Peer-to-peer Optimization in Large Unreliable Networks with Branch-and-Bound and Particle Swarms*

Balázs Bánhelyi[1], Marco Biazzini[2], Alberto Montresor[2], and Márk Jelasity[3]

[1] University of Szeged, Hungary, banhelyi@inf.u-szeged.hu
[2] University of Trento, Italy, biazzini,montreso@dit.unitn.it
[3] University of Szeged and HAS, Hungary, jelasity@inf.u-szeged.hu

Abstract. Decentralized peer-to-peer (P2P) networks (lacking a GRID-style resource management and scheduling infrastructure) are an increasingly important computing platform. So far, little is known about the scaling and reliability of optimization algorithms in P2P environments. In this paper we present empirical results comparing two P2P algorithms for real-valued search spaces in large-scale and unreliable networks. Some interesting, and perhaps counter-intuitive findings are presented: for example, failures in the network can in fact significantly *improve* performance under some conditions. The two algorithms that are compared are a known distributed particle swarm optimization (PSO) algorithm and a novel P2P branch-and-bound (B&B) algorithm based on interval arithmetic. Although our B&B algorithm is not a black-box heuristic, the PSO algorithm is competitive in certain cases, in particular, in larger networks. Comparing two rather different paradigms for solving the same problem gives a better characterization of the limits and possibilities of optimization in P2P networks.

1 Introduction

Whereas large scale parallel computing is normally performed using GRID technologies, it is an emerging area of research to apply peer-to-peer algorithms in distributed global optimization. P2P algorithms can replace some of the centralized mechanisms of GRIDs that include monitoring and control functions. For example, network nodes can distribute information via "gossiping" with each other and they can collectively compute aggregates of distributed data (average, variance, count, etc) to be used to guide the search process [1]. This in turn increases robustness and communication efficiency, allows for a more fine-grained control over the parallel optimization process, and makes it possible to utilize large-scale resources without a full GRID control layer and without reliable central servers.

The interacting effects of problem difficulty, network size, and failure patterns on optimization performance and scaling behavior are still poorly understood in P2P global

* This work was supported by the European Space Agency through Ariadna Project "Gossip-based strategies in global optimization" (21257/07/NL/CB). M. Jelasity was supported by the Bolyai Scholarship of the Hungarian Academy of Sciences. B. Bánhelyi was supported by the Ferenc Deák Scholarship No. DFÖ 19/2007, Aktion Österreich-Ungarn 70öu1, and OTKA T 048377.

optimization. In this paper we present empirical results comparing two P2P algorithms for real-valued search spaces in large-scale and unreliable networks: a distributed particle swarm optimization (PSO) algorithm [2] and a novel P2P branch-and-bound (B&B) algorithm based on interval arithmetic. Although our B&B algorithm is not a black-box heuristic, the PSO algorithm is competitive in certain cases, in particular, in larger networks. Some interesting, and perhaps counter-intuitive findings are presented as well: for example, failures in the network can in fact significantly *improve* the performance of P2P PSO under some conditions.

Other P2P heuristic optimization algorithms include [3, 4, 2]. They all build on gossip-based techniques [1, 5] to spread and process information, as well as to implement algorithmic components such as selection and population size control. Our focus is somewhat different from [3, 4] where it was assumed that population size is a fixed parameter that needs to be maintained in a distributed way. Instead, we assume that the network size is given, and should be exploited as fully as possible to optimize speed. In this context we are interested in understanding the effect of network size on performance, typical patterns of behavior, and related scaling issues.

In the case of B&B, we are not aware of any fully P2P implementations. The closest approach is [6], where some components, such as broadcasting the upper bound, are indeed fully distributed, however, some key centralized aspects of control remain, such as the branching step and the distribution of work. In unreliable environments we focus on, this poses a critical problem for robustness.

2 The Algorithms

Our target networking environment consists of independent nodes that are connected via an error-free message passing service: each node can pass a message to any target node, provided the address of the target node is known. We assume that node failures are possible. Nodes can leave and new nodes can join the network at any time as well.

In our P2P algorithms all nodes have identical roles running identical algorithms. Joining and failing nodes are tolerated automatically via the inherent (or explicit) redundancy of the algorithm design. This is made possible via a randomized communication substrate, the *peer sampling service* both algorithms are based on.

The peer sampling service enables all the nodes to send a message to a *random* node from the network at any time. This very simple communication primitive is called the *peer sampling service* that has a wide range of applications [7]. In this paper we will use this service as an abstraction, without referring to its implementation; lightweight, robust, fully distributed implementations exist based on the analogy of *gossip* [7]. The algorithms below will rely on two particular applications of the peer sampling service. The first is gossip-based broadcasting where, nodes periodically communicate pieces of information they consider "interesting" to random other nodes. This way, information spreads exponentially fast. The second is diffusion-based load balancing, where nodes periodically test random other nodes to see whether those have more load or less load, and then perform a balancing step accordingly. This process models the diffusion of the load over a random network.

Algorithm 1 P2P B&B

```
1:  loop                                                              ▷ main loop
2:      I ← priorityQ.getFirst()                                      ▷ most promising interval; if queue empty, blocks
3:      (I_1, I_2) ← branch(I)                                        ▷ cut the interval in two along longest side
4:      min_1 ← upperBound(I_1)                                       ▷ minimum of 8 random samples from interval
5:      min_2 ← upperBound(I_2)
6:      min ← min(min,min_1,min_2)                                    ▷ current best value known locally
7:      b_1 ← lowerBound(I_1)                                         ▷ calculates bound using interval arithmetic
8:      b_2 ← lowerBound(I_2)
9:      priorityQ.add(I_1, b_1)                                       ▷ queue is ordered based on lower bound
10:     priorityQ.add(I_2, b_2)
11:     priorityQ.prune(min)                                          ▷ remove entries with a higher lower bound than min
12:     p ← getRandomPeer()                                           ▷ calls the peer sampling service
13:     sendMin(p, min)                                               ▷ gossips current minimum
14:     if p has empty queue or local second best interval is better than p's best then
15:         sendInterval(p, priorityQ.removeSecond())                 ▷ gossip-based load balancing step
16: procedure ONRECEIVEINTERVAL(I(⊆ D), b)
17:     priorityQ.add(I, b)                                           ▷ D ⊆ ℝ^d is the search space, b is lower bound of I
18: procedure ONRECEIVEMIN(min_p)
19:     min ← min(min_p,min)
```

Based on gossip-based broadcasting, a distributed implementation of a PSO algorithm was proposed [2]. Here we will use a special case of this algorithm, where particles are mapped to nodes: one particle per node. The current best solution, a key guiding information in PSO, is spread using gossip-based broadcast. This means we have a standard PSO algorithm where the number of particles equals the network size and where the neighborhood structure is a dynamically changing random network.

We now move on to the discussion of P2P B&B. Various parallel implementations of the B&B paradigm are well-known [8]. Our approach is closest to the work presented in [9] where the bounding technique is based on interval-arithmetic [10]. The important differences stem from the fact that our approach is targeted at the P2P network model described above, and it is based on gossip instead of shared memory. The basic idea is that, instead of storing it in shared memory, the lowest known upper bound of the global minimum is broadcast using gossip. In addition, the intervals to be processed are distributed over the network using gossip-based load balancing. The algorithm that is run at all nodes is shown in Algorithm 1. The lower bound for an interval is calculated using interval arithmetic [10], which guarantees that the calculated bound is indeed a lower bound. We start the algorithm by sending the search domain D with lower bound $b = \infty$ to a random node.

3 Experimental Results

The algorithms described above were compared empirically using the P2P network simulator PeerSim [11].We selected well-known test functions as shown in Table 1. We included Sphere2 and Sphere10 as easy unimodal functions. 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. These two functions were designed to mislead local optimizers. Finally, Levy4 is not unlike Griewank10, but more asymmetric, and involves higher amplitude noise as

	Function $f(x)$	D	$f(x^*)$	K
Sphere2	$x_1^2 + x_2^2$	$[-5.12, 5.12]^2$	0	1
Sphere10	$\sum_{i=1}^{10} x_i^2$	$[-5.12, 5.12]^{10}$	0	1
Griewank10	$\sum_{i=1}^{10} \frac{x_i^2}{4000} - \prod_{i=1}^{10} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^{10}$	0	$\approx 10^{19}$
Schaffer10	$0.5 + (\sin^2(\sqrt{\sum_{i=1}^{10} x_i^2}) - 0.5)/$ $(1 + (\sum_{i=1}^{10} x_1^2)/1000)^2$	$[-100, 100]^{10}$	0	≈ 63 spheres
Levy4	$\sin^2(3\pi x_1) +$ $\sum_{i=1}^{3}(x_i - 1)^2(1 + \sin^2(3\pi x_{i+1})) +$ $(x_4 - 1)(1 + \sin^2(2\pi x_4))$	$[-10, 10]^4$	-21.502356	71000

Table 1. Test functions. D: search space; $f(x^*)$: global minimum value; K : # local minima.

well. Levy4 is in fact specifically designed to be difficult for interval arithmetic-based approaches.

We considered the following parameters, and examined their interconnection during the experiments: **network size** (N): the number of nodes in the network; **running time** (t): the duration while the network is running (note that it is is *not* the sum of the running time of the nodes), the unit of time is one function evaluation; **function evaluations** (E): the number of *overall* function evaluations performed in the network; **quality** (ϵ): the difference of the fitness of the best solution found in the entire network and the optimal fitness. For example, if $t = 10$ and $N = 10$ then we know that $E = 100$ evaluations are performed.

Messages are delayed by a uniform random delay drawn from $[0, t_{eval}/2]$ where t_{eval} is the time for one function evaluation. In fact, t_{eval} is considerable in realistic problems, so our model of message delay is rather pessimistic. To simulate churn, in some of the experiments 1% of nodes are replaced during a time interval taken by 20 function evaluations. The actual wall-clock time of one function evaluation has a large effect on how realistic this setting is. We assume that the startup of the protocol is synchronous, that is, all nodes in the network are informed at a certain point in time that the optimization process should begin.

We focus on two key properties: (i) scaling with the constraint of a fixed amount of available function evaluations, and (ii) with the constraint of having to reach a certain solution quality. Our first set of experiments involves running the two algorithms with and without churn until 2^{20} function evaluations are consumed.[4]

Solution quality is illustrated in Figure 1. Due to severe length restrictions we comment on the plots very briefly: we can see that (1) P2P B&B has the good property of refusing to utilize all resources if the function is too easy, eg Sphere; (2) B&B can never benefit from larger network sizes w.r.t. quality by definition, whereas PSO can; (3) on Levy4 PSO outperforms the more "sophisticated" B&B; (4) churn is always harmful for B&B by definition whereas it can be beneficial for PSO!

As of running time, P2P PSO always fully utilizes the network by definition, assuming a synchronous startup of the protocol, so its running time is $2^{20}/N$. In the case

[4] We note here that for B&B, one cycle of Algorithm 1 was considered to take 20 evaluations, that is, in addition to the $2 \cdot 8 = 16$ normal evaluations, the interval-evaluation was considered to be equivalent to 4 evaluations (based on empirical tests on our test functions).

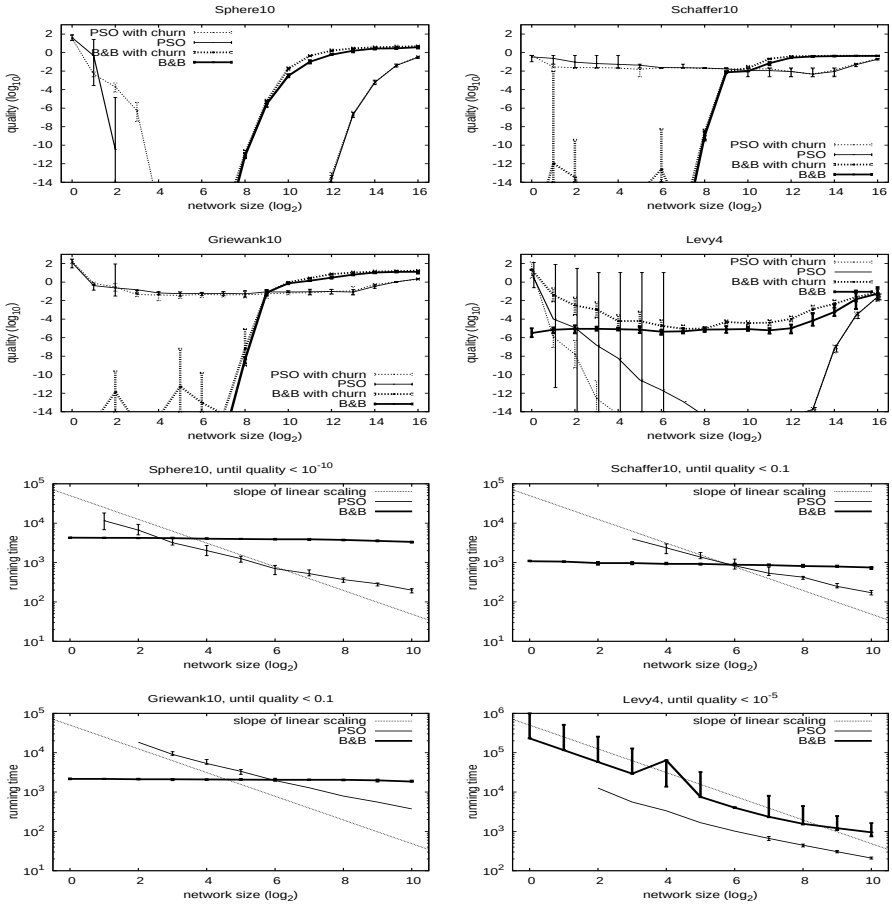

Fig. 1. For quality plots the average digits of precision, for running time plots the median is plotted, for both cases with error bars indicating the 10% and 90% percentiles of 10 experiments (100 experiments for the more unstable Levy4). Note that lower values for quality are better (indicate more precise digits). Missing error bars or line points indicate that not enough runs reached the quality threshold within 2^{20} evaluations (when the runs were terminated).

of P2P B&B, the situation is more complex, as illustrated by Figure 2: running time is sensitive to problem difficulty and churn.

Running time results in Figure 1 illustrate our second question on scaling: the time needed to reach a certain quality. The most remarkable fact that we observe is that on problems that are easy for B&B, it is extremely fast (and also extremely efficient, as we have seen) on smaller networks, but this effect does not scale up to larger networks. In fact, the additional nodes (as we increase network size) seem only to add unnecessary overhead. On Levy4, however, we observe scaling behavior similar to that of PSO.

Finally, to conclude the paper, we note that unlike in small-scale controlled environments, in P2P networks system parameters (like network size and churn rate) effectively play the role of algorithm parameters, but are non-trivial to observe and exploit. An ex-

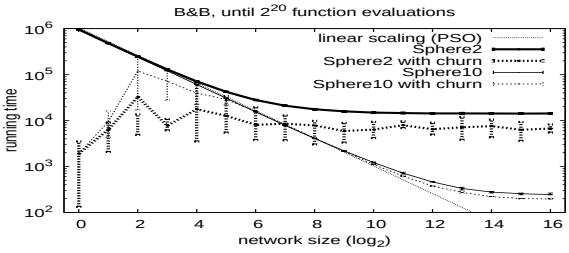

Fig. 2. Running time of P2P B&B to reach 2^{20} evaluations. Average is shown with error bars indicating the 10% and 90% percentiles of 10 experiments

citing research direction is to monitor these parameters (as well as the performance of the algorithm) in a fully-distributed way, and to design distributed algorithms that can adapt automatically.

References

1. Kermarrec, A.M., van Steen, M., eds.: ACM SIGOPS Operating Systems Review 41. (October 2007) Special issue on Gossip-Based Networking.
2. Biazzini, M., Montresor, A., Brunato, M.: Towards a decentralized architecture for optimization. In: Proc. of IEEE IPDPS, Miami, FL, USA (April 2008)
3. Wickramasinghe, W.R.M.U.K., van Steen, M., Eiben, A.E.: Peer-to-peer evolutionary algorithms with adaptive autonomous selection. In: Proc. of GECCO, ACM Press New York, NY, USA (2007) 1460–1467
4. Laredo, J.L.J., Eiben, E.A., van Steen, M., Castillo, P.A., Mora, A.M., Merelo, J.J.: P2P evolutionary algorithms: A suitable approach for tackling large instances in hard optimization problems. In: Proc. of Euro-Par. Volume 5168 of LNCS., Springer-Verlag (2008) 622–631
5. Jelasity, M., Montresor, A., Babaoglu, O.: A modular paradigm for building self-organizing peer-to-peer applications. In Di Marzo Serugendo, G., Karageorgos, A., Rana, O.F., Zambonelli, F., eds.: Engineering Self-Organising Systems. Volume 2977 of LNAI., Springer (2004) 265–282 invited paper.
6. Bendjoudi, A., Melab, N., Talbi, E.G.: A parallel P2P branch-and-bound algorithm for computational grids. In: Proc. of IEEE CCGRID, Rio de Janeiro, Brazil (2007) 749–754
7. Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.M., van Steen, M.: Gossip-based peer sampling. ACM Transactions on Computer Systems **25**(3) (August 2007) 8
8. Talbi, E.G., ed.: Parallel Combinatorial Optimization. Wiley (2006)
9. Casado, L.G., Martinez, J.A., Garcia, I., Hendrix, E.M.T.: Branch-and-bound interval global optimization on shared memory multiprocessors. Optimization Methods and Software **23**(5) (2008) 689–701
10. Ratschek, H., Rokne, J.: Interval methods. In Horst, R., Pardalos, P.M., eds.: Handbook of Global Optimization. Kluwer (1995)
11. PeerSim: http://peersim.sourceforge.net/