Ephemeral Computing and Bioinspired Optimization
Challenges and Opportunities

Carlos Cotta[1], Antonio J. Fernández-Leiva[1], Francisco Fernández de Vega[2], Francisco Chávez[2], Juan J. Merelo[3], Pedro A. Castillo[3], David Camacho[4] and Gema Bello-Orgaz[4]

[1]*Dept. Lenguajes y Ciencias de la Computación, Universidad de Málaga, Málaga, Spain*

[2]*Dept. Tecnología de los Computadores y de las Comunicaciones, Universidad de Extremadura, Mérida, Spain*

[3]*Dept. Arquitectura y Tecnología de los Computadores, Universidad de Granada, Granada, Spain*

[4]*Dept. Ingeniería Informática, Universidad Autónoma de Madrid, Madrid, Spain*

{*ccottap,afdez*}*@lcc.uma.es,* {*fcofdez,fchavez*}*@unex.es,* {*jmerelo, pacv*}*@ugr.es,* {*david.camacho,gema.bello*}*@uam.es*

Keywords: Ephemeral Computing, Bioinspired Optimization, Evolutionary Computation, Complex Systems, Autonomic Computing, Distributed Computing.

Abstract: Computational devices with significant computing power are pervasive yet often under-exploited since they are frequently idle or performing non-demanding tasks. Exploiting this power can be a cost-effective solution for solving complex computational tasks. Device-wise, this computational power can some times comprise a stable, long-lasting availability windows but it will more frequently take the form of brief, ephemeral bursts, mainly in the presence of devices "lent" voluntarily by their users. A highly dynamic and volatile computational landscape emerges from the collective contribution of numerous such devices. Algorithms consciously running on these environments require specific properties in terms of flexibility, plasticity and robustness. Bioinspired algorithms are particularly well suited to this endeavor, thanks to their intrinsic features: decentralized functioning, intrinsic parallelism, resilience, and adaptiveness. The latter is essential to exert advanced self-control on the functioning and/or structure of the algorithm. Much has been done in providing self-adaptation capabilities to these techniques, yet the science of self-$\star$ bionspired algorithms is still nascent, in particular regarding to higher-level self-adaptation, and self-management in the context of large scale optimization problems and distributed ephemeral computing technologies. Deploying bioinspired techniques on this scenario will also pave the way for the application of other techniques on this computational domain.

1 EPHEMERAL COMPUTING: WHAT AND WHY

This paper revolves around the notion of *Ephemeral Computing* (Eph-C) which can be defined as *"the use and exploitation of computing resources whose availability is ephemeral (i.e., transitory and short-lived) in order to carry out complex computational tasks"*. The main goal in Eph-C is thus making an effective use of highly-volatile resources whose computational power (which can be collectively enormous) would be otherwise wasted or under-exploited. Think for example of the pervasive abundance of networked handheld devices, tablets and, lately, wearables –not to mention more classical devices such as desktop computers– whose computational capabilities are often not fully exploited. Hence, the concept of ephemeral computing partially overlaps with ubiquitous computing, volunteer computing and distributed computing (how these research fields deal with the concept of ephemerality is explained in next section) but exhibits its own distinctive features, mainly in terms of the extreme dynamism of the underlying resources, and the ephemerality-aware nature of the computation which autonomously adapt to the ever-changing computational landscape, not just trying to fit to the inherent volatility of the latter but even trying to use it for its own advantage.

In light of the computational context described above, it is clear that the algorithmic processes deployed onto it should be flexible (to work on a variety of computational scenarios), resilient (to cope with sudden failures and with the phenomenon of churn (Stutzbach and Rejaie, 2006)), (self-)adaptive (to react autonomously to changes in the environment and optimize its own performance in a smart way), and intrinsically decentralized (since centralized control

strategies cannot consistently comprehend the state of the computational landscape and decisions emerging from them would lag behind the changing conditions of the latter). Some bioinspired algorithms, like the Evolutionary algorithms (EAs) fit nicely into this scenario. However, few works have previously considered the interest of endowing evolutionary algorithms with the capability for coping with transient behaviors in underlaying computer systems. Moreover, in the age when the term Big Data (Lohr, 2012) is present in many initiatives requiring large amount of computational resources for storing, processing, and learning from huge amount of data, new methods and algorithms for properly managing heterogeneous computing resources widely distributed along the world are required. Energy consumption must also be considered from the point of view of both algorithms and hardware resources, given the large differences among large computing infrastructures typically devoted to running massively parallel algorithms when compared to smart devices optimized for reducing battery consumption. It is of the foremost interest to research on the basic features allowing to provide efficient and reliable ephemeral evolutionary services.

2 EPHEMERAL COMPUTING IN PERSPECTIVE

According to the Oxford Dictionary, the term *ephemeral* means "lasting for a very short time". It thus encompasses things or events with a transitory nature, with a brief existence. A number of phenomena and resources in computer science are endowed with that feature (e.g. in computing networking, an ephemeral port is a TCP port, for instance, dynamically assigned to a client application for a brief period of time, in contrast with well known ports) (Borella et al., 2000). Ephemeral behaviors can be also observed in the way users collaborate in volunteer networks of computers.

Although ephemeral phenomena naturally arise in several areas such as ubiquitous computing, volunteer computing or traditional research areas like distributed computing, some issues arise when dealing with ephemeral behavior. In cloud computing (Armbrust et al., 2010), for instance, the opposite is usually looked for: *persistence*. Although services are commonly associated with computations among autonomous heterogeneous parties in dynamic environments, exceptions must be handled to take corrective actions. Ephemeral services are thus commonly seen more as a problem than a solution (Huhns and Singh, 2005).

On the other hand, in ubiquitous computing the main goal is to leverage computation everywhere and anywhere, so that computation can occur using any kind of device, in any location, starting and ending at any time and using any format and during any amount of time. The main efforts in this area have been oriented to design and develop the underlying technologies needed to support ubiquitous computing (Lyytinen and Yoo, 2002) (like advanced middleware, operating systems, mobile code, sensors, microprocessors, new I/O and user interfaces, networks or mobile protocols). However, and in the same way it happens with cloud computing, the main target in ubiquitous computing is to allow stable and persistent computation processes perform a complete execution of the programs. When this area handles the concept of ephemeral devices, services or computation, the main solution is to stop the process, or processes, and resume once new devices are ready (Wang et al., 2004). Previous hypothesis and assumptions can be extrapolated to distributed computing, where the concept of ephemeral services can be a problem that could eventually generate a failure in the execution of the process (Sharmin et al., 2005).

As stated before, the main focus of Eph-C is different from the above approaches: rather than trying to build layers onto the network of ephemeral resources in order to "hide" their transient nature and provide the illusion of a virtual stable environment, Eph-C applications are fully aware of the nature of the computational landscape and are specifically built to live (and optimize their performance) in this realm. Note this does not imply the latter have a lower-level vision of the underlying computational substrate, or at least not markedly so. In fact, most low-level features can be abstracted without precluding attaining a more accurate vision of this fluctuating substrate.

To some extent, some of these ephemerality issues are also present in areas such as volunteer computing (VC) (Sarmenta and Hirano, 1999), whereby a dynamic collection of computing devices collaborate in solving a massive computational task, decomposing it into small processing chunks. Most VC approaches follow a centralized master/slave scheme though, and typically deal with resource volatility via redundant computation. A much more decentralized, emergent approach can be found in amorphous computing (Abelson et al., 2000), but that paradigm is more geared towards programmable materials and their use to attack massive simulation problems. Massive problems are also the theme in ultrascale computing, where issues such as scalability, resilience to failures, energy management, and handling of large volume of data are of paramount importance (Kamil

et al., 2005; Network for Sustainable Ultrascale Computing, 2014). Note however Eph-C is not necessarily exascale nor it is oriented towards supercomputing.

3 BIOINSPIRED ALGORITHMS AND EPHEMERALITY

The term *bioinspired algorithms* usually refers to methods that draw some inspiration from Nature to solve search, optimization or pattern recognition problems. If we focus on optimization problems, the most prominent bioinspired paradigms are evolutionary computation and swarm intelligence. We are particularly interested in these kinds of population-based search and optimization algorithms, which have a natural path to distributed computing by simply distributing the population among the different computing nodes, the issue being how to do it in an algorithmically efficient and scalable way. Eph-C, besides the obvious fact that the contribution of a node might come and go at any time, adds several other dimensions to the design of algorithms:

- *Inclusion*: all nodes should have a meaningful contribution to the final result, and they should be incorporated to the distributed system in such a way that they do.
- *Asynchrony*: nodes communicate with the others without a fixed schedule due to their different performance.
- *Resilience*: the sudden disappearance of computing nodes must not destabilize the functioning of the algorithm.
- *Emergence*: the nature of the computational environment does not allow a centralized control and requires decentralized, emergent behavior.
- *Self-adaptation*: the algorithm should adapt itself to the changing computational landscape.

This latter issue is particularly important, and encompasses a number of self-$\star$ properties (Babaoglu et al., 2005) the system must exhibit in order to exert advanced control on its own functioning and/or structure, e.g., self-maintaining in proper state, self-healing externally infringed damage (Frei et al., 2013), self-adapting to different environmental conditions (Nogueras and Cotta, 2015b), and even self-generating new functionalities just to cite a few examples, see also (Cotta et al., 2008; Eiben, 2005). Quite interestingly, these properties are frequently intrinsic features of the system, that is, emergent properties of its complex structure, rather than the result of endowing it with a central command. This also implies there is no need for a central control in the system; every node schedules itself. This decentralization implies a certain fault-tolerance due to the lack of a single point of failure, but it also means resilience must be built into the algorithms present in each node so that their sensitivity to changes in the rest of the system is minimal. This will include measures such as population sizing and the conservation of diversity in each node, as indicated by Cantú-Paz in (Cantú-Paz, 1998) but taken to new meanings in this context. Indeed, models and algorithms have to be designed to be fault-tolerant (Nogueras and Cotta, 2015a) so that inclusion of new nodes will be done in a self-adaptive way, but also in such a way that its disappearance from the network will not have a big impact on performance. In fact, VC systems, which are an early example of Eph-C, have been proved to be fault tolerant to a certain point (González Lombraña et al., 2010), but this fault tolerance will have to be taken into account not just at the implementation level (backing up solutions, for instance) but also at the model and algorithm level, measuring the impact of different churn models (Laredo et al., 2008; Nogueras and Cotta, 2015c).

4 EPHEMERAL COMPUTING-BASED APPLICATIONS

This section provides a short revision on some of those bioinspired methods and applications that could be affected by Eph-C characteristics.

4.1 Big Data & Bio-inspired Clustering

The data volume and the multitude of sources have experienced an exponential growing with a new technical and application challenges. The data generation has been estimated as 2.5 quintillion bytes of data per day[1]. This data comes from everywhere: sensors used to gather climate, traffic, air flight information, posts to social media sites (i.e. Twitter or Facebook as popular examples), digital pictures and videos (YouTube users upload 72 hours of new video content per minute[2]), purchase transaction records, or cell phone GPS signals to name a few. The classic methods, algorithms, frameworks or tools for data management have become both, inadequate for

[1] http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html

[2] http://aci.info/2014/07/12/the-data-explosion-in-2014-minute-by-minute-infographic/)

processing these amount of data, and unable to offer effective solutions to deal with the data growing. The management, handling and extraction of useful knowledge from these data sources is currently one of the most popular and hot topics in computing research.

In this context, Big Data is a popular phenomenon which aims to provide an alternative to traditional solutions database and data analysis, leading to a revolution not only in terms of technology but also in business. It is not just about storage of and access to data, Big Data solutions aim to analyze data in order to make sense of that data and exploiting its value.

One of the current main challenges in Data Mining related to Big Data problems is to find adequate approaches to analyze massive data online (or data streams). Due classification methods requires from a previous labelling process, these methods need high efforts for real-time analysis. However, due to unsupervised techniques do not need this previous process, clustering becomes a promising field for real-time analysis. Clustering is perhaps one of the most popular approaches used in *unsupervised machine learning* and in Data Mining (Han and Kamber, 2006). It is used to find hidden information or patterns in an unlabelled dataset and has several applications related to biomedicine, marketing (Haider et al., 2012), or image segmentation (Pascual et al., 1999) amongst others. Clustering algorithms provide a large number of methods to search for "blind" patterns in data, some of these approaches are based on Bio-inspired methods such as evolutionary computation (Goldberg, 1989), swarm intelligence (Bonabeau et al., 1999) or neural networks amongst others.

In the last years, and due to the fast growing of a large Big Data-based problems, new challenges are appearing in previous research areas to manage the new features and problems that these type of problems produce. New kinds of algorithms, as *online clustering* or *streaming clustering* are appearing to deal with the main problems related to Big Data domains. When data streams are analyzed, it is important to consider the analysis goal, in order to determine the best type of algorithm to be used. We could divide data stream analysis in two main categories:

- *Offline analysis*: we consider a portion of data (usually large data) and apply an offline clustering algorithm to analyze this data.
- *Online analysis*: the data are analyzed in real-time. These kinds of algorithms are constantly receiving new data instances and are not usually able to keep past information. The most relevant limitations of these systems are: the data order matters and can not be modified; the data can not be stored or re-analyzed during the process; the results of the analysis depend of the time the algorithm has been stopped. The main problem of these algorithms is that they need a specific space to update the information. This reduces the possibilities of the new algorithm.

From our previous experience in different complex and industrial problems in different areas from Social Networks Analysis (Bello-Orgaz et al., 2014; Bello-Orgaz et al., 2012), Project Scheduling, Videogames, Music classification, Unmmaned Systems, or Bio-informatics, we have designed and developed several bioinspired algorithms for clustering or graph-based computing with the aim to handle Big Data-based problems. We can distinguish from two main types of algorithms, those that have combined evolutionary strategies (mainly genetic algorithms) (Menéndez et al., 2014a; Menéndez et al., 2014b) and the second ones which have been designed using swarm intelligence approaches (ant colonies optimization algorithms) (Gonzalez-Pardo and Camacho, 2015).

4.2 Social-based analysis and mining

With the large number and fast growing of Social Media systems and applications, Social-based applications for Data Mining, Data Analysis, Big Data computation, Social Mining, etc. has become an important and hot topic for a wide number of research areas. Although there exists a large number of existing systems (e.g., frameworks, libraries or software applications) which have been developed, and currently are used in various domains and applications based on Social Media. The applications and their main technologies used are mainly based on Big Data, Cloud or Grid Computing. The concept of Ephemeral computing has been rarely considered.

Most of the current challenges under study in Social-based analysis and mining are related to the problem of efficient knowledge representation, management and discovery. Areas as Social Network Analysis (SNA), Social Media Analytics (SMA) and Big Data, have as main aims to track, trends discovery or forecasting, so methods and techniques from: Opinion Mining, Sentiment Analysis, Multimedia management or Social Mining are commonly used. For example, when anyone tries to analyze how a Social Network is evolving using a straightforward representation based on a graph, but ignoring the information flow between nodes the information extracted from this analysis will be very limited. Other simple example, based on SNA, is an application that could try to extract behavioral patterns among users

connected to a particular social network without taking into account their connections, their strengthens, or how their relationships are evolving through time. Social Big Data analysis, instead, aims to study large-scale Web phenomena such as Social Networks from a holistic point of view, i.e., by concurrently taking into account all the socio-technical aspects involved in their dynamic evolution.

Previous domains could be joined into a more general application area named *Social Big Data*. This area, or application domain, comes from the joining efforts of two domains: Social Media and Big Data. Therefore, Social Big Data will be based on the analysis of very-large to huge amount of data, which could belong to several distributed sources, but with a strong focus on Social media. Hence, Social Big Data analysis (Cambria et al., 2013; Manovich, 2011) is inherently interdisciplinary and spans areas such as Data Mining, Machine Learning, Statistics, Graph Mining, Information Retrieval, Linguistics, Natural Language Processing, Semantic Web, Ontologies, or Big Data Computing, amongst others. Their applications can be extended to a wide number of domains such as health and political trending and forecasting, hobbies, e-business, cyber-crime, counter terrorism, time-evolving opinion mining, social network analysis, or human-machine interaction.

Taking into account the nature of Social Big Data sources and the necessary processes and methods that will be required for data processing, the knowledge models, and possibly the analysis and visualization techniques to allow discover meaningful patterns (Kaisler et al., 2013), the potential application of Eph-C features could generate a new kind of algorithms that would be suitably applied in ephemeral environments.

5 CONCLUSIONS

Ephemeral computation provides an interesting new, and promising, research area with significant differences when it is compared against other areas as grid computing, or traditional distributed computing. Although Eph-C presents some features close to volunteer computing or amorphous computing, the combination of their main features: *inclusion, asynchrony, resilience, emergence, and self-adaptation*, defines it more precisely.

Therefore, the main focus of Eph-C is different from the above approaches, rather than trying to build layers onto the network of ephemeral resources in order to "hide" their transient nature and provide the illusion of a virtual stable environment, Eph-C applications are fully aware of the nature of the computational landscape and are specifically built to live (and optimize their performance) in this realm.

Related to the application of traditional methods and techniques from Machine Learning to Big Data problems, our previous experience has shown the high performance that bioinspired algorithms can achieve in huge, open and dynamic problems, showing how bioinspired approaches can be used to improve the performance of unsupervised approaches. In the near future, and taking into account the new restrictions and features imposed by Eph-C environments, a new suit of algorithms able to efficiently handle the new challenges in data management and knowledge discovery in large Big Data-based problems will be studied and analyzed.

ACKNOWLEDGEMENTS

This work is supported by MINECO project EphemeCH (TIN2014-56494-C4-1-P, -2-P, -3-P and -4-P) – Check `http://blog.epheme.ch`.

REFERENCES

Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Knight Jr, T. F., Nagpal, R., Rauch, E., Sussman, G. J., and Weiss, R. (2000). Amorphous computing. *Communications of the ACM*, 43(5):74–82.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al. (2010). A view of cloud computing. *Communications of the ACM*, 53(4):50–58.

Babaoglu, O. et al., editors (2005). *Self-star Properties in Complex Information Systems*, volume 3460 of *Lecture Notes in Computer Science*. Springer, Berlin Heidelberg.

Bello-Orgaz, G., Menéndez, H., Okazaki, S., and Camacho, D. (2014). Combining social-based data mining techniques to extract collective trends from twitter. *Malaysian Journal of Computer Science*, 27(2).

Bello-Orgaz, G., Menendez, H. D., and Camacho, D. (2012). Adaptive k-means algorithm for overlapped graph clustering. *International journal of neural systems*, 22(05).

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Inc., New York, NY, USA.

Borella, M. S., Grabelsky, D., Nessett, D. M., and Sidhu, I. S. (2000). Method and system for locating network services with distributed network address translation. US Patent 6,055,236.

Cambria, E., Rajagopal, D., Olsher, D., and Das, D. (2013). Big social data analysis. *Big data computing*, 13:401–414.

Cantú-Paz, E. (1998). A survey of parallel genetic algorithms. *Calculateurs paralleles, reseaux et systems repartis*, 10(2):141–171.

Cotta, C., Sevaux, M., and Sörensen, K., editors (2008). *Adaptive and Multilevel Metaheuristics*, volume 136 of *Studies in Computational Intelligence*. Springer.

Eiben, A. (2005). Evolutionary computing and autonomic computing: Shared problems, shared solutions? In (Babaoglu et al., 2005), pages 36–48.

Frei, R., McWilliam, R., Derrick, B., Purvis, A., Tiwari, A., and Di Marzo Serugendo, G. (2013). Self-healing and self-repairing technologies. *International Journal of Advanced Manufacturing Technology*, 69(5-8):1033–1061.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 1st edition.

González Lombraña, D., Laredo, J. L. J., Fernández de Vega, F., and Merelo Guervós, J. J. (2010). Characterizing fault-tolerance of genetic algorithms in desktop grid systems. In *Evolutionary Computation in Combinatorial Optimization*, pages 131–142. Springer.

Gonzalez-Pardo, A. and Camacho, D. (2015). Solving project scheduling problems through swarm-based approaches. *International Journal of BioInspired Computation (IJBIC)*, In press.

Haider, P., Chiarandini, L., and Brefeld, U. (2012). Discriminative clustering for market segmentation. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 417–425, New York, NY, USA. ACM.

Han, J. and Kamber, M. (2006). *Data mining: concepts and techniques*. Morgan Kaufmann.

Huhns, M. N. and Singh, M. P. (2005). Service-oriented computing: Key concepts and principles. *Internet Computing, IEEE*, 9(1):75–81.

Kaisler, S., Armour, F., Espinosa, J. A., and Money, W. (2013). Big data: Issues and challenges moving forward. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 995–1004. IEEE.

Kamil, S., Shalf, J., Oliker, L., and Skinner, D. (2005). Understanding ultra-scale application communication requirements. In *Workload Characterization Symposium, 2005. Proceedings of the IEEE International*, pages 178–187. IEEE.

Laredo, J. L. J., Castillo, P. A., Mora, A. M., Merelo, J. J., and Fernandes, C. (2008). Resilience to churn of a peer-to-peer evolutionary algorithm. *Int. J. High Performance Systems Architecture*, 1(4):260–268.

Lohr, S. (2012). The age of big data. *New York Times*, 11 February. [Online; accessed 5-September-2014].

Lyytinen, K. and Yoo, Y. (2002). Ubiquitous computing. *Communications of the ACM*, 45(12):63–96.

Manovich, L. (2011). Trending: the promises and the challenges of big social data. *Debates in the digital humanities*, pages 460–475.

Menéndez, H. D., Barrero, D. F., and Camacho, D. (2014a). A genetic graph-based approach for partitional clustering. *International journal of neural systems*, 24(03).

Menéndez, H. D., Otero, F. E., and Camacho, D. (2014b). Macoc: a medoid-based aco clustering algorithm. In *Swarm Intelligence*, pages 122–133. Springer International Publishing.

Network for Sustainable Ultrascale Computing (2014). The future of ultrascale computing under study. [Online; accessed 8-September-2014].

Nogueras, R. and Cotta, C. (2015a). Studying fault-tolerance in island-based evolutionary and multimemetic algorithms. *Journal of Grid Computing*. doi:10.1007/s10723-014-9315-6 [online].

Nogueras, R. and Cotta, C. (2015b). Studying self-balancing strategies in island-based multimemetic algorithms. *Journal of Computational and Applied Mathematics*. doi:10.1016/j.cam.2015.03.047 [online].

Nogueras, R. and Cotta, C. (2015c). Towards resilient multimemetic systems on unstable networks with complex topology. In Papa, G., editor, *Advances in Evolutionary Algorithm Research*. Nova Science Pub. In Press.

Pascual, A., Barcéna, M., Merelo, J., and Carazo, J.-M. (1999). Application of the fuzzy kohonen clustering network to biological macromolecules images classification. In Mira, J. and Sánchez-Andrés, J., editors, *Engineering Applications of Bio-Inspired Artificial Neural Networks*, volume 1607 of *Lecture Notes in Computer Science*, pages 331–340. Springer Berlin Heidelberg.

Sarmenta, L. F. and Hirano, S. (1999). Bayanihan: Building and studying web-based volunteer computing systems using java. *Future Generation Computer Systems*, 15(5):675–686.

Sharmin, M., Ahmed, S., and Ahamed, S. I. (2005). Saferd (secure, adaptive, fault tolerant, and efficient resource discovery) in pervasive computing environments. In *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, volume 2, pages 271–276. IEEE.

Stutzbach, D. and Rejaie, R. (2006). Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC '06, pages 189–202, New York, NY, USA. ACM.

Wang, B., Bodily, J., and Gupta, S. K. (2004). Supporting persistent social groups in ubiquitous computing environments using context-aware ephemeral group service. In *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*, pages 287–296. IEEE.