TN-Grid and gene@home project: Volunteer Computing for Bioinformatics

Francesco Asnicar[1], Nadir Sella[1], Luca Masera[1], Paolo Morettin[1], Thomas Tolio[1], Stanislau Semeniuta[1], Claudio Moser[2], Enrico Blanzieri[1], and Valter Cavecchia[3]

DISI, University of Trento, Via Sommarive 9, Povo, Trento, Italy
f.asnicar@unitn.it,nadirsella@gmail.com,morettin.paolo@gmail.com,tolio.thomas@gmail.com,stanislau.semeniuta@unitn.it,enrico.blanzieri@unitn.it
CRI, Fondazione Edmund Mach, S. Michele all'Adige, Italy
claudio.moser@fmach.it
CNR-IMEM, Via alla Cascata 56/C, Povo, Trento, Italy
valter.cavecchia@cnr.it

Abstract. The ability to reconstruct and find genes that belong or are connected to a gene regulatory network is of essential importance in biology, in order to understand how the biological processes of an organism work. The main limitation in performing gene network expansion is related to the huge amount of computations needed to discover new candidate genes. Given these premises we decided to adopt the BOINC platform that allows us to use the very powerful computational resources of the volunteers. We set up a BOINC server in which we developed a specific work generator that implements our gene network expansion algorithm. Furthermore, we developed an *ad hoc* version of the PC algorithm (PC++) able to run in the BOINC environment, on the client computers. We present and discuss some statistics and preliminary scientific results of the gene@home BOINC project, the first one hosted by the TN-Grid infrastructure.

Keywords: Volunteer Computing, BOINC, Bioinformatics, Distributed Computing, Gene Network Expansion, System Biology, Computational Biology

1 Introduction

TN-Grid[1] has been thought and developed as a service platform, as a way to give to local research groups, in search of powerful computing infrastructures, a guided access to the power of the world-wide, volunteers based, distributed BOINC computing network. The idea was to use TN-Grid for informing people (researcher, technicians, and students) about BOINC, uncovering its strengths and weaknesses, discussing about integrating their own algorithms and their scientific pipelines into the BOINC framework, and eventually to help them

[1] http://gene.disi.unitn.it/test/

doing this task. Providing access to such a big computational power may also help scientists to broaden their investigation outlooks, going to areas that would have been unfeasible to approach without it.

TN-Grid is the result of a joint effort made by two institutions of the Italian National Research Council (CNR), namely the Institute of Materials for Electronics and Magnetism (IMEM) and the Institute of Cognitive Sciences and Technologies (ISTC), both having local branches in Trento, Italy.

TN-Grid is a so-called *umbrella* project, which means that it is open to host different scientific projects even belonging to distant scientific areas. The first scientific project that we hosted is gene@home, that is a collaboration with Edmund Mach Foundation (FEM) and the Department of Information Engineering and Computer Science (DISI) of the University of Trento. We plan to add other projects to the system in the near future.

At the time of writing TN-Grid is the only public, BOINC based, active project in Italy.

2 Gene@home

The gene@home project is the first one hosted by the TN-Grid infrastructure. The project was born in the fall of 2013 with the collaboration of the students of the Laboratory of Biological Data Mining course. The gene@home project is multi-disciplinary that spans different disciplines: Computer Science, Biology, Statistics, and Data Analysis and can be also defined as a distributed computational biology project. The final goal is the creation of an automatic system for performing Gene Network Expansion in such a way that could be easily used by biologists through a web interface.

As described in the following Section 2.1, network expansion is a complex task aimed to discover relations among genes involved in a particular biological process. In our study, the task is performed by the PC-Iterative Method (PC-IM), using the PC algorithm [14] for inferring causal relations among genes. The biological species we studied so far are *Arabidopsis thaliana* and *Escherichia coli*, and we expanded 2 different local gene networks for the former and 13 for the latter.

2.1 Gene Network Expansion

Gene Network Expansion (GNE) is a research topic in Computational Systems Biology that deals with the discovery of functional dependencies within genes of a species, and genes that take part in the specific biological process to be studied. In biological processes, genes can act as enhancers or inhibitors of the activity of other genes, through a process named Regulation of Gene Expression. Regulation processes are representated by Biological pathways. Nowadays, we have an incomplete knowledge about pathways: discovering new genes is hence important for completing biological pathways, and therefore for gene-specific medical studies, fostering novel methods for pharmaceutical treatment [3].

Inputs of the network expansion algorithm are Omics data[2]. GNE differentiates from the most commonly used Network Inference (NI). NI reconstructs the complete set of gene interactions without the restriction of finding the ones that take part in a specific process, but with a not completely satisfying accuracy and sensitivity when analyzing single biological pathways. Our method for GNE, on the other hand, improves NI's results returning a ranked set of genes interacting with the local gene network of interest.

2.2 PC Algorithm

The PC algorithm [14] is a causal structure discovery method, that can be applied to find causal relations among variables of a system, when an input quantity data matrix representing the system entities is available. As scale-free networks, biological networks are characterized by a power law function on the degree of the nodes [1], and PC algorithm showed to be a valid method to test causal relations in sparse networks, as the biological ones [10].

A pseudo-code of the essential part of the PC algorithm is reported in Algorithm 1. At first, the PC algorithm creates a complete graph, assuming that all the variables are correlated with each other. Nodes of the graph correspond to data matrix variables, hence genes. Once created the complete graph, the algorithm tests the direct correlation between each pair of variables, removing non-correlating edges that do not present a statistically significant correlation, computed using Pearson's correlation coefficient. Then, the algortihm starts to condition all couples of variables X_i, X_j, with $i \neq j$ to all the sets S of neighbors of X_i, such that $S \in Neigh(X_i)$ and $|S| = l$, removing non-correlating edges when conditioned to the set S. This part is inserted in a loop, where the cardinality of S, called level l, increases at each cycle, up to $|Neigh(X_i)|$. This conditioning cycle is the most computationally expensive part of the algorithm. The number of sets of n elements, over a set of k elements (k-*Subset*) is given by the binomial formula $\binom{n}{k}$, that gives a factorial complexity to the algorithm.

Because the removal of edges depends on both input data and variables order (see Section 2.4), it is not possible to know in advance at which level the algorithm will halt: this means that it is not possible to exactly predict the execution time. In our experiments, however, it has never taken more than a few hours run-time on an ordinary laptop.

2.3 PC-IM Algorithm

The PC algorithm is just the core part of the method we used to discover candidate genes for expanding a gene regulatory network. The GNE task is performed by the PC-IM algorithm, which requires as input an already characterized Local

[2] Omics refers to many fields in Biology: Genomics, Transcriptomics, Metagenomics, Proteomics, Metabolomics. Omics aims at the collective characterization and quantification of pools of biological molecules that translate into the structure, function, and dynamics of an organism or organisms.

```
Data: T, Set of transcripts, E expression data
Input: Significance level α
Result: An undirected graph with causal relationship between transcripts
Graph G ← complete undirected graph with nodes in T;
l ← −1;
while l < |G| do
    l ← l + 1;
    foreach ∃u, v ∈ G s.t. |Adj_G(u) \ {v}| ≥ l do // Adj_G(u) adjacent nodes of
    u in G
        if v ∈ Adj_G(u) then
            foreach A ⊆ Adj_G(u) \ {v} s.t. |A| = l do
                if u, v are conditionally independent given A w.r.t. E with
                significance level α then
                    remove edge {u, v} from G;
                end
            end
        end
    end
end
return G;
```

Algorithm 1: PC Algorithm: skeleton procedure [8].

Gene Network (LGN) and gene expression data [4]. The PC-IM algorithm is said to be iterative because the analysis of the whole set of genes is computed multiple times, a parameter that we refer to as *iterations*. Each iteration is performed over a random permutation of the input variables, mitigating in this way the order-dependency issue of the PC algorithm.

Given a LGN, an observation data set, and the size of the graphs into which divide the set of genes of the organism, the PC-IM algorithm generates non-overlapping blocks of extra-LGN genes. To each of these extra blocks, the LGN genes are added. An additional extra block with partially overlapping genes may be added in the case that the data set is not a multiple of graph size minus the number of gene in the LGN. At this point, a single PC algorithm is executed for each block. This process is repeated for the number of iterations. The final output of this process is an ordered list of candidate genes found to be connected with the input LGN.

2.4 PC* Algorithm

The PC algorithm analyzes pairs of variables following the arbitrary order of the variables, in our case genes or microarray probes. If the variables are permuted, the output will change because when an edge in the graph is removed, its absence affects the future conditioning sets. In fact, when the execution removes an edge with conditioning sets of dimension l, it cuts away some conditional dependency to check with conditioning sets of the same dimension.

The PC* algorithm solves the order-dependent problem of the input, postponing the edge removal at the end of each loop, just before increasing the size of the conditioning sets. In more detail, at each level l edges are not removed from the graph, but instead they are marked as "to remove". This allows the algorithm to check a larger space of possible conditional dependencies for a given size of the conditional set S. Since PC* algorithm checks many more conditions, its execution time takes much longer than a single PC run. From the tests we did, PC* returns a subset of the union set of outputs of multiple PCs.

3 BOINC

An execution of PC-IM requires, depending on the parameters, up to thousands of runs of the PC algorithm on input data of relevant size. This setting is particularly suitable for a BOINC project, for this reason we decided to implement the PC-IM method using the BOINC infrastructure [2].

3.1 PC++ Algorithm, BOINC Version

Starting from the R implementation of the PC algorithm, included in the *"pcalg"* package [7, 9], we implemented a C++ version of the *"skeleton"* procedure of the PC algorithm, since we did not need the final DAG (Direct Aciclic Graph) estimation phase. We chose C++ because we needed high computational performances.

Our implementation showed an impressive speed-up of 240 in execution time, and conversly reducing the RAM consumption of about 10 times. We carefully optimized the most CPU demanding subroutines, i.e. the creation of all the $\binom{n}{k}$ subsets and the evolution of the causal dependency-testing formula, when the conditioning set is big. To solve these problems, we used more efficient data structures and switched from recursion to dynamic programming.

3.2 BOINC Server

The BOINC server is the part of the BOINC infrastructure that performs distribution management, data maintenance and project information visualization. Our BOINC server is running on a Virtual machine with 2 GB of RAM and two cores AMD Opteron™ Processor 6276. The basic components included in a BOINC server are:

- a database server (MySQL);
- the BOINC daemons (to name few of them: *scheduler*, *feeder*, *work generator*, *transitioner*, *validator*, *assimilator*, and *file deleter*);
- a web server (Apache).

The MySQL database stores the data related to the BOINC part of the project (e.g. users, computers, workunits, statistics). We made use of MySQL also to store the data regarding the dispatch and management of all PC-IM

executions. This allows us to keep track of which workunits, input observation data and relative output files are related to a specific GNE task. Among all the BOINC daemons, the only one that we modified is the work generator. All the other daemons that are running on our BOINC server have not been modified.

3.3 Work Generator

The work generator generates the workunits. To easily manage and keep track of the PC-IM executions, we first designed and implemented a database (we will refer to it from now on as the *gene database*) using the already running MySQL daemon for BOINC. We also use the *gene database* to manage the input data, users, notifications, and it will be also the middle layer between the work generator and the future web-interface where biologists will schedule new PC-IM executions.

The work generator was implemented using the Python programming language, that allowed us a fast and high-level coding. We collected some measures about the performance of the work generator, such as the single workunit creation time and the overall workunits creation time necessary to complete a single PC-IM. Since we did not find any bottlenecks, we decided to not implement the work generator using more efficient languages. The work generator exploits our *gene database* to know and keep track of the work that will be generated or that has been generated, as well as the possibility to notify the user that submits the specific PC-IM when it is almost finished.

Since BOINC APIs are accessible only through C++ functions and not Python scripts directly, we built two simple C++ programs that wrap all the necessary BOINC functions for the work generator.

Since there is not a direct relation between the execution time of a single PC and the dimension of the inpu, PC time execution can largely vary. So, it's hard for the work generator to exactly estimate a workunit time execution. To overcome this issue, we are planning to build a complete benchmark machine that will execute a few instances of PCs, eventually with different parameters, and use it to estimate the running time of the workunits.

3.4 Post Processing

The processing of the partial results of a PC-IM in order to get the candidate lists starts in the client application, as soon as a single workunit finishes. Indeed, since each workunit cannot contain a PC execution of a different PC-IM, we were able to insert a first partial counting of the arcs found by the PC executions of that workunit, reducing also in this way the size of the output file that the volunteers return back to the server.

The gene@home project has two issues that, in general, creates difficulties related to the BOINC distribution of work. The size of our input data files is generally in the order of one hundred MBs, and the size of our results averages a dozen MBs. Thanks to the developers of BOINC we got an update of the

BOINC server that now implements the distribution and receiving of workunits and results in a gzip compressed format.

Since a single run of PC-IM can produce a very large number of workunits, we developed an *ad hoc* program that is in charge of moving the results of the workunits of a PC-IM, when all of them are returned. The script that moves the results exploits the *gene database*, where for every PC-IM executed by the work generator, we store the number of workunits that has been produced.

On the server, a validation step of the returned workunits is performed, and then the canonical result is moved on a dedicate server, that has a large store capacity. Currently we are using a double validation method, that consists in sending each workunit to two diverse volunteers in order to be able to validate the results later. The returned results then must be equal bitwise. Because of the nature of our project, we have not find a way to internally validate a result of a single workunit, without requiring the double validation phase.

Externally to the TN-Grid infrastructure, we have a pipeline of Python programs that complete the processing of the partial results of each PC-IM.

4 Educational and Social Aspects

The gene@home project was born from a conversation between Prof. Enrico Blanzieri (University of Trento), Dr. Valter Cavecchia (CNR), and Dr. Claudio Moser (FEM). Its realization and running involved students and BOINC volunteers.

4.1 Gene@home as a Course Project

In the first semester of academic year 2013-2014, the project was proposed to the students of the Biological Data Mining Laboratory course held in the master Computer Science program of the University of Trento. Claudio Moser and his collaborators at Foundation Edmund Mach provided biological annotated data and a preliminary version of the method implemented in R. Claudio Moser also covered the relevant biological topics during the course. The initial ambitious goal set was to systematically expand networks of interest of *Arabidopsis thaliana*. The attendance of the course increased steadly and eventually 22 students formed four groups devoted respectively to: 1) write the BOINC application; 2) manage the BOINC server; 3) preprocess the input data and post-process the results and 4) take care of communication and of the web site.

Students, now the first five authors of this paper, developed a C++ application for directly communicating with the BOINC client through the provided BOINC API. The executables were built for different operating systems and architectures, Windows (both 32 and 64 bit), Linux (both 32 and 64 bit), and Mac OS X. After having a first version of the server and the client applications, the students tested the whole system, finally concluding the pre-alpha stage. The same students continued, in the form of a Research Project the activity in the second semester and gained extra academic credits.

The continuation of the BOINC project was proposed also during the first semester of the academic year 2014-2015 and one student, Stanislau Semeiuta, with the CUDA implementation of PC* while others worked on the application of gene@home to *E. coli*.

Overall, the class reached almost all the technical goals, with a large part of the students really engaged and who expressed a positive evaluation of the course with the exception of a small minority. Introducing BOINC in the teaching activity involved the students on several topics of distributed systems and it proved to be a good way of gaining technical and collaborative competences in a medium-size software project. Moreover, the students shared the general reasearch goals and many of them worked beyond expectation.

4.2 BOINC Community

We contacted the administrators of the largest Italian BOINC users community (BOINC.Italy[3]) announcing the second, alpha phase of our project and asking them and their users to join us using the BOINC invitation code mechanism. This procedure implies registering the user on the projects web page before attaching the client to the project, also passing through CAPTCHA verification. This will filter the server from spammers and bots, minimizing the burden of the maintenance tasks.

After some time, some of the most active users in the BOINC world contacted us asking information about our project. We decided then to send the invitation code to anyone interested, explicit allowing them to re-distribute the invitation code to other people, but not to publishing it in public posts. Until today this rule was fully respected. Some statistics sites, e.g., BOINCstats[4] and Free-DC[5] also requested permission to collect and manage statistics data from our server. So, by providing a continuous flow of workunits, we started to see an increasing number of participants (see Figure 1A).

The computational power provided by the volunteers increased, reaching a peak of 1.5 gigaflops during December 2014 (see Figure 1B).

However, the credit per day, which is a good estimate of the "instant" power of the system, is recently (at mid February 2015) decreasing (see Figure 1C). From this chart, we may also notice that the average power (recent average credit, averaged over a week period, RAC) is also decreasing. There are many possible reasons for this:

- Most of the users are power users, that are users which provide high computational power. However, power users also like to frequently switch their computational power to different projects, pursuing their own interests, challenges, credit milestones, and badges. At the time of writing we count 175 registered users and 821 registered computers, with an average of 4.7 computers per user, with powerful machines running 24/7;

[3] http://www.boincitaly.org
[4] http://boincstats.com
[5] http://stats.free-dc.org

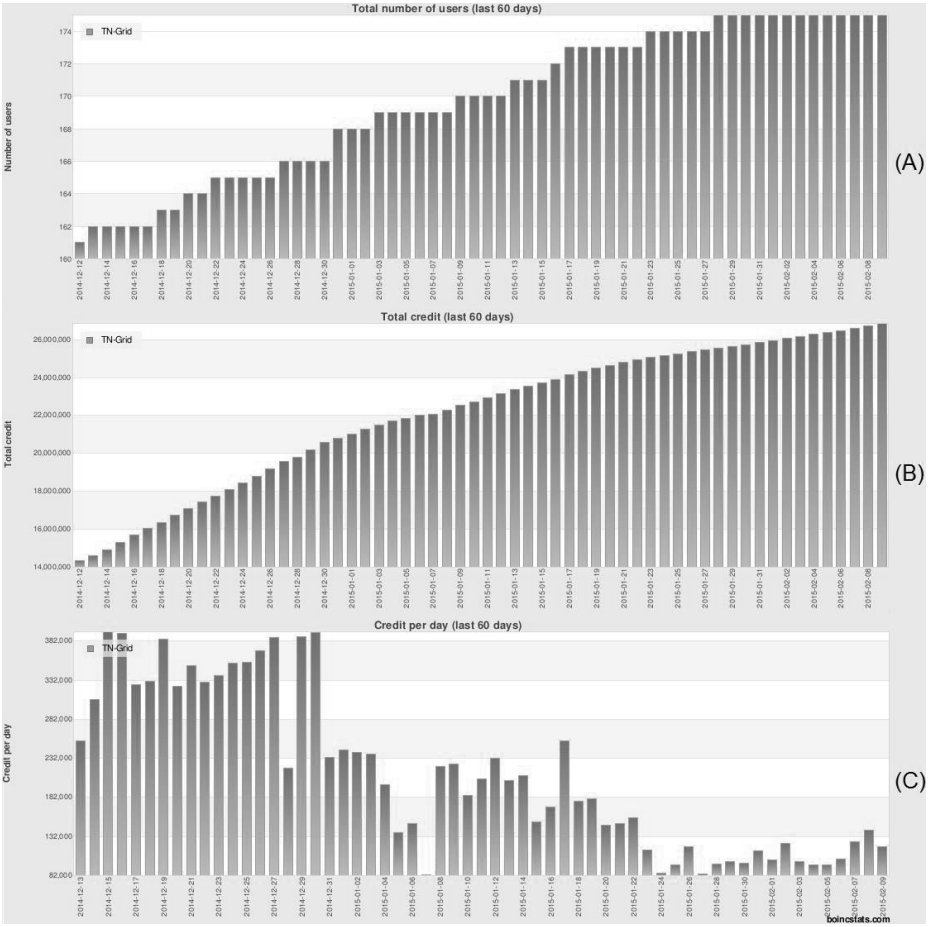


Fig. 1: Snapshots were taken on February 10^{th} (source BOINCstats). (A) Total number of users on gene@home during the last 60 days. (B) Total credit production (cumulative) on gene@home during the last 60 days. The credit trend depicted above is proportional to the flops values (1 gigaflop machine, running 24 hours a day, produces 200 units of credit in 1 day). (C) Amount of assigned credits per day on gene@home during the last 60 days.

– We were not very informative about the status of our project and also news were issued rarely, losing our own appeal. Now, we need to keep the interest high, providing more frequent status updates and general information about our progresses.

In the forthcoming future, we will move to a new phase, switching to a semi-public stage, posting the invitation code to the project's home page. We are also designing credit badges, that can attract volunteers interested in collecting achievements.

5 Results

During the execution of the gene@home project we collected two types of data: statistical data about the performance of the BOINC server and application, and scientific results obtained after having analyzed the results of the workunits computed by the volunteers.

5.1 BOINC Results

After running the server for some months, we were able to collect some statistics about the reliability of the gene@home application, i.e. the ratio between valid and invalid results. This impacts both the scientific quality and the user experience of our project. Users really dislike running an application who ends up producing errors, thus not getting credits for the work done and wasting time and electric power: they could leave the project.

Statistics are automatically prepared by the BOINC server every day (see Figure 2), the data are taken as snapshots of the BOINC workunits database. In the summary Table 1, errors count includes only the computational errors: compute errors, validated errors, and invalid results. Results returned after the deadline or aborted by users, or download errors were not considered as errors.

Table 1: BOINC statistics of the gene@home protject taken on four different days during the year 2014 (reference period: the previous seven days). Total number of returned results (#Over), successfully computed (Success), validated (Valid), pending validation (#Initial), and faulty (#Errors).

Date	**#Over**	**Success (%)**	**Valid (%)**	**#Initial**	**#Errors**
22 April	15543	15392 (99.0%)	15340 (98.7%)	19	85
20 May	69536	68621 (98.7%)	67096 (97.7%)	1450	89
16 December	33232	31798 (96.2%)	29525 (88.8%)	2147	38
24 December	91315	89536 (98.1%)	87584 (95.9%)	1716	61

Results presented in Table 1 prove that our application is very reliable, although we still have some validation issues. We are still having, although rarely,

validation problems (see Figure 2). Computers running the same task may return different results due to incorrect client software configuration. Otherwise, the problem could arise from a small bug inside the application checkpoint mechanism linked to a stop-and-restart after the very first seconds of running. We still need to further investigate this issue.

We are distributing executables built for various operating systems and architectures; namely Windows (both 32 and 64 bit), Linux (both 32 and 64 bit), and Mac OSX. We need to build statically linked executable for Linux for users running very old Linux kernels.

Handling of so-called *leftovers*. We distribute work in "batches", i.e. sets of workunits belonging to the same sub-problem (a single run of a PC-IM). Sometimes it happens that almost all the workunits of the same set are returned and very few of them are not processed and waiting for timeouts before being re-distributed. We would like to find a way to compute this kind of workunits on a dedicated server in order to reduce the time needed to complete a PC-IM. One possible solution would be to use BOINC `restrict_wu_to_user()` mechanism to send all such workunits to a reliable, dedicated, and active user.

TN-Grid: Result summary

95988 results

Server state	# results
Inactive	0
Unsent	1080
Unknown	0
In progress	3593
Over	91315

'Over' results

Outcome	# results
---	0
Success	89536
Couldn't send	0
Computation error	1104
No reply	570
Didn't need	12
Validate error	0
Abandoned	93

'Success' results

Validate state	# results
Initial	1716
Valid	87584
Invalid	51
Workunit error - check skipped	0
Checked, but no consensus yet	10
Task was reported too late to validate	175

File Delete state	# results
Initial	1726
Ready to delete	0
Deleted	87810
Delete Error	0
Total files deleted	87810

'Client error' results

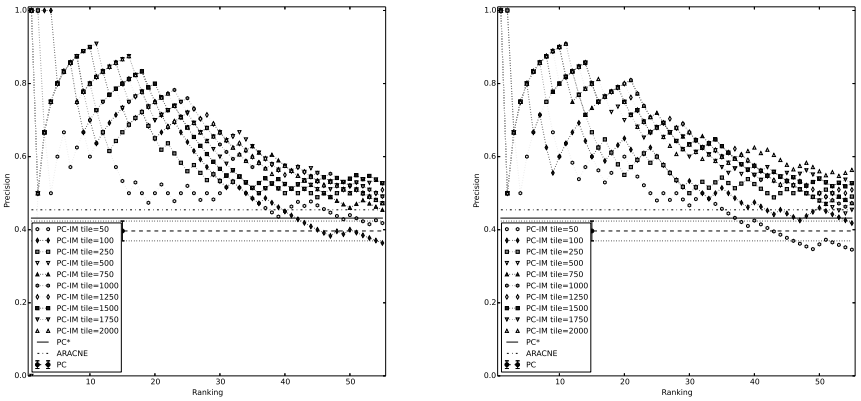
Client state	# results
Downloading	21
Processing	0
Compute error	10
Uploading	0
Done	10
Aborted by user	1063

Fig. 2: Summary of the server statistics of the TN-Grid infrastructure running the gene@home project. Seven days snapshot, taken December 24$^{\text{th}}$, 2014.

5.2 Preliminary Report on the Scientific Results

The experiments in Figure 3 were conducted using the Flower Organ Specification Gene Regulatory Network (FOS) of the model plant *A. thaliana*, composed of 15 genes connected with 54 edges [6, 13]. The data is composed of gene expression values available in the PLEXdb database [5] and consists of 393 hybridiza-

tions of 22,810 microarray probes[6]. Plots in Figure 3 represent the trend of the precision of several PC-IM executions, ran with different values of tile size, and $\alpha = 0.05$. The precision is computed by comparison with a manually curated classification of the probes of *A. thaliana*. The comparison between the two plots permits to appreciate the change in precision by varying the iteration parameter i. The two plots show also, as a comparison reference, the precision computed on the results of the three major competitors: PC, PC*, and ARACNE [12, 11] (using the default parameters), the complete scientific results are in the process of being published.



(a) $i = 20$ (b) $i = 2000$

Fig. 3: Precision comparison of *A. thaliana* on FOS network. PC (average and variance), PC*, ARACNE, and PC-IMs with different tile size with a fixed $\alpha = 0.05$. We considered the first 55 genes of the result lists of each experiment and for each result lists we computed 55 precision values by considering an increasing list that initially contains the first gene found. All PC-IMs in Figure 3a have a number of iterations $i = 20$, in Figure 3b the number of iterations $i = 2000$.

6 Ongoing Developments

PC-IM performs a lot of computation, and even if our C++ implementation of the PC algorithm is really fast, we tried to achieve even better performances. For this reason we tried also solutions exploiting multithreading and Graphics processing unit (GPU) computing.

[6] Probe is a general term for a piece of DNA or RNA that corresponds to a gene or a sequence of interest that has been labelled by biologists.

6.1 From Multithreading to GPU Computing

One of the goals is the performance improvement of a single PC run: this would help the PC-IM to be runnable on standard local machines. We opted for code parallelization. After some analysis, we concluded that the *skeleton* procedure is not trivially parallelizable, due to the edge removal-associated consequences that require complicated synchronization strategies to create a parallel version equivalent to the single threaded one. Instead, PC* is trivially parallelizable.

Initially, we introduced multithreaded processing using the Intel Threading Blocks (TBB) library[7] to parallelize PC*. In our implementation, we launch a number of threads, each taking as input one gene and the separation set size, that search among all genes for those ones that are conditionally independent given a size-specific separation set. Once checked all the pairs for one gene, it goes to the next unprocessed one. As edge removal is postponed, there is no synchronization between processing parts of threads. The only place that needs synchronization is the mutually exclusive list of unprocessed genes, whose access time is negligible with respect to the time to process gene expression data.

Table 2: Time (reported in milliseconds) to process one level of PC*. Columns marked with **CPU** report timings of 4-threads execution of the algorithm. Organisms: **At** stand for *Arabidopsis thaliana* and **Ec** for *Escherichia coli*, respectively.

Tile size	**CPU 1000**	**GPU 1000**	**CPU 2000**	**GPU 2000**	**CPU 100**	**GPU 100**	**CPU 200**	**GPU 200**
Organism	**At**	**At**	**At**	**At**	**Ec**	**Ec**	**Ec**	**Ec**
Separation set size								
0	47	5	200	9	<1	<1	<1	<1
1	2940	600	18000	1350	<1	<1	80	3
2	1180	3100	9000	8000	90	40	890	320
3	68	100	600	220	320	100	2950	980
4	10	44	100	90	580	190	5330	1630
5	15	44	15	100	500	220	5515	2380
6	10	44	15	74	490	290	4437	3170
7			10	74	390	390	3690	4975
8			10	74	230	490	2500	6630
9					93	430	1800	8380
10							650	7020
11							230	5840
12							80	3950
13							15	2260

[7] https://www.threadingbuildingblocks.org

The final implementation produces exactly the same results as the single threaded PC*, but much faster. In our experiments, we have observed that it take less time by a factor of T to get the results, where T is a number of processing threads.

Then we decided to move to GPU computing, choosing to use NVIDIA CUDA for its better development tools with respect to OpenCL. The algorithm is conceptually the same as the TBB-based, but it has to take into account the need of transfer data between CPU and GPU memories, the differences in computation models of GPUs and CPUs, and the fact that single GPU threads are slower than single CPU ones.

Our NVIDIA GPU-based implementation showed to be coherent with the previous ones, so we evaluated its pros and cons. The most important timings are presented in Table 2. It can be seen from both tables that the GPU version significantly outperforms the CPU one on small sizes of separations sets. As we were using the parallel implementation of PC* on a 4 core machine, that gives approximately a speed-up of 50 for separation set size of 0 and 1 with respect to the initial single core version. We also observe that the performance boost depends on the nature of data being processed. Table 2 shows that, for this particular data, it is beneficial to run the GPU version up to a separation set size of 4-6, while it is not the case with the other data that we have tested.

7 Conclusion

The first project hosted on the TN-Grid infrastructure is gene@home that, involving volunteer computing, implements a gene network expansion algorithm. We presented our project from different points of view: the technical side in which we described the implementation and setup of the BOINC platform, the educational aspect that involved students of the University of Trento, and the social part that involves the relationship with the BOINC community of volunteers participating in our project.

We gave some details of the way we setup our BOINC server, the server software, and the features of the client application that we developed. The gene@home project started during the Laboratory of Biological Data Mining course in 2013-2014 at the University of Trento and engaged a group of students to what is now a long-term project. The social aspect of the participation in the gene@home project by BOINC users is important, and we discussed in particular the trend of the points assigned to the volunteers. In fact, gene@home initially was very attractive thanks also to the novelty of the problem, now we realized that we need to communicate the results in a steadier way in the near future.

The empirical data on gene@home comprises both statistical results of the BOINC performance that we obtained during the last year of executions, and a preliminary report of the scientific result that shows the effectiveness of the method.

8 Acknowledgments

The authors wish to thank Giulia Malacarne and Emanuela Coller of Edmund Mach Foundation, Daniele Campana, Giulia Corn, Laura Escobar, Ahmed Fadhil, Marco Giglio, Davide Giovannini, Bhuvan Hrestha, Erinda Jaupaj, Paolo Leoni, Laura Malvaso, Trung Nguyen, Quiynh Nguyen, Eko Susilo, Daniele Tovazzi, Chau Tran, and all the volunteers, in particular the BOINC Italy group.

References

1. R. Albert. Scale-free networks in cell biology. *Journal of cell science*, 118(21):4947–4957, 2005.
2. D. P. Anderson. Boinc: A system for public-resource computing and storage. In *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*, pages 4–10. IEEE, 2004.
3. R. Arroyo, G. Suñé, A. Zanzoni, M. Duran-Frigola, V. Alcalde, T. H. Stracker, M. Soler-López, and P. Aloy. Systematic identification of molecular links between core and candidate genes in breast cancer. *Journal of molecular biology*, 2015.
4. E. Coller. *Analysis of the PC algorithm as a tool for the inference of gene regulatory networks: evaluation of the performance, modification and application to selected case studies*. PhD thesis, DISI, University of Trento, 4 2013.
5. S. Dash, J. Van Hemert, L. Hong, R. P. Wise, and J. A. Dickerson. PLEXdb: gene expression resources for plants and plant pathogens. *Nucleic Acids Res.*, 40(Database issue):D1194–1201, Jan 2012.
6. C. Espinosa-Soto et al. A gene regulatory network model for cell-fate determination during Arabidopsis thaliana flower development that is robust and recovers experimental gene expression profiles. *Plant Cell*, 16(11):2923–2939, Nov 2004.
7. A. Hauser and P. Bühlmann. Characterization and greedy learning of interventional Markov equivalence classes of directed acyclic graphs. *Journal of Machine Learning Research*, 13:2409–2464, 2012.
8. M. Kalisch and P. Bühlmann. Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *J. Mach. Learn. Res.*, 8:613–636, May 2007.
9. M. Kalisch, M. Mächler, D. Colombo, M. H. Maathuis, and P. Bühlmann. Causal inference using graphical models with the R package pcalg. *Journal of Statistical Software*, 47(11):1–26, 2012.
10. M. H. Maathuis et al. Predicting causal effects in large-scale systems from observational data. *Nat. Methods*, 7(4):247–248, Apr 2010.
11. A. A. Margolin et al. ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7 Suppl 1:S7, 2006.
12. A. A. Margolin et al. Reverse engineering cellular networks. *Nat Protoc*, 1(2):662–671, 2006.
13. Y. E. Sanchez-Corrales et al. The Arabidopsis thaliana flower organ specification gene regulatory network determines a robust differentiation process. *J. Theor. Biol.*, 264(3):971–983, Jun 2010.
14. P. Spirtes, C. N. Glymour, and R. Scheines. *Causation, prediction, and search*, volume 81. MIT press, 2000.