Continuously Informed Heuristic A* - Optimal path retrieval inside an unknown environment

Athanasios Ch. Kapoutsis, Christina M. Malliou, Savvas A. Chatzichristofis and Elias B. Kosmatopoulos

***Abstract*— This paper deals with the problem of retrieving the optimal path between two points inside an unknown environment, utilizing a robot-scouter. The vast majority of the path planning frameworks for an unknown environment focuses on the problem of navigating a robot, as soon as possible, towards a pre-specified location. As a result, the final followed path between the start and end location is not necessarily the optimal one, as the objective of the robot at each timestamp is to minimize its current distance to the desirable location. However, there are several real-life applications, like the one formulated in this paper, where the robot-scouter has to find the minimum path between two positions in an unknown environment, which is going to be used in a future phase. In principle, the optimal path can be guaranteed by a searching agent that adopts an A*-like decision mechanism. In this paper, we propose a specifically-tailored variation (CIA*) of the A* algorithm to the problem in hand. CIA* inherits the A* optimality and efficiency guarantees, while at the same time exploits the learnt formation of the obstacles, to on-line revise the heuristic evaluation of the candidate states. As reported in the simulation results, CIA* achieves an enhancement in the range of 20-50%, over the typical A*, on the cells that have to be visited to guarantee the optimal path construction. An open-source implementation of the proposed algorithm along with a Matlab GUI are available[1].**

I. Introduction

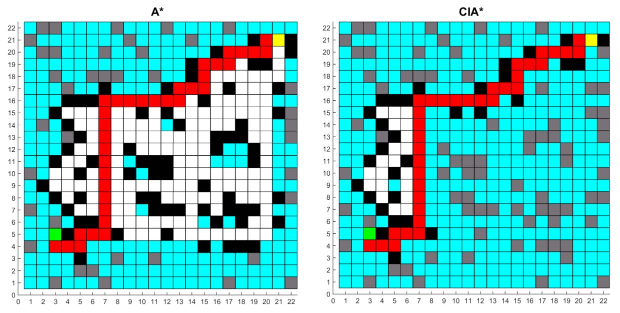
Retrieving the minimum feasible path between two points inside a completely unknown terrain is not trivial. For the problem in hand, it is assumed that a robot-scouter is available and it has means for solving the underlying pose estimation problem [1] and the estimation of obstacles in close proximity [2]. Therefore, the pursued problem can be reduced to *ensure the optimal path retrieval between two given locations* by *exploring the minimum portion of the environment*. In a nutshell, we seek to answer the question: What is the minimum course of action for the searching robot, so as the minimum feasible path between two locations to be retrieved?

This problem encountered in several real-world applications where 1) it is worthwhile to search for the optimal path in order to utilize it in the future 2) a preparatory stage along with the required resources are available. Two indicative real-life instances where this problem may find direct application in, are listed below:

- A physical disaster has occurred and a big portion of the road network has been destroyed. In such situation, it is considered vital to find the optimal path, so as to establish efficient supply lines among sub-regions which are in need.
- A military formation of troops with multiple heavy vehicles is asked to traverse an unexplored area, so as to take position for an upcoming battle. It is a common practice to utilize one or more scouts to find the shortest safe path for the troops to pass through.

The authors are with the School of Electrical and Computer Engineering, Democritus University of Thrace, Xanthi, Greece 67100 (email: {akapouts,cmalliou,schatzic,kosmatop}@ee.duth.gr)

[1]https://github.com/athakapo/CIAstar



(a) Typical A* with constant heuristic (b) Proposed Continuously Informed heuristic A*

Fig. 1. Comparison in terms of expanded cells. The retrieved path is depicted with red color, while the white cells denote the visited area by the robot-scouter.

Typical approaches to the specific area of research can be separated into two categories. The first class consists of A* variances that calculate the minimum path inside an a priori known map. Therefore, the search is performed off-line and the different variants either exploit problem-specific characteristics, e.g. [3], or provide additional features, such as "anytime" solutions [4] [5]. The other side of the spectrum encloses approaches that navigate a moving robot between two locations inside an unknown graph. The specific problem formulation addresses the problem of guiding the robot to the goal position as quickly as possible [6] [7], without necessarily retrieving the optimal path from the starting location.

However, to the authors' best knowledge, very few publications can be found available in the literature that address the pursued problem set-up. For example, the authors of [8] proposed an algorithm (PHA*) that visits all the A*-derived *mandatory* positions in such a way so as to minimize the intermediate steps between them. While, the current paper focuses on the problem of minimizing the *mandatory* positions that have to be visited in order to retrieve the minimum path, neglecting the task of navigating the robot to these positions. Nevertheless, these two tasks are largely orthogonal, i.e. any algorithm evaluated in the present paper can be straightforwardly combined with PHA* (or any other similar) approach.

In this paper, we propose the Continuously Informed Heuristic A* (CIA*). Our work relies on the A* algorithm [9] and inherits the guarantees about the optimality of the retrieved path. CIA*, however, deviates from the original A* on the calculation of the heuristic distance-to-the-goal estimation. Instead of utilizing an underestimated constant heuristic function, such as in the original A*, CIA* essentially revises the heuristic evaluations based on the imposed limitations of the acquired knowledge about the obstacles' formation.

Figure 1 outlines a performance comparison between the CIA* and the traditional A*. Yellow and green cells denote the fixed starting and ending cell. The summation of white and red cells corresponds to the visited locations that each algorithm requires in order to conclude about the optimal path. Moreover, black, cyan and grey cells denote the discovered obstacles, the unexplored empty cell and the unknown occupied cells, respectively. Although both the algorithms retrieved the same minimum feasible path (marked with red color), CIA* achieved 67% improvement on the number of visited cells by the robot. More precisely, traditional A* required 177 cells visitations to accomplished the desirable task, while the effect of continuously informed heuristic reduced it to only 57 cells[2]. Overall (see *Simulation* section), the utilization of the proposed algorithm achieves more than 20% improvement, as compared to the typical A* algorithm, in terms of the visited cells by the robot-scouter, while at the same time the minimum path retrieval is always guaranteed (Lemma 1). Furthermore, the achieved performance is comparable with the weighted A*[10], which speeds-up the search procedure in the expense of the optimality in the returning path (bounded sub-optimal variance of A*).

II. Problem Statement and Terminology

It is assumed that the operational environment is constrained within a rectangle, which has been discretized into identical uniform-cost grid cells:

$$\mathcal{U} = \{x, y : x \in [1, rows], y \in [1, cols]\}$$

Each cell may be *occupied* (by some kind of obstacle) or *traversable* and its state is considered a priori unknown. The occupied cells cannot be traversed by the robot-scouter and cannot be included in the returning path. The set of occupied cells (obstacles) is represented as:

$$B = \{(x, y) \in \mathcal{U} : (x, y) \text{ is } \textit{occupied}\}$$

During the search stage, the robot is able to perceive the state (traversable or occupied) of an ith cell only if it has visited at least one *adjacent* cell. Two cells (x_i, y_i) and (x_j, y_j) are considered *adjacent* if:

$$\|x_i - x_j\| + \|y_i - y_j\| \leq 1 \qquad (1)$$

[2]The interested readers are referred to http://tinyurl.com/CIA-star-comp where they can watch a video about the evolution on the expanded cells for this experiment.

This constraint is in-line with the operation of real-life robots, which are able to learn the structure of an unknown environment by their exteroceptive sensors [11].

At each timestamp, the robot chooses (expands) a cell, from the list of previously discovered unoccupied cells, to continue its search. More precisely:

Definition 1: As <u>*expanded*</u> *cell is denoted any cell, where the robot has to deploy its sensors to learn about the adjacent morphology.*

Definition 2: As <u>*successors*</u> *of an expanded cell are denoted all the adjacent cells (1) which are not occupied by an obstacle.*

The objective of the search is to find the minimum path between (x_s, y_s) and (x_g, y_g).

Definition 3: As <u>*path*</u> *between the cells* (x_s, y_s) *and* (x_g, y_g) *is considered every sequence of cells*

$$S_N = \{(x_1, y_1), \ldots, (x_{N-1}, y_{N-1})\}$$

where the following constraints are hold

- $(x_i, y_i) \in \mathcal{U} \setminus B,\ \forall\, i \in [1, \ldots, N-1]$
- (x_i, y_i) and (x_j, y_j) are *adjacent* according to (1), $\forall\, i = j + 1$, including s and g cells

Definition 4: A path S_N *between the cells* (x_s, y_s) *and* (x_g, y_g) *with length* N*, is considered* <u>*optimal*</u> *if* $N \leq N'$*, where* N' *is the length for every other possible path that connects* s *and* g*.*

The problem pursued in this paper is to design an algorithm that is able to guarantee the optimal path retrieval, by performing the minimum number of robot movements. Using the previously defined terminology, the robots' movements can be expressed as the number of *expanded* cells[3]. Thus, the problem is deduced to minimize the number of *expanded* cells from which the optimal path can be tightly derived.

III. A* - Motivation

The problem as formulated in the previous section might be considered as a special case of the problem of finding shortest paths in arbitrary graphs, where the A* algorithm has been proven to be the optimal strategy [12]. More precisely,

Theorem 1: A* is optimally efficient for <u>a given admissible heuristic function</u>, in the sense that <u>no other algorithm</u> can guarantee to expand fewer nodes than A*, without running at risk of missing the optimal solution.

For the sake of simplicity, the ith cell's coordinates (x_i, y_i) are represented by n_i. A heuristic function h is called *admissible* if it always underestimates the path to the goal:

$$h(n) \leq h^*(n),\ \forall n \in \mathcal{U} \qquad (2)$$

[3]The minimum number of robot's movements – apart from the expanded cells – is also correlated with the distance between two consecutive expanded cells. For example, it may be more beneficial for the robot to explore a small area around its current cell before it continues with the next expanded cell. In principle, these are two orthogonal tasks that can be performed at the same time. In the present paper, we retain the performance evaluation only to the number of expanded cells.

where $h^*(n)$ denotes the actual distance between the cell n and the goal. Furthermore, as the chosen heuristic h approximates the real distance function, the number of expanded cells also approximate the number of cells that constitute the *optimal* path (Definition 4) [13]. In other words, if the searching robot had a heuristic function that accurately estimates the real distance between each cell and the goal position, it would expand only the cells that constitute the minimum path.

As a result, one could design an exploration algorithm for the problem in hand, where the searching robot would visit (*expand*) the cells as calculated from the A* mechanism. This procedure would definitely provide the optimal path, by performing the minimum amount of operations. By Theorem 1 it is clear that any attempt to construct some other searching algorithm would be a waste.

However, as defined in the previous section, in this paper we tackle the problem of finding the optimal path inside an unknown, physical environment utilizing the minimum number of robot's operations. The kind of operations fall into two different classes: the computational operations needed for the calculation of the next movement and the movement itself. Contrary to the usual off-line problem, the real environment requirement re-balances the cost of each operation. More precisely, each time the searching robot expands a new cell, it has to spend actual time and energy to reach it. In contrast, the operations involved in the calculation of these expanded cells cost only clock's cycles inside the robot's processor. Therefore, as stated in the problem formulation section, the main objective is to reduce the number of expanded cells giving less priority to the needed computation time for such selection mechanism.

In a nutshell, the proposed methodology utilizes the basics of A* methodology, but in every timestamp, it carefully revises the heuristic function, in order to include all the so-far learnt terrain characteristics. More details about this update, along with the preservation of the A* optimality guarantees are discussed in the following section.

IV. Continuously Informed Heuristic A*

The basic idea behind the proposed Continuously Informed heuristic A* (CIA*) algorithm relies on the fact that the heuristic evaluation can be securely revised, utilizing the acquired knowledge about the obstacles' locations. This information about the obstructed cells is "propagated" back to all successors, and is exploited to update their distance estimation to the goal. This procedure may severely increase the over-optimistic constant heuristic, leading to a major decrease in the number of expanded cells.

To highlight the necessity of such an approach, figure 2 presents a representative example, where the overall search procedure could be enhanced by a further study on the heuristic values. Following the introduced notation from the introduction, white, black, gray and cyan cells denote the currently expanded, discovered obstacles, unknown obstacles and undiscovered empty cells, respectively. Start and goal cells are denoted by yellow and green cells respectively. Until the depicted timestamp, the typical A* works efficiently, as the information about the obstacles formation cannot affect the estimated path to the goal. However, at the specific timestamp, the discovered obstacles' formation rises explicit limitations regarding the best path that can be achieved by a sub-set of all the discovered successors. Concretely, as it is illustrated in the zoomed areas in figure 2, the most appealing choice (to continue the searching procedure) for the A* with constant admissible heuristic, would be the upper one cell ($f = g + h = 34$). Based on the already-explored obstacles' layout, however, the heuristic value of this cell ($h = 15$) is now infeasible, and in fact cannot be less than 17. Such an update to the heuristic estimation, leads to the selection of the lower cell, as now this cell has the same f value and the ties are broken in favor of the one with the minimum estimation to the goal (like typical A*). Such an alternated heuristic update mechanism can significantly reduce the number of expanded cells, as depicted in the comparison of figure 1. Therefore, an approach which identifies such cases and carefully revises the heuristic values can be utilized in cases where the optimal path between two points inside an unknown environment, have to be returned. This idea is exploited by the proposed CIA* which is described in the next sub-section.

A. Algorithm Description

The complete CIA* algorithm is presented in Algorithms 1 & 2, while the two major pillars that characterize the CIA* update procedure are analyzed in the following paragraphs.

The first pillar, which is the main CIA* contribution relies on the calculation of the heuristic value (Algorithm 2). To achieve this calculation, additionally to the basic A* data structures, a set of all the so-far discovered obstacles is also maintained. This set, together with the CLOSED set (set of expanded nodes), is utilized to define the BLOCKED set (Alg. 2 line 2). In this phase, CIA* examines if the formation of BLOCKED set further restricts the minimum path that can be found between a cell from the OPEN set and the goal. To do so, an iterative procedure is utilized where the minimum sized rectangle is calculated, in which, the corresponding cell from the OPEN set and the goal cell are connected. More precisely, the heuristic update procedure (figure 3) first annotates all the cells that belong to the BLOCKED set as occupied (Alg. 2 line 6) and then gradually increases the rectangle defined by the goal and the cell. This process terminates when the two cells marked as connected or when the rectangle reaches the maximum user-defined bound (denoted as r). In each step of this procedure, the information about the connectivity of the two cells can be derived by the connected components algorithm [14]. The following sub-section is devoted to justifying this selection. Figure 3 gives an example of the proposed iterative update procedure. More precisely, figure 3 (b) depicts the case with offset 0, where there is no path between the two cells. However, when the rectangle is increased by an offset of 1 cell (figure 3 (c)), the two cells are now connected. Therefore, the heuristic value of this cell can be increased by a *correction factor* which is

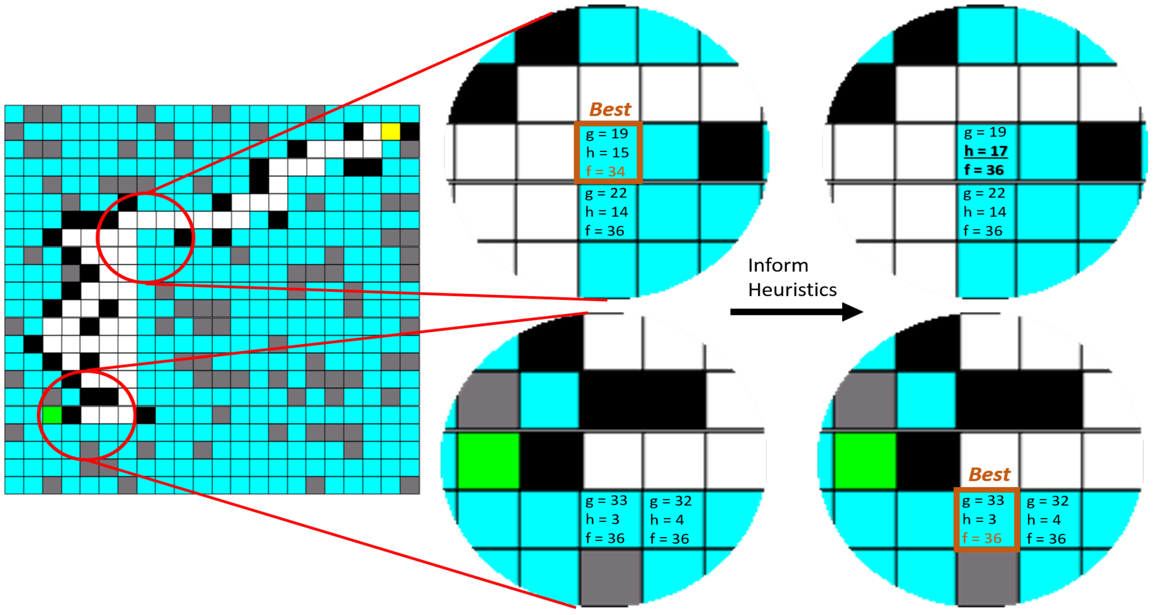


Fig. 2. Key functionality of Continuously Informed heuristic A*

defined as two times the size of this offset (Alg. 2 lines 8 & 12). Conceptually, the *correction factor* represents the value by which the heuristic value can be further increased from the constant A* estimation (Manhattan distance), without jeopardizing the admissibility property (eq. 2).

The second pillar aims to tackle problems regarding the cells evaluation, which may be induced by the utilization of the continuously informed heuristic function. In essence, any comparison between cells that have been previously evaluated with different heuristic functions would be unfair and could jeopardize the admissibility condition. In order to tackle the aforementioned reference problems, CIA* revises the heuristic value (Alg. 2) of the minimum extracted cell, so as to be up-to-date with the current knowledge about the obstacles' locations. The updated cell is repositioned back to the OPEN set and the next cell with the minimum value is extracted. The aforementioned process is repeated (Alg. 1 lines 9-13) until the currently extracted cell's heuristic evaluation has already been changed or it cannot further be affected by the new occupied cells' information. In other words, the proposed algorithm does not explicitly revise all the cells that are affected from the current obstacles layout. This feature provides a twofold efficiency guarantee. On one hand, the algorithm does not have to search for the exact cells that are affected, as it examines only the ones which are extracted from the OPEN set (with $\mathcal{O}(1)$ complexity, if it is implemented as a Fibonacci heap). On the other hand, it avoids updates on cells which their evaluations are not "close" to the current minimum value.

B. Study on the heuristic update component

This sub-section presents a study about the component responsible for the update on the heuristic value incorporating obstacles' information. In general, any search algorithm can serve as heuristic value update mechanism, e.g. Dijkstra's algorithm [15]. However, Dijkstra's algorithm does not exploit geometrical features of the problem, rendering its use prohibitive in terms of time consuming. A more appealing algorithm would be the typical A* approach (specifically-tailored to this sub-problem), which guides the search utilizing the geometry of the environment. Even though A* performs significantly better than the Dijkstra's algorithm, a significant overhead is added to the overall computation time. This result, nevertheless, is reasonable as the A* calculates the path between the two given cells. However, the path itself is not useful at this step, as only the length of such a path is enough to update the heuristic estimation. Therefore, to decrease the time needed for the heuristic update, we adopt the connected component algorithm. In the rest of this sub-section, we present an indicative comparison study between

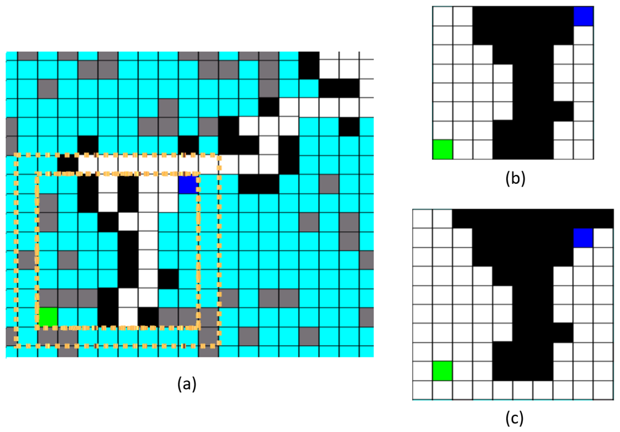


Fig. 3. Sub-figure (a) illustrates the terrain with the currently known information. When searching with offset 0, there is no path between the two cells (sub-figure b). In contrast, when offset is increased by 1, the two cells located at the same connected component (sub-figure c).

Algorithm 1 Continuously Informed A* Algorithm

```
Input: $n_g$, $n_s$, boundaries, $r$
Output: Optimal Path to the $n_g$
 1: OPEN = $\emptyset$, CLOSED = $\emptyset$, OBST = $\emptyset$
 2: Insert $\langle n_s, \text{InformedHeuristic}(n_s)\rangle$ in the OPEN set
 3: loop
 4:   if OPEN set is empty then
 5:     return failure
 6:   end if
 7:   Extract the cell $n_e$ with the minimum $f_e = g_e + h_e$ value from the OPEN set
 8:   $h_u := \text{InformHeuristic}\,(n_g, n_e, \text{CLOSED}, \text{OBST}, r)$ Compute its updated heuristic
 9:   while $h_e < h_u$ do
10:     Insert $\langle n_e, g_e + h_u\rangle$ to OPEN set
11:     Extract the cell $n_e$ with the minimum $f_e = g_e + h_e$ value from the OPEN set
12:     $h_u := \text{InformHeuristic}\,(n_g, n_e, \text{CLOSED}, \text{OBST}, r)$ Compute its updated heuristic
13:   end while
14:   Insert $n_e$ to the CLOSED set
15:   if $n_e = n_g$ then
16:     return pathTo($n_e$)
17:   end if
18:   Expand $n_e$ and calculate all its valid *Successors*
19:   Update OBST based on the obstacles found in $n_e$ neighborhood
20:   for all $n_s$ in *Successors* do
21:     if $n_s$ is not inside the CLOSED set then
22:       if $n_s$ is inside the OPEN set then
23:         Keep the entry with the shortest distance from the start
24:       else
25:         Insert to OPEN set $\langle n_s, g_e + 1 + \text{InformHeuristic}\,(n_g, n_s, \text{CLOSED}, \text{OBST}, r)\rangle$
26:       end if
27:     end if
28:   end for
29: end loop
```

Algorithm 2 Continuously Informed Heuristic

```
 1: function InformHeuristic($n_g, n, \text{CLOSED}, \text{OBST}, r$)
 2: BLOCKED:= CLOSED $\cup$ OBST
 3: $i := 0$
 4: while $i \leq r$ do
 5:   Define rectangle $d_i$ between $n_g$ and $n$ with offset $i$
 6:   Mark-up any cell of $d_i$ that belongs to BLOCKED as occupied
 7:   if $n$ and $n_g$ belong to the same connected component in $d_i$ then
 8:     return ManhattanDist($n_g, n$) $+ 2i$
 9:   end if
10:   $i := i + 1$
11: end while
12: return ManhattanDist($n_g, n$) $+ 2(r + 1)$
13: end function
```

connected components algorithm and A* methodology, for the problem of updating the heuristic value.

To properly assess the effect of the algorithms in terms of computation time needed, we consider the following 3 scenarios. Every scenario was repeated 1000 times to average out the effect of the initial conditions and was performed on a 2.7 GHz Intel Core i5, with 8 GB RAM, running macOS Sierra. In the first scenario, we consider a grid of 50x50 cells with no obstacles. For this scenario, the overall average execution time for A* case was 51.94 sec, while the version with connected components algorithm needed only 0.50 sec. To study the effect of obstacles on the two evaluated methodologies, we augment the initial environment with 500 obstacles. The same trend is observed in this second scenario (50x50 grid with 500 obstacles), where the execution time becomes 88.49 sec for the A* version and 0.69 sec for the version with connected components algorithm. Finally, in the third scenario, the grid was changed to 100x100 cells with 0 obstacles to studying the scalability of each module. Again, the connected components version of the algorithm outperformed the one with A*, as the first needed – on the average – only 1.5 sec to execute one experiment, while the second one needed 331.63 sec. The purpose of this paragraph was to just evaluate the performance of the sub-component which is responsible for the heuristic update, while an extensive analysis for the proposed algorithm as a whole is presented in section V.

C. Path Optimality

The A* algorithm guarantees the optimal path construction, under the condition that the chosen heuristic function is admissible (eq. 2).

Lemma 1: CIA* algorithm inherits A* guarantees regarding the optimal path construction.

Proof: CIA* algorithm, as described in Algorithms 1 and 2, constructs a different heuristic function h_i for each ith timestamp. In a nutshell, the heuristic function's update can be described as $h_{i+1} = h_i + \nu_i$. In order to retain the admissibility of the heuristic function, the ν_i should be always $\nu_i \leq (h^* - h_i)$. Apparently, in the case where the obstacles' information does not further restrict the path length to the goal, then $\nu_i = 0$. Whenever the path between the cell and the goal is further restricted by the discovered obstacles, the ν_i value will be increased. Let's assume that at some point $\nu_i > (h^* - h_i)$ and therefore the optimality of the constructed path is jeopardized. The acquisition of such a value for the ν_i is equivalent with an over-increase of the searching rectangle (Algorithm 2). Such an increase is possible only if the connected components algorithm fails to identify that with the current obstacles formation the two cells are connected. However, this is a contradiction, as the connected components algorithm guarantees the proper labeling of all the distinct components by its construction (see [16], [14]). Therefore, the initial assumption, that $\nu_i > (h^* - h_i)$, must be false and the CIA* heuristic functions is constantly admissible:

$$h_i \leq h_{i+1} \leq h^*$$

■

V. Simulation Results

This section presents simulation results using the proposed CIA* algorithm. A MATLAB-based simulator has been developed, where the following simulation environment was constructed:

- 4 different terrain sizes: [50×50], [100×100], [150×150] and [200×200] cells.
- For each terrain size, 5 different percentage levels of occupied cells were employed, varying from 0% to 30% of the available cells.

The proposed approach (CIA*) is evaluated in every experimental instance (size, obstacles, initial and goal cell) against 2 different algorithms from the literature. As the baseline case, an algorithm which expands the cells as calculated from the traditional A* implementation was utilized. The resulted path is by definition the minimum possible. This case "acts" as a representative for all the algorithms that preserving the optimality of the retrieved path by exploiting A* mechanism (e.g. [8]).

A variation of A*, which is called weighted A*, was utilized as second evaluation methodology. For the sake of improvement in the performance, weighted A* violates the admissibility condition of the A*, by multiplying the heuristic evaluation by a term w. This behavior increases the priority to the cells which are closer to the goal, speeding-up the searching procedure. Half a century after its original publication, weighted A* remains the best-performing algorithm for general-purpose bounded suboptimal search [17].

Table I summarizes the simulation results. The comparison among the three algorithms is based on the number of expanded cells and the deviation from the minimum path. At first, the number of the expanded cells [Expanded] is depicted for each algorithm along with the corresponding standard deviation (SD). Additionally, for CIA* and weighted A*, two more fields were introduced: i) the [Encahnc.] field which depicts the percentage enhancement from the A*, in terms of expanded cells and ii) the [Dev_Opt] field which presents the discrepancy from the minimum path (as returned from the A*). In order to average out the effect of the initial conditions (obstacles, start and goal cell) from the produced results, we repeated every experimental instance 1000 times, setting randomly the obstacles locations along with the initial and goal cell.

As reported in table I, when the number of obstacles is equal to zero, all the three algorithms for all the evaluation scenarios, terminate having expanded the same number of expanded cells and returning the same optimal path. This behavior was quite expected as both the CIA* algorithm and the weighted A*, deviate their functionality from the traditional A* regarding the obstacles handling.

As the number of obstacles is increased, the CIA* achieves a significant improvement in the number of expanded cells, in contrast to the A*, while at the same time, the resulting path is always the minimum possible. The enhancement in the performance increases as the number of grid size and/or obstacles grows. This should be an anticipated behavior, as the CIA* improvement is directly derived from the obstacles patterns (Algorithms 1 and 2). On the other hand, weighted A* achieves also a remarkable performance, in terms of expanded cells. In this algorithm, the increase in the performance is inversely proportional to the number of obstacles. This is also an anticipated behavior, as the weighted A* "runs" more quickly to the goal cell. Thus, the less obstacles the weighted A* meets, the fewest cells are expanded.

Although the improvement in performances of weighted A* and CIA* is similar, they differ in a very crucial aspect. Weighted A* does not always return the minimum path and the average deviation from the optimal path (in terms of cells) is exhibited in column [Dev_Opt]. In essence, weighted A* performance should not be compared with A* and CIA*, as its produced path is not acceptable for the problem in hand (Section II). Nevertheless, weighted A* is indeed included in the simulation table I, to highlight the compromises in [Dev_Opt] that have to be made in order to meet the CIA* performance improvements.

Finally, regarding the computational complexity, CIA* performs multi-times the operations needed by the A*, as it may calculate one cell's heuristic value multiple times. Therefore, for the case where all the obstacles' locations are a priori known - typical problem formulation for A* - the plan should be designed off-line, utilizing A* efficiency guarantees (Theorem 1). However, the performance improvements, especially in dense terrains, were so dramatical that in some instances, CIA* overcomes A* even in terms of absolute time (no distinction between the robot operations - Section III). In other words, there are cases, where it would be more beneficial to utilize CIA* methodology, in order to produce off-line optimal paths.

VI. Conclusions

A new algorithm for the problem of finding a safe/obstacle-free path inside completely unknown environments utilizing a searching robot has been introduced. The pursued problem can be translated into the optimal path retrieval between two points, by employing the minimum course of action for the moving agent. The proposed methodology utilizes the backbone of the well-known A* algorithm to retain its optimality and efficiency guarantees. Additionally, CIA* exploits the gathered information regarding the obstacles layout, to reduce the number of expanded cells from 20% to 50%. This performance improvement does not come at an expense of the optimality of the returning path, as CIA* is designed to maintain the A* admissibility condition for the heuristic function. The above feature is of paramount importance in many real-life applications, where the optimal path is a prerequisite and there is only limited time or energy for a searching robot to retrieve it.

Acknowledgment

This paper has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme under grant

Terrain	Obstacles - Occupied	A*	CIA* (Proposed)			Weighted A*		
[Rows x Cols]		Expanded (SD)	Expanded (SD)	Enhanc.	Dev_Opt	Expanded (SD)	Enhanc.	Dev_Opt
[50x50]	0 - 0%	35.07 (16.48)	35.07 (16.48)	0%	0	35.07 (16.48)	0%	0
[50x50]	375 - 15%	71.04 (67.08)	57.24 (41.46)	19.43%	0	58.27 (45.73)	17.98%	0.17
[50x50]	500 - 20%	87.00 (98.61)	62.98 (50.22)	27.61%	0	72.52 (60.85)	16.65%	0.14
[50x50]	625 - 25%	115.00 (115.96)	76.82 (67.44)	33.20%	0	99.96 (92.56)	13.08%	0.08
[50x50]	750 - 30%	161.46 (163.55)	104.96(103.64)	34.99%	0	146.81 (146.50)	9.07%	0.06
[100x100]	0 - 0%	68.67 (32.52)	68.67 (32.52)	0%	0	68.67 (32.52)	0%	0
[100x100]	1500 - 15%	216.53 (279.06)	148.78 (132.96)	31.29%	0	142.21 (114.20)	34.32%	0.72
[100x100]	2000 - 20%	283.66 (320.82)	187.10 (175.49)	34.04%	0	195.99 (167.39)	30.91%	0.71
[100x100]	2500 - 25%	357.85 (362.05)	210.64 (200.86)	41.14%	0	253.43 (227.65)	29.18%	0.49
[100x100]	3000 - 30%	526.58 (519.10)	294.61 (296.39)	44.05%	0	397.19 (384.78)	24.57%	0.31
[150x150]	0 - 0%	98.75 (50.45)	98.75 (50.45)	0%	0	98.75 (50.45)	0%	0
[150x150]	3375 - 15%	466.00 (737.18)	296.13 (283.70)	36.45%	0	239.90 (197.02)	48.52%	1.80
[150x150]	4500 - 20%	569.09 (661.33)	334.06 (317.45)	41.30%	0	326.82 (280.25)	42.57%	1.42
[150x150]	5625 - 25%	804.96 (834.00)	439.56 (470.08)	45.39%	0	504.16 (473.85)	37.37%	1.18
[150x150]	6750 - 30%	1105.32 (1049.59)	574.93 (549.11)	47.99%	0	749.85 (738.71)	32.16%	0.67
[200x200]	0 - 0%	131.95 (64.96)	131.95 (64.96)	0%	0	131.95 (64.96)	0%	0
[200x200]	6000 - 15%	767.56 (977.02)	479.94 (501.27)	33.86%	0	345.89 (289.06)	54.94%	2.77
[200x200]	8000 - 20%	1028.96 (1262.72)	589.72 (633.33)	42.69%	0	525.11 (494.75)	48.97%	2.44
[200x200]	10000 - 25%	1394.96 (1451.76)	688.36 (741.58)	50.65%	0	761.38 (742.04)	45.42%	1.88
[200x200]	12000 - 30%	1891.45 (1687.14)	922.32 (948.57)	51.23%	0	1201.12 (1142.85)	36.50%	1.30

TABLE I
SIMULATION RESULTS

agreement no 740593 (ROBORDER) and by the European Commission (FIRE+ challenge, Horizon 2020) that aims to provide for research, technological development and demonstration under grant agreement no 645220 (RAWFIE).

REFERENCES

[1] S. Thrun *et al.*, "Robotic mapping: A survey," *Exploring artificial intelligence in the new millennium*, vol. 1, pp. 1–35, 2002.

[2] A. Discant, A. Rogozan, C. Rusu, and A. Bensrhair, "Sensors for obstacle detection-a survey," in *2007 30th International Spring Seminar on Electronics Technology (ISSE)*. IEEE, 2007, pp. 100–105.

[3] D. D. Harabor, A. Grastien *et al.*, "Online graph pruning for pathfinding on grid maps." in *AAAI*, 2011.

[4] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime search in dynamic graphs," *Artificial Intelligence*, vol. 172, no. 14, pp. 1613–1643, 2008.

[5] J. Van Den Berg, R. Shah, A. Huang, and K. Goldberg, "Ana*: anytime nonparametric a*," in *Proceedings of Twenty-fifth AAAI Conference on Artificial Intelligence (AAAI-11)*, 2011.

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

[7] S. Koenig and M. Likhachev, "D*lite," in *Eighteenth National Conference on Artificial Intelligence*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 2002, pp. 476–483. [Online]. Available: http://dl.acm.org/citation.cfm?id=777092.777167

[8] A. Felner, R. Stern, A. Ben-Yair, S. Kraus, and N. Netanyahu, "Pha*: finding the shortest path with a* in an unknown physical environment," *Journal of Artificial Intelligence Research*, pp. 631–670, 2004.

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

[10] R. Ebendt and R. Drechsler, "Weighted a search–unifying view and application," *Artificial Intelligence*, vol. 173, no. 14, pp. 1310–1342, 2009.

[11] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 3946–3952.

[12] R. Dechter and J. Pearl, "Generalized best-first search strategies and the optimality of a*," *J. ACM*, vol. 32, no. 3, pp. 505–536, Jul. 1985.

[13] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009.

[14] L. Di Stefano and A. Bulgarelli, "A simple and efficient connected components labeling algorithm," in *Image Analysis and Processing, 1999. Proceedings. International Conference on*. IEEE, 1999, pp. 322–327.

[15] J. Van Den Berg, P. Abbeel, and K. Goldberg, "Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 895–913, 2011.

[16] A. Rosenfeld and J. L. Pfaltz, "Sequential operations in digital picture processing," *Journal of the ACM (JACM)*, vol. 13, no. 4, pp. 471–494, 1966.

[17] J. T. Thayer and W. Ruml, "Faster than weighted a*: An optimistic approach to bounded suboptimal search." in *ICAPS*, 2008, pp. 355–362.