Synchronous Cellular Automata Scheduler with Construction Heuristic to Static Task Scheduling in Multiprocessors

Murillo G. Carneiro
Universidade Federal de Uberlândia
Avenida João Naves de Ávila, 2121
Uberlândia - MG, Brazil 38400-902
carneiro.murillo@gmail.com

Gina M. B. Oliveira
Universidade Federal de Uberlândia
Avenida João Naves de Ávila, 2121
Uberlândia - MG, Brazil 38400-902
gina@facom.ufu.br

ABSTRACT

Static Task Scheduling Problem (STSP) in multiprocessors is a NP-Complete problem. HLFET is one of the simplest well-known heuristics designed to deal with it. Meta-heuristics like genetic algorithms and simulated annealing had also been applied to this problem. Cellular Automata (CA) have been recently used to solve STSP. The main feature of CA-based scheduling is the extraction of knowledge while scheduling an application and its subsequent reuse in other instances. An evolutionary algorithm is applied to search for efficient CA rules in learning phase. Previous works showed this approach is promising. However some desirable features have not been successfully exploited yet, such as: (i) the massive parallelism inherent to CA, (ii) the usage of an arbitrary number of processors and (iii) the reuse of evolved rules with competitive results. This paper presents a new model called SCAS-H (Synchronous Cellular Automata Scheduler initialized by Heuristics). Its major innovation is the usage of a heuristic based on HLFET to start up the CA rule evolution. Program graphs found in literature and others randomly generated were used in the experiments. Results show that SCAS-H overcame related models both in scheduling results as computational performance and it presented competitive results with meta-heuristics.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods, Scheduling*; I.2.6 [**Artificial Intelligence**]: Learning—*knowledge acquisition*

General Terms

Algorithms, Experimentation

Keywords

SCAS-H, cellular automata, task scheduling, synchronous updating, construction heuristics

1. INTRODUCTION

In the context of multiprocessors, scheduling aims to allocate a set of computational tasks that compose a parallel application into architecture nodes. In case of the Static Task Scheduling (STSP) investigated here, all information about the tasks is known a priori and an optimal solution for an instance of STSP is such that the precedence constraints among tasks are satisfied and the makespan is minimized.

Cellular Automata (CA) are discrete dynamical systems composed by lattice and transition rule. They have the potential to emerge a complex global behavior from simple interactions between local units. Previous works [1, 2, 4, 5] pointed to the promising use of Cellular Automata (CA) rules to solve STSP. In such works, a genetic algorithm (GA) is employed in a learning phase to found adequate rules to schedule a parallel program. A promising skill of such previous approaches is their ability to extract knowledge of the scheduling process of a parallel application and reuse it in others instances. In fact, in traditional heuristics and meta-heuristics approaches to STSP, a computational effort is used to solve an instance of the problem and when a new instance is presented to the algorithm, the process need to start again from scratch. The major motivation to study cellular automata scheduling approaches is the possibility to discover transition rules starting from given parallel applications which are able to scheduling them and besides they have a generalization ability to be used to schedule new instances of STSP.

In addition, the possibility of implementation on parallel hardware together with the simplicity of its basic components are among the most notable features of CA [3]. These characteristics turn them adequate to be implemented in massive parallel architectures like FPGA speeding up the throughput. However, almost all previous models presented in literature are not able to explore the inherent parallelism into CA because they use an asynchronous sequential updating of cell states (only one cell can update its state at a time) [2]. Recently, a scheduler model called *Synchronous Cellular Automata Scheduler* (SCAS) [1] has presented good scheduling results while employing a synchronous updating of cells. However it was identified limitations in such model and previous studies specially when a number of processors above two is used in the system architecture. In our investigations we discover that one reason of such weakness is due to the guideline that CA transition rules evolved in the learning phase must be able to perform the schedule starting from any initial configuration (IC) of the lattice. This lack of sensibility required in the previous models turns the evolutionary search complex and time-consuming.

The main objective of this work is to introduce and eval-

Copyright is held by the author/owner(s).
GECCO'12 Companion, July 7–11, 2012, Philadelphia, PA, USA.
ACM 978-1-4503-1178-6/12/07.

uate a new model called *Synchronous Cellular Automata-Based initialized by Heuristic* (SCAS-H). The major innovation of the new model is the usage of a construction heuristics to start up the evolution of lattice cells. As a consequence the evolved rules need to be able to perform the schedule starting only from an specific initial configuration of the lattice, given by a deterministic construction heuristics. Here, a modified version of HLFET (a well-know heuristic for STSP) named DHLFET (that is HLFET without its random choices) was used as the construction heuristics to establish the initial allocation of tasks. As its predecessor SCAS [1], the resultant method uses a linear and simple neighborhood and it is suitable to be implemented in parallel hardware since it employs a synchronous cells updating.

2. METHODOLOGY

The major modification in SCAS-H refers to the way the CA lattice is initialized to perform the schedule, which reflects in a change in what kind of transition rule ability GA is searching for. In previous works [1, 2, 4, 5], the scheduler model focuses on the capacity of a transition rule to evolve any random initial lattice to a configuration that represents the optimal allocation allocation of tasks. Therefore, during the learning phase, each rule is evaluated according to its performance in scheduling a set of initial lattices (S_{Latt}). Besides, in the normal phase, the quality of any evolved rule is measured using it to schedule a new set of random lattices. However, the search for this independence to initial lattice makes the rules search complex and computationally intensive, embarrassing GA convergence. Furthermore, we believe that the more important generalization is not related to the initial lattice used to start the scheduling process, instead it is related to the capacity of a rule to schedule others instances in reuse mode.

The main steps in evaluation used in SCAS-H are: (i) applying a deterministic construction heuristic chosen a priori to obtain an initial allocation which defines the IC of the lattice; (ii) temporal evolution of the lattice using each rule transition r in P for T time steps; (iii) the final lattices in time T define the final allocations of tasks which are ordered in each processor using a scheduling policy obtaining makespan associated to each rule r; (iv) rule fitness is equal to makespan obtained using it. The best rule in P presents the smallest makespan. Algorithm 1 defines the major steps of SCAS-H in learning mode being that: P is the size of population, P_{cross} is the number of generated individuals in crossover, P_{mut} is mutation rate, selection is given by simple tournament $Tour$ and IC_H is the IC of lattice obtained by DHLFET. In reuse mode, the CA is equipped with a set of rules in repository and SCAS-H receives a new program graph to schedule. DHLFET is used in reuse mode in step (i) and steps (ii) to (iv) are executed for each rule in repository, returning the scheduling with the smallest makespan.

Algorithm 1 SCAS-H Learning Mode

1: Randomly sorting a population of P rules
2: Generate initial configuration of lattice with DHLFET (IC_H) //step(i)
3: Compute the fitness of the P rules //step(ii) to (iv)
4: **while** (!$condition\_finish$) **do**
5: Selecting pairs of rules in P using simple tournament ($Tour = 2$) to generate P_{cross} rules
6: Applying the single-point crossover in pairs selected
7: Submit P_{cross} rules to mutation P_{mut}
8: Compute the fitness of the P_{cross} rules //step(ii) to (iv)
9: Sorting $P + P_{cross}$ based on fitness and choose the P best rules for next generation
10: **end while**
11: P is stored in rules database

3. EXPERIMENTS AND DISCUSSION

Previous works have pointed to the difficult to employ more than two processors in CA-based scheduling. Here the scalability of the new model in respect to the number of processors was investigated. Program graphs found in literature and others randomly generated were used to test the performance of SCAS-H in architecture systems with 2, 3 and 4 processors. Experiments were performed for two modes of the new model. Results showed that SCAS-H overcame related models both in learning and reuse phases and specially when the number of processor is increasing. The computation cost was also reduced when compared to these models because SCAS-H not use a set of IC but only one initial configuration given by DHLFET. Besides, SCAS-H results were compared with those obtained using DHLFET (which can be thought as the kick-off for SCAS-H) and some meta-heuristics: two different genetic algorithms approaches and a simulated annealing-based algorithm. Considering learning phase, SCAS-H returned results as good as genetic approaches which performed better than DHLFET and simulated annealing. Considering reuse of rules, SCAS-H was competitive with the best results found by meta-heuristics, while returning a low computational cost (SCAS-H is about 150 times faster than genetic algorithms runs). Future works include the investigation of new structures for a more effective reuse in CA-based scheduling.

4. ACKNOWLEDGMENTS

M.G.C. thanks to CNPq for financial support and G.M.B.O. is grateful to CNPq and FAPEMIG.

5. REFERENCES

[1] M. G. Carneiro and G. M. B. Oliveira. Cellular automata-based model with synchronous updating for task static scheduling. In *Proceedings of 17th International Workshop on Cellular Automata and Discrete Complex System*, pages 263–272, 2011.

[2] F. Seredynski and A. Y. Zomaya. Sequential and parallel cellular automata-based scheduling algorithms. *IEEE Transactions on Parallel and Distributed Systems*, 13(10):1009–1022, 2002.

[3] M. Sipper. *Evolution of Parallel Cellular Machines, The Cellular Programming Approach*. Springer, 1997.

[4] A. Swiecicka, F. Seredynski, and A. Y. Zomaya. Multiprocessor scheduling and rescheduling with use of cellular automata and artificial immune system support. *IEEE Transactions on Parallel and Distributed Systems*, 17(3):253–262, 2006.

[5] P. M. Vidica and G. M. B. Oliveira. Cellular automata-based scheduling: A new approach to improve generalization ability of evolved rules. *Brazilian Symposium on Artificial Neural Networks (SBRN'06)*, pages 18–23, 2006.