

ON THE DEVELOPMENT OF A REDUCED APPROACH TO THE RECURSIVE COORDINATE BISECTION IN THE CONTEXT OF PARALLEL SIMULATIONS OF THE MATERIAL POINT METHOD

João G. C. S. Duarte¹, Adeildo S. R. Júnior¹, Diogo T. Cintra¹, Ricardo G. Borges²

¹*Laboratory of Scientific Computing and Visualization, Center of Technology, Federal University of Alagoas
Av. Lourival Melo Mota, S/N, 57072-900, Maceió/Alagoas, Brazil
joao.duarte@lccv.ufal.br, adramos@lccv.ufal.br, diogotc@lccv.ufal.br*

²*Project Manager, CENPES, Petrobrás
Av. Horácio Macedo, 950, 21941-915, Rio de Janeiro/Rio de Janeiro, Brazil
garske@petrobras.com.br*

Abstract. This paper aims to develop a reduced approach to the Recursive Coordinate Bisection (RCB) in the context of parallel simulations of the Material Point Method (MPM). The simulation environment considers the usage of parallel computing, a powerful tool that allows us to increase computational performance and to reduce the simulation time. In this environment, a set of computers provides processors, allocates processes and communicates data among them. MPM can be used to simulate various engineering problems, including those involving submarine landslides, installation of torpedo anchors and dynamical analysis of structures. Numerical simulations of MPM can be computationally intensive, especially for those containing a large number of particles. In this work, the geometrical information of particles inside each solution sub-domain is reduced into statistical information of centroid and standard deviation in order to avoid excessive data communication during simulations. The root process simulates the entire domain geometry from the collected data for each sub-domain. The partitioning algorithm and statistical strategy were developed using the C++ language and the Message Passing Interface (MPI) library. The MPI library allows the exchange of data among the several processors in a distributed memory system. The partitioning quality obtained in this method depending on input parameters such as the number of processes, the mesh discretization for each sub-domain and the number of points generated in the statistical domain. Numerical experiments were conducted in order to measure a load balance metric and the processing time (speed up). The obtained values for the parametric study were compared to an analytical reference of balance. The results show that the partitioning method generates better load balance values depending on the combination of the input parameters.

Keywords: Parallel Computing, Message Passing Interface, Material Point Method, Recursive Coordinate Bisection, Computational Statistics

1 Introduction

With the advance of computing power and data processing, large-scale problem simulations, from wide different engineering fields, become possible and recurrent. To simulate these problems, numerical methods with good computational performance are necessary to reproduce suitable and reliable results. In this sense, to perform the simulation of the dynamics of many particles, the Material Point Method (MPM) can be used. MPM allows the discretization of a continuous environment in a finite number of material points, such as dynamical analysis of structures, simulation of submarine landslides, and installation of torpedo anchors. In a recent work, Kafaji K. J. al Kafaji [1] defines MPM combines the best aspects of both Lagrangian and Eulerian formulations and avoids as much as possible the shortcomings of them. However, aiming to deal with a large number of particles present in simulations, the MPM requires domain partitioning techniques to improve the computational performance of the method and to ensure load balancing between sub-domains created.

Domain partitioning techniques provide a parallel simulation scenario, as Cintra et al. [2] approaches. Within these partitioning techniques, some operate by geometrically dividing the original domain of points into sub-domains, being widely used in problems with points referenced in the Cartesian system. Each sub-domain generated contains approximately the same number of points and is assigned to a processor, which will perform the calculations related to the respective points it contains. As computational numerical methods, domain partitioning techniques provide load unbalances in the simulation. No matter how small, load unbalances are inherent in the way they are implemented. They tend to be greater in dynamic domains, and according to Eibl and Rde [3] in his work, make the simulation slower due to the loss of computational performance. To overcome this problem, the domain partitioning must happen each simulation step, and the new sub-domains assigned to the processors, replacing the oldest sub-domains.

This paper is structured as follows. In section 2, we describe the work approach, arguing about the applied methodology. Section 3 describes the techniques applied, implementation procedures for each technique, and the tools used. The results obtained from the study are shown and discussed in the section 4. The conclusions are explained in section 5.

2 Our approach

The choice of the appropriate domain partitioning technique for the simulation is an extremely important factor to obtain reliable results and to achieve the best performance. Furthermore, it is responsible for geometrically delimiting the sub-domains. To establish the connection between them, some computational library of parallel computing is necessary. Our approach is based on two types of geometric domain partitioning technique to generate the sub-domains (ranks) used in all simulations: 1) the Recursive Coordinate Bisection (RCB) and 2) the Reduced Recursive Coordinate Bisection (RRCB). To measure the quality of domain partitioning techniques, we adopted the load balance and speedup comparative parameters to investigate which technique produces the best results considering a dynamic domain with thousands of particles. The speedup parameter is measured by comparing the execution time of the serial simulation with the parallel simulation, both of the same model.

3 Domain partitioning techniques and implementation procedure

In a dynamic problem simulation, the material points on the background mesh travel from one subdomain to another. This fact implies data traffic between the processors being used in the simulation. The exchange of data is an important factor for the performance of parallel simulations, since the greater the number of data traveling between the processors, larger the communication between them, which increases the simulation time of the problem in question. Therefore, the techniques chosen for this work tend to reduce the communication area between the sub-domains, increasing the computational performance of the algorithm. To establish this communication between the ranks of the application, the computer library Message Passing Interface (MPI) was adopted throughout the study, and the C++ language was used in the implementation of the computational code. MPI establishes a distributed memory system, as shown by Pacheco in his book Pacheco [4]. Thus, communication between processors is the responsibility of the programmer, which increases the complexity of the implementation, but prevents Data Races between processes.

In the context of our application, an initial parallel decomposition of the original domain is performed without taking into account the load balance in the first simulation step, to create the sub-domains of the application. When the partitioning algorithm is called during the execution of the program, it renews the previously created sub-domains and the load balance is restored. The root process, or rank 0, it's responsible for managing the partitioning of the entire simulation domain. Therefore, he receives the information of interest from the other sub-domains and together with your information performs the previously chosen partitioning technique; RRCB or RCB, in this work. With the end of the partitioning algorithm, rank 0 sends the boundary boxes to the other processes and simulation continues. In the next simulation step, the partitioning happens again since the domain is dynamic, but without the need for an initial parallel partitioning.

RRCB, as its name suggests, is derived from RCB. In this sense, both partition the domain similarly. Having the number of processors used in the simulation, the partitioning algorithm performs its operations in the root process. Partitioning starts when the Cartesian amplitudes of the domain are analyzed. When discovering which Cartesian direction (X or Y, to 2D problems) has the greatest amplitude, the algorithm saves the midpoint of that direction and performs a perpendicular cut to that axis. From that, two new sub-domains appear. The procedure

is repeated recursively until the number of sub-domains created is equal to the number of processors used in the simulation. The difference between the RRCB and RCB methods is in the information of interest that the zero process receives at the beginning of the partitioning algorithm.

While in the RCB the root process receives the points from the other sub-domains of the application as information of interest and together with its points performs the partitioning of that domain (which is the original domain), in the RRCB the root process receives the mean and standard deviation of the geometric position of the points of the other sub-domains as information of interest and together with these variables of their points, generates a statistical domain (which is different from the original domain). This fact implies in less transfer of data between the processors since not all points are sent, but the statistical information that represents a set of points.

Given a real domain (the simulation domain), it's possible to divide him with a auxiliary mesh. In each cell of the mesh, there will be a certain number of points from the real domain. In RRCB, the points belonging to a cell, are reduced to statistical information of mean and standard deviation of the geometrical information of them, in the X and Y directions. The more refined the mesh is, the more information of mean and standard deviation is generated. Such information is sent to root process. In it, new points are generated in a pseudo-random way from an internal function of the C++ language to create a statistical domain. This new statistical domain will be the target of the partitioning algorithm, thus replacing the original domain. When the partitioning process is finished, the boundary boxes are sent to the other processes. The pseudo-random generation of points ensures that the X and Y coordinates do not become random variables, which would drastically change the way of dealing with this problem. Therefore, the points will always be generated in the same Cartesian positions. Fig. 1 shows the flow chart of the RCB and RRCB. In the RCB, steps 2, 3, 4, 5, 6, and 7 are executed in the root process, while in the RRCB are the 3, 4, 5, 6, 7, and 8 steps.

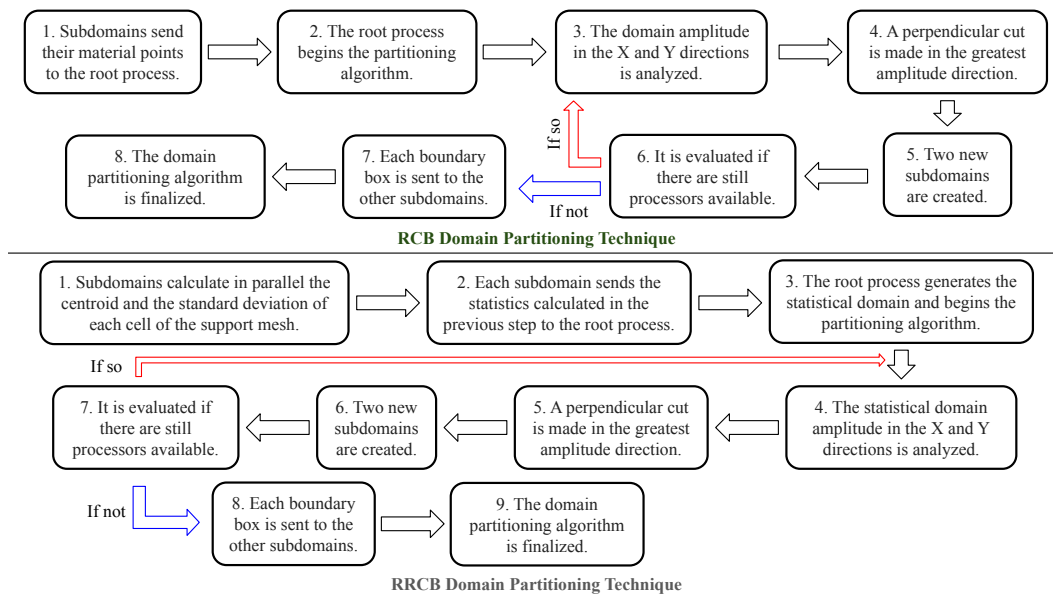


Figure 1. Flowchart of RCB and RRCB.

4 Discussion and results

To collect the results of the above-mentioned strategy, we adopted a computational model of a slope, shown in Fig. 2. The colors represent the sub-domains created by RCB technique, illustrating the partitioning algorithm.

The computational model simulates how the gravitational force acts on a slope, where we measure the displacement of the slope and the current process stresses and strains, how Xu et al. [5] approaches. Moreover, we accomplish an initial analysis to discover the performance of the RCB in the computational model adopted. Table 1 shows the reached speedup of the RCB applied to the parallel simulation compared to the serial simulation of the problem. To calculate the speedup parameter, eq. 1 has used, where the subscript ST means Simulation Time.

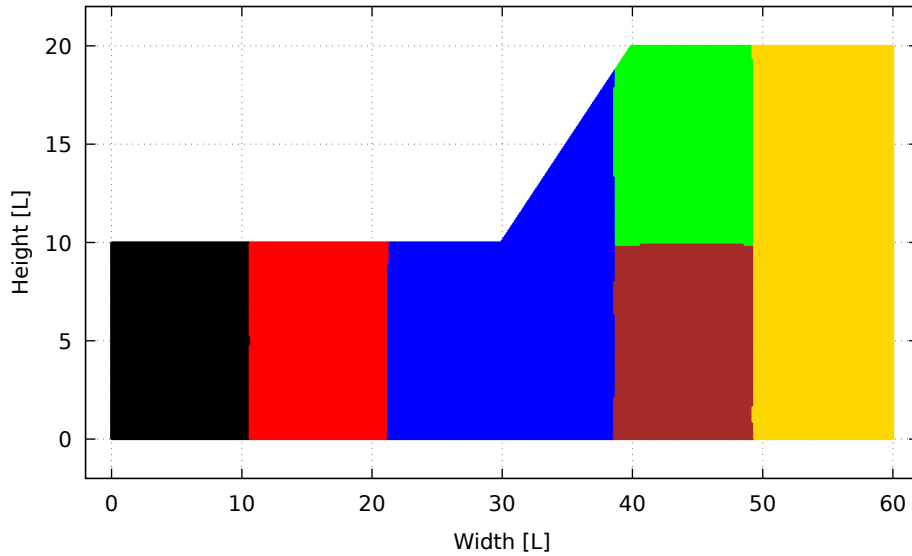


Figure 2. Partitioning slope illustration. Six processes were used in this example.

$$Speedup = \frac{Serial_{ST}}{Parallel_{ST}} \quad (1)$$

Table 1. Results obtained from the computational simulation of a model with 54,440 material points and 5,412 nodes.

Serial		Parallel		Speedup
Number of Processors	Simulation Time	Number of Processors	Simulation Time	
1	20min15s	2	12min35s	1.68
		3	12min26s	1.65
		4	8min23s	2.45
		5	8min56s	2.30
		6	9min28s	2.17

Note that the parallel simulation obtained better results when compared to the serial simulation time, mainly with four processors. Due to this fact, a fixed number of four processors was used to collect the results from the RRCB, since the parallel simulation with this amount of cores proved to be the best in terms of speedup. In the simulations using the RRCB, the following factors were varied: discretization of the support mesh, multiplier coefficient of the standard deviation (σ) and the number of points generated to create the statistical domain. The variation of these factors implies different results of load unbalance and speedup. To illustrate the strategy applied in RRCB, Fig. 3 shows the simulation real domain, the support background mesh (red lines), and the centroid of each cell in it (green points). To prevent the Fig. 3 being overloaded with too many drawings, we omitted the support meshes of the other sub-domains, as well as the centroid of their respective cells.

Figure 4 shows the statistical domain generated by the information of centroid and standard deviation from each simulation sub-domains. Note that there are some "flaws" in the generated statistical domain. This is expected, since it is not possible to guarantee that the entire cell of the support mesh is filled with material points. We can guarantee, using standard deviation and the mean (centroid of a cell) of the geometric positions of the points in the real domain, that the particles will be in interval $[\bar{x} + k\sigma_x, \bar{x} - k\sigma_x]$ to X direction, and $[\bar{y} + k\sigma_y, \bar{y} - k\sigma_y]$ to Y direction, when being generated pseudo-randomly. The \bar{x} and σ_x represents the mean and standard deviation in the X direction, respectively; \bar{y} and σ_y represents the mean and standard deviation in Y the direction, respectively; and k represents the multiplier coefficient of the standard deviation previously defined arbitrarily, to guarantee the

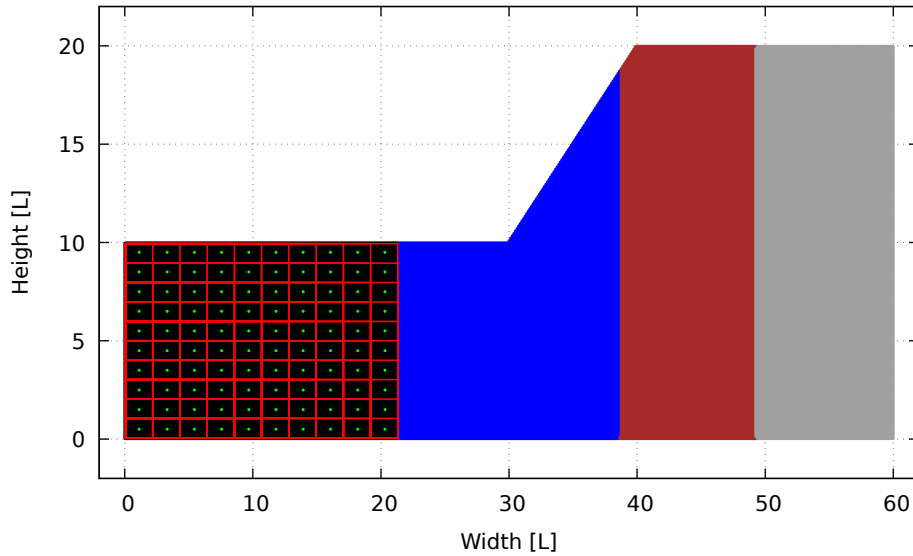


Figure 3. The real domain of the simulation. Four processors and a 10x10 support mesh were used to calculate the centroid and the standard deviation of each cell.

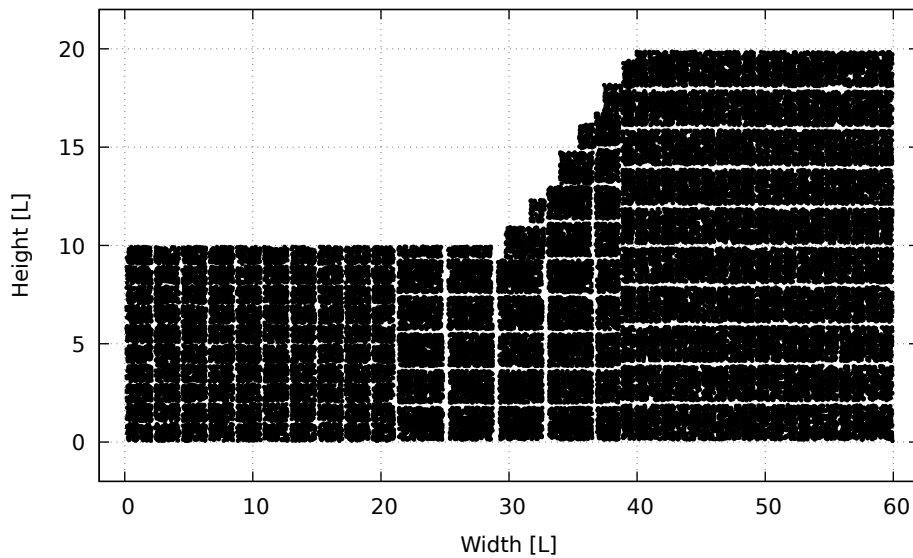


Figure 4. The statistical domain generated by the statistical information.

generated points are in the desired interval. As our objective is to build the boundary boxes for each simulation sub-domain, it doesn't matter if the statistical domain is equal or not to the real domain, but if the coordinates of the pseudo-randomly generated points are similar to the original points. When observing the inclined region, we noticed an inaccuracy in the generation of material points compared to the real domain. To correct this problem, we would need to refine the support mesh, which demands a larger computational effort. Fig. 5 shows the speedup of the partitioning algorithm applied in the statistical domain with 54,440 points generated (100% of the real domain), 27,226 points generated (approximately 50% of the real domain), 13,652 points generated (approximately 25% of the real domain), and 6,868 points generated (approximately 12.5% of the real domain).

When analyzing the four charts presented in Fig. 5 and comparing them with the speedup in Table 1, note that in certain configurations there was an improvement in speedup parameter using 4 processors in the parallel simulation. With fewer material points generated in the statistical domain, fewer operations are needed to find the boundary boxes of the simulation sub-domains. This fact tends to increase the computational performance of the entire simulation. This premise is proven in the following situations: 1) Discretization of the support mesh in

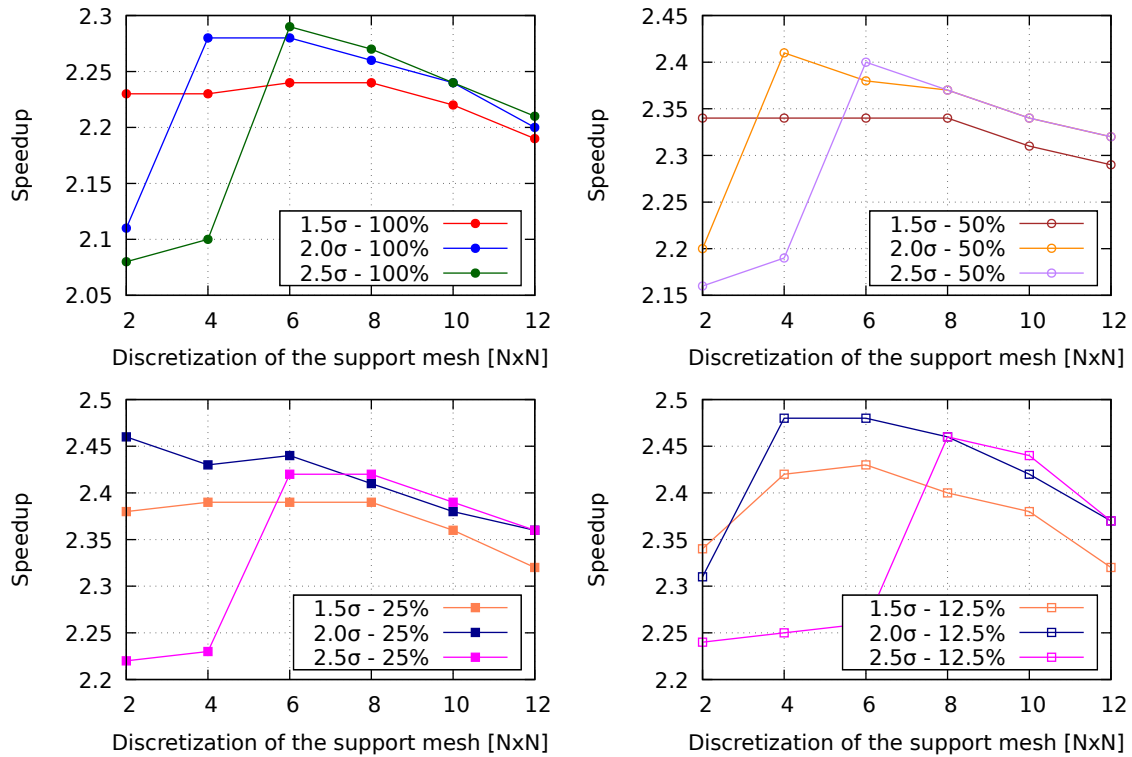


Figure 5. Reached speedup using the statistical domain with 54,440 points (top left chart), 27,226 points (top right chart), 13,652 points (bottom right chart) and 6,868 points (bottom left chart).

2x2, 1.5 σ , and 25% of the real domain; 2) Discretization of the support mesh in 4x4, 2.0 σ , and 12.5% of the real domain; 3) Discretization of the support mesh in 6x6, 2.0 σ , and 12.5% of the real domain; 4) Discretization of the support mesh in 8x8, 2.0 σ , and 12.5% of the real domain. These four configurations performed better when compared to parallel simulation using the same 4 processes and RCB as a domain partitioning technique. But speedup is not the only parameter that must be studied to conclude if one approach is better than the other. It must be ensured that the load balance is controlled to validate the results obtained. Thus, the four configurations that performed better in the RRCB and the configuration that performed better in the RCB were used in the load balance test.

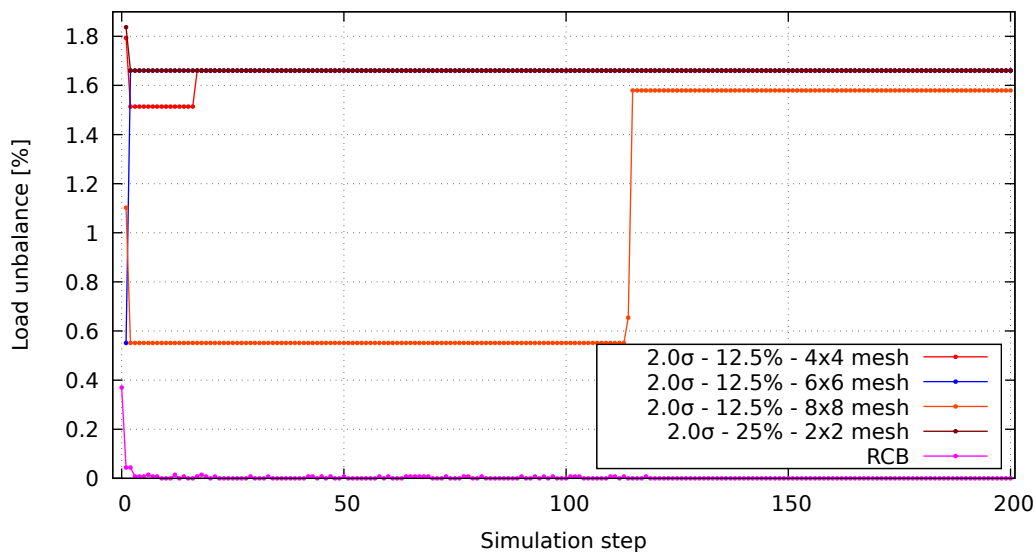


Figure 6. Load unbalance history in five different configurations.

Figure 6 shows the history of load unbalance during the simulation, calculated by the eq. 2. In it we can see that the classic RCB has greater control of the load balance of the simulation. The load unbalance is greater in the RRCB due to the pseudo-random generation of material points based on statistical information. One possible solution to reduce the unbalance in the RRCB is to increase the discretization of the support mesh in exchange for losing Speedup. Another possible solution is to use machine training techniques to train a model that optimizes the generation of the statistical domain and reduces the load imbalance of the simulation.

$$Lu = \frac{N_I - N_R}{N_I} * 100\% \quad (2)$$

Where:

- Lu is the load unbalance
- N_I is the ideal number of material points by sub-domain calculated analytically
- N_R is the actual number of material points present in the simulation sub-domain

5 Conclusions

At the end of the studies, the RRCB technique proved to be a good alternative to reduce the volume of data exchange between the sub-domains that exist in the RCB technique, with a small increase in load imbalance. However, we believe that the RRCB technique can be improved, looking for other, more accurate ways of generating the statistical domain and, thus, improving the load imbalances caused by it, which is considerably greater compared in the classic RCB. As future work, we intend to implement a machine learning system that will assist us in generating points in the RRCB statistical domain and reducing load unbalances. As we will increase the number of material points in the computational model, the communication between the simulation sub-domains becomes a problem, which tends to compromise all computational performance and slow down the entire simulation.

Acknowledgements. We would like to thank the Federal University of Alagoas and the Laboratory of Scientific Computing and Visualization for all the opportunities offered to develop this research and for the magnificent infrastructure that enabled the development of the work. We also thank CENPES/Petrobras for the partnership and promotion of the project, because without the company's participation the project would not be possible.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

- [1] K. J. al Kafaji, I., 2013. *Formulation of a Dynamic Material Point Method (MPM) for Geomechanical Problems*. PhD thesis, Universität Stuttgart.
- [2] Cintra, D. T., Willmersdorf, R. B., Lyra, P. R. M., & Lira, W. W. M., 2016. A hybrid parallel DEM approach with workload balancing based on HSFC. *International Journal for Computer-Aided Engineering and Software*, vol. 33, pp. 2264–2287.
- [3] Eibl, S. & Rude, U., 2018. A systematic comparison of dynamic load balancing algorithms for massively parallel rigid particle dynamics. *CoRR*, vol. abs/1808.00829.
- [4] Pacheco, P., 1997. *Parallel Programming with MPI*. Morgan Kaufmann.
- [5] Xu, Q., Jianyun, C., Li, J., & Yue, H., 2014. A genetic algorithm for locating the multiscale critical slip surface in jointed rock mass slopes. *Mathematical Problems in Engineering*, vol. 2014, pp. 1–10.