VISUALISATION OF ADVANCED SEARCH

Vassili Kurman*, Kalin Penev,
Faculty of Technology, Southampton Solent University,
Southampton SO14 0YN, UK,
5kurmv81@solent.ac.uk, Kalin.Penev@solent.ac.uk

ABSTRACT

This article presents investigation on visualisation of search processes. Existing evolutionary and adaptive algorithms for search and optimisation, in certain extent, may differ from each other in behaviour and in obtained results. An intention for future analysis of search algorithms, their behaviour and differences motivates the development of tools for visual representation of the search process. By developing a 3D graphical interface for Computational Intelligence Software such as Free Search [1] [2], Particle Swarm Optimisation [3], Differential Evolution [4] and Genetic Algorithm [5] [6] it is possible to build a scene with test function and individuals, moving on the landscape of that test function towards their goals.

1. INTRODUCTION

A number of test functions available in the literature can support examination of these advanced search algorithms. 3D graphical interface implements the following test functions: “Norwegian test function” [7], “Himmelblau test function” [8] and “Rosenbrock test function” [8] (Figure 1). In this application the above named test functions can be implemented only by using two dimensions.

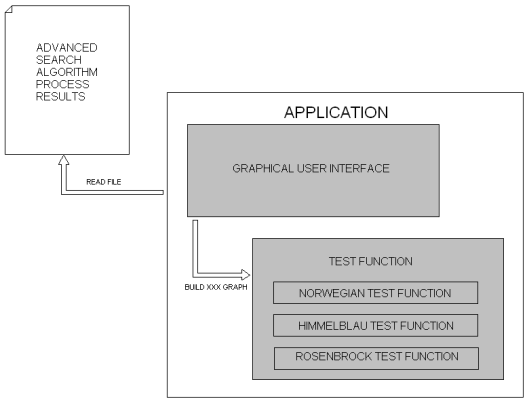

Figure 1. Visualisation model

For the purpose of the investigation the “Norwegian test function”, “Himmelblau test function” and “Rosenbrock test function” are predefined inside the application.

Object-oriented programming paradigm, and particularly Java/Java3D programming language has been used as implementation tool to develop current application. In the current case Java has advantages over other programming languages:

- Java application, including Java3D, can be run on any Java Virtual Machine regardless of computer architecture and platform.
- Java one of the fastest-growing and most widely used programming languages in the modern computing industry
- Java3D allows to build a 3D scene with complex objects and interact with scene/objects by using mouse or keyboard

2. APPLICATION ARCHITECTURE

Current application structure follows one of the main rules in design patterns: View, Control and Model (VCM) are separated. Control manages whole application process. View is responsible for displaying GUI and Model presents the main components of the application, such as individuals, axis, test functions, file reader. User has opportunity to choose test function, file with advanced search algorithm process results and some other parameters. After user selects all parameters needed, Control object passes control for building 3D scene to the 3D scene constructor, which sends message to the “Axis” constructor to build specified axis. After that 3D scene constructor sends message to the “Test Function” constructor with the name of test function to build. “Test function” constructor gets name of test function, builds 3D object by calling appropriate method and sends back specified test function object. Once 3D scene constructor has axis and test function objects, it passes control back to the Control object, which reads the file with one of the named above advanced search algorithm process results. After file is read, Control adds this objects (population) to 3D scene. After scene (Virtual Universe) is build, it is passed to View for displaying 3D scene with specified test function object and advanced search algorithm process results, visualising populations movement.

3. SCENE (VIRTUAL UNIVERSE) STRUCTURE

To be able to display a 3D scene with all objects specified earlier, Virtual Universe should be created first. Locale, which provides a reference point in the Virtual Universe object should be attached to that universe. Two main branch group attached to the Locale. One is responsible for view and called view branch graph and another is responsible for content, called content branch graph [10]. SimpleUniverse

* Corresponding author

utility class, used in current application, creates a scene graph including Virtual Universe and Locale objects and complete view branch graph. Content branch graph created separately and includes three branch groups: axis, test function and population. Population has included in it ten individuals. Each individual created separately as transform group and has its own attributes, such as colour and position. Scene graph model is drawn on the Figure 2 using standard set of symbols.

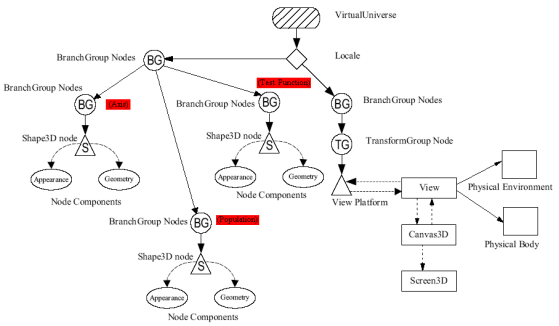

Figure 2. Virtual Universe model

On left hand side is drawn content branch graph and on right hand side is view branch graph.

3. TESTING

Three test functions "Norwegian test function", "Himmelblau test function" and "Rosenbrock test function" has been tested in application. In the Figure 3 is shown build scene graph with Norwegian test function, axis and population of ten individuals. As you can see all individuals are situated on the surface of the test function. It is much better to view the results of the Advanced Search Algorithm Process on the three dimensional graph then just read the numbers.

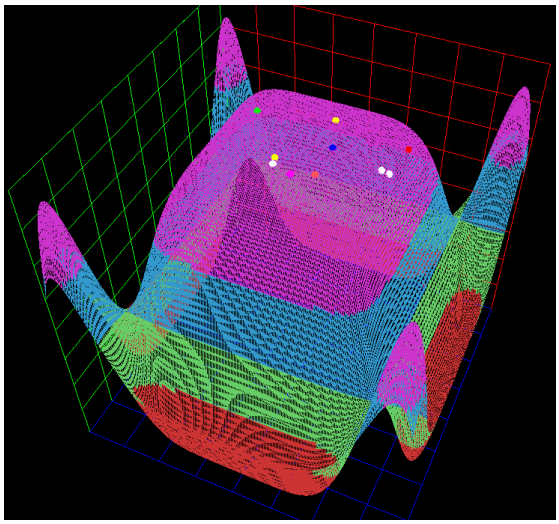

Figure 3. Norwegian test function and population

As addition to the scene graph, results of advanced search also displayed on the GUI for each individual in every point of time. It gives opportunity not only to watch the migration of population on the scene, but also to see the location, represented in X, Y and Z axis as double value.

Figure 4 below shows "Himmelblau test function", build by application. This test function has four maximum, therefore reaching one of them by individuals is acceptable.

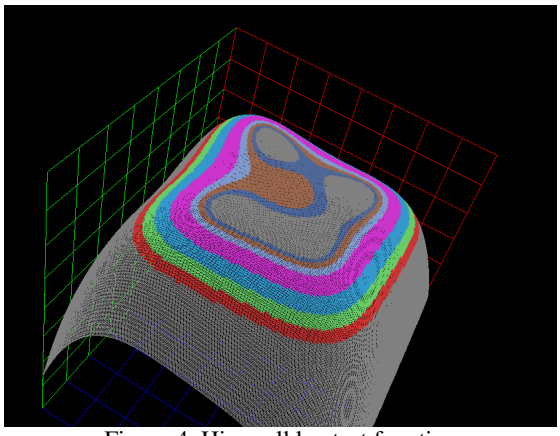

Figure 4. Himmelblau test function

"Rosenbrock test function" can be found on Figure 5. This test function has also one maximum, like described above "Norwegian test function".

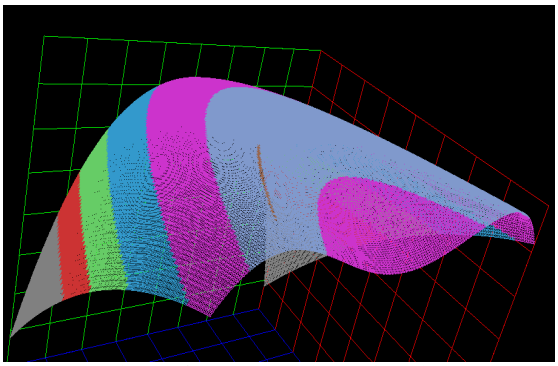

Figure 5. Rosenbrock test function

Different advanced search algorithm process results are available for all of the described above three test functions.

Besides the development of visualisation Free Search has been tested additionally on 20 and 50 dimensional Keane test function [13]

The best-achieved results for 20 dimensions is *Fmax20* = 0.80361910412558868. The constraint value is p = 0.75000000000000011. The values of the variables for 20 dimension variant of the test function are presented in Table 1.

The best achieved results for 50 dimensional search space is: *Fmax50*= 0.83526234835811175. The

constraint value is p=0.75000000000000144. The values of the variables for 50 dimension variant of the test function are presented in Table 2.

Table 1. Variables for
Fmax20 = 0.80361910412558868

x[0]=3.1624606581160428	x[1]=3.1283314501744788
x[2]=3.0947921408449481	x[3]=3.0614505981142819
x[4]=3.0279291985421692	x[5]=2.9938260980128195
x[6]=2.9586687259577391	x[7]=2.9218422645526609
x[8]=0.49482513593227934	x[9]=0.48835709655412185
x[10]=0.48231641293878313	x[11]=0.47664472480676501
x[12]=0.47129549301840407	x[13]=0.4662309976778744
x[14]=0.46142006012626469	x[15]=0.45683663647967088
x[16]=0.45245879070189687	x[17]=0.4482676202908692
x[18]=0.444247015003221	x[19]=0.44038285472829547

Table 2. Variables for
Fmax50 = 0.83526234835811175

x[0]=6.2835798267580847	x[1]=3.1699376751683297
x[2]=3.1560747532210827	x[3]=3.1423609876822067
x[4]=3.1287695128804933	x[5]=3.1152747677794008
x[6]=3.1018528643099077	x[7]=3.0884805697893061
x[8]=3.0751349193791264	x[9]=3.0617943870048903
x[10]=3.0484368324836968	x[11]=3.0350390513169692
x[12]=3.0215778555508499	x[13]=3.0080295265034538
x[14]=2.9943676921348237	x[15]=2.9805647630738066
x[16]=2.9665903787854409	x[17]=2.9524114588141499
x[18]=2.9379900900526348	x[19]=2.9232835995757385
x[20]=0.48823744461430302	x[21]=0.4859339247152083
x[22]=0.48368262801977796	x[23]=0.48148247436463903
x[24]=0.47932985897132679	x[25]=0.47722236064278706
x[26]=0.47515901874040484	x[27]=0.47313739151810102
x[28]=0.47115578871323049	x[29]=0.46921217059792986
x[30]=0.46730534115459976	x[31]=0.46543436895099993
x[32]=0.46359705310040189	x[33]=0.46179196226461594
x[34]=0.46001884072140975	x[35]=0.45827604291755331
x[36]=0.45656221349463588	x[37]=0.45487685610416123
x[38]=0.45321820664233081	x[39]=0.45158651633361024
x[40]=0.4499802229086835	x[41]=0.44839853044824601
x[42]=0.44684043291379633	x[43]=0.44530576219726076
x[44]=0.44379365198290671	x[45]=0.44230324041446462
x[46]=0.44083364944754766	x[47]=0.43938498943231946
x[48]=0.43795647121687242	x[49]=0.43654683305916353

These results exceed the published results achieved by modified version of Differential Evolution and require reconsiderations of the assumptions and conclusions about global maxima of this function [12]. Let's note that even better the above results cannot be considered as global maxima. The reasons for this are the search space is continuous and Free Search can perform unlimited by stagnation optimisation, and the results can be clarified to an arbitrary precision if necessary.

4. CONCLUSION

Visualisation of advanced search helps to see how population behaves itself on each situation and it is easier to analyse the process.

Further research could be concerned with implementation of other test functions and visualisation of other search algorithms. Also integrating visualisation application directly into the Advanced Search Algorithm Processing Software will give benefits to see what is happening with individuals and how they behave themselves directly without any middle links, like saving results in the text file and afterwards reading them back.

REFERENCES

1. Parmee I.C. 2004, *Adaptive Computing in Design and Manufacturing VI,* UK: Springer, pp. 295-306
2. Penev K., and Littlefair G., 2005, *Free Search – A Comparative Analysis*, Information Sciences Journal, Elsevier, Volume 172, Issues 1-2, pp 173-193.
3. Eberhart R. and Kennedy J., 1995, *Particle Swarm Optimisation, Proceedings of the IEEE International Conference on Neural Networks*, vol.4, 1942-1948.
4. Price K., and Storn R., 1997, *Differential Evolution*, Dr Dobb's Journal 22 (4), April, pp. 18-24
5. Holland, J. H, 1975, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor
6. Eshelman L.J. and Schaffer J.D., 1993, *Real-coded genetic algorithms and interval schemata, Foundations of Genetic Algorithms 2*, Morgan Kaufman Publishers, San Mateo, pp. 187-202
7. Federici D., 2005, The Norwegian University of Science and Technology, http://www.idi.ntnu.no/~federici/subai/index1.html, last visited 16.01.2008
8. Himmelblau D., 1972, *Applied Non-linear Programming*, New York: McGraw-Hill
9. Rosenbrock H., 1960, *An automate method for finding the greatest or least value of a function*, Computer Journal 3 (1960), pp. 175-184
10. Java 3D API Tutorial, http://java.sun.com/developer/onlineTraining/java3d/index.html, last accessed 20.03.2008
11. Java 3D 1.3 API Documentation http://java.sun.com/products/java-media/3D/forDevelopers/J3D_1_3_API/j3dapi/index.html, last visited 19.03.2008
12. Feoktistov V., 2006, Differential Evolution: In Search of Solutions (Springer Optimization and Its Applications) (Hardcover), Springer-Verlag New York Inc.
13. Keane A. J., 1996, A Brief Comparison of Some Evolutionary Optimization Methods, In

V.Rayward-Smith, I. Osman, C. Reeves and G.D. Smith, J. Wiley (Editors), Modern Heuristic Search Methods, pp 255-272.