

Route planning with GA^*

Brian Logan and Riccardo Poli

School of Computer Science, University of Birmingham
Birmingham B15 2TT UK

{B.S.Logan,R.Poli}@cs.bham.ac.uk

ABSTRACT

In previous work we proposed a new evolutionary algorithm, GA^* , which incorporates features of both the classical search algorithm A^* and genetic algorithms. In this paper we describe the application of GA^* to a hard optimisation problem, route planning in complex terrains for Computer Generated Forces (CGF). We report the performance of the algorithm on a large number of route-planning problems and compare its performance with that of a standard GA and classical search techniques. Our results indicate that the plans produced by GA^* are comparable in cost with those produced by a standard GA but require an order of magnitude fewer fitness evaluations, resulting in a significant speedup.

I. INTRODUCTION

In previous work [11; 12] we analysed the relationships between classical, numerical and evolutionary search techniques. Our analysis allowed us to identify and list the important components of search algorithms in an ‘evolutionary computation cookbook’ and suggested how to combine these components to generate new algorithms. Based on this work we proposed a new evolutionary algorithm, GA^* , which we believe incorporates the best features of both A^* , a well-known classical search algorithm, and genetic algorithms.

In [11] we presented some preliminary results of applying GA^* to a hard optimisation problem, route planning for Computer Generated Forces. Computer Generated Forces (CGF) are software agents which simulate the behaviour of military units or equipment in a distributed interactive simulation environment. Such agents must be able to plan routes in complex terrains which include both continuous and discrete features based on uncertain and changing information and under time pressure. In this paper we present a more rigorous evaluation of GA^* , based on an analysis of its performance on a large number of route-planning problems, and compare its performance with that of a standard genetic algorithm and classical search techniques.

The paper is organised as follows. In Section 2 we describe the GA^* algorithm. In Section 3 we discuss the problem of route planning for CGF and highlight some of the problems of existing planners. In Section 4 we outline a novel approach to route planning, based on the idea of plan refinement by continuous deformation, which overcomes some of these problems. In Section 5, we report the

results obtained using this approach together with both GA^* and a standard GA to solve 50 planning problems in a sample terrain model. Finally, in Section 6 we summarise the results and identify a number of directions for future research.

II. THE GA^* ALGORITHM

A^* is a well-known search algorithm [2] for finding the shortest path in a graph. Many forms of problem solving can be viewed as search in a graph in which the nodes represent (possibly partial) solutions to the problem to be solved and the arcs represent steps between solutions. For example, the nodes may represent cities and the arcs roads; or the nodes may be positions in a game and the arcs the legal moves.

Starting from one or more *initial nodes*, A^* searches for a *goal node* which satisfies a goal condition. The search proceeds by applying one or more search operators to the initial node(s) to produce one or more successor nodes which are added to a list of candidates for further expansion. At each cycle the lowest-cost unexpanded node is chosen for further expansion. A^* uses a cost function of the form $g(n) + h(n)$, where $g(n)$ is the cost of going from the initial node to node n , and $h(n)$ (the heuristic function) is an estimate of the cost of reaching the goal node from node n . The search continues until a goal node is found or all the nodes in the graph have been visited. To prevent loops and avoid wasting effort, the algorithm maintains a list of unexpanded or candidate nodes (the open list) and a list of nodes which have already been visited (the closed list). The A^* algorithm is summarised in Figure 1.

-
1. Initialise the open list with the starting node(s) and evaluate them.
 2. Repeat until the open list is empty:
 - (a) Remove the first node from the open list and add it to the closed list.
 - (b) Check if the termination criterion is satisfied (e.g. the node represents a goal node), if so stop.
 - (c) Expand the node by applying the search operators.
 - (d) Reject any invalid successors (e.g. those already in the open list or in the closed list).
 - (e) Evaluate the remaining successors and insert them in the open list in order of cost.
-

Fig. 1. A^* search algorithm.

-
1. Initialise the open list with the starting node(s) and evaluate them.
 2. Repeat until the termination criterion is satisfied:
 - (a) Select an expansion operator according to a probabilistic or deterministic strategy.
 - (b) Select the necessary number of nodes from the open list using rank selection with probability p .
 - (c) Apply the operator and, when appropriate, add the first of the selected nodes to the closed list.
 - (d) Reject any invalid successors (e.g. those representing nodes already visited)
 - (e) Evaluate the remaining successors and insert them in the open list in order of cost.
-

Fig. 2. GA^* search algorithm.

A^* is both complete (i.e. given sufficient resources, it is guaranteed to find a solution if there is one) and optimal (i.e. it is guaranteed to find the best solution). Among optimal algorithms of this type—algorithms that search outwards from the initial node(s)— A^* is optimally efficient for any given heuristic function $h(n)$ which underestimates the real cost of going from n to a goal node, in the sense that no other optimal algorithm is guaranteed to expand fewer nodes than A^* . For large real-life problems, the memory requirements of A^* mean that it is impractical in its pure form, and in most cases it is necessary to use one of the ϵ -admissible variants of A^* such as A_ϵ^* [8], which are guaranteed to find solutions that can be worse than optimal by at most ϵ .

GA^* is a generalisation of A^* inspired by our previous work [11; 12] in which the operators are not constrained to be unary (i.e. an operator can take more than one node as input) and selection of the operators and the next node(s) to be expanded is probabilistic rather than deterministic. GA^* uses a selection procedure, similar to rank selection in GAs, controlled by a parameter p which determines whether the current best unexpanded node should be selected for expansion or whether one of the next best should be considered. The algorithm is shown in Figure 2.

Clearly, with a selection probability of 1 and only unary operators, GA^* is equivalent to A^* . However, in general, GA^* is a hybrid of EAs and classical search methods which incorporates the best features of both as described below.

Firstly, the expansion list (i.e. the population) in EAs is usually limited and cannot grow. This means that EAs are incomplete search procedures. Although this also happens in classical search, a lot of effort has been devoted to designing algorithms that keep as much information as possible on the past search. In GA^* this information can be used very effectively to drive the future search (via crossover and backtracking) and also to avoid wasting computation by reconsidering the same solution more than once. This also maintains the diversity in the population.

Secondly, classical and numerical search methods only consider unary expansion operators. One of the major

sources of power of EAs derives from their use of binary operators (crossover) which are usually based on the idea of building blocks. Their introduction in GA^* can significantly improve the power of the technique (especially when $h(n)$ is not an underestimate). Operators with arity greater than two could provide additional benefits.

Finally, selection in most classical algorithms is deterministic while in most EAs it is probabilistic. When certain conditions on the quality measure are satisfied deterministic selection can lead to optimal expansion strategies which guarantee, for example, minimum memory requirements, minimum number of expansions, optimum use of the available memory, etc. On the other hand probabilistic selection also leads to important forms of optimality, like the optimum exploration/exploitation tradeoff, i.e. the optimum compromise between the need to sample the search space to collect information and the need to produce good solutions as soon as possible (e.g. at runtime in an anytime algorithm). For this reason we have used probabilistic selection in GA^* .

III. THE PROBLEM: ROUTE PLANNING FOR CGF

Computer Generated Forces (CGF) are software agents which simulate the behaviour of military units or equipment in a distributed interactive simulation environment. Such systems are becoming increasingly important in areas as diverse as staff training and equipment procurement. For example, CGF agents offer the potential of dramatically reducing the cost and complexity of mounting training exercises for commanders, with many of the functions carried out by large teams of human controllers replaced by intelligent software agents. Such agents must achieve their goals in complex, uncertain and changing environments.

Route planning in ‘realistic’ terrain is a critical task for CGF agents, as many of the agent’s higher-level goals can only be accomplished if the agent is in the right place at the right time. The problem can be viewed as one of finding a minimum-cost (or low-cost) route between two locations in a digitised terrain model which represents a complex terrain of variable altitude. The cost of a particular route is typically a complex function of factors such as the distance travelled, the time required to execute the plan and the visibility of the route. The problem is complicated by the non-linearity of the cost of going from one point to another which varies with the magnitude and the sign of the local gradient (e.g. moving downhill costs much less than moving uphill) as well as the distance travelled.

A^* in its various forms has been used in a number of CGF systems, for example for planning road routes [1], avoiding static obstacles [5], avoiding moving obstacles [4] and for planning concealed routes [6]. However these systems suffer from a number of problems. They typically work by incrementally extending an initial partial plan until the goal node is reached. Starting from one or more fixed points (often the start point or destination), the planner successively selects atomic plan steps on the basis of their cost and/or the estimated cost of completing the plan. In the case of discretised terrain models, the plan steps are often

taken to be the straight line segments connecting the cell centres (though interpolation is also used, see for example [3]). This can work well if the resolution of the terrain model is adequate, but results in artifacts in the case of coarse grained models. Moreover these planners are typically incapable of repairing a plan following a change in the environment that invalidates the unexecuted portion of the current (partial) plan.

IV. THE REPRESENTATION

We have therefore developed an alternative approach which is based on searching the space of complete plans. Rather than incrementally extending a partial plan, we start with a complete plan and refine it by deforming it. We use a novel representation for plans based on the idea that a complete path between any two points A and B can be considered as the result of applying a set of deformation functions to some initial path connecting A and B. If the deformation functions are orthogonal (or at least linearly independent) any plan can be obtained as a unique linear combination of such functions. For example, in the experiments described below, we used the set of eleven independent triangular plan deformation functions shown in Figure 3.

This approach allows a very compact plan representation, since we only need to remember a fixed length vector of coefficients for each plan (one for each deformation function). As a result, the actual route is only implicitly represented and in particular there is no list of points traversed by the plan.

Starting with a initial plan (e.g. the straight line segment connecting A and B) the search proceeds in a manner similar to best-first search. New plans are generated by changing the deformational coefficients of the linear combination of deformations representing the parent plan. At each iteration the deformation functions are applied to the unexpanded plan with the lowest cost. Usually, this is a successor (deformation) of the current plan. However if the current plan is a local minimum, an exhaustive search of the list of unexpanded plans will result. (Since we are

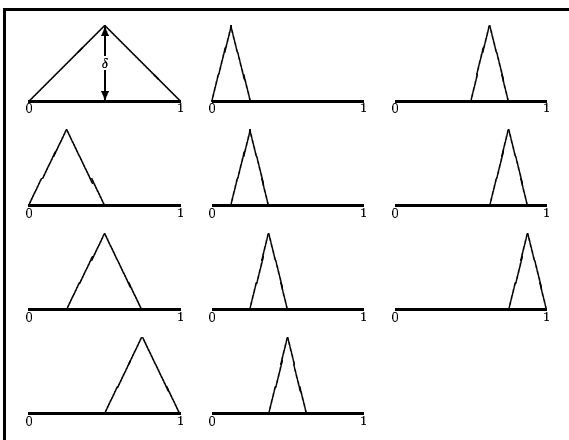


Fig. 3. Eleven independent deformation functions.

searching in the space of plans and hence have no clearly defined goal state, some blind search is inevitable.) The search terminates when a user-specified expansion limit, n , is reached.

The coefficients of the linear combinations of plan deformation functions map naturally into the the state representations used by GA^* and the finite-length chromosomes of a GA, and our approach has the advantage that the planner is ‘anytime’ in that the planner can always return the best plan found so far. Moreover, plan repair in response to changes in the agent’s environment is straightforward, as the deformation operators can be applied directly to the current plan.

V. PRELIMINARY RESULTS

In this section we briefly describe the performance of GA^* in planning routes in a 256×256 grid of spot heights representing a $2\text{km} \times 2\text{km}$ region of a synthetic terrain model¹ and compare its results with the plans produced by a standard GA. We used 50 randomly generated problems consisting of pairs of start and destination positions which were at least four cells apart.

In our experiments, we used a simplified version of GA^* (a best-first version with $h(n) = 0$ and no crossover) and twenty two search operators which increment or decrement by a fixed amount, δ , the deformation coefficients of the plan being expanded. The selection probability, p , was 0.1.

For the GA we used a binary representation in which each plan-deformation coefficient could vary in the range $[-1,+1]$. Each parameter was represented by 5 bits, which give a search resolution of 0.0625 (slightly greater than that available to GA^*). The algorithm is a generational GA with population size 200. It used one-point crossover with probability 0.7, mutation with probability 0.01 (per bit) and tournament selection with a tournament size of 7. The GA was run for 50 generations, which corresponds to approximately 8,500 fitness evaluations per run.

For the purposes of comparison, we also solved the same problems using an A_ϵ^* planner similar to those described in the CGF literature with $\epsilon = 0.1$, i.e. the plans produced are guaranteed to be within 10% of the optimum.

All planners used the same cost function which takes as its input a list of cell coordinates in the discretised terrain model. In the case of GA^* and the GA, we sampled the plan to produce the list of coordinates. In the case of A_ϵ^* the operators encode motion from cell to cell and the resulting list of cells can be used directly as input to the cost function. The A_ϵ^* heuristic function, $h(n)$ was assumed to be the cost of the straight line plan from the current position to the destination.

The resulting plans contained on average about 140 individual steps. Figure 4 shows a typical plan between the points (17, 127) and (244, 30) in the terrain model generated by GA^* after 25 expansions (lighter grey levels represent higher altitudes).²

¹We are grateful to Richard Penney and Jeremy Baxter at DRA Malvern for providing the data.

²Note that to aid the presentation, the z values shown in Figure 4

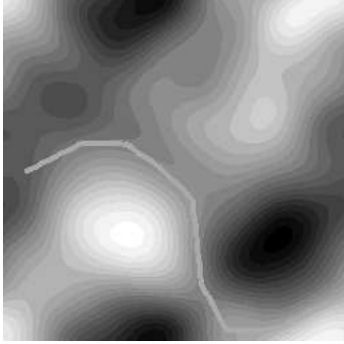


Fig. 4. An example route produced by GA^* .

Table 1 shows the average cost of the plans produced by GA^* for various values of the expansion limit n . The average cost of the plans produced by the GA was 2631. The average cost of the plans produced by A_ϵ^* was 2335, giving an absolute lower bound cost estimate of 2124.

Expansions	5	25	50	100	500
Plan cost ($p = 0.1$)	3113	2703	2614	2604	2604
Plan cost ($p = 1.0$)	2852	2675	2675	2675	2675

Table 1. average cost of plans produced by GA^* .

As can be seen, GA^* produces plans which are, on average, within 25% of the (inferred) optimum. By comparing these results with those obtained by setting the selection probability, p , to 1.0, i.e. turning GA^* into a best-first search algorithm, we can see that rank selection prevents GA^* from becoming trapped in local minima, and the completeness of the algorithm guarantees that even if it is temporarily trapped by a minimum it eventually explores other parts of the search space.

VI. CONCLUSION

In this paper, we have outlined a new search algorithm and described an implementation of the algorithm which plans routes in continuous terrains. Our results show that the plans produced by GA^* are comparable in cost with those produced by the GA and within 25% of the optimum. However GA^* requires on average only 939 evaluations per plan whereas the GA requires approximately 8,500 evaluations per plan. This translates into an order of magnitude speedup for GA^* over the GA, as most of the resources used by both algorithms is spent performing fitness evaluations.

For agents in an uncertain and changing environment, planning in the space of complete plans has a number of advantages over the ϵ -admissible planners described in the CGF literature. Our approach is inherently anytime, in that the planner can always return the best plan found so far. In addition, plan repair in response to changes in the agent's environment can use the current best plan as

the starting point directly without having to estimate how much of the existing plan can be reused.

However the current implementation has a number of limitations. The current set of deformation functions is not well suited to planning routes around discrete obstacles. Moreover plans can never extend beyond the normals to the line connecting the start and end points of the plan at the endpoints.³ However, there is no reason why the set of basis functions could not be expanded to include step or other functions, splines, polylines, or even orthogonal deformation procedures which would behave differently according to the terrain features. Such functions might be enabled and disabled by some more abstract examination of the terrain model and/or the progress of the planner. In addition, the current implementation of the planner is rather slow, as each operator application typically requires costing an entire plan, rather than computing the incremental change in cost resulting from the operator application as with A_ϵ^* . We are currently investigating plan representations that allow us to cost operator applications rather than the resulting complete plans. This problem is currently unsolved.

We are hopeful that these limitations can be overcome and believe that this is an area worth exploring. We are currently investigating a number of extensions to the basic framework described above, including alternative search strategies, e.g. varying plan deformation factor δ with the depth of the search, using different perturbation functions at different times, and using hierarchical search, i.e. sampling the plan produced after a small number of expansions and using the planner to plan routes between the endpoints of the resulting plan segments. Another area which we hope to investigate is plan reuse or case-based planning, to exploit the ability of the perturbation operators to adapt existing, similar plans (for example plans with similar start and end positions, and similar cost functions) to solve new problems.

ACKNOWLEDGEMENTS

The authors wish to thank Aaron Sloman and all the members of the Cognition and Affect and EEBIC (Evolutionary and Emergent Behaviour Intelligence and Computation) groups at the School of Computer Science, University of Birmingham for useful discussions and comments. This research is partially supported by a grant from the Defence Research Agency (DRA Malvern) and a grant under the British Council-MURST/CRUI agreement.

REFERENCES

- [1] C. Campbell, R. Hull, E. Root and L. Jackson. *Route planning in CCTT*, in Proceedings of the Fifth Conference on Computer Generated Forces and Behavioural Representation, Technical Report, Institute for Simulation and Training, pp. 233–244, 1995.

have been discretised into twenty steps. However all tests were conducted using the original 'continuous' model.

³More precisely, the distance along the straight line segment from the start point to the end point is monotonic in d , the distance along the plan.

- [2] P. E. Hart, N. J. Nilsson and B. Raphael. *A Formal basis for the heuristic determination of minimum cost paths*, IEEE Trans. Syst. Sci. Cybern. SSC-4(2), pp. 100–107, 1968.
- [3] B. Hoff, M. F. Howard and D. Y. Tseng. *Path planning with terrain utilisation in ModAF*, in Proceedings of the Fifth Conference on Computer Generated Forces and Behavioural Representation, Technical Report, Institute for Simulation and Training, pp. 255–263, 1995.
- [4] C. Karr, M. Craft and J. Cisneros. *Dynamic obstacle avoidance for Computer Generated Forces*, in Proceedings of the Fifth Conference on Computer Generated Forces and Behavioural Representation, Technical Report, Institute for Simulation and Training, pp. 245–254, 1995.
- [5] C. Karr and S. Rajput. *Unit route planning*, in Proceedings of the Fifth Conference on Computer Generated Forces and Behavioural Representation, Technical Report, Institute for Simulation and Training, pp. 295–304, 1995.
- [6] M. Longtin and D. Megherbi. *Concealed routes in ModSAF*, in Proceedings of the Fifth Conference on Computer Generated Forces and Behavioural Representation, Technical Report, Institute for Simulation and Training, pp. 305–314, 1995.
- [7] P. Norvig. *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp*, Morgan Kaufman, 1992.
- [8] J. Pearl. A_ϵ^* – *An algorithm using search effort estimates*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 4, No. 4, pp. 392–399, 1982.
- [9] J. Pearl. *Heuristics: intelligent search strategies for computer problem solving*, Addison-Wesley, 1984.
- [10] I. Pohl. *The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computational issues in heuristic problem solving*, Proc. IJCAI-73, pp. 20–23, 1973.
- [11] R. Poli and B. Logan. *Evolutionary computation cookbook: recipes for designing new algorithms*, Proceedings of the Second Online Workshop on Evolutionary Computation, Japan, 11–22 March 1996
- [12] R. Poli and B. Logan. *On the relations between search and evolutionary algorithms*, Cognitive Science Research Paper CSRP-96-7, School of Computer Science, University of Birmingham, 1996.