COMPUTATION TOOLS 2023

The Fourteenth International Conference on Computational Logics, Algebras, Programming, Tools, and Benchmarking

ISBN: 978-1-68558-050-6

June 26 - 30, 2023

Nice, France

COMPUTATION TOOLS 2023 Editors

Petre Dini, IARIA, USA/EU

COMPUTATION TOOLS 2023

Forward

The Fourteenth International Conference on Computational Logics, Algebras, Programming, Tools, and Benchmarking (COMPUTATION TOOLS 2023), held between June 26 - 30, 2023, continued a series of events dealing with logics, algebras, advanced computation techniques, specialized programming languages, and tools for distributed computation. Mainly, the event targeted those aspects supporting context-oriented systems, adaptive systems, service computing, patterns and content-oriented features, temporal and ubiquitous aspects, and many facets of computational benchmarking.

Similar to the previous edition, this event attracted excellent contributions and active participation from all over the world. We were very pleased to receive top quality contributions.

We take here the opportunity to warmly thank all the members of the COMPUTATION TOOLS 2023 technical program committee, as well as the numerous reviewers. The creation of quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and effort to contribute to COMPUTATION TOOLS 2023. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations and sponsors. We also gratefully thank the members of the COMPUTATION TOOLS 2023 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope COMPUTATION TOOLS 2023 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the area of computational logics, algebras, programming, tools, and benchmarking. We also hope that Nice provided a pleasant environment during the conference and everyone saved some time to enjoy this beautiful city.

COMPUTATION TOOLS 2023 Steering Committee

Cornel Klein, Siemens AG, Germany
Claus-Peter Rückemann, Westfälische Wilhelms-Universität Münster (WWU) / DIMF / Leibniz Universität Hannover, Germany

COMPUTATIONAL TOOLS 2023 Publicity Chairs

José Miguel Jiménez, Universitat Politecnica de Valencia, Spain
Sandra Viciano Tudela, Universitat Politecnica de Valencia, Spain

COMPUTATION TOOLS 2023

Committee

Table of Contents

Capability and Applicability of Measurement Tools for AI Model’s Environmental Impact 1
Rui Zhou, Tao Zheng, Xin Wang, Lan Wang, and Emilie Sirvent-Hien

A Formal Model for the Simulation of Mobile Networks 9
Emanuele Covino

Capability and Applicability of Measurement Tools for AI Model's Environmental Impact

Rui Zhou, Tao Zheng, Xin Wang, Lan Wang
Orange Innovation China
Beijing, China
e-mail: {rui.zhou, tao.zheng, xin2.wang, lan.wang}@orange.com

Emilie Sirvent-Hien
Orange Innovation
Châtillon, France
e-mail: emilie.hien@orange.com

***Abstract*—More and more of Artificial Intelligence (AI) systems have been adopted by Information and Communication Technology (ICT) solutions to make effective digital transformation. In recent years environmental impact of AI systems has been investigated and methodologies have been developed to calculate their cost. In this paper, we survey, analyze, and evaluate three types of tools for counting the energy consumption/CO2 emission of AI systems. By verifying them in sets of experiments, including centralized and distributed on devices architecture, we compare ease of use of tools, simulation result vs real measurement and finally bring advice to help AI developers to take into account environmental cost of AI models with measurement tools.**

Keywords-FLOPs; PUE; PSF; TDP.

I. INTRODUCTION

The global average temperature in the past decades has increased more than 1°C compared to the pre-industrial baseline (1850-1900) [1]. Such climate change has caused more extreme weather events, rising seas, reduction of biodiversity, and negative impact to global health and safety. The global warming is a critical issue facing all mankind. Paris agreement sets a global objective for the temperature increase below 2°C. Many nations, regions, industries, companies, and individuals have put in place climate actions on their agendas. The current rise is more rapid primarily as the result of greenhouse gas emissions by burning fossil fuels for energy used in industry, transport, building, etc., which took 73.2% of global greenhouse gas emissions according to the data obtained in the year 2016 [2].

Using Communication Technology (ICT) solutions in these sectors can have a calculated potential to reduce greenhouse gas emissions by up to 15% [3]. Their own contribution to greenhouse gas emissions should not be ignored. Our focus is on Artificial Intelligence (AI) as more and more of them have been adopted by ICT solutions to make the effective digital transformation. It is expected that the Artificial Intelligence industry will be worth $190 billion by 2025, with global spending in AI systems reaching $57 billion by 2021 already [4]. The demand for computing these AI systems is growing exponentially. The intensive computation nowadays not only takes place in datacenters but also in a huge amount of edge devices closed to consumers and enterprises to support AI applications to process big data with low latency and large bandwidth requirements.

Scientists and researchers have started to investigate the environmental impact of AI systems in recent years and have developed methodologies to calculate their costs. For example, Schwartz and Doge et al. [5] refer to the AI systems that focus on accuracy without any estimation on the economic, environmental, or social cost of reaching the claimed accuracy as Red AI. They have proposed a simplified estimation of the cost of an AI which grows linearly with the cost of processing a single example, the size of the training dataset, and the number of hyperparameter experiments. OpenAI [6] has pointed out that among the three factors driving the advance of AI: algorithmic innovation, data, and the amount of computing available for training, computing is unusually quantifiable. The number of FLoating point of OPerations (FLOPs) (adds and multiplies) in the described architecture per training example can be counted. If there is not enough information to directly count FLOPs, Graphics Processing Unit (GPU) training time, how many GPUs used and a reasonable guess at GPU utilization can be used to estimate the number of operations performed. Strubell et al. [7] have quantified the computational and environmental cost of training several popular Natural Language Processing (NLP) models. The total power required at a given instance during training is related to the average Power Usage Effectiveness (PUE) for datacenter multiplying the sum of average power draw from all Central Processing Unit (CPU) sockets, average power draw from all Dynamic Random Access Memory (DRAM) (main memory) sockets, and average power draw of a GPU during training multiplied by the number of GPU. The greenhouse gas emission equivalent per kilowatt-hour is then calculated based on data provided by the U.S. Environmental Protection Agency (EPA). Google research team R. So et al. [8] have evaluated Large Transformer models which have been central to recent advances in NLP and have developed a more efficient variant with a smaller training cost than the original transformer and other variants for auto-regressive language modelling. Patterson et al. [9] have calculated the energy use and carbon footprint of several recent large models and found that large but sparsely activated Deep Neural Networks (DNNs) can consume less energy than the large, dense DNNs without sacrificing accuracy despite using as many or even more parameters. The geographic location and specific data center infrastructure matters to reduce the greenhouse gas emission equivalent of Machine Learning (ML) workload. Patterson et al. [10] have shared interesting information that the inference represents about 3/5 of total ML usage at Google across three years, due to the many billion-user services that use ML. The combined emissions of training and serving need to be minimized. Lacoste et al. [11] have developed a tool called “Machine Learning Emissions Calculator” for ML community to approximate the environmental impact of training ML models. Ligozat and Luccioni [12] have proposed a practical guide to quantifying carbon emissions for ML researchers and practitioners. To analyse the carbon impact of ML, besides the ML model emissions of greenhouse gas due to the power consumption incurred by the equipment at the running time, other dimensions of model impact should be considered such as model preparation overhead, static consumption of the equipment, infrastructure, as well as the overall life cycle analysis of the equipment. The authors also have suggested the most important steps to take for practitioners and institutions for example, as an institution, deploying computation in low-carbon regions, providing institutional tools for tracking emissions, capping computational usage, carrying out awareness campaigns, and facilitating institutional offsets.

Standardizations have begun to tackle the subject, and for example, a new work item proposal has been under discussion in the Joint Technical Committee on Artificial Intelligence (JTC 21) of European Committee for Standardization (CEN) and European Committee for Electrotechnical Standardization (CENELEC). The new work item is about “Green and sustainable AI” which will establish a framework for quantification of the environmental impact of AI and its long-term sustainability and encourage AI developers and users to improve the efficiency of AI use [13]. The “CEN/CENELEC standardization landscape for energy management and environmental viability of green datacenters [14]” defines Key Performance Indicators (KPIs) that address energy and environmental control. However, these KPIs are focused on datacenters and currently do not address the distributed or IoT energy and environmental control. These aspects should be addressed in the AI standards also including sustainable development goals [15].

As discussed before, quantifying greenhouse emissions for any AI system is very important and several simplified and applicable methods have been developed in recent studies. Some open-source tools are available applying and integrating these methods. These tools have been classified into three major categories: priori measurement tools which usually calculate operational points in training and inference; on-the-fly measurement tools which measure power consumption, etc., when an AI system is running on hardware; posteriori measurement tools refer to these tools to approximate greenhouse gas emissions for a given computation. In our experiments, we deep dive into PowerAPI [16], PyJoules [17], and other open-source tools, such as Keras-flops [18], Torchstat [19], torchsummaryX [20], Flops-counter [21], JouleHunter [22], Jtop [23], CarbonAI [24], MLCO2 [11] and Green Algorithm [25], in order to have first-hand experience and to understand their capabilities and limitations.

Most of the recent research work focuses on the environmental impact of ML models at the training stage. As [10] mentioned, the energy consumed at the inference stage was more than the energy consumed at training for a given few years. So our idea is to set up a framework to evaluate the AI systems at both the training and inference stages. Considering large scale of AI systems is running on the edge side, we set up a heterogenous edge platform with various device types where we can use the proper orchestration tool that we have evaluated to deploy ML model on these different edge devices which somehow can simulate distributed AI applications deployed in real scenarios. Both X86-based and ARM-based hardware are used in the platform. We have designed methodologies to perform sets of experiments to measure the power consumption of the ML model incurred on hardware for the training stage and inference stage. Unlike a data center, the power consumed by these edge devices is mainly coming from CPU/GPU computation and memory usage with very limited overhead for cooling components if they have any. Even so, we have measured the static power consumption of edge devices to get a more precise measurement of AI model power consumption by subtracting the static power consumption. We also select various types of ML models for one AI use case and compare the power consumption of these ML models when they are running on edge devices in addition to their performance.

Our objectives for the studies are to verify the measurement tools and improve them; to obtain greenhouse gas emissions of various ML models; to benchmark performance vs environmental impact; and to develop greener ML models. These experimental results can provide useful information and recommendations for organizations to build an institutional toolbox to track greenhouse gas emissions of AI systems and offer responsible AI applications to our customers. The scope and key point of our analysis are the comparison of training and inference energy/CO2 consumptions among different AI models solving the same problem, not the comparison between the centralized server and edge devices.

The organization of this paper is as follows. Section 1 is an introduction; Sections 2 to 4 analyze the three categories of measurement tools respectively; Section 5 provides our experiments and results analysis, and Section 6 gives the conclusion.

II. Priori measurement tools analysis

To evaluate the ML model’s power consumption by comparing the model’s calculation amount, the priori measurement tools are employed. They are used to evaluate AI models and algorithms through computing the flops/mult-adds/other parameters.

There are two usages of priori measurement tools:

- as an inline module to measure the AI model, for example, first install as a python module, and then call some tools’ functions in the model source program to get the model's related computing information.

- as a Command Line Interface (CLI) tool to handle the model's source program, for example, first install the tool, and execute the tool to process the model source program to get the model's related computing information.

Because priori measurement tools handle the source code of AI programs, they always process one specific framework and support a subset of types of layers.

For testing priori measurement tools, a test environment was built, and related AI frameworks and some candidate priori measurement tools were installed first; then, constructed some demo AI models with/without special layers based on relevant frameworks; finally, computed and compared these demo AI models' Flops/Mult-Adds and other related measurements using candidate priori measurement tools and evaluated them through these computed results.

We launched five tests for four priori measurement tools: keras-flops, torchstat, torchsummaryX and flops-counter.

Keras-flops as a python module can calculate the FLOPs of neural network architecture written in Tensorflow. Test1 verifies keras-flops' support for Conv2dTranspose layer. In this test, we constructed a model including Conv2dTranspose layer to test this tool's capability with two python programs (with/without Conv2dTranspose Layer). The difference value of flops means that Conv2dTranspose layer is supported by keras-flops. Test2 verifies keras-flops' support for Conv3dTranspose layer. According to the supporting table, Conv3dTranspose layer is not supported by keras-flops. In this test, we constructed a model including Conv3dTranspose layer and test the tool's capability with two python programs (with/without Conv3dTranspose layer). The same value of flops means that Conv3dTranspose layer is not supported by keras-flops.

Torchstat is a lightweight neural network analyzer based on PyTorch. Its usage is as a python module to measure an AI model or as a CLI tool to handle a python program including an AI model. Test3 verifies Torchstat's support for Conv2d layer and ConvTranspose2d layer. In this test, we constructed Convolutional Neural Network (CNN) models including Conv2d layer and ConvTranspose2d layer to test the tool's capability.

TorchsummaryX is also a tool based on the Pytorch framework. This tool can handle Recurrent Neural Network (RNN), Recursive Neural Network, or models with multiple inputs. In the test4, we constructed two models with Conv2d layer and ConvTranspose2d layer respectively. We can find Mult-Adds remains unchanged. It means that Convtranspose2d layer is supported for Mult-Adds by torchsummaryX.

Flops-counter is based on the PyTorch framework and designed to compute the theoretical number of multiply-add operations in CNNs. It can also compute the number of parameters and print the per-layer computational cost of a given network. In the test5, we constructed two models with Conv2d layer and ConvTranspose2d layer respectively. We can find the computational complexity (i.e., number of multiply-add operations) remains unchanged. It means that Convtranspose2d layer is supported for Mult-Adds by torchsummaryX.

Torchstat, torchsymmaryX, and flops-counter are all based on the PyTorch framework, but their outputs are different. Torchstat outputs numbers of parameters, amount of Multiply+Adds, number of flops, and memory usage. torchsummaryX and flops-counter just provide numbers of parameters and amount of multiply+adds. According to some feedback from Internet, the results of torchstat's MAdd and FLOPs are wrong, which should be swapped. The summary of the four tools is shown in the following Table I.

TABLE I. SUMMARY OF FOUR PRIORI MEASUREMENT TOOLS

priori measurement tools	**support framework**	**outputs**
keras-flops	Tensorflow	FLOPs
torchsummaryX	PyTorch	FLOPs, Multi-Add, memory, total params
torchstat	PyTorch	Multi-Add, total params
flops-counter	PyTorch	Multi-Add, total params

Through our five tests, we can find that the effectiveness of priori measurement tools relies on their detailed implementation. The application of priori measurement tools is limited. The tools we tested just support one special framework (Tensorflow or PyTorch) and a subset of types of model layers. In practice, most AI models usually include some specific layers (e.g., 3D ConvTranspose layer) which cannot be calculated by our tested priori measuring tools.

III. ON-THE-FLY MEASUREMENT TOOLS ANALYSIS

The most direct and precise way to measure an AI program's power consumption is to measure it in real-time while the process is going on. We named this type of tool the "on-the-fly tool". For this purpose, we have carried out the research and study of relevant tools and later carried out the comparison test and verification of their real use situation.

After preliminary selection, we choose the following three measurement tools as candidates, they are PowerAPI series (JouleHunter, PyJoules), CarbonAI, and Jtop. They have their own methods and application scenarios, and following their official instructions and guidance documents, we conducted a series of tests and applications on them.

The first one is the PowerAPI, the goal of this project is to provide a set of tools to go forward greener computing, the idea is to provide software-defined power meters to measure the power consumption of the program, the core of this project is the PowerAPI toolkit for building such power meters [29].

PowerAPI is a middleware toolkit for building software-defined power meters. Software-defined power meters are configurable software libraries that can estimate the power consumption of software in real-time. A power meter built on PowerAPI normally has two components -- the sensor and the formula. The sensor is also a software, which worked like the physical world sensor, queries the hardware's (host machine) data, and collects raw data correlated with the power consumption of the software. All data will be stored in an external database to make the data available to the formula. For the other component, the formula is a computational module that uses the collected data to determine power consumption. Both are connected by a database that is used to transfer information. The global architecture of a power meter is represented in the Figure 1 below [26].

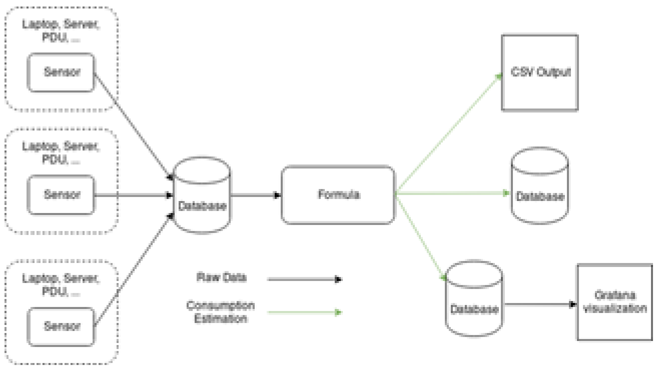


Figure 1. The global architecture of a power meter

For convenience and quick use, PowerAPI has provided several useful components. As Hwps-Sensor (Hardware Performance Counter), is a tool using the Running Average Power Limit (RAPL) technology to monitor the Intel CPU performance counter and power consumption of the CPU. Also, some matched formulas like “SmartWatts Formula” used for physical Linux machine, “VirtualWatts” used for a virtue machine, etc.

PowerAPI also packages up (with sensor and formula) a set of ready-to-use tools for diverse needs. Here Joule Hunter and PyJoules are the two we selected and used for our research.

JouleHunter runs on Linux machines with Intel RAPL support. This technology has been available since the Sandy Bridge generation [27]. JouleHunter can show what part of your program code is consuming considerable amounts of energy in detail. JouleHunter works similarly to pyinstrument [28], as it forked the repo and replaced time measuring with energy measuring. This tool can be easily installed and used with one or two command line(s), two key components of hardware the intel CPU and ram’s power consumption can be printed out. However, from its official documentation and real test we can see that JouleHunter this tool has its limitation, such as it only worked with Linux OS and no calculation for GPU power consumption.

Another software toolkit from PowerAPI is the PyJoules, which can be used to measure the energy footprint of a host machine along with the execution of a piece of Python code. Except for Intel CPU socket package and RAM (only for Intel server architectures), it also can monitor the energy consumed by the GPU of the host machine, supporting both for Intel integrated GPU (for client architectures) and Nvidia GPU (Uses the Nvidia "Nvidia Management Library" technology to measure the power consumption of Nvidia devices. The energy measurement Application Programming Interface (API) is only available on Nvidia GPU with Volta architecture 2018) [29]. PyJoules can only work with AI program coding on Python, and it should be installed and imported as a function into the target main project python file. It will report the total power consumption during the code is running. Its results contain not only the target project’s power consumption, thus including the OS and other applications running at the same time if have. That means it calculates the global power consumption of all the processes running on the machine during this period. With PyJoules, to get the closest measure to the real power consumption of the measured program, we need to try to eliminate any extra programs (such as graphical interface, background running task, etc.) that may alter the power consumption of the host machine and keep only the code under measurement. Same as JouleHunter, PyJoules currently can only work on GNU/Linux, and does not support on Windows and MacOS.

CarbonAI is another project which aims to raise AI the developers’ awareness of the AI’s carbon footprint. Firstly, like PyJoules it provides a python package that allows developer to monitor power consumption. Then based on the measurement results, CarbonAI will do a transition between power consumption and CO2 emissions, to provide a more intuitive understanding of how much our AI development is doing to the environment. For example, training an AI model for 100 rounds is the equivalent of driving from Paris to Marseille. Also, the power consumption results of CarbonAI are given as a CSV file, which includes most key devices of a host machine, like CPU, GPU, and RAM, but also the name of the country where the package was used (based on the IP or what the user set). So, the amount of CO2 emitted by the usage will be depends on the country and the energy mix used by the country to produce electricity. For combability, a different form that PyJoules only support Linux OS, CarbonAI package is compatible with most platforms (Linux, Windows, and MacOS) with the varying installation process.

At present, apart from x86 architecture servers and devices, ARM-based devices are also widely used in various fields of AI. However previous tree tools only work well on x86 hardware platforms, all of them will get several issues or bugs. For ARM platforms, Nvidia provides their official tool Jtop for the Jetson series, a platform designed for AI development and use cases. Jtop is one of jetson-stats, a package for monitoring and controlling NVIDIA Jetson (Xavier NX, Nano, AGX Xavier, TX1, TX2) Works with all NVIDIA Jetson ecosystems. Jtop can be run independently and show the real-time usage data of CPU, GPU, and RAM and also the actual frequency of the hardware. With its built-in graph user interface, we can easily read the results, but only the immediate frequency, so for the final power consumption we need manually calculate the Total power consumption using w=p*t, and “t” is the duration of the target AI program. Compared to other tools, although the result of power consumption cannot be directly obtained, Jtop can provide the usage rate of each device.

All those tools are not very difficult to install and use, some of them can be installed with several command lines, like JouleHunter, CarbonAI, and Jtop; and for PyJoules, we can add it into our application code just like a function. For compatibility, except CarbonAI supports Linux, Windows and MacOS (we only use it on Linux machines), other tools currently can only be used on Linux. Most tools currently only support Intel CPU and RAM, for GPU's power consumption, we need external components or 3rd part tools. For the programming language, most tools are built up as a python package, so they only worked with AI apps coded with python.

For the usage, the reported power consumption is not only the power consumption of the code you are running. This includes the global power consumption of all the processes running on the machine during this period, thus including the OS and other applications. So, we need to eliminate any extra programs and get the value of the devices when idling as the base level if possible. This will give the closest measure to the real power consumption of the measured code.

IV. POSTERIORI MEASUREMENT TOOLS ANALYSIS

AI researchers also proposed to estimate carbon emissions of the AI computation by posteriori tools, i.e., ML CO2 Impact and Green Algorithms. The key methodology of the tools is to estimate the power consumption after the computation process and achieve the carbon emissions from power consumption and related carbon intensity.

As shown in Table II, ML CO2 Impact tool is designed to estimate the carbon emissions produced by training ML models. The inputs include the geographical zone of the server, the type of GPU, and the training time, and the output is the approximate amount of CO2e. The inventors collected available public data for the computation including the Thermal Design Power (TDP) of the hardware, the location of the hardware, and the related carbon intensity (CO2e emissions per kWh).

TABLE II. ML CO2 IMPACT AND GREEN ALGORITHMS

	ML CO2 Impact	**Green Algorithms**
Energy consumption	runtime * power draw for GPU	runtime * (power draw for cores * usage + power draw for memory) * PUE * PSF
Hardware type	Mainly GPU type	GPU, CPU, CPU/GPU co-existing case, number of cores, memory
Usage factor	100% by default	100% by default and configurable
Other factors	/	Power Usage Effectiveness: the extra energy needed to operate the data center (cooling, lighting , etc.) Pragmatic Scaling Factor: multiple identical runs (e.g. for testing or optimization)

Green Algorithms tool aims to estimate the carbon footprint of any computational task. Compared with ML CO2 Impact tool, it requires extra inputs of memory size, real usage factor of the processing core, PUE, and Pragmatic Scaling Factor (PSF).

Different from the on-the-fly measurement tools, the power consumption model of both tools uses TDP and runtime to achieve the power consumption, which means in the calculation the usage of cores is 100% by default. Green Algorithms tool allows to configure the real usage of cores and takes more quantifiable elements into consideration, i.e., memory power, PUE, and PSF, allowing users to estimate the power consumption more flexibly.

Considering carbon intensity, it is known that fossil fuels have the highest carbon footprints, for example, coal emits 820g of CO2e per kWh of electricity produced [30], while electricity generated by wind, solar, hydro, or nuclear power emits lower amounts of carbon footprints, i.e., 12g CO2e /kWh for wind, and 27~48g CO2e /kWh for all types of solar. In different countries and regions, even different electric power companies, the energy structure differs from others, and various energy sources would be used to generate electricity. The location matters as all servers are connected to local grids and they will have different amounts of CO2e emissions when consuming or generating the same amount of electricity, for example, 174g CO2e emission per kWh of electricity for France and 741g CO2e /kWh for Germany (at 12:00 PM on December 1, 2022) [31].

Both tools refer to public data for carbon intensity. They provide data reference of data centers including Google Cloud Platform, Amazon Web Services, and Azure. However, we can see clearly that there is no unified data source, location scope, and effective time due to differences in the data sources of the two tools.

From the preliminary analysis of the above two tools, we know that when evaluating carbon emission impact, both power consumption and carbon intensity should be considered. Parameters like hardware type, PUE, the usage of the core, and memory will contribute to the energy consumption. For carbon intensity, the location matters because of the different energy mix in different countries and regions, but there are various data sources that may provide quite different location scopes and effective time. Also, we notice that the goal of these tools is to make people aware of the carbon emission impact, to provide a quick tool to evaluate the carbon emission during machine learning work and to recommend carbon reduction actions like selecting the cloud provider or server location wisely, buying carbon offsets, choosing clean energy, and improving AI algorithms to be green.

V. EVALUATION EXPERIMENTS

Our objectives are to verify the measurement tools and have some comments and suggestions proposed for measurement tools for AI models through our experiments.

We have developed a systematic methodology to carry out our experiments. First, a cloud-edge platform was set up with heterogenous hardware either X86-based, or ARM-based. These hardware devices have similar levels of computation capabilities to commercial end devices such as LiveBox, controllers on vehicles, etc. Then, we selected an AI application, in our case, we choose Person Re-Identification (Re-ID) which is the task of associating the same person taken from different cameras or from the same camera on different occasions [32]. Person Re-ID have wide usage in smart building and smart city scenario. Many Person Re-ID open-source models are accessible using various AI architectures, such as CNN, Transformer, or Long Short Term Memory (LSTM). Based on criteria, such as performance, release date, accessibility, etc., we have selected several Person Re-ID models with different AI model architectures. After that, we measured their power consumption during the training stage and inference stage when they are running on various types of hardware in rounds of experiments.

We build a benchmark to compare the results of the on-the-fly and posteriori measurement tools. In the first experiment, the power consumption of Fast-ReID (CNN) model is measured by processing AI inference on a 500s video of a single person. The results of PyJoules and Jtop tools are selected as a baseline of the real-time measured power consumption. MLCO2 Impact and Green Algorithms tools are used to estimate the power consumption afterward, respectively. As is shown in Table III, three types of hardware have been evaluated: a server with GeForce GTX 1080 Ti, Intel Xeon E5-2678 v3 and a memory of 64GB, and two edge devices – one with Intel i7-8559U and a memory of 16GB, and the other with NVIDIA Jetson AGX Xavier, ARMv8 Processor rev 0 (v8l) and a memory of 32GB. For Green Algorithms tool, there is a default CPU usage of 100% and a configurable CPU usage that can be estimated based on the observation of the AI processing experiment. The PUE used in the calculation is 1 because we use local private infrastructure instead of cloud services and ignore the power consumption of cooling or lighting. The memory power draw only depends on the size of memory available (0.3725 W per GB).

In the second experiment, as is shown in Table IV, four different Re-ID models are evaluated at the training stage: Fast-ReID (CNN), st-ReID (CNN), DeepPerson (LSTM), and Trans-ReID (Transformer). Since both GPU and CPU cores will be used in the AI training process, power consumption should be considered for both.

VI. Conclusion

We have been carrying out series of experiments to verify the measurement tools. For different types of measurement tools, we found that:

The effectiveness of priori measurement tools relies on their detailed implementation. The application of priori measurement tools is limited. The tools just support one special framework and a subset of types of model layers.

The on-the-fly tools can be used during the processes of AI programs; however, they are limited. PyJoules or JouleHunter can be used to get power consumption (CPU, GPU, RAM) of large AI programs on different x86 architectures devices, while for architectures ARM devices, only Jtop supported. Ideally, it is better to develop and use the same cross-platform tool. However, the comparison of experimental and estimated results shows that the error of the on-the-fly measurement tools is acceptable.

The posteriori measurement tools can be used for power consumption estimation after the AI processing by knowing the runtime and the parameters of hardware (CPU, GPU, memory, etc.). For resource-constrained edge devices, the resources usually tend to be nearly full of use, and the tools with a default configuration are able to make a quick estimation of the power consumption. For servers that have more resources and stronger processing capabilities, if extra information can be given, for example, the real usage of the cores, the Green Algorithms tool will be optimized to make close estimations to real-time measured power consumption. Both tools can provide different CO2e emissions due to different locations where the AI computation is processed. The researchers aim to remind people to carefully select the cloud providers and locations for AI services when carbon impacts should be taken into consideration.

We have selected an AI use case: Re-ID which can be realized by various types of AI architecture: CNN, LSTM, and Transformer. Once the specific AI model for each type is selected, the power consumption of the selected AI models is measured during the training and inference stages when they are running on different edge devices. The experimental results show that the total training power consumption of the AI model is determined by the training algorithm and training time. Training power consumption is in proportion to the training time in general. With the measurement tool, it can quantify the power consumption to make a more accurate assessment for different AI models. The power consumption of AI applications is positively correlated with the complexity of application scenarios when the hardware capability allows.

Our plan includes continuously investigating and improving the measurement tools and verifying them in experiments. With these tools, researchers and scientists may be able to design more power-efficient AI models without sacrificing model performance.

References

[1] WiKi climate change. [Online]. Available from: https://en.wikipedia.org/wiki/Climate_change/ [retrieved: 05, 2023].

[2] H. Ritchie, “Sector by sector: where do global greenhouse gas emissions come from?”. [Online]. Available from: https://ourworldindata.org/ghg-emissions-by-sector/ [retrieved: 05, 2023].

[3] Ericsson, “ICT’s potential to reduce greenhouse gas emissions in 2030”. [Online]. Available from: https://www.ericsson.com/en/reports-and-papers/research-papers/exploring-the-effects-of-ict-solutions-on-ghg-emissions-in-2030/ [retrieved: 05, 2023].

[4] Matleena, “Amazing AI Statistics (2022): Stunning Growth of AI”. [Online]. Available from: https://zyro.com/blog/ai-statistic/ [retrieved: 05, 2023].

[5] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, “Green AI”, Communications of the ACM, vol. 63, No. 12, pp. 54-63, Dec. 2020.

[6] OpenAI, http://openai.com/ [retrieved: 05 2023].

[7] E. Strubell, A. Ganesh, and A. McCallum, “Energy and Policy Considerations for Deep Learning in NLP”, 57th Annual Meeting of the Association for Computational Linguistics (ACL). Florence, Italy. Jul. 2019, doi: 10.18653/v1/P19-1355.

[8] D. R. So, et al., “Primer: Searching for Efficient Transformers for Language”, 35th Conference on Neural Information

Processing Systems (NeurIPS 2021), virtual, 2021, doi: 10.48550/arXiv.2109.08668.

[9] D. Patterson, et al., “Carbon Emissions and Large Neural Network Training”, 2021, doi: 10.48550/arXiv.2104.10350.

[10] D. Patterson , et al., “The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink”, Computer, vol. 55, pp. 18-28, Jul 2022.

[11] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, “Quantifying the Carbon Emissions of Machine Learning”, 2019, doi: 10.48550/arXiv.1910.09700.

[12] A. Ligozat and S. Luccioni, “A Practical Guide to Quantifying Carbon Emissions for Machine Learning researchers and practitioners”, [Online]. Available from: https://hal.archives-ouvertes.fr/hal-03376391/document/ [retrieved: 05, 2023].

[13] CEN-CENELEC JTC 21 “Artificial Intelligence”, [Online]. Available from: https://www.cencenelec.eu/areas-of-work/cen-cenelec-topics/artificial-intelligence/ [retrieved: 05 2023].

[14] CEN/CENELEC, “Standardization landscape for energy management and environmental viability of green data centres”, [Online]. Available from: ftp://ftp.cencenelec.eu/EN/EuropeanStandardization/HotTopics/ICT/GreenDataCentres/GDC_report_summary.pdf [retrieved: 05 2023].

[15] Walshe, R., Casey, K., Kernan, J., Fitzpatrick, D., “AI and big data standardization: Contributing to United Nations sustainable development goals”, Journal of ICT Standardization, pp. 77–106, 2022.

[16] PowerAPI, http://www.powerapi.org/ [retrieved: 05 2023].

[17] PyJoules, https://github.com/powerapi-ng/pyJoules/ [retrieved: 05 2023].

[18] Keras-flops, https://github.com/tokusumi/keras-flops/ [retrieved: 05 2023].

[19] Torchstat, https://github.com/Swall0w/torchstat/ [retrieved: 05 2023].

[20] torchsummaryX, https://github.com/nmhkahn/torchsummaryX/ [retrieved: 05 2023].

[21] flops-counter, https://github.com/sovrasov/flops-counter.pytorch/ [retrieved: 05 2023].

[22] JouleHunter, https://github.com/powerapi-ng/joulehunter/ [retrieved: 05 2023].

[23] Jtop, https://pypi.org/project/jetson-stats/ [retrieved: 05 2023].

[24] CarbonAI, https://github.com/Capgemini-Invent-France/CarbonAI/ [retrieved: 05 2023].

[25] L. Lannelongue, J. Grealey, and M. Inouye, “Green algorithms: quantifying the carbon footprint of computation”. Advanced science, vol 8, issue 12, 2021, doi: 10.48550/arXiv.2007.07610.

[26] Global architecture picture, https://raw.githubusercontent.com/powerapi-ng/powerapi ng.github.io/master/images/powerAPI_archi.png/ [retrieved: 05 2023].

[27] Sandy Bridge generation, https://fr.wikipedia.org/wiki/Intel#Historique_des_microprocesseurs_produits/ [retrieved: 05 2023].

[28] Pyinstrument, https://pyinstrument.readthedocs.io/ [retrieved: 05 2023].

[29] Volta architecture, https://en.wikipedia.org/wiki/Volta_(microarchitecture)/ [retrieved: 05 2023].

[30] https://www.world-nuclear.org/ [retrieved: 05 2023].

[31] https://app.electricitymaps.com/map/ [retrieved: 05 2023].

[32] Person-re-Identification, https://paperswithcode.com/task/person-re-identification [retrieved: 05 2023].

TABLE III. FAST-REID, SINGLE PERSON, RUNNING TIME = 500S, AI INFERENCE

Measurement tools	**GPU/CPU type**	**Power consumption**				**Carbon emissions**[a]
		CPU (W)	***Usage***	***Memory (GB)***	***Total (Wh)***	***CO2e(mg)***
1 On the fly - PyJoules	Server:				7.2	
2 A posteriori - MLCO2 Impact	GPU: GeForce GTX 1080 Ti	120	100%		16.67	633.46
3 A posteriori – Green Algorithms	CPU: Intel Xeon E5-2678 v3 Memory: 64GB	120	30%/100%	64	8.31/19.98	426.18/ 1002
1 On the fly - PyJoules	Intel machine II:				4.1	
2 A posteriori - MLCO2 Impact	CPU: Intel i7-8559U	28	100%		3.89	147.82
3 A posteriori – Green Algorithms	Memory: 16GB	28	70%/100%	16	3.55/4.72	182.04/ 241.86
1 On the fly - Jtop	ARM:				3.8	
2 A posteriori - MLCO2 Impact	NVDIA Jetson AGX Xavier	30	100%		4.17	158.46
3 A posteriori – Green Algorithms	CPU: ARMv8 Processor rev 0 (v8l) Memory: 32GB	30	70%/100%	32	4.57/5.82	234.45/ 298.55

a. The reference location is Europe, France.

TABLE IV. FAST-REID, ST-REID, DEEPPERSON, AND TRANS-REID, AI TRAINING

Measurement tools	**Test ML model**	**Running time(s)**	**Power consumption**						**Carbon emissions**[a]
			GPU (W)	***Usage***	***CPU (W)***	***Usage***	***Memory (GB)***	***Total (Wh)***	***CO2e(g)***
1 On the fly - PyJoules	Fast-ReID,	4625						125.06	
2 A posteriori - MLCO2 Impact	CNN		120	100%	45	100%		211.98	8.06
3 A posteriori – Green Algorithms			120	66%/100%	45	10%/100%	64	128.16/242.61	7.08/12.44
1 On the fly - PyJoules	st-ReID,	7988						212.26	

2 A posteriori - MLCO2 Impact	CNN		120	100%	45	100%		366.12	13.91
3 A posteriori – Green Algorithms			120	66%/100%	45	10%/100%	64	238.62/419.01	12.24/21.49
1 On the fly - PyJoules	DeepPerson, LSTM	8326						201.74	
2 A posteriori - MLCO2 Impact			120	100%	45	100%		381.61	30.35
3 A posteriori – Green Algorithms			120	66%/100%	45	10%/100%	64	248.72/436.74	26.69/46.87
1 On the fly - PyJoules	Trans-ReID, Transformer	17426						451.7	
2 A posteriori - MLCO2 Impact			120	100%	45	100%		798.69	30.35
3 A posteriori – Green Algorithms			120	66%/100%	45	10%/100%	64	520.55/914.09	26.69g/46.87

a. The reference location is Europe, France.

A Formal Model for the Simulation of Mobile Networks

Emanuele Covino

Dipartimento di Informatica
Universitá degli Studi di Bari Aldo Moro
Bari, Italy
emanuele.covino@uniba.it

***Abstract*—We introduce MOTION (MOdeling and simulaTIng mObile ad-hoc Networks), a Java tool that simulates a well known protocol for mobile networks (the Ad-hoc On-demand Distance Vector - AODV); its definition is based on the Abstract State Machine formal model used within the framework ASMETA (ASM mETAmodeling). Morover, we suggest that some protocols for mobile networks could be used to provide a formal definition of social structures and to analyze the related properties.**

***Index Terms*—AODV, Abstract State Machines, Mobile ad-hoc networks, Mobile computing, Social network analysis.**

I. Introduction

Wireless communication among both stationary and mobile devices in absence of physical infrastructure can be established and performed by means of the Mobile Ad-hoc NETwork technology (MANET) [1] [22] [33]. While stationary devices cannot change their location in the network, mobile devices are free to move randomly, entering or leaving the network and changing their relative positions. Each device can broadcast messages inside its radio range only, implying that, outside this area, communication is possible by means of some sort of cooperation among intermediate devices, exclusively. Thus, a communication protocol capable of handling this lack of predictable topology is needed. One of the most popular routing protocols for MANET's is the Ad-hoc On-demand Distance Vector (AODV) [32], together with several variants introduced in order to reduce communication failures due to topology changes. For example, Reverse-AODV (R-AODV) [6] [25] builds all possible routes between source and destination devices: when the primary route fails (typically the shortest one), communication is still provided by the alternative routes. More recently, variants have been proposed to cope with congestion issues [12] [24] and to improve the security on communications, using cryptography to secure data packets during their transmission (Secure-AODV) [41], and adopting the so-called *trust methods*, in which nodes are part of the communication if and only if they are considered trustworthy (Trusted-AODV) [12] [26]. This research area is receiving more attention in the last few years, in the context of smart mobile computing, cloud computing and Cyber Physical Systems [15] [31].

MANET's technology raises several problems related to the analysis of performance, synchronization and concurrency of the network. Moreover, the request of computing services characterized by high quality levels, broad and continuous availability, and inter-operability over heterogeneous platforms, increases the complexity of the systems' architecture. Therefore, it is quite important to be able to verify qualities like responsiveness, robustness, correctness and performance, starting from the early stages of the system's development. In order to do this, many studies are executed with the support of simulators [3] [36] [39]. They can be used to measure and to evaluate performances and to compare different solutions, implementing the network at a low abstraction level but, by their intrinsic nature, they cannot support proofs of correctness, synchronization and deadlock properties, and they cannot model MANET's with a higher abstraction level of specification. To overcome this limitations, formal methods are used to create a model the system. For instance, the process-calculus [35], the Calculus of Mobile Ad Hoc Networks (CMN) [28], and the Algebra for Wireless Networks (AWN) [14] capture essential characteristic of nodes, such as mobility or packets broadcasting. Petri nets have been employed to study the modeling and verification of routing protocols [40], and the evaluation of protocols performances [13].

This kind of state-based models provide a suitable way of representing algorithms, and they are typically equipped with tools (such as CPN Tools [23]) that allow to simulate the algorithms, directly. However, they lack expressiveness, because they only show a single level of abstraction, and they do not provide simple ways for refinements of the executable code. These characteristics are intrinsic in the Abstract State Machine model (ASM) that provides a way to describe algorithms in a simple abstract pseudo-code, which can be translated into a high-level programming language source code [5] [17]. Formal methods are satisfactory for reasoning about properties of the system they describe, but they rarely are useful for studying performance results [8].

In this paper, we use the ASM formalism to define a MANET and to simulate its behaviour; this is achieved by introducing MOTION (MOdeling and simulaTIng mObile ad-hoc Networks), a tool operating within the framework ASMETA (ASM mETAmodeling) [2] [16]. In particular, we adopt the AODV protocol to manage the evolution of the network and to show the behaviour of the application. In Section II, we recall concepts and definitions of mobile ad-hoc networks and of the specific protocol adopted in order to capture the dynamic behavior of nodes in the network. In Section III, we recall the basics about Abstract State Machine's [4] [5]. We will use this formalism to define and study properties of the network. In Section IV, we outline the definition and behaviour of MOTION, implementing the previous protocol by means of the ASM's formalism. In Section V, we discuss how the mobile networks' model could be used to represent social groups and to study the related interactions (for instance, those occurring within social networks). Conclusions and future work can be found in Section VI.

II. Mobile Ad-hoc NETworks and a routing protocol

Networks of mobile nodes, usually connected by means of a wireless communication system, have been dubbed MANET. Each node of the network can be considered as an autonomous agent that re-arranges its position without conforming to a fixed topology. During its lifetime it can enter or leave the network, and it can change its position, continuously; this means that routes connecting the nodes can rapidly change, because of their mobility and of the limited range of transmission. When a piece of information has to find his path from a source node towards a destination, a routing protocol is needed. In general, a routing protocol specifies how nodes communicate with each other in order to distribute the information within the network; routing algorithms determine this choice, according to some specific principle, and they are able to adjust the route when changes occur, such as disabled or partially available connections, loops, obstructions, or starvation.

Several routing protocols have been proposed; among them, the *Ad-hoc On-demand Distance Vector* (AODV) [32] is one of the most popular (indeed, a number of simulation studies are dealing with it, representing a reliable baseline for comparison to the results of simulations executed with MOTION). It is a reactive protocol that combines two mechanisms, the *route discovery* and the *route maintenance*, in order to store some knowledge about the routes into *routing tables*. Each node has its own routing table that consists of a list of all the discovered (and still valid) routes towards other nodes in the network; in particular, the routing table entry of the node i concerning a node j includes the *address* of j, the last known *sequence number* of j, the *hop count* field (a measure of the distance between i and j), and the *next hop* field (identifying the next node in the route between i and j). The sequence number is an increasing integer maintained by each node that expresses the freshness of the information about the respective node. When an *initiator node* wants to start a communication session towards the *destination node*, it checks if a route is currently stored in its routing table. If this happens, the communication can start. If there aren't any routes to the destination, the initiator sends a *route request* (RREQ) to all its neighbors. This message includes the initiator address, the destination address, the sequence number of the destination (i.e., the most recent information about the destination), and the hop count, initially set to 0, and increased by each intermediate node. When an intermediate node N receives an RREQ, it creates a routing table entry for the initiator, or, if the entry already exists, it updates its sequence number and next hop. Then, the process is iterated: N checks if it knows a route to the destination with corresponding sequence number greater than the number contained into the RREQ (this means that its knowledge about the route is more recent). If so, N sends back to the initiator a *route reply* (RREP); otherwise, N updates the hop count field and broadcasts once more the RREQ to all its neighbors. The process ends successfully when a route to the destination is found. While the RREP travels towards the initiator, the routing tables of the traversed nodes are updated, creating an entry for the destination, when needed. Once the initiator receives back the RREP, the communication can start. The mobile nature of the nodes can create new routes or break some of them, because new links are established betweens pairs of nodes or because one or more links are no more available; when this happens, a route maintenance is executed in order to notify the error and to invalidate the corresponding routes, propagating a *route error* (RERR) into the network.

III. Abstract State Machines

An ASM [5] M is a tuple (Σ, S, R, P_M). Σ is a *signature*, that is, a finite collection of names of total functions; each function has arity n, and the special value *undef* belongs to the range (*undef* represents an undetermined object, the default value). Relations are expressed as particular functions that always evaluate to *true*, *false* or *undef*.

S is a finite set of *abstract states*. The concept of abstract state extends the usual notion of state occurring in finite state machines: it is an algebra over the signature Σ, i.e., a non-empty set of objects together with interpretations of the functions in Σ. Pairs of function names, together with values for their arguments, are called *locations*: they are the abstraction of the notion of memory unit. Since a state can be viewed as a function that maps locations to their values, the current configuration of locations, together with their values, determines the current state of the ASM.

R is a finite set of *rule declarations* built starting from the *transition rules* skip, update $(f(t_1, t_2, \ldots, t_n) := t)$, conditional (**if** ϕ **then** P **else** Q), let (**let** $x = t$ **in** P), choose (**choose** x **with** ϕ **do** P), sequence (P **seq** Q), call $(r(t_1, \ldots, t_n))$, block (P **par** Q) (see [5] for their operational semantics). The rules transform the states of the machine, and they reflect the notion of transitions occurring in traditional transition systems. A distinguished rule P_M, called the *main rule* of the machine, represents the starting point of the computation.

A *move* of a ASM, in a given state, consists of the simultaneous execution of all the rules whose conditions evaluates to true in that state. Since different updates could affect the same location, it is necessary to impose a consistency requirement: a set of updates is said to be *consistent* if it contains no pairs of updates referring to the same location. Therefore, if the updates are consistent, the result of a move is the transition of the machine from the current state to another; otherwise, the computation doesn't produce a next state. A *run* is a (possibly infinite) sequence of moves: they are iterated until no more rules are applicable.

The aforementioned notions refer to the *basic* ASMs. However, there exist some generalisations (e.g., Parallel ASMs and Distributed ASMs) [17]. Parallel ASMs are basic ASMs enriched with the rule forall x with ϕ do P, to express the simultaneous execution of the same ASM P over x satisfying the condition ϕ. A Distributed ASM is intended as a finite number of independent agents, each one executing its own underlying ASM: it is capable of capturing the formalization of multiple agents acting in a distributed environment. A run, which is defined for sequential systems as a sequence of computation steps of a single agent, is defined as a partial order of moves of finitely many agents, such that the three conditions of co-finiteness, sequentiality of single agents, and coherence are satisfied. Roughly speaking, a global state corresponds to the union of the signatures of each ASM together with interpretations of their functions.

IV. Defining MANET's by means of ASM

In [9], we have given a description of a MANET's behaviour based on the parallel ASM model and we have introduced a preliminary version of MOTION that allows to define its parameters (such as mobility and level of activity of a node), to run the network, and to collect the output data of the simulation. In this paper, we provide a refinement that allows the user to visualize the dynamic evolution of the network, step by step: the mobility of nodes within the network, the path from a source to a destination and the overall evolution of the network can be monitored and studied. The complete package can be found in [21].

MOTION is developed within the ASMETA framework [16]; the behaviour of the network is modelled using the AsmetaL language, and then the model is executed by the AsmetaS simulator. The executions of MOTION and ASMETA are interleaved: first, MOTION captures the parameters of the network (number of nodes and their level of mobility, for instance) and includes them into an AsmetaL file; then, it runs AsmetaS according to those parameters. AsmetaS executes an ASM move, simulating the behavior of the protocol over the current network's configuration. The control goes back to MOTION at the end of each move: the information related to the move (such as the new positions of the nodes, the sent/received requests, the relations among the nodes) are recorded and, in this new version, the current topology of the network is visualised (showing the successful communication attempts between pairs of nodes, the connections established, and the failed attempts). Then, MOTION invokes AsmetaS for the next move. At the end of the simulation, MOTION reads the final log file, parses it, and stores the collected results in a csv file. Note that these interleaved calls require a considerable amount of interaction work; this is done in order to collect the information about the evolution of the network step by step, and to use it for the analysis of the behaviour of the network itself.

In more details, MOTION expresses the network topology by means of an *adjacency matrix C*, such that $c_{ij} = 1$ if i and j are neighbors, 0 otherwise, for each pair of nodes i and j. The mobility of nodes is implemented by updating the adjacency matrix at every step of the simulation; each c_{ij} is randomly set to 0 or 1, according to a mobility parameter defined by the user. The new values of the matrix are used to execute the next ASM move, accordingly. The relations among nodes are expressed by means of predicates, as expected: for instance, the reachability between two agents a_i and a_j is expressed by the predicate isLinked(a_i, a_j), which evaluates to *true* if there exists a coherent path from a_i to a_j, to *false* otherwise; the predicate knowsActiveRouteTo(a_j, a_j) states that a_i has an active path leading to a_j into its routing table.

The AODV routing protocol has been formally modeled through ASMs in [4], for the first time. MOTION redefines the protocol by means of new predicates and rules, also adding a parameter *Timeout*, the waiting time for the route reply, to avoid infinite loops when searching for a route. Each node of the network represents a device or an agent. In what follows, we show some of the high-level rules of MOTION (notice the use of **forall** in order to run AODVSPEC on every node in the network, and to look for a route from a given source a to the remaining nodes dest).

```
MAIN RULE AODV =
    forall a ∈ Nodes do AODVSPEC(a)


AODVSPEC(a)=
  forall dest ∈ Nodes with dest ≠ a do
    if WaitingForRouteTo(a, dest) then
      if Timeout(a, dest) > 0 then
        Timeout(a, dest) := Timeout(a, dest)-1
      else
        par
          WaitingForRouteTo(a, dest) := false
          ca-fail(a, dest) := ca-fail(a,dest)+1
        endpar
      endif
  if WishToInitiate(a) then PREPARECOMM(a)
  if not Empty(Message) then ROUTER
```

WaitingForRouteTo expresses that the discovery process previously started is still running. In this case, if the waiting time for RREP is not expired (i.e., Timeout() $>$ 0), the time-counter is decreased; otherwise, the search for the route is ended. If *WishToInitiate* evaluates to true (depending on a *initiator probability* parameter), the node wants to start a communication, and the following rule PREPARECOMM is called.

```
PREPARECOMM(a) =
 forall dest ∈ Nodes with dest ≠ a do
   choose wantsToCommWith ∈ Boolean with true do
```

```
if wantsToCommWith then
    par
        if not waitingForRouteTo(a,dest) then
            ca-tot(a, dest) := ca-tot(a, dest) + 1
        endif
        if knowsActiveRouteTo(a,dest) then
            par
              StartCommunicationWith(dest)
              waitingForRouteTo(a, dest) := false
            endpar
        else
            if not waitingForRouteTo(a, dest) then
              par
                GenerateRouteReq(dest)
                WaitingForRouteTo(a, dest) := true
                Timeout(a,dest) := Timeout
              endpar
            endif
        endif
    endpar
endif
```

Finally, if the node has received a message (either RREQ, RREP or RERR), ROUTER is called, with

```
ROUTER = ProcessRouteReq;
         ProcessRouteRep;
         ProcessRouteErr
```

where each sub-rule expresses the behavior of the node, depending on the type of the message received. Thanks to this formalization, some properties have been proven in the past, such as the starvation freeness for the protocol, the properness of the message received back by the initiator of any communication, and the capability to intercept black holes into the network.

An actual simulation in MOTION is performed in a number of sessions established by the user (10 sessions, Figure 1), each of which has a duration (50 moves, Figure 1); during each session, the network has a number of agents (hosts) defined by the user. Each agent tries to initiate a communication towards a destination: the probability that one of them acts as an initiator is defined by setting the parameter *Initiator Probability* (10 per cent, Figure 1). Thanks to the intrinsic parallelism in the execution of the ASM's rules, more attempts can be executed simultaneously. A communication attempt is considered successful if the initiator receives an RREP within the waiting time expressed by the parameter *Timeout*; otherwise, the attempt is considered failed.

In MOTION, agents' mobility is defined by the user by means of two parameters, namely *Initial connectivity* and *Mobility level*. The former defines the initial topology of the MANET: it expresses the probability that each agent is directly linked to any other agent. During the simulation, the mobility of agents is expressed by the random re-definition of the values of the *adjacency matrix C*. More precisely, for each pair of agents (a_i, a_j), and for each move of the ASM, the values of C are changed with a probability expressed by *Mobility level*.

The new version of MOTION starts from an interface that allows to set the parameters of the network (Figure 2); in this case, six agents populate the network, with a high value of initial connectivity and a low level of mobility. The chance that an agent starts a communication is set to 20 per cent. When the simulation is started, some new dynamic windows are visualised, in contrast with the previous version of the tool. For instance, a step of the network evolution can be seen in Figure 3. The window *mobility model* represents the connectivity matrix, that is, the existing direct connections among nodes; because of the high initial connectivity, we can find a big number of *successful connections* and no *failed connections*. After several moves, Figure 4 shows a new mobility model, and a new set of successful or failed connections.

V. Social networks analysis

Social structures can be investigated by means of methods and tools of social network analysis. A model often used to represent these structures is a graph or network, that is, a collection of nodes connected by arcs; the former are associated with people or agents, while the latter represent any kind of relation, interaction or influence between pairs (or groups) of agents [30]. This idea has been applied in a large number of studies, about social media networks [18] [20], information circulation [19] [29], business networks, knowledge networks [7] [11]. In particular, social network analysis is a key technique in modern sociology, demography, communication studies, market economy, sociolinguistic, cooperative learning, being able to represent data by means of a simple data structure, a graph, and to analyze the intrinsic interactions using the standard methods and measures provided by mathematics and computer science [38]. The interest of scientists is surely driven by the availability of the so-called big data; between 1990 and 2005, the new (virtually) unbounded computational power has been applied to the concept of self-organizing systems, providing the definition of models and simulations of a big number of social activities. In the mid 1990s, physicist and mathematicians started to analyze big data from financial markets, resulting in the development of econophysics [27]; in the 2000s, the focus shifted on big data generated by the Internet and the social networks, looking for characteristic patterns that exists in social interactions, no matter the technology, and revitalizing the research in sociophysics [34] and in computational social sciences. Many studies are executed with the support of simulators that are suitable to compare different social structures and several scenarios, according to the parameters of the network.

In general, networks used to represent social interactions are static, meaning that the location of nodes and the related ties don't change as time goes by; every change that may happen in the social group is not captured by this model. Aside static networks, mobile networks exist: they have a flexible structure, and their topology changes dynamically, given that nodes can join or leave the network during their lifetime, that communication among them depends on the availability of a connection, and that connections can have different strength. This reflects the dynamic nature of ties that exists between agents in a social group. Computer science provides methods to define and represent these kind of networks, together with algorithms that allow to broadcast a message from a source to a destination, mimicking the spread of information, opinions, or consensus into the group. In order to do this, agents should behave according to a cooperation protocol. We suggest that the MANET models, as well as other models of mobile networks, could be used to represent a social group and to study the related interactions [10]. MOTION could be used by social scientists to represent and study social interactions. For instance, a high value of the *initial connectivity* parameter, together with a low level of *mobility*, represent strong ties within a very cohesive group, meaning that the members of the group do not change their opinion or do not end a relation easily. On the contrary, a high mobility means that the group is prone to change opinions very easily. The *initiator probability* measures how much a member of a social group is inclined to spread information inside the network. It appears that the properties of a MANET match the properties that can be found in a social group, like starvation of information, fake information spreading, popularity of opinions, and so on. One could follow the propagation of a message (an opinion, an influence) inside the social group that is represented by the network, and to study how this is affected by the mobility of the agents or by the strength of the ties inside the group itself.

VI. Conclusions and future work

MANET is a technology used to perform wireless communications among mobile devices in absence of physical infrastructure. It is widely used in the context of smart mobile computing, cloud computing and Cyber Physical Systems. Several routing protocols have been developed, and problems have been raised about the measurement of performances, and also about the formal analysis of qualities like responsiveness, robustness, correctness. In order to address these problems, both simulators and formal description methods are needed. The former allow us to measure performances through direct simulation, but they aren't suitable to investigate the properties of the networks. This can be achieved when using formal methods, but they can hardly be used to measure performance. In this paper, we have introduced MOTION, a Java application in which MANET's are modeled as an Abstract State Machine by means of the AsmetaL representation. This representation can be used to prove formal properties of the network, as well as can be simulated by means of the simulation engine AsmetaS. MOTION can collect the results of this simulation that can be used for performances' analysis. We have validated MOTION on the Ad-hoc On-demand Distance Vector protocol.

Several variants of routing protocols for mobile networks have been proposed in the past; among them, the *NACK-based Ad-hoc On-demand Distance Vector* (N-AODV), that improves the awareness that each host has about the network topology, and the *Blackhole-free N-AODV* (BN-AODV), that detects the presence of malicious nodes leading to a blackhole attack.

One of the disadvantages of the AODV protocol is the poor knowledge that each node has about the network topology. In fact, each node n is aware of the existence of a node m only when n receives an RREQ, either originated by, or directed to m. In order to overcome this limitation, the NACK-based AODV routing protocol has been proposed and modeled by means of a Distributed ASM. A *Not ACKnowledgment* (NACK) control packet is added in the route discovery phase. Whenever an RREQ originated by n and directed to m is received by the node p that doesn't know anything about m, p unicasts the NACK to n, with the purpose to state the ignorance of p about m. In this way, n (as well as all the nodes in the path to it) receives fresh information about the existence and the relative position of p. Therefore, on receiving the NACK, all the nodes in the path to p add an entry in their respective routing tables, or update the pre-existing entry. N-AODV has been experimentally validated through simulations, showing its efficiency and effectiveness: the nodes in the network actually improve their knowledge about the other nodes and, in the long run, the number of RREQ decreases, with respect to the AODV protocol.

All routing protocols assume the trustworthiness of each node; this implies that MANET's are prone to the *black hole attack* [37]. In AODV and N-AODV a black hole node produces fakes RREPs in which the sequence number is as great as possible; the initiator establishes the communication with the malicious node and the latter can misuse or discard the received information. The black hole can be supported by one or more *colluders* that confirm the trustworthiness of the fake RREP. The Black hole-free N-AODV protocol allows the honest nodes to intercept the black holes and the colluders, thanks to two control packets: each intermediate node n receiving an RREP must verify the trustworthiness of the nodes in the path followed by the RREP; to do this, n produces a *challenge packet* (CHL) for the destination node, and only the latter can produce the correct *response packet* (RES). If n receives RES, it sends the RREP, otherwise the next node towards the destination is a possible black hole.

We are currently working on the final definition of the ASM for the N-AODV and the BN-AODV protocols, together with the extension of MOTION to those protocols. Moreover, a complexity analysis of the network's protocols and the related algorithms could be performed; a change of the structure that represents the connectivity among the nodes (from adjacency matrix to adjacency list, for instance), could lead to a dramatic improvement of the resource-consumption during the simulation of the behaviour of the network.

REFERENCES

[1] D. P. Agrawal and Q.-A. Zeng, Introduction to wireless and mobile systems. Cengage learning - Fourth Edition, Boston, 2016.

[2] P. Arcaini, A. Gargantini, E. Riccobene, and P. Scandurra, ”A model-driven process for engineering a toolset for a formal method,” Software: Practice and Experience, vol. 41(2), pp. 155–166, 2011.

[3] S. Basagni, M. Mastrogiovanni, A. Panconesi, and C. Petrioli, ”Localized protocols for ad hoc clustering and backbone formation: A performance comparison,” IEEE Trans. Parallel Distrib. Syst., vol. 17(4), pp. 292–306, 2006, doi:10.1109/TPDS.2006.52, URL https://doi.org/10.1109/TPDS.2006.52

[4] E. Börger and A. Raschke, Modeling Companion for Software Practitioners. Springer, 2018. doi:10.1007/978-3-662-56641-1. URL https://doi.org/10.1007/978-3-662-56641-1

[5] E. Börger and R. Stärk, Abstract State Machines: A Method for High-Level System Design and Analysis. Springer Verlag, Berlin, 2003.

[6] L. Bononi, G. D’Angelo, and L. Donatiello, ”Hla-based adaptive distributed simulation of wireless mobile systems,” Proceedings of the seventeenth workshop on Parallel and distributed simulation, p. 40, IEEE Computer Society, 2003.

[7] J. Brennecke and O. Rank, ”The firm’s knowledge network and the transfer of advice among corporate inventors — A multilevel network study,” Research Policy, 46(4), pp. 768-783, 2017.

[8] R. Calinescu, C. Ghezzi, M. Kwiatkowska, and R. Mirandola, ”Self-adaptive software needs quantitative verification at runtime,” Communications of the ACM, vol. 55(9), pp. 69–77, 2012.

[9] E. Covino and G. Pani, ”Analysis of Mobile Networks’ Protocols Based on Abstract State Machines,” in A. Raschke et al. (Eds.): Logic, computation and rigorous methods, LNCS 12750, pp. 187-198, 2021.

[10] E. Covino and G. Pani, ”Analysis of a Formal Model for Social Groups,” ICOMP’22 - The 23rd Int’l Conf on Internet Computing and Internet of Things, July 25th-28th, 2022, Las Vegas, USA.

[11] A. D’Andrea, F. Ferri, and P. Grifoni, ”An Overview of Methods for Virtual Social Network Analysis,” in Abraham, Ajith (Ed.), Computational Social Network Analysis: Trends, Tools and Research Advances, Springer, pp. 3-25, 2009.

[12] N. Das, S. K. Bisoy, and S. Tanty, ”Performance analysis of TCP variants using routing protocols of manet in grid topology,” Cognitive Informatics and Soft Computing, pp. 239–245, Springer, 2019.

[13] F. Erbas, K. Kyamakya, and K. Jobmann, ”Modelling and performance analysis of a novel position-based reliable unicast and multicast routing method using coloured Petri nets,” 2003 IEEE 58th Vehicular Technology Conference, VTC 2003-Fall, Vol. 5, pp. 3099–3104, IEEE, 2003.

[14] A. Fehnker et al., ”A process algebra for wireless mesh networks,” European Symposium on Programming, pp. 295–315, Springer, 2012.

[15] A. Garcia-Santiago, J. Castaneda-Camacho, J. F. Guerrero-Castellanos, G. Mino-Aguilar, and V. Y. Ponce-Hinestroza, ”Simulation platform for a VANET using the truetime toolbox: Further result toward cyber-physical vehicle systems,” IEEE 88th Vehicular Technology Conference (VTC-Fall), IEEE, pp. 1–5, 2018.

[16] A. Gargantini, E. Riccobene, and P. Scandurra, ”A metamodel-based language and a simulation engine for abstract state machines,” J. UCS, vol. 14(12), pp. 1949–1983, 2008. doi:10.3217/jucs-014-12-1949. URL https://doi.org/10.3217/jucs-014-12-1949

[17] U. Glässer, Y. Gurevich, and M. Veanes, ”Abstract communication model for distributed systems,” IEEE Trans. Software Eng., 30(7), pp. 458–472, 2004. doi:10.1109/TSE.2004.25. URL https://doi.org/10.1109/TSE.2004.25

[18] M. Grandjean, ”A social network analysis of Twitter: Mapping the digital humanities community,” Cogent Arts and Humanities, 3(1), 2016.

[19] M. Grandjean, ”Analisi e visualizzazioni delle reti in storia. L’esempio della cooperazione intellettuale della Società delle Nazioni,” Memoria e Ricerca, 2, pp. 371–393, 2017.

[20] L. Hagen, T. Keller, S. Neely, N. DePaula, and C. Robert-Cooperman, ”Crisis Communications in the Age of Social Media: A Network Analysis of Zika-Related Tweets,” Social Science Computer Review, 36(5), pp. 523-541, 2018.

[21] URL https://github.com/Angelo997/VisualMotion.git.

[22] M. Ilyas (Ed.), The Handbook of Ad Hoc Wireless Networks, (1st ed.). CRC Press, 2002.

[23] K. Jensen, L. M. Kristensen, and L. Wells, ”Coloured Petri nets and CPN tools for modelling and validation of concurrent systems,” International Journal on Software Tools for Technology Transfer, vol. 9(3-4), pp. 213–254, 2007.

[24] N. Kaur and R. Singhai, ”Analysis of traffic impact on proposed congestion control scheme in AODV,” Wireless Personal Communications, pp. 1–24, 2019.

[25] C. Kim, E. Talipov, and B. Ahn, ”A reverse AODV routing protocol in ad hoc mobile networks,” International Conference on Embedded and Ubiquitous Computing, pp. 522–531, Springer, 2006.

[26] X. Li, M. R. Lyu, and J. Liu, ”A trust model based routing protocol for secure ad hoc networks,” in 2004 IEEE Aerospace Conference Proceedings, Vol. 2, pp. 1286–1295, IEEE, 2004.

[27] R. Mantegna and H. Stanley, Introduction to Econophysics: Correlations and Complexity in Finance. Cambridge University Press, 1999.

[28] M. Merro, ”An observational theory for mobile ad hoc networks,” Information and Computation, vol. 207(2), pp. 194–208, 2009.

[29] H. R. Nasrinpour, M. R. Friesen, and R. D. McLeod, ”An Agent-Based Model of Message Propagation in the Facebook Electronic Social Network,” arXiv:1611.07454, 2016.

[30] E. Otte and R. Rousseau, ”Social network analysis: a powerful strategy, also for the information sciences,” Journal of Information Science, 28(6), pp. 441–453, 2002.

[31] A. P. Pandian, J. I.-Z. Chen, and Z. A. Baig, ”Sustainable mobile networks and its applications,” Mobile networks and application, vol. 24(2), pp. 295–297, 2019.

[32] C. E. Perkins, E. M. Belding-Royer, and S. R. Das, ”Ad hoc on-demand distance vector (AODV) routing,” RFC 3561 (2003), pp. 1–37, doi:10.17487/RFC3561. URL https://doi.org/10.17487/RFC3561.

[33] R. R. Roy, Handbook of Mobile Ad Hoc Networks for Mobility Models. Springer, 2011.

[34] F. Schweitzer, ”Sociophysics,” Physics Today, 71(2), p. 40, 2018.

[35] A. Singh, C. Ramakrishnan, and S. A. Smolka, ”A process calculus for mobile ad-hoc networks,” Science of Computer Programming, vol.75(6), pp. 440–469, 2010.

[36] D. A. Tran and H. Raghavendra, ”Congestion adaptive routing in mobile ad-hoc networks,” IEEE Trans. Parallel Distrib. Syst., vol. 17(11), pp. 1294–1305, 2006. doi:10.1109/TPDS.2006.15. URL https://doi.org/10.1109/TPDS.2006.151.

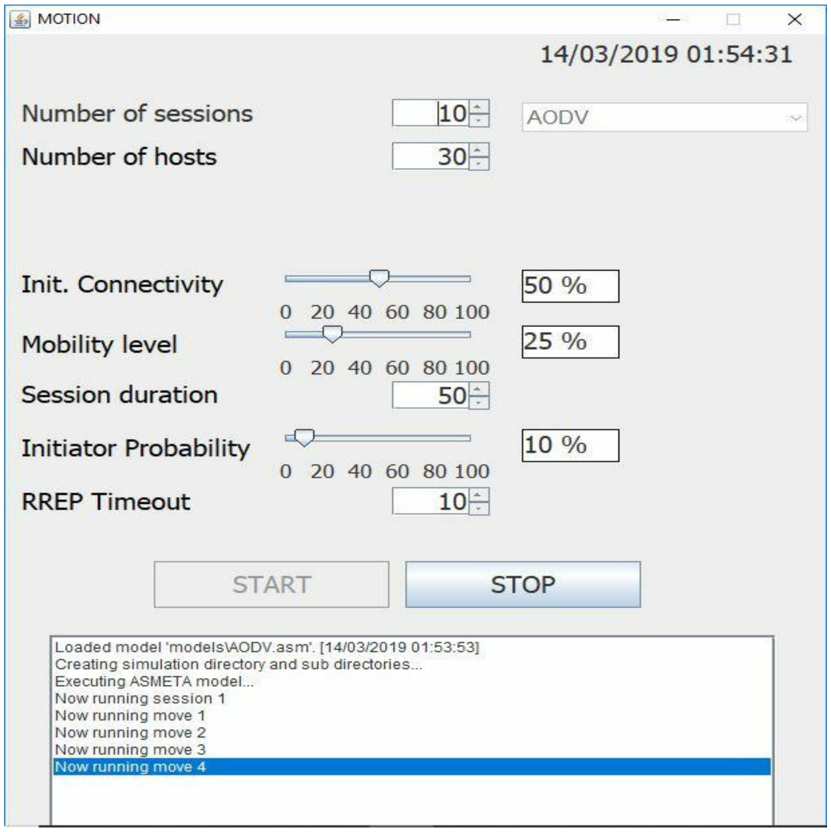
[37] F.-H. Tseng, L.-D. Chou, and H.-C. Chao, ”A survey of black hole attacks in wireless mobile ad hoc networks,” Human-centric computing and information sciences, 1,4, 2011.

[38] S. Wasserman and K. Faust, Social Networks Analysis: Methods and Applications. Cambridge University Press, 1994.

[39] J. Wu and F. Dai, ”Mobility-sensitive topology control in mobile ad hoc networks,” IEEE Trans. Parallel Distrib. Syst., vol. 17(6), pp. 522–535, doi:10.1109/TPDS.2006.73, URL https://doi.org/10.1109/TPDS.2006.73.

[40] C. Xiong, T. Murata, and J. Leigh, ”An approach for verifying routing protocols in mobile ad hoc networks using Petri nets,” in Proceedings of the IEEE 6th Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication, Vol. 2, pp. 537–540, IEEE, 2004.

[41] M. G. Zapata, ”Secure ad hoc on-demand distance vector routing,” ACM SIGMOBILE Mobile Computing and Communications Review, 6(3), pp. 106–107, 2002.

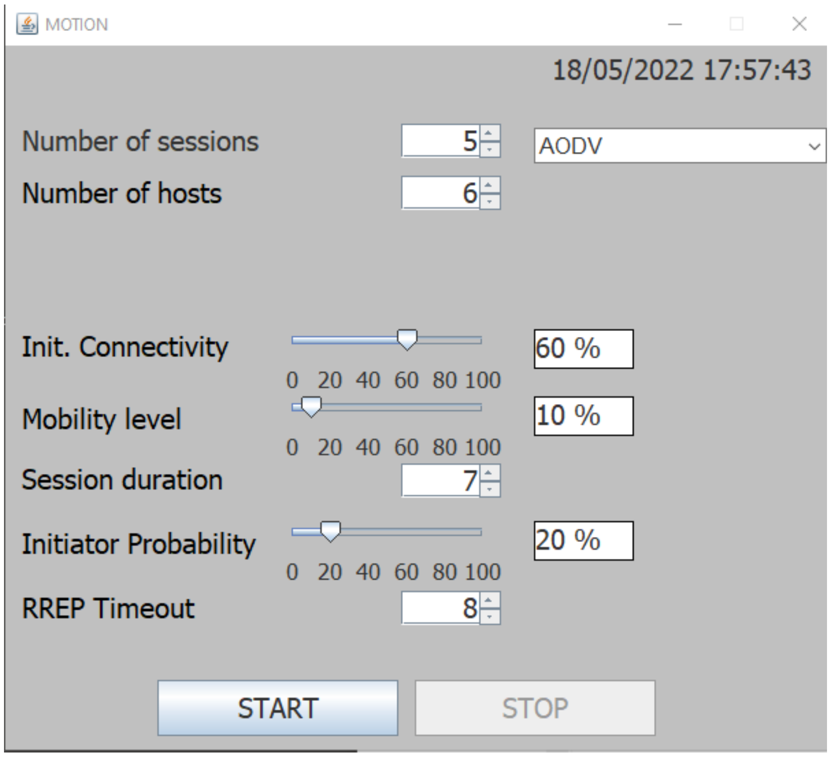


14/03/2019 01:54:31

Number of sessions 10 AODV
Number of hosts 30
Init. Connectivity 0 20 40 60 80 100 50 %
Mobility level 0 20 40 60 80 100 25 %
Session duration 50
Initiator Probability 0 20 40 60 80 100 10 %
RREP Timeout 10

```
Loaded model 'models\AODV.asm'. [14/03/2019 01:53:53]
Creating simulation directory and sub directories...
Executing ASMETA model...
Now running session 1
Now running move 1
Now running move 2
Now running move 3
Now running move 4
```

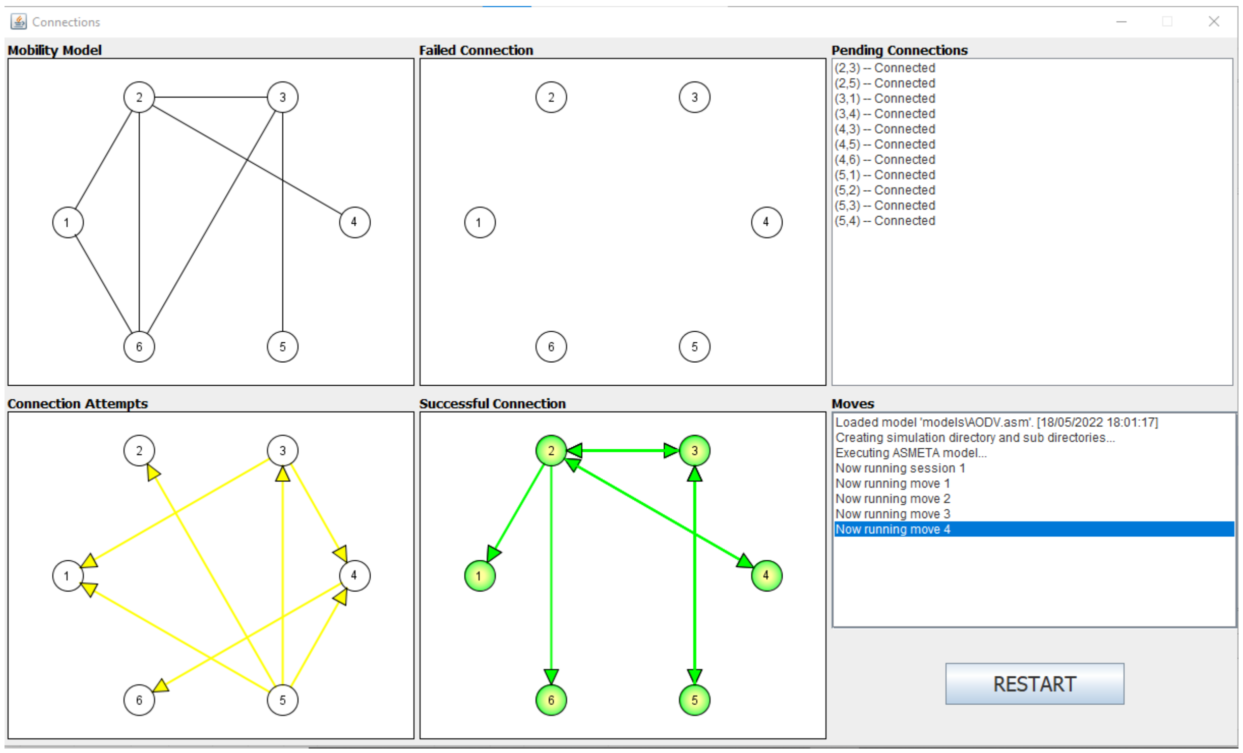
Fig. 1. MOTION's user interface for AODV protocol



18/05/2022 17:57:43

Number of sessions 5 AODV
Number of hosts 6
Init. Connectivity 0 20 40 60 80 100 60 %
Mobility level 0 20 40 60 80 100 10 %
Session duration 7
Initiator Probability 0 20 40 60 80 100 20 %
RREP Timeout 8

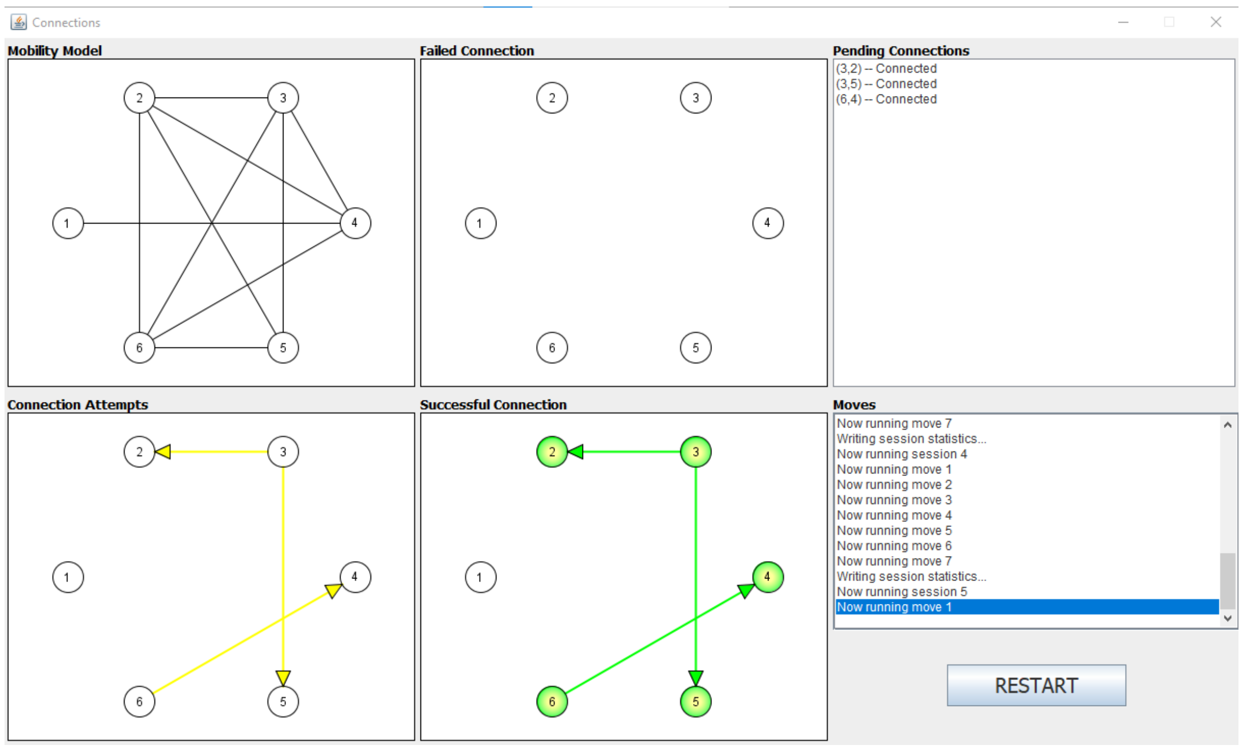
Fig. 2. MOTION's new user interface



```
(2,3) -- Connected
(2,5) -- Connected
(3,1) -- Connected
(3,4) -- Connected
(4,3) -- Connected
(4,5) -- Connected
(4,6) -- Connected
(5,1) -- Connected
(5,2) -- Connected
(5,3) -- Connected
(5,4) -- Connected
```

```
Loaded model 'models\AODV.asm'. [18/05/2022 18:01:17]
Creating simulation directory and sub directories...
Executing ASMETA model...
Now running session 1
Now running move 1
Now running move 2
Now running move 3
Now running move 4
```

Fig. 3. Evolution of the network



```
(3,2) -- Connected
(3,5) -- Connected
(6,4) -- Connected
```

```
Now running move 7
Writing session statistics...
Now running session 4
Now running move 1
Now running move 2
Now running move 3
Now running move 4
Now running move 5
Now running move 6
Now running move 7
Writing session statistics...
Now running session 5
Now running move 1
```

Fig. 4. Evolution of the network, after several steps