Genetic Programming and Evolvable Machines manuscript No.
(will be inserted by the editor)

Tracer Spectrum: A visualisation method for distributed evolutionary computation

Michael O'Neill, Anthony Brabazon, Erik Hemberg

Abstract We present a novel visualisation method for island-based evolutionary algorithms based on the concept of *tracers* as adopted in medicine and molecular biology to follow a biochemical process. For example, a radioisotope or dye can be used to replace a stable component of a biological compound, and the signal from the radioisotope can be monitored as it passes through the body to measure the compound's distribution and elimination from the system. In a similar fashion we attach a *tracer dye* to individuals in each island, where each individual in any one island is marked with the same *colour*, and each island then has its own unique colour signal. We can then monitor how individuals undergoing migration events are distributed throughout the entire island ecosystem, thereby allowing the user to visually monitor takeover times and the resulting loss of diversity. This is achieved by visualising each island as a spectrum of the tracer dye associated with each individual. Experiments adopting different rates of migration and network connectivity confirm earlier research which predicts that island models are extremely sensitive to the size and frequency of migrations.

1 Introduction

Distributed forms of evolutionary computation are growing in popularity amongst practitioners, and a recent special issue of the Genetic Programming and Evolvable Machines journal was dedicated to parallel and distributed evolutionary algorithms [17, 18]. In the past ten years there have been a few studies which have attempted a detailed analysis of the behaviour of these algorithms. This is an unenviable challenge due to the added complexity brought about by additional issues beyond those addressed by standard theoretical analysis, such as the topology of population connections, migration rates and sizes, and the homogeneity versus heterogeneity of the different populations. Notable studies that undertake such an analysis include Cantú-Paz [1], Fernández et al [6] and Skolicki [13,14], which analyse the impact of migration rates and sizes, population sizing and network topology.

Natural Computing Research & Applications Group
Complex and Adaptive Systems Laboratory
University College Dublin
Ireland
E-mail: m.oneill@ucd.ie, anthony.brabazon@ucd.ie, erik.hemberg@ucd.ie

We present a visualisation tool which can be used to analyse the behaviour of distributed evolutionary algorithms in real-time. Such a tool can present the possibility of facilitating the choice of particular parameter settings and alerting the user to undesirable situations that may arise, such as loss of diversity, and can also be used to verify existing and future theoretical analysis of these distributed systems.

The remainder of this letter is structured as follows. A brief background to visualisation in evolutionary computation and distributed forms of evolutionary computation is provided with an emphasis on the island model in Section 2, which is followed by an introduction to the Tracer Spectrum visualisation approach in Section 3. Finally some conclusions and future work are outlined in Section 5.

2 Background

2.1 Island-based evolutionary computation

Island-based evolutionary computation takes place when multiple populations of a single evolutionary run are physically or logically separated from each other across a number of Islands [16]. Each island could be mapped to a single CPU/GPU core, to physically different machines connected over a network, or even just logical partitions within data structures and memory. Transfer of individuals is allowed to occur between the different islands through a process called *migration*, and the manner in which this is allowed to occur determines the structure of the network connecting these islands. Different network topologies might exist, ranging from a fully connected arrangement where individuals can migrate from any island to any other island, to more restricted topologies such as a ring. The rate of migration, the size of migration (i.e., the number of individuals that can participate in a single migration event), and the topology of the island network are all parameters of this model. The fitness function on each island and the representation of individuals on each island are two other degrees of freedom which can be explored within island-based evolutionary computation.

Clearly the additional degrees of freedom that an island-based approach to evolutionary computation presents adds an additional complexity to the algorithms employed. This in turn makes the analysis of such a model much more complex, and it can become difficult to understand how best to tune your island model to obtain optimal performance on any particular problem [13,6]. To facilitate our understanding of the behaviour of island models and to monitor the behaviour of any one run in real time, visualisation methods can be employed. In this letter we outline a novel visualisation method and briefly demonstrate its application. We now give a brief introduction to visualisation in evolutionary computation.

2.2 Visualisation in evolutionary computation

Many methods exist for visualisation in the field of evolutionary computation and include methods to: understand the impact of genetic operators, understand the search space and how it is being explored, and monitor diversity and dynamics of the population [2,8,12,3,11,9,10,15,7,20]. Some, like population fitness plots, are taken for granted as part of the standard toolkit. A good overview is provided by Hart [8]. Most of the methods in the literature are focused on standard (i.e., non-distributed) forms

of evolutionary computation, and this leaves a gap in the standard toolkits for visualisation methods particularly focused on distributed forms of evolutionary computation. We address this important open area of research in this letter by outlining an approach explicitly developed to analyse and monitor the behaviour of individuals undergoing migration in distributed evolutionary computation.

3 Tracer spectrum

In distributed evolutionary computation it is challenging to understand the dynamics of the population and in particular the impact migration has on the various subpopulations that comprise the global system. By associating a label (a tracer) to individuals which reflects, for example, their origin, interactions and fitness, we can begin to monitor and analyse the behaviour of distributed evolutionary computation. In medicine it is common practice to administer labeled compounds to a patient, and then to subsequently monitor how that compound becomes distributed through the body. This provides an invaluable tool for doctors to form a diagnosis in various diseases. In a similar fashion, we attach a tracer dye to individuals within the population and then monitor the progress of that tracer through the system. The tracer in this case is a dye, and the state of the dye is capable of undergoing modification as the individual interacts with other individuals labeled with different coloured dye. This enables us to use the tracer to monitor the origin of individuals (as we can associate different colour dyes with individuals belonging to different subpopulations) and interactions with individuals from other subpopulations (as during crossover events the dye of each individual is effectively mixed together in proportion to the amount of material being exchanged).

In the tracer spectrum visualisation approach each individual has associated with it a tracer attribute. In the following examples this will take the form of a gray-scale colour where the red, green and blue components are equal and are represented with values from the range 0 to 1.0. When individuals within a population interact through crossover events they exchange their tracer attribute values in proportion to the amount of genetic material that is being exchanged. Individuals undergoing crossover with the exact same tracer attribute value produce children of the same colour as themselves. Sample pseudocode outlining this process is provided in Figure 1.

```
P1.tracer = P1.tracer*percentSwapped + P2.tracer*(1.0-percentSwapped);
P2.tracer = P2.tracer*percentSwapped + P1.tracer*(1.0-percentSwapped);
```

Fig. 1 An outline of the code which blends the values of tracer attributes associates with two individuals ($P1$ and $P2$) undergoing crossover. *percentSwapped* refers to the proportion of the individual exchanged relative to the genotypic size of the parents. In this case an evolutionary algorithm with fixed-length genotypes is illustrated.

The tracer spectrum approach can be adopted for populations of either fixed or variable-length representations. As each individual has a single colour value associated with its entire genetic content, regardless of whether individuals in a population are fixed or variable-length structures the colour of the tracer colour can be calculated by determining the proportion of each individual being exchanged.

(a) gen=0

(b) gen=10

(c) gen=20

Fig. 2 Fig. 2(a) is the generation zero tracer spectra for a three island network with island 1(left), island 2(center) and island 3(right). Fig. 2(b) displays generation 10, and Fig. 2(c) generation 20. The migration rate is 10% of the total population size of each individual island with a frequency of every 6th generation. A rapid convergence of each spectrum is observed.

(a) gen=10

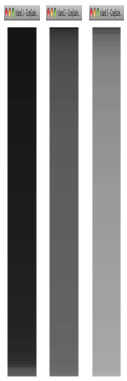
(b) gen=20

(c) gen=30

Fig. 3 Fig. 3(a)illustrates the generation 10 tracer spectra for a three island network with island 1(left), island 2(center) and island 3(right). Fig. 3(b) generation 20, and Fig. 3(c) generation 30. The migration rate is 1% of the total population size of each individual island with a frequency of every 6th generation. Relative to Figure 2 a slow convergence of each spectrum is observed (note that an additional ten generations are presented in this figure compared to the 10% migration).

(a) gen=10

(b) gen=20



(c) gen=30

Fig. 4 Fig. 4(a) illustrates the generation 10 tracer spectra for a three island network with island 1(left), island 2(center) and island 3(right). Fig. 4(b) displays generation 20, and Fig. 4(c) generation 30. The migration rate is 1% of the total population size of each individual island with a frequency of every generation. Relative to Figure 3 with a migration frequency of every generation a much faster convergence of each spectrum is observed.

If we wished to use a non-gray-scale, full technicolor approach the same approach can be adopted for each component of an RGB colour. In terms of adapting the visualisation to suit colour vision impaired users the choice of colours should be considered carefully to improve readability. Troiano et al. [19] describe a novel approach to adapting a GA to evolve a colour palette. In a distributed evolutionary algorithm, each population (or island) is assigned a unique tracer value. In the following example there are three islands, where the first island is assigned the colour black (0.0), the second island gray (0.5) and the third island white (1.0).

The code was developed for ECJ v18 [5], and we have made an open source release of the code available [1]. If we plot the tracer value of each population member as a line we can display a spectrum of the tracer values present for each island. This is illustrated with an example run in the top line of Figure 2 for the three island setup, where each island is displayed as an independent tracer spectrum. In each spectrum, each member of the population is represented as a coloured line (row) displayed the full width of the island image. The population members lines (rows) are then stacked in the island column. In the experiments conducted here the population size is 500, and as such there are 500 rows in each island tracer spectrum.

Parameters of the tracer spectrum visualisation are the width and height of each line, which would need to be calibrated depending on the number of individuals (lines/rows) and the number of islands. The tracer spectrum for each island can either be stacked as rows in a single column, or placed side by side with each island as a separate column. In the later approach narrower lines would be adopted, whereas it would be possible to accomodate wider lines with the former approach.

Following a migration event new tracer values are introduced into the spectrum for the receiving population. These new values undergo a blending process when crossover events occur between these immigrants and the existing population. This process is illustrated in Figures 2 and 3 for a typically adopted 10% migration rate (e.g., as adopted in [6,1]) and lower 1% rate respectively. The problem adopted in these examples is the simple binary string OneMax problem of length 500. Other parameters include a population size of 500, crossover probability of 0.7, bit-flip mutation probability of 0.01, tournament selection with tournament size of 2, elitism (1 individual). The frequency of migration events is every 6 generations, and the island topology is a fully connected graph.

If we change the frequency of migration to occur every generation we observe the following spectra in Figure 4. As can be seen a much faster mixing of the tracer is observed when compared to the lower frequency migration event setups.

In effect we have a visualisation tool which can monitor the mixing that is occurring within the island ecosystem, allowing the observation of the convergence of the populations to an equilibrium (relaxation time). With this information the practitioner can tune the island-based parameter settings for the problem at hand. In addition, it provides a tool by which theoretical developments can be tested. The Tracer Spectrum visualisation is not without limitations. As noted earlier careful attention needs to be paid to the selection of the colour palette associated with each island, in terms of a user's ability to discern differences. Also, as one needs to visualise an increasing number of islands it will become increasingly difficult to observe differences between members of each island. However, if the purpose of these methods is to observe the origin of each

[1] The Tracer Spectrum classes (implemented in Java) for ECJ are available from http://ncra.ucd.ie/tracerspectrum/

gene in a population an extension to the method is proposed in the following section which overcomes this island number scalability issue.

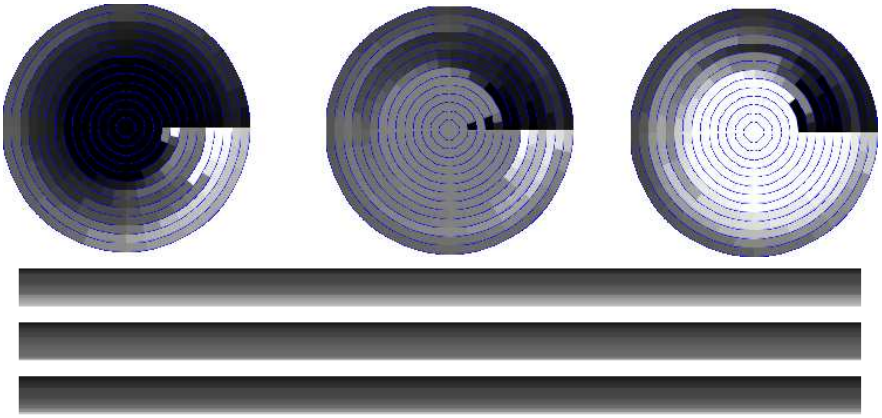


Fig. 5 Tracer spectrum (bottom) and tree ring (top) example for generation 13 on three islands. Note how the tree rings capture all generations up to 13 (outer ring, the inner ring is generation 0), whereas the tracer spectrum presents the current generation at any point in time. Note that we have also changed the tracer spectrum width parameter from the earlier examples.

4 Variations on tracer spectrum

We present a number of variations on the theme of tracer spectrum that can be used as either alternative visualisations or extensions to the basic idea. It is hoped these provide inspiration for other variations which might be possible.

4.1 Alternative views

An aesthetically pleasing alternative to the standard visualisation presented in Figures 2, 3 and 4 is to adopt *tree rings*. This allows a more compact representation of a population, as instead of presenting each member of the population as a line, and each line stacked vertically, an individual in a tree ring is represented by a small arc or segment of the ring. Figure 5 provides an example of a tree ring visualisation for three islands running for 13 generations, sitting alongside the equivalent tracer spectrum visualisation for generation 13. Working from the centre core of the tree rings for each island we see generation zero. Each subsequent ring moving out from the core represents the next generation. Note the tracer colour of the center of each tree ring island. In the leftmost tree ring the core is black as each individual of the initial generation originates from this island alone. Similarly the center and rightmost tree ring cores are gray and white respectively denoting the tracer colour of the second and third islands. The outer ring for each island's tree ring represents the 13th generation, and in each case it can be observed that a significant amount of migration and subsequent exchange of genetic material has occured by this time.

A powerful advantage of the tree ring visualisation over the standard tracer spectrum display is that it captures all generations to date in a single visualisation. In the case of tracer spectra they only capture a single generation at a time, and do not give a sense of how the island populations are changing from one generation to the next (except when observing the tracer spectra live during a run when the visualisation is effectively animated). One potential limitation of the tree ring approach is that as the population sizes increase the arc size available to each individual of the population will decrease.

4.2 Adding fitness information

Incorporating fitness information into a tracer spectrum is trivial. As we are adopting RGB colors the *alpha* (transparency) component of a color can be used to represent fitness. The alpha values themselves are normalised into the legal range (e.g., 0.0 to 1.0, or 0 to 255). Therefore, the closer an individual is to the ideal fitness the less transparent the color of their line will be.

In the case of the tree ring visualisation, the average fitness of each generation can be represented by the width of each tree ring. An example of this in operation is presented in Fig. 6. Tree ring widths are determined in this example according to the following equation,

$$width \; = \; m + s * (f_t - f_{t-1})/f_t$$

where m is a constant representing the minimum ring width (10 in this example), s is a scaling term to normalise ring widths (set to 200 in this example), f_t is the average fitness of the current population, and f_{t-1} is the average fitness of the previous generation. As such ring widths are in the range 10 to 210 in Fig. 6.

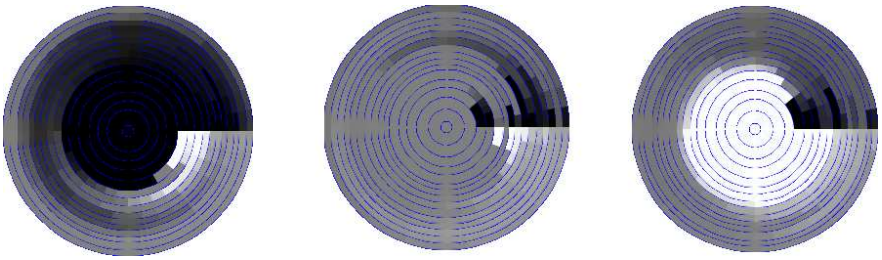


Fig. 6 An example of the tree ring visualisation incorporating average population fitness as ring width for 15 generations on three islands.

4.3 Tracing origin

Although we have not yet implemented this approach we think it is worth noting that the method can be extended to trace the origin of genes within a population. As per the original method described in the previous section each island has associated with it a tracer dye colour. Each gene/locus (or whatever the fundamental unit of the representation adopted is) in an individual is labelled with an attribute value denoting

this colour. The colour of each locus is fixed for the duration of a run. When individuals undergo a crossover event the island origin of each allele in the population is clearly visible as the genes which are exchanged retain their respective tracer attribute value. An illustration of this process can be observed in Fig. 7 for linear structures and Fig. 8 for GP tree structures.

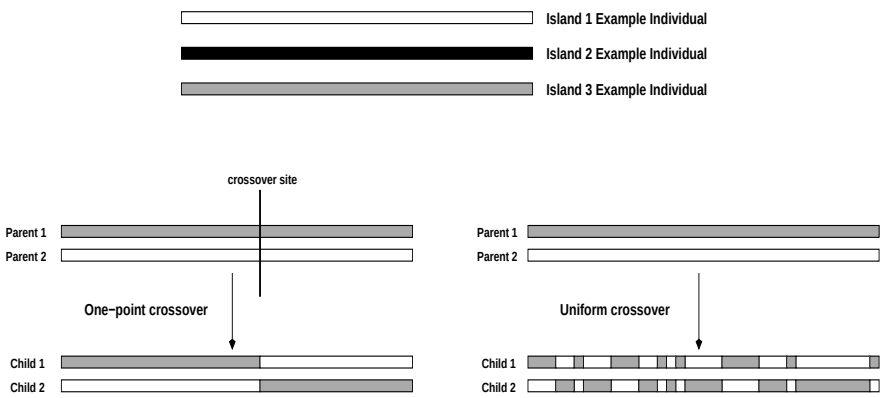


Fig. 7 An illustration of how island origin can be traced using the fixed dye method. During genetic operations the tracer dye is not modified in this variant of the visualisation. The origin of each gene in the population can be traced back to its island of origin.

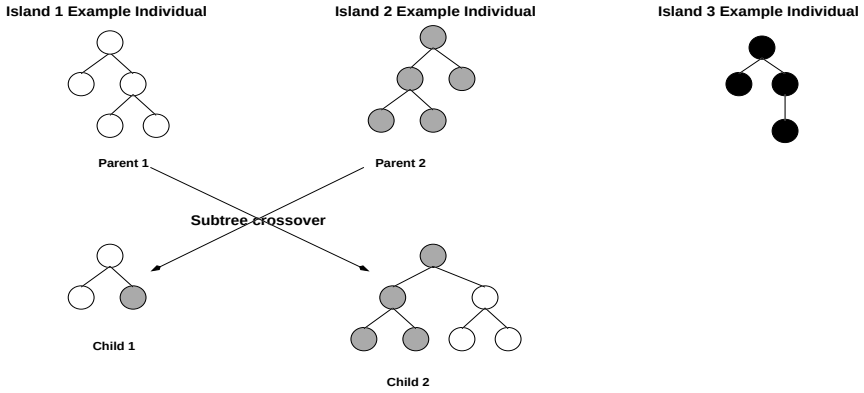


Fig. 8 An illustration of how island origin can be traced using the fixed dye method for GP trees. Each node is marked with the origin island tracer colour. It is then possible to trace the origin of each node in a population back to its island of origin.

The proposed tracing origin visualisation can also handle variable-length structures. In the event of an absence of a gene at any locus in the individuals undergoing crossover, the dye colour will be unchanged for the locus that is present. In the case of genetic operators introducing new dimensions into the population the island in which the new loci are added determines the original dye colour of these new loci. In the case of GP trees a variation of the visualisation would be required. In this case each node of the

tree is labelled with its originating island's tracer colour. As individuals are exposed to crossover events, as per the linear version, each node retains its originating island tracer colour. For each individual at each generation it is therefore possible to visualise the island origin of each tree node.

5 Conclusions & future work

A novel visualisation tool, Tracer Spectrum, for distributed evolutionary algorithms is introduced. This is a valuable contribution to the visualisation toolkit, which to date is lacking specific visualisation tools to monitor distributed EC systems. Initial use of the tool provides visual evidence to support recent findings with respect to the importance of the frequency and size of migration events in an island-based framework. The tracer spectrum tool can be used to validate theoretical developments in distributed evolutionary computation, and as a tool for the practitioner to monitor the flow of individuals through their distributed ecosystem and their corresponding relaxation time (i.e., convergence to an equilibrium), facilitating the detection of undesirable behaviour such as loss of diversity. Tracer spectrum also has a role to play as an educational tool to demonstrate distributed evolutionary computation, and can help to build intuition in researchers and practitioners. A number of extensions and variations on the basic tracer spectrum idea were proposed, and it is hoped that these may provide some inspiration for future research directions in this area.

Acknowledgments

We are grateful to the comments and insights of the anonymous reviewers, which have helped to improve this letter. This research is based upon works supported by the Science Foundation Ireland under Grant No. 08/IN.1/I1868.

References

1. Cantú-Paz, E. (1999). Migration policies and takeover times in parallel genetic algorithms. IlliGAL Report No. 99008.
2. Collins T.D. (1999). Evolutionary Computation Visualization Workshop. In Wu, A.S. (Ed.) Proceedings of the 1999 Genetic and Evolutionary Computation Conference Workshop Programme. Orlando, Florida.
3. Collins J.J., O'Neill M. (2000). Visualisation of evolutionary algorithms using principal components analysis. In Proceedings of the Fifth International Symposium on Artificial Life and Robotics AROB 2000, pp. 247-250, Japan.
4. Daida J.M., Ward D.J., Hilss A.M., Long S.L., Hodges M.R., Kriesel J.T. (2004). Visualizing the loss of diversity in genetic programming. In Proceedings of 2004 IEEE Congress on Evolutionary Computation, pp.1225-1232. IEEE Press.
5. ECJ v18: A Java-based evolutionary computation research system. `http://www.cs.gmu.edu/~eclab/projects/ecj/`
6. Fernández F., Tomassini M., Vanneschi L. (2003). An empirical study of multipopulation genetic programming. Genetic Programming and Evolvable Machines 4(1):21-51.
7. Franken N. (2009). Visual exploration of algorithm parameter space. In CEC'09': IEEE Congress on Evolutionary Computation, pp.389-398. IEEE Press.
8. Hart E., Ross P. (2001). GAVEL - a new tool for genetic algorithm visualization. IEEE Transactions on Evolutionary Computation, 5(4):335-348.

9. Khemka N., Jacob C. (2009). VISPLORE: a toolkit to explore particle swarms by visual inspection. In GECCO 2009: Proceedings of the 11th Conference on Genetic and Evolutionary Computation, Montreal, QC, pp.41-48. ACM Press.
10. Kim Y-H., Lee K.H., Yoon Y. (2009). Visualizing the search process of particle swarm optimization. In GECCO 2009: Proceedings of the 11th Conference on Genetic and Evolutionary Computation, Montreal, QC, pp.49-56. ACM Press.
11. Llorá X., Sastry K., Alías F., Goldberg D.E., Welge M. (2006). Analyzing active interactive genetic algorithms using visual analytics. In GECCO 2006: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, Seattle, WA, USA, pp.1417-1418. ACM Press.
12. Pohlheim H. (1999). Visualization of Evolutionary Algorithms - Set of Standard Techniques and Multidimensional Visualization. In Proceedings of GECCO 1999, Vol.1., pp.533-540. Morgan Kaufmann.
13. Skolicki Z. (2007). An analysis of island models in evolutionary computation. PhD Thesis, George Mason University, Fairfax, VA.
14. Skolicki Z., De Jong K. (2005). The influence of migration sizes and intervals on island models. In Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2005, Washington, DC, USA, pp. 1295-1302, ACM.
15. Stan-Bishop C., Barone L.C., While R.L. (2009). Visualisation of building blocks in evolutionary algorithms. In CEC'09: IEEE Congress on Evolutionary Computation, pp.2431-2438. IEEE Press.
16. Tomassini M. (1999). Parallel and distributed evolutionary algorithms: A Review. Chapter 7 in Miettinen et al. (Eds) Evolutionary Algorithms in Engineering and Science. Wiley.
17. Tomassini M., Vanneschi L. (2009). Introduction: special issue on parallel and distributed evolutionary algorithms, part I. Genetic Programming and Evolvable Machines 10(4):339-341.
18. Tomassini M., Vanneschi L. (2010). Guest editorial: special issue on parallel and distributed evolutionary algorithms, part two. Genetic Programming and Evolvable Machines 11(2):129-130.
19. Troiano L., Birtolo C., Miranda M. (2008). Adapting palettes to color vision deficiencies by genetic algorithm. In GECCO 2008: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, Atlanta, GA, USA, pp.1065-1072. ACM Press.
20. Valdes J.J., Barton A.J. (2007). Visualizing high dimensional objective spaces for multi-objective optimization: A virtual reality approach. In CEC'07: IEEE Congress on Evolutionary Computation, pp.4199-4206. IEEE Press.