Ph.D. thesis abstract

Ph.D. est omnis divisum in partes tres: first of all, the testing branch. Then, the algorithm optimization and parallelization branch. Lastly, the Artificial Intelligence (AI) branch.

Testing. Chips are becoming larger and larger as new modules are added to improve performance and safety measures.

In the testing domain, together with my colleagues at CAD group, we began developing tools for the evaluation of test programs. Those tools include an EVCD file analyzer, a toolchain built around it and a new metric for improving the speed of test routines development. The toolchain is able to speed up the ability to evaluate large test programs on an automotive System-on-Chip (SoC) from the SP58 family including more than 20 million gates, reducing the wall-clock time of execution from days to hours for larger files.

On the other hand, the new metric called connectivity is able to speed-up automatic test generation using a well-known evolutionary tool called µGP. Moreover, it can also be used by programmers, providing a fast feedback with comparison to the standard fault coverage. However, we do not aim to replace the fault coverage metric, as it is the standard that provides complete information; thus, the connectivity, by computing partial information, is able to detect quick-to-fix errors in the execution of the program, providing instruction-level feedback to the test engineer.

Algorithms and Parallelization. Algorithms from NP class solve relevant problems today, and effort is put into improving their current performance. In particular, together with my colleagues and thesis students we focused on the Maximum Common Subgraph (MCS) problem, a well-known NP-hard problem that is used in molecule mining and even security. We improved the state of the art algorithm McSplit and its variants, among which the latest McSplitDAL, by applying the PageRank algorithm to classify vertices of the graphs. We were able to improve the original result up to 1.07 times on graphs of size up to 100 vertices. We also tested different vertex classification algorithms on larger graphs of up to 7000 vertices, finding that, while PageRank is not always the winning metric, it provides the most stable improvements.

As Central Processing Units (CPUs) get more and more capable, and the multi-core paradigm became standard for large computations, also the many-core approach is becoming more important over the years: from Artificial Intelligence to computer graphics, Graphics Processing Units (GPUs) are ubuquitous in today's world, and research is being put into adapting or improving currently known algorithms into the many-core approach. In particular, together with my colleagues and thesis students we focused on the Graph Coloring (GC) problem, a well-known NP-complete mathematical

problem. We were able to improve one of the state of the art algorithms on GPU, called Jones-Plassman-Luby, with a custom CUDA implementation, improving over the most famous implementation, Gunrock, up to 62 times, with a geometric mean of 3.16 times.

Artificial Intelligence. Together with Université Paris Saclay, we developed a system that is able to abstract rules using given knowledge of basic concepts. In particular, the objective of the system is building an inductive explanation of a game, detecting objects, categories and rules that the objects follow. Using a standard image detection library, OpenCV, we were able to feed videos involving 2 games, Arkanoid and Pong. By using a basic image recognition we were able to detect objects and their movements, detecting their velocities and positions. After this step, we were able to detect interactions between objects, creating a cause-effect relationship between interactions and change of status. In the end, we use a genetic algorithm provided by the Inspyred library to induce a set of categories and rules that apply to each category, involving interactions between them.