Efficient Task Allocation Algorithm for Parallel Real-Time EMT Simulation

B. Bruned, I. M. Martins, P. Rault, S. Dennetiere

***Abstract*--Real-time EMT simulation relies on multi-cores computers to accelerate the simulation through parallelization. It also increases simulation accuracy allowing the use of a lower time step. First, the network has to be split into several tasks using a separation technique. Then, each task has to be allocated/mapped to a processor. This paper focuses on this problem which can be formulated as a TAP (Task Allocation Problem). To find an optimal task allocation operational research techniques can be used. Heuristics such as graph partitioning allow to get fast solutions. Their performances are asserted with very large networks and real-time simulator architectures, both from TSO grids. Exact resolution methods are used to verify solution quality. The validation of each task mapping strategy is done through a real EMT case study which involves real-time Hardware-in-the-Loop simulation.**

***Keywords*: Real-time simulation, Parallel simulation Optimization, Task Allocation Problem, graph partitioning, Hardware-in-the-Loop.**

I. Introduction

The need of real-time EMT simulation has increased with the development of electronics devices in the transmission network related to the high penetration of wind power farms and HVDC links. Since 2011, the French TSO, RTE, has created his own real-time laboratory SMARTE to study interaction between this new power equipment. Hard-in-the-Loop simulation, which connects a real-time simulator to a replica of the on-site control system, allows to perform accurate EMT studies close to on-field phenomena. Otherwise, the utility of replica is various from maintenance activities to real-time event studies which have occurred on the network [1]. In order to improve accuracy, detailed network are used for EMT simulation [2] although interesting network reduction methods based on frequency equivalent [3] [4] help in certain cases to accelerate the simulation.

To cope with large networks, real-time EMT tools take advantage of the parallelization offered by the multi-cores supercomputers used as real-time simulators [2]. Indeed, it accelerates the simulation and respects the time constraint to be able to interact with hardware device. The parallelization is automatically performed in two steps. First, the network is separated into several tasks. Then, each network task is mapped to the simulator's processors before starting the parallel simulation. The stability of a real-time simulation depends strongly on the result of this task mapping.

Previous works [2] [5] have demonstrated the efficiency of graph partitioning algorithms [6] on some Software-in-the-Loop (SIL) examples. However, no full study has been done to assert the performance on industrial cases and the optimality of found solutions. This paper proposes to fill this gap. After reminding the problem formulation as a Task Allocation Problem (TAP) [7] and presenting heuristic techniques, very large realistic network instances are tested with real architectures to verify algorithms performance. Then, a deep analysis of the whole graph partitioning algorithm allows to understand its advantages and limits. Additionally, exact solutions from a linear programming formulation are first used in this paper to assert the quality of solutions of the graph partitioning algorithms. Lastly, in complement to previous SIL examples, a Hardware-in-the-Loop (HIL) set-up of three-terminal HVDC grid with DC Circuit Breakers validates the efficiency of the task allocation algorithm and discusses the mapping strategy.

All algorithm implementations and testing have been done on the real-time EMT tools HYPERSIM [8] which proposes a fully –automatic network parallelization.

II. Task Allocation Problem

A. Task Separation

The first step of parallelization is to split the network into several tasks which will be run in parallel on several cores. Two main separation techniques are used for the split.

The first one relies on decoupling element as power lines. If the propagation delay is greater than the simulation time step, tasks can be separated through the lines as the delay allows to transmit computed value for the next time step. Based on this principle, for real-time, a topology analysis is automatically performed to split the network into sub-networks according to power lines. Otherwise, off line tools perform parallelization directly on the nodal resolution [9] [10].

When the decoupling is not possible through power lines, others techniques have to be used. Hybrid method resolutions based on Nodal formulation and State-Space allow to separate the network into State-space equivalents that can be solved in parallel [11]. Also the Multi-Area Thévenin Equivalents (MATE) method [12] or Compensation Method (nonlinear networks) [13] can split the network without using the natural delay of power lines and can be fully automated [14].

B. Bruned, P. Rault and S. Dennetiere are with RTE, Paris la Défense France (emails: boris.bruned@rte-france.com, pierre.rault@rte-france.com, sebastien.dennetiere@rte-france.com). I. M. Martins is with ENSTA (email: ian.martins@ensta-paristech.fr).

Paper submitted to the International Conference on Power Systems Transients (IPST2019) in Perpignan, France June 17-20, 2019.

The second step of parallelization consists in assigning tasks to processors (Task Mapping Problem).

B. Task Mapping Problem formulation

The Task Allocation Problem, TAP, is a well-known problem in the literature of combinatorial optimization [7] [15]. It consists of mapping a network of elements called "tasks" to a set of connected containers called here of "processors". A task is said to be allocated to a processor when it is mapped uniquely to it.

Each task has a cost of allocation/an estimated execution time and each processor has a budget to it, the time-step constraint for real-time simulation. A TAP solution is said to be valid if every task is allocated and there is no budget overflow, i.e., the time-step constraint is respected. An optimal solution is a valid one that minimizes the communications cost – the sum of the weighted connections between tasks allocated on different processors.

Given the NP-complete complexity [16] of the problem, the use of heuristic-base approaches becomes necessary to find good solutions to the problem using a small amount of time. Two heuristics [5] have been implemented.

C. Heuristic algorithms

1) A*

Based on the A* algorithm, the first method [5] follows a tree-shaped scheme. Each separation is composed by ordered pairs (T_i, P_j) indicating that the task T_i is allocated to the processor P_j. This tree has as many levels as tasks and as many leaves as combinations of tasks and processors. At each level a new task is allocated to a processor and it halts when all the tasks are allocated. By choosing the next pair, the algorithm tries to minimize the communication cost and to balance processor loads.

2) Graph Partitioning

The second method is based on graph partitioning techniques [6]. These heuristics are commonly used to automatically parallelize EMT simulations [5] [14] [16]. The goal of the algorithm is to map a "Source Graph", SG, to a "Target Graph", TG. It consists of partitioning the former and then mapping the resulting subsets of source vertices to a target vertex. Source edges, in the other hand, are mapped to a subset of edges in the TG. This subset is the smaller path between two target vertices previously adjacent in the SG. Minimizing the communication cost, the algorithm agglomerates adjacent source vertices with heavy connecting edges in target vertices close to each other, if not in the same. Furthermore, it keeps the balance of weights of the target vertices according to a constraint δ [17], the Load Imbalance Ratio, LIR, that will be discussed later.

One can see the network of tasks as the SG and the "architecture", the ensemble of processors and its connections, as the TG. The TAP is naturally formulated to this case.

For the rest of this paper, the second method will be further developed. The SCOTCH library [6] [17] is used to process the graph partitioning. Its results will be compared with those of the first method.

D. Performance Tests on large instances

To figure out the performance of both algorithms, setting an upper bound, they have to be tested on very large and realistic networks. Large instances come mainly from national transmission networks. For instance in Figure 1, the whole French transmission network (400kV + 225kV) reach thousands of tasks (3486 buses, 1056 lines and 274 transformers).

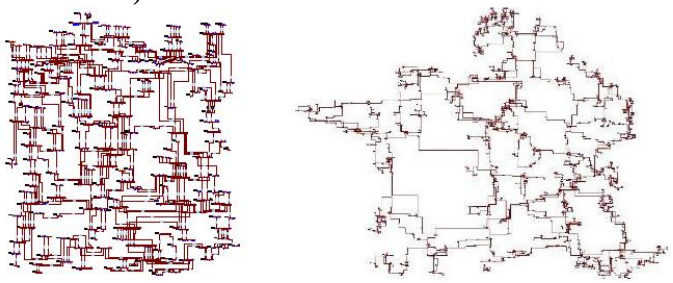

Figure 1 French 400 kV (on the left) and 225 kV grid

For architecture instances, as the real-time simulation runs on a single supercomputer (no cluster of pcs), the TG is often complete (all inter-processor communication links exist). However, the TG is not necessary homogenous (same communication cost between each processor). The SGI UV100 architecture [18] (96 processors in total) used as a real-time simulator on RTE real-time laboratory contains 4 chassis alternatively with 1 or 2 blades each. A blade contains 2 sockets/CPUs with 8 cores/processors each. The inter-chassis communication cost is different from the inter-blade one which is different from the inter-socket communication and from the inter-core one as well. This architecture is clearly heterogeneous.

Table I below presents the performance results for two very large networks for both heuristics (A* and Scotch with balance strategy and δ=0.01) on the UV100 simulator with a 40µs time step. To set up the performance the following criteria are used: ***NbTask***, The number of tasks; ***NbProc***, The number of processors used; ***Comm***, The total number of communication, i.e., the number of signal multiplying by the communication link cost (respectively 10, 16, 43, and 65 respectively for inter-cores, sockets, blades, and chassis communication links); ***Var***, The processor load variance to measure the load balancing; ***Time***, The execution time of the task mapping in seconds (run on host with Intel i7-4910MQ CPU @ 2.90GHz).

TABLE I
PERFORMANCE RESULTS FOR LARGE NETWORK INSTANCES

Instance	NbTask	Mapping Strategy	NbProc	Comm	Var	Time
French 400kV grid	460	A*	16	15096	4.93	0.98
		Scotch	16	6960	0.14	0.06
French 400kV + 225 kV grid	1510	A*	79	148974	14.35	81.1
		Scotch	79	123216	0.34	0.17

Graph partitioning technique is faster than A* algorithm and gives better solutions (lower communication cost and better balancing). In absolute, the execution time is quite fast (no more than few seconds). Only graph partitioning techniques and homogeneous architecture are considered later.

III. GRAPH PARTITIONING FOR AN EFFICIENT TASK MAPPING

A. A fast graph partitioning algorithm

1) Overview of the algorithm

The graph partitioning algorithm is composed of several routines [6] [17], being the Recursive Bipartitioning, RB, the main one. This algorithm recursively bipartitions subsets of both SG and TG. The bipartitioning is executed by a Greedy Graph Partitioning algorithm, GPA. At each recursive step, a subset of the SG will be partially mapped to a subset of the TG. In the next recursive step, the resulting sub-subsets in both graphs will be mapped accordingly to theirs parents' mapping.

Others than the RB, more algorithms and post-processing methods are available, notably the Multi-level method, ML, which has three distinct phases. It is important for the performance. It consists of a coarsening phase, in which the graph is reduced to a smaller equivalent one, a partition phase, where algorithms like RB and GPA are used, and an uncoarsening phase, in which the graph grows back to its original size. Another important method is the Exactifier, EX. It is a post-processing method that balances the partition trying to increase the least the communication cost. A combination of these methods is called a "strategy".

Two strategies are proposed, "Quality" and "Balance". The former prioritizes the minimization of the cost function described above, resulting most of times in a slower and more unbalanced partition. The latter prioritizes a balanced partition at the expense of the quality criteria. Both strategies have the same core structure: an external ML to reduce huge graphs to big graphs with 5000 nodes during the first coarsening phase (step 1) followed by a RB (step 2) that, at each sub domain, uses an internal ML coarsening phase to reduce the partitions to tiny graphs with 120 nodes (step 3) to finally do the bipartition using the GPA (step 4). During the uncoarsening phases, internal and external (steps 5 and 6, respectively), some post-processing methods are used. At the end of this process (step 7), in the case of "Balance", the EX method is used to control the LIR.

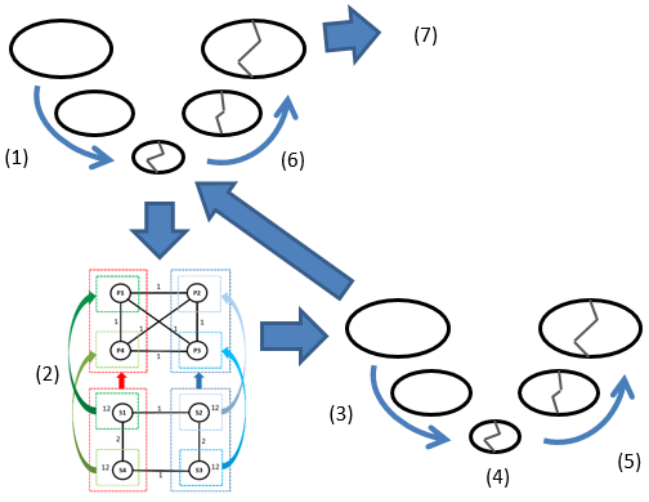

Figure 2 Illustrative scheme of the graph partitioning algorithm and its steps.

2) Algorithm limits

As it was explained, the mapping is executed with no restrictions on the total allocation cost in each processor. A verification is made after: if there is at least one processor in which the time-step is exceeded, an available processor will be added to the architecture and the partitioning algorithm will be run again. This process will repeat until either a mapping is valid or there is no more available processors to be added. An observable consequence of the validation method is a possible excessive use of processors. Since a strategy is a particular heuristic combination, it is possible that it will miss a valid solution and have to increase the number of processors.

B. Hyper parameter tuning

To overcome previous issues, some modifications were implemented. A benchmarked network with 802 vertices and 2052 edges (332 buses, 513 lines and 42 transformers) has been chosen for testing the tuning of hyper parameters. The time step is set to 40μs.

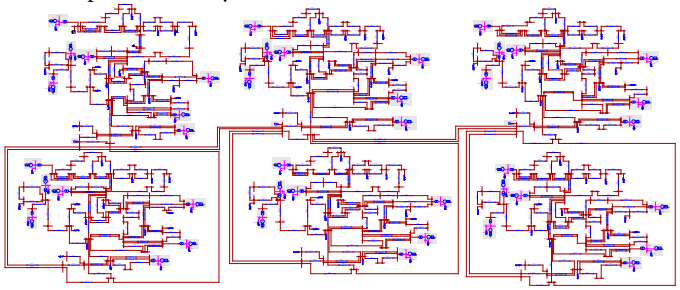

Figure 3 Benchmarked example of 802 tasks

1) Load Imbalance Ratio (LIR)

A relevant metric to characterize a partition is its LIR, $0 < \delta \leq 1$, which measures the total imbalance of charges between processors after the partition. It is a constraint in the communication minimization problem. Table II presents different results obtained when δ varies for both quality and balance strategies ("-" means no solution has been found).

TABLE II
VALUES FOR δ COMPARISON

δ	Strategy	NbProc	Comm	Var	Time
0.25	Quality	19	390	8.874	0.036
	Balance	-	-	-	-
0.10	Quality	18	390	4.644	0.036
	Balance	18	402	4.004	0.033
0.01	Quality	17	450	0.463	0.023
	Balance	17	480	0.032	0.018

As δ decreases, the number of processors used in both strategies also decreases, particularly, using smaller δ than 0.25 allowed "Balance" to find a solution with 19 processors or less. The communication, however, increased. This happens because more communicating tasks are forcedly separated to different processors. Since there were fewer attempts, the execution times were reduced and so was the variance since there were fewer processors and so the tasks were better concentrated.

2) Specific Strategy

As it was highlighted previously, the creation of a specific strategy for transmission networks could offer better results than generic ones. Table III shows the results of this new strategy:

TABLE III
RESULTS FOR THE SPECIFIC STRATEGY

δ	NbProc	Comm	Var	Time
0.25	18	396	4.979	0.029
0.10	18	384	3.647	0.036
0.01	17	444	0.040	0.026

Similarly to "Quality", the new strategy made a partition in all three cases, but it managed to find solutions with fewer or the same number of processors and with the communication cost in the same magnitude. Its variance is comparable with the one obtained in the previous section.

3) Random Seed

Another possible source of optimization is the random seed used in the algorithm. Randomness is used mainly during the GPA, when the first node is chosen to start the bipartition. A lucky choice may result in a better bipartition. Currently, there are iterations over the GPA keeping the best partition among them. Iterating over the seed, the algorithm was able to improve some of the partitions made in the standard case. Table IV shows the results for δ = 0.25:

TABLE IV
RANDOM SEED RESULTS

Strategy	NbProc	NbIter	Comm	Var	Time
Quality	18	26	372	6.435	0.257
Balance	19	32	396	12.13	0.202
Specific	18	21	396	5.417	0.167

The column *NbIter* indicates the number of iterations before finding the first valid partition. In this example, the random seed was iterated 10 times before restart all over adding a new processor. Since the starting point is 16 processors, *NbIter* = 26 as in the first line means that using the "Quality" strategy, the solution was found after the sixth random seed iteration with 18 processors. It is better than the previous except time for the additional work done. Similarly for "Balance", a solution was found with 19 processors, which is a huge improvement from the previous case where no valid partition was found. Finally for the new strategy, the partition found is exactly the same than before.

Vary the random seed showed to be a relevant optimization, but its natural random character prevents this method to guarantee a valid solution.

C. Validation toward exact solutions

1) Modeling

To model the problem, two new Boolean variables must be introduced. The first, the allocation variable x_{ij} is 1 if the task i is allocated to processor j and 0 otherwise. The second is the dilatation variable ρ_{ij}^{kl} which values 1 if, for a pair of communicating tasks, task i is allocated to processor j and task k is allocated to processor l, and 0 otherwise. The linear formulation of the problem is:

$$\min \frac{1}{T}\left(\sum_j \left|\bar{T} - \frac{1}{\mu}\sum_i t_i\, x_{ij}\right|\right) + \left(\sum_{\{i,k\}}\left(\sum_j\left(\sum_{l\neq j} w_{ik}\,\rho_{ij}^{kl}\right)\right)\right) \quad (1)$$

Subject to:

$$\forall i, \sum_j x_{ij} = 1 \;\; ; \;\; \forall j, \sum_i t_i\, x_{ij} \leq \mu \quad (2)$$

$$\forall \{i,k\}, \forall j,l \mid j \neq l, \qquad \rho_{ij}^{kl} \geq x_{ij} + x_{kl} - 1 \quad (3)$$

Where T is the total tasks evaluation time and $\bar{T}$ is the average time per processor. The constant μ is the time constraint, t_i is the evaluation time of task i and w_{ik} is the weight of the communication between tasks i and k.The main expression has two distinct parts. The first, the sum over the processors, is the LIR. The second one, the triple sum, is the Communication Cost, CC. The subsequent equations are, respectively, the allocation's uniqueness, the time-step constraint and the relation of both Boolean variables.

A benchmarked model of 35 buses is used to compare the obtained solutions with those of LP. It has 103 tasks to be partitioned among 3 processors, resulting in about 10^{49} possible combinations.

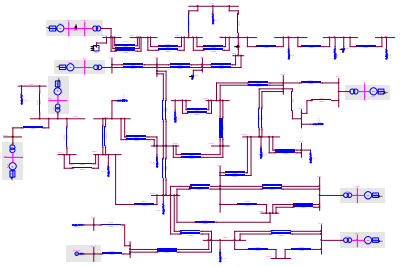

Figure 4 35 buses model

2) Pareto Front

For the problem studied, the two criteria analyzed are the LIR and the CC. The Pareto front allows a comparison of both criteria magnitudes and serves as a lower bound for the solutions of the problem, allowing to formalize what a "good-enough" solution is. To find it, the method used is to vary the proportional weight either magnitude has.

To resolve the linear problem, the chosen parallel solver was FICO™ Xpress solver [19] and it has been run on a 190 cores cluster. Figure 5 below shows the comparison between the Pareto front and heuristic results. The blue dots forming the line are the exact solutions of the LP. The values in parenthesis are the respective weights of each side of the expression. The marker's sizes represent the chosen δ values (0.01, 0.05, 0.1 and 0.25).

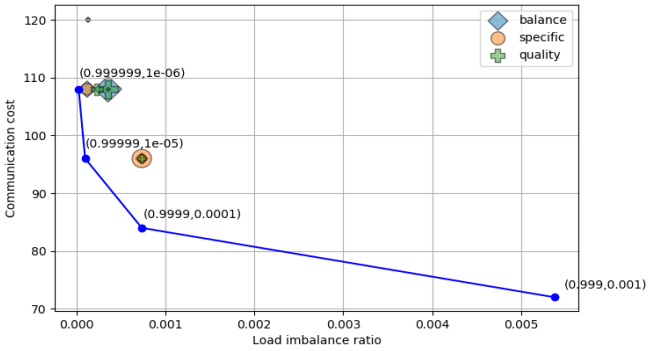

Figure 5 Pareto front for the 35 buses model

When compared with the Pareto optimal solutions, one can notice that heuristic solutions tend to prioritize the CC, that is, it deteriorates, even if not much, the LIR to be able have the same CC of an exact solution. Besides, the weights in parenthesis mean a strong preference for the LIR in the exact solutions (more reasonable proportions than those at the bottom-right blue dots result in that same solution). Solutions

are scattered through the left side figure.

Finally, to measure the distance between the Pareto front and the obtained solutions, the metric used was the smallest Euclidean distance between a heuristic solution v and an exact solution $p \in PF$, the Pareto front, that is:

$$\min_{p \in PF}(\|p - v\|_2) \quad (5)$$

Table V shows the measured distance. All distances are in order of 10^{-4}, which one can assume to be close enough.

TABLE V
DISTANCE TO THE PARETO FRONT FOR THE 35 BUSES MODEL

δ	Balance	Specific	Quality
0.25	3.30	6.37	3.30
0.10	0.95	0.95	2.01
0.05	6.37	6.37	6.37
0.01	120000	120000	3.30

IV. HARDWARE-IN-THE-LOOP TEST CASE

A. Test case overview

To illustrate the impact of different mapping strategies, an HIL simulation test case is presented in this paper. This system was developed for the Best Path DEMO#2 [20] and it is detailed in [21]. It consists in a three-terminal HVDC grid including DC circuit breakers (DCCBs), as shown in Figure 6. Two MMCs (Station 2 and 3) are controlled by a generic controller in HYPERSIM. The last converter (Station 1) is controlled through simulator IOs, by industrial controllers provided by ABB. Similarly, the DCCB models are also controlled by ABB control hardware.

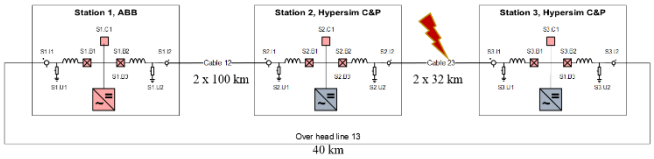

Figure 6 Overview of the three terminals DC grid

The objective of this HIL set-up (Figure 7) is to assess the efficiency of DC grid protection algorithm as well as the action DCCB control into a DC grid, for different DC faults. Detailed results of the DCCB control can also be found in [21].

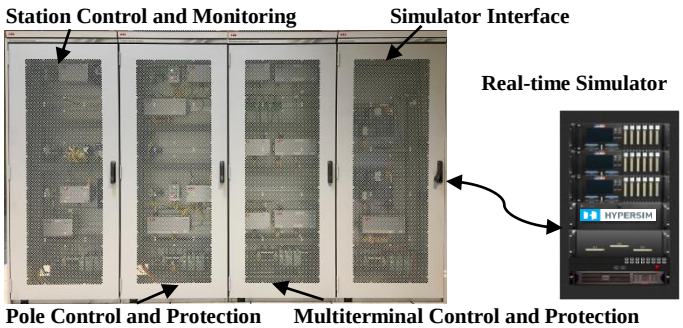

Figure 7 Overview of the HIL set-up, replica provided by ABB

B. Task Mapping results

The DC grid can be divided into 85 tasks. The main tasks can be listed by load importance as follow:

- 3 converters stations with DC breaker,
- Control system of 2 converters,
- 6 DC lines (2 sections for each DC line),
- The rest is dedicated to the IOs for control replicas (DC breaker and VSC control) and FPGA MMC valve models.

Three scotch strategies, among them the specific one from III. B. 2) , have been tested with an imbalance ratio δ=0.01. The time step has been set to 30µs. The last column indicates the steady state execution time of the most loaded processor. The simulations were performed on an OP5031 target with 32 cores (2 CPU Intel Xeon E5-2697A v4 @ 2.60GHz - 16 cores). Only specific and balance strategies succeed to respect the real-time constraint. It has been observed that favoring CPU load balancing instead of task communication deals better with erroneous task time estimates. For the simulation, the task mapping from the balance strategy is kept for the DCCB validation.

TABLE VI
TASK MAPPING RESULT FOR EACH SCOTCH STRATEGY

Strategy	NbProc	Comm	Var	Time	Max RT ExecTime
Quality	6	375	18.47	0.0028	31.5
Balance	6	524	0.09	0.0029	24.0
Specific	6	522	0.27	0.0113	24.0

C. Simulation Results

The scenario consists of a permanent negative pole-to-ground fault on the shorter DC cable. The fault event occurs at t=200ms in the following figures. Figure 8 shows that, during the transients, the 30µs time step constraint is respected in the three most loaded processors (each task corresponds one converter station, identified by one color). As results, the balance strategy has better performances than quality as observed in steady state.

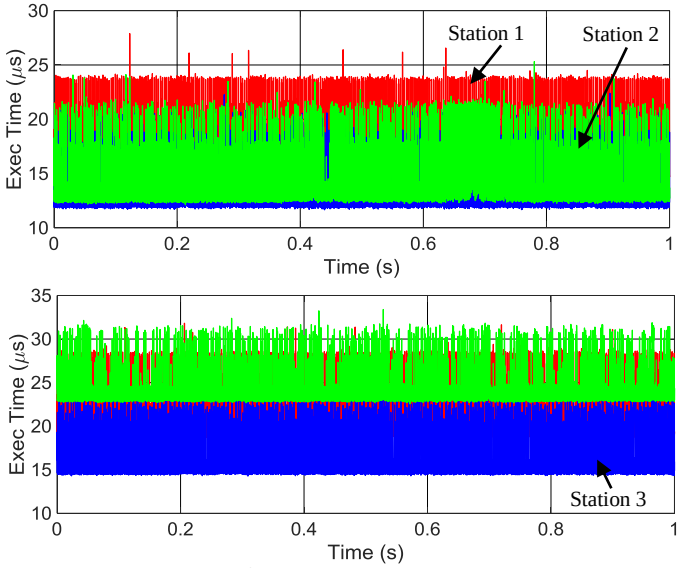

Figure 8 Execution time for converters stations during transients, (top: balance strategy – bottom: quality strategy)

The last graph shows that the inter-task communications is negligible. This confirms that, for this case, balancing the processor loads is more important than minimizing the inter-processor communication, which justifies the choice of balanced strategy. This conclusion may not be applicable to heterogeneous architectures.

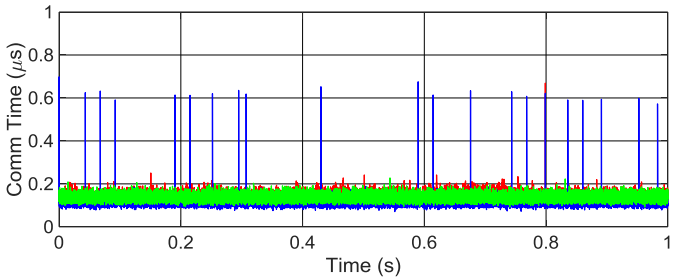

Figure 9 Inter-processor communication time for each converter station

Thanks to the DC grid protection implemented in the control cubicles, the faulty cable is isolated from the rest of the DC grid with DCCBs. After some transients due to the fault and DCCB operation, the DC voltage of each station returns to its operational point +/- 320kV as shown in Figure 10.

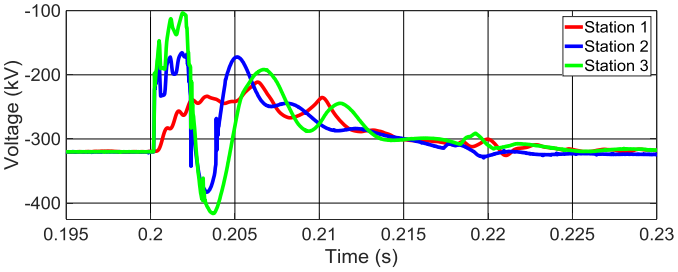

Figure 10 Negative pole voltage at each converter station terminal subjected to clearance of permanent DC cable fault

V. Conclusions

This paper has highlighted deeply that graph partitioning algorithm is one of the most efficient heuristic to proceed an optimal solution of the task mapping problem for real-time EMT simulations. First, tests within an industrial tool over realistic networks have shown a good tradeoff in terms of execution time and quality of the solution. Then, the tuning of hyper parameters allows the engineer to increase the quality of solutions while respecting the time step constraint. Additionally, comparisons with exact methods have strengthened the confidence of finding almost-optimal solutions. Lastly, an industrial real-time Hardware-in-the-Loop simulation has validated the use of this technique where balance strategy should be preferred over the quality one to best deal with erroneous task time estimates.

VI. Acknowledgment

First, the authors would like to acknowledge Eric Lemieux and Philippe Le Huy from Hydro-Québec. Discussions on SCOTCH and architecture have fed this paper. Also many thanks for the help provided by optimization experts Dr. Jean Maeght and Dr. Manuel Ruiz from RTE R&D to handle linear programming formulations and solvers. Last acknowledge to ABB who allows us to publish on the example of HVDC grid set-up which was developed for the Best Paths project (co-funded by the European Union's Seventh Framework Programme for Research, Technological Development and Demonstration under the grant agreement no. 612748).

VII. References

[1] H.Saad, Y. Vernay, S. Dennetiere, P. Rault, B.Clerc, "System Dynamic Studies of Power Electronics Devices with Real-Time Simulation - A TSO operational experience," Cigre Paris Session, 2018.

[2] P. Le-Huy, M. Woodacre, S. Guérette, É. Lemieux, "Massively Parallel Real-Time Simulation of Very-Large-Scale Power Systems," Proceedings of the IPST conference IPST2017, Seoul, Republic of Korea, June 2017.

[3] Y. Vernay, B. Gustavsen, "Application of Frequency-Dependent Network Equivalents for EMTP Simulation of Transformer Inrush Current in Large Networks," International Conference on Power System Transients (IPST) conference, Vancouver, Canada, 2013.

[4] B. Bruned, C. Martin, S. Dennetière, Y. Vernay, "Implementation of a unified modelling between EMT tools for Network Studies," Proceedings of the IPST conference, Seoul, Republic of Korea, June 2017.

[5] S. Dennetiere, B. Bruned, H. Saad, E. Lémieux, "Task separation for real-time simulation of the CIGRE DC grid benchmark," Proceedings of the PSCC conference PSCC2018, Dublin, Ireland, 2018.

[6] F. Pellegrini, J. Roman, "SCOTCH: A Software Package for Static Mapping by Dual Recursive Bipartitioning of Process and Architecture Graphs," HPCH'96, Brussels, Belgium, April 15-19, 1996.

[7] A. Ernst, H. Jiang, M. Krishnamoorthy," A new Lagrangian Heuristic for the Task Allocation Problem," Industrial Mathematics, 137-158, 2006.

[8] V. Q. Do, J.-C. Soumagne, G. Sybille, G. Turmel, P. Giroux, G.Cloutier, S. Poulin. "Hypersim, an Integrated Real-Time Simulator for Power Networks and Control Systems," ICDS'99, Vasteras, Sweden, May 25-28, 1999.

[9] A. Abusalah, O. Saad, J. Mahseredjian, U. Karaagac, L. Gerin-Lajoie, I. Kocar, "CPU Based Parallel Computation of Electromagnetic Transients For Large Scale Power Systems," Proceedings of the IPST conference IPST2017, Seoul, Republic of Korea, 2017.

[10] Rikido Yonezawa and Taku Noda, "A Study of Solution Process Parallelization for an EMT Analysis Program Using OpenMP," Proceedings of the IPST conference IPST2017, Seoul, Republic of Korea, 2017.

[11] C. Dufour, J. Mahseredjian, J. Belanger, "A Combined State-Space Nodal Method for the Simulation of Power System Transients," IEEE Transactions on Power Delivery, vol. 26, no, 2, pp. 928-935, 2011.

[12] M. A. Tomim, J. R. Marti, L. Wang, "Parallel Computation of Large Power System Network Solutions using the Multi-Area Thévenin Equivalents (MATE) Algorithm," Proceedings of the 16th PSCC conference, Glasgow, Scotland, 2008.

[13] W. F. Tinney, "Compensation Methods for Network Solutions by Optimally Ordered Triangular Factorization," IEEE Trans. on Power Apparatus and Systems, vol. PAS-91, no. 1, pp. 123-127, Jan. 1972.

[14] S. Fan, H. Ding, A. Kariyawasam, "Parallel Electromagnetic Transients Simulation with Shared Memory Architecture Computers," IEEE Trans. on Power Delivery, Vol. 33, Issue 1, Feb. 2018.

[15] J. Aguilar and E. Gelenbe, "Task Assignment and Transaction Clustering Heuristics for Distributed Systems," Information Sciences 199-219, March 1997.

[16] P. Zhang, J. R. Marti, H. W. Dommel, "Network Partitioning for Real-Time Power System Simulation," Proceedings of the IPST conference IPST'05, Montréal, Canada, June 2005.

[17] F. Pellegrini, "SCOTCH and LIBSCOTCH 6.0 User's Guide," LaBRL, University of Bordeaux, Septembre 2014.

[18] SGI® Altix® UV 100 System User's Guide.

[19] FICO™ Xpress Optimization Suite (2017) Xpress-Optimizer Reference manual Release 31.10.

[20] Best Paths project online. http://www.bestpaths-project.eu/

[21] P. Rault, M. Yazdani, S. Dennetière, C. Wikstrom, H. Saad, "Industrial Application of DCCB in HVDC grids – Modelings and Testing with Physical Controls," IPST conference IPST2019, Perpignan, France, 2019.