On Evaluating Graph Partitioning Algorithms for Distributed Agent Based Models on Networks

Alessia Antelmi[1], Gennaro Cordasco[2], Carmine Spagnuolo[1(✉)], and Luca Vicidomini[1]

[1] Dipartimento di Informatica, Università degli Studi di Salerno, Fisciano, Italy
a.antelmi@studenti.unisa.it, {cspagnuolo,lvicidomini}@unisa.it
[2] Dipartimento di Psicologia, Seconda Università degli Studi di Napoli, Caserta, Italy
gennaro.cordasco@unina2.it

Abstract. Graph Partitioning is a key challenge problem with application in many scientific and technological fields. The problem is very well studied with a rich literature and is known to be NP-hard. Several heuristic solutions, which follow diverse approaches, have been proposed, they are based on different initial assumptions that make them difficult to compare. An analytical comparison was performed based on an Implementation Challenge [3], however being a multi-objective problem (two opposing goals are for instance load balancing and edge-cut size), the results are difficult to compare and it is hard to foresee what can be the impact of one solution, instead of another, in a real scenario. In this paper we analyze the problem in a real context: the development of a distributed agent-based simulation model on a network field (which for instance can model social interactions).

We present an extensive evaluation of the most efficient and effective solutions for the balanced k-way partitioning problem. We evaluate several strategies both analytically and on real distributed simulation settings (D-Mason). Results show that, a good partitioning strategy strongly influences the performances of the distributed simulation environment. Moreover, we show that there is a strong correlation between the edge-cut size and the real performances. Analyzing the results in details we were also able to discover the parameters that need to be optimized for best performances on networks in ABMs.

Keywords: Agent-Based Simulation Models · Graph partitioning · D-Mason · Parallel computing · Distributed systems · High performance computing

1 Introduction

Networks are everywhere. Complex interactions between different entities play a sensible role in modeling the behavior of both society and natural world. Such

S. Hunold et al. (Eds.): Euro-Par 2015 Workshops, LNCS 9523, pp. 367–378, 2015.
DOI: 10.1007/978-3-319-27308-2_30

networks – which comprises World Wide Web, metabolic networks, neural networks, communication and collaboration networks, and social networks – are the subject of a growing number of research efforts. Indeed, many interesting phenomena are structured as networks (i.e., sets of entities joined in pairs by lines representing relations or interactions).

The study of networked phenomena has experienced a particular surge of interest due to the increasing availability of massive data about the static topology of real networks as well as the dynamic behavior generated by the interactions among network entities. The analysis of real networks topologies has revealed several interesting structural features, like the small-world phenomena as well as the power-law degree distribution [10], which appear in several real network and can be extremely helpful for the design of artificial networks. On the other hand, understanding the dynamic behavior generated by complex network systems is extremely hard. Networks are often characterized by a dynamic feedback effect which is hard to predict analytically.

More generally, complex systems require innovative study methodologies. In this regard, in recent years, the two branches of the classical sciences, theoretical and experimental, have merged in the computational sciences where scientists design mathematical models and perform computational simulations of complex phenomena on real systems which are too complex to be studied analytically on theoretical grounds as well as too risky/expensive to be tested experimentally [23]. In particular, Agent-Based simulation Models (ABMs) have spread in many fields, from social sciences to the life sciences, from economics to artificial intelligence. Successes of the computational sciences have led to an increased demand for computation-intensive software implementations, in order to improve the performance of ABMs in terms of both size (number of agents) and quality (complexity of interactions). Such an amount of computing power can only be achieved by parallel computing (indeed, serial-processing speed is reaching a physical limit [22]). However, exploiting parallel systems is not an easy task; many parallel applications fail to leverage on the hardware parallelism and experience scalability limits.

The computer science community has responded to the need for tools and platforms that can help the development and testing of new models in each specific field by providing tools, libraries and frameworks that speed up and make easier the task of developing and running parallel ABMs for complex phenomena. For instance, D-Mason [5,6,26,28] is a parallel version of the Mason [4,16,17] library for running ABMs on distributed systems. D-Mason adopts a framework-level parallelization mechanism approach, which allows the harnessing of computational power of a parallel environment and, at the same time, hides the details of the architecture so that users, even with limited knowledge of parallel computer programming, can easily develop and run simulation models.

Mason like several other ABMs systems provides one or more fields to represent the space where the agents lie and interact with the other agents. A field is a specific data structure relating various objects (agents) or values together. With more details, Mason provides a number of built-in fields, such as 2D/3D

geometric discrete and continuous spaces plus a network field typically used to model social interactions. Currently, D-Mason allows modellers to parallelize simulation based on geometric fields. It adopts a space partitioning approach [9] which allowed the balancing of workload among the resources involved for the computation with a limited amount of communication overhead.

The space partitioning approach described above is devoted to decomposing ABMs based on geometric fields. On the other hand, when agents lie and/or interact on a network [1] – where the network can represent social, geographical or even a semantic space – a different approach is needed. The problem is to (dynamically) partition the network into a fixed set of sub-networks in such a way that: (i) the components have roughly the same size and (ii) both the number of connections and the communication volume between vertices belonging to different components are minimized.

1.1 Our Results

In this paper we provide an extensive evaluation of the most efficient and effective solutions for the problem defined above, which is well known in literature as the graph partitioning problem. We will evaluate several algorithms both analytically and on a real distributed simulation settings. Results shows that a good partitioning strategy strongly influences the performances of the distributed simulation environment. Analyzing the results in detail we were also able to discover the parameters that need to be optimized for the best performances on networks in ABMs.

2 The Graph Partitioning Problem

Finding good network partitions is a well-studied problem in graph theory [2]. Several are the problems that motivate the study of this problem. They range from computer science problems like integrated circuit design, VLSI circuits, domain decomposition for parallel computing, image segmentation, data mining [13,15], etc. to other problems raised by physicists, biologists, and applied mathematicians, with applications to social and biological networks (community structure detection, structuring cellular networks and matrix decomposition [12,18]).

The most common formulation of the balanced graph partitioning problem is the following:

Balanced k-way partitioning (G, k, ϵ).

Instance: A graph $G = (V, E)$, an integer $k > 1$ (number of components) and a rational ϵ (imbalance factor).

Problem: Compute a partition Π of V into k pairwise disjoint subsets (components) $V_1, V_2, \ldots, V_k$ of size at most $(1 + \epsilon)\lceil |V|/k \rceil$, while minimizing the size of the edge-cut $\sum_{i<j} |E_{ij}|$, where $E_{ij} = \{\{u, v\} \in E : u \in V_i, v \in V_j\} \subseteq E$.

This problem has been extensively studied (see [3] for a comprehensive presentation) and is known to be NP-hard [11].

Being a hard problem, exact solutions are found in reasonable time only for small graphs. However the applications of this problem require to partition much larger graphs and so several heuristic solutions have been proposed.

The graph partitioning problem was faced using several approaches. Two version of this problem have been considered: the former takes into account the coordinate information in the space of the vertices (this is common in graphs describing a physical domain) while, in the latter problem, vertices are coordinate free. In this paper we discuss the coordinate free problem which better fits ABMs' domain.

The graph partitioning coordinate free problem requires combinatorial heuristics to partition them. For instance, considering the simplest version of the partitioning problem (2-way partitioning), that is find a bisection of the graph $G = (V, E)$ that minimize the size of the cut. A really simple solution of the problem uses the breadth first search (BFS) visit of the graph to generate a subgraph $T = (V, E' \subseteq E)$ of G also called a BFS tree. Given the subgraph T, is possible to find a cut to generate two disjoint subnetwork N_1 and N_2 such that (i) $N_1 \cup N_2 = V$ and (ii) $|N_1| \approx |N_2|$. The fact that T has been built using the BFS ensures that the size of the edge-cut is bounded.

This solution, which works well for planar graphs, is not efficient for complex graph. An additional approach is the Kernighan-Lin (KL) algorithm [15] that, starting with two sets N_1 and N_2 (describing a partition of V), greedily improves the quality of the partitioning by iteratively swapping vertices among the two sets. This solution converges to the global optimum if the initial partition is fairly good. Other approaches are the Spectral partitioning [21] and the Multilevel Approach [14]. We will focus on the most promising techniques that either use a multilevel approach or a distributed algorithm that exploits a local search approach.

METIS is a graph multilevel k-way partitioning suite developed in the Karypis lab of University of Minnesota. Shortly, METIS comprises three phases: during the coarsening phase the vertices are collapsed in order to decrease the size of the initial graph G. Consequently, starting from $G = G_0$ a sequence of graphs $G_0, G_1, \ldots, G_\ell$ is generated. Then a k-way partitioning is performed on the smallest graph G_ℓ. Then, during the uncoarsening phase the partitioning is refined, using a variant of the KL algorithm, and is projected to larger graphs on the sequence.

KaHIP (Karlsruhe High Quality Partitioning) is a suite of graph partitioning algorithms. The suite comprises two main algorithms *KaFFPa* (Karlsruhe Fast Flow Partitioner) [20], which is a multilevel graph partitioning algorithm, and *KaFFPaE* (KaFFPa Evolutionary) that uses an evolutionary algorithm approach. In this paper we analyze KaFFPa. KaFFPa, like METIS, uses the multilevel graph partitioning approach but it uses a different strategy for the uncoarsening phase of the algorithm which exploits a local search method instead of the KL approach.

Ja-be-Ja [19] exploits a distributed computing approach. It uses a local search technique (simulated annealing), to find a good partitions of the graph minimizing the edge-cut size. The energy of the system is measured by counting the

number of edges that have endpoints in different components. Ja-be-Ja starts with a random balanced partitioning and then it iteratively applies the local search heuristic to obtain a configuration having a lower energy state (edge-cut size). The size of the initial components is preserved since Ja-be-Ja allows only the swapping of vertices among two components.

3 Experiment Setting

We report on simulation experiments that compare five k-way partitioning algorithms on several networks, taken from [27]. The data sets we considered include networks having different structural features (see Table 1). For each network, partitions into $k = 2, 4, 8, 16, 32$ and 64 components have been considered.

We compare the analytical results obtained (i.e., size of the edge-cut, number of communication channels required and imbalance) by each algorithm with the real performances (overall simulation time) in an ABM scenario.

Table 1. Networks.

Name	# of nodes	# of edges	Avg deg.	Max deg.	Triangles	Clust. Coeff.	Modul.
uk	4824	6837	2.83	3	1	0	0.7934
data	2851	15093	10.59	17	24442	0.485719	0.7596
4elt	15606	45878	5.88	10	30269	0.40765	0.6274
cti	16840	48232	5.73	6	362	0.004895	0.9063
t60k	60005	89440	2.98	3	0	0	0.5419
wing	62032	121544	3.92	4	6685	0.055595	0.5403
finan512	74752	261120	6.99	54	211456	0.503401	0.6469
fe_ocean	143437	409593	5.71	6	0	0	0.5947
powergrid	4941	6594	2.67	19	651	0.1065	0.6105

Simulation Environment. To evaluate real performances we developed a toy distributed SIR (Susceptible, Infected, and Removed) simulation, where, for each simulation step, each agent (a vertex of the network) has to communicate with its neighbors. The SIR simulation has been developed on top of D-Mason, exploiting the novel communication strategy which realizes a Publish/Subscribe paradigm through a layer based on the MPI standard [7,8]. Simulations have been performed on cluster of eight computer nodes, each equipped as follows:

- Hardware:
  - CPUs: 2 x Intel(R) Xeon(R) CPU E5-2680 @ 2.70GHz (#core 16, #threads 32)
  - RAM: 256 GB
  - Network: adapters Intel Corporation I350 Gigabit
- Software:
  - Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic x86_64)

- Java JDK 1.6.25
- OpenMPI 1.7.4 (feature version of Feb 5, 2014).

Simulation results, on k-way partitioning, have been obtained using k logical processors (one logical processor per component). We notice that, when the simulation is distributed, the communication between agents in the same component is much faster than the communication between agents belonging to different components. On the other hand, balancing is important because the simulation is synchronized and evolves with the speed of the slowest component.

The Competing Algorithms. We have analyzed five algorithms, briefly discussed in Sect. 2:

- Multilevel approach:
  - **METIS:** (cf. Sect. 2);
  - **METIS Relaxed:** this version of the METIS algorithm uses a relaxed version of the balancing constraint (i.e., a larger value of the parameter ϵ), in order to improve on other parameters (like the edge-cut size);
  - **KaFFPa:** (cf. Sect. 2);
- Distributed Computing Approach:
  - **Ja-be-Ja:** (cf. Sect. 2). Unfortunately, we were not able to find a real implementation of the algorithm. We used an implementation available on the public Ja-be-Ja GitHub repository [24]. This implementation is not truly distributed but is simulated through the use of the Java library GraphChi [25], that enables modellers to simulate a distributed computation on multi-cores machines. Clearly the computational efficiency of this implementation is limited and, for this reason, we could only run 100 iterations of the algorithm for each test setting. We assume that the poor results of the algorithm (cf. Sect. 4) are, at least, partially due to the small number of iteration used in our tests. In order to better evaluate the real performances of the algorithm, a real distributed implementation of the Ja-be-Ja algorithm is needed.
- **Random:** This algorithm assigns each vertex to a random component. It always provides an optimal balancing. We use this algorithm as baseline in our comparisons.

Performance Metrics. Let $G = (V, E)$ the analyzed network and let $\Pi = (V_1, V_2, \ldots, V_k)$ the partition provided by a given algorithm, we evaluate algorithms' performances using the following metrics:

- Edge-cut size (W), the total number of edges having their incident vertices in different components;
- Number of communication channels (E), two components U_1 and U_2 requires a communication channel when $\exists v_1 \in U_1,\ v_2 \in U_2$ such that $(v_1, v_2) \in E$. In other words, we are counting the number of edges in the supergraph S_G obtained by clustering the nodes of each component in a single node. We notice that this unconventional metric is motivated by our specific distributed ABMs

scenario. In our simulation environment, a communication channel, between two components U_1 and U_2, is established when at least two vertices (agents) $u_1 \in U_1$ and $u_2 \in U_2$ share an edge. Thereafter, the same communication channel is used for every communication between U_1 and U_2, consequently, these additional communications have less impact on system performances;
– Imbalance (I), the minimum value of ϵ such that each component has size at most $(1+\epsilon)\lceil |V|/k \rceil$.

Moreover, we evaluate the real performances of each strategy by measuring the overall simulation time (T) to perform 10 simulation steps on the distributed SIR simulation.

Summarizing our experiments compares the performances (both analytically and on a real setting) of 5 k-way partitioning algorithms ($A \in$ {METIS, METIS Relaxed, KaFFPa, Ja-be-Ja and Random}) with $k \in \{2, 4, 8, 16, 32, 64\}$ on 9 networks ($N \in$ {uk, data, 4elt, cti, t60k, wing, finan512, fe_ocean, powergrid}). Overall we performed $5 \times 6 \times 9 = 270$ tests.

4 Results

4.1 Analytical Results

Figures 1, 2 and 3 depict the analytical results. For each plot the networks appear along the X-axis, while the values of the measured parameter appear along the Y-axis. We present the results only for $k \in \{4, 64\}$ because of space limitations; results for the other values of k exhibit similar behaviors.

Analyzing the results from Figs. 1 and 2 we notice that the performances of the multilevel approach algorithms are comparable both in terms of edge-cut size and number of communication channels. Ja-be-Ja performances are a bit worse (this is probably due to the small number of iteration used in our tests as observed in Sect. 3) but always better than the random strategy.

Results on imbalance are fluctuating (see Fig. 3). In general all the algorithms provide a quite balanced partition. Apart from the random strategy that by construction provides the optimal solution, no strategy dominates the others.

4.2 Real Setting Results

Figure 4 reports on the results obtained in the real simulation setting. The results are consistent with the analytical ones, in terms of both edge-cut size and number of communication channels, although the gaps are amplified. The results thus confirm that the choice of partitioning strategy has a significant impact on performance in a real scenario.

In order to better understand how the metrics evolves according to k, Figure 5 depicts four plots which describes, for each algorithm, the growth of the Edge-cut size (top-left), the Imbalance (top-right), the number of communication channels (bottom-left) and the Simulation time on the f_ocean network as function of the parameter k (X-axis).

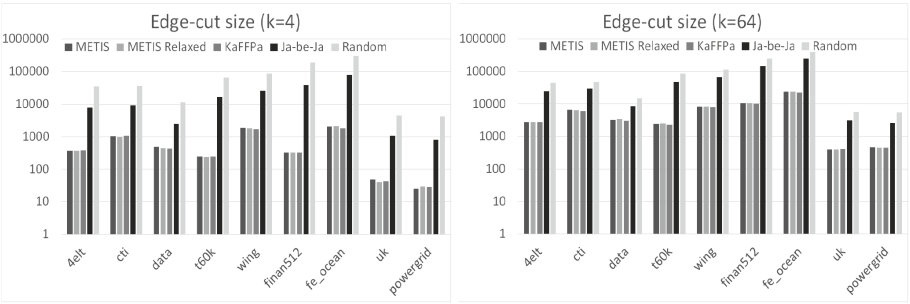

Fig. 1. Edge-cut size (W) comparison:(left) $k = 4$, (right) $k = 64$. Y-axes appear in log scale.

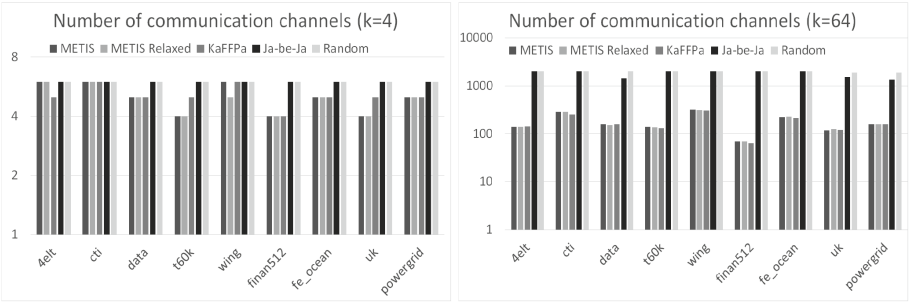

Fig. 2. Number of communication channels (E) comparison:(left) $k = 4$, (right) $k = 64$. Y-axes appear in log scale.

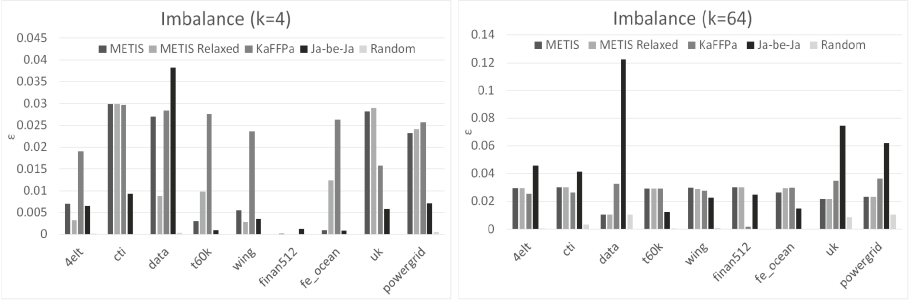

Fig. 3. Imbalance (I) comparison: (left) $k = 4$, (right) $k = 64$.

4.3 Correlation Between Analytical and Real Setting Results

Analyzing the results from Figs. 1–4, we observe that the performances of the distributed simulations are influenced by the analytical metrics. In order to better evaluate the correlation between the overall simulation times and the performances of the algorithm (measured considering the edge-cut size, the number of

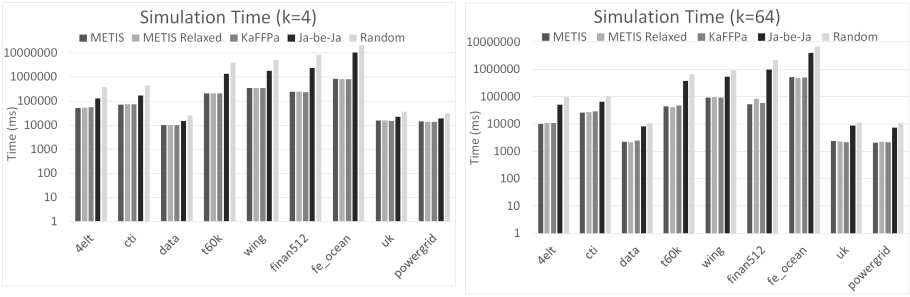

Fig. 4. Simulation time (T) comparison:(left) $k = 4$, (right) $k = 64$. Y-axes appear in log scale.

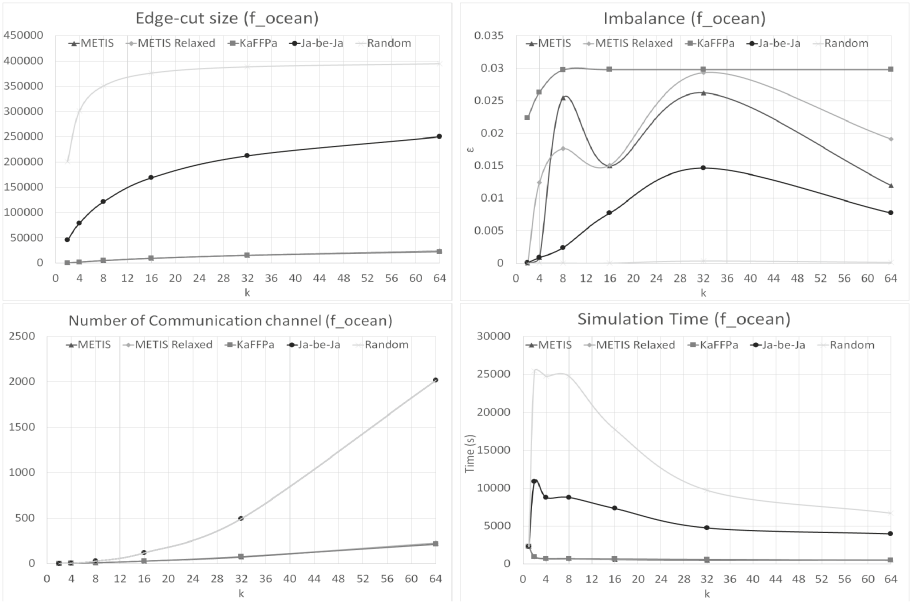

Fig. 5. Edge-cut size (top-left), Imbalance (top-right), Number of communication channels (bottom-left) and Simulation Time(bottom-right) on the f_ocean network, $k \in \{2, 4, 8, 16, 32, 64\}$.

communication channels and the imbalance), we measured the correlation using a statistical metric: the Pearson product-moment Correlation Coefficient (PCC). PCC is one of the measures of correlation which quantifies the strength as well as direction of the relationship between two variables. The correlation coefficient ranges from -1 (strong negative correlation) to 1 (strong positive correlation). A value of 0 implies that there is no correlation between the variables. We computed the correlation PCC between simulation time (T) and the three analytical metrics (W, E and I), with all the considered value of the parameter k.

In particular, we considered four variables that are parametrized by the class N of Networks ($N \in$ {uk, data, 4elt, cti, t60k, wing, finan512, fe_ocean, powergrid}), the considered algorithm ($A \in$ {METIS, METIS Relaxed, KaFFPa, Ja-be-Ja and Random}), and the parameter $k \in \{2, 4, 8, 16, 32, 64\}$. The variable $T(N, A, k)$ denotes the Simulation time; the variable $W(N, A, k)$ denotes Edge-cut size; $E(N, A, k)$ denotes the Number of communication channels; finally the variable $I(N, A, k)$ denotes the Maximum Imbalance. Table 2 presents the correlation values obtained.

We observed that:

- there is a strong positive correlation between simulation time and edge-cut size (the PCC always over 0.92);
- there is a weak/moderate positive correlation between simulation time and the number of communication channels[1] (the PCC ranges between 0.22 and 0.4). Moreover this correlation seems to be increasing in k;
- there is a weak negative correlation between simulation time and imbalance (the PCC ranges between -0.22 and -0.32).

This final result is counterintuitive: theoretically, the greater the imbalance, the larger the simulation time should be and this should lead to a positive correlation. The key observation is that a small amount of imbalance has a limited impact on the simulation time but can be extremely helpful for reducing both the edge-cut size and the number of communication channels, which seems to have a sensible payoff in terms of real performances.

Table 2. Correlation between analytical and real setting results.

	k					
	2	4	8	16	32	64
$r(T,\ W)$	**0.9256**	**0.9392**	**0.9431**	**0.9424**	**0.9473**	**0.9474**
$r(T,\ E)$	N.A.	0.2265	0.3094	0.3349	0.3509	0.3922
$r(T,\ I)$	-0.2244	-0.2750	-0.2903	-0.3188	-0.2971	-0.3025

5 Conclusion

We considered the problem of partitioning a network into k balanced components such that the number of edges that cross the boundaries of components is minimized. We evaluated, both analytically and on a real distributed ABM scenario, the performances of 5 heuristic approaches, which, to the best of our knowledge, are the current state-of-the-art on the problem. Experimental results show that the choice of the partitioning strategy strongly influence the performance a real distributed environment. Moreover analytical results (the edge-cut

[1] The correlation between $T(N, A, 2)$ and $E(N, A, 2)$ cannot be computed, since for $k = 2$ all the partitioning strategy require exactly 1 communication channel and so $E(N, A, 2)$ has standard deviation equal to 0.

size in particular) correlate with the overall simulation time in a real setting. On the other hand, according to our results, the quality of the balance among components does not relate to the real performances on the field. Likely, this result is due to the fact that we analyzed of very small imbalance range. As a future work, we plan to investigate heuristics which allow identifying more efficient partitionings, at the expense of a minor balance.

References

1. Alam, S., Geller, A.: Networks in agent-based social simulation. In: Heppenstall, A.J., Crooks, A.T., See, L.M., Batty, M. (eds.) Agent-Based Models of Geographical Systems, pp. 199–216. Springer, Netherlands (2012). http://dx.doi.org/10.1007/978-90-481-8927-4_11
2. Alpert, C.J., Kahng, A.B.: Recent directions in netlist partitioning: a survey. Integr. VLSI J. **19**, 1–81 (1995)
3. Bader, D.A., Meyerhenke, H., Sanders, P., Wagner, D. (eds.): Graph Partitioning and Graph Clustering - 10th DIMACS Implementation Challenge Workshop, Georgia Institute of Technology, Atlanta, GA, USA, February 13–14, 2012. Proceedings, Contemporary Mathematics, vol. 588, American Mathematical Society (2013). http://dx.doi.org/10.1090/conm/588
4. Balan, G.C., Cioffi-Revilla, C., Luke, S., Panait, L., Paus, S.: MASON: a java multi-agent simulation library. In: Proceedings of the Agent 2003 Conference (2003)
5. Cordasco, G., De Chiara, R., Mancuso, A., Mazzeo, D., Scarano, V., Spagnuolo, C.: A Framework for distributing Agent-based simulations. In: 9th International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Platforms (2011)
6. Cordasco, G., De Chiara, R., Mancuso, A., Mazzeo, D., Scarano, V., Spagnuolo, C.: Bringing together efficiency and effectiveness in distributed simulations: the experience with D-MASON. SIMULATION Trans. Soc. Model. Simul. Int. **89**(10), 1236–1253 (2013)
7. Cordasco, G., Mancuso, A., Milone, F., Spagnuolo, C.: Communication strategies in distributed agent-based simulations: the experience with D-Mason. In: an Mey, D., Alexander, M., Bientinesi, P., Cannataro, M., Clauss, C., Costan, A., Kecskemeti, G., Morin, C., Ricci, L., Sahuquillo, J., Schulz, M., Scarano, V., Scott, S.L., Weidendorfer, J. (eds.) Euro-Par 2013. LNCS, vol. 8374, pp. 533–543. Springer, Heidelberg (2014)
8. Cordasco, G., Milone, F., Spagnuolo, C., Vicidomini, L.: Exploiting D-Mason on parallel platforms: a novel communication strategy. In: Proceedings of the 2nd Workshop on Parallel and Distributed Agent-Based Simulations (PADABS), Euro-Par 2014 (2014)
9. Cosenza, B., Cordasco, G., De Chiara, R., Scarano, V.: Distributed load balancing for parallel agent-based simulations. In: Proceedings of the 19th International Euromicro Conference on Parallel, Distributed, and Network-Based Processing, (PDP 2011), pp. 62–69 (2011)
10. Easley, D., Kleinberg, J.: Networks, Crowds, and Markets: Reasoning About a Highly Connected World. Cambridge University Press, New York (2010)
11. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman & Co, New York (1990)

12. Gupta, A.: Graph partitioning based sparse matrix orderings for interior-point algorithms. IBM Thomas J, Watson Research Division (1996)
13. Karypis, G., Aggarwal, R., Kumar, V., Shekhar, S.: Multilevel hypergraph partitioning: applications in VLSI domain. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **7**(1), 69–79 (1999)
14. Karypis, G., Kumar, V.: Multilevel k-way partitioning scheme for irregular graphs. J. Parallel Distrib. Comput. **48**(1), 96–129 (1998)
15. Kernighan, B., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell Syst. Tech. J. **49**(2), 291–307 (1970)
16. Luke, S., Cioffi-revilla, C., Panait, L., Sullivan, K.: MASON: a new multi-agent simulation toolkit. In: Proceedings of the 2004 SwarmFest Workshop (2004)
17. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G.: MASON: a multiagent simulation environment. Simulation 81(7), 517–527 (2005). http://dx.doi.org/10.1177/0037549705058073
18. Newman, M.E.J.: Modularity and community structure in networks. Proc. Nat. Acad. Sci. (PNAS) **103**(23), 8577–8582 (2006)
19. Rahimian, F., Payberah, A.H., Girdzijauskas, S., Jelasity, M., Haridi, S.: JA-BE-JA: a distributed algorithm for balanced graph partitioning. In: 7th International Conference on Self-Adaptive and SelfOrganizing Systems. IEEE (2013)
20. Sanders, P., Schulz, C.: Think locally, act globally: highly balanced graph partitioning. In: Bonifaci, V., Demetrescu, C., Marchetti-Spaccamela, A. (eds.) SEA 2013. LNCS, vol. 7933, pp. 164–175. Springer, Heidelberg (2013)
21. Spielmat, D., Teng, S.H.: Spectral partitioning works: planar graphs and finite element meshes. In: Proceedings 37th Annual Symposium on Foundations of Computer Science, pp. 96–105, October 1996
22. Sutter, H.: The free lunch is over: a fundamental turn toward concurrency in software. Dr. Dobb's J. **30**(3), 202–210 (2005)
23. Tekin, E., Sabuncuoglu, I.: Simulation optimization: a comprehensive review on theory and applications. IIE Trans. **36**(11), 1067–1081 (2004)
24. https://github.com/fatemehr/jabeja: Ja-be-Ja GitHub repository, Accessed on May 2015
25. https://github.com/GraphChi/graphchi-java: GraphChi's Java version, Accessed on May 2015
26. https://github.com/isislab-unisa/dmason: D-MASON Official GitHub Repository, Accessed on June 2015
27. http://staffweb.cms.gre.ac.uk/~wc06/partition/: The Graph Partitioning Archive, Accessed on May 2015
28. http://www.dmason.org: D-MASON Official Website, Accessed on May 2015