**(11) (2019). DOI 10.3390/s19112461
2. Bruni, M.E., Beraldi, P., Khodaparasti, S.: A fast heuristic for routing in post-disaster humanitarian relief logistics. Transportation Research Procedia **30**, 304–313 (2018). DOI 10.1016/j.trpro.2018.09.033
3. Choi, Y., Chen, M., Choi, Y., Briceno, S., Mavris, D.: Multi-UAV trajectory optimization utilizing a NURBS-based terrain model for an aerial imaging mission. Journal of Intelligent and Robotic Systems: Theory and Applications (2019). DOI 10.1007/s10846-019-01027-9
4. Conesa-Muñoz, J., Pajares, G., Ribeiro, A.: Mix-opt: A new route operator for optimal coverage path planning for a fleet in an agricultural environment. Expert Systems with Applications **54**, 364–378 (2016). DOI 10.1016/j.eswa.2015.12.047
5. Faied, M., Mostafa, A., Girard, A.: Vehicle routing problem instances: Application to multi-UAV mission planning. In: AIAA Guidance, Navigation, and Control Conference (2010). DOI 10.2514/6.2010-8435
6. Fernández-Lozano, J.J., Mandow, A., Martín-Guzman, M., Martín-Avila, J., Serón, J., Martínez, J.L., Gomez-Ruiz, J.A., Socarrás-Bertiz, C., Miranda-Paez, J., García-Cerezo, A.: Integration of a canine agent in a wireless sensor network for information gathering in search and rescue missions. In: IEEE International Conference on Intelligent Robots and Systems, pp. 5685–5690 (2018). DOI 10.1109/IROS.2018.8593849
7. Garrido, S., Moreno, L., Martín, F., Álvarez, D.: Fast marching subjected to a vector field-path planning method for Mars rovers. Expert Systems with Applications **78**, 334–346 (2017). DOI 10.1016/j.eswa.2017.02.019
8. Műhlbacher, C., Gspandl, S., Reip, M., Steinbauer, G.: Adapting edge weights for optimal paths in a navigation graph. Mechanisms and Machine Science **49**, 372–380 (2018). DOI 10.1007/978-3-319-61276-8_41
9. Muñoz, P., R-Moreno, M.D., Castaño, B.: 3Dana: A path planning algorithm for surface robotics. Engineering Applications of Artificial Intelligence **60**, 175–192 (2017). DOI https://doi.org/10.1016/j.engappai.2017.02.010
10. Muñoz-Morera, J., Alarcon, F., Maza, I., Ollero, A.: Combining a hierarchical task network planner with a constraint satisfaction solver for assembly operations involving routing problems in a multi-robot context. International Journal of Advanced Robotic Systems **15**(3) (2018). DOI 10.1177/1729881418782088
11. Ropero, F., Muñoz, P., R-Moreno, M.: TERRA: A path planning algorithm for cooperative UGV-UAV exploration. Engineering Applications of Artificial Intelligence **78**, 260–272 (2019). DOI 10.1016/j.engappai.2018.11.008
12. San Juan, V., Santos, M., Andújar, J.: Intelligent UAV map generation and discrete path planning for search and rescue operations. Complexity **2018** (2018). DOI 10.1155/2018/6879419
13. Gonzalez-de Soto, M., Emmi, L., Garcia, I., Gonzalez-de Santos, P.: Reducing fuel consumption in weed and pest control using robotic tractors. Computers and Electronics in Agriculture **114**, 96–113 (2015). DOI 10.1016/j.compag.2015.04.003
14. Wang, H., Zhang, H., Wang, K., Zhang, C., Yin, C., Kang, X.: Off-road path planning based on improved ant colony algorithm. Wireless Personal Communications **102**(2), 1705–1721 (2018). DOI 10.1007/s11277-017-5229-5