Noname manuscript No.
(will be inserted by the editor)

Statistic Methods for Path-Planning Algorithms Comparison

Pablo Muñoz · David F. Barrero · María D. R-Moreno

Received: date / Accepted: date

Abstract The path-planning problem for autonomous mobile robots has been addressed by classical search techniques such as A* or, more recently, Theta* or S-Theta*. However, research usually focuses on reducing the length of the path or the processing time. The common practice in the literature is to report the run-time/length of the algorithm with means and, sometimes, some dispersion measure. However, this practice has several drawbacks, mainly due to the loose of valuable information that this reporting practice involves such as asymmetries in the run-time, or the shape of its distribution. Run-time analysis is a type of empirical tool that studies the time consumed by running an algorithm. This paper is an attempt to bring this tool to the path-planning community. To this end the paper reports an analysis of the run-time of the path-planning algorithms with a variety of problems of different degrees of complexity, indoors, outdoors and Mars surfaces. We conclude that the time required by these algorithms follows a lognormal distribution.

Keywords Path-Planning · Run-time analysis · Robotics · Planetary exploration

Pablo Muñoz
Departamento de Automática, Universidad de Alcalá, Spain
Tel.: +34 91 8856603
Fax: + 34 91 885 6923
E-mail: pmunoz@aut.uah.es

David F. Barrero
Departamento de Automática, Universidad de Alcalá, Spain
Tel.: +34 91 8856920
Fax: + 34 91 885 6923
E-mail: david@aut.uah.es

María D. R-Moreno
Departamento de Automática, Universidad de Alcalá, Spain
Tel.: +34 91 8856607
Fax: + 34 91 885 6923
E-mail: mdolores@aut.uah.es

1 Introduction

Path-planning is focused on finding an obstacle-free path between an initial position and a goal, trying as far as possible that this path is optimal. The path-planning problem representation as a search tree over discretized environments with blocked or unblocked squares has been widely discussed. Algorithms such as A* [1] allow one to quickly find routes at the expense of an artificial restriction of heading changes of $\pi/4$. However, there have been many improvements such as A*PS, a variant of A* that is based on smoothing the path obtained by A* at a later stage, or its application to non-uniform costs maps in Field D* [2] or, more recently, Theta* [3] which aims to remove the restriction on heading changes that generates A* and gets better path lengths. The main difference between A* and Theta* is that the former only allows that the parent of a node is its predecessor, while in the last, the parent of a node can be any node. This property allows Theta* to find shorter paths with fewer turns compared to A*. However, this improvement implies a higher computational cost due to additional operations to be performed in the expansion nodes process as it will be explained in section 2.1. S-Theta* algorithm [4] is based on Theta* but introduces in the evaluation function a parameter to optimize the changes in the direction of the route. The field of application is straight forward: robots whose rotation cost is greater than the movement in straight line (what is a vast majority). Other approximations want to reduce the processing time via heuristics [5] or improving the efficiency of the algorithms [6]. It is worth mentioning that these algorithms work on fully observable environments except Field D*, that can face partially observable environments applying a replanning scheme. In [7],

an extensive comparison between Theta*, A*, Field D* and A*Post Smoothed (A*PS) [8] is performed.

The study of the run-time from a statistical perspective is known as *run-time analysis*. The common practice in the literature is to report the run-time of the algorithm with means and, sometimes, some dispersion measure. However, this practice has several drawbacks, mainly due to the loose of valuable information that this reporting practice involves such as asymmetries in the run-time, or the shape of its distribution.

Run-time analysis overcomes this limitation by considering all the data with its focus on the statistical properties of the algorithm run-time. The main tool of run-time analysis is the *Run-Time Distribution* (RTD) which is the empirical statistical distribution of the run-time; in this way, the RTD is fully characterized. Other statistics such as the mean can be obtained from the RTD, and it opens exciting parametric statistical analysis tools. For instance, once the run-time distribution of an algorithm is identified, parametric hypothesis tests can be used to compare the run-time of two algorithms, providing a more rigorous comparison methodology.

This work is an attempt to provide a first run-time analysis of some classical path-planning algorithms, A* [1], Theta* [7] and S-Theta* [4]. We follow an empirical approach, running the algorithms in grid-maps of different difficulties, some of them artificially generated and others taken from the Mars surface, fitting the resulting run-time distribution with some statistical distributions. We show that, in the same line as the related run-time literature in other fields, the run-time of the three algorithms under study follow a lognormal distribution. We also observe a dependence between the run-time distribution and the problem difficulty.

The paper is structured as follows. First, we introduce the path-planning representation and algorithms that are object of study. Then, we describe the RTD analysis, including some related literature mainly from Metaheuristics. Next, in section 4 we perform the run-time analysis. We review some related work in section 5 and finally some conclusions are outlined.

2 Notation and Algoritms

The most common terrain discretization in path-planning is a regular grid with blocked and unblocked square cells [9]. For this kind of grids we can find two variants: (i) the center-node in which the mobile element is in the center of the square; and (ii) corner-node, where nodes are the vertex of the square. For both cases, a valid path is that starting from the initial node reaches the goal node without crossing a blocked cell. In our experiments we have employed the corner-node approximation, but our algorithm also works with the center-node representation.

A node is represented as a lowercase letter, assuming p a random node and, s and g the start and goal nodes respectively. Each node is defined by its coordinate pair (x, y), being x_p and y_p for the p node. A solution has the form $(p_1, p_2, ..., p_{n-1}, p_n)$ with initial node $p_1 = s$ and goal $p_n = g$. As well, we have defined four functions related to nodes: (i) function $succ(p)$ that returns a subset with the visible neighbors of p; (ii) function $parent(p)$ that indicates who is the parent node of p; (iii) $dist(p, t)$ that represents the straight line distance between nodes p and t; and (iv) function $angle(p, q, t)$ that gives as a result the angle (in degrees) formed by segments $\overline{pq}$ and $\overline{pt}$ (that is, the angle in opposition of the $\overline{qt}$ segment) of the triangle formed by $\triangle pqt$, in the interval $[0°, 180°)$.

In the next subsections, we will briefly explain the path-planning algorithms used in the experiments.

2.1 Theta* Algorithm

Theta* [3] [7] is a variation of A* for any-angle path-planning on grids. A* is simple to implement, is very efficient, and has lots of scope for optimization [1]. But it has an important limitation: it typically uses 8 neighbors nodes, so it restricts the path headings to multiples of $\pi/4$, causing that A* generates a sub-optimal path with zig-zag patterns.

But Theta* has been adapted to allow any-angle route, i.e. it is not restricted to artificial headings of $\pi/4$ as is the case of A* with 8 neighbors. Although there are works like [10] that subdivide cells easing this restriction, or others that use more neighbors, but they are ad-hoc solutions. There are two variants for Theta*: Angle-Propagation Theta* [3] and Basic Theta* [7]. We assume that talking about Theta* refers to the last one. The difference between these two versions is how the calculation of the line of sight between pairs of nodes is performed, being in Basic Theta* simpler but with a higher computational complexity in the worst case than the variant Angle-Propagation. Moreover, the latter sometimes indicates that two nodes do not have line of sight when in fact they do, slightly increasing the length of the resulting path.

Theta* shares most of the code of the A* algorithm and the only variation is how the Vertex are updated. They both use the following node evaluation function:

$$\mathrm{F}(t) = \mathrm{G}(t) + \mathrm{H}(t) \tag{1}$$

Where:

- $G(t)$: represents the cumulative cost to reach the node t from the initial node. It is the length of the shortest path from the start node to the t node.
- $H(t)$: is the heuristic value, i.e, the estimated distance to the goal node. In the case of A* uses the Octile heuristic and Theta* uses the Euclidean heuristic.

Both algorithms use the reference to the parent node, that is, $parent(t) = p \Rightarrow t \in succ(p)$, however, Theta* eliminates this restriction and allows that the parent node of t is any node q accessible whenever there is a line of sight between t and q.

Then, Theta* will expand the most promising nodes, i.e. it first expands the nodes with lower values for $F(t)$. In the case that the expanded node is the goal, the algorithm will return the path by traversing the parents pointers backwards from the goal to the start node. If instead the list of candidate nodes is empty, it means that it is impossible to reach the goal node from the initial node and the algorithm will return a failure.

When dealing with the successor nodes, the first thing that Theta* does is to check if among the successors of the current position p, being $t \in succ(p)$, and the parent of the current node $q = parent(p)$, there is a line of sight. The high computational cost associated to this checking process is due to the algorithm used (a variant for drawing lines [11] that only uses integer operations for verification) which linearly grows as a function of the distance between the nodes to evaluate. In case of no line of sight, the algorithm behaves as A* and the parent node of t is p. However, if the node t can be reached from q, the algorithm will check whether the node t has already been reached from another node, and then, only update the node t if the cost of reaching it from q is less than the previous cost. Following the described expansion process, Theta* only has heading changes at the edges of the blocked cells.

2.2 S-Theta* Algorithm

The Smooth Theta* (S-Theta*) algorithm [4] developed from Theta*, aims to reduce the amount of heading changes that the robot should perform to reach the goal. To do this, a modified cost function $F(t)$ has been used, as shown in eq. 2.

$$\mathrm{F}(t) = \mathrm{G}(t) + \mathrm{H}(t) + \alpha(t) \tag{2}$$

The new term $\alpha(t)$ gives us a measure of the deviation from the optimal trajectory to achieve the goal as a function of the direction to follow, conditional to traversing a node t. This value represents the deviation in the trajectory to reach the goal node g through the node t in relation to the straight-line distance between the parent of its predecessor ($t \in succ(p)$ and $parent(p) = q$) and the goal node. Using that value, the algorithm tries to find the new shortest route between the successor to the current position, t, and the goal node, considering the heading change during the search process. Also, this term causes that nodes far away from that line will not be expanded during the search.

3 Run-time analysis

The basic tool used for run-time analysis is the RTD, term that was introduced by Hoos and Stützle [12]. Let us name $rt(i)$ the run-time of the ith successful run, and n the number of runs executed in the experiment, then the RTD is defined as the empirical cumulative probability $\hat{P}(rt \leq t) = \#\{i|rt(i) \leq t\}/n$, where $\#\{i|rt(i) \leq t\}$ is the number of successful trials at time t, and n the number of trials, it turns out that $\#\{i|rt(i) \leq t\} \leq n$ [13]. It is simply the empirical probability of finishing the algorithm run before t. There is an extensive literature about RTDs, mainly in Metaheuristics, but there are also several studies in classical AI problems.

There are many studies about RTD in different problems and algorithm, the section 5 introduces some of these research in the field of Metaheuristics. Interestingly there are some results that appear across different problems and algorithms, there are three omnipresent distributions in RTD analysis: Exponential, Weibull and Lognormal. These three distributions play a central role in Reliability Theory, suggesting a link. This observation help us to select a first subset of distributions to fit our data.

In order to carry out the run-time analysis, we need to gather empirical data in a controlled way. Given the need of a uncertainty source, experiments may vary two factors, the random seed and the problem. The former is common in the Metaheuristics and random search literature, while the latter is used with deterministic algorithms. Even though this distinction does not use to be clear in the literature, we think that it is critical since it determines the comparability of the experiments and their interpretation.

In the following section we study the presence of the three distributions (exponential, Weibull and Lognormal) in the RTD of A*, Theta* and S-Theta* for path-planning problems of varying difficulty.

4 Results

In order to assess the run-time behavior of the path-planning algorithms we have performed an experimental approximation[1]. These algorithms are deterministic, so, on the contrary than other run-time analysis performed in Metaheuristics, there is no variability due to the random seed. In this case, the variation comes from the problem, to be more specific, we used a random map generator that can control the percentage of obstacles in the map, and therefore, we can set the complexity of the problem. We have set three types of experiments:

- Outdoors environments: we have generated 2000 synthetic maps increasing the number of blocked cells from 5% to 40%.
- Indoors environments: we have generated 1800 maps with different sizes. These maps are generated from the random combination of square patterns that represent different configurations of corridors and rooms.
- Mars surface environment: in order to show that the analysis of the algorithms developed can be used in real environments, we have performed several experiments on real maps of the Mars surface.

The algorithms were implemented in Java. In order to make a fair comparison, the implementation of the three algorithms use the same methods and structures to manage the information grid[2]. To measure the run-time we have employed `System.currentTimeMillis()`. Reporting time in this way has several drawbacks [14] such has hardware dependence that difficults results comparison, but in this case we think it is justified because the algorithms contains computations that are not well captured by machine-independent time measures, such us the number of expanded nodes. Of course, the price we have to pay is an increased difficulty to repeat and compare these results, but given that our interest is studying the statistical properties of the run-time rather than compare algorithms, we think that in this case it is an acceptable drawback.

Next subsections describe in details the results obtained in each environment.

4.1 Outdoors maps

We have generated random maps with different ratio of random obstacles, in particular we used 5%, 10%, 20%, 30% and 40% of obstacles. For each ratio of obstacles we generated 2,000 random maps of 500 *times*500 nodes and solved the map with each of the three algorithms under study. This procedure yields $2,000 \times 5 = 10,000$ runs for each one of the three algorithms. The initial point in the map is always the corner at the top left of the map, and the goal node is placed at the right, locating between the bottom corner and randomly 20% up nodes. The map generator was implemented with guarantees to keep at least one path from the start node to the goal node. To do this, when an obstacle is set, his periphery is protected to avoid overlapping obstacles. Thus, it is always possible surround an obstacle to avoid it.

We have firstly performed an exploratory analysis of the results. To this end we depicted several histograms of the run-times, as can be seen in Fig. 1. The histograms were grouped by algorithm and problem hardness and they show some interesting facts.

We observe that the shape of the run-time histograms varies in function of the algorithm and problem hardness. S-Theta* has a longer tail and its shape is more asymmetrical in comparison to the rest of the algorithms, this fact is more clear in hard problems than in easy ones. The run-time of A* presents a smaller variance than the other two algorithms, and it increases with the problem hardness. The run-time required to

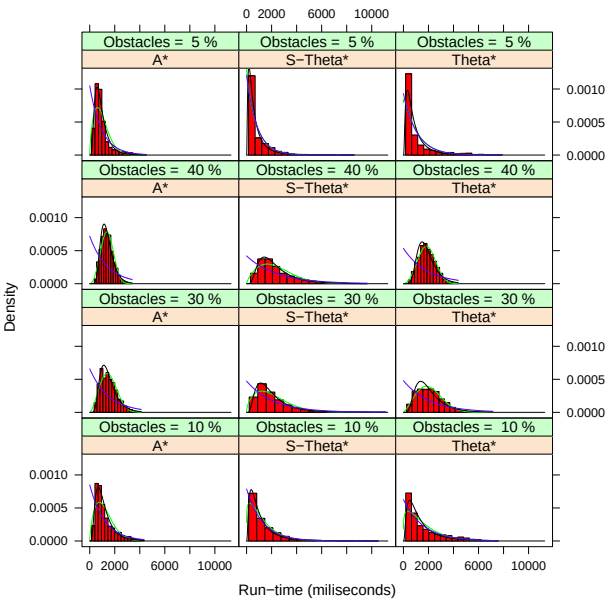

Fig. 1 Histogram of the run-time, measured in miliseconds, of the three path-planning algorithms under study. Histograms have been grouped by algorithm and ratio of obstacles. The histograms have been adducted with three distributions that appear overlapped: Lognormal (black), exponential (blue) and Weibull (green). Color figure online.

[1] All the code, scripts and datasets required to repeat these experiments are available on `http://atc1.aut.uah.es/~david/datasets.php#ki2013`

[2] The execution was done on a 2 GHz Intel Core i7 with 4 GB of RAM under Ubuntu 10.10 (64 bits)

solve hard problems has more variance than easy problems. In any case, the shape and range of values of Theta* and S-Theta* are similar, which seems reasonable given that they are variations of the same algorithm.

Looking for a statistical distribution able to fit data is more interesting. With the results reported in the literature we have tried to fit data to the three distributions that appear to play a more important role in RTD analysis, the Exponential, Weibull and Lognormal distribution. So, the histograms shown in Fig. 1 are depicted with a set of overlapping distributions, which corresponds to the three statistical distributions previously mentioned. The parameters of these distributions were fitted by maximum likelihood. As the reader can observe in Fig. 1, the distribution that fits better our data in most cases is the lognormal. The exceptions to this observation are Theta* and S-Theta* solving maps with a very low number of obstacles; in that case the exponential distribution seems to describe the run-time behavior better than the lognormal. Even in this case, the A* run-time is well described by the lognormal, that is able to describe its peaks better than the exponential.

Given the observations of the exploratory analysis, it seems reasonable to focus the study on the lognormal distribution and hypothetize that the run-time of the A*, Theta* and S-Theta* algorithms in path-planning problems follow a lognormal distribution. The parameters of the lognormal distributions that we obtained using maximum-likelihood are shown in Table 1. In order to evaluate the hypothesis about the RTDs lognormality, it is desirable to provide additional evidences. One of the most interesting properties of the lognormal distribution is its close relationship to the normal distribution, actually, lognormal data can be converted to normal data by simply taking logarithms. We can use this property to verify whether the RTD is lognormal or not in a more formal way.

To verify the lognormality of the RTD, we first plotted a QQ-plot of the logarithm of the run-time against a normal distribution, which is shown in Fig 2. If the logarithm of the run-time is normal, we can conclude that the run-time is lognormal. Fig. 2 confirms our initial suspicion about the lognormality of the RTD. In addition, the QQ-plots also show the relationship between the distribution and the problem hardness. Theta* and S-Theta* algorithms produce less lognormal RTDs in very easy problems. On the contrary, the influence of the problem hardness on A*, if it has any, is not so evident, the QQ-plot is almost a line for all the ratios of obstacles. In summary, A* RTDs seem lognormal for any number of obstacles, while the run-time of Theta* and S-Theta* algorithms seem to follow a lognormal distribution, but easy problems fit worse.

Table 2 Shapiro-Wilk test of normality of the logarithm of the run-time. With these p-values we cannot reject the hypothesis of normality of the logarithm of the run-time, which means that we cannot reject the lognormality of the run-time. Only Theta* and S-Theta* with a ratio of 5% of obstacles provide evidence to let us reject the normality of the RTD

Algorithm	Obst %	W	p-value	Significance
A*	5	0.9772	0.7473	
A*	10	0.9579	0.2743	
A*	20	0.9808	0.8475	
A*	30	0.9378	0.0792	
A*	40	0.9546	0.2237	
Theta*	5	0.8998	0.0083	$\alpha = 0.001$
Theta*	10	0.9437	0.1142	
Theta*	20	0.9479	0.1487	
Theta*	30	0.9343	0.0641	
Theta*	40	0.9444	0.1194	
S-Theta*	5	0.8322	0.0003	$\alpha = 0.001$
S-Theta*	10	0.9409	0.0965	
S-Theta*	20	0.9771	0.7428	
S-Theta*	30	0.9829	0.8968	
S-Theta*	40	0.9876	0.9725	

A more rigorous analysis of the lognormality of the run-time is desirable. For this reason we have performed a Shapiro-Wilk test of normality of the logarithm of the run-time, whose result is shown in Table 2. In order to avoid undesirable effects of the large number of

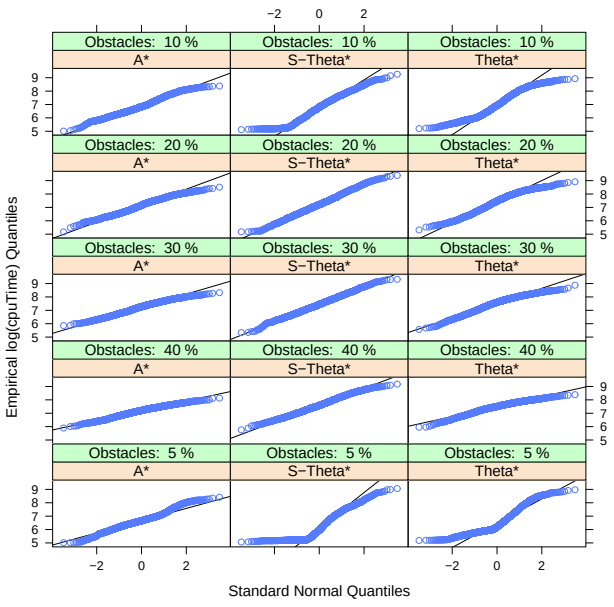

Fig. 2 Quantile plot that assesses the normality of the logarithm of the run-time of the three algorithms under study: A*, Theta* and S-Theta*. Given the relationship between the lognormal and normal distributions, the logarithm of lognormal data must transform it into normal data

Table 1 Mean ($\bar{x}$), standard deviation (s) and estimated parameters of the lognormal distribution ($\hat{\mu}, \hat{\sigma}$) fitted by maximum likelihood. Each row corresponds to an experiment with 2,000 problems with the given rate of obstacles in the map

Algorithm	Obst (%)	$\bar{x}$	s	$\hat{\mu}$	$\hat{\sigma}$
A*	5	953.6	637.6	6.690	0.567
A*	10	1175.9	765.7	6.886	0.600
A*	20	1439.9	762.6	7.131	0.542
A*	30	1516.2	651.3	7.228	0.449
A*	40	1390.9	478	7.176	0.361
Theta*	5	1074.3	1266.5	6.520	0.883
Theta*	10	1595.6	1438.9	6.989	0.887
Theta*	20	2028	1272.1	7.395	0.700
Theta*	30	2084.5	987.4	7.515	0.534
Theta*	40	1871.2	667	7.464	0.390
S-Theta*	5	829.2	999	6.202	0.975
S-Theta*	10	1271	1216.7	6.749	0.916
S-Theta*	20	1803.1	1506.8	7.220	0.744
S-Theta*	30	2111.3	1500.4	7.445	0.648
S-Theta*	40	2375.65	1460.6	7.601	0.589

samples, we have computed the test using 30 random runs. Results shown in the table confirms our previous observations in the histograms and the QQ-plot. The p-values in almost all the cases are quite high, which means that the hypothesis of lognormality is compatible, with a high probability, with our data. However, there are two exceptions, the p-value of Theta* and S-Theta* with a ratio of 5% of obstacles is quite small, 0.0083 and 0.0003 respectively, which drives us to reject the null hypothesis (i.e. the lognormality of the data) with $\alpha = 0.001$.

We were curious about the different RTDs reported by the literature and the reason of those differences. In order to obtain some clue for its answer, we performed a simple experiment. Instead of plotting run-time histograms grouped by algorithm and problem hardness, we tried to plot histograms joining all the runs belonging to the same algorithms, in this way, problems of different hardness were merged. As in the exploratory analysis, we plotted the three main distributions in RTD literature to check rapidly whether data fits or not that distribution. The result can be seen in Fig. 3. We have to consider that this figure shows in fact an overlapping of several distributions, it can be seen as the sum of each column in Fig. 3.

Fig. 3 is a quite interesting result. The lognormal distribution still seems to fit very well the A* run-time, however, the Theta* algorithms are not so clear. The left tails of their histogram seems to have disappeared, or at least it is sufficiently small to not appear clearly in the histogram. This fact introduces the exponential distribution in the discussion, it can be seen that this distribution is able to fit data quite well, nonetheless, the lognormal distribution still provides a excellent fit. So, it makes us conjecture that depending on how the run-time is visualized the statistical model that fit the RTD may change. In particular, it is well known that joining random variables of a certain distribution may produce a random variable of another distribution. This fact could explain, in part, the diversity of RTDs found in the literature, and it is an additional motivation to take care about how experimentation and data processing are done.

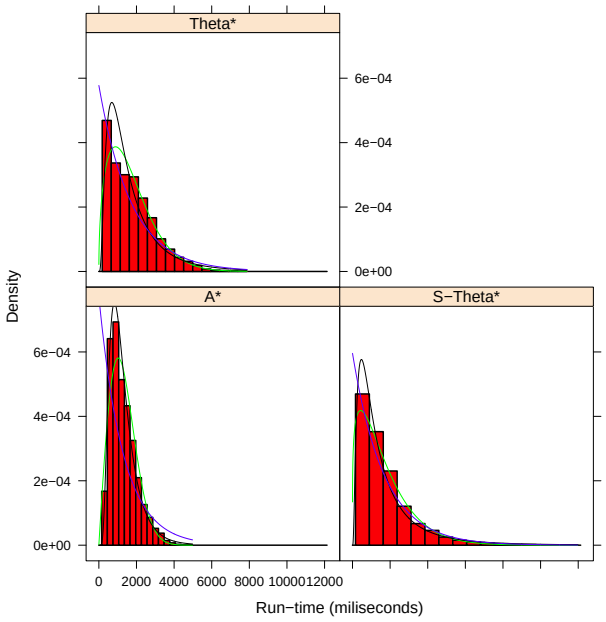

Fig. 3 Histogram of the run-time, measured in miliseconds, of the three path-planning algorithms under study. Histograms have been grouped by algorithm, but not by ratio of obstacles, so each histogram contains runs of different problem hardness. The histograms have been adjusted with three distributions that appear overlapped: Lognormal (black), Weibull (green) and exponential (blue). Color figure online.

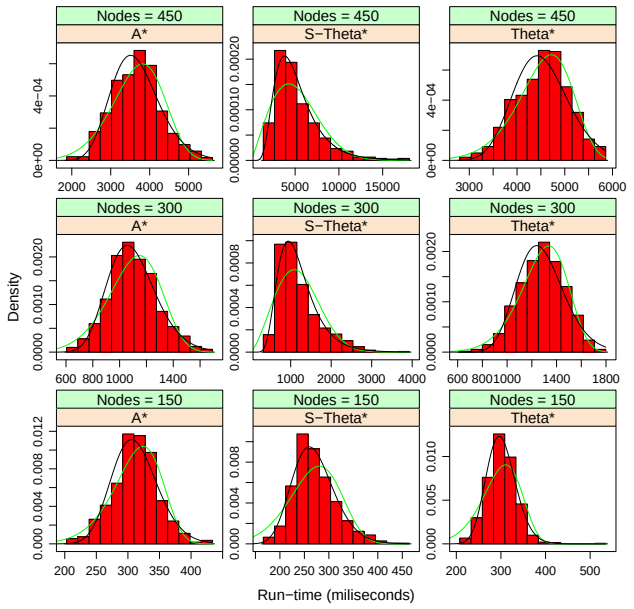

Fig. 4 Histogram of the run-time, measured in miliseconds, of the three path-planning algorithms executed with indoor maps. Histograms have been grouped by algorithm and number of nodes in the map. The histograms have been adjusted with two distributions that appear overlapped: Lognormal (black) and Weibull (green). Color figure online.

With this evidence, it seems reasonable to assume the lognormality of the RTD in problems with a ratio of obstacles higher than 5% of the three algorithms under study.

4.2 Indoors maps

For indoor maps, we have run the algorithms over 1800 maps with different sizes, 150x150, 300x300 and 450x450 nodes (600 maps per size), always starting from the upper left corner (0, 0) and reaching the target in the opposite corner bottom. The indoor maps are generated from the random combination of 30x30 nodes square patterns that represent different configurations of corridors and rooms. These patterns are designed in a way that we can access to the next pattern through doors on each side, symmetrically placed.

We begin exploring the run-time by visualizing them in histograms (see Fig. 4). It shows a shape quite similar to the one shown in Fig. 1 with the outdoor maps. Regardless of the algorithm, the histogram shape shows a bell-shaped distribution that, in some cases is asymmetric while in others it is not. There is a clear relationship between the difficulty of the map, given by the number of nodes, and the data mean shown in the histogram: the harder is the set of indoor maps, the higher is the time to get the path.

Figure 4 also depicts overlapped to the histogram, the distribution of the lognormal and Weibull distributions. Their parameters were estimated using maximum-likelihood. In general both distributions are good models to our data, however, the lognormal distribution seems to fit the data better. The lognormal distribution fits better data in the peak of the histogram, this is quite clear for the S-Theta* algorithm, that has sharp peaks.

Like in outdoor maps, it seems that the lognormal distribution is a good choice to fit the run-time. Table 3 contains the estimated parameters $\hat{\mu}$ and $\hat{\sigma}$ of the lognormal distribution fitted using maximum-likelihood. It shows clearly how the sample mean increases with the number of nodes, which is a fact previously observed in the histogram.

At this point, the lognormal distribution seems to be a good candidate to model the run-time, also in the indoor maps. However, it is desirable to test this hypothesis with more rigorous statistical tools. Firstly, we depicted a quantile plot of the logarithm of the run-time in indoor maps in Fig. 5. As in the outdoor case, the logarithm of lognormal data must follow a normal distribution, and so, a straight line in Fig. 5 means that the represented data is lognormal. And this is, actually, the fact shown in Fig. 5: the data forms clearly

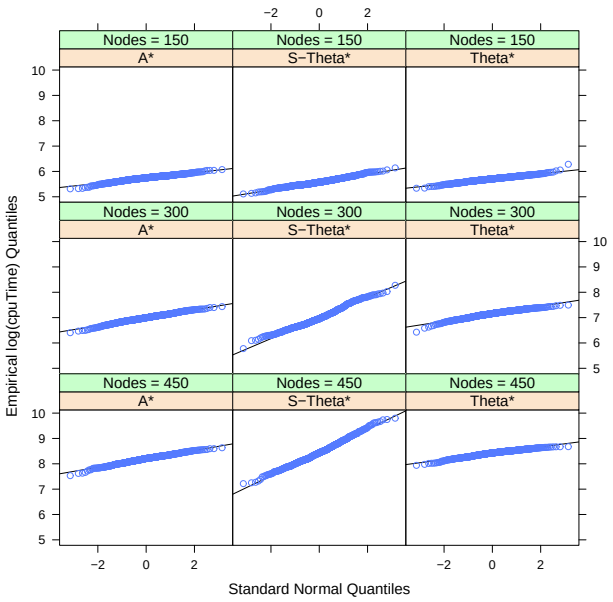

Fig. 5 Quantile plot that assesses the normality of the logarithm of the run-time of the three algorithms under study: A*, Theta* and S-Theta*. These algorithms have been run with indoor maps. Given the relationship between the lognormal and normal distributions, the logarithm of lognormal data must transform it into normal data

Table 3 Mean ($\bar{x}$), standard deviation (s) and estimated parameters of the lognormal distribution ($\hat{\mu}, \hat{\sigma}$) fitted by maximum likelihood for indoor maps.

Algorithm	Nodes	$\bar{x}$	s	$\hat{\mu}$	$\hat{\sigma}$
A*	150	312.1	35.6	5.737	0.116
A*	300	1101.3	180.3	6.991	0.166
A*	450	3666.8	617.1	8.193	0.172
Theta*	150	301.4	33.14	5.703	0.109
Theta*	300	1281.4	185	7.145	0.150
Theta*	450	4530	565.3	8.410	0.129
S-Theta*	150	270.2	44.4	5.586	0.160
S-Theta*	300	1191.6	510.1	7.002	0.393
S-Theta*	450	5230.2	2647.5	8.453	0.459

Table 4 Shapiro-Wilk test of normality of the logarithm of the run-time in indoor maps. With these p-values we cannot reject the hypothesis of normality of the logarithm of the run-time, which means that we cannot reject the lognormality of the run-time in indoor maps.

Algorithm	Nodes	W	p-value	Significance
A*	150	0.9741	0.6577	
A*	300	0.9631	0.3712	
A*	450	0.9789	0.7945	
Theta*	150	0.9579	0.2740	
Theta*	300	0.9812	0.8571	
Theta*	450	0.9391	0.0859	
S-Theta*	150	0.9888	0.9833	
S-Theta*	300	0.9708	0.5614	
S-Theta*	450	0.9732	0.6305	

a straight line, supporting the idea that they are well described by a lognormal distribution.

We have also performed a formal Shapiro-Wilk test of normality, as shown in Table 4. As in the previous section, we have performed the test on the logarithm of the run-time. The result is more clear than the shown in outdoor maps. In indoor maps, the run-times we gathered are compatible with the lognormality hypothesis in all the cases, regardless of their difficulty. Once we have strong evidences of the lognormality of the run-time, we can apply powerful parametric statistics to get statistically sounded conclusions.

4.3 Mars surface maps

Finally, we have tested the path-planning algorithms in a small set of Mars maps. We have chosen the landing site of the Mars Science Laboratory (MSL) – Curiosity– on the Gale Crater, using the Digital Terrain Model (DTM) obtained by the Mars Reconnaissance Orbiter and its High Resolution Imaging Science Experiment instrument. This DTM[3] has a vertical resolution of 30 cm and a horizontal resolution of 1 meter.

Due to the huge size of the DTM, we have taken a set of regions with 1000x1000 points and we have computed an histogram filter to mark lower and higher areas of the map as obstacles (the employed path-planning algorithms work on flat surfaces).

Table 5 shows the results of comparing on a real Martian surface the three path-planning algorithms, using the following paremeters: path length, turns in degrees and the runtime in miliseconds. We can see that in all cases A* is the worst in two of the three parameters used: path length and total turns. Theta* obtains the best path length but the worst runtime. In the contrary, S-Theta* obtains close lengths with the lowest runtime of the three algorithms and the less number of turns necessary to reach the goal. Previous results for synthetic outdoor maps (see fig. 1) show that A* trends to have less runtime than Theta* or S-Theta*. This results show that having a small set of maps (i.e., 3) we cannot conclude that one algorithm (in this case, S-Theta*) is better than the other two. Figure 6 shows the solution path of each path planning algorithm on these maps used.

5 Related work

The RTD analysis has been applied to a wide variety of problems and algorithms in the fields of Metaheuristic, with some interesting results. Hoos and Stützle applied RTD analysis to different algorithms and problems. They found that the RTD of WSAT algorithms applied to the 3SAT problem is exponential when the parameters setting is optimal, shifted exponential or Weibull when the parameters setting is not optimal [12]. Analogously, they studied in [15] the RTD of some other stochastic local search algorithms, such as GWSAT,

[3] DTM of MSL Landing Site in Gale Crater can be obtained at https://hirise.lpl.arizona.edu/dtm/dtm.php?ID=ESP_023957_1755

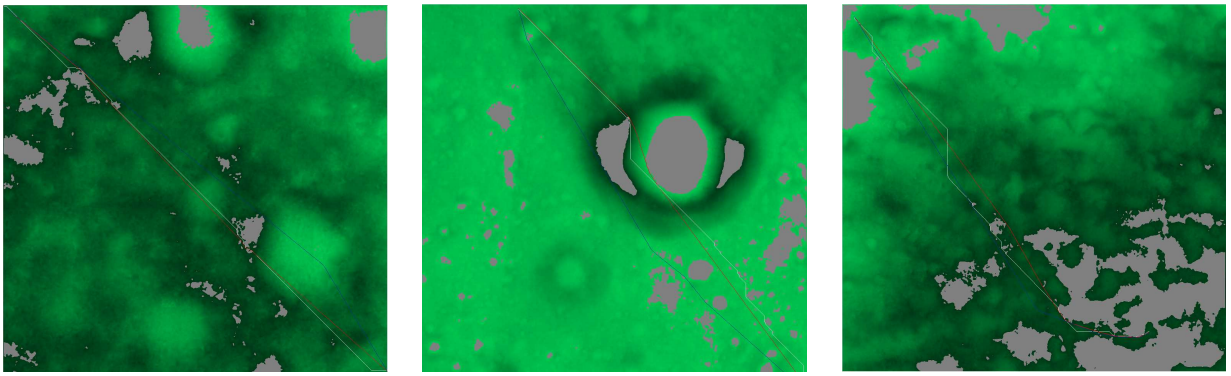

Fig. 6 Paths found for a Mars maps. White: A*; red Theta* and blue for S-Theta*. From left to right: map with 5.79% of obstacles, 8.78% and 17.73% blocked cells.

Table 5 Results for Mars maps

Algorithm	Obst (%)	Path length	Turns in degrees	Runtime (ms)
A*	5.79	1412.40	1215.00	6234
Theta*	5.79	1382.25	86.82	7857
S-Theta*	5.79	1393.56	19.94	4679
A*	8.78	1456.84	2385.00	18636
Theta*	8.78	1393.67	312.00	22080
S-Theta*	8.78	1394.96	280.37	16647
A*	17.73	1261.24	3105.00	36786
Theta*	17.73	1193.48	141.02	39486
S-Theta*	17.73	1209.47	97.70	17342

GSAT with tabu-lists, TMCH and WMCH, to solve instances of SAT and CSPs, finding that, again, when parameters are optimal, the RTD follows an exponential, and otherwise RTD fits a Weibull distribution. Curiously, this result only holds for hard instances, in easy instances they did not find statistical significance. Hard instances are those difficult to solve, in our case, ones with higher number of blocked cells (for random maps) and bigger map size (for indoor maps). In a later work [16], they observed that the RTD of hard 3SAT instances solved with WalkSAT algorithm also follows an exponential distribution, and more interestingly, the higher the difficulty of the problem, the higher the fit is found. In a more recent work, Hoos and Stützle [13] compared various versions of GSAT and WalkSAT algorithms to solve some problems coded as 3SAT (random 3-SAT, graph coloring, block world and logistic planning), finding that these algorithms also have exponential RTDs for hard problems and high values of noise parameter. More importantly, they found that RTDs of easy problems are no exponential, despite their tails are still exponential.

Chiarandini and Stützle [17] studied the RTD of ILS, ACO, Random Restart Local Search and two variants of SA applied to the course timetabling problem, finding that the Weibull distribution approximates well the RTDs. They report, however, that in SA, the RTD in hard problems can be approximated, at least partially, using a shifted exponential distribution. On the contrary than Hoos, Stützle and Chiarandini, Frost *et al.* [18] studied the RTD using the same algorithm, backtracking, with different problem instances of the CSP. The RTD of the algorithm running on solvable instances [19] follows a Weibull distribution, while unsolvable instances generate lognormal run-times. However, only the lognormal distribution for solvable problems had statistical significance. In addition, Barrero *et al.* studied the RTDs in tree-based Genetic Programming [20], finding lognormal RTDs whose goodness of fit depends on the problem difficulty.

6 Conclusions

Along this paper we have performed a run-time analysis of A*, Theta* and S-Theta* to study their RTD statistical properties and the influence of the problem hardness. We have used three types of maps in order to perform the study: Outdoor, indoor and real martian maps.

We found that the run-time of those algorithms, when applied to a set of non-trivial path-planning problems of similar hardness, follow a lognormal distribution in the three cases. The evidence we have found in this line is strong. However, we also observed that the goodness-of-fit is weaker for the Theta* algorithms when the problem is easy in outdoor maps.

The results shown is this paper are aligned with other run-time analysis done for other search algorithms and problems, suggesting an underlying common behavior. The lognormal distribution has described our data quite well. This is an exciting topic that, given its generality, is worth to study.

This observation leads to a better knowledge of A*, Theta* and S-Theta algorithms, but it also has practical implications. The common procedure to compare run-times followed in the literature is a naïve comparison of means, which is a weak method from a statistical point of view. If the RTD of the algorithm is known, strong parametric methods could be used, and in particular lognormal RTDs open the use of well-known (and easy) statistics to enhance the experimental methods used to study path-planning algorithms. So, in order to compare the run-time of two algorithms with a sound statistical basis, we can -and should- use hypothesis testing (Student's t-test with the logarithm of the runtime) or ANOVA if several algorithms are involved.

Acknowledgements Pablo Muñoz is supported by the European Space Agency (ESA) under the Networking and Partnering Initiative (NPI) *Cooperative systems for autonomous exploration missions*. This work was partially supported by the Spanish CDTI project COLSUVH, leaded by the *Ixion Industry and Aerospace* company.

References

1. I. Millington and J. Funge, *Artificial Intelligence for Games*, 2nd ed. Morgan Kaufmann Publishers, 2009.
2. D. Ferguson and A. Stentz, "Field D*: An interpolation-based path planner and replanner," in *Proceedings of the International Symposium on Robotics Research (ISRR)*, October 2005.
3. A. Nash, K. Daniel, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," in *In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2007, pp. 1177–1183.
4. P. Munoz and M. D. R-Moreno, "S-Theta*: low steering path-planning algorithm," in *Procs. of the Thirty-second SGAI International Conference on Artificial Intelligence*, Cambridge, UK, December 2012.
5. ——, "Improving efficiency in any-angle path-planning algorithms," in *6th IEEE International Conference on Intelligent Systems IS'12*, Sofia, Bulgaria, September 2012.
6. S. Choi, J. Y. Lee, and W. Yu, "Fast any-angle path planning on grid maps with non-collision pruning," in *IEEE International Conference on Robotics and Biomimetics*, Tianjin, China, December 2010, pp. 1051–1056.
7. K. Daniel, A. Nash, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," *Journal of Artificial Intelligence Research*, vol. 39, pp. 533–579, 2010.
8. A. Botea, M. Muller, and J. Schaeffer, "Near optimal hierarchical path-finding," *Journal of Game Development*, vol. 1, pp. 1–22, 2004.
9. P. Yap, "Grid-based path-finding," in *Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, vol. 2338. Springer Berlin / Heidelberg, 2002, pp. 44–55.
10. G. Ayorkor, A. Stentz, and M. B. Dias, "Continuous-field path planning with constrained path-dependent state variables," in *ICRA 2008 Workshop on Path Planning on Costmaps*, May 2008.
11. J. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal*, vol. 4, pp. 25–30, 1965.
12. H. Hoos and T. Stützle, "Evaluating Las Vegas Algorithms – Pitfalls and Remedies," in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*. Morgan Kaufmann Publishers, 1998, pp. 238–245.
13. ——, "Local Search Algorithms for SAT: An Empirical Evaluation," *Journal of Automated Reasoning*, vol. 24, no. 4, pp. 421–481, 2000.
14. R. Barr and B. Hickman, "Reporting Computational Experiments with Parallel Algorithms: Issues, Measures, and Experts' Opinions," *ORSA Journal on Computing*, vol. 5, pp. 2–2, 1993.
15. H. Hoos and T. Stützle, "Characterizing the Run-Time Behavior of Stochastic Local Search," in *Proceedings AAAI99*, 1998.
16. ——, "Towards a Characterisation of the Behaviour of Stochastic Local Search Algorithms for SAT," *Artificial Intelligence*, vol. 112, no. 1-2, pp. 213–232, 1999.
17. M. Chiarandini and T. Stützle, "Experimental Evaluation of Course Timetabling Algorithms," Intellectics Group, Computer Science Department, Darmstadt University of Technology, Darmstadt, Germany, Tech. Rep. AIDA-02-05, April 2002.
18. D. Frost, I. Rish, and L. Vila, "Summarizing CSP Hardness with Continuous Probability Distributions," in *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence (AAAI'97/IAAI'97)*. AAAI Press, 1997, pp. 327–333.
19. S. Epstein and X. Yun, "From Unsolvable to Solvable: An Exploration of Simple Changes," in *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.
20. D. F. Barrero, B. Castaño, M. D. R-Moreno, and D. Camacho, "Statistical Distribution of Generation-to-Success in GP: Application to Model Accumulated Success Probability," in *Proceedings of the 14th European Conference on Genetic Programming, (EuroGP 2011)*, ser. LNCS, vol. 6621. Turin, Italy: Springer Verlag, 27-29 Apr. 2011, pp. 155–166.