MACHINE LEARNING TECHNIQUES FOR APPROXIMATION OF OBJECTIVE FUNCTIONS IN TRAJECTORY OPTIMISATION

Christos Ampatzis and Dario Izzo

Advanced Concepts Team, European Space Agency, ESTEC, 2201 AZ Noordwijk, The Netherlands

ABSTRACT

Objective function landscape approximation methods in evolutionary optimisation can be a beneficial technique, since they replace part of the calls to the original, often computationally expensive objective function, with calls to a faster and computationally cheaper function approximator, such as a polynomial or an artificial neural network. Moreover, in some cases, the approximate model smoothens rough fitness landscapes, facilitating the stochastic search. In this paper, we apply a neural network-based approximation technique to spacecraft interplanetary trajectory problems. These multimodal problems, recently introduced in the global optimisation community, can be very complex and characterised by the prevalence of many local optima and, in the worst cases, a heavy computation load involved with the calculation of the objective value of a given input vector. We perform the first steps to integrate an approximated model into a trajectory optimisation process, building a hybrid system where both the original trajectory model and the approximated model are carefully used in the optimisation process without degrading the quality of the final trajectory found.

Key words: Artificial Neural Networks, trajectory optimisation, approximation, objective function.

1. INTRODUCTION

The computational cost involved in the solution of global optimisation problems is often directly linked to the very large number of fitness function evaluations needed. Among the many approaches that emerged in the last decades aiming at reducing this cost by using the information gathered during the optimisation process, is making use of an approximation of the original, often expensive, objective function (see [Jin05] for a good review of these methods). These approaches make use of mathematical models or machine learning techniques based on learning and interpolation from original input vector/objective function pairings. Among other machine learning techniques for function approximation (as Support Vector Machines, for example), artificial neural networks (ANNs) have been employed to approximate the objective functions in global optimisation problems [JMS02]. Originally, ANNs were summoned to speed up a potentially expensive optimisation process, as for certain processes, calculating an objective value may take hours or days of computation time. However using an approximate fitness function during the evolution may also lead to better results and a better convergence, because this way a very rugged fitness landscape can be smoothened.

Neural networks are known to be powerful computational tools that are widely used in a very broad range of applications; from stock market predictive models to controlling autonomous robots [NF00, ATT$^+$09]. Loosely inspired by biological neural networks, artificial neural networks are able to efficiently compute complex input/output mappings thanks to their inherent parallelism and simplicity: a single hidden layer feed forward neural network is capable of approximating any continuous, multivariate function to any desired degree of accuracy, provided enough neurons are used [Has95]. When using ANNs to approximate the fitness landscape during an optimisation process, the capability of the network to "learn" the fitness trend is a crucial characteristic of the network that will clearly affect the benefit resulting from its use and that clearly is problem dependent.

In this paper, we study the possibility to use an approximated model during the evolutionary optimisation process of interplanetary trajectories, by presenting some preliminary but very encouraging results. This class of test functions for global optimisation has been recently introduced as such by Vinkó et al. [VIB07] in 2006. Each function, essentially, expresses a spacecraft trajectory performance index (often the final spacecraft mass at arrival) as a function of the strategy adopted to reach the mission goals (e.g., launch date, thrusting location and magnitude, fly-by sequence etc.). A large number of different test function instances can be derived from a rather general description, by specifying different goals or possible strategies (for example, a rendezvous mission to Jupiter will look radically different from a fly-by mission to Pluto, depending on the spacecraft propulsion system adopted, the launcher used or the launch window). It is beyond the scope of this paper to give the exact expression of each function; instead, we just stress here that both its minimal value and its taxonomy vary con-

Proc. 'IJCAI–09 Workshop on Artificial Intelligence in Space', Pasadena, California, US 17–18 July 2009 (ESA SP-673, September 2009)

siderably by changing the function domain, the planet ephemerides and a large number of parameters values (e.g., the strength of the Sun gravity). Thus, it is extremely important, when performing any test, to use standardized instances of the problem. In our experiments, we used some of the problem instances posted in the Advanced Concepts Team Global Trajectory Optimisation (GTOP) database together with the MATLAB code there provided [VIB07].[1]

Trajectory optimisation problems have been demonstrated to have a high degree of complexity [MBNB04]. The fitness landscape for many problems can be high dimensional and it has many local optima, where most global optimisers tend to get stuck. Typically, for such problems, the global optimum is not known in advance. Moreover, some of these problems, especially low-thrust problems, tend to be characterised by computationally heavy calculations involved with the computation of the objective value for a given input vector. For the great majority of these problems an extensive amount of function evaluations is required until an acceptable solutions is found. These characteristics are a good hint that the use of an approximate model may contribute to speed up the optimisation process in terms of computation time; when a large amount of function evaluations is required, the use of ANNs will significantly affect the computational resources required. Arguably, the introduction of the approximate model during the optimisation process creates a new optimisation algorithm with a different performance which needs to be evaluated.

Interplanetary trajectory optimisation problems can be particularly hard to handle because of the issues outlined above, and typically researchers have to resort to complex strategies in order to obtain good results in a reasonable time. One example is parallel architectures, as in [IRA09], where the island-model for global optimisation is used. Other approaches typically include the mix of global and local search [ACLS08], the use of space pruning [IBM$^+$07] or the development of ad hoc algorithms exploiting knowledge of the search space.

This article is organised as follows. In section 2 we provide the implementation details and essentially the optimisation algorithm introduced. In section 3 we describe the three different trajectory optimisation problems we apply our algorithm to, and in section 4, we compare the results of the algorithm making use of the approximation to the algorithm using only the original fitness function. Finally, in section 5 we draw conclusions and propose future directions to be explored.

2. THE IMPLEMENTATION

Our implementation features an ANN as an approximate fitness model, which gets created and trained on-line. This means that we do not waste original fitness evaluations to collect enough data and train a neural network that approximates the original objective function with some arbitrary precision in advance, and then use this to perform the optimisation. Instead, we commence with the optimisation process employing the original objective function, collecting input/output pairs along the way, which we use to adaptively train a randomly created ANN. Once the ANN is trained with the data collected during the first phase, we perform the optimisation with the ANN instead of the original objective function. This loop is continued until a termination criterion is fulfilled.

A very important issue in evolutionary optimisation with approximate fitness functions is avoiding convergence to false optima. According to [JMS02], a general tip for ensuring the correct convergence is to use controlled evolution, both individual and generation based. Generation-based evolution refers to the fact that a part of the total evaluations (or generations) is performed with the original objective function and the rest with the approximate one. On the other hand, individual-based controlled evolution requires that while using the ANN, some individuals in the population are still evaluated with the original fitness function. This may reduce the benefit we gain from using a faster process but it proves to guarantee correct convergence and avoidance of false optima.

The pseudo-code below describes our experimental setup; J refers to the original objective function, $\mathcal{P}$ and $\mathcal{P}'$ refer to the current and the new population created by the evolutionary algorithm used, respectively. We also indicate with $\mathcal{N}$ the neural network approximation to the objective function.

```
create a random population $\mathcal{P}$
while function evaluations are allowed
    evaluate $\mathcal{P}$ with respect to $J$
    for X generations do:
        create $\mathcal{P}'$ from $\mathcal{P}$ using the selected algorithm
        evaluate $\mathcal{P}'$ with respect to $J$
        insert $\mathcal{P}'$ into $\mathcal{P}$
    for Y generations do:
        create $\mathcal{P}'$ from $\mathcal{P}$ using the selected algorithm
        evaluate $\mathcal{P}'$ with respect to $J$
        train $\mathcal{N}$ using $\mathcal{P}'$
        insert $\mathcal{P}'$ into $\mathcal{P}$
    evaluate the worst half of $\mathcal{P}$ with respect to $\mathcal{N}$
    for Z generations do:
        create $\mathcal{P}'$ from $\mathcal{P}$ using the selected algorithm
        evaluate the worst half of $\mathcal{P}'$ with respect to $\mathcal{N}$
        evaluate the best half of $\mathcal{P}'$ with respect to $J$
        insert $\mathcal{P}'$ into $\mathcal{P}$
create $\mathcal{P}'$ from $\mathcal{P}$ using the selected algorithm
evaluate $\mathcal{P}'$ with respect to $J$
insert $\mathcal{P}'$ into $\mathcal{P}$
```

In the experiments we have performed we have used $X = 75$, $Y = 25$ and $Z = 100$. As global optimiser we

[1]GTOP is a database containing the exact definition of some global optimisation spacecraft trajectory problems and their best putative solutions, see `www.esa.int/gsp/ACT/inf/op/globopt/`

have used Differential Evolution [SP97], and in particular $DE/rand-to-best/1/exp$ and $DE/rand/1/bin$ simply referred to as DE3 and DE7, respectively, with the following parameter values:

- Amplification factor $F = 0.8$;
- Crossover constant $CR = 0.8$;
- Population size $NP = 20$.

Every iteration of the Differential Evolution algorithm consists of three phases: mutation, crossover and selection. During the mutation phase, for every individual in the population, a so-called mutant individual is calculated. For the two aforementioned DE variants, the formulae used to calculate the mutant $v_{i,G+1}$ of the i-th individual used to create the generation $G+1$ are the following:

For DE3:

$$v_{i,G+1} = x_{i,G} + F \cdot (x_i - x_{\star,G}) + F \cdot (x_{r_1,G} - x_{r_2,G});$$

for DE7:

$$v_{i,G+1} = x_{r_1,G} + F \cdot (x_{r_2,G} - x_{r_3,G});$$

where G is the iteration number, $x_{\star,G}$ is the best individual found up to iteration G, r_{1-3} are 3 different, randomly selected indices of individuals, and $F > 0$ is the constant amplification factor (one of the algorithm parameters). DE3 uses the exponential crossover strategy and DE7 the binary crossover. Reinsertion is used, that is, the selection method resembles the greedy approach—a new individual replaces the old one in the population if and only if it has a better value of the objective function. In case this value has been calculated with the neural network, the new individual replaces the old only if also its fitness evaluated with the original fitness function has a better value.

The neural network we have decided to use is a Multi-Layer Perceptron (MLP) with two hidden layers, each one composed of 25 neurons. The transfer function of the neurons in the first hidden layer is the logistic function, while neurons of the second hidden layer use the tangent as transfer function. Output neurons are characterised by a linear fitness function. The training algorithm employed for the neural network is the Levenberg Marquardt algorithm [Mar63] as implemented in MATLAB, and the training epochs are fixed to 25 per dataset to be learned. The number of epochs which ultimately defines how well the network approximates the input/output vector pairs has been chosen deliberately to be rather low in order to avoid overtraining. Notice that due to our implementation, the size of the dataset to be learned every time has a fixed size of 20 input/output vector pairs (equal to the population size), where the input vector size is 22 and the output vector is one dimensional and encodes the fitness of the input vector. It is important to notice that the choices outlined above are arbitrary and are meant to provide a first implementation of the new algorithm. In future work, we intend to optimise all decisions involved in the implementation of our system.

3. EXPERIMENTS

In order to assess the performance of the approximate model with respect to the original model, we run the following experiments. Since the global optima are unknown, we cannot use any definition of a successful run, as in finding the global optimum. Instead, we allow a maximum number of function evaluations, set to 1,000,000 or 2,000,000, depending on the problem considered, and we record at the end of the evolution the minimum, mean and maximum of the objective values achieved across 20 runs of the optimisation process. Notice that the optimisation problems considered are minimisation problems, that is, better solutions have lower objective values. In the following we present the three different problems we consider as a test-bed for the developed algorithm.

3.1. MGA Global Optimisation Problems - Cassini 1

A simple benchmark to test global optimisation algorithms in Space Mission Design related problems is the Multiple Gravity Assist (MGA) problem. In mathematical terms this is a finite dimension global optimisation problem with non-linear constraints. It can be used to locate the best possible trajectory that an interplanetary probe equipped with a chemical propulsion engine may take to go from the Earth to another planet or asteroid. The spacecraft is constrained to thrust only at planetary encounters.

The Cassini1 problem is an MGA problem related to the Cassini spacecraft trajectory design problem. The objective of this mission is to reach Saturn and to be captured by its gravity into an orbit having pericenter radius $r_p = 108950$ km, and eccentricity $e = 0.98$. The planetary fly-by sequence considered is Earth, Venus, Venus, Earth, Jupiter, Saturn (as the one used by the Cassini spacecraft). As objective function we use the total deltaV accumulated during the mission, including the launch deltaV and the various deltaV one needs to give at the planets and upon arrival to perform the final orbit injection. The input state vector is six dimensional.

3.2. MGA-DSM Global Optimisation Problems - Cassini2

In Multiple Gravity Assists with one Deep Space Maneuver problems (MGADSM), a spacecraft is required to perform a given mission in the interplanetary medium using impulsive thrusters that are allowed to fire only once between two subsequent planets of a preassigned fly-by sequence. This creates an unconstrained global optimisation problem, as far as further constraints are not explicitly specified. In the case of Cassini2, the target planet is Saturn and the assigned planetary sequence is Earth-Venus-Venus-Earth-Jupiter-Saturn, resulting in a problem dimension $D = 22$. The mission goal is the insertion around Saturn in an orbit similar to the orbit the Cassini spacecraft achieved in 2004. The fact that deep space maneuvers are allowed between each one of the planets leads to a higher dimensional problem with a much higher complexity, with respect to the MGA problem Cassini1. Also, in the objective function evaluation, a rendezvous problem rather than an orbital insertion as in Cassini1 is considered.

3.3. MGA-DSM Global Optimisation Problems - Messenger

This trajectory optimisation problem represents a rendezvous mission to Mercury modelled as an MGA-DSM problem. The selected fly-by sequence is the same used in the first part of the Messenger mission. It is well known that a significant reduction of the required deltaV is possible if a number of resonant fly-bys follow the first Mercury encounter. The input state vector has a dimension of 18.

4. RESULTS

The results for all experiments performed are summarised in tables 1, 2, 3, 4, 5, 6. We performed Welch's unpaired t-test by comparing the means, the maximum, the standard deviation and the amount of samples for the two algorithms; what we observe is that the performance of all hybrid algorithms (using the ANN approximation of either the original mga or mgadsm objective function) is not statistically different from the performance of the respective original algorithm (mga or mgadsm). This is a proof that the introduction of the approximate model does not lead to performance degradation for the algorithm.

Moreover, we observe that the hybrid model is able to find for all experiments performed, along the 20 runs, the same or a better best solution (in terms of minimum value found by the twenty runs). In fact, the solutions discovered are close to the best solutions found to the problems tackled, as reported in the GTOP database mentioned earlier. Even if the performance of the optimisation is not significantly improved, this improvement of the best solution found is an encouraging hint that the approximate model may also introduce an improvement in solution quality rendered, along with the inherent speedup related to faster function evaluations. Interestingly, these improvements are more visible in the case of DE7 than for DE3. This implies that the global optimisation algorithm used influences the behaviour of the hybrid algorithm employing the approximate model. This of course comes as no surprise as every algorithm's performance is dependent on the problem and the algorithm, but a methodological approach to understanding which algorithms seem to be better suitable for approximation models can be very beneficial and will be subject of future work.

We finally investigated the convergence properties of the

Cassini 1, DE7		
	ANN+mga	mga
min	5.034	5.034
max	16.722	12.542
mean	9.450	8.517
std	3.3765	3.001

Table 1. The minimum, maximum, mean and the standard deviation of the best fitness values obtained by 20 runs after 1,000,000 evaluations of the ANN+mga objective function and the original mga function, for Cassini1, using DE7 .

Cassini 1, DE3		
	ANN+mga	mga
min	4.9307	5.0304
max	16.722	16.818
mean	9.576	11.161
std	3.001	2.6371

Table 2. The minimum, maximum, mean and the standard deviation of the best fitness values obtained by 20 runs after 1,000,000 evaluations of the ANN+mga objective function and the original mga function, for Cassini1, using DE3.

Cassini 2, DE7		
	ANN+mgadsm	mgadsm
min	8.6693	10.1859
max	26.2018	21.5495
mean	15.7956	17.1691
std	4.9271	3.4968

Table 3. The minimum, maximum, mean and the standard deviation of the best fitness values obtained by 20 runs after 1,000,000 evaluations of the ANN+mgadsm objective function and the original mgadsm function, for Cassini2, using DE7.

Cassini 2, DE3		
	ANN+mgadsm	mgadsm
min	16.1905	16.538
max	21.3807	24.479
mean	20.4857	20.993
std	1.1956	1.985

Table 4. The minimum, maximum, mean and the standard deviation of the best fitness values obtained by 20 runs after 1,000,000 evaluations of the ANN+mgadsm objective function and the original mgadsm function, for Cassini2, using DE3.

Messenger, DE7		
	ANN+mgadsm	mgadsm
min	8.9835	10.9550
max	19.1289	12.9891
mean	12.1506	11.9721
std	2.3563	1.0435

Table 5. The minimum, maximum, mean and the standard deviation of the best fitness values obtained by 20 runs after 2,000,000 evaluations of the ANN+mgadsm objective function and the original mgadsm function, for Messenger, using DE7.

Messenger, DE3		
	ANN+mgadsm	mgadsm
min	10.0300	10.090
max	14.0660	13.996
mean	12.3546	12.002
std	1.2725	1.102

Table 6. The minimum, maximum, mean and the standard deviation of the best fitness values obtained by 20 runs after 2,000,000 evaluations of the ANN+mgadsm objective function and the original mgadsm function, for Messenger, using DE3.

hybrid algorithm with respect to the original algorithm. In figure 1 we can see for Cassini2 and DE7, the mean, minimum and maximum objective value along the twenty evolutionary runs, as the number of fitness evaluations grows, for the two algorithms. What we can observe is a very similar convergence behaviour, which confirms our claim that the new hybrid algorithms behaves very closely to the original optimisation algorithm. We can see that the evolution of the mean over time is very similar, and the only differences are the slightly lower minimum and higher maximum value achieved by the hybrid algorithm.

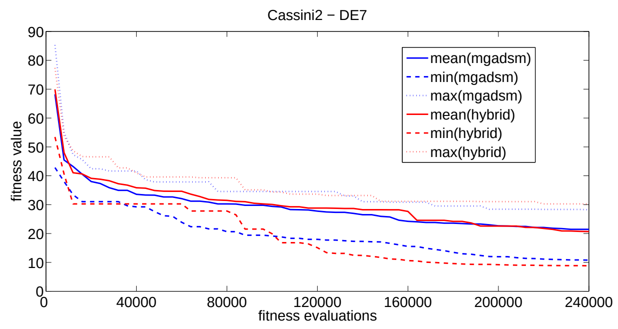

Figure 1. Convergence properties of the hybrid model with respect to the original model for the case of the problem of Cassini2, using DE7.

5. CONCLUSION

In this paper, we have shown how trajectory optimisation problems can profit from the introduction of approximate models. We have shown that hybrid algorithms evolving at times with the original and at times with the approximate objective function (ANN) can be characterised by similar convergence properties and a statistically proven similar performance. However, objective functions evaluations with the ANN are much "cheaper" than those performed with the original fitness functions of the trajectory optimisation problems. The fact that the hybrid model consistently found better solutions than the original optimisation process is also encouraging and provides a hint that the approximation models can also lead to better solutions apart from the evident speedup, provided they undergo the right tuning.

As the implementation and the results presented are rather preliminary, in future work, we intend to study more in depth the impact of the implementation parameters on the final performance, in order to reap the maximum benefit out of the approximate model, without running the risk of wrong convergence. Moreover, we intend to apply this technique to other, different trajectory optimisation problems, such as low-thrust problems. These problems are harder to solve for conventional optimisation techniques since they are on the side of constraint optimisation and since their dimension is usually much higher than problems treating impulsive thrust. Moreover, function evaluations for these problems tend to be more computationally expensive, which makes them suited for approximation models.

REFERENCES

[ACLS08] B. Addis, A. Cassioli, M. Locatelli, and F. Schoen. Global optimization for the design of space trajectories. *Computational Optimization and Applications*, 2008.

[ATT+09] C. Ampatzis, E. Tuci, V. Trianni, A. L. Christensen, and M. Dorigo. Evolving self-assembly in autonomous homogeneous robots: experiments with two physical robots. *Artificial Life*, 15(4):465–484, 2009.

[Has95] M. Hassoun. *Fundamentals of Artificial Neural Networks*. MIT Press, 1995. p.48.

[IBM+07] D. Izzo, V.M. Becerra, D.R. Myatt, S.J. Nasuto, and J.M. Bishop. Search space pruning and global optimisation of multiple gravity assist spacecraft trajectories. *Journal of Global Optimization*, 38(2):283–296, 2007.

[IRA09] D. Izzo, M. Ruciński, and C. Ampatzis. Parallel global optimisation meta-heuristics using an asynchronous island-model. In *Proceedings of IEEE Congress on Evolution Computation (CEC09)*, pages 2301–2308, 2009.

[Jin05] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12, 2005.

[JMS02] Y. Jin, M.Olhofer, and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6(5):481–494, 2002.

[Mar63] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11:431–441, 1963.

[MBNB04] D.R. Myatt, V.M. Becerra, S.J. Nasuto, and J.M. Bishop. Advanced global optimisation tools for mission analysis and design. Technical Report 03-4101a, European Space Agency, the Advanced Concepts Team, 2004. Available on line at www.esa.int/act.

[NF00] S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press, Cambridge, MA, 2000.

[SP97] R. Storn and K. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[VIB07] T. Vinkó, D. Izzo, and C. Bombardelli. Benchmarking different global optimisation techniques for preliminary spacecraft trajectory design. In *Proceedings of the 58th International Astronautical Congress, Hyderabad, India*, 2007. Paper IAC-07-A1.3.01.