Gossiping Differential Evolution: a decentralized heuristic for function optimization in P2P networks

Marco Biazzini
University of Trento, Italy
biazzini@disi.unitn.it

Alberto Montresor
University of Trento, Italy
alberto.montresor@unitn.it

***Abstract*—P2P-based optimization has recently gained interest among distributed function optimization scientists. Several well-known optimization heuristics have been recently re-designed to exploit the peculiarity of such a distributed environment. The final goal is to perform high quality function optimization by means of inexpensive, fully decentralized machines, which may either be purposely organized in a P2P network, or voluntarily join a running P2P optimization task. In this paper we present the *GoDE* algorithm (Gossip-based Differential Evolution), which obtains remarkable results on several test functions. We describe in detail the algorithm design and the epidemic mechanism that greatly improves the performance. Experimental results in a simulated environment show how *GoDE* adapts to network scale and how the epidemic communication protocol can make the algorithm achieve good results even in presence of a high churn rate.**

***Keywords*- peer-to-peer, function optimization, differential evolution, heuristic, churn**

I. Problem and Goal

When confronted with a function optimization problem, scientists are presented with a large choice space of algorithms and techniques. Recently, a new dimension has been added to this space: distribution. Optimization tasks could be distributed among a collection of independent machines, with the obvious goal of obtaining a speed-up.

Most of the existing research in distributed optimization, however, has been focused on tightly-coupled architectures, such as parallel systems [14] or high-performance clusters, usually managed by a central coordinator [15]. These systems either have strict synchronization requirements or rely completely on a central server, which coordinates the work of clients and acts as a state repository.

Quite recently, the possibility to perform optimization tasks in a P2P decentralized network of solvers has been investigated and explored. Our target networking environment consists of independent nodes that are connected via an error-free message passing service. All nodes have an identical role running an identical algorithm. Joining and failing nodes are tolerated automatically via the inherent (or explicit) redundancy of the algorithm design. While such a distributed environment sounds like "no big issue" concerning nodes cooperation, communication and connectivity, it is not trivial at all from the point of view of executing a distributed function optimization task.

To simplify the discussion, we briefly define some basic terminology. The system we target is composed by a (potentially large) number of *solvers*, each of them being a running instance of an *optimization algorithm*. Each solver iteratively evaluates one or more points of an *objective function*, with the goal of finding a better objective value (minimum or maximum). Solvers may share information with each other about the progress of the optimization process (solution quality). Subsequent iterations of the local instance of the algorithm may depend not only on the objective values that have been found locally, but also on the information coming from other solvers in the network.

Performing function optimization tasks in a P2P distributed environment requires the user to pay attention to some novel and interesting design issues:

- *Synchronization*. Depending on the kind of algorithms and on the way their computations that are distributed among the solvers, there may be performance decays due to synchronization needs. Not all the algorithms have strict synchronization requirements; for those which have them, it is likewise fundamental to know *when* the synchronization is strictly needful and when it is not. Thus, it is important to achieve a good understanding of the relation between performance drifts and ratio of communication events, in order to find, for each kind of algorithm, the maximal amount of synchronization occurrences that does not penalize performance.
- *Information sharing*. How much information has to be shared among the distributed instances of an algorithm? Of course, it depends on what algorithm is running, but also on how many of its instances are active in the network at the same time. A careful consideration of *what* has to be shared and what can be kept as local is crucial for each kind of algorithm.
- *Convergence and communication rate*. *How often* shared information must be updated? The spreading of the information relies on both the frequency of message exchanges and the interconnection topology. The behavior of the same algorithms in different kinds of P2P topologies may present significant differences. A tradeoff between a good information exchange and a rapid advancement of each individual search process has to be found. Moreover, particular attention has to be paid to the effect of delays

when propagating updates.

Each of these aspects demands careful analysis and understanding. The present contribution shows how a specific distributed meta-heuristic (Differential Evolution) can be designed in a P2P fashion and how spreading relevant information in an epidemic fashion helps improving the solution quality, in spite of overlay dynamism and churn.

The rest of the paper is organized as follows. Some background and the description of our novel algorithm are given in Section II. Experimental results based on extensive simulations are discussed in Section III. Section IV summarizes recent related works, while in Section V we present some final remarks.

II. Gossiping Differential Evolution

This section provides a brief description about the original DE meta-heuristic and describes in details our novel *GoDE* algorithm, a distributed flavor of DE augmented by means of an epidemic diffusion mechanism among cooperating solvers.

A. Differential Evolution overview

DE is a well-known and broadly studied method for the optimization of multidimensional functions that particularly suits multimodal (i.e. having more than one minimum) functions, introduced by Storn and Price in 1997 [13].

Basically, DE is a scheme for generating trial parameter vectors, commonly denoted as *individuals*. We can see an individual as a vector of coordinates in the objective function domain, thus denoting a point in which the function may be evaluated. A group of individuals is called a *population*.

DE generates new individuals by linearly combining two or more individuals already included in the current population, through the execution of a specific *operator*. The set of available operators defines different DE variants; an example is given in Table I, which summarizes the most common instances.

In the table, T is the new individual generated by the combination of other individuals. A, B, C, D, E are individuals chosen at random within the current population, while $Best$ is the individual representing the current optimal solution. $Fact \in [0 \dots 2]$ is a user-defined real constant factor that controls the amplification of the differential variation.

Operators are usually classified according to the notation $DE/x/y/z$ proposed in [13], where:

- x specifies the vector to be mutated; it can be either RAND, meaning a random selection from the whole population, or BEST, meaning that the best known solution $Best$ is used.
- y is the number of difference vectors used (1 or 2).
- z denotes the way the probability of performing the linear combinations is drawn; it can assume either the value BIN, that corresponds to independent binomial experiments for each dimension; or the value EXP, that corresponds to a conditional probability for each dimension w.r.t. the precedent one.

Notation	Linear combination
DE/rand/1/*	$T = E + Fact \cdot (A - B)$
DE/rand/2/*	$T = E + Fact \cdot (A + B - C - D)$
DE/best/1/*	$T = Best + Fact \cdot (A - B)$
DE/best/2/*	$T = Best + Fact \cdot (A + B - C - D)$

TABLE I
DE operators

Until a termination criterion is met (e.g. number of iterations performed, or adequate fitness reached), the DE algorithm generates trial individuals by repeatedly applying one or more operators to a given population and substitutes population members if their quality is greater. No global probability distribution is required to generate trial individuals while moving toward the optimum (being it a maximum or a minimum), thus the algorithm is completely local and self-organizing. It is easy to see that, provided a way to properly distribute the overall population, DE is a suitable candidate to distributed optimization tasks.

It is well known that distributed versions of evolutionary algorithms can outperform sequential ones (compared under the same computational effort, measured in number of function evaluations). This motivates the growing amount of research about the ways the local populations can be made interact with each other in order to improve the final result.

An important requirement for distributed DE algorithm is the preservation (if not the enhancement) of diversity, to avoid a premature convergence of the local populations to sub-optimal solutions [4]. Our novel *GoDE* algorithm implements a simple yet very effective migration policy that proves to boost the performance of the heuristic while adapting well to the number of solvers cooperating in the P2P network.

B. GoDE

In the decentralized version of DE proposed in this paper, each solver is associated with an *independent* population of individuals, potentially exploring a distinct portion of the search space.

Solvers exchange individuals between different populations by means of an epidemic protocol. In this way, each population is modified in two ways: by the *local* computation performed by the solver, and by *foreign* individuals obtained from other solvers.

This approach makes the various *GoDE* instances capable of exploiting a far richer population, while operating on the usual amount of local individuals. Moreover, poorly performing individuals are *gradually* substituted by better newcomers, so that a local population is not simply "killed" and overtaken by a bald new one. This preserves the precious local diversity of the global population and the fair partitioning of the function domain space among the solvers.

GoDE assumes the existence of a peer sampling service that returns a uniform random sample of the entire network population. In our implementation, this service is provided by NEWSCAST, that maintains a random topology over the collection of solvers.

Apart from the *local* best known individual, denoted as *lBest*, each *GoDE* solver maintains information about the *global* best-known individual, denoted *gBest*.

GoDE solvers spread information about their current population in an epidemic fashion, following a *push-pull* policy. The heuristic propagates two pieces of information:

- the current global best individual;
- the sub-optimal individual that has been more recently introduced in their local population.

The rate of epidemic spreading is correlated to the length of a function evaluation; we assume that an epidemic exchange for the global best individual is initiated after each function evaluation, while an epidemic spread of the sub-optimal individual is initiated with probability G after every function evaluations.

GoDE iterates the following actions until the termination criterion is met:

1) Process any new message from other nodes in the network, updating the internal state as needed;
2) Perform the next optimization step, thus evaluating the function and updating the internal state;
3) According to the defined communication strategy, spread relevant information to other solvers.

The time to propagate a new current best solution to every node this way takes $O(\log N)$ periods in expectation where N is the network size [10]. In principle, the time to propagate the selected population individual would asymptotically be the same, but it is quite unlikely that the very same individual can be spread to more than one node without being modified by the DE operators. We believe that this is the very reason why our mechanism turns out to be so effective in achieving good results: the global population gradually improves, because the single individuals, one at a time in each local population, gently improve, thus focusing the evaluated search domain toward more promising areas.

III. Experimental Results

The results we present are obtained by implementing *GoDE* on PeerSim [6]. Table II recalls some characteristics of the test functions we challenge. Sphere10 is an easy unimodal function. Rosenbrock10 and Zakharov10 are non-trivial unimodal functions. The rest of the functions are multimodal. Griewank10 is similar to Sphere10 with high frequency sinusoidal "bumps" superimposed on it. Schaffer10 is a sphere-symmetric function where the global minimum is surrounded by deceptive spheres. Rastrigin10 is highly multimodal and its local minima are regularly distributed. It is a difficult test case for most optimization methods.

The set $\mathcal{O}$ of DE operators we use is shown in Table III. We adopt the standard notation presented in Table I, keeping the proper semantics. We recall that, while operators with *gBest* use the global best solution in the network (as learned through gossip), the *lBest* variants ignore the global best solution and just consider gossiped newcomers among their population individuals.

We assume that every node maintains a local population of 8 individuals. In all our experiments we vary the following parameters:

- **network size** (N) the number of nodes in the network;
- **individual gossip rate** (G) the probability to send, after every function evaluation, the sub-optimal individual in the local population that has been updated more recently.

A total number of 2^{20} function evaluations are performed during each experiment, equally divided among the N solvers.

We run 10 independent experiments for all parameter combinations using

$$N \in \{2^1, \ldots, 2^{14}\} \text{ and } G \in \{0, 0.016, 0.125, 0.25, 0.5, 1\}$$

combined with all the operators in Table III, on all test functions.

For every experiment, we trace the best solution known in the overall network every time 8 new iterations of the solver have been performed in each node. Thus we collect the current best result at the end of each DE generation cycle. The sets of experiments described so far are run separately in different network conditions. We considered the following scenarios:

- No churn.
- Churn drawn by replacing 5% of nodes during a time interval taken by 20 function evaluations.
- Churn drawn by replacing 10% of nodes during the same time interval.

Our metric of interest is based on the number of function evaluations performed at each node. Objective functions can differ in complexity by several orders of magnitude. Thus, generally we cannot fix, in absolute terms, the amount of time these churn rates represent. Considering that a non-trivial function may take 1 second per evaluation, the simulated scenarios are extremely challenging, because 5% and 10% of the nodes may fail and be substituted every 20 seconds.

Function $f(x)$	D	$f(x^*)$	K
Sphere10	$[-5.12, 5.12]^{10}$	0	1
Rosenbrock10	$[-100, 100]^{10}$	0	1
Zakharov10	$[-5, 10]^{10}$	0	1
Griewank10	$[-600, 600]^{10}$	0	$\approx 10^6$
Schaffer10	$[-100, 100]^{10}$	0	≈ 63 spheres
Rastrigin10	$[-5.12, 5.12]^{10}$	0	$\approx 10^6$

TABLE II
Test functions. D: search space; $f(x^*)$: global minimum value; K: number of local minima.

Notation	**Linear combination**
DE/gBest/1/exp	$T = gBest + Fact \cdot (A - B)$
DE/lBest/1/exp	$T = lBest + Fact \cdot (A - B)$
DE/rand/1/exp	$T = E + Fact \cdot (A - B)$
DE/rand/2/exp	$T = E + Fact \cdot (A + B - C - D)$
DE/randToGBest/2/exp	$T = E + Fact \cdot (gBest - A + C - D)$
DE/randToLBest/2/exp	$T = E + Fact \cdot (lBest - A + C - D)$

TABLE III
The set $\mathcal{O}$ of operators used by *GoDE*

In the rest of this section, we show a collection of plots representing a selection of our results. Our comments and remarks derive from the analysis of all sets, of course, but being the overall number of experiment sets more than 3000, we collect here just some representative examples. On the vertical axis, we always plot the quality of the solution achieved by the algorithm, that is the difference between the best value found so far in the entire network and the the real optimum (the minimum value of the function we tackle). All the values shown in the plots are the averages of the best values found over ten experiments.

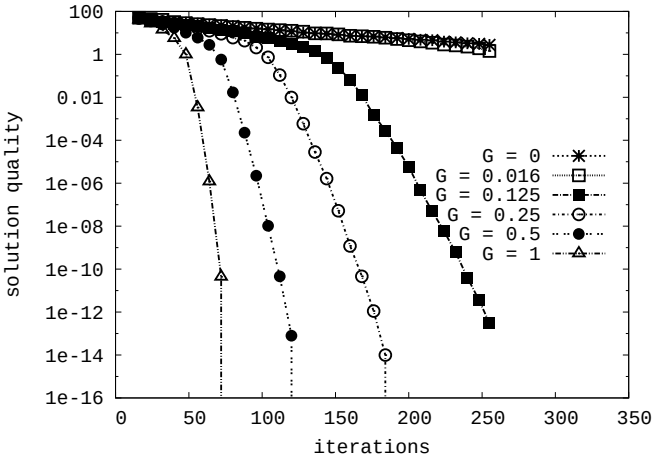


Fig. 1. Rastrigin - DE/gBest/1/exp. $N = 2^{12}$. How the gossip rate G improves performance.

The first evidence we find is that the individual spreading mechanism is definitely effective. All the operators remarkably improve their performance when the gossip rate G is set to a non-zero value. There is an interesting dependence from the network size, though, which is worth investigating in some details. It is not true that setting $G > 0$ is always beneficial to the computation, but it can have a significant impact if done in the appropriate way with respect to the network size, leading to a huge performance boost.

Generally we see that lower gossip rates improve the performance in small and medium networks, while higher gossip rates have a surprisingly good impact in large networks. Of course there are no "magic numbers" that can suit all the network sizes for all the operators, but this kind of behavior appears in all our experiments, with a surprising consistency. Figure 2 illustrates how an appropriate choice of G leads to very good results even when using a very large number of nodes. This fact is not usual at all and shows that exploiting large decentralized P2P networks to perform function optimization tasks can be not only efficient, but most of all effective. A large number of function evaluations can be partitioned among thousands of distributed solvers, resulting in a lighter workload per machine, while ending up in an equally good, if not better, result.

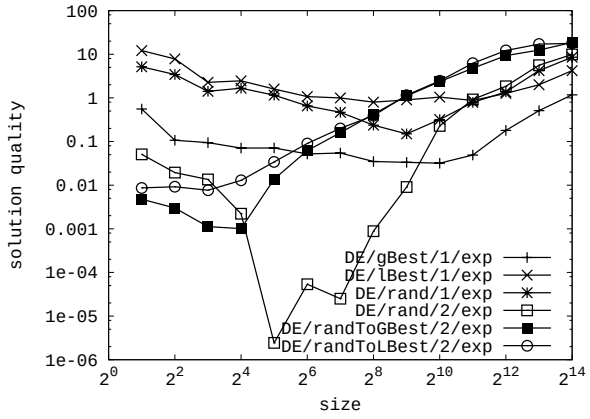
Not only our gradual shuffling of population individuals improves the *final* results, but it also speeds up the improvement *during* the computation, with respect to the case in which no population gossiping is provided. As we examine the experiments in which an operator is able to find the optimum of a function even if $G = 0$, we see that most of the time, for $N > 2^4$, a small gossiping rate helps finding the optimum within a smaller number of function evaluations. This effect is even more evident in those cases when the operator is not performing well by itself, as Figure 1 clearly illustrates.

Figure 3 shows how performance changes in presence of churn. As we can see, population gossiping helps even in case of high churn. Paying attention to these graphs, we see something unexpected: churn can even be helpful for large networks, given that an adequate gossip rate is enabled! We find — not only in this specific case: as we said, these considerations can be generally drawn out of the whole set of experiments we performed — that when churn is expected, a higher gossip rate is preferable. As a matter of fact, making individuals circulate the network after every function evaluation can successfully cope with the higher churn rate, provided we have networks that are large enough to guarantee a proper amount of individuals. In some cases, churn even helps improving the final outcome, as Figure 4 clearly shows.

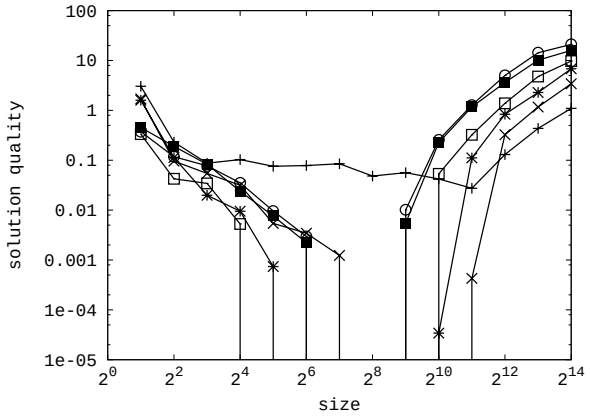
Our result give us some insights about how the behavior of a given operator changes with respect to the two variable parameters. If the gossip rate is too high with respect to the number of nodes in the network, it has the undesired effect to make the distributed algorithm converge too quickly to a sub-optimal value. Anyway, if churn is really high — thus causing a sort of frequent local restart of the algorithm in the affected nodes — a high gossip rate helps the distributed algorithm keeping the explorative drift under control, achieving nearly optimal performance even in such an adverse situation.

In a recent work about a P2P flavor of Particle Swarm Optimization (PSO) [2], we showed that in some test cases churn can be beneficial for the P2P-PSO heuristic. We think this similar behavior is due to common reasons. When churn occurs during a P2P-PSO optimization, the particles actually restart in random positions. Churn during *GoDE* optimization makes local populations disappear and be substituted by new individuals. These two scenarios present important similarities and, in both cases, what actually happens is that the exploration of the search space is boosted. Of course, the performance benefit is not guaranteed. It may greatly improve the solution quality if the particles/individuals are suffering for lack of possibilities to escape from local minima. But it just spoils the chance to improve, if the solvers are on a good "track" within a promising attraction basin.

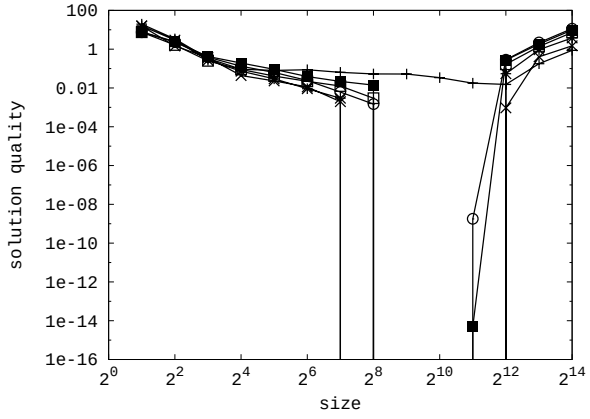
According to our experience, *GoDE* shows a higher resilience than P2P-PSO with respect to this issue. It seems that the way the individuals are combined by DE operators is more capable to provide recovering from "bad choices" or unfortunate events. Any new generation of DE individuals is not only a bunch of better candidate solutions; it is a better domain region to search within.



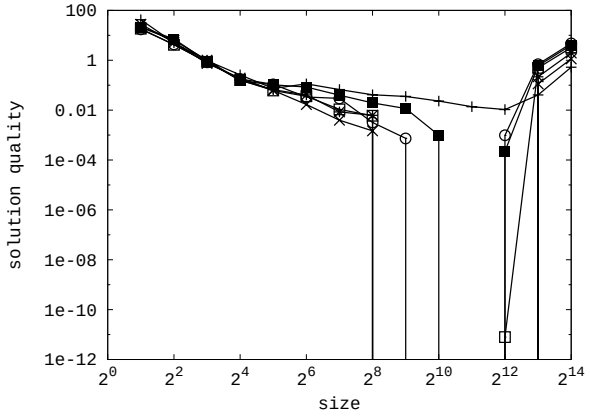
(a) $G = 0$



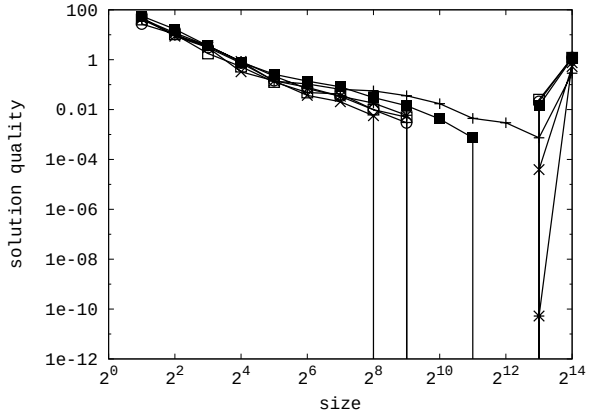
(b) $G = 0.016$



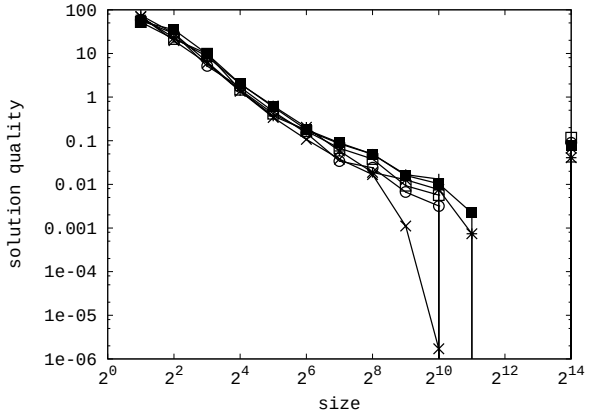
(c) $G = 0.125$



(d) $G = 0.25$

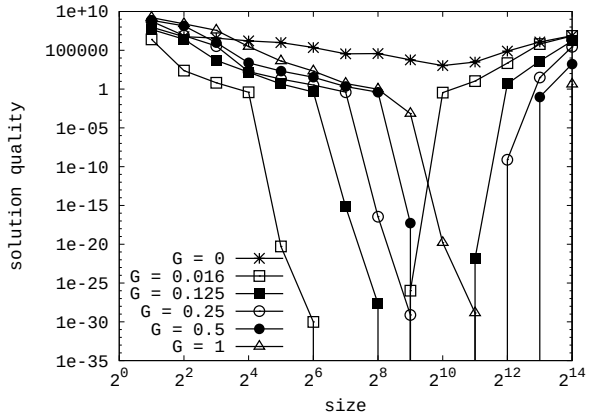


(e) $G = 0.5$

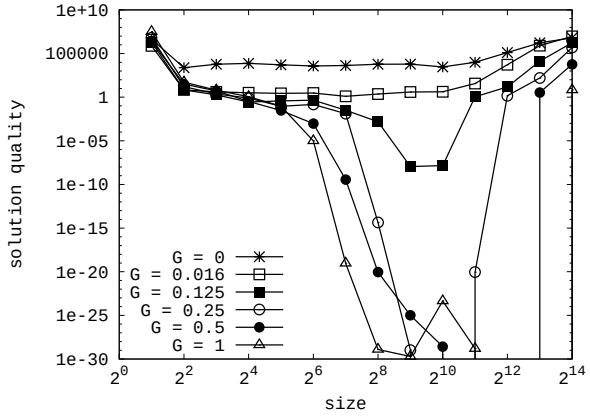


(f) $G = 1$

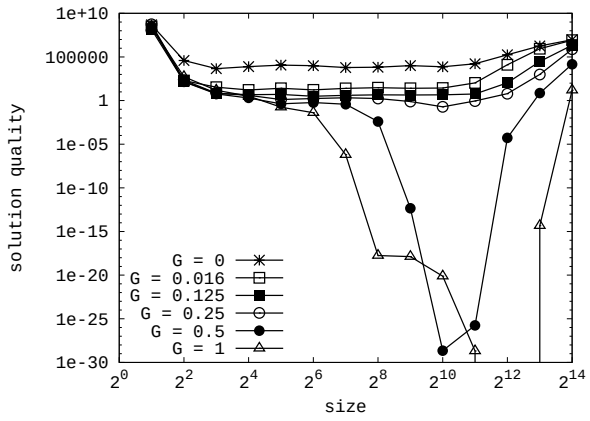
Fig. 2. Griewank - How the gossip rate G improves performance in different networks. No churn.



(a) DE/Rand/1/Exp – No churn



(b) DE/Rand/1/Exp – 5% churn



(c) DE/Rand/1/Exp – 10% churn

Fig. 3. Rosenbrock - How the gossip rate G improves performance in different networks.

This happens because individuals with low fitness can still help by providing diversification as needed, whereas PSO particles which are performing poorly are quite useless to the swarm. We think this fact makes *GoDE* more able to cope with different problems in those cases when a specific tuning is not possible. We are not talking about state-of-the-art solutions, but about good outcomes given a limited (or null) amount of information about the problem. It is understood that a "good parametrization" can change this situation quite radically.

Trying to come up with general guidelines to successfully set up our DE operators for a P2P decentralized optimization task, we may observe what follows:

- For small networks (up to 2^6 nodes), a very small gossip rate is preferable, to avoid premature sub-optimal convergence of the local populations.
- For large networks (more than 2^{10} nodes) gossip should better occur with a probability of 0.5 or higher.
- Churn change the situation in a way that is hard to predict with respect to the relation between performance and network size. Anyway, almost in any case a high gossip rate produce better results.
- Operators biased toward exploration generally capitalize more the gossiping of sub-optimal individuals.

It is hard to state a general conclusion, because the behavior of the different operators and their absolute performance always differ as problems change. Anyway, the patterns we describe appear in almost any case and we think they show a clear correlation between performance, gossip rate and network size. More precisely, the gossiping mechanism makes the set of local populations at each node behave like a unique large global one. Anyway the effect is not simply analogous to the one we could see if we had a single node hosting all the individuals that are actually spread on our network. In that case, the generation cycle length would be so huge, that the convergence of the individuals to a good outcome would require an unacceptable amount of time.

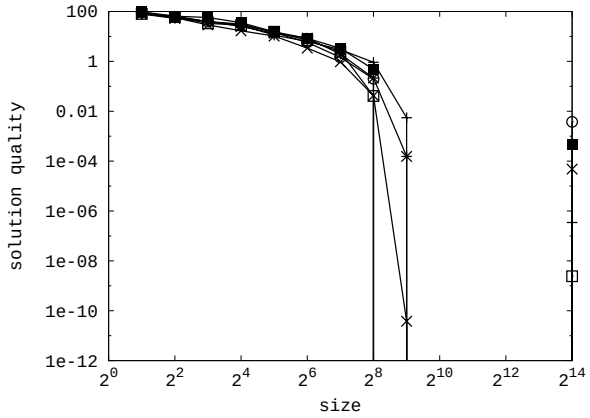
The parallel islands are meant to cope with this problem, although this paradigm may suffer from premature convergence to local sub-optima, that pins down the computation to a poor performance. Among the various known migration policies that have been devised to solve this issue, we believe that our variable gossiping diffusion of sub-optimal individuals proves to be both effective and robust.

IV. Related work about P2P optimization

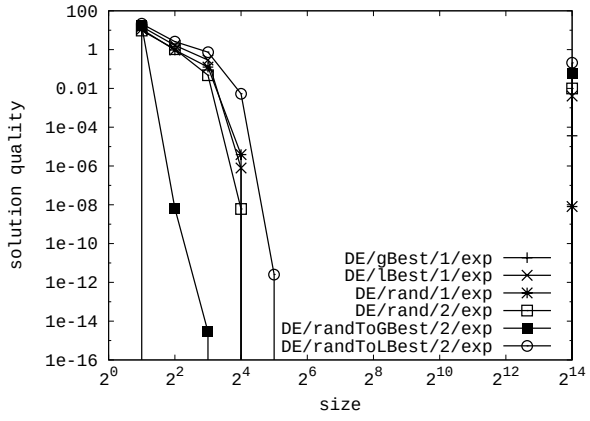
P2P heuristic optimization is quite a newborn branch of distributed optimization. As it happens in all the happy beginnings, researchers are extensively exploring the various issues this new field entails. We give here a brief overview of the recent publications, mostly related to P2P implementations of population-based algorithms.

In [9] the authors presented some preliminary evaluations of a parallel hybrid MO-EA (multi-objective evolutionary algorithm) deployed in a P2P environment. Several results show that is possible to successfully parallelize such an evolutionary algorithm in a P2P fashion, exploiting the resources available in a network of 120 heterogeneous PCs.

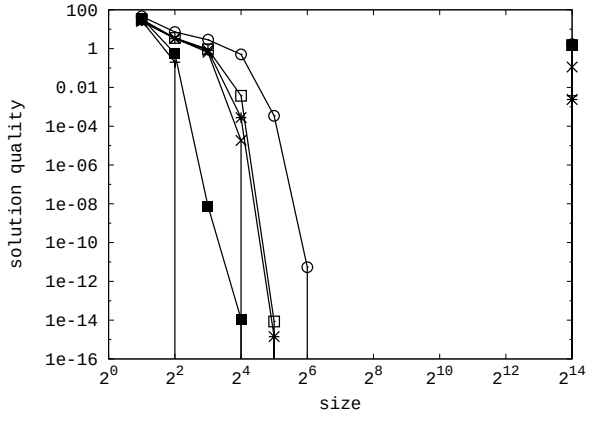
In [5] an extensive experimentation was performed, to test the fault tolerance of the island model on genetic algorithms (GA), when executing them on a distributed system. The results show that this model can be trusted when running



(a) $G = 1$ – No churn



(b) $G = 1$ – 5% churn



(c) $G = 1$ – 10% churn

Fig. 4. Rastrigin - Churn may increase population diversity and improve performance.

experiments on a non-reliable parallel or distributed infrastructure, the quality of the outcomes being at most 2% worse (on average) than when a reliable infrastructure is employed, without adding any special techniques required for dealing with faults.

Battiti et al. [3] presented an implementation of Particle Swarm Optimization (PSO) in a distributed peer-to-peer environments. Global information sharing among processes is managed by epidemic protocols, that ensure spreading of relevant data generated during the search. The results show that the message complexity needed to outperform a single-node sequential algorithm can be low and that the proposed approach is therefore viable.

Almost the same PSO distributed algorithm has been proposed by different authors in [11] to address multi-objective optimization problems, in which the main goal is to find a set of values lying on the Pareto-frontier that can be found for the objectives being tackled. In spite of the different experimental environment (simulated very large networks in the former case, small real network in the latter), the results confirm the usefulness of the approach.

Recently, the same authors [12] proposed new models to substitute failing particles of the same kind of multi-objective PSO algorithm, in such a way that the capability of the affected swarm to explore the search space can be enhanced. The initialization of the new particles is performed using a combination of binary search to fill the gaps in the space between the two known Pareto-front extremes and edge extension, to improve the exploration beyond the known Pareto-front.

Van Steen et al. [16] presented a fully decentralized evolutionary algorithm in which the population size is kept stable by locally adapting the surviving rate of the individuals according to global population estimations, performed by means of gossiping protocols. The parent and survivor selection can be done completely autonomously and asynchronously, without central control, yet avoiding the risk of population explosion or implosion.

Laredo et al. [7] proposed the Gossiping-based Evolvable Agent model, where every individual of an evolutionary algorithm's population self-schedules its own action as an agent (evolvable agent) and dynamically self-organizes its neighborhood via NEWSCAST (gossip-based). Tests run in a really distributed deployment with multi-threaded configurations confirm that P2P evolutionary algorithms are competitive with respect to not-distributed ones.

Recently, in [8] the authors presented an extensive evaluation of a generic P2P evolutionary algorithm with respect to different (simulated) network size and churning conditions, while tackling a known hard test function. Results show that the gossiping enabled small-world structure of the network makes the algorithm very robust even when very high churn rates are applied.

Finally, we proposed a distributed DE algorithm based on epidemic communication in [1]. The main purpose was to compare different hyper-heuristics using DE operators on several test functions. While that work paid great attention to periodically spreading the best known results, no interaction was provided among distinct populations on different nodes. That is precisely the issue we address with *GoDE*.

Our *GoDE* meta-heuristic proves to be able to cope with churn while scaling well up to thousands of nodes. Without reaching the extreme design of the Evolvable Agents (which would be, if every node hosted a single individual, instead of a small population), it is able to overcome the known drawbacks

of an island-based design (local population size maintenance, premature sub-optimal convergence, poor adaptivity to different problems) by wisely using an epidemic spread of individuals among local populations, that both improves the final performance of the algorithm and speed up the gradual convergence to the global optimum.

V. Conclusion

P2P distributed function optimization is a recently born research field that is attracting the interest of both the scientific community and the industries. We introduced some relevant aspects that an effective and scalable distributed optimization algorithm design must take into account.

We proposed a novel gossip-based Differential Evolution algorithm, that implements the traditional DE heuristic in a P2P fashion. *GoDE* exploits the epidemic paradigm to share relevant information in a P2P network of cooperating solvers, coping with the lack of a central coordinator and the possibility of high churn.

We showed how, by choosing the appropriate gossip rate to spread local individuals — besides the current best results — during the computation, *GoDE* is able to achieve optimal results even in presence of very high churn rates. The key point of our design is that not only the best individuals, but also a proper management of the global population are crucial aspects of a distributed DE heuristic. This issue is not trivial, given that the population is spread among several distinct nodes belonging to a P2P overlay network.

We showed that our gossip-based migration policy is able to cope with adverse network conditions. Moreover, it can somehow turn the adversity in a benefit, transforming the destabilizing effect of churn in a beneficial restart mechanism, that may even speed up the final convergence to the global optimum.

By an extensive experimentation in a simulated environment, we proved that *GoDE* generalizes well with respect to various DE operators and different test functions, thus showing to be able to deal with different problems, successfully exploiting a number of resources that can vary between less than 10 to tens of thousands.

Our future work will be testing the performance of our algorithm in a real deployment and moving to the design of distributed meta-heuristics that combines different optimization algorithms in a P2P fashion.

VI. Acknowledgements

This work is supported by the Autonomous Security project, financed by MIUR Programme PRIN 2008.

References

[1] Marco Biazzini, Balázs Bánhelyi, Alberto Montresor, and Márk Jelasity, *Distributed hyper-heuristics for real parameter optimization*, Proceedings of the 11th Genetic and Evolutionary Computation Conference (GECCO'09) (Montreal, Québec, Canada), July 2009, pp. 1339–1346.

[2] Marco Biazzini, Balázs Bánhelyi, Alberto Montresor, and Márk Jelasity, *Peer-to-peer optimization in large unreliable networks with branch-and-bound and particle swarms*, Applications of Evolutionary Computing (Mario Giacobini, Anthony Brabazon, Stefano Cagnoni, Gianni A. Di Caro, Anikó Ekárt, Anna Isabel Esparcia-Alcázar, Muddassar Farooq, Andreas Fink, and Penousal Machado, eds.), Springer, 2009, pp. 87–92.

[3] Mauro Brunato, Roberto Battiti, and Alberto Montresor, *Gosh! gossiping optimization search heuristics*, Proceedings of the Learning and Intelligent Optimization Workshop (LION 2007), 2007.

[4] E. Cantú-Paz, *Migration policies, selection pressure and parallel evolutionary algorithms*, Journal of Heuristics, 7(4), 2001.

[5] J. Ignacio Hidalgo, Juan Lanchares, Francisco Fernández de Vega, and Daniel Lombra na, *Is the island model fault tolerant?*, GECCO '07: Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation (New York, NY, USA), ACM, 2007, pp. 2737–2744.

[6] Márk Jelasity, Alberto Montresor, Gian Paolo Jesi, and Spyros Voulgaris, *The Peersim simulator*, http://peersim.sf.net.

[7] J.L.J. Laredo, P.A. Castillo, A.M. Mora, and J.J. Merelo, *Exploring population structures for locally concurrent and massively parallel evolutionary algorithms*, Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on, June 2008, pp. 2605–2612.

[8] J.L.J. Laredo, P.A. Castillo, A.M. Mora, J.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.

[9] N. Melab, M. Mezmaz, and E-G. Talbi, *Parallel hybrid multi-objective island model in peer-to-peer environment*, IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 6 (Washington, DC, USA), IEEE Computer Society, 2005, p. 190.2.

[10] Boris Pittel, *On spreading a rumor*, SIAM Journal on Applied Mathematics **47** (1987), no. 1, 213–223.

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

[12] 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 (2009), 1515 – 1522.

[13] Rainer Storn and Kenneth Price, *Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces*, Journal of Global Optimization **11** (1997), no. 4, 341–359.

[14] E. G. Talbi, *Parallel combinatorial optimization*, John Wiley and Sons, USA, 2006.

[15] T. Vinkó and D. Izzo, *Learning the best combination of solvers in a distributed global optimization environment*, Proceedings of the Workshop on Advances in Global Optimization: Methods and Applications (AGO 2007) (Mykonos, Greece), 2007, pp. 13–17.

[16] W. R. M. U. K. Wickramasinghe, M. van Steen, and A. E. Eiben, *Peer-to-peer evolutionary algorithms with adaptive autonomous selection*, Proc. of GECCO, ACM Press New York, NY, USA, 2007, pp. 1460–1467.