A high performance algorithmic variant of MATSim road traffic simulator

Sara Moukir, Nahid Emad, Stéphane Baudelocq

2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) | 979-8-3503-1199-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/IPDPSW59300.2023.00149

A High Performance Algorithmic Variant of MATSim Road Traffic Simulator

Sara Moukir
LI-PaRAD and Maison de la Simulation
University of Paris Saclay / UVSQ
Eiffage Energie Systèmes
Vélizy-Villacoublay, France
sara.moukir@universite-paris-saclay.fr
sara.moukir@eiffage.com

Nahid Emad
LI-PaRAD and Maison de la Simulation
University of Paris Saclay / UVSQ
Gif-sur-Yvette, France
nahid.emad@uvsq.fr

Stéphane Baudelocq
Eiffage Energie Systèmes
Vélizy-Villacoublay, France
stephane.baudelocq@eiffage.com

***Abstract*—In this paper, we propose a new high-performance computing approach to road traffic simulation. Multi-agent road traffic simulators are the most accurate and realistic that currently exist, however they are very resource intensive and the data are massive when simulating a metropolis or a region. The main contribution of this paper is the presentation of a new concept based on the Unite and Conquer approach, allowing to set up several parallel executions with different parameters. Applied on MATSim, one of the most present multi-agent traffic simulators in the literature, it allows to reduce the number of iterations needed to converge to the optimal solution. In this paper, we show how the Unite and Conquer approach applied to MATSim brings a substantial gain in computing time and points out its potential application to other multi-agent simulators.**

***Index Terms*—road traffic simulation, big data analysis, high performance computing, complex and heterogeneous dynamic system, Unite and Conquer, MATSim, parallel computing**

I. Introduction

The study of road traffic is similar to the study of a complex and heterogeneous dynamic system. Actually, the individual will of each agent combined with the diversity of the means of transport make the simulation difficult. The microscopic and erratic nature of this phenomenon requires a more complex paradigm than that of a simple mathematical model. Multi-agent simulation turns out to be a relevant paradigm to express this phenomenon effectively. Indeed, a multi-agent simulation offers a dynamic environment and a behavior specific to each agent framed by a certain number of rules. It makes it possible to obtain a co-evolutionary simulation that best imitates the real phenomenon of road traffic. Agent-based road traffic simulators such as MATSim [1], SimMobility [2] or POLARIS [3] have been developed and used over the past few years to simulate the transport systems of several different cities. They have the ability to accurately replicate movement patterns. Given their disaggregated representation of travel demand and supply as well as rich spatial and temporal detail, these agent-based transport models have the potential to enable unprecedented detailed analysis.

MATSim is a highly customizable and flexible framework for simulating transportation systems [1]. It is specifically designed for large-scale, long-term transportation planning studies and can simulate the interactions between individuals, vehicles, and infrastructure. POLARIS and SimMobility are also transportation simulation tools, but they may not have the same level of customization and flexibility as MATSim. Additionally, MATSim has been widely used and validated in research studies, whereas it may not be as well-established for POLARIS and SimMobility or others agent-based traffic simulators. As a consequence, we have chosen to focus on MATSim as the object of our research. The MATSim framework consists of several modules which can be combined or used stand-alone. Its modularity, extensibility and performance caught our attention. These modules support are as follows : demand modeling, agent-based mobility simulation, Replanning, and output analysis methods [1]. However, the computational requirements are increasingly important. For example, from our experiments MATSim takes about 9 minutes on average to run an iteration for moving about 187,000 agents in its Los Angeles scenario (MATSim Los Angeles 1%). As a result, high performance computing approach turns out to be relevant to have reasonable execution times. Multi-threading is already used in some sections of MATSim, but the use of high performance computing field is still exploitable [4].

In this paper, we propose a parallel algorithm design based on the Unite and Conquer [5] approach for MATSim. Unite and Conquer is a heuristic approach to reduce computation time by integrating a jump processing on an algorithm - it considers a 2-step process - first : compute a starting set, second: merge the results and then go back to the first step. Providing such an iteration allows to explore a sub-optimal solution with different methods. This approach has no guarantee to get an optimal solution, it is a custom method and depending on the parameters, it can bring a substantial speed-up.

The rest of the article is organized as follows: section 2 describes the challenges of the parallelization carried out, section 3 describes in a more optimal way the operation of the traffic simulator MATSim, and section 4 describes our algorithm proposal. Finally, section 5 presents our experiments and the conditions in which they were performed our results.

979-8-3503-1199-0/23/$31.00 ©2023 IEEE
DOI 10.1109/IPDPSW59300.2023.00149

II. Related work

Several works have been carried out to date to try to effectively improve the execution time of multi-agent road traffic simulators thanks to parallel / distributed computing techniques.

Different HPC strategies have been deployed. One of these strategies consists in dividing the road network into several sub-networks so as to assign each one of them to a computation unit [6]. This strategy raises other issues, the main one being that of inter-sub-network communication, which dominates in terms of execution time. Among the microscopic road traffic simulators using these strategies, TRANSIMS [7], AIMSUN [7] and Paramics [7] are the most popular parallel traffic road simulators, although they are not based on the multi-agent paradigm.

Regarding agent-based simulators, the distribution of agents between the different computing units is part of the strategies used to speed up the execution [8]. The same problem of inter-sub-network communication remains since for each time step, all the computation units must know the state of the whole network. Indeed, the advancement of agents on the network depends on the movements of other agents, managed by other hosts; hence the continuous communication between computation unit to inform of the local dynamics of each of them.

Other strategies have been applied to MATSim in particular, such as the implementation of a specific library for parallelization [9] or the modification of the operation of certain modules. The Replanning module, for example, has been revised so that this step executed according to other parameters, so as to have faster convergence [10]. The simulation module has also been modified, so that there is an "event-driven" approach instead of the "time-step based" approach (QSIM vs. JDEQSIM and HERMES)[11, 12]. Here we propose a new algorithmic concept for the Replanning module based on the use of parallelism. Our goal is to obtain a faster convergence and then a reduced execution time.

III. MATSim Simulator

A. Overall functioning

MATSim is an open-source framework for implementing large-scale agent-based transport simulations implemented in Java. The traditional operation of MATSim consists in simulating each agent individually, each having a set of plans. These agents travel on a specific road network. Response to travel request is obtained by combining data from several different sources, so as to generate a synthetic population representative of the simulated geographical area. An agent in a multi-agent simulation is a virtual representation of an entity that can sense, reason and act in the simulated environment, with a set of rules or behaviors that dictate how it interacts with other agents and the environment. Agents are simulated on the road network, executing their initial plans and interacting with each other. The execution of a plan makes it possible to attribute a score to it, thanks to a predefined Scoring function. During the Replanning phase, a portion of agents can randomly modify their plans. This process generates and selects a completely new plan for a given agent, while the rest of the agents continue with their existing plan sets. The plan set has a finite size nb_plans: when a new plan is generated and the agent has reached a size number of plans in its set, the plan with the lowest score is deleted to make room for the newly generated one.

Each agent will therefore have for the next iteration, a different plan selected. This plan will be newly generated or not, depending on whether the agent is part of the agent portion authorized to modify and generate a new plan during the Replanning phase. Plans are re-scored and new plans are generated and selected. The loop continues in this way over several iterations max until the agents can no longer improve their scores and then reach an equilibrium.

These operating mechanisms of MATSim are distributed according to three sequential phases, or modules: execution of Mobsim, Scoring and Replanning [13]. Let us briefly recap the operation of MATSim through these modules, shown schematically in Figure 1 :

- **Input :** As input, MATSim takes a number of files, including one containing the initial plan for each agent for the day. There is also the network, which corresponds to the road network and the public transport network, specific to each simulated geographical entity. There are of course many other inputs, including some parameters but for the sake of clarity we limit ourselves to these elements.
- **Execution module (Mobsim) :** executes the daily plans of each agent in the population. This module simulates the way agents carry out their daily activities and move from one place of activity to another ;
- **Scoring module :** uses a utility function (taking into account both activity and movement) to evaluate the daily plans based on the performance of each agent in the execution module ;
- **Replanning module :** adjusts the plan elements (e.g. departure time or mode of transportation) based on plan scores, to adapt plans to traffic flow.
- **Output :** At the end of the execution, we get the scores of each plan for each agent, new adjusted plans for the portion of agents that underwent the Replanning, a file containing the daily events (such agent arrives at such activity planned in his schedule, or leaves it, at such time...). There is also a set of statistics specific to the execution in the form of text files and or graphics.

B. Replanning module

During the Replanning phase, a portion R of the agents can randomly modify their plans, thus generating and selecting new plan. By default, only 10% of the agents are concerned by the Replanning phase, i.e. $R = 0.1$, but the user is free to change this value. The set of plans has a finite size: the plan with the lowest score is "overwritten" over time. Each with their new plan selected, agents are once again simulated

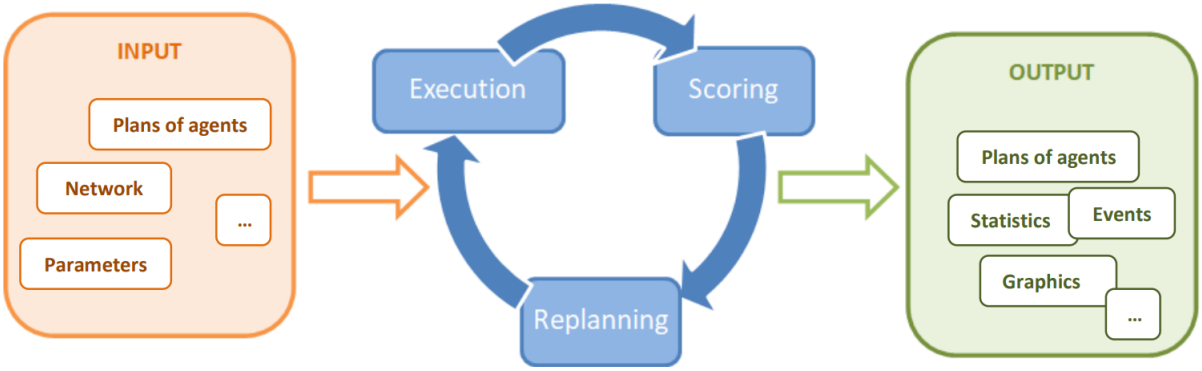

Fig. 1. Iterative MATSim loop

on the transport network, their executed plans are scored and new plans are generated and selected. This loop continues for several iterations until agents can no longer improve their scores, so achieve a balance. Generally, there are four strategies [14]:

- **Plan selector** : Selection of one of the existing plans in the agent's set of plans (up to nb_plans). There are several strategies for the plan selector, among them : KeepLastSelected keeps the plan selected in the previous iteration, BestScore selects the plan with the highest score from the previous iteration, SelectRandom performs random selection between the plans, ChangeExpBeta changes to a different plan, with probability dependant on $e^{\Delta_{score}}$ where Δ_{score} is the score difference between two plans. In the Los Angeles scenario we worked on, it is the BestScore strategy that is found by default and that we kept for our experiments. The weight of this strategy relative to the others will be noted $\rho_{bestscore}$.
- **Route innovation** : Mutate a random existing plan by re-routing trips. Routes are computed based on the traffic conditions of the previous iteration. There are different routing algorithms such as Djikstra or A*. The weight of this strategy relative to the others will be noted $\rho_{reroute}$.
- **Time innovation** : Mutate a random existing plan by randomly shifting all activity end times backwards or forwards. The weight of this strategy relative to the others will be noted ρ_{time}.
- **Mode innovation** : Mutate a random existing plan by changing the mode of transport. The weight of this strategy relative to the others will be noted $\rho_{transportmode}$.

Each module is given a weight determining the probability, by which the course of action represented by the module is taken. The strategy modules' weights are normalized, in case they do not sum to one.

Algorithm 1 MATSim algorithm

1: **Start.** Choose max, nb_plans and the strategies probabilities weights such as $\rho_{bestscore} + \rho_{reroute} + \rho_{time} + \rho_{transportmode} = 1$, R ...
2: **Iterate.** For i =1, ... , max
Run one iteration of MATSim : Mobsim, Scoring and Replanning only by the portion of R agents to which the Replanning applies. The Replanning module duplicates one plan for each agent independently (if $i < nb\_plans$) or modifies the existing ones.

IV. A NEW VARIANT OF MATSIM ALGORITHM

We propose an approach whose goal is to reduce the convergence time, in order to obtain a shorter execution time. To do this, we exploit the potential offered by parallel computing. Our approach consists in launching several instances of MATSim in parallel. These instances are the same: they relate to the same scenario, the same characteristics, except for the variables linked to the Replanning module. The weights are distributed differently according to the instances, which will produce, during the Replanning phases of each instance, different plans iteration after iteration despite the starting plans being the same. Thus, the weight distribution allows to cover a wider set of possible solutions.

In addition to this, our approach sets up, every $step$ iterations, a synchronous communication between the instances so as to make an exchange of plans between them. After a given iteration, instance i (with $i \in \{1, ..., N\}$ and N the number of instances) will know the scores of other instances for a given agent a_k, (with $k \in \{1, ..., M\}$ and M the number of agents).

Consequently, instance i will know which instance has the plan with the best score for agent a_k. It can decide to recover it or not, according to a very specific criterion. This criterion is as follows: consider the instance α and the instance β (α, $\beta \in [1; N]$).

For a given agent a_k, instance α gets score S^k_α and instance β gets score S^k_β, with $S^k_\alpha < S^k_\beta$ and $S^k_\beta > S^k_i$ $\forall i \in [1; N]$.

The α instance could then simply retrieve the plan associated with the S^k_β score from β and run the next iteration.

However, if each instance retrieves for each agent each time the plan of the instance that made the best score, we would have instances that would all execute the same plan for each of the agents.

The idea is then to set up a criterion allowing the instances to decide whether it is relevant to recover the plan of another instance or to give it up, even if its own score is lower than that of another instance for a given agent. We have chosen to focus on the dispersion measure between the scores obtained by the different instances for the same agent to define a plan exchange criterion. In this way, the criterion would make it possible to rely indirectly on the performance of the plans to validate or not the exchanges, while limiting them to the instances having obtained the worst scores relative to those of the other instances for each agent.

This criterion is the following: consider the vector V_{S^k} of size N, containing the scores of all the instances for the agent a_k.

If the absolute value of the subtraction of S_α^k (α being the instance concerned by this decision) by S_β^k is greater than the standard deviation of V_{S^k}, then the plan exchange is favorable and α recovers the plan associated with the score S_β^k from the β instance.

The next iteration will run MATSim with plans that had scores that were considered good, and this process will repeat itself after a $step$ number of iterations. This approach is inspired by the Monte-Carlo method, aimed at calculating a numerical value, and using random processes, that is to say probabilistic techniques. Let's summarize the essential parameters listed in Algorithm 2:

- N : choose the number of MATSim instances that will run in parallel
- max : MATSim stops after a predefined number of iterations. It doesn't stop by itself when convergence is obtained.
- M is the number of agents in the current scenario of MATSim
- $step$: Choose a number of iteration steps between which communication will take place between the different instances
- $iter\_processing := max \div step$, this is the number of times the processing and communications are performed.
- Choose the weights for probabilities associated to Replanning strategies. All these strategies are described in section III.B. The assigned weights are such that $\rho_{bestscore} + \rho_{reroute} + \rho_{time} + \rho_{transportmode} = 1$. Each instances has a different distribution of weights.
- Set $nb\_plans = 1$. In this way, each instance will focus on optimizing a single plan.

Figure 2 illustrates the algorithm in a very basic way, with a single agent a_k and two instances i and j. We assume that the agent a_k is "concerned" by the Replanning module. A certain number $step$ of iterations of MATSim is carried out in a parallel way between the instances i and j. MATSim stops, or pauses its execution and performs communications between the two instances. They communicate the plans and

Algorithm 2 Muti-MATSim

1: **Start.** Choose parameters
2: **Iterate.** For $i = 1$, ..., $iter\_processing$, **do in parallel**

Run one of the N instances of MATSim for $step$ iterations

Communications

Send the scores and the plans of all agents to others N ranks / instances

Receive the scores and plans of all agents of the others N ranks.

Compute the best score obtained among the N ranks for each agent, and identify the associated rank. During this step, the calculation of the standard deviation std_k between the N scores for an agent $a_k \forall k \in [1, M]$ is performed.

Exchange plans

- Let agent a_k and its score obtained by the current instance $i = S_i^k$.
- Consider that the best score among the N scores obtained by all instances for agent a_k is S_{best}^k such as : $S_{best}^k > S_i^k \forall i \in [1, N]$

for each agent $a_k \forall k \in [1, M]$:

if $|S_{best}^k - S_i^k| > std_k$ **then** get the plan associated to S_{best}^k for the agent a_k

the associated scores resulting from the last $step$ iterations for the a_k agent. The best score of the plan obtained between the instances is assigned to the value S_{best}, and each instance evaluates its own score S_i^k or S_j^k with respect to S_{best} via the comparison operation displayed on the schema. Thus, according to the results of these comparisons, the execution of MATSim will resume at the last iteration with a new plan or not. This operation is repeated $iter\_processing$ number of times in total. Of course, there are actually more instances and even more agents in reality. We have limited ourselves here to $N = 2$ and $M = 1$ for a better understanding.

V. Experiment and performance

A. Hardware resources

To perform the different experiments for this work we used "Ruche" [15]. Ruche is a project created by two institutions, CentraleSupélec and ENS Paris-Saclay, to pool their computing resources within the Paris-Saclay University, host them at IDRIS and set up a common support team. a Lenovo supercomputer with the following features:

- A Spectrum Scale GPFS parallel file system (380 Tio of usable space, IOs rate : 9 GB/s)
- Intel OPA network 100 Gbit/s
- 192 compute nodes comprising 2 Intel Xeon Gold 6230 20 cores @ 2.1 GHz (Cascade Lake) and 192 GB of RAM
- 14 compute nodes comprising 4 Intel Xeon Gold 6230 20 cores @ 2.1 GHz (Cascade Lake) with 1.5 TB of RAM and 480 GB of SSD disk

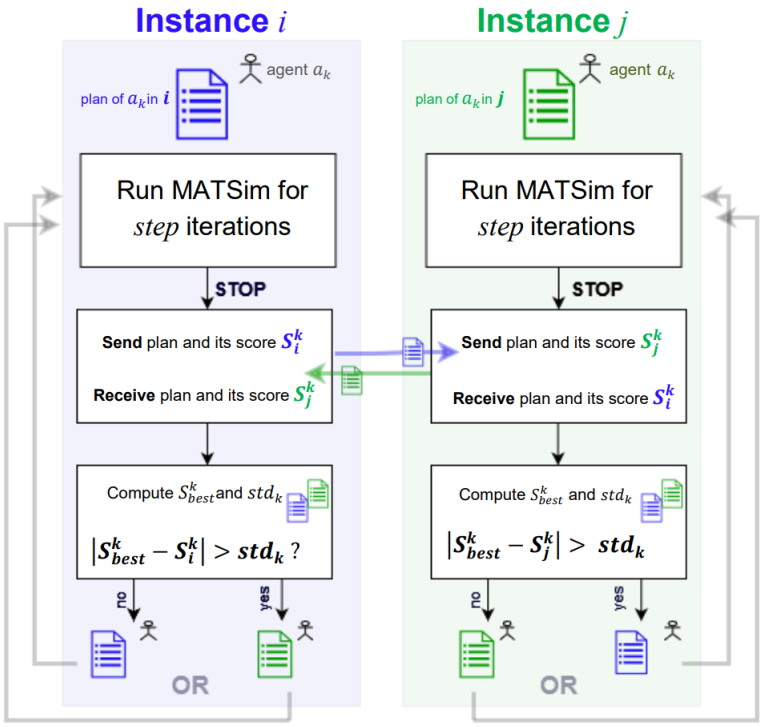

Fig. 2. Illustration of the new variant of MATSim algorithm with two ranks / instances and one agent

- 10 GPGPU nodes comprising 2 Intel Xeon Gold 6230 20 cores @ 2.1 GHz (Cascade Lake), 768 GB of RAM and 4 NVIDIA Tesla V100 PCIe graphic cards with 32 GB of GRAM
- 5 GPGPU nodes comprising 2 Intel Xeon Gold 6230 20 cores @ 2.1 GHz (Ice Lake), 1024 GB of RAM and 4 NVIDIA Tesla A100 NVLink graphic cards with 40 GB of GRAM

B. Scenario and setup

Several scenarios based on several metropolises are available for the simulation of road traffic with MATSim. Among them, we find Berlin, Paris, Seoul, New York, Los Angeles and many others. We chose to simulate Los Angeles 0.1% initially given the number of agents that this scenario contains, with neither too much nor too little for a first test of our approach [16]. For this experiment, we decided to assign the following values to the variables:

- The number of MATSim instances that will run in parallel $N = 4$
- MATSim stops after a predefined number of iterations $max = 300$,
- The number of agents in the Los Angeles 0.1% MATSim scenario $M = 19123$
- The number of iteration steps between which processing will take place between the plans of the different instances $step = 50$
- Set $nb\_plans = 1$. In this way, each instance will focus on optimizing a single plan

Regarding the different strategies used :

- **Plan Selector:** given that $nb\_plans = 1$, each agent has a unitary space and therefore only one plan. We have kept the ChangeExpBeta strategy because it is the one used by default in the Los Angeles scenario, but in our approach this data is not influential.
- **Route innovation**: this strategy is indeed active for this approach, we call it here Reroute.
- **Innovation mode**: this strategy is also active, we will call it ModeChoice. Among those available for this innovation, we randomly chose the SubtourModeChoice strategy. This allows you to use several modes of transport on sub-tours in a single day.
- **Time innovation**: the TimeAllocationMutator module is active in our approach. We'll abbreviate it as TimeMutator.

The weights used for each of the strategies are shown in Table I, for each instance. Since we had to compare our approach to a classic run of MATSim Los Angeles 0.1%, the weights for the classic version are also shown. It is possible to configure these weights specifically for each sub-population on MATSim. We did not apply it for the freight sub-population, but only for "persons".

The weights change very slightly between the different instances: we wanted to assess the impact of different settings while maintaining correct accuracy compared to the baseline. As specified above, we have set up four instances of MATSim executed in parallel. This same script is used to process data between *step* iterations (modification of XML files, com-

TABLE I
DISTRIBUTION OF WEIGHTS BETWEEN THE DIFFERENT STRATEGIES

	PlanSelector	Reroute	ModeChoice	TimeMutator
Instance 0	0.85	0.05	0.05	0.05
Instance 1	0.8	0.1	0.5	0.05
Instance 2	0.5	0.2	0.1	0.2
Instance 3	0.8	0.05	0.05	0.1
Baseline	0.85	0.05	0.05	0.05

Fig. 3. Score progress of the four instances VS. the baseline

munications between instances...). MATSim instances have been treated as "shells": we haven't changed anything except the various editable parameters on the XML files. The four instances were launched in parallel on four different Ruche compute nodes, with memory set at 100GB for each instance. Regarding communications, the Message Passing Interface (MPI) library was used in its version for Python mpi4py. mpi4py is implemented on top of the MPI specification and exposes an API which grounds on the standard MPI-2 C++ bindings. The baseline version of MATSim Los Angeles 0.1% was launched under the same conditions, sequentially on a Ruche supercomputer compute node with 100 GB memory.

C. Discussion and results

Running the MATSim Los Angeles 0.1% baseline simulation took 8 hours, while our approach consisting of the four instances of MATSim Los Angeles 0.1% ran in 8 hours and 40 minutes. According to our profiling analysis of the execution time, the extra 40 minutes correspond to synchronous communications between instances as well as file processing between $step$ time steps.

Note that when MATSim is deliberately stopped before the end at a specific iteration, there is a whole time for data generation and shutdown to be devoted, as well as additional time for its restart at the last iteration.

To evaluate our approach, let's look at how the average of the scores executed evolves over the iterations and how it converges for each of the instances. Figure 3 shows the evolution of the average scores of executed plans for all agents. The "peaks" each time represent the exchanges of plans between the instances. Indeed, the adopted plan is associated with a score which explains this effect. The baseline, in black, shows convergence starting around the 200th iteration. The sudden growth towards the 240th iteration is due to the fact that MATSim Los Angeles 0.1% is set by default to stop all innovation after 80% of the iterations have been completed. As a result, for each agent, the plan with the best score is executed, which explains this sudden and sustained increase. As a reminder, the baseline is set with $nb\_plans = 5$, while the four instances are set with $nb\_plans = 1$. The plan score for each agent is then always the best score, and the trend observed for the baseline with the sudden increase in the average of executed plans is therefore not found. If we consider that the convergence of the average scores is reached around the 240th iteration for the baseline, the average score associated with it is around 105. We observe that this score is reached earlier by all the instances executed in parallel . From the 52nd iteration, instances 0 1 and 3 reach the score considered as the convergence score for the baseline, while instance 2 reaches this score around the 152nd iteration. The exchange of plans between the instances seems beneficial: the sudden increase in the scores of the plans just after the exchanges seems to decrease more or less according to the instances just after systematically, while remaining higher than the average score obtained before the exchange procedure. On average, the baseline as well as the parallel instances each require 1.6 minutes per iteration. Running parallel instances requires about 8 minutes every 50 iterations (here $iter\_step = 50$). The convergence score, previously established at around 105 after around 6.4 hours for the baseline. For the parallel version, this score is reached after about 1.5 hours by instance 0. Table II shows these elements: it indicates for each instance the iteration at which the convergence score is reached as well as the associated necessary duration. Also, we can see the speed-up compared to the baseline version.

TABLE II
ITERATION AND EXECUTION TIME TO REACH THE "CONVERGENCE SCORE" FOR EACH INSTANCE AND SPEED-UP COMPARED TO THE BASELINE

	Iteration	Duration	Speed-up compared to baseline
Baseline	225	6h	
Instance 0	51	1.5h	4
Instance 1	51	1.5h	4
Instance 2	72	2h	3
Instance 3	160	4.6h	1.3

It is customary to check the consistency of some metrics when providing a new approach for MATSim compared to a baseline simulation. We have chosen here to check the shares of each mode of transport between our parallel approach and the baseline simulation, as well as the consistency of the road volume between the two versions.

1) Comparison of mode shares between the two versions: Figure 4 is generated directly by running the MATSim Los Angeles 0.1% baseline. It makes it possible to observe the

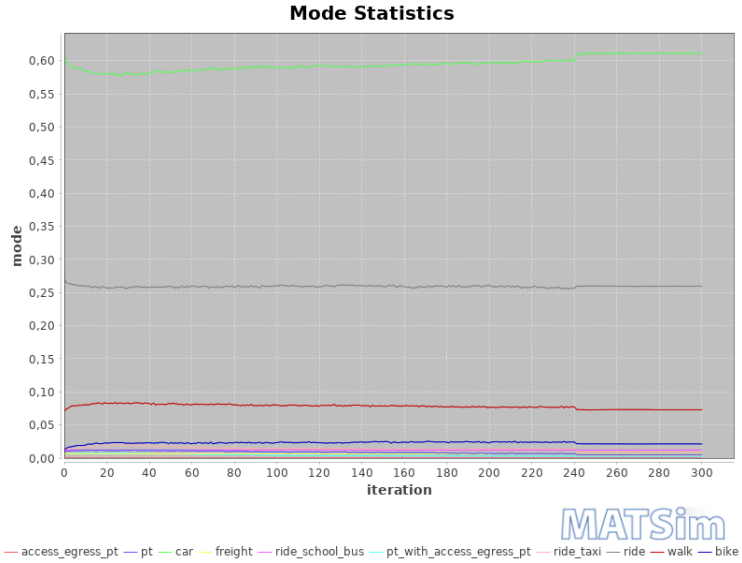

Fig. 4. Mode shares of the baseline MATSim Los Angeles 0.1%

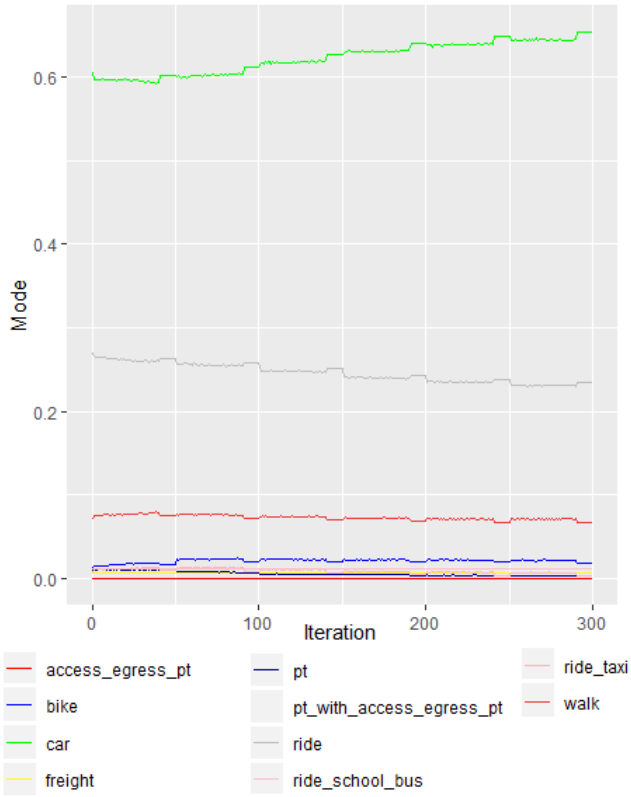

Fig. 5. Mode shares of the parallel version of MATSim Los Angeles 0.1%

distribution of the modes of transport according to the iteration. Figure 5 is similar, it was generated from output data via RStudio. It corresponds to instance 0. The distribution seems quite similar between the two versions and the figures more or less correspond. The car seems to be the preferred means of transport at more than 60%, followed by public transport. The distribution of the other modes of transport is quite similar between the results of the baseline and those of our approach. The other trends are also similar on the two versions. Instances 1, 2 and 3 show more or less the same results.

2) Traffic volume comparison: The volume of road links is also an interesting indicator for comparing our approach with the baseline and checking the consistency between the two versions. Figure 6 compares two leg histograms. These are files automatically generated by MATSim at each iteration. A leg histogram depicts the number of agents arriving, departing or en route, per time unit. There is one generated for each mode of transport, and one confusing them all. Versions with all modes of transport are displayed here. We have voluntarily used the result of the 225th iteration for the baseline, and the result of the 51st iteration for the parallel version, because these are respectively the iterations for which the executions reach what is considered to be the ”convergence score”. We observe a trend and similar patterns between the two graphs. There are admittedly very slight differences, but they can be considered as negligible on the whole. We can easily say that the number of agents arriving, departing or en route is completely consistent between the two versions and this at each hour of the day on the parallel version compared to the baseline.

Figure 7 shows two graphs representing the volume of vehicles by road link over the 7:00 a.m. - 8:00 a.m. time slot, one for the baseline and the other for the parallel version. Only 10% of the road links are represented here, they were chosen at random. We also observe here the same trend between the two versions, with slight negligible differences. We did the same checks on a dozen other time slots with the same results. We can say that the road volume between the two versions is rather similar.

Although there are other metrics that could further push the consistency checks between the two versions, those presented in this paper confirm the idea that the results provided by our approach maintain precision and give results that are completely consistent with the baseline.

VI. Conclusion

The results obtained allow to obtain a speed-up of 4 at best, potentially reducing the total execution time of MATSim by several hours. Our approach certainly requires more resources, but our analyzes allow us to affirm that the accuracy remains correct and that the output is roughly similar in comparison to the baseline. To go further, it may be relevant to test the scalability, first using a scenario with a larger volume of data (such as MATSim Los Angeles 1% and 10% or the different existing scenarios for Berlin). For a future study, we will take advantage of other properties of the Unite and Conquer approach in our experiments. For example, we will be able to exploit the potential of multi-level parallelism. In addition, we can consider the use of asynchronous communications and fault tolerance. We can also consider greatly increasing the number of parallel instances and working in shared memory.

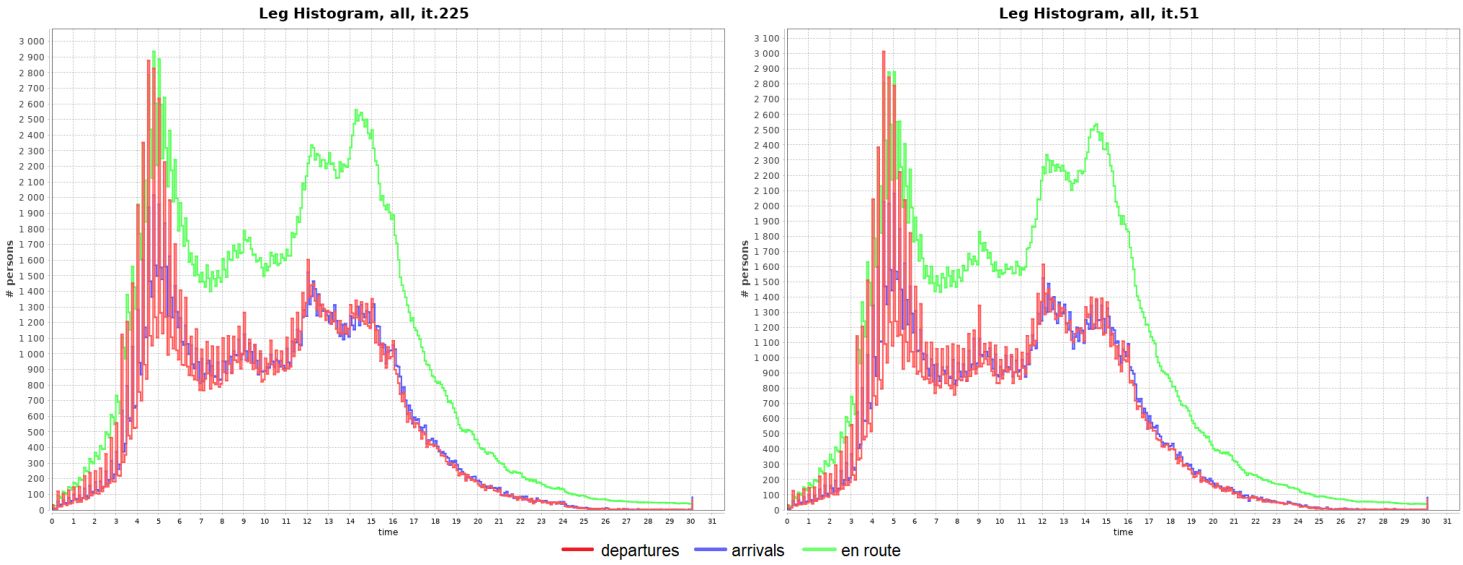

Fig. 6. Road volume on 10% of links from 8 a.m. to 9 a.m.

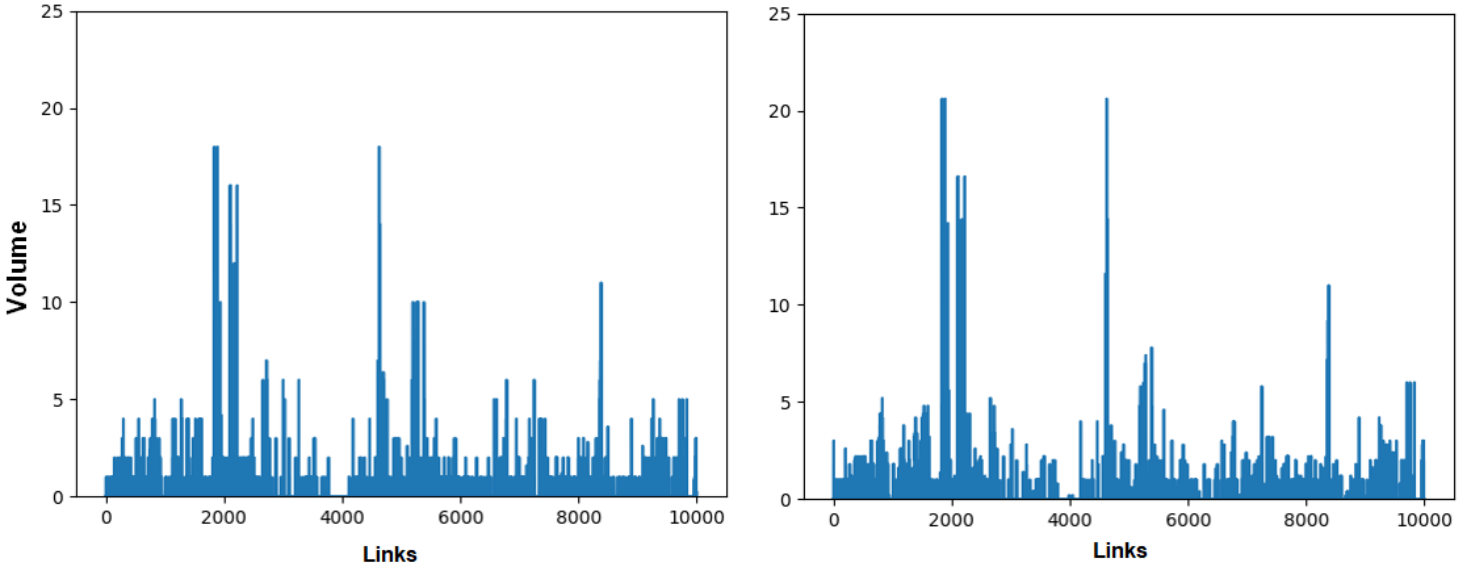

Fig. 7. Leg histogram : iter. 225 in baseline VS. iter 51 in parallel version

The modification of certain parameters, such as the iteration steps during which processing between instances is to be exploited. We could bring this approach to other road traffic simulators based on multi-agent systems such as SUMO or POLARIS for a future paper. Finally, the application of the Unite and Conquer approach to the multi-agent traffic simulator MATSim provides a real gain that paves the way for a scaling effect for acceleration.

REFERENCES

[1] Horni, A, Nagel, K and Axhausen, K W. 2016. Introducing MATSim. In: Horni, A, Nagel, K and Axhausen, K W. (eds.) The Multi-Agent Transport Simulation MATSim, Pp. 3–8. London: Ubiquity Press. DOI: http://dx.doi.org/10.5334/baw.1. License: CC-BY 4.0

[2] Adnan, M., F. C. Pereira, C. M. L. Azevedo, K. Basak, M. Lovric, S. Raveau, Y. Zhu, J. Ferreira, C. Zegras and M. Ben-Akiva (2016) SimMobility: A multi-scale integrated agent-based simulation platform, paper presented at the 95th Annual Meeting of the Transportation Research Board Forthcoming in Transportation Research Record, Washington, D.C., January 2016.

[3] Auld, J., M. Hope, H. Ley, V. Sokolov, B. Xu and K. Zhang (2016) POLARIS: Agent-based modeling framework development and implementation for integrated travel demand and network and operations simulations, Transportation Research Part C: Emerging Technologies, 64, 101-116. 101–116.

[4] Nagel, K. 2016. Other Experiences with Computational Performance Improvements. In: Horni, A, Nagel, K and Axhausen, K W. (eds.) The Multi-Agent Transport Simulation MATSim, Pp. 267–268. London: Ubiquity Press. DOI: http://dx.doi.org/10.5334/baw.40. License: CC-BY 4.0 DOI: http://dx.doi.org/10.5334/baw.52. License: CC-BY 4.0
[5] Nahid Emad, Serge Petiton. Unite and Conquer approach for high scale numerical computing. International Journal of Computational Science and Engineering, 2016, 14, pp.5 - 14. ⟨10.1016/j.jocs.2016.01.007⟩. ⟨hal-01609342⟩
[6] Tomas Potuzak, "Reduction of Inter-process communication in Distributed Simulation of Road Traffic", 2020
[7] Johannes Nguyen, Simon T. Powers, Neil Urquhart, Thomas Farrenkopf, Michael Guckert "An overview of agent-based traffic simulators", 2021
[8] Matthieu Mastio, Mahdi Zargayouna, Omer Rana, Gérard Scemama "Méthodes de distribution pour les simulations de mobilité des voyageurs", 2015
[9] Zhiyuan Ma, Munehiro Fukuda, "A multiagent spatial simulation library for parallelizing transport simulation", 2015
[10] Chengxiang Zhuge, Mike Bithell, Chunfu Shao, Xia Li, Jian Go "An improvement in MATSim computing time for large-scale travel behaviour microsimulation", 2019
[11] Horni, A and Nagel, K. 2016. More About Conguring MATSim. In: Horni, A, Nagel, K and Axhausen, K W. (eds.) The Multi-Agent Transport Simulation MATSim, Pp. 35–44. London: Ubiquity Press. DOI: http://dx.doi.org/10.5334/baw.4. License: CC-BY 4.0
[12] Dan Graur, Rodrigo Bruno, Joschka Bischoff, Marcel Rieser, Wolfgang Scherr, Torsten Hoefler, Gustavo Alonso "Hermes: Enabling efficient large-scale simulation in MATSim", 2021
[13] Rieser, M, Horni, A and Nagel, K. 2016. Let's Get Started. In: Horni, A, Nagel, K and Axhausen, K W. (eds.) The Multi-Agent Transport Simulation MATSim, Pp. 9–22. London: Ubiquity Press. DOI: http://dx.doi.org/10.5334/baw.2. License: CC-BY 4.
[14] Nagel, K, Kickh ofer, B, Horni, A and Charypar, D. 2016. A Closer Look at Scoring. In: Horni, A, Nagel, K and Axhausen, K W. (eds.) The Multi-Agent Transport Simulation MATSim, Pp. 23–34. London: Ubiquity Press. DOI: http://dx.doi.org/10.5334/baw.3. License: CC-BY 4.0
[15] http://mesocentre.centralesupelec.fr/
[16] https://github.com/matsim-scenarios/matsim-los-angeles (on the date of 01-20-2023)