Search Methodologies for Efficient Planetary Site Selection


Abstract—Landing on distant planets is always a challenging task due to the distance and hostile environments found. In the design of autonomous hazard avoidance systems we find the particularly relevant task of landing site selection, that has to operate in real-time as the lander approaches the planet’s surface. Seeking to improve the computational complexity of previous approaches to this problem, we propose the use of non-exhaustive search methodologies. A comparative study of several algorithms, such as Tabu Search and Particle Swarm Optimization, was performed. The results are very promising, with Particle Swarm Optimization showing the capacity to consistently produce solutions of very high quality, on distinct landing scenarios.

I. INTRODUCTION

Landing on distant planets is always a challenging task due to the distance and hostile environments found. It usually takes the form of semi-hard or soft landings. In the first, the lander has airbags inflated around it shortly before its release from a parachute, and then bounces onto the ground until it comes to rest. The second consists in achieving quasi null velocity at landing by propulsive braking. For both cases the landing zone has to be safe - gentle slopes, small boulders, no cliffs or ridges - in order not to tear the airbags or have the lander tip over at landing. This requires a detailed knowledge of the planet’s surface features beforehand, and either the selection of a safe area large enough to cope with the landing trajectory dispersions, or a pin-point landing capability [1].

However, scientifically interesting zones often consist precisely of inherently hazardous craterized and erosion-modelled landscapes and detailed maps of the planet’s surface, that might be available for Mars and the Moon, will not be for less visited bodies. Moreover, even if maps were available and pin-point landing a reality, there is no guarantee that the ground characteristics would not change between the time of the picture and the actual landing. Wind, geological activity or meteorites could modify the surface. Since in addition real time monitoring of the descent is excluded due to communication delays and high entry dynamics, autonomous hazard avoidance capability is essential for mission success in those areas (see Figure 1).

Of particular relevance to an autonomous hazard avoidance system is the ability to select landing sites in real-time as the lander approaches the planet’s surface, and to dynamically adjust that choice as better information becomes available along the descent. In previous work, we have shown good results using a Multi-Attribute Decision Making model [2], [3] for this problem [4], [5], [1]. In this paper we present an alternative approach, with significantly lower computational complexity, based on non-exhaustive search methodologies [6] for selecting the best landing site. We perform a comparative study of several non-exhaustive search algorithms, from Hill Climbing to Tabu Search and Particle Swarm Optimization, using a Random Walk as a baseline.

This paper is organized as follows. In the next section we give a brief description of the Hazard Avoidance Problem, and the landing site selection component, which is the subject of this paper. In the third section we describe the motivation for our approach, and the algorithms considered. Section IV describes the criteria and procedures being used for evaluating landing sites. We conclude with an experimental analysis of the approach. Conclusions are drawn from the present work, and directions for further study presented.

II. THE HAZARD AVOIDANCE PROBLEM

Hazard avoidance includes three separate critical functions [1]:
- Hazard mapping that estimates ground features based on imaging sensor data (camera or Lidar), and creates hazard maps;
- Site selection that chooses a suitable landing site based on available hazard maps, mission, propulsion and guidance constraints;
- A robust guidance to reach the selected target.

Improving the landing site selection process implies greater onboard autonomy, due to communication time delays and data volume involved. ASTRIUM Space Transportation has been consistently improving the hazard avoidance techniques for on-board piloting autonomy [7], [1] (denoted piloting function).
A. Landing site selection

In this work we focus our attention to the site selection problem. The goal is to provide an adequate target-landing site, evaluated with respect to a set of requirements:

- The site should be safe in terms of maximum local slope, light level and terrain roughness;
- The site should be reachable with the available fuel;
- The site should be visible from the camera along the descent phase.

In the final phase of descent, starting at around 2.7 km above the surface, the lander is continuously mapping and evaluating the terrain below it according to a set of criteria (e.g. Figure 2 depicts the shadow criterion). It then feeds that information to the site selection component, which will in turn provide the navigation system with the landing coordinates. At all times there is a degree of uncertainty with the evaluations of each site, that decreases as the lander approaches the surface.

III. SEARCH METHODOLOGIES

A. Motivation

In previous work, using the Multi-Attribute Decision approach [4], [1], pixels on the maps were seen as alternative landing sites, and evaluated exhaustively. Though a feasible solution on a regular PC, the time available on the onboard processor is already exceeded with just the preliminary data preparation task. We have a relatively small number of alternatives (around 260000), nevertheless the amount of time needed to evaluate them is still too high.

The solution we propose here is the application of non-exhaustive search methodologies that try to locate the best site using an intelligent navigation through a minimal sample of the set of alternatives. We trade off absolute certainty in finding the best site, for a very significant reduction in computational time, and are interested in learning how that affects the quality of the system’s choices.

As we will see below, the quality function, after normalizing and aggregating the criteria, generates a very regular search space (see Figures 3 and 4), with a gradient pointing in the direction of better sites. That structure is currently not being exploited to guide the search. Another feature that makes this a viable approach is the existence of functions for converting between site’s terrain (3D) and image (2D) coordinates. This allows us to map easily between pixels in different maps. Given there is a significant similarity between the search spaces encountered by the lander in successive iterations along the descent, we can use those functions to start immediately the search in a very good region on the map.

The proposed approach can also contribute in different ways to the performance of a hazard avoidance system. A procedure that starts the search at already fairly good sites, and gradually improves its choice is able to produce an answer at any time, with an error rate that decreases the longer the procedure is allowed to execute. Should the guidance system, at some point in the descent, require an answer before the end of the time allocated to the site selection process, it is able to give an adequate one. This is an advantage over procedures that have to execute a greater number of steps before any solution can be returned. An additional gain in performance can also be obtained through a tighter integration of the hazard avoidance and site selection systems: some hazard maps such as the reachability and fuel costs need not be generated for all sites anymore, but can instead be evaluated as sites are visited by the search algorithm.

B. Algorithms

All algorithms implemented, with the exception of Particle Swarm Optimization, are based on the same type of neighborhood structure, i.e., they are based on the same type of movement (movement being defined as the process of switching from one solution to another). In this work we use a simple movement that consists of going to any position on the search space that is at a distance of one unit from the current site.

1) Random Walk: This algorithm randomly chooses a neighbor to the current solution, and indiscriminately moves to it [6]. This algorithm was implemented as a baseline in performance for the problem at hand.

2) Hill Climbing: The classic local search algorithm [6]. It also randomly selects a candidate solution from the current neighborhood at each step, but unlike Random Walk, the candidate is only accepted as the new solution if it is better than the current one.

3) Steepest Ascent: A variant of Hill Climbing. Steepest Ascent [6] differs in the fact that the whole neighborhood is evaluated, and the algorithm deterministically moves to best solution in it if its quality exceeds that of the current solution.
4) Great Deluge: This algorithm was initially proposed [8] as an alternative to Simulated Annealing. The rational for this method is the following: it begins with an initial “water level” equivalent to the initial solution quality and a pre-selected rate to increase it. At each step, the algorithm randomly chooses a neighbor to the current solution, and moves there, but only if its quality is above the rising water level. The algorithm implemented in this work is based on [9].

5) Tabu Search: Tabu search (TS) [10], [11] is a meta-heuristic that has successfully been applied to find good feasible solutions for hard optimization problems. In general it can be described as a neighborhood search method incorporating techniques for escaping local optima and avoid cycling. A basic TS comprises the following concepts:

- Initial Solution - can be obtain randomly or using a more sophisticated method;
- Movement - defines how the search goes from one solution to another and also defines the neighborhood structure;
- Tabu List - contains movements that are considered taboo;
- Aspiration Criteria - may revoke tabu status.

A simple run’s step may be described as follows. Exhaustively evaluate all neighboring points of current solution and select the best one, amongst those that are not considered tabu. Search proceeds from there and the tabu list is updated.

The tabu list will contain all solution that are considered tabu. Solutions stay there for a period equal to the tabu tenure value, one the system’s parameters.

We implemented an aspiration criterion by default, that is applied when in one iteration all neighboring solutions are considered tabu. Instead of terminating the search, the tabu status of the “less” tabu solution is revoked, i.e., the one that has been tabu for a longer period leaves the tabu list.

6) Particle Swarm Optimization (PSO): In the Particle Swarm Optimization [12], [13] algorithm we have, as the name indicates, a swarm of n particles collectively exploring the search space. Each particle \( i \in \{1, \ldots, n\} \) is defined by \( \vec{x}_i \), its current position on the search space, \( \vec{p}_i \), the best position visited by the particle so far, and \( \vec{v}_i \), its velocity.

At each step, the particle updates its velocity vector, and afterwards updates its position in the search space with \( \vec{x}_i \leftarrow \vec{x}_i + \vec{v}_i \).

The core of the PSO algorithm lies in the equation used to update particles’ velocity. Though several alternatives exist, the canonical algorithm uses the equation with “constriction coefficients”:

\[
\vec{v}_i \leftarrow \chi \left( \vec{v}_i + \vec{U}(0, \phi_1) \otimes (\vec{p}_i - \vec{x}_i) + \vec{U}(0, \phi_2) \otimes (\vec{p}_g - \vec{x}_i) \right)
\]

where \( \vec{U}(0, \phi) \) represents a vector of random numbers uniformly distributed in \([0, \phi]\) which is randomly generated at each step and for each particle, and \( \otimes \) is the component-wise multiplication. \( \vec{p}_g \) refers to the best position visited so far by the particle’s neighbors. The constriction factor \( \chi \) is usually set to 0.7298, along with both \( \phi_1 \) and \( \phi_2 \), also called the acceleration coefficients, set to 2.05 [14].

An important concept in PSO is the population topology. Particles are arranged in a graph, and only interact with their immediate neighbors. Classical topologies that have been used with PSO are the gbest (“global best”) and lbest (“local best”) topologies. gbest is basically a fully connected graph, where all particles influence each other. In gbest, information on best positions found by each particle is immediately available, and convergence is faster. In lbest, using a radius of \( k \), particles are arranged in a ring, and particle \( i \) is connected to each particle in \( \{i \pm j \mod n : j = \pm 1, \pm 2, \ldots, \pm k\} \).

Though slower to converge, lbest is less vulnerable than gbest to getting stuck in local optima [12].

IV. QUALITY FUNCTION

All criteria, corresponding to the hazard maps, are given in a matrix structure of 512 by 512 candidate landing sites. The slope map provides the minimum angle to be expected at each pixel, in degrees. The texture map provides the variance value for each pixel. The shadow map or camera image provides the values of light in a grey scale (0-255). The fuel map provides the fuel values, in kg, that are necessary to reach the spot corresponding to each pixel. The reachability map provides values between 0 and 1, meaning that for 0 value the site is not reachable and all other values until 1 mean reachable with a certain degree. The distance map provides the Euclidean distance between the current target and each site on the image.

Observing the different hazard maps provided, we can see that we have both qualitative and quantitative variables and we need to convert and normalize the data so we can compare them. In our case, we have one more particularity: the data has some degree of uncertainty. To deal with the two problems just mentioned we “normalize” the data by using fuzzy sets [2], [15]. The rational for this approach is the following.

We define a fuzzy set that represents the linguistic concept [16] that is considered best to represent the ideal alternative. For instance, the ideal site is the one that has “high remaining fuel”, “low slope”, “low texture”, “sufficient light”, “high reachability” and “low distance”. This approach guarantees normalized and comparable data definitions, besides allowing us to represent data with linguistic concepts. To normalize the slope and shadow criteria we use trapezoidal membership functions. For the fuel, texture and distance criteria we use gaussian membership functions. The reachability criteria already has values between 0 and 1, hence we considered that this map does not need normalization.

With our attributes already normalized we can apply the aggregation operator to evaluate each site. The function implemented is based on the generalized mixture operator. In this particular case we use a weighted sum, where the weights depend on criteria satisfaction levels, instead of having the typical constant weights. The generalized mixture operator.
used is defined below [17], [18]:

\[ W(x) = \sum_{i=1}^{n} w_i(x) x_i \] (2)

where \( w_i(x) = \frac{f_i(x_i)}{\sum_{i=1}^{n} f_i(x_i)} \), \( i = 1, \ldots, n \), and \( x \) is the vector with information about the criteria values.

We compute the weights \( w_i \), that express the relative importance of each criterion, using the following linear weighting functions:

\[ f_i(x_i) = \alpha_i \frac{1 + \beta_i x_i}{1 + \beta_i}, \quad i = 1, \ldots, n \] (3)

where \( \alpha_i, \beta_i \in [0, 1] \) [17].

As mentioned before the quality function was already studied and detailed on previous works [4], [1] related to hazard avoidance problems. See Figure 3 for a complete representation of the search space generated by this quality function. Observing the image surface we can see a strong gradient, which is very positive when using search method that are gradient-based; however some local optima can be found.

V. EXPERIMENTAL ANALYSIS

A. Experimental Design and Setup

Our goal with the present study is to understand how the different search methodologies perform on the site selection problem, when using the quality function previously described. For that purpose, we evaluate their performance in simulated landings, where the algorithms have to provide the lander at each iteration during the descent with the coordinates of the best site, given the current observations.

The simulated landings used hazard maps provided by EADS Astrium, depicting two distinct planetary surfaces. The CRATERS dataset, shown in Figure 2, represents a highly craterized surface, where good landing sites are interspersed by many hazardous features, like high slope or texture. The DUNES dataset represents a smooth planetary surface, punctuated by a high number of dunes, that make sets of contiguous sites invalid choices.

Given that some of the criteria’s evaluations are dependent on the lander’s history up to that point, measures had to be taken to ensure that the search spaces encountered by each algorithm along the descent were the same. Therefore, we defined as current target at the beginning of each iteration the best site from the previous iteration, even if the search algorithm could not find it then (this information being provided by the “oracle” described in the next section). In the first iteration, the coordinates of the site in the middle of the image are used instead. Being a population-based search procedure, PSO implements this specification in a particular way. One particle in the swarm is placed in the lander’s current target, but all others are randomly initialized in the search space, as usual. Furthermore, we also set the lander to always perform a retargeting after each iteration towards the best site on the current map, thus assuring that the descent trajectory is always the same. Under these conditions, we guarantee that the experimental conditions remain the same for all search algorithms, in all iterations of the used datasets.

Several measures were also taken to reduce the number of variables involved (simplifying the analysis), while trying to keep conditions as close as possible to the ones produced by our full system. For instance, in this paper we are only considering the search algorithms’ ability to select the best site, and not yet producing and dealing with lists containing the \( k \) top sites from previous iterations, as done in [4].

The stopping criterion for all algorithms was defined as the evaluation of 2621 sites’ quality, which correspond to 1% of the total number of sites on the image. When using our exhaustive approach [1], a filtering step performed at the beginning is able to remove around 40% of the sites from further consideration. The computational cost of evaluating the remaining sites is however still too high. By setting this parameter to 1%, we aim for a significantly lower
computational cost, that places this approach at a clear disadvantage. That way we’ll be better able to determine the algorithm and parameter tuning that best compares with the exhaustive approach. The final system will likely perform a higher number of evaluations, though that will depend on hardware constraints that aren’t completely defined at this point.

All experiments with the PSO algorithm used the values for the constriction factor and acceleration coefficients parameters presented in Section III-B.6. Several combinations of values were tested in other parameters (e.g. swarm size ∈ {9, 16, 25} and neighbourhood topology ∈ {gbest, lbest − 1, lbest − 2}). In the experiment using the Tabu Search algorithm, the tabu tenure value was set to 250. These are values which are frequently used in the literature.

B. Results

The algorithms’ performance is evaluated in terms of ranks (see Figure 4). Prior to the experiments, an “oracle” with complete knowledge of the search spaces was generated for each dataset. For each iteration along the descent, an exhaustive evaluation of all sites was performed. Then, quality values were sorted, and ranks assigned to each site based on its quality value’s position in the sorted list. The best site is assigned a rank of 0, and the worst site a rank of 512^2 − 1 (unless several sites share that same quality value, in which case they all get assigned the rank of the first site among them to occur on the sorted list). A site’s rank can be interpreted as the number of sites on the map better than itself. This gives us a basis for comparing performance that is independent of the dataset and the iteration. Note that the search algorithms only see the values returned by the quality function. The ranks are only used for a posteriori analysis of the results.

In each iteration of the descent the search algorithm will run for 2621 steps, each step corresponding to the evaluation of one site. We track the algorithm’s progress by measuring at each step the rank of the best site it has found so far. At the end of the descent, we average the values collected in each iteration, and it is this time series that constitutes the basic unit for evaluating the algorithms’ performance.

In Figure 5 we can see the performance of the several tested algorithms, as they converge to the best site with each additional evaluation. The presented time series, corresponding to the non-deterministic algorithms (which all are, with the exception of Steepest Ascent and Tabu Search), are the average of 100 simulated landings.

In Table I we see the average and standard deviations of the ranks of the best site found by each search algorithm within the limit of 2621 site quality evaluations.

C. Analysis

In this section we analyze the results depicted in Table I and Figure 5.

Random Walk, originally implemented to provide a lower bound on performance, obtained very good solutions. That is due to the careful selection of the site at which search starts, described in Section V-A. Being that site in the previous iteration’s best region, we are already able to guarantee a minimum acceptable level of performance. A Random Walk, starting in the middle of a good region, more often than not ends up stumbling on good sites in its neighborhood as it moves. Being blind to sites’ quality, it succeeds where the next algorithms fail.

Hill Climbing, and its variant Steepest Ascent, surprise by their poor performance. This is especially evident on the DUNES dataset, that despite corresponding to a smoother planetary surface, presents many obstacles (the dunes) to these local search algorithms. This is due to the structure of the search space, that despite the apparent smooth gradient that can be observed in Figures 3 and 4, contains enough irregularities, or local optima, for blocking the progression of the algorithms.

The Great Deluge algorithm showed a moderate performance level, though its capacity to escape local optima proved still insufficient.

Tabu Search was an early winner in our experiments. Its tabu list proved to be the needed mechanism for moving around in these search spaces, unhindered by the irregularities in them that hampered the other algorithms’ performance. Tabu Search is however also benefiting from the same careful positioning of the starting location that gives Random Walk its good performance. On a more general level, all the previous approaches are still incomplete solutions, for they lack a needed global exploration capability. Different regions may, in time, become superior choices, but because of their starting locations, the local search algorithms are very limited in their capability to detect those changes. We could add random-restart strategies to these algorithms, by which they would be randomly reinitialized when they get stuck in the search. Those different “epochs” would however be independent of each other, and would not benefit from the...
knowledge the algorithm had previously acquired. Instead, our focus shifted to PSO, which implements a more elegant solution to this problem.

Particle Swarm Optimization showed very good levels of performance. In the majority of iterations, it was able to find the absolute best site, and when it did not, the alternative provided had a rank very close to 0. It always outperformed Tabu Search in the DUNES dataset. In the CRATERS dataset, it can also outperform Tabu Search, conditional to the correct parametrization. It is interesting to note that while all other algorithms performed considerably worse in DUNES than in CRATERS, PSO actually improved its performance. This indicates PSO might be a good general solution for this problem.

As expected, PSO configurations with the \textit{gbest} topology converge faster than those with the \textit{lbest-k} topology, but fail to reach the same quality level. It was expected during implementation that better results would come out of the PSO configurations with swarm size 9. A smaller swarm size, with the same limit in the number of quality evaluations means particles are updated a greater number of times. Experiments showed the opposite. The greater the swarm size, the better the quality tended to be, even though convergence was slower. The PSO algorithm benefited from having additional particles sampling the search space (and informing their neighbors), even though collectively particles were updated a smaller number of times.

VI. CONCLUSIONS

In this paper we present an alternative method to the Multiple Attribute Decision approach for selecting the best site to land a spacecraft. This new approach consists of using search methodologies to iteratively explore potentially good sites instead of evaluating all of them. Several search methodologies were implemented and tested, namely, Hill Climbing, Steepest Ascent, Great Deluge, Tabu Search and Particle Swarm Optimization.

Results show Particle Swarm Optimization to be able to consistently produce solutions of very high quality, from the evaluation of a small set of alternatives. Its regularity in different datasets suggests it may also be the general approach that was sought.

A. Future Work

Having identified in PSO a very efficient algorithm for this problem, we plan to extend our studies with analysis of its performance when using known variations on components like the neighbourhood topology and the velocity update equation.

The different iterations along a descent are currently seen as mostly independent search problems. When switching between iterations, we could map particles’ current position as well as memory position vectors to locations in the new
map, and re-evaluate them. We would then see the whole descent as a single search problem, in a dynamic environment where the quality function changes over time. We would gain better capability to track moving optima, at the cost of decreased global exploration capability.

As future work we also plan to test an hybrid approach, combining Particle Swarm Optimization and Tabu Search. It is hoped the hybridization will provide PSO’s capacity for global exploration, along with Tabu Search’s performance on local exploration.

ACKNOWLEDGMENT

This work was partially financed by EADS-Astrium Space Transportation under contract ASTRIUM-4572019617, and by ESA under contract ESTEC 21744/08/NL/CBI.

REFERENCES


