A Hooke-Jeeves based Memetic Algorithm for Solving Dynamic Optimisation Problems

Irene Moser[1] and Raymond Chiong[2]

[1] Faculty of ICT, Swinburne University of Technology, Melbourne, Australia
imoser@swin.edu.au
[2] School of Computing & Design, Swinburne University of Technology (Sarawak Campus), Jalan Simpang Tiga, 93350 Kuching, Sarawak, Malaysia
rchiong@swinburne.edu.my

Abstract. Dynamic optimisation problems are difficult to solve because they involve variables that change over time. In this paper, we present a new Hooke-Jeeves based Memetic Algorithm (HJMA) for dynamic function optimisation, and use the Moving Peaks (MP) problem as a test bed for experimentation. The results show that HJMA outperforms all previously published approaches on the three standardised benchmark scenarios of the MP problem. Some observations on the behaviour of the algorithm suggest that the original Hooke-Jeeves algorithm is surprisingly similar to the simple local search employed for this task in previous work.

Keywords: Hooke-Jeeves pattern search, extremal optimisation, dynamic function optimisation, moving peaks problem.

1 Introduction

One of the fundamental issues that makes optimisation problems difficult to solve is the dynamically changing fitness landscapes [1]. In problems with dynamic fitness landscapes, the task of an optimisation algorithm is normally to provide candidate solutions with momentarily optimal objective values for each point in time.

The Moving Peaks (MP) problem is a good example of this kind of problem. It consists of multidimensional landscape with a definable number of peaks, where the height, the width and the position of each peak are altered slightly every time a change in the environment occurs. Created by Branke [2] as a benchmark for dynamic problem solvers, the MP problem has been used by many for the testing of algorithms for dynamic function optimisation.

In previous work [3], an algorithm called Multi-phase Multi-individual Extremal Optimisation (MMEO) was designed. MMEO exhibits great simplicity, but it works extremely well for the MP problem. Thorough analysis [4] has shown that the success story of MMEO is largely due to the local search component in it, and this has motivated us in studying the Hooke-Jeeves (HJ) algorithm, a very simple local search that was proposed by Hooke and Jeeves over 40 years ago [5].

In this paper, we present a new Hooke-Jeeves based Memetic Algorithm (HJMA) for solving the MP problem. HJMA is a hybridisation of HJ pattern search and Extremal Optimisation (EO), an optimisation heuristic that was first introduced by Boettcher and Percus [6] in 1999. Based on a very simple principle of mutating a single solution according to a power-law distribution, EO was designed to exclude bad solutions rather than finding good solutions. In other words, EO was not intended to show any convergence behaviour. This characteristic seemed to make EO a very promising choice for dynamic implementations – as no convergence exists, EO is expected to automatically adapt the current working solution according to the feedback received from the objective function. If the objective function returns fitness values that reflect the current search space, the algorithm is expected to be able to adapt to changes regardless of the severity or the frequency of the changes. However, experimental studies [4] revealed that EO alone does not work well for the MP problem, as far as the solution quality is concerned. The need to check for duplicates during the local search phase has an unexpectedly large impact on the solution quality. As such, this study reports on the hybridisation of EO with HJ.

The rest of this paper is organised as follows. Section 2 introduces the background of the MP problem by describing some of the existing solution methods. Following which, section 3 discusses the HJMA in detail. We then present our experimental results in section 4. Finally, section 5 concludes our studies and highlights some potential future work.

2 Background

The MP problem can be formally defined with the following function:

$$F(\vec{x},t) = \max(B(\vec{x}), \max_{1..m} P(\vec{x}, h_i(t), w_i(t), \vec{p}_i(t))) \tag{1}$$

where $B(\vec{x})$ is the base landscape on which the m peaks move, with each peak P having its height h, width w and location $\vec{p}$. It is necessary to note that the location, width, height, and movement of the peaks are all free parameters. For the purposes of performance comparison, three standardised sets of parameter settings, called Scenario 1, Scenario 2 and Scenario 3 respectively, are defined. Most of the benchmark results have been published predominantly for Scenario 2 with 10 peaks that move at an average distance of 1/10 of the search space in a random direction, mainly due to its appropriateness in terms of difficulty and solvability.

Many solutions for the MP problem have been presented since its inception, and one of the pioneering solutions can be found in the work of its creator, Branke, based on a genetic algorithm (GA) [7]. In his seminal work, Branke used a memory-based multi-population variation of a GA to store and retrieve individuals when a change occurs. However, he found that the approach is less useful for more drastic changes to the landscape. For this reason, he collaborated with others to develop the self-organising scouts algorithm [8]. This extended GA approach is based on a forking mechanism which starts with a single population, and then dividing off subpopulations with a designated search area and size. Comparing to the standard GA, it shows a much better performance. More results of the self-organising scouts approach were subsequently published by Branke and Schmeck in [9], where they also introduced the offline error[1] as a standard metric for performance measure.

Other related works that have applied GA to the MP problem can be found in [10], [11], [12], [13], [14] and [15].

Apart from GA, particle swarm optimisation (PSO) is another popular method that has been used extensively in dynamic optimisation domain. Blackwell [16], who introduced charged particles (hence CPSO) that repel each other and circle around neutral particles of the swarm for better convergence behaviour in dynamic environments, was among the first to study PSO for the MP problem. Afterwards, Blackwell and Branke [17] applied a multi-population version of the same approach as multi-CPSO to the same problem. They also introduced multi-Quantum Swarm Optimisation (multi-QSO), a variation whose charged particles move randomly within a cloud of fixed radius centred around the swarm attractor. All these approaches perform well on the MP problem.

Other PSO-based studies include Parrott and Li [18] who adapted the speciation technique from GA to PSO, Janson and Middendorf [19] who proposed to employ a tree structure where each particle uses the best location found by the individual immediately above it in the tree structure in addition to its own best find, as well as Wang et al. [20] who used Branke's [7] idea of employing 3 populations originally for GA to PSO. However, the best result by PSO comes from Blackwell and Branke [21] who added anti-convergence to the exclusion and quantum/charged particle features they first conceived in [16] and [17] respectively. In [21], Blackwell and Branke reported an offline error of 1.72 from solving Scenario 2.

There are also other types of solutions for the MP problem, such as the differential evolution (DE) [22], the stochastic diffusion search (SDS) [23] inspired by neural networks, and the B-cell algorithm (BCA) [24]. Among these, the DE approach by Mendes and Mohais [22] produced almost equal quality as Blackwell and Branke's PSO [21], with an offline error of 1.75 from solving the same settings.

The good performances of Blackwell and Branke's PSO and Mendes and Mohais' DE have encouraged Lung and Dumitrescu [25] to develop a hybridised algorithm that combines PSO and Crowding DE, called Collaborative Evolutionary-Swarm Optimisation (CESO), in which equal populations of both methods collaborate. Their offline error of 1.38 on Scenario 2 with 10 peaks surpasses those of Blackwell and Branke's as well as Mendes and Mohais'.

While all these approaches are impressive, the best solution in the literature comes in the very simple MMEO algorithm by Moser and Hendtlass [3]. MMEO is a multi-phase multi-individual version of EO. As has been established [4], a large proportion of the quality of its hitherto unsurpassed results is contributed by its local search component, which is rather straightforward and deterministic. It outperforms all available approaches to date with an offline error of 0.66 on Scenario 2.

[1] At each evaluation, the difference between the maximum height of the landscape and the best-known solution to date is recorded. It is then averaged by the number of evaluations (note that this measure has some vulnerabilities: it is sensitive to the overall height of the landscape, the number of peaks, and the number of evaluations before change).

In spite of the outstanding results, the local search of MMEO still carries redundant steps which cause unnecessary function evaluations. Furthermore, the step lengths used in [3] were chosen intuitively without careful consideration. As such, we believe that there is still room for improvements. In this study, we intend to explore the potential of HJ pattern search, and examine how the abovementioned issues could be overcome through the development of HJMA.

3 A Novel Hybrid Approach

In this section, we present our novel HJMA approach. The basic EO algorithm is used as the basis for performing the global search, while HJ will be incorporated during the local search process.

3.1 Global Search

This task is achieved by an adaptation of the EO algorithm. Unlike other population-based approaches, EO works to improve only a single solution using mutation. This solution consists of multiple components which are assigned individual fitness values. Based on the Bak-Sneppen model [26] of self-organised criticality (SOC), EO eliminates the worst component by exchanging it for another element at each iteration.

The initial solution is always created randomly. Variations are made to this initial solution using a "stepwise" sampling scheme that changes each of the dimensional variables at a time to produce a set of candidates. The sampling scheme produces a predefined number of equally distanced candidates in every dimension (see [27] for details). These candidate solutions are then ordered according to fitness.

This provides a rank k (where 1 is the worst) for each solution. The solution to be adopted can then be chosen with a probability of $k^{-\tau}$ where the only free parameter τ, usually a value between 1.1 and 3.0, is set to infinity. This setting eliminates the possibility for uphill moves which are often beneficial when EO is used as a standalone algorithm. In combination with a local search, the use of uphill moves has proved to be less desirable.

The algorithm then adopts one of the candidates as the next solution and proceeds to the local search phase.

3.2 Local Search

After the global search phase, the local search process takes place using the HJ pattern search algorithm. As described in the original paper [5], HJ starts with an exploratory move in which all dimensional variables in turn are changed by a predefined step. As improvements are equally likely in both directions along the dimensional axes, this takes at least twice as many function evaluations as there are dimensions, at most equally many. The pattern move then repeats all changes that were found to be successful in the exploratory move and uses a single function evaluation to evaluate the effect of the combined change.

The implementation of HJ algorithm is formalised as follows:

Hooke-Jeeves Pattern Search Algorithm

1. Obtain initial base point x^t . Determine set of step lengths.
2. Move the base point along every one of the *d* dimensional axes at a time and evaluate the result. Adopt each new point if improvement on the previous point. This takes at least *d*, at most *2d* evaluations. If any of the moves was successful, go to 3. If none was successful, go to 4.
3. Repeat the successful moves in a combined pattern move. If the new point has a better fitness, assume it as the new base point. Return to 2 whichever the outcome.
4. Adjust step length to next smaller step. If there is a smaller step, continue from 2. If not, terminate.

The HJ procedure repeats until no improving change can be made in any dimension. The step size is reduced and the procedure repeated until there are no more step sizes.

For our experiments, the use of exponential decline in step sizes has proved most successful.

$$s_j = s_{j-1} * b^j \qquad (2)$$

The sequence described by equation (2) was used for the experiments with HJMA. The initial value s_0 has to be determined on the basis of knowledge about the search space, and was set to {8, 9, 10, 11, 12} for our experiments. The power base *b* was set to the values {0.2, 0.3, 0.5}, as none of the individual values proved consistently superior to the others. The results presented did not always use the same step length sequences.

3.3 HJMA

The complete HJMA algorithm differs from MMEO [3] only in the local search part. Unlike MMEO, HJMA uses the HJ algorithm, which is used with different step length sequences. Also, the HJMA local search records which direction along each dimensional axis was used for the last improvement and checks this direction first.

In all other respects, the algorithms are identical. HJMA also eradicates duplicates after every exploratory and pattern move, and stores the solution when it cannot be improved further and it is not identified as a duplicate. The HJMA algorithm also comprises a fine-tuning phase where the best solution in memory is improved using a further step on the exponential sequence. The complete algorithm is outlined below:

Hooke-Jeeves based Memetic Algorithm

1. Find new solution by stepwise sampling of the space in each dimension. Evaluate solutions and rank by resulting fitness (quality). Choose new individual using power-law distribution.

2. Use HJ pattern search to optimise the new solution locally. Stop if too close to other solution. Stop when no further improvement is possible.
3. Store new solution if it was not removed as a duplicate in step 2.
4. Check whether the existing individuals can be improved by further local optimisation, i.e. a change has occurred.
5. Fine-tune best individual using HJ but sample closer to current position. Stop when no further improvement is possible.

4 Experiments and Results

In our experiments, we compare HJMA to MMEO [3] and CESO [25] from the literature. Additionally, we also compare it with an improved MMEO where some redundancies in the step lengths have been removed. As in the HJ implementation, the local search in the new MMEO also records the direction in which every dimensional variable is likely to improve.

The experimental results on all three scenarios are summarised in Table I, averaged over 50 runs with 100 changes to each run. The corresponding standard error is calculated by dividing the standard deviation and the square root of the number of runs.

Table 1. Offline error and standard error for all scenarios.

Scenario	CESO [25]	MMEO [3]	new MMEO	HJMA
1	-	0.10 ± 0.01	0.06 ± 0.01	0.06 ± 0.01
2	1.38 ± 0.02	0.66 ± 0.20	0.25 ± 0.08	0.25 ± 0.10
3	-	3.77 ± 1.72	1.43 ± 0.54	1.57 ± 0.73

From Table 1, it is obvious that HJMA clearly outperforms CESO and MMEO. It also shows comparable performance to the newly improved MMEO.

4.1 Varying the number of Peaks

The results obtained in Table 1 have used a total of 10 peaks for all the scenarios. In general, it is easier to find the global maximum when the number of peaks is small. However, it is easier to score on the offline error when the landscape is elevated (more peaks). To evaluate the performance of HJMA when different number of peaks is present, we test it with experiments on 1, 10, 20, 30, 40, 50 and 100 peaks for comparison. Experimental results obtained for different number of peaks are presented in Table 2.

For CESO, the best result has been obtained with the one peak setup. While HJMA did not perform well with one peak, it obtained better results than CESO and MMEO in all other instances. The improved MMEO shares similar results with HJMA. The reasons behind the similarity in results as well as the exceptionally poor performance in the scenario with the single peak will be the subject of further studies.

Table 2. Offline error and standard error for varying number of peaks.

No. peaks	CESO [25]	MMEO [3]	new MMEO	HJMA
1	**1.04 ± 0.00**	11.3 ± 3.56	7.47 ± 1.98	7.08 ± 1.99
10	1.38 ± 0.02	0.66 ± 0.20	**0.25 ± 0.08**	0.25 ± 0.10
20	1.72 ± 0.02	0.90 ± 0.16	0.40 ± 0.11	**0.39 ± 0.10**
30	1.24 ± 0.01	1.06 ± 0.14	0.49 ± 0.10	**0.49 ± 0.09**
40	1.30 ± 0.02	1.18 ± 0.16	**0.56 ± 0.09**	**0.56 ± 0.09**
50	1.45 ± 0.01	1.23 ± 0.11	0.59 ± 0.10	**0.58 ± 0.09**
100	1.28 ± 0.02	1.38 ± 0.09	**0.66 ± 0.07**	**0.66 ± 0.07**

4.2 Varying the Dimensionality

The dimensionality, tantamount to the complexity of the problem, is expected to have a large impact on the performances of different algorithms. The standard scenarios used to obtain the results in Table 1 have five dimensions. In order to investigate the effect of varying dimensionality in the search space, we test out HJMA and other algorithms on experiments with different dimensionality values. Numerical results on 10, 50 and 100 dimensions are presented in Table 3.

Table 3. Offline error and standard error for varying dimensionality.

Dimensions	CESO [25]	MMEO [3]	new MMEO	HJMA
10	2.51 ± 0.04	2.44 ± 0.77	2.25 ± 0.85	**2.17 ± 0.80**
50	6.81 ± 0.07	206.3 ± 35.7	6.22 ± 1.6	**5.79 ± 1.4**
100	24.60 ± 0.25	480.5 ± 70.1	17.8 ± 6.9	**16.5 ± 5.4**

As can be observed from Table 3, CESO reported an average offline error of 2.51 in the 10 dimensions search space. The performance of CESO deteriorated drastically when the dimensionality increases, with average offline errors of 6.81 and 24.60 for 50 dimensions and 100 dimensions respectively. On the other hand, HJMA is able to maintain a fairly competitive performance even when the dimensionality is increased to 100. It is also the first time where HJMA has shown distinguishably better results than the improved MMEO.

5 Conclusion

In this paper, we have proposed a new hybrid algorithm – HJMA – for solving dynamic function optimisation problems. HJMA significantly outperformed the best algorithms for the MP problem currently available in the literature. It has also maintained its outstanding performance in challenging environments, i.e. search spaces with different number of peaks and different dimensionality. The HJ pattern search has been particularly robust compared to other local search algorithms when the dimensionality is high.

In general, there is still room for HJMA to improve, considering the fact that it has been devised within a short period of time. Future work will investigate the portability of HJ with other types of metaheuristics.

References

1. T. Weise, M. Zapf, R. Chiong, and A. J. Nebro, “Why is optimization difficult?,” in R. Chiong (ed.), *Nature-Inspired Algorithms for Optimisation*, Berlin: Springer-Verlag, 2009, pp. 1-50.
2. J. Branke, “The moving peaks benchmark,” http://www.aifb.uni-karlsruhe.de/~jbr/MovPeaks/, viewed 08/11/08.
3. I. Moser, and T. Hendtlass, “A simple and efficient multi-component algorithm for solving dynamic function optimisation problems,” in *Proc. IEEE Congress on Evolutionary Computation (CEC ‘07)*, Singapore, 2007, pp. 252-259.
4. I. Moser, “Applying extremal optimisation to dynamic optimisation problems,” Ph.D. thesis, Faculty of Information and Communication Technologies, Swinburne University of Technology, Australia, 2008.
5. R. Hooke, and T. Jeeves, “Direct search solutions of numerical and statistical problems,” *Journal of the Association for Computing Machinery*, vol. 8, pp. 212-229, 1961.
6. S. Boettcher, and A. G. Percus, “Extremal optimization: methods derived from co-evolution,” in *Proc. Genetic and Evolutionary Computation Conference (GECCO ‘99)*, Orlando, Florida, USA, 1999, pp. 825-832.
7. J. Branke, “Memory-enhanced evolutionary algorithms for changing optimization problems,” in *Proc. IEEE Congress on Evolutionary Computation (CEC ‘99)*, Washington, DC, USA, 1999, pp. 1875-1882.
8. J. Branke, T. Kaußler, C. Schmidt, and H. Schmeck, “A multi-population approach to dynamic optimization problems,” in I. C. Parmee (ed.), *Adaptive Computing in Design and Manufacturing (ACDM ‘00)*, Berlin: Springer-Verlag, 2000, pp. 299-308.
9. J. Branke, and H. Schmeck, “Designing evolutionary algorithms for dynamic optimization problems,” in S. Tsutsui and A.Ghosh (eds.), *Theory and Application of Evolutionary Computation: Recent Trends*, Berlin: Springer-Verlag, 2002, pp. 239-362.
10. G. R. Kramer, and J. C. Gallagher, “Improvements to the *CGA enabling online intrinsic,” in *Proc. NASA/DoD Conference on Evolvable Hardware*, Chicago, Illinois, USA, 2003, pp. 225-231.
11. X. Zou, M. Wang, A. Zhou, and B. Mckay, “Evolutionary optimization based on chaotic sequence in dynamic environments”, in *Proc. IEEE International Conference on Networking, Sensing and Control*, Taipei, Taiwan, 2004, pp. 1364-1369.
12. C. Ronnewinkel, and T. Martinetz, “Explicit speciation with few a priori parameters for dynamic optimization problems,” in *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, San Francisco, California, USA, 2001, pp. 31-34.
13. L. T. Bui, J. Branke, and H. A. Abbass, “Diversity as a selection pressure in dynamic environments,” in *Proc. Genetic and Evolutionary Computation Conference (GECCO ‘05)*, Washington, DC, USA, 2005, pp. 1557-1558.
14. L. T. Bui, J. Branke, and H. A. Abbass, “Multiobjective optimization for dynamic environments,” in *Proc. IEEE Congress on Evolutionary Computation (CEC ‘05)*, Edinburgh, UK, 2005, pp. 2349-2356.
15. S. W. Fentress, “Exaptation as a means of evolving complex solutions,” Master’s Thesis, School of Informatics, University of Edinburgh, UK, 2005.
16. T. M. Blackwell, “Swarms in dynamic environments,” in *Proc. Genetic and Evolutionary Computation Conference (GECCO ‘03)*, Chicago, Illinois, USA, 2003, pp. 1-12.
17. T. Blackwell and J. Branke, “Multi-swarm optimization in dynamic environments,” in G. R. Raidl (ed.), *Applications of Evolutionary Computing*, LNCS vol. 3005, Berlin: Springer-Verlag, 2004, pp. 489-500.
18. D. Parrott, and X. Li, “A particle swarm model for tracking multiple peaks in a dynamic environment using speciation,” in *Proc. IEEE Congress on Evolutionary Computation (CEC ‘04)*, Portland, Oregon, USA, 2004, pp. 105-116.
19. S. Janson, and M. Middendorf, “A hierachical particle swarm optimizer for dynamic optimization problems,” in G. R. Raidl (ed.), *Applications of Evolutionary Computing*, LNCS vol. 3005, Berlin: Springer-Verlag, 2004, pp. 513-524.
20. H. Wang, D. Wang, and S. Yang, “Triggered memory-based swarm optimisation in dynamic environments,” in M. Giacobini et al. (eds.), *Applications of Evolutinary Computing*, LNCS vol. 4448, Berlin: Springer-Verlag, 2007, pp. 637-646.
21. T. Blackwell, and J. Branke, “Multi-swarms, exclusion and anti-convergence in dynamic environments,” *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 51-58, 2006.
22. R. Mendes, and A. Mohais, “DynDE: a differential evolution for dynamic optimization problems,” in *Proc. IEEE Congress on Evolutionary Computation (CEC ‘05)*, Edinburgh, UK, 2005, pp. 2808-2815.
23. K. D. Meyer, S. J. Nasut, and M. Bishop, “Stochastic diffusion search: partial function evaluation in swarm intelligence dynamic optimization,” in A. Abraham et al (eds.), *Stigmergic Optimization*, Studies in Computational Intelligence vol. 31, Berlin: Springer-Verlag, 2006, pp. 185-207.
24. K. Trojanowski, “B-cell algorithm as a parallel approach to optimization of moving peaks benchmark tasks,” in *Proc. 6th International Conference on Computer Information Systems and Industrial Management Applications (CISIM ’07)*, Elk, Poland, 2007, pp. 143-148.
25. R. I. Lung, and D. Dumitrescu, “A collaborative model for tracking optima in dynamic environments,” in *Proc. IEEE Congress on Evolutionary Computation (CEC ‘07)*, Singapore, 2007, pp. 564-567.
26. P. Bak, and K. Sneppen, “Punctuated equilibrium and criticality in a simple model of evolution,” *Physics Review Letters*, vol. 74, pp. 4083-4086, 1993.
27. S. Boettcher, and A. G. Percus, “Nature’s way of optimizing,” *Artificial Intelligence*, vol. 119, no. 1, pp. 275-286, 2000.