NuChart-II: a Graph-Based Approach for Analysis and Interpretation of Hi-C Data.

Fabio Tordini[1], Maurizio Drocco[1], Ivan Merelli[3], Luciano Milanesi[3], Pietro Liò[2], and Marco Aldinucci[1]

[1] Department of Computer Science, University of Turin, Corso Svizzera 185, 10149 Torino, Italy
[2] Computer Laboratory, University of Cambridge, Trinity Lane, Cambridge CB2 1TN, UK
[3] Institute for Biomedical Technologies - Italian National Research Council, via F.lli Cervi 93, 20090 Segrate (Mi), Italy

Abstract. Long-range chromosomal associations between genomic regions, and their repositioning in the 3D space of the nucleus, are now considered to be key contributors to the regulation of gene expressions, and important links have been highlighted with other genomic features involved in DNA rearrangements. Recent Chromosome Conformation Capture (3C) measurements performed with high throughput sequencing (Hi-C) and molecular dynamics studies show that there is a large correlation between co-localization and co-regulation of genes, but these important researches are hampered by the lack of biologists-friendly analysis and visualisation software. In this work we present NuChart-II, a software that allows the user to annotate and visualize a list of input genes with information relying on Hi-C data, integrating knowledge data about genomic features that are involved in the chromosome spatial organization. This software works directly with sequenced reads to identify related Hi-C fragments, with the aim of creating gene-centric neighbourhood graphs on which multi-omics features can be mapped. NuChart-II is a highly optimized implementation of a previous prototype package developed in R, in which the graph-based representation of Hi-C data was tested. The prototype showed inevitable problems of scalability while working genome-wide on large datasets: particular attention has been paid in optimizing the data structures employed while constructing the neighbourhood graph, so as to foster an efficient parallel implementation of the software. The normalization of Hi-C data has been modified and improved, in order to provide a reliable estimation of proximity likelihood for the genes.

1 Scientific Background

The representation and interpretation of omics data is complex, also considering the huge amount of information that are daily produced in the laboratories all around the world. Sequencing data about expression profiles, methylation patterns, and chromatin domains are difficult to describe in a systemic view. The question is: how is it possible to represent omics data in an effective way? This problem is critical in these years that see an incredible explosion of the available molecular biology information. In particular, the integration and the interpretation of omics data in a systems biology way is complex, because approaches such as ontology mapping and enrichment analysis assume as prerequisite an independent sampling of features, which is clearly not satisfied while looking at long-range chromatin interactions.

In this context, recent advances in high throughput molecular biology techniques and bioinformatics have provided insights into chromatin interactions on a larger scale, which can give a formidable support for the interpretation of multi-omics data. The three-dimensional conformation of chromosomes in the nucleus is important for many cellular processes related to gene expression regulation, including DNA accessibility, epigenetics patterns and chromosome translocations [1]. The Chromosome Conformation Capture (**3C**) technology [2] and the subsequent genomic variants (Chromosome Conformation Capture on-Chip [3] and Chromosome Conformation Capture Carbon Copy [4]) are revealing the correlations between genome structures and biological processes inside the cell and permit to study the nuclear organisation at an unprecedented resolution.

The combination of high-throughput sequencing with the above-mentioned techniques, (generally called **Hi-C**), allows the characterisation of long-range chromosomal interactions genome-wide [5]. Hi-C gives information about coupled DNA fragments that are cross-linked together due to spatial proximity, providing data about the chromosomal arrangement in the 3D space of the nucleus. If used in combination with chromatin immunoprecipitation, Hi-C can be employed for focusing the analysis on contacts formed by particular proteins.

In a previous work [6], we developed an R package called *NuChart*, which allows the user to annotate and statistically analyse a list of input genes with information relying on Hi-C data, integrating knowledge about genomic features that are involved in the chromosome spatial organisation. NuChart works directly with sequenced reads to identify the related Hi-C fragments, with the aim of creating gene-centric neighbourhood graphs on which multi-omics features can be mapped. Gene expression data can be automatically retrieved and processed from the Gene Expression Omnibus and ArrayExpress repositories to highlight the expression profile of genes in the identified neighbourhood. The Hi-C fragment visualisation provided by NuChart allows the comparison of cells in different conditions, thus providing the possibility of novel biomarkers identification. Although this software has been proved to be a valid support for Hi-C analysis, the implementation relying on the R environment is a limiting factor for the scalability of the algorithm, which can not cover large genomic regions due to the high computational effort required. Moreover, its overhead in managing large data structures and its weaknesses in exploiting the full computational power of multi-core platforms make NuChart unfit to scale up to larger data sets and highly precise data analysis (which requires many iterations of graph building process).

Normalization. 3C-based techniques employed for the characterization of the nuclear organization of genomes and cell types have widespread among scientific communities, fostering the development of a number of systems biology methods designed to analyse such data. Particular attention is given to the detection and normalisation of systematic biases: the raw outputs of many genomic technologies are affected both by technical biases, arising from sequencing and mapping, and biological factors, resulting from intrinsic physical properties of distinct chromatin states, that make difficult to evaluate the outcomes.
Yaffe and Tanay [8] proposed a probabilistic model based on the observation of the genomic features. This approach can remove the majority of systematic biases, at the expense of very high computational costs, due to the observation of paired-end reads spanning all possible fragment end pairs. Hu et al [7] proposed a parametric model based on a Poisson regression. This is a simplified, and less computationally intensive normalisation procedure than the one described by Yaffe and Tanay, since it corrects the systematic biases in Hi-C contact maps at the desired resolution level, instead of modelling Hi-C data at the fragment end level. The drawback here is that the sequence information is blurred within the contact map. The first NuChart prototype [6] solved this issue by exploiting Hu et al. solution to estimate a score to each read, identifying half of the Hi-C contact instead of normalizing the contact map, thus preserving the sequence information. NuChart-II leverage this solution proposing an *ex-post* normalisation, that is used to estimate a probability of physical proximity between two genes, expressed as a score assigned to an edge connecting two nodes in the neighbourhood graph.
Among the related works, the majority of applications rely on the creation of contact maps for the interpretation of Hi-C data, combining Principal Component Analysis and Hierarchical Clustering with this representation. The visualization and exploration of Hi-C data assumes a dramatic importance when analysing Hi-C data. To the best of our knowledge, no other tool proposes a gene-centric, graph-based visualization of the neighbourhood of a gene, as NuChart-II does.

Parallel Computing Tools. Over the years, research on loop parallelism has been carried on using different approaches and techniques that vary from automatic parallelisation to iterations scheduling. In this paper we elected *FastFlow* [10] as a viable tool for re-writing NuChart. We then compared the results we have obtained, against *OpenMP* and *TBB*, which represent to a major extent the most widely used and studied frameworks for loop parallelisations.
Intel Threading Building Blocks (TBB) [12] is a library that enables support for scalable parallel programming using standard C++. It provides high-level abstractions to exploit task-based parallelism, independently from the underlying platform details and threading mechanisms. The TBB `parallel_for` and `parallel_foreach` methods may be used to parallelise independent invocation of the function body of a for loop, whose number of iterations is known in advance.
OpenMP [13] uses a directive based approach, where the source code is annotated with pragmas (`#pragma omp`) that instruct the compiler about

the parallelism that has to be used in the program. In OpenMP, two directives are used to parallelise a loop: the `parallel` directive declares a parallel region which will be executed concurrently by a pool of threads; the `for` directive is placed within the parallel region to distribute the loop iterations to the threads executing the parallel region.
FastFlow is a parallel programming environment originally designed to support efficient streaming on cache-coherent multi-core platforms [10]. It is realised as a pattern-based C++ framework that provides a set of high-level programming patterns (aka *algorithmic skeletons*). FastFlow exposes a `ParallelFor` pattern [11] to easily deal with loop parallelism.

2 Materials and Methods

Here we present *NuChart-II*, an improved C++ version of the first R prototype software, which enables the user to annotate and visualize Hi-C data in a gene-centric fashion, integrating knowledge data about genomic features that are involved in the chromosome spatial organisation. In the development of this new version of the software, particular attention has been paid at optimising the data structures employed for the management of the information concerning the neighbourhood graph, in order to facilitate the parallel implementation of the algorithm. The computational effort required for the generation of large graphs can now be easily addressed according to the parallelism the application exhibits, properly exploiting the computational power offered by modern multi-core architectures. The re-engineering of the software has been conducted on top of FastFlow, using the `ParallelFor` pattern.
The general idea behind this package is to provide a complete suite of tools for the analysis of Hi-C data. A typical Hi-C analysis will start with the pre-processing of FASTQ files with HiCUP, which produces paired reads files in SAM (or BAM) format [4]. These SAM files represent the main input of NuChart-II, along with a list of genes and an interval of genomic coordinates that should be analysed in terms of Hi-C contacts for the creation of the neighbourhood graph. Hi-C data are analysed using a gene-centric approach: if a Hi-C contact between two genes is present – i.e. there is a paired read that supports their proximity in the nuclear space – an edge is created between their representative vertices. The scalability aspect assumes crucial importance in NuChart-II: while the previous R implementation was not suitable to perform thorough explorations due to bottlenecks in memory management and limitations in the exploitation of the available computational resources, the novel algorithm is fully scalable and can be used for exploring Hi-C contact genome-wide. This is functional to our objective of creating a snapshot of the general organisation of the DNA in the nucleus, which is essential, for example, to exploit chromosome conformation data in cytogenetics analysis or to create a metrics of the distance between the different fragment in terms of contacts.

[4] see the HiCUP documentation for more details: `http://www.bioinformatics.babraham.ac.uk/projects/hicup/`

The novel implementation presented here refines the normalisation phase, which is now conducted *a-posteriori*, during the edge weighing phase: the weight assumes the role of a "confidence score" that qualifies the reliability of each contact represented on the neighbourhood graph.

2.1 Neighbourhood Graph Construction

We recall that a graph is a formal mathematical representation of a collection of vertices (V), connected by edges (E) that model a relationship among vertices. In this context, vertices represent genes. Two genes are *connected* if there exists a paired-end Hi-C data belonging to both of them. We define this paired-end Hi-C data as a *connection*, meaning a spatial relationship between two genes. We can express these concepts in a more formal fashion:

$$G = (V, E)$$

$$\textbf{where}$$

$$V = \{\ g \mid g \in \mathit{Genes}\ \}$$

$$\textbf{and}$$

$$E = \{\ (g_1, g_2) \mid g_1 \rightleftharpoons g_2\ \wedge\ g_1, g_2 \in V\ \}$$

The resulting graph G is an *undirected, weighted* graph with a symmetric binary relation between the adjacent vertices – i.e., if g_1 is a neighbour for g_2, then g_2 is a neighbour for g_1 – while the weights on the edges provide a likelihood of physical proximity for the adjacent vertices, as a result of the normalisation phase. The neighbourhood graph can be defined as the induced *subgraph* obtainable starting from a given root vertex v, and including all vertices adjacent to v and all edges connecting such vertices, including the root vertex.

Graph Construction. The graph construction starts from one or more root genes and proceeds until all the nodes of the graph have been visited, or up to the desired "distance" from the root: a search at level 1 yields all the genes directly adjacent to the root (which is at level 0); a search at level i returns all directly adjacent genes for each gene discovered up to level $i - 1$, starting from the root.
The procedure exhibits a typical data-parallel behaviour, in which any arbitrary subset of Hi-C Reads can be processed independently from each other. This means it can be parallelised in a seamless way by processing those parts elected as main computational cores within a `ParallelFor` loop pattern, whose semantic amounts to execute in parallel the instructions inside the loop, provided they are independent from each other. Taking inspiration from the work of Hong et al. [14], the algorithm proceeds as a *Breadth First Search*. At the end of each level iteration, the parallel execution is synchronized: at this point thread-local next-level containers are processed and a partial graph is constructed with the genes discovered at the current BFS level. The definitive graph is built in batch at the end of the BFS execution.

Algorithm 1.1 *Graph Construction (pseudo-code)*

```
1  ParallelNeighboursGraph ($root$, L_MAX, NTH) {
2      Q = $\Gamma$ = Graph := $\emptyset$
3      C[NTH] = V[NTH] = E[NTH] := $\emptyset$
4      $lv$ := 0
5
6      push $root$ in Q
7      while (Q not $\emptyset$ and $lv$ < L_MAX) {
8          pop $q$ from Q
9          // find Hi−C Reads for $q$
10         ParallelFor ($r$ in Reads, NTH) {
11             if ($r$.Start in $q$[Start, Stop] and $r$.Chr == $q$.Chr)
12                 add $r$ to C[thid]
13         }
14         // find neighbour genes for $q$
15         ParallelFor ($c$ in C[thid], NTH) {
16             $intra$ := 0
17             for_each ($g$ in Genes) {
18                 if ($g$ overlaps $c$.PairedEnd)
19                     add $g$ to V[thid]
20                     add ($q$, $g$) to E[thid]
21                     $intra$ := $intra$ + 1
22             }
23             HandleIntergenicCase(Genes,$intra$)
24         }
25         // level synchronisation
26         $\Gamma$ := BuildPartialGraph(V[thid],E[thid])
27         for_each ($v$ in V[thid]) { // next level vertices
28             if (not $v$.Visited)
29                 push $v$ in Q
30         }
31         $lv$ := $lv$ + 1
32         C[thid] = V[thid] = E[thid] := $\emptyset$
33     }
34     Graph := BuildGraph($\Gamma$)
35 }
```

The Hi-C Reads exploration phase and the genes discovery have been split, in order to avoid mixing the working sets involved in the two phases. This helps minimising the cache thrashing and permits to obtain substantial performance improvements. The pseudo-code of the parallel graph construction is reported in listing 1.1: this high-level approach required some adjustment to the BFS procedure, and the introduction of new thread-local containers needed to handle concurrent write accesses to shared data structures. Specifically, C[NTH], V[NTH] and E[NTH] are used to store per-thread data, where NTH is the number of threads in use and *thid* identifies thread's own container, such that $0 \leq thid <$ NTH. Q represents our working queue that contains the genes discovered throughout the computation. L_MAX determines the maximum distance from the root that has to be reached (-1 means explore all graph). Γ is used at every level synchronisation to store partial graphs, that will be merged into a definitive graph at the very end of the graph construction process.

The algorithm starts searching for those paired Hi-C reads whose first pair fragment falls within gene's coordinates. This yields a list of connections containing those chromosome fragments where neighbour genes may be located (rows 10–13): upon this list of connections the search for neighbour genes takes place, in parallel on the set of connections (rows 15–24). Inter-genic cases are optionally handled: when the identified chromosome fragment is inter-genic (i.e., no intra-genics have been found), the corresponding genomic position is annotated on the graph as a *singularity point* (with a different graphical representation). Singularity points may be expanded to the closest proximal genes (*after* and *before* genes), searched within a predefined range (defined as a constant $\pm k$) from the paired end coordinates: when an after gene and a before gene are found (or either one of the two), an edge between the singularity point and the proximal genes is created. When all graph's levels have been explored, the graph is built in batch, by processing the partial graphs stored in Γ (row 34).

Edge Weighing. This phase encompasses the normalisation process, which is needed in order to remove systematic biases arising from sequencing and mapping. The weight assumes the role of a "confidence score" that qualifies the reliability of each contact represented on the neighbourhood graph. We recall that an edge identifies the existence of Hi-C fragments belonging to both connected genes; for each edge, a contact map (M) is constructed directly modelling the read count data at a resolution level of 1 MB. Hi-C data matrix is symmetric, thus we consider only its upper triangular part, where each point of $M_{i,j}$ denotes the intensity of the interaction between positions i and j. Using the local genomic features that describe the chromosome (fragment length, GC-content and mappability) we can set up a *generalized linear model* (GLM) with Poisson regression, with which we estimate the maximum likelihood of the model parameters.
The generalized linear model with Poisson regression has been implemented adopting the *Iteratively Weighted Least Squares* algorithm (IWLS) proposed by Nelder and Wedderburn [15] using the *GNU Scientific Library* [16]. The listing below reports a pseudo-code of the function:

Algorithm 1.2 *Edge Weighing (pseudo-code)*

```
ComputeEdgeScore($edge$, $thrsh$) {
    LenM = GccM = MapM := $\emptyset$  // cover matrices
    B := $\emptyset$
    Conv := **true**

    // populate cover matrices using genomic features
    ...

    X := ToMatrix(LenM, GccM)
    Y := BuildContactMap($edge$.Chr1, $edge$.Chr2)

    **while** (Conv) {
        ApplyLinkFunction(Y)
        B := ApplyGLM(Y, X)
        Conv := CheckConvergence($thrsh$)
    }

    $edge$.Score := $f$(B)
}
```

Listing 1.2 shows a pseudo-code of the weighing algorithm: the function `ApplyGLM` writes the best-fit parameters in vector B, which is the result of the regression: these coefficients are used to calculate the score (i.e.

the estimation of physical proximity) for the edge connecting the two genes.
The edges weighing phase is an embarrassingly parallel application: any arbitrary subset of the edges can be processed independently from each other by executing its body in a parallel loop pattern. This data-parallelism can be properly exploited to boost up performances and drastically reduce execution time, by just calling the function listed in algorithm 1.2 within Fastflow's `ParallelFor`.

3 Results

The novel makings of NuChart-II have been exploited to verify how Hi-C can be used for citogenetics studies. In particular, we focused on Philadelphia translocation, which is a specific chromosomal abnormality that is associated with chronic myelogenous leukaemia (CML). The presence of this translocation is a highly sensitive test for CML, since 95% of people with CML have this abnormality, although sometimes it occurs also in acute lymphoblastic leukaemia (ALL) and in acute myelogenous leukaemia (AML). The result of this translocation is that a fusion gene is created from the juxtaposition of the ABL1 gene on chromosome 9 (region q34) to part of the BCR (”breakpoint cluster region”) gene on chromosome 22 (region q11). This is a reciprocal translocation, creating an elongated chromosome 9 (called der 9), and a truncated chromosome 22 (called the Philadelphia chromosome). The Hi-C technique can be used to study such kind of translocations, and subsequently answer to questions such as “are this kind of chromosomal translocations occurring between nearby chromosomes?”,LMO2 just by exploiting NuChart-II.
With NuChart-II we compared the distance of some couples of genes that are known to create translocation in CML/AML. In particular, our analysis relies on data from the experiments of Lieberman-Aiden [9], which consist in 4 lines of karyotypically normal human lymphoblastoid cell line (GM06990) sequenced with Illumina Genome Analyzer, compared with 2 lines of K562 cells, an erythroleukemia cell line with an aberrant karyotype. Starting from well-established data related to the cytogenetic experiments, we tried to understand if the Hi-C technology can successfully be applied in this context, by verifying if translocations that are normally identified using Fluorescence *in situ* hybridization (FISH) can also be studied using 3C data.
We studied 5 well known couples of genes involved in translocations and we analysed their Hi-C probability contacts in physiological and diseased cells. Considering a p¡0.05 threshold for validating the presence of an edge in the graph, we see that ABL1 and BCR are distant 2 or 3 contacts in sequencing runs concerning GM06990 with HindIII as digestion enzyme (SRA:SRR027956, SRA:SRR027957, SRA:SRR027958, SRA:SRR027959), while they are in close contact in sequencing runs related to K562 with digestion enzyme HindIII (SRA:SRR027962 and SRA:SRR027963). Therefore, there is a perfect agreement between the positive and the negative presence of Hi-C contacts and FISH data.

This implies from one side that the DNA conformation in cells is effectively correlated to the disease state and also that Hi-C can be reliable in identifying these cytogenetic patterns. At the same way, AML1 and ETO are in close proximity in leukaemia cells (SRA:SRR027962 and SRA:SRR027963), while they are far 2 or 3 contacts in normal cells (SRA:SRR027956, SRA:SRR027957, SRA:SRR027958, SRA:SRR027959). Considering the translocation CBFβ-MYH11, they are distant 2 or 3 contacts in GM06990 (SRA:SRR027956, SRA:SRR027957, SRA:SRR027958, SRA:SRR027959), while are proximal in K562 (SRA:SRR027962, but not in SRA:SRR027963). We had no appreciable results for NUP214 and DEK translocation and for PML and RARα translocation, which however are more rare in this kind of disease.
These results are very important, because with the decreasing of the next-generation sequencing the Hi-C technique can be an effective diagnostic option for cytogenetic analysis, with the possibility of improving and refining the molecular biological view on the chromosomal architecture and to provide a more generalised description of the nuclear organisation.

Performance. NuChart-II has been completely re-engineered, with the aim to solve the memory issues that burdened the R prototype: both the graph construction and the edge weighing phases are bounded to the memory size required to hold the data. Concerning the graph construction, it has been tuned to properly use the memory hierarchy and fully exploit cache locality while minimising cache trashing: it can now build a chart of the whole genome within few minutes, which is a rather different execution time with respect to the R prototype.
The weighing task performs tight loops doing Floating Point arithmetic calculations on data that fit the L3 cache and can benefit from compiler optimization and vectorization. When executed in parallel, each worker thread gets a bunch of edges to work on, according to the grain size, and a reference to a static collection of data, containing the genomic features to be used within the process. During this task, memory consumption reaches a size that largely exceeds the capacity of the L3 cache in our target architecture, leading to a heavy memory traffic as the number of working threads increases. This memory overhead is anyway balanced by the heavy calculation performed by each worker thread: the implementation with FastFlow shows a quasi-linear speedup, and when compared against OpenMP and Intel TBB implementations, the recorded performance is substantially similar. Figure 1 shows some results concerning the sole weighing phase, in terms of speedup and execution time: the three frameworks reach approximately similar levels of speedup and scalability, as the number of working threads increases.

Outputs. NuChart-II provides both textual output and graphic visualization: textual and tabular outputs are useful to examine the genomic regions explored, and comprise *a)* a list of all the edges resulting from the graph construction, with the weight calculated for each edge; *b)* a list of all discovered genes, with the level (i.e. the distance from the root)

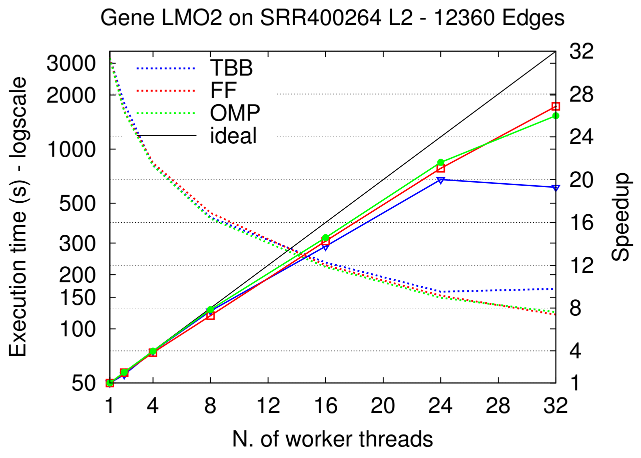

Fig. 1 – *Execution time and speedup for the weighing phase of gene* ***LMO2*** *according to Dixon et al. SRA:SRR400264 experiment: 12360 edges processed.*

where the gene has been found; *c)* a more verbose output of the execution, that reports in detail all the edges of the graph, showing all the genomic information about the two linked genes.
NuChart-II supports plotting with *iGraph* and *GraphViz*: these tools perform nicely with small-to-medium sized graphs, but cannot provide useful representation of huge graphs with more than ten thousand edges (as it happens when the deepness of the graph increases or inter-genic contacts are expanded). We are working on viable solutions to address this problem and exploit novel techniques for interactive and dynamic graph visualisation.

4 Conclusion

The added value of this software is to provide the possibility of analysing Hi-C data in a multi-omics context, by enabling the capability of mapping on the graph vertices expression data, according to a particular transcriptomics experiment, and on the edges genomic features that are known to be involved in chromosomal recombination, looping and stability.
The novel implementation of the *NuChart-II* allows the software to scale genome-wide, which is crucial to exploit its full capability for a correct analysis, interpretation and visualisation of the data produced using the Hi-C technique. We think that the possibility of having suitable descriptions of how genes are localised in the nucleus, enriched by genomic features that can characterise the way they are able to interact, can be extremely useful in the years to come for the interpretation of multi-omics data.

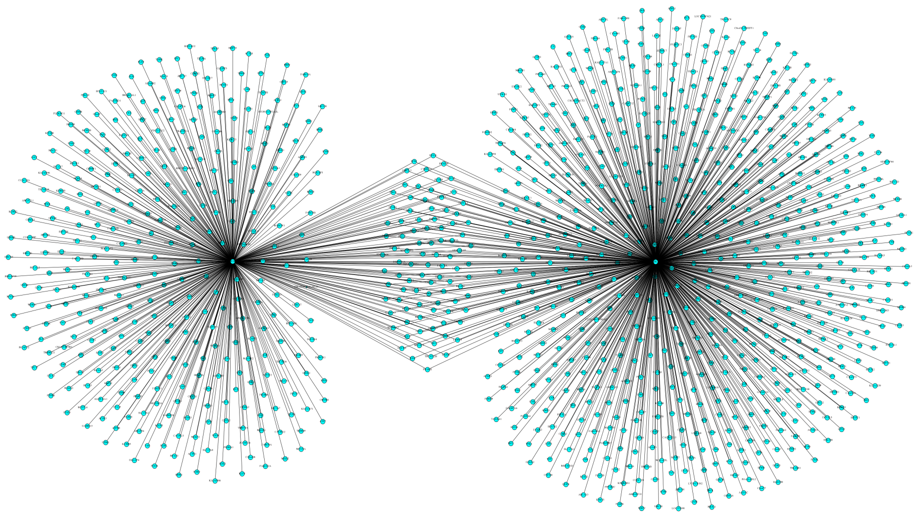

Fig. 2 – *Neighbourhood graph with genes **ABL1** and **BCR**, according to LiebermanAiden's SRA:SRR027956 experiment*

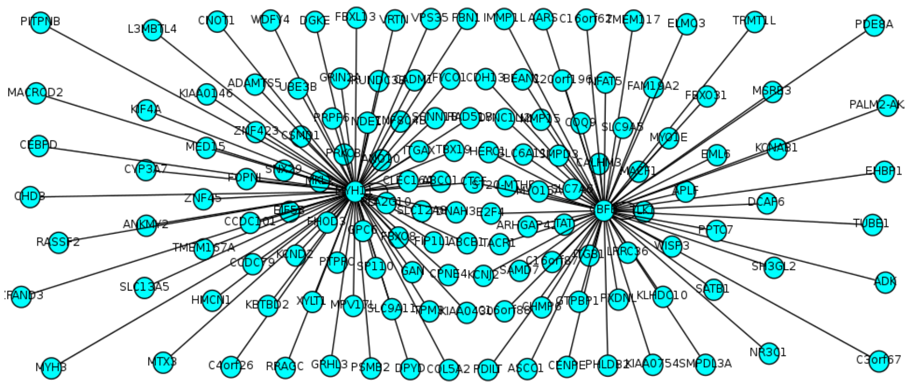

Fig. 3 – *Neighbourhood graph with genes **CBFB** and **MYH11** according to LiebermanAiden's SRA:SRR027957 experiment*

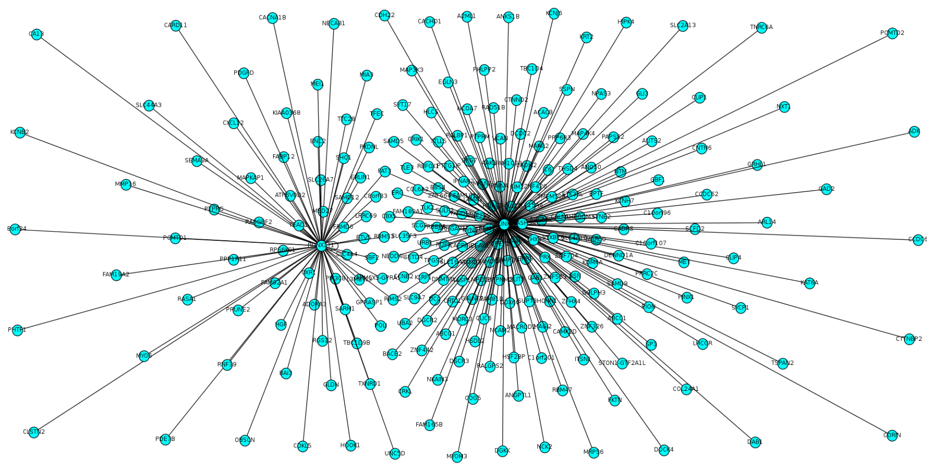

Fig. 4 – *Neighbourhood graph with genes **AML1** and **ETO** according to LiebermanAiden's SRA:SRR027963 experiment*

Acknowledgments

This work has been supported by the Flagship (PB05) InterOmics, MIUR HIRMA (RBAP11YS7K), EU MIMOMICS, EU Paraphrase (GA 288570), EU REPARA (GA 609666) projects.

References

1. Ling JQ, Hoffman AR, "Epigenetics of Long-Range Chromatin Interactions," *Pediatric Research*, 61:11R-16R, 2007.
2. Dekker J, Rippe K, Dekker M, Kleckner N, "Capturing chromosome conformation," *Science*, 295:1306-1311, 2002.
3. Simonis M, Klous P, Splinter E, Moshkin Y, Willemsen R et al., "Nuclear organization of active and inactive chromatin domains uncovered by chromosome conformation capture-on-chip (4C)," *Nature Genetics*, 38:1348- 1354, 2006.
4. Dostie J, Richmond TA, Arnaout RA, Selzer RR, Lee WL, Honan TA, Rubio ED, Krumm A, Lamb J, Nusbaum C, et al. "Chromosome conformation capture carbon copy (5C): A massively parallel solution for mapping interactions between genomic elements," *Genome Res*, 16:1299-1309, 2006.
5. Duan Z, Andronescu M, Schultz K, Lee C, Shendure J, et al. "A genome-wide 3C-method for characterizing the three-dimensional architectures of genomes," *Methods*, 58(3):277-88, 2012.
6. Merelli I, Liò P, Milanesi L, "NuChart: an R package to study gene spatial neighbourhoods with multi-omics annotations," *PLoS One*, 8(9):e75146, 2013.

7. Hu M, Deng K, Selvaraj S, Qin Z, Ren B, Liu JS, "HiCNorm: removing biases in Hi-C data via Poisson regression," *Bioinformatics*, 28(23):3131-3133, 2012.
8. Yaffe E, Tanay A, "Probabilistic modeling of Hi-C contact maps eliminates systematic biases to characterize global chromosomal architecture," *Nature genetics*, 43:1059-1065, 2011.
9. Lieberman-Aiden E, van Berkum NL, Williams L, Imakaev M, Ragoczy T, et al. "Comprehensive mapping of long-range interactions reveals folding principles of the human genome," *Science*, 326:289-293, 2009.
10. Aldinucci, M and Danelutto, M and Kilpatrick, P and M. Torquati, "Fastflow: high-level and efficient streaming on multi-core," in *Programming Multi-core and Many-core Computing Systems*, ser. Parallel and Distributed Computing, S. Pllana and F. Xhafa, Eds. Wiley, 2014, ch. 13.
11. M. Danelutto and M. Torquati, "Loop parallelism: a new skeleton perspective on data parallel patterns," in *Proc. of Intl. Euromicro PDP 2014: Parallel Distributed and network-based Processing*, M. Aldinucci, D. D'Agostino, and P. Kilpatrick, Eds. Torino, Italy: IEEE, 2014. [Online]. Available: `http://calvados.di.unipi.it/storage/paper_files/2014_ff_looppar_pdp.pdf`
12. Intel Threading Building Blocks, project site, 2013, `http://threadingbuildingblocks.org`.
13. L. Dagum and R. Menon, "OpenMP: An industry-standard api for shared-memory programming," *IEEE Comput. Sci. Eng.*, vol. 5, no. 1, pp. 46–55, Jan. 1998.
14. S. Hong, T. Oguntebi, and K. Olukotun, "Efficient parallel graph exploration on multi-core cpu and gpu," in *Proceedings of the 2011 International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 78–88. [Online]. Available: `http://dx.doi.org/10.1109/PACT.2011.14`
15. J. A. Nelder and R. W. M. Wedderburn, "Generalized linear models," *Journal of the Royal Statistical Society, Series A, General*, vol. 135, pp. 370–384, 1972.
16. M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, M. Booth, and F. Rossi, *Gnu Scientific Library: Reference Manual.* Network Theory Ltd., Feb. 2003. [Online]. Available: `http://www.worldcat.org/isbn/0954161734`