GridVis: Visualisation of Island-Based Parallel Genetic Algorithms

Evelyne Lutton[1], Hugo Gilbert[2], Waldo Cancino[3], Benjamin Bach[3], Pierre Parrend[4,5], and Pierre Collet[4]

[1] INRA, UMR GMPA, 1 Av. Brétignières, 78850, Thiverval-Grignon, France
evelyne.lutton@grignon.inra.fr
[2] ENSTA ParisTech, 828, Boulevard des Maréchaux, 91762 Palaiseau Cedex, France
hugo.gilbert@ensta-paristech.fr
[3] INRIA Saclay-Ile-de-France, AVIZ team, Bâtiment 660, 91405, Orsay Cedex, France
waldo.cancino@gmail.com, benjamin.bach@inria.fr
[4] ICube laboratory, Strasbourg University, and ECCE, CS-DC UNESCO UniTwin, 7, rue René Descartes, 67084 Strasbourg, France
pierre.collet@unistra.fr
[5] ECAM Strasbourg-Europe 2, Rue de Madrid, Schiltigheim, CS 20013, 67012 Strasbourg Cedex
pierre.parrend@ecam-strasbourg.eu

Abstract. Island Model parallel genetic algorithms rely on various migration models and their associated parameter settings. A fine understanding of how the islands interact and exchange informations is an important issue for the design of efficient algorithms. This article presents GridVis, an interactive tool for visualising the exchange of individuals and the propagation of fitness values between islands. We performed several experiments on a grid and on a cluster to evaluate GridVis' ability to visualise the activity of each machine and the communication flow between machines. Experiments have been made on the optimisation of a Weierstrass function using the EASEA language, with two schemes: a scheme based on uniform islands and another based on specialised islands (Exploitation, Exploration and Storage Islands). [6]

Keywords: Parallel evolutionary algorithms, Island model, Visualisation, EASEA, grid model

1 Introduction

Island Models are a popular way to parallelise Evolutionary Algorithms: The classical evolutionary model of a single population that performs a community search on an unknown search space is replaced by a set of subpopulations, living their own life in parallel on different machines. This scheme is another way to control the balance between diversity preservation and focus of search:

[6] This work has been funded by the French National Agency for research (ANR), under the grant ANR-11-EMMA-0017, EASEA-Cloud Emergence project 2011, http://www.agence-nationale-recherche.fr/.

– It has been proved that having multiple subpopulations helps to preserve genetic diversity, since each island can potentially follow a different search trajectory through the search space[1].
– Global search ability is maintained by periodically exchanging individuals between machines in a process called *migration*.

Migration is thus an important component of islands models, which is controlled by parameters, such as *migration interval* to set the number of generations between two migrations, and *migration size* to define the number of migrating individuals. Due to these additional parameters, it is obvious that a careful parameter setting is a condition to get efficient island models schemes. Theoretical and experimental studies may help to find typical settings, however *ad hoc* tuning remains the favourite method for addressing most of the real life optimisation problems. Visualisation of algorithmic behaviour is the approach that becomes more and more popular for this purpose, as soon as it is able to provide infomation in a condensed, visual, and easily interpretable way [2,3,4]. The visualisation of evolutionary algorithms is now drawing more and more attention in the community (see for instance the VizGec Workshop series at GECCO conference[7]), and specialised visualisation tools are distributed for various purpose (see Section 2).

In this paper, we present a visualisation tool specifically developed for island-based evolutionary algorithms, called GridVis. Thanks to the EASEA language, communications between machines are collected during execution into log files local to each machine, then grouped, and visualised after execution as a heatmap matrix. Various tools allow examining data at different scales with respect to groups of machines, time or fitness of individual exchanged. Detailed views of the activity of each machine (sending or receiving) are also available.

The remainder of this paper is organised as follows. Section 2 gives an overview of visualisation challenges for evolutionary algorithms, with a focus on what is specific to parallel evolutionary algorithms. Section 2 also presents the EASEA language and its parallel implementation based on an Island Model. GridVis is presented in Section 3, followed by an experimental analysis in Section 4. Results of our experiments are discussed in Section 4.2. Section 5 concludes our work and discusses future research directions.

2 Background

Visualising evolutionary algorithms (EA) is complex, since many different scales and many different objects need to be visualised. Existing tools fall into two major groups: *a)* off-line tools or post-mortem analysis, which try to give an image as precise as possible of all the phenomena which occured during one or several runs [5,6,7,8,9,10,8], and *b)* on-line tools, which are usually less complex as they monitor what is currently happening during a run [11,12,13,14,15]. The following issues are desirable objectives for EA visualisation [7,16,17,18], they consider different levels:

[7] http://www.vizgec.ex.ac.uk

- *Individual level:* How to visualise a solution to the problem: both genome and phenotype? How to deal with problem dependent data?
- *Population level:* How to display statistics, convergence, loss of diversity, and lineage of a good solution?
- *Process level:* How to highlight the effects of genetic operators and other parameter settings?
- *Output:* How to visualise the result, particularly in non-standard EAs like multi-objective or cooperative-coevolution EAs?

Visualising parallel and multi-population EAs is a topic which is explored only to a limited extend in the literature so far. For instance, Pohlheim [4] succinctely presents some tools to visualise sub populations and migration effects as 2D coloured plot diagrams.

Stakes in visualising generic parallel processes are depicted in particular in [19,20,21]. [20] underlines the importance of monitoring asynchronous, distributed algorithms on multiple processors for detecting inconsistencies in the algorithms, to get performance parameters and to develop a conceptual understanding of the algorithm's behaviour. Morrow and Gosh also highlight the computational cost of such a visualisation system if used on-line. Brown et al. [21] report on dynamic visualisations of parallel algorithms for a specific architecture (torus computers), based on a language for encoding optimisation algorithms.

This topic is tightly bound with a broader related research topic: program visualisation. [22], for instance, reports a survey (until 1993) and proposed a taxonomy of program visualisation tools (visualisations for performance tuning, debugging, teaching or understanding the behaviour of programs). A more recent survey of 18 different algorithm visualisation systems [23] focusses on their use in education. It seems clear that the use of a description language makes visualisations less dependent from architecture. GridVis has been developed in a similar spirit, based on the EASEA language described hereafter.

EASEA (for EAsy Specification of Evolutionary Algorithms) [24,25] was initially designed to assist users in producing an evolutionary algorithm from a given problem description. It is based on a C-like language that contains code for the genetic operators (crossover, mutation, initialisation and evaluation) and the genome structure. These functions are written in a dedicated description file, the *.ez file*. Out of them, EASEA generates a complete evolutionary algorithm with potential parallelisation of evaluation over GPGPUs[26], or over a cluster of heterogeneous machines, in the case of an island model. The generated source file for the evolutionary algorithm is user-readable. It can be used as-is, or as a primer, to be manually extended by an expert programmer.

The island model is an efficient and simple way to parallelise evolutionary algorithms [27], because it often results in important speedup. In a cluster of computers, every node, which can be seen as an island, runs a complete evolutionary algorithm, which can be seen as an island. A migration mechanism allows periodically exporting some individuals to the other nodes. EASEA implements islands using

– exchanges between nodes limited by the migration interval, *i.e.* the migration of one individual every n generations , and the migration size, the number of individuals that migrate. This protocol sets up a very lightweight asynchronous communication.
– a loosely connected model that is based on UDP, which allows parallelising over neighbour or distant machines (cluster or grid computing).

Extensions of EASEA to grid and cloud computing are currently under development through the EASEA-Cloud project[8].

3 GridVis

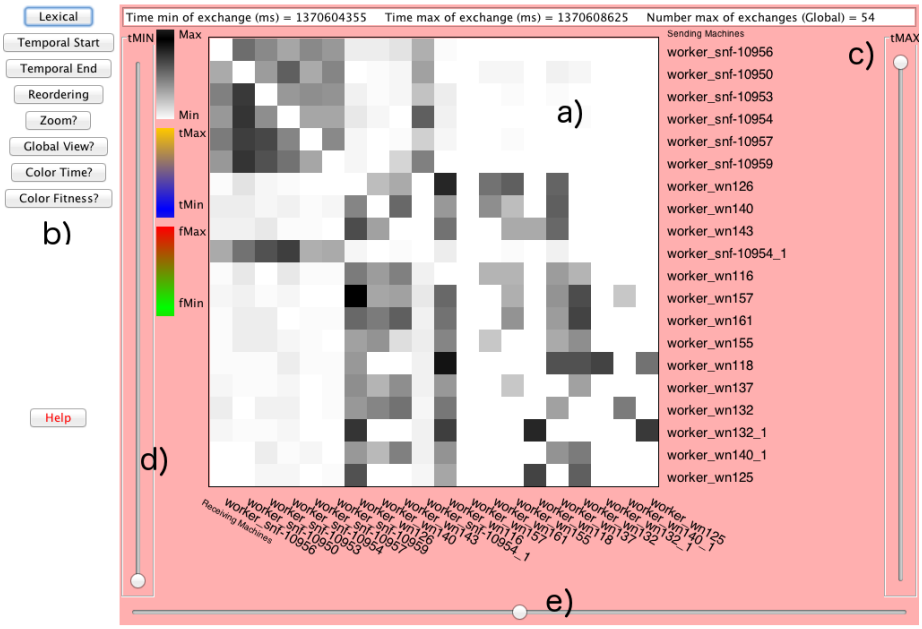


Fig. 1. GridVis Interface visualisation of a grid with 20 machines, using a matrix representation (a). User control is provided by (b) buttons to change the matrix row and column order, and (c,d,e) the shown time interval.

GridVis has been developed in Java, to monitor how the islands communicate: Which machines exchange individuals? When and how much individuals are exchanged? How fit are they? Which machines are the central ones? Are there clusters of exchange? We model the computer cluster that is running the island model, as dynamic network with weighted edges (number and fitness of individuals) and use an adjacency matrix for visualization (Figure 1(a)). Each computer

[8] ANR-11-EMMA-0017, Emergence project 2011, http://www.agence-nationale-recherche.fr/

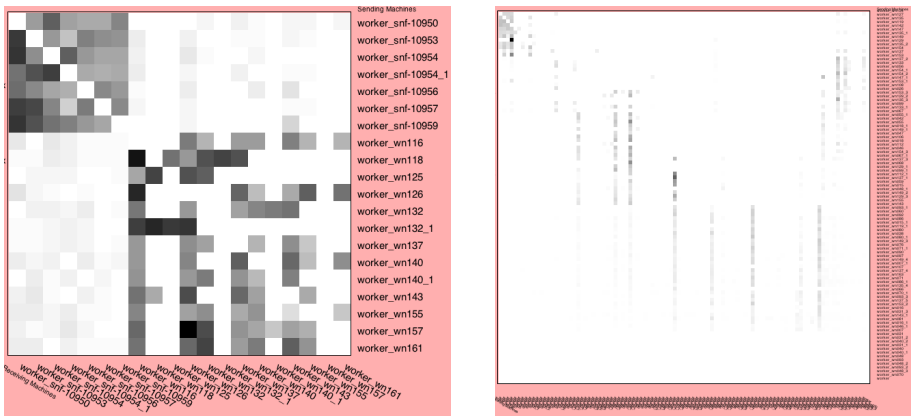
in the cluster appears twice in the matrix, once as row and once as column. Cells in the matrix show information about the exchange between computers during evolution, for example, the amount of individuals exchanged (from row to column). Similar to heat maps [28], exchange is mapped to darkness. Darker cells indicate a higher exchange rate.

While matrix visualizations have recently been applied to dynamic networks [29,30], heatmap visualisations have been used for many different purposes. For evolutionary algorithms visualisation, they have been used for instance for facilitating the exploration and interpretation of Pareto fronts [31]. This visualization scheme has been chosen for the following reasons:

- *Visual simplicity*: Brigthness and colour perception is pre-attentive, and clusters of islands with high exchange rates appear close, due to row and column reordering optimisation.
- *Scalability*: Matrices are well suited for visualising large networks (typical clusters contain about 100 machines) and with many relations between machines (individuals are potentially exchanged between all machines) [32].

Figure 1 shows the GridVis display for a grid of 20 and 100 machines (islands), respectively. The number of individuals sent during a time interval $[t_{MIN}, t_{MAX}]$ from machine i to machine j is given by the grey level of cell (i, j) (white corresponds to no exchange, black is the highest count). Machines are identified by their names on row and columns. The time interval $[t_{MIN}, t_{MAX}]$, which determines the shown exchanges in the matrix, can be dynamically modified using sliders: independently for t_{MIN} (Figure 1(b)) and t_{MAX} (Figure 1(c,d)) and as a sliding interval using the bottom bar (Figure 1(e)). Numerical values (in ms for the time) are given in the white frame above the heat map. The following options can be activated on demand, using the buttons in Figure 1(b):

- *Row and column reordering*: Rows and columns can be ordered (i) *lexically*, (ii) by time, or (iii) by similarity of activity. Lexical reordering consists in ordering according to line and column labels. Figure 2(a) illustrates a lexical reordering. The clusters (dark areas in the matrix), which appear using the lexical ordering, indicate that the algorithm choses machines for exchanging individuals, based on the machine's name; nodes of the first cluster had very different names from those of the second. *Temporal reordering* sorts the machines according to when the first individual was sent (Figure 2(b)). *Similarity ordering* tries to place machines (rows and columns) with similar exchange behaviour, close together in order to make sub-clusters of machines visually appearing as dark areas in the matrix. Our ordering optimisation is done using a Traveling Salesman problem (TSP) resolver that takes the number of exchanged individuals as similarity.
- *Zoom*: Individual machines can be selected to get a focussed view, which shows the selected machines only, while using the entire space in the matrix.
- *Grey level rescaling:* each cell has a grey value computed by interpolation between 0 and the maximal number of individuals exchanged in the current



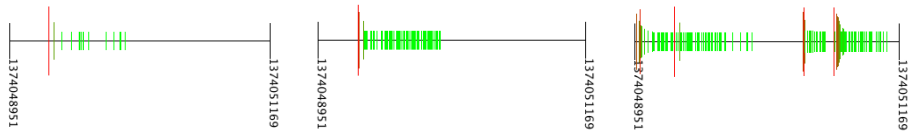
(a) Lexical Reordering (b) Temporal Reordering

Fig. 2. Examples for row and column reordering strategies in GridVis. (a) **Lexical reordering**: Here, individuals are almost exclusively exchanged between machines of similar names. (b) **Temporal Reordering**: Here, after a period of exchange between a group of machines (upper-left grey block), some machines worked alone at different times (cells appearing as vertical bars). This effect is due to the grid middelware (Glite) for which the choice of number of islands is controlled but not their synchronisation.

time interval. The grey values are dynamically rescaled when the time interval is changed. The "global view" button cancels this rescaling so that grey levels correspond to the absolute global count of each cell.

– *Colour representation*: The cell colour intensity still represents the number of individuals exchanged, while a colour scale from blue via purple and beige to orange (*time coloring*) represents the time at which the first individual has been exchanged (t_{MIN} is blue and t_{MAX} is orange) (Figure 5(a)). Alternately, the fitness of the exchanged individuals can be shown as a colour ranging from yellow (low fitness) via orange to red (high fitness) (*value encoding*). For each machine we indicate the average best fitness in the considered time interval on the matrix diagonal using that same colour encoding (Figure 5(b)).

Detailed views on the exchange between each pair of machines are available by clicking on the corresponding cell in the matrix. A bar chart as shown in Figure 3(a), indicates *when* individuals have been exchanged in the considered time interval (bars) and *how fit* they have been (colour and length of bars, encoded redundantly). For example, machine j in Figure 3(c) received during three periods, separated by interuptions. At the beginning of each period, many individuals with high fitness are exchanged (red bars). Then rapidly the fitness decreases (the aim is a minimisation of the fitness function). Each period corresponds to the start of a new block of machines. It is obvious that after a while



(a) From i to j (b) Sent by i (c) Received by j

Fig. 3. Detailed views visualizing individuals exchanged between two machines, i and j, over time (horizontal bar). Each vertical bar represents an individual, its position on the black horizontal line corresponds to when he was exchanged, received or sent, between t_{MIN} and t_{MAX}. The colours and widths of the bars corresponds to fitness value (long and red is high, short and green is low).

the fitness of exchanged individuals rapidly reaches again the best so far global fitness.

Likewise, clicking a machine's label in the matrix *columns*, shows the same chart indicating when and how much individuals the machine received from the grid (Figure 3(c)). Clicking a machine's name on a *row* shows what it sent (Figure 3(b)).

4 Experimental analysis

4.1 Setup

Experiments were run using EASEA [33]. The test case aims at **minimising** a Weierstras test function with 10 variables. GridVis has been used to analyse two sets of experiments:

1. *EASEA-Grid experiments*, performed on the Complex Systems Virtual Organization of the European Grid Infrastructure (EGI). Experiments have been performed with 20 and 100 islands. Parameters are given in Table 1.
2. *EASEA-Cluster experiments*, performed on a 24-core machine. In this experiment 15 specialized islands (Exploitation, Exploration and Storage Islands) have been used, according to [24]. Table 2 displays the parameters of each type of island.

For each island connection events have been logged (timestamp, source, destination, individual to be transmitted and its fitness) for each sent/received individual throughout the execution. At the end of each experiment, log-files were collected and grouped in a single file to be displayed by GridVis.

4.2 Results

The *EASEA-Grid experiments* have been used to generate Figures 1 to 3, where de-synchronisation effects have been made evident using a temporal reordering, see figure 2(b). The second set of experiments is analysed below.

Parameter	20-Islands	100-Islands
Nb of generations	1000	500
Population size	2048	512
Crossover probability	0.8	0.8
Mutation probability	0.3	0.3
Surviving parents	100%	100%
Surviving offspring	100%	100%
Elitism	Strong	Strong
Elite	1	1

Table 1. Parameters for *EASEA-Grid*

	Island type		
Parameter	Exploring	Exploiting	Storing
Nb of islands	10	4	1
Nb of generations	70	70	70
Population size	40	40	40
Mutation probability	0.8	0.7	0.3
Crossover probability	0.8	0.7	0.3
Surviving parents	100%	100%	100%
Surviving offspring	100%	100%	100%
Elitism	Weak	Strong	Strong
Elite	0	1	1

Table 2. Parameters for *EASEA-Cluster*

A first global view has been generated for *EASEA-cluster* on Figure 4. Machines 2930 to 2939 are exploring, machines 2940 to 2943 are exploiting, and a single storing machine is used (number 2944). A temporal reordering shows the three clusters (Figure 5(a)), and colour time and colour fitness views (Figures 5(a) and 5(b)) make the different roles of machines types clear.

The following exchange rules have been used *(a)* Exploring machines send their individuals to every machine except the storing machine. *(b)* Exploiting machines send their individuals to every machine except the exploring machines. *(c)* The storing machine receives individuals but does not send any individuals.

Figure 5(a) shows a large cluster of machines (the 10 exploring machines), related by blue cells. They communicate efficiently. The smaller cluster of exploiting machines is in brown, which makes evident that exploiting machines have been started after exploring machines. Figure 5(b) then shows that higher fitness individuals are exchanged in the small cluster of exploiting machines (darker cells), which is coherent with the respective role of exploring and exploiting machines.

Now, let us examine the content of exchanges: as a machine always sends its best individual, the fitness of this individual is an instantaneous image of the state of the corresponding island. Exploring machines decrease their fitness quickly at the beginning of the evolution and then stagnate (Figure 6*(a)*), while exploiting machines improve their solutions later (figures 6*(b)* and 6*(c)*). This

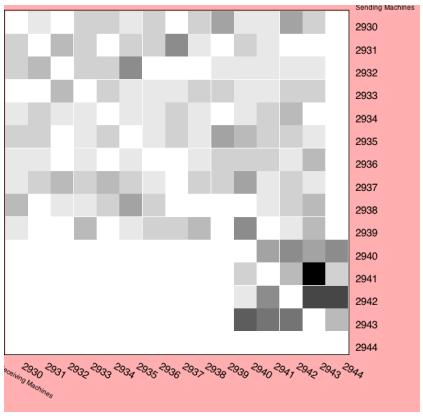
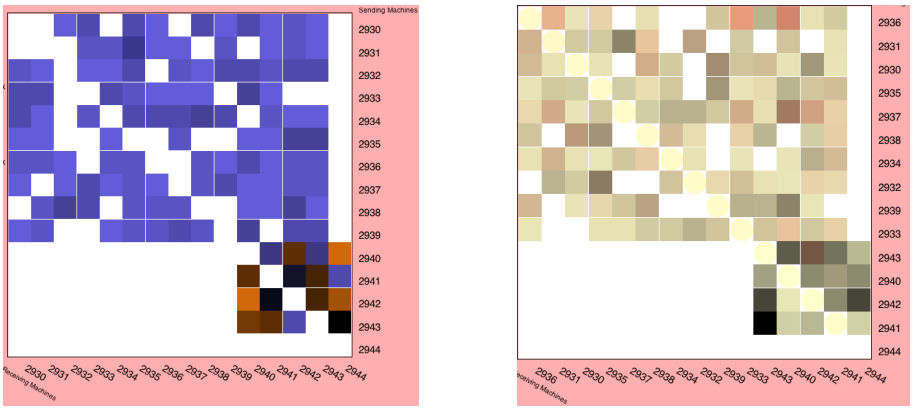


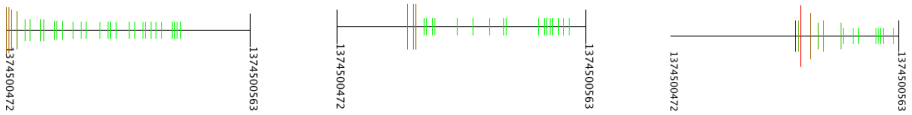
Fig. 4. *EASEA-cluster experiment*: global view for the 15 machines.



(a) Colour indicating time of interaction (from purple to orange).

(b) Colour indicating fitness (from yellow to red)

Fig. 5. *EASEA-cluster experiment* results. Rows and columns are ordered according to time.



(a) Exploring Sent (b) Exploiting Sent (c) Exploring Received

Fig. 6. *EASEA-cluster experiments*: Individuals sent by an exploiting machine (a), sent (b) and received (c) by an exploring machine.

fact can be verified on a view visualizing the end of the run (Figure 7): individuals that were sent by exploring machines are coloured red (indicating low performance) while better individuals are manipulated by the exploitation cluster. The role of the storing is to collect best results, it thus only receives dark yellow coloured individuals.

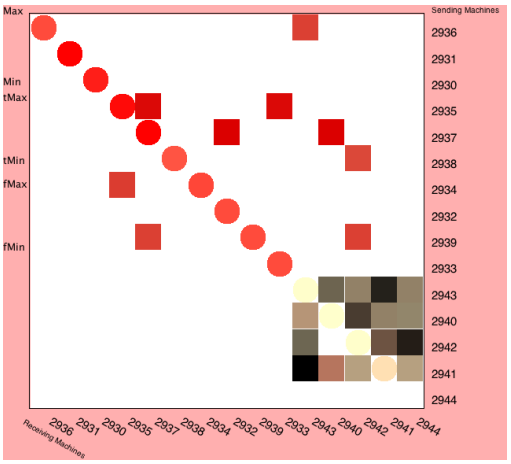


Fig. 7. *EASEA-cluster experiments*: Fitness information about the exchanges, zoom. Yellow areas correspond to better fitness (minisation aim).

5 Conclusions and future works

GridVis has proved to be a useful tool to understand the exchange of indivisuals in a grid or on a cluster of machines, runnig an island-based model. The activity of the machine is monitored and the time and quality of the exchanges are easily visualized. The kind of representation that GridVis offers helps the user with characterising a good launch on the grid, thus facilitates the parameters tuning task.

Future work will consider the development of GridVis for dynamic visusalisation, allowing on-line monitoring and parameter adjustments during execution. The integration of GridVis into a development framework based on EASEA for intensive computation purpose will also be considered (within the EASEA-Cloud Emergence project).

References

1. Whitley, D., Rana, S., Heckendorn, R.B.: The island model genetic algorithm: On separability, population size and convergence. Journal of Computing and Informa-

tion Technology **7** (1999) 33–48
2. Lutton, E., Fekete, J.D.: Visual analytics of ea data. In: Genetic and Evolutionary Computation Conference, GECCO 2011. (2011) July 12-16, 2011, Dublin, Ireland.
3. Lutton, E., Tonda, A., Gaucel, S., Foucquier, J., Riaublanc, A., Perrot, N.: Food model exploration through evolutionary optimization coupled with visualization: application to the prediction of a milk gel structure. In: From Model Foods to Food Models. DREAM Project's International Conference. (June 2013)
4. Pohlheim, H., AG, D.: Understanding the Course and State of Evolutionary Optimizations Using Visualization: Ten Years of Industry Experience with Evolutionary Algorithms. Artificial Life **12** (2006) 217–227
5. Spears, W.M.: An overview of multidimensional visualization techniques. In: Evolutionary Computation Visualization Workshop. (1999) T. D. Collins, editor, Orlando, Florida, USA.
6. Routen, T.: Techniques for the visualisation of genetic algorithms. In: The First IEEE Conference on Evolutionary Computation. Volume II. (1994) 846–851
7. Shine, W., Eick, C.: Visualizing the evolution of genetic algorithm search processes. In: Proceedings of 1997 IEEE International Conference on Evolutionary Computation, IEEE Press (1997) 367–372
8. Wu, A.S., Jong, K.A.D., Burke, D.S., Grefenstette, J.J., Ramsey, C.L.: Visual analysis of evolutionary algorithms. In: In Proceedings of the 1999 Conference on Evolutionary Computation (CEC'99), IEEE Press (1999) 1419–1425
9. Hart, E., Ross, P.: Gavel - a new tool for genetic algorithm visualization. IEEE Trans. Evolutionary Computation **5**(4) (2001) 335–348
10. Mach, M. Zetakova, Z.: Visualising genetic algorithms: A way through the Labyrinth of search space. In: Intelligent Technologies - Theory and Applications. IOS Press, Amsterdam (2002) 279–285 P. Sincak - J. Vascak - V. Kvasnicka - J. Pospichal (eds.).
11. Bedau, M.A., Joshi, S., Lillie, B.: Visualizing waves of evolutionary activity of alleles. In: Proceedings of the 1999 GECCO Workshop on Evolutionary Computation Visualization. (1999) 96–98
12. Bullock, S., Bedau, M.A.: Exploring the dynamics of adaptation with evolutionary activity plots. Artif. Life **12** (March 2006) 193–197
13. Pohlheim, H.: Visualization of evolutionary algorithms - set of standard techniques and multidimensional visualization. In: GECCO'99 - Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco, CA. (1999) 533–540
14. Pohlheim, H.: Geatbx - genetic and evolutionary algorithm toolbox for matlab http://www.geatbx.com/.
15. Computer, A.K., Kerren, A.: Eavis: A visualization tool for evolutionary algorithms. In: Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 05. (2005) 299–301
16. Parmee, I., Abraham, J.: Supporting implicit learning via the visualisation of coga multi-objective data. In: CEC2004, Congress on Evolutionary Computation, 19-23 June. Volume 1. (2004) 395 – 402
17. Collins, T.D. In: Visualizing evolutionary computation. Springer-Verlag New York, Inc., New York, NY, USA (2003) 95–116
18. Daida, J., Hilss, A., Ward, D., Long, S.: Visualizing tree structures in genetic programming. Genetic Programming and Evolvable Machines **6** (2005) 79–110
19. Kohl, J., Casavant, T.: A software engineering, visualization methodology for parallel processing systems. In: Computer Software and Applications Conference, 1992. COMPSAC '92. Proceedings., Sixteenth Annual International. (1992) 51–56

20. Morrow, T.M., Ghosh, S.: Divide: Distributed visual display of the execution of asynchronous, distributed algorithms on loosely-coupled parallel processors. In: In Proc. Visualization '93, IEEE Computer Society Press (1993) 166–173
21. Brown, J., Martin, P., Paku, N., Turner, G.: Visualisations of parallel algorithms for reconfigurable torus computers. In: Computer Human Interaction Conference, 1998. Proceedings. 1998 Australasian. (1998) 152–159
22. Price, B.A., Baecker, R., Small, I.S.: A principled taxonomy of software visualization. J. Vis. Lang. Comput. **4**(3) (1993) 211–266
23. Urquiza-Fuentes, J., Velázquez-Iturbide, J.A.: A survey of successful evaluations of program visualization and algorithm animation systems. Trans. Comput. Educ. **9**(2) (June 2009) 9:1–9:21
24. Maitre, O., Krueger, F., Querry, S., Lachiche, N., Collet, P.: Easea: specification and execution of evolutionary algorithms on gpgpu. Soft Computing **16**(2) (2012) 261–279
25. Collet, P., Lutton, E., Schoenauer, M., Louchet, J.: Take it EASEA. In Schoenauer, M., Deb, K., Rudolf, G., Yao, X., Lutton, E., J.J., M., Schwefel, H.P., eds.: Parallel Problem Solving from Nature - PPSN VI 6th International Conference, Paris, France, Springer Verlag (September 16-20 2000) LNCS 1917.
26. Tsutsui, S., Collet, P.: Massively Parallel Evolutionary Computation on Gpgpus. Natural Computing Series. Springer-Verlag New York Incorporated (2013)
27. Alba, E., Tomasini, M.: Parallelism and evolutionary algorithms. IEEE Transactions on Evolutionary Computation **6**(5) (October 2002) 443–462
28. Wilkinson, L., Friendly, M.: The history of the cluster heat map. The American Statistician **63**(2) (2009) 179–184
29. Brandes, U., Nick, B.: Asymmetric relations in longitudinal social networks. IEEE Transactions on Visualization and Computer Graphics **17**(12) (December 2011) 2283–2290
30. Bach, B., Pietriga, E., Fekete, J.D.: Visualizing Dynamic Networks with Matrix Cubes. In: SICCHI Conference on Human Factors in Computing Systems (CHI), Toronto, Canada, ACM (April 2014)
31. Pryke, A., Mostaghim, S., Nazemi, A.: Heatmap visualization of population based multi objective algorithms. In Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T., eds.: Evolutionary Multi-Criterion Optimization. Volume 4403 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2007) 361–375
32. Ghoniem, M., Fekete, J.D., Castagliola, P.: A comparison of the readability of graphs using node-link and matrix-based representations. In: Proceedings of the IEEE Symposium on Information Visualization. INFOVIS '04, Washington, DC, USA, IEEE Computer Society (2004) 17–24
33. Lutton, E., Collet, P., Louchet, J.: EASEA comparisons on test functions: Galib versus eo. In: EA01 Conference on Artificial Evolution, Le Creusot, France (October 2001)