A Flexible P2P Gossip-based PSO Algorithm

Marco Biazzini
INRIA – Bretagne Atlantique
Rennes, France
Marco.Biazzini@inria.fr

***Abstract*—It is becoming more and more interesting, in the domain of distributed function optimization, the study of fully decentralized optimization algorithms, deployed on large networks of heterogeneous computational units. Several issues arise on such a system design, among which the proper way of distributing and making use of shared information, in absence of a centralized coordination, is a prominent one. We introduce the design of a P2P gossip-based Particle Swarm Optimization (PSO) algorithm, that is capable to implement different policies with respect to the use of global information, as this becomes available via gossiping during the computation. Such a PSO flavor is easy to tune, in order to implement different strategies, while balancing exploration and exploitation. Preliminary experimental results are shown to assess the usefulness of the proposal.**

Keywords-P2P function optimization; distributed Particle Swarm Optimization; P2P distributed computation.

I. INTRODUCTION

Distributed function optimization has a long research history [1]. Usual applicative scenarios assume the availability of either a dedicated parallel computing facility, or specialized clusters of machines. In both cases, the coordination of the distributed task is modeled in a centralized fashion, greatly simplifying its management. The limitations shown by the scalability and the robustness of these approaches are well known.

Recently, researchers have paid increasing attention to systems organized in decentralized P2P networks of solvers, distributed on a large collection of loosely-coupled machines, that cooperate to solve a common task [2], [3]. The long term goal of this kind of studies is to come up with an algorithmic design that can provide reliably good results in unsupervised and possibly heterogeneous systems.

The common requirement in these cases is that the optimization tasks must be successfully and effectively performed without any specialized infrastructure or central coordinating server being required. Ideally, these systems should self-organize themselves in a completely decentralized way, avoiding single points of failure and performance bottlenecks. The advantages of such approach are thus extreme robustness and scalability, plus the capability of exploiting existing (unused or underused) resources, like idle computer labs within a given organization, or a volunteer computing system architecture.

A reasonable approach is to partition the optimization job in a pool of independent tasks to be performed, and assign them to the available nodes, taking care of balancing the load. This can be done either using a centralized scheduler, or using a decentralized approach. This multi-algorithm approach is well known and widely used and it can be achieved in a decentralized fashion as well [4]. Anyway, it kind of "smoothes down" the challenge of finding a distributed algorithmic design, in that it uses each interconnected machine as a separate solver, rather than finding a proper decentralized design for a given algorithm to enable the cooperation of several, possibly very numerous resources.

An interesting research trend investigates a P2P approach, where a distributed algorithm spreads the load of a *single* optimization task among a group of nodes, in a robust, decentralized and scalable way [5]. By offering such a possibility, the need of solving an optimization task in a bounded time and/or with a given precision could be achieved by easily deploying identical solvers in a large-scale network and either focusing on the quality (to obtain a more accurate result by a specific deadline) or on the speed-up (to perform a predefined amount of computation over a function in the shortest possible time).

In this paper we present a novel Particle Swarm Optimization (PSO) design, that can exploit such a distributed environment and maximize the impact of a gossip-based information sharing mechanism, to avoid getting stuck in suboptimal region of the problem domain. Experiments in a real deployment show the viability of the approach and the effectiveness of the design.

In the following, Section II presents a brief description of the standard PSO algorithm and discuss some of its distributed variants. Section III outlines the distributed PSO algorithm we devise and details the novelty of its design. Section IV characterizes the distributed scenario we target and implement in our experiments. Then experimental results are presented and discussed in Section V. We draw our conclusion in Section VI.

II. BACKGROUND

We provide in this section a brief description of the standard PSO algorithm. We also recall some recent works on PSO in fully decentralized systems, to better contextualize our contribution.

A. Particle Swarm Optimization

PSO [6] is a nature-inspired method for finding global optima of functions of continuous variables. The search is performed iteratively updating a small number N (usually in the tens) of random "particles" (solvers), whose status information includes the current position vector $\boldsymbol{x}_i$, the current speed vector $\boldsymbol{v}_i$, the optimum point $\boldsymbol{p}_i$ and the *fitness* value $f(\boldsymbol{p}_i)$, which is the "best" solution the particle has achieved so far. The particle swarm optimizer also tracks the global best

 ISBN: 978-1-61208-318-6

position $\boldsymbol{g}$, in which the swarm has achieved the best fitness value obtained so far by any particle in the population.

At each iteration, every particle updates its velocity and position as described by the following equations:

$$\boldsymbol{v}_i = \boldsymbol{v}_i + c_1 \cdot rand() \cdot (\boldsymbol{p}_i - \boldsymbol{x}_i) + c_2 \cdot rand() \cdot (\boldsymbol{g} - \boldsymbol{x}_i) \quad (1)$$

$$\boldsymbol{x}_i = \boldsymbol{x}_i + \boldsymbol{v}_i \quad (2)$$

In these equations, $rand()$ is a random number in the range $[0, 1]$, while c_1 and c_2 are learning factors, whose default values are conventionally set as $c_1 = c_2 = 2$. The pseudo code of the procedure is given in Algorithm 1.

```
foreach particle i do
  | Initialize i;
end
while maximum iterations or
      minimum error criteria is not attained do
  | foreach particle i do
  | | Compute current fitness value f(x_i);
  | | if f(x_i) is better than f(p_i) then
  | | | p_i ← x_i;
  | | end
  | end
  | g ← bestOf(p_i), i = 1 to N;
  | foreach particle i do
  | | Compute velocity v_i according to equation 1;
  | | Update position x_i according to equation 2;
  | end
end
```

Algorithm 1: The standard PSO algorithm.

Particle speeds on each dimension are bounded to a maximum velocity $vmax_i$, specified by the user.

B. PSO on incomplete topologies

As one of the most investigated heuristics inspired from nature, PSO is the subject of study of a vast scientific production [7]. Given the remarkable behavioral diversity that can be obtained by tuning its parameters and shaping the way swarms interact with each others, numerous distributed variants have been brought up as well [8], [9], [10].

The above-described version of PSO assumes that all particles agree on the global best point found so far, and is often referred to as the "classical" or "full-information" version. Effects of incomplete topologies on the performance of PSO have been studied for different types of graphs [11]. Such studies were motivated by the observation that incomplete topologies may prevent the system from concentrating too much on early-found local optima, therefore improving solution quality. Full information has generally been shown to outperform partial topologies [12]. Yet, our work focuses on cases where incomplete information is a consequence of the network topology, and global data maintenance is not practical. Some recent publications presented PSO flavors adapted for P2P overlay networks [13], [14].

Improvements with respect to naiver versions and robustness even in faulty environments have been shown [15], [16], that rely mostly on the periodic diffusion of the current best solution among the distributed solvers and exploit the effectiveness of gossip protocols in spreading relevant information among peers. In general, gossip-based distributed computing has shown to be able to drive the gradual improvement of evolutionary algorithms, while achieving both scalability and quality [17].

Within this context, it is still poorly understood how to optimally use the information that is gossiped from node to node. We argue that is not only the rate of gossiping that affects the performance, as it has been shown [18]. Once we have the most up-to-date information available at each peer, the local solver still can choose whether to use it as soon as possible, or to schedule the utilization in a strategic way. The contribution of this paper is to present the design of a P2P gossip-based PSO algorithm that is capable to implement different policies with respect of the use of global information. Differing from the works cited above, our approach focus on the way information generated remotely is handled locally at each solver, rather than on studying the performance of the information spreading protocol, some global population control mechanism or the optimal setting of the basic PSO parameters.

III. Algorithm Description

The distributed PSO algorithm we propose offers a novel way to improve the exploration of the search domain, not to cut the search short towards the current best solution (likely to be suboptimal). The idea is to have an algorithm that can choose among different policies. These policies determine how and when the global information about the current best value found — available at any time as communicated by the other peers — should be used. At least two good reasons not to immediately consume the shared data come to mind:

1) to mitigate the event of an early convergence to suboptimal attraction basins, by not moving too fast towards the current optimal value;
2) to enhance the exploration of the search space, by avoiding that the swarms get too close to each other at an early stage.

By applying policies that define how to use the knowledge about the current global optimum, the PSO algorithm can be tuned in a way that is easier for the user to understand, with respect to the tuning of the various PSO parameters, whose behavioral effects are often obscure or at least debatable. We are aware that top quality results on hard problems can only be achieved by a careful and clever tuning of the algorithm on function characteristics. Our contribution and the results of our experiments point out, nonetheless, that attention should be paid not only to the core algorithmic parameters, but also to the way the shared information is diffused and consumed by the various agents of the distributed optimization task.

The key idea of our proposal is to maintain an ever-refreshing knowledge of the best point evaluated so far by any swarms in the network, but without necessarily substituting the local swarm's global best with this value. The overall global best should be rather used at a time and in a way that serves a given strategy. Table I gives the description of the notation we use in the following.

 ISBN: 978-1-61208-318-6

TABLE I: Description of the adopted notation.

Notation	*Meaning*
gb	the local swarm's global best
ogb	the overall global best, periodically gossiped among the solvers
p	the given policy to apply
E	overall number of function evaluations performed in the network

We can describe the general design of our distributed PSO algorithm as follows. Each swarm iteratively performs these steps:

1) *update* the **ogb** via a gossip message exchange
2) *apply* **p** to decide about using either **gb** or **ogb** to move the current particle
3) *move* the particle according to the decision taken
4) *evaluate* the function in the particle's position
5) *update* the records about the local and global best values (both **gb** and **ogb**) as needed

The strategic decisions that will impact the behavior of the algorithm are then implemented in the policy **p**. The policy may be simple or very complex, may use a limited amount of local knowledge or it may use any shared information (beside the value of **ogb**) that can be made available via peer-wise communication among the solvers. Trying to give a minimal set of requirements, we consider that a good policy should indicatively specify:

– how to decide to use **ogb** in a given PSO iteration (what triggers the decision, if it depends and involves the whole swarm or the single particles, etc.);
– for how many subsequent iterations **ogb** will substitute **gb** to compute the speed of the particles (how long each application of the current global optimum will last in the local solver);
– when this substitution shall permanently or temporarily end (what determines the end of each "**ogb** session").

In Section V we show experimental results obtained by running the described distributed PSO algorithm with a simple policy.

IV. Distributed Framework Characteristics

We target the general framework described in [18], thus considering a parallel islands scenario, in which several swarms of particles are initialized at random over a function domain. Each swarm is hosted by a peer and peers are distributed in a random overlay. During the search, every swarm periodically exchanges information with another swarm hosted by a peer, that is selected at random from the local neighborhood. At each peer, the neighborhood is maintained by means of a peer sampling service, implemented by the NEWSCAST gossip protocol [19].

All the communication mechanisms are based on gossip algorithms implemented on top of this service. Each peer always propagates the current best solution to others, by periodically sending it to one randomly chosen peer. Upon receiving this information, a peer updates its own current best solution, but only if the received one is better. If this is not the case, no further information is sent back to the sender. Thus the epidemic protocol implements a simple *push* approach. The period of gossip is assumed to be at least that of one function evaluation, thus a communication event is triggered after each function evaluation at all peers. As it is known by the behavior of the epidemic protocols, the time to propagate this way a new-found best solution to every node is logarithmic (in expectation) with respect to the size of the network.

This kind of lightweight and asynchronous communication among swarms suits well a large-scale, possibly heterogeneous environment. Though it could be beneficial in terms of absolute performance, the swarms are not required to perform a similar number of function evaluations in a given time, nor they have tight time constraints to perform mutual information exchanges.

TABLE II: Test functions. L: number of local minima.

	Function $f(x)$	L
Rosenbrock20	$\sum_{i=1}^{19} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	1
Zakharov20	$\sum_{i=1}^{20} x_i^2 + (\sum_{i=1}^{20} i x_i/2)^2 + (\sum_{i=1}^{20} i x_i/2)^4$	1
Rastrigin20	$200 + \sum_{i=1}^{20} [x_i^2 - 10\cos 2\pi x_i]$	$\approx 10^6$
Griewank20	$\sum_{i=1}^{20} x_i^2/4000 - \prod_{i=1}^{20} \cos\left(x_i/\sqrt{i}\right) + 1$	$\approx 10^{19}$

V. Experimental Results

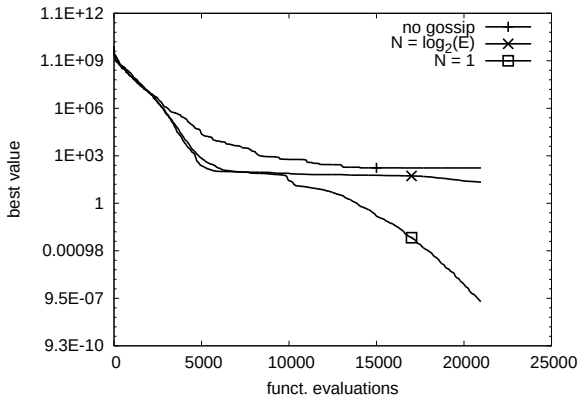
The results presented in this section have been obtained in a real distributed environment. We use our open source Java implementation of a distributed optimization framework [20] and the grid facilities provided by Grid5000 [21] In this kind of experiments we do not focus on the absolute performance of the algorithm, but rather on the differences among the adopted configurations.

We deploy 50 solvers (swarms) on an equal number of machines on the grid. Their random P2P overlay is maintained by the NEWSCAST gossip protocol, running on each machine. At each peer, the local neighborhood constantly represents a random subset of the network. At the beginning of each PSO iteration, a swarm updates its local information according to the messages that have been received since the update phase of the previous iteration. At the end of each PSO iteration, a peer solver is selected at random by each peer in a local neighborhood of 20 peers as the recipient of an update message. No churn and no faults are considered in this scenario. At each peer, the solver consists of a swarm of 4 particles, whose parameters are set as follows: $w_1 = 0.9$, $w_2 = 0.4$, $c_1 = 2$, $c_2 = 2$.

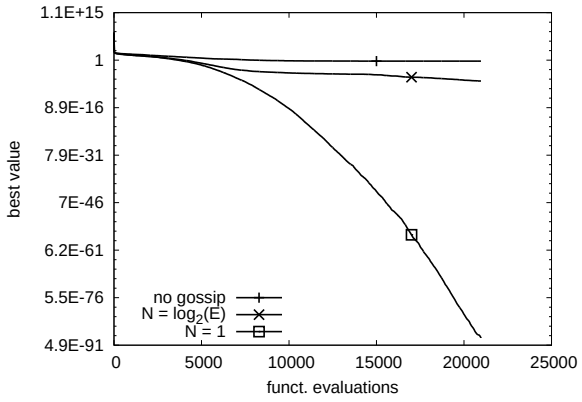
We evaluate four well known benchmark functions, described in Table II. They differ in the number and the distribution of their local minima, whereas the value of the global minimum is 0 for all of them. We perform 10 runs for each experiment, taking the average best value. The overall number of function evaluations in the network, equally partitioned among the solvers, is set as $E = 2^{20}$. The policy adopted for these experiments is the following:

– start using **ogb** instead of **gb** whenever **gb** has not been improved in the latest N iterations;

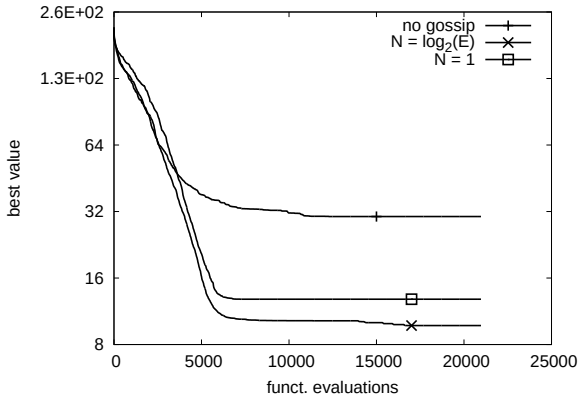
 ISBN: 978-1-61208-318-6



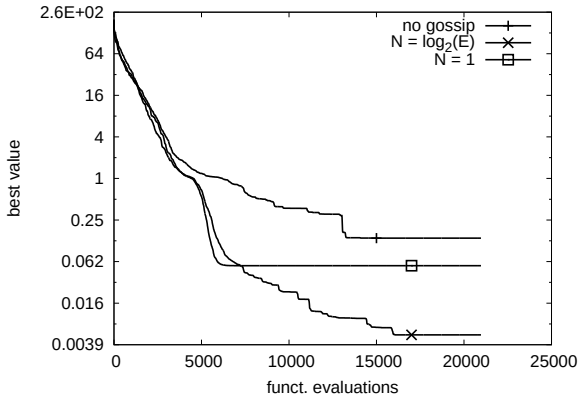
(a) Rosenbrock20



(b) Zakharov20



(c) Rastrigin20



(d) Griewank20

Fig. 1: Results on unimodal and multimodal test functions.

– continue using **ogb** until each particle in the swarm has been moved at least once *AND* **gb** is not improved (this being both a duration and a stopping criterion).

For each function, we run different experiments by choosing $N = log_2(E)$ and $N = 1$. This latter value means that **ogb** is always used instead of **gb**, thus making PSO behaving like the version presented in previous works [18], [15]. We run one more set of experiments with no data exchange among the peers (thus making **ogb** of no use), to compare the performance.

As Figure 1 shows, our design allows to implement a flexible PSO algorithm, whose performance varies according to the strategy implemented by the given policy. The results we show for the Rosenbrock function (Figure 1(a)) seem to confirm the idea that the faster the gossiped information is spread and used among the solvers, the better will be the final result. It can be clearly seen how the outcomes get increasingly better while we choose to use **ogb** more and more often, with respect to **gb**. The same trend is visible in the outcomes or the Zakharov function (Figure 1(b)), where the distributed PSO algorithm is known to perform really well. The logarithmic scale of the vertical axis emphasizes the effect of the full exploitation of **ogb**, but we can see that by applying our policy we are anyway able to improve the baseline by some orders of magnitude. We notice that both the Zakharov and the Rosenbrock functions are unimodal.

The case for the Rastrigin function (Figure 1(c)) and for the Griewank function (Figure 1(d)) is quite different. It turns out that, by slowing down the rate at which PSO prefer **ogb** over **gb** throughout the computation, we can actually avoid an early convergence to suboptimal basins. The results with the Rastrigin function are particularly significant, because PSO is known to have severe troubles in getting out from its local minima. Thus, by implementing a simple policy about the usage of the available global knowledge, we are able to obtain improvements that would otherwise cost a long time spent in tuning the basic PSO parameters towards a "good" configuration. We notice that both the Rastrigin and the Griewank functions are multimodal.

As a general remark, our design seems to improve PSO's ability of escaping from local minima, whereas it slows down the pursuit of the global optimum when PSO is searching smoother domains. Thus, we may conclude that using the proposed policy is useful while optimizing functions that are known (or expected) to present several local (suboptimal) attractors.

Furthermore, we notice that the policy implemented in our experiment is very simple and static. This can be the main reason why, being anyway capable to improve the solution quality for multimodal functions, it is not effective enough to prevent PSO to be eventually trapped to suboptimal basins, as the long horizontal lines of Figures 1(c) and 1(d) clearly show. The same reason could be behind the analogous phenomenon of excessively slow improvement showed by Figures 1(a) and 1(b). A policy that dynamically adapts to the current state of the computation, as new information becomes available to each peer via usual decentralized mechanisms, can lead to better results and is currently being investigated.

 ISBN: 978-1-61208-318-6

The limited set of results obtained so far is not enough to assess a generalized pattern that holds in different scenarios. Nonetheless, we believe it may point out a novel profitable research direction, which have a large potential to be deepened and extended. Particularly interesting may be the analysis of the behavior of the algorithm with respect to different values of N, or with respect to different local swarms sizes. Both of these parameters can have a significant impact on the performance of the algorithm, which we intend to examine in our future work.

VI. Conclusion

The contribution of this paper belongs to the emerging domain of P2P-distributed function optimization. It is particularly important in such a domain, besides the tuning of the optimization algorithm itself, the way useful information is shared among the participants and how each of them chooses to use it.

We presented the design of a P2P gossip-based PSO algorithm that is capable to implement different policies with respect to the use of gossiped information about the overall best point known at any time in the network. Preliminary experimental results show how the performance of such an algorithm can vary, depending on how quickly each solver makes use of the information sent by other peers. The outcomes show that the quality of the optimization can benefit from adopting flexible strategies like the one we propose. This may lead to a better exploration of the function domain and help avoiding early suboptimal convergence.

Our results, as those of previous works [5], [16] on P2P decentralized function optimization, show that exploiting large scale, loosely coupled and possibly heterogeneous distributed systems to obtain good quality results is a viable approach. Particular care must be dedicated not only to modify the algorithms to fit the specific distributed environment, but also to model the diffusion of information among the participants in the most effective way. Among the other possible research directions, we think that is of utmost interest the study of how different overlay communication topologies and different computational capacities of the peers may affect the overall quality of the results.

Acknowledgements

Experiments presented in this paper were carried out using the Grid5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see https://www.grid5000.fr).

This work was partially funded by Bretagne regional project SOFTLIVE.

References

[1] E. G. Talbi, *Parallel Combinatorial Optimization*. John Wiley and Sons, USA, 2006.

[2] J. Laredo, P. Castillo, A. Mora, and J. Merelo, "Exploring population structures for locally concurrent and massively parallel evolutionary algorithms," in *Evolutionary Computation, 2008. CEC 2008.*, June 2008, pp. 2605–2612.

[3] N. Melab, M. Mezmaz, and E.-G. Talbi, "Parallel hybrid multi-objective island model in peer-to-peer environment," in *Proceedings of IPDPS'05*. Washington, DC, USA: IEEE Computer Society, 2005, p. 190.2.

[4] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," *ACM Transactions on Computer Systems*, vol. 23, no. 3, pp. 219–252, August 2005. [Online]. Available: cikkek/tocs04.pdf

[5] J. Laredo, P. Castillo, A. Mora, J. Merelo, and C. Fernandes, "Resilience to churn of a peer-to-peer evolutionary algorithm," International Journal of High Performance Systems Architecture, 2008, volume 1, Number 4.

[6] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," *IEEE Int. Conf. Neural Networks*, pp. 1942–1948, 1995.

[7] R. Poli, "Analysis of the publications on the applications of particle swarm optimisation," Journal of Artificial Evolution and Applications, November 2007.

[8] U. K. Wickramasinghe and X. Li, "Using a distance metric to guide pso algorithms for many-objective optimization," in *Proceedings of the 11th Genetic and Evolutionary Computation Conference (GECCO'09)*, Montreal, Québec, Canada, Jul. 2009, CONFERENCE, pp. 1339–1346.

[9] T. Desell, M. Magdon-Ismail, B. Szymanski, C. Varela, H. Newberg, and D. Anderson, "Validating evolutionary algorithms on volunteer computing grids," in *Distributed Applications and Interoperable Systems*, ser. Lecture Notes in Computer Science, F. Eliassen and R. Kapitza, Eds. Springer Berlin / Heidelberg, 2010, vol. 6115, pp. 29–41. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-13645-0_3

[10] M. R. Khouadjia, L. Jourdan, and E.-G. Talbi, "Adaptive Particle Swarm for Solving the Dynamic Vehicle Routing Problem," in *In Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications (AICCSA)*, Tunisie, Sep. 2010, p. 10.1109/AICCSA.2010.5587049. [Online]. Available: http://hal.archives-ouvertes.fr/hal-00525446/en/

[11] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proc. 4^{th} Congress on Evolutionary Computation (CEC'02)*, May 2002, pp. 1671–1676.

[12] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, Jun. 2004.

[13] I. Scriven, A. Lewis, D. Ireland, , and J. Lu, "Distributed multiple objective particle swarm optimisation using peer to peer networks," in *IEEE Congress on Evolutionary Computation (CEC)*, 2008.

[14] I. Scriven, A. Lewis, and S. Mostaghim, "Dynamic search initialisation strategies for multi-objective optimisation in peer-to-peer networks," *IEEE Congress on Evolutionary Computation, CEC '09*, pp. 1515 – 1522, 2009.

[15] M. Biazzini, B. Bánhelyi, A. Montresor, and M. Jelasity, "Peer-to-peer optimization in large unreliable networks with branch-and-bound and particle swarms," in *Applications of Evolutionary Computing*, Mario Giacobini *et alii*, Ed. Springer, 2009, pp. 87–92.

[16] M. Biazzini and A. Montresor, "Gossiping differential evolution: a decentralized heuristic for function optimization in p2p networks," in *Proceedings of the 16th International Conference on Parallel and Distributed Systems (ICPADS'10)*, Dec. 2010.

[17] J. L. J. Laredo, E. A. Eiben, M. Schoenauer, P. A. Castillo, A. M. Mora, F. Fernandez, and J. J. Merelo, "Self-adaptive gossip policies for distributed population-based algorithms," 2007. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0703117

[18] M. Biazzini, A. Montresor, and M. Brunato, "Towards a decentralized architecture for optimization," in *Proc. of IPDPS'08*, Miami, FL, USA, April 2008.

[19] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen, "Gossip-based peer sampling," *ACM Transactions on Computer Systems*, vol. 25, no. 3, p. 8, August 2007.

[20] M. Biazzini and A. Montresor, "P2poem: Function optimization in p2p networks," *Peer-to-Peer Networking and Applications*, pp. 1–20, 10.1007/s12083-012-0152-8. [Online]. Available: http://dx.doi.org/10.1007/s12083-012-0152-8

[21] https://www.grid5000.fr.

 ISBN: 978-1-61208-318-6