GOSH! Gossiping Optimization Search Heuristics*

Mauro Brunato, Roberto Battiti, Alberto Montresor
Università di Trento, Dipartimento di Informatica e Telecomunicazioni
via Sommarive 14 — I-38050 Trento — Italy
brunato|battiti|montresor@dit.unitn.it

January 23, 2007

Abstract

While the use of distributed computing in search and optimization problems has a long research history, most efforts have been devoted to parallel implementations with strict synchronization requirements or to distributed architectures where a central server coordinates the work of clients by partitioning the search space or working as a status repository.

In this paper we discuss the distributed implementation of global function optimization through decentralized processing in a peer-to-peer fashion, where relevant information is exchanged among nodes by means of epidemic protocols.

A key issue in such setting is the degradation of the quality of the solution due to the lack of complete information about the global search status. A tradeoff between message complexity and solution quality must be investigated.

Preliminary computational results in a simplified setting, reported in the experimental section, show that research in the field is motivated.

1 Motivation and Scenario

The use of parallel and distributed computing for solving complex optimization tasks is an area which has been investigated extensively in the last decades, see for example [20], for a recent book on the subject, and [1] for an old contribution by our research group.

Most of the previous work assumes the availability of either a dedicated parallel computing facility, or, in the worst case, specialized clusters of networked machines that are coordinated in a centralized fashion (master-slave, coordinator-cohort, etc.). While these approaches simplify management, they normally show severe limitations with respect to scalability and robustness.

*Work supported by the project BIONETS (IST-027748) funded by the FET Program of the European Commission. Accepted at the Learning and Intelligent Optimization Workshop LION2007, Andalo (Italy), February 12-17, 2007.

Recently, the peer-to-peer (P2P) paradigm for distributed computing has demonstrated that networked applications can scale far beyond the limits of traditional distributed systems, without sacrificing efficiency and robustness; applications composed of millions of nodes are not uncommon. The applicative area is not limited to content distribution (file sharing), but covers also scientific purposes related to solving massive computation problems in the absence of a dedicated infrastructure (see for example the BOINC [18], project, which unifies several pre-existent distributed computing applications under the same umbrella).

P2P systems are characterized by an high level of dynamism: nodes join and leave the system continuously, often in an unexpected and "rude" manner. This phenomenon is called *churn*. Together with the large scale, churn constitutes a significant problem for P2P systems; no node may have an up-to-date knowledge of the entire system, and the maintenance of consistent distributed information becomes difficult if not impossible.

Originated in the context of databases [8], epidemic and gossip protocols have proved to be able to deal with the high levels of unpredictability associated with P2P systems. Apart from the original goal of information dissemination (messages are "broadcast" through random exchanges between nodes), they are now used to solve several different problems: membership management [13], aggregation [14, 16], topology management [12], resource sharing [], etc.

In this position paper, we propose to adopt the P2P paradigm, and in particular the epidemic and gossip-based approach, to perform global function optimization in a completely decentralized manner. The goal is to enable the exploitation of unused computational resources (such as personal desktop machines) to parallelize the optimization process, without requiring a fixed and centralized infrastructure as the one associated with Grids.

We focus our attention onto stochastic local search [10] schemes based on memory, where little or no information about the function to be optimized is available at the search. In this context, the knowledge acquired from function evaluations at different input points during the search can be mined to build models so that the future steps of the search process can be optimized. An example is the on line adaptive self-tuning of parameters while solving a specific instance proposed by Reactive Search [3]. Recent development of interest considers the integration of multiple techniques and the feedback obtained by preliminary phases of the execution for a more efficient allocation of the future effort (see for example the a-teams scheme in [7], the portfolios proposals [11, 9], the racing schemes [6, 19], dynamic restart policies [15])

In the P2P scenario depicted above, the crucial issues and tradeoffs to be considered when designing distributed optimization strategies are:

Coordination and interaction One has a choice of possibilities ranging from independent search processes reporting the end results, to fully coordinated "swarms" of searchers exchanging new information after each step of the search process.

Synchronization In a peer-to-peer environment the synchronization must be

very loose to avoid wasting computational cycles while waiting for synchronization events.

Type and amount of exchanged information It ranges from the periodic exchange of current configurations and related function values (see for example PSO [17] and genetic algorithms []) to the exchange of more extensive data about past evaluations, possibly condensed into local heuristic models of the function [5].

Frequency of gossiping, convergence issues We consider a simple basic interaction where a nodes picks a random neighbor, exchanges some information and updates its internal state (memory). The spreading of the information depends both on the gossiping frequency and interconnection topology. Tradeoffs between a more rapid information exchange and a more rapid advancement of each individual search process are of interest.

Effects of delays on "distributed snapshots" Because of communication times, congestion and possible temporary disconnections, the received information originated from a node may not reflect accurately the current state, so that decisions are made in a suboptimal manner.

2 Epidemic Protocols

The research on the application of epidemic/gossip protocols in distributed systems started with the seminal work of Alan Demers [8], who proposed several models for the decentralized dissemination of database updates. The models are as follows:

Anti-Entropy Each node periodically selects a random peer and performs an *information exchange* with it; the exact meaning of *exchange* depends on the specific application. In the case of database updates, exchanges may be characterized as *push* – the originator of the exchange sends its own updates to the peer; *pull* – the originator asks for updates from the peer; *push-pull* – both the operations are performed.

Gossip Whenever a node receive an update, it selects a small number k of peers and sends the update to them; if the update has already been received, the node may also decide to stop the spreading of the update with probability p. Values k and p presents a trade-off between the probability of reaching all the nodes and the communication overhead given by redundant messages.

One of the key requirement for implementing an epidemic protocols is the capability of selecting a random peer among all nodes. This is easy when the set of participants is fixed, small and known *a priori*; it is a problem by itself in case of dynamic, large-scale distributed systems. To solve this issue, the concept of *peer sampling service* has been devised; this service provides each node with a

uniform random sample of the entire population of a P2P networks [13]. Interestingly enough, protocols implementing the peer sampling service are epidemic as well: push-pull exchanges are performed, in which random subsets of nodes are shuffled among the nodes.

Once this issue is solved, a large collection of problems may be solved on top of the peer sampling service. Different problems require specific exchange mechanisms; for example, it is possible to compute the average aggregation over a collection of values stored at peer nodes by simply averaging the values exchanged by a pair of nodes [14].

3 Memory-based search heuristics

Hard optimization problems can often be expressed as objective function minimization tasks, and in the general case the only available feature of the resulting function is its point-wise evaluation.

Under these narrow assumptions (the entire knowledge about the function must be deduced from function evaluations, without help from partial derivatives - like gradients or Hessians — or from specific smoothness conditions like Lipschitz continuity), a global optimization algorithm just performs a sequence of point-wise evaluations, possibly driven by some heuristic approximation of function properties such as local minima distribution. Therefore, an optimization algorithm is inherently sequential: the next evaluation point is determined by the past search history, possibly by the current state alone (Markovian algorithms).

In particular, we are considering two sequential search schemes Particle Swarm Optimization (PSO [17]) and Memory-based Reactive Affine Shaker (M-RASH [5]). Both schemes rely on global information for different purposes, therefore their distributed implementation is not straightforward

3.1 Particle Swarm Optimization

Particle swarm optimization [17] is a population-based stochastic optimization technique, inspired by the social behavior of bird flocking, for finding global optima of functions of continuous variables. Search is performed along a small number n (usually in the tens) of trajectories, the i-th trajectory being represented by a "particle" whose status information includes the current position vector $\boldsymbol{x}_i$, the current speed vector $\boldsymbol{v}_i$, the optimal point $\boldsymbol{p}_i$ along the past trajectory and its value $f(\boldsymbol{p}_i)$.

The only global information maintained by the algorithm, and accessible to all particles, is the global optimum position $\boldsymbol{g}$, chosen among the particle's best positions $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$.

The algorithm proceeds by updating one particle at a time: first, the speed vector is modified by adding random components oriented towards the particle's best position $\boldsymbol{p}_i$ and the global best position $\boldsymbol{g}$. Second, the particle's speed is

added to the position vector $\boldsymbol{x}_i$. The function is computed at the new position and optimal values are updated as needed.

3.2 Memory-based RASH

The RASH optimization heuristic [2, 4] defines a single trajectory (as opposed to the multiple trajectories that characterize PSO) that rapidly converges to a local minimum. The critical parameter in the RASH optimizer is the initial point in the trajectory: trajectory evolution can be seen as a dynamical system where local minima work as trajectory attractors, which approximately partition the function domain into attraction basins (the approximation is caused by the stochastic component of the algorithm).

Every optimization run provides information that can be used to determine which starting point is best suited in the next run. Every run yields information in the form of a starting point and a final objective function value. The collection of information from all previous runs can be used to build a partial representation of the overall function landscape to answer one of the following questions:

- Where should the next run start from, in order to achieve the best expected final value?
- Or else: which starting point has an estimated error bar on the expected value which could possibly lead to a value less than the best encountered so far?

In the first case, the next starting point is the one that, based on prior information, should yield the best improvement. In the second case, we shall likely select a starting point in a region which has not been explored, and whose outcome can provide a better result than expected.

Our preliminary experiments indicate that search efficiency of a RASH optimizer increases with its immersion within a memory-based iterated scheme (M-RASH [5]).

4 GOSH! Gossiping Search Heuristics

The distributed realization of a global optimization algorithm ranges between two extremes:

- **Independent execution of stochastic processes** — Global optimization algorithms are stochastic by nature; in particular, the first evaluation is not driven by prior information, so the earliest stages of the search require some random decision. Different runs of the same algorithm can evolve in a very different way, so that the parallel independent execution of identical algorithms with different random seeds permits to explore the tail of the outcome distribution towards lower values.

- **Complete synchronization and sharing of information** — Some optimization algorithms can be modeled as parallel processes sitting in a multi-processor machine supporting shared data structures. Processes can be coordinated in such a way that every single step of each process (i.e., decision on the next point to evaluate) is performed while taking into account information about all processes. A paradigm of this case is the Particle Swarm algorithm, where the optimum is searched along multiple trajectories whose evolution depends both on personal and global history through a very simple rule. The particle swarm method is usually implemented as a single process operating on an array of evaluation points, but its parallelization is straightforward, provided that the cost of sharing global information does not overcome the advantage of having many function evaluations performed simultaneously.

Between the two extremal cases presented above (no coordination or complete information), a wide spectrum of algorithms can be designed to perform individual searches with some form of loose coordination. Some paradigmatic cases of "collaborative search" are now collected under the BOINC initiative [18]. Although distributed, these projects are based on the repetition of a simple loop: every involved machine receives from a central server a subset of the search space (signal samples, number intervals), performs an exhaustive coverage of the subset and reports the results, receiving another search subset.

If the configuration space is to be searched by stochastic means, as opposed to exhaustive, these distribution methods are less appealing: having a central coordinator choose a partition of the search space may not be the best choice if the search space is large and only a small portion can be visited, hopefully by concentrating the search in the most promising areas.

We investigate the distribution of the two heuristics described in Sections 3.1 and 3.2 by using epidemic protocols to maintain a loose coordination among all processes.

- **Gossiping Particle Swarms** — Distributing the computational effort (objective function evaluation and trajectory update) among many peers is possible, but not straightforward, given that coordination through the knowledge of the global best position $\boldsymbol{g}$ has to be implemented. Therefore we are studying distribution strategies that depend on two criteria:
  - Every node manages a number of trajectories according to the classic version, which is the limit case of a single participant node.
  - The requirement of an up-to-date global optimum is relaxed: every node maintains its own value for $\boldsymbol{g}$ and, when convenient, updates its value and communicates it to other nodes by gossiping.

  The number of particles per node and the gossiping frequency are the main parameters considered in our analysis, to study the tradeoff between message complexity and quality of solutions.

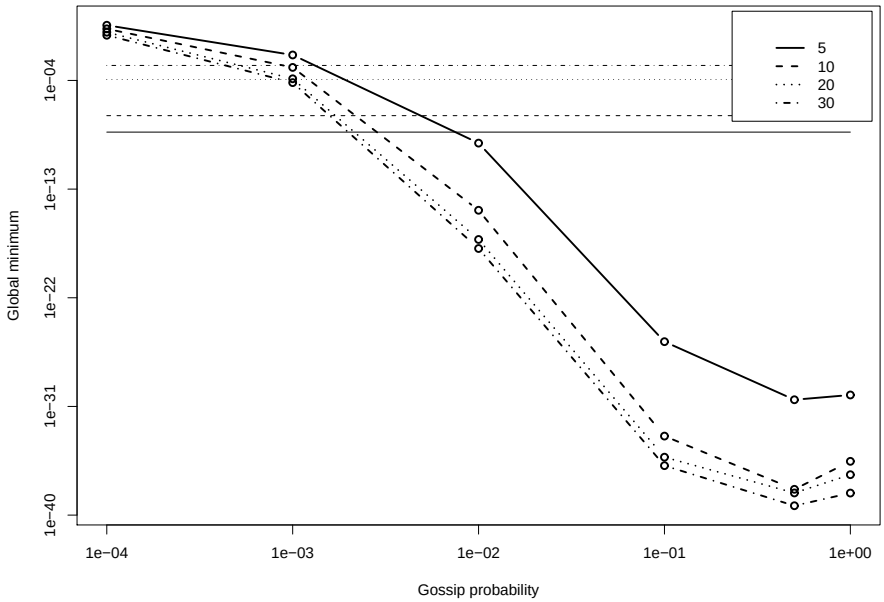

Figure 1: PSO with various message exchange probabilities; horizontal lines refer to the single-node scenario

- **Anti-entropy Memory-based RASH** — In order to distribute the M-RASH algorithm, every node should maintain its own past history and use it to model the function landscape and locate the best suitable starting point. Occasionally, pairs of nodes shall communicate and share relevant information about their past history in order to build a better common model. The frequency and randomness of information exchanges are the crucial parameters in this case.

5 Experimental results

Our preliminary experiments on the distribution of the particle swarm heuristic consider the case in which particle i maintains the best point $\boldsymbol{p}_i$ along its trajectory and, in place of the global best $\boldsymbol{g}$, a "personal" version $\boldsymbol{g}'_i$ which is updated by exchanging information with other particles. In particular, after every evaluation, particle i decides with a fixed probability to exchange its version $\boldsymbol{g}'_i$ of the global best with another random particle.

The results of these preliminary experiments on gossiping optimization heuristics are shown in Figure 1. Every point represents the average outcome of 30 runs where every particle performs 1000 ten-dimensional Zakharov function eval-

uations. The different lines represent experiments with a different number of particles (5 to 30). The plots represents the global minimum after 1000 evaluations per particle versus the probability of gossiping, from 10^{-4} to 1.

As a measure of comparison, it is interesting to see what a single node would do if charged with the same 1000-evaluation computational load that it sustains in the distributed version, but managing all particles and thus with less evaluations per particle. The horizontal lines in Figure 1 represent the average outcome of 30 runs of the classical, sequential Particle Swarm algorithm with 1000 function evaluations and with a number of particles ranging from 5 to 30. Note that, due to the small number of available evaluations, solutions that explore less trajectories (but for a larger number of steps) yield better results.

The simulations show that a higher gossiping probability generates better results, as expected. However, a small message exchange (one in 100 evaluations) allows the distributed version to be competitive with respect to the sequential one.

6 Conclusions

This paper has presented ongoing work on the implementation of memory-based search algorithms in 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.

Preliminary results on the Particle Swarm heuristic show that the message complexity needed to outperform a single-node sequential algorithm can be low, and that the proposed approach is therefore viable.

References

[1] R. Battiti and G.Tecchiolli. Parallel biased search for combinatorial optimization: Genetic algorithms and tabu. *Microprocessor and Microsystems*, 16:351–367, 1992.

[2] R. Battiti and G. Tecchiolli. Learning with first, second, and no derivatives: a case study in high energy physics. *Neurocomputing*, 6:181–206, 1994.

[3] Roberto Battiti and Mauro Brunato. Reactive search: machine learning for memory-based heuristics. In Teofilo F. Gonzalez, editor, *Approximation Algorithms and Metaheuristics*. Taylor and Francis Books (CRC Press), 2006. in press.

[4] Roberto Battiti, Mauro Brunato, and Srinivas Pasupuleti. Do not be afraid of local minima: Affine shaker and particle swarm. Technical Report DIT-05-049, University of Trento, Via Sommarive, 14, 38050 Povo(TN) - Italy, May 2005. Available at: http://rtm.science.unitn.it/~battiti/archive/pso.pdf.

[5] Mauro Brunato, Roberto Battiti, and Srinivas Pasupuleti. A memory-based rash optimizer. In Ariel Felner, Robert Holte, and Hector Geffner, editors, *Proceedings of AAAI-06 workshop on Heuristic Search, Memory Based Heuristics and Their applications*, pages 45–51, Boston, Mass., 2006. ISBN 978-1-57735-290-7.

[6] Vincent A. Cicirello and Stephen F. Smith. *Principles and Practice of Constraint Programming CP 2004*, volume 3258 of *Lecture Notes in Computer Science*, chapter Heuristic Selection for Stochastic Search Optimization: Modeling Solution Quality by Extreme Value Theory, pages 197–211. Springer Berlin / Heidelberg, 2004.

[7] Pedro S. de Souza and Sarosh N. Talukdar. Asynchronous organizations for multi-algorithm problems. In *SAC '93: Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing*, pages 286–293, New York, NY, USA, 1993. ACM Press.

[8] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing (PODC'87)*, pages 1–12, Vancouver, British Columbia, Canada, August 1987. ACM Press.

[9] Carla P. Gomes and Bart Selman. Algorithm portfolios. *Artif. Intell.*, 126(1-2):43–62, 2001.

[10] H. H. Hoos and T. Stützle. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, 2005.

[11] Bernardo A. Huberman, Rajan M. Lukose, and Tad Hogg. An economics approach to hard computational problems. *Science*, 275:51–54, January 3 1997.

[12] Márk Jelasity and Ozalp Babaoglu. T-Man: Gossip-based overlay topology management. In Sven A. Brueckner, Giovanna Di Marzo Serugendo, David Hales, and Franco Zambonelli, editors, *Engineering Self-Organising Systems: Third International Workshop (ESOA 2005), Revised Selected Papers*, volume 3910 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 2006.

[13] Márk Jelasity, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. The peer sampling service: Experimental evaluation of unstructured gossip-based implementations. In Hans-Arno Jacobsen, editor, *Middleware 2004*, volume 3231 of *Lecture Notes in Computer Science*, pages 79–98. Springer-Verlag, 2004.

[14] Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(1):219–252, 2005.

[15] H. Kautz, E. Horvitz, Y. Ruan, C. Gomes, and B. Selman. Dynamic restart policies, 2002.

[16] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)*, pages 482–491. IEEE Computer Society, 2003.

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

[18] The BOINC Project. `http://boinc.berkeley.edu/`.

[19] M. J. Streeter and S.F. Smith. A simple distribution-free approach to the max k-armed bandit problem. In *Proceedings of the Twelfth International Conference on Principles and Practice of Constraint Programming (CP 2006)*, 2006.

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