JESSE BATT (RESEARCH PAPER BY PIERRE DELISLE)

PARALLEL ANT COLONY OPTIMISATION: ALGORITHMIC MODELS AND HARDWARE IMPLEMENTATIONS

INTRODUCTION

- Ant Colony Optimisation is a constructive population based approach based on the social behaviour of ants.
- It is well known to be a powerful method to solve academic and industrial combinational problems
- There are two main approaches to parallel ACO CPU implementations - one colony of parallel ants (on processor cores etc) and multiple parallel ant colonies, each assigned to processors.
- The paper in question here aims to complement existing parallel ACO models but apply them to working with more high performance computing architectures

TWO PARALLELISATION STRATEGIES FOR THE ANT SYSTEM

- 1. Low Level Synchronous - distribute ants to processors in a master-slave fashion. The master broadcasts the pheromone structure to slaves at each iteration. The slaves then complete their tours in parallel and send them back to the master.
- 2. Partially Asynchronous - aiming to reduce the aforementioned communication/synchronisation overhead by allowing the algorithm to perform a given number of iterations without actually exchanging information

- With this amount of communication happening and how frequently it occurs, the LLS method described here will incur a large amount of communication overhead.
- In reducing the communication overhead, the second method described here would be the favourable of the two.

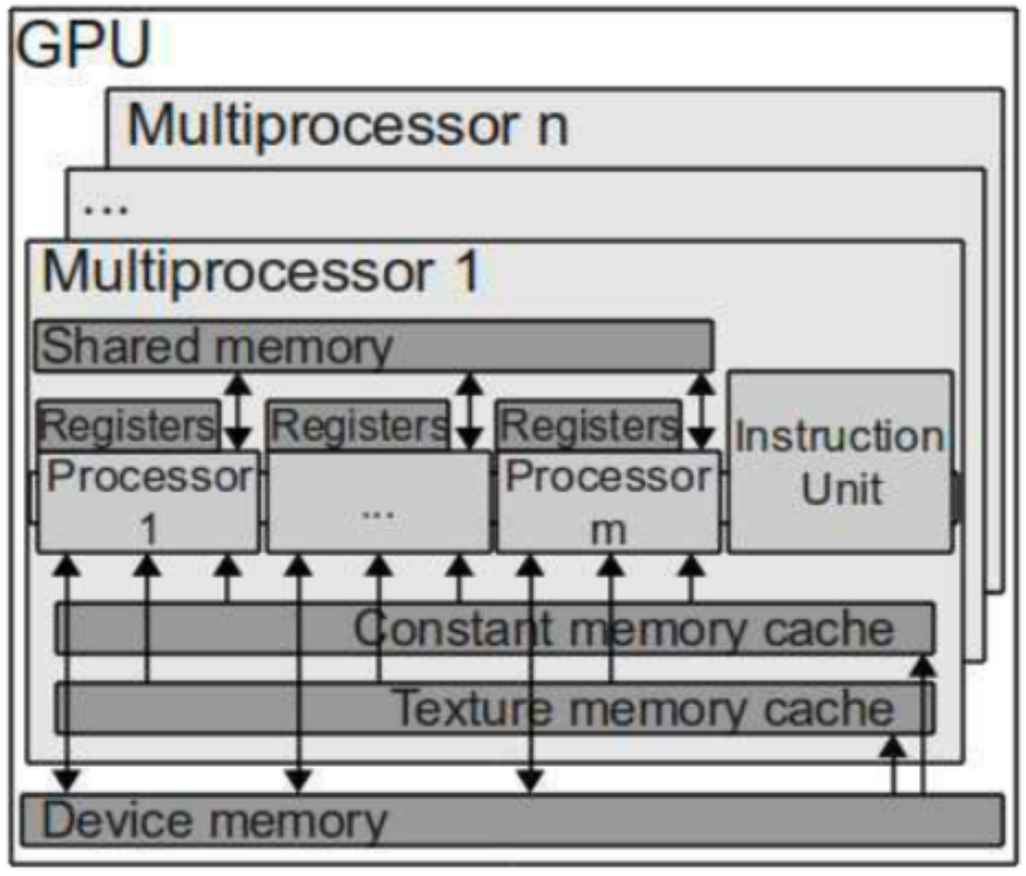
A NEW ARCHITECTURE ORIENTED TAXONOMY FOR PARALLEL ACO

- In order to efficiently implement parallelisation it is important to consider the architecture
- Clusters/Networks of Workstations (COWs/NOWs) - distributed memory architectures where each processor has its own memory
- Symmetric Multi Processor (SMP) / Multicore Systems - shared-memory architectures where the processors are connected to a common memory
- Grids - pools of heterogeneous and dynamic computing resources that are geographically widespread.
- GPUs - devices that are used in computers to manipulate computer graphics.

GPU EXAMPLE – NVIDIA GPU ARCHITECTURE

- The conventional **NVIDIA GPU** includes many streaming multiprocessors and streaming processors which execute multiple coordinated threads.

- Several memories are distinguished on this special hardware, differing in size, latency and access type.



ACO GRANULARITY LEVELS FOR PARALLELISATION

- **Colony** level - the execution of a whole ACO algorithm as a task and assigning it to a processor
- **Iteration** level - a hybrid of colony and ant level. To share the iterations of the algorithm between available processors
- **Ant** level - the distribution of the tasks included in an iteration to available processors. One or many ants are assigned to each processing element.
- **Solution** level - the main operations that are considered for parallelisation are the state transition rule and solution evaluation

COMPUTATIONAL ENTITIES

- **System -** a unified computational resource which may be a standard workstation or a cluster.
- **Node -** part of a system to which tasks can be assigned. A system may then be composed of a single node which is the case of the standard workstation, or of multiple nodes which is the case of clusters.
- **Process -** manages and executes sequential and parallel programs.
- **Block -** intermediate entity between process and thread. Blocks are comprised of many threads
- **Thread -** sequential flow of instructions that is part of a block. Threads are always sequential and executes instructions on a processor at a given time

MEMORY

- In the context of ACO algorithms, Memory serves as a container for pheromone information, problem data and other parameters
- **Local Memory** - directly accessible by the entities of a given level with a fast access time
- **Global Memory -** can also be accessed directly by entities at a given level but has a slow access time
- **Remote Memory** - can NOT be directly accessed by the entities, but the information can be made available by an explicit operation between entities

A NEW ARCHITECTURE ORIENTED TAXONOMY FOR PARALLEL ACO

ACO granularity	Computational entity	Memory
Colony	System	Local
Iteration	Node	Global
Ant	Process	Remote
Solution element	Block	
	Thread	

- The above is a proposed taxonomy for an Architecture based taxonomy which seeks to link each computational entity and memory structure to each level of ACO granularity.

CASE STUDY – MULTI COLONY PACO ON AN SMP AND MULTICORE ARCHITECTURE

- Multiple colonies - global shared memory
- Single system, single node
- Colonies are executed in parallel and spawn multiple parallel ants
- Colonies assigned to processes
- Ants assigned to threads

$$COLONY(global\ process)\text{-}ITERATION(global\ process)\text{-}ANT(global\ thread)$$

CASE STUDY – MULTI COLONY PACO ON AN SMP AND MULTICORE ARCHITECTURE (CONT.)

- In the proposed implementation, search processes are independent - as many copies of data structures as there are colonies.
- Pheromone structures and ACO parameters are private and exclusive to each thread
- This implies minimal communication and synchronisation overhead but, in a practical context, colonies still need to access this data simultaneously via shared system resources

CASE STUDY – MULTI COLONY PACO ON AN SMP AND MULTICORE ARCHITECTURE (CONT.)

- Use Parallel Region at the beginning of the sequential algorithm to create as many threads as colonies and dedicate a memory location for storing the Best Global Solution which is accessible by all threads

- At the end of the Parallel Region, a Critical Section ensures that each thread verifies their best solution against the BGS

CASE STUDY – TRAVELLING SALESMAN (INDEPENDENT COLONIES)

Problem	Nb. of cores	Speedup	Avg. tour length	Best tour length	Closeness
rat783	1	-	8,824	8,810	99.80
	2	1.98	8,823	8,806	99.81
	4	3.69	8,820	8,815	99.84
	8	5.93	8,829	8,822	99.74
d2103	1	-	80,511	80,466	99.92
	2	1.97	80,573	80,466	99.85
	4	4.00	80,508	80,477	99.93
	8	6.92	80,501	80,463	99.94
pla7397	1	-	23,365,444	23,353,738	99.55
	2	1.99	23,352,192	23,332,663	99.61
	4	3.80	23,380,613	23,350,736	99.48
	8	7.80	23,425,288	23,396,612	99.29
usa13509	1	-	20,465,969	20,414,755	97.58
	2	1.89	20,376,567	20,250,719	98.03
	4	3.65	20,443,190	20,423,250	97.70
	8	7.30	20,441,068	20,410,519	97.71

Table 2. Multiple independent colonies: number of cores, speedup, average tour length, best tour length and relative closeness of the average tour length to the optimal solution.

CASE STUDY – TRAVELLING SALESMAN (COOPERATING COLONIES)

Problem	Nb. of cores	Speedup	Avg. tour length	Best tour length	Closeness
rat783	1	-	8,824	8,810	99.80
	2	1.95	8,822	8,810	99.82
	4	3.69	8,819	8,815	99.86
	8	5.72	8,816	8,812	99.89
d2103	1	-	80,511	80,466	99.92
	2	1.95	80,475	80,450	99.97
	4	3.81	80,489	80,450	99.95
	8	6.85	80,484	80,454	99.96
pla7397	1	-	23,365,444	23,353,738	99.55
	2	2.00	23,348,946	23,322,729	99.62
	4	3.89	23,358,733	23,334,364	99.58
	8	7.75	23,356,251	23,350,596	99.59
usa13509	1	-	20,465,969	20,414,755	97.58
	2	2.02	20,456,702	20,392,284	97.63
	4	3.20	20,450,581	20,414,972	97.66
	8	5.55	20,434,287	20,375,145	97.74

Table 3. Multiple cooperating colonies - Global best exchange each 10 cycles: number of cores, speedup, average tour length, best tour length and relative closeness of the average tour length to the optimal solution.

CASE STUDY – TRAVELLING SALESMAN (CONT.)

- Average tour length obtained with multiple cooperating colonies is closer to the optimal solution than with independent colonies or sequential execution.
- From this you can conclude that this implementation can be effectively implemented on SMP and multi-core nodes with up to 8 processors

CASE STUDY – PARALLEL ANTS ON GPUS

- Single Ant Colony on a GPU
- Ants assigned to blocks, solution elements assigned to threads
- Ants communicate with slow GPU device memory
- Solution Elements communicate with the faster shared memory of a multiprocessor
- In this case we will refer to the CUDA API for GPUs
- NOTE - GPUs use Streaming Multiprocessors, made up of Streaming Processors

$$COLONY(process) - ITERATION - (\text{process}) - ANT(^{global\ block}) - SOLUTION\_ELEMENT(local\ thread)$$

.

CASE STUDY – PARALLEL ANTS ON GPU

- Ants assigned to a CUDA Block, runs tour in parallel on a specific SM of the GPU
- One thread dedicated to tour construction
- Other threads used for Solution Element level parallelism
- 1 Ant per Block, per SM, so it is possible to make use of Shared Memory
- Some difficulties arise from memory management - CPU<–>GPU data transfers and Global Memory access times incur considerable overhead
- Shared Memory can be used to combat this
- In context of ACO, the 3 main data structures needed by all ants are the Pheromone Matrix, the Penalty Matrix and Transition Candidate Lists but they are too large for Shared Memory, so have to reside in Global Memory instead.

CASE STUDY – MMAS ALGORITHM ON GPU

- Number of blocks = number of ants
- Number of threads = size of candidate list
- Number is tour constructions is global

- Speedup increases with problem size, so shows scalability.
- An anomaly with the d2103 cities problem - large workload and data structures cause potential memory access latency which the parallelisation doesn't keep up with

Problem		Speedup	Stützle and Hoos	Avg. tour length	Best tour length	Closeness
eil51	Sequential	-	427.80	427.32	426	99.69
	Parallel	6.84	-	427.20	426	99.72
kroA100	Sequential	-	21,336.90	21,314.36	21,282	99.85
	Parallel	8.12	-	21,317.32	21,282	99.83
d198	Sequential	-	15,952.30	15,973.84	15,913	98.77
	Parallel	11.13	-	15,961.64	15,851	98.85
lin318	Sequential	-	42,346.60	42,341.72	42,107	99.26
	Parallel	11.03	-	42,325.32	42,147	99.29
rat783	Sequential	-	-	9,042.44	8,923	97.32
	Parallel	15.58	-	9,002.32	8,899	97.77
fl1577	Sequential	-	-	24,490.30	24,201	89.83
	Parallel	19.47	-	24,287.80	23,938	90.84
d2103	Sequential	-	-	82,754.30	82,378	97.14
	Parallel	17.64	-	82,756.00	82,547	97.13

Table 4. GPU implementation: speedup, average tour length from Stützle and Hoos original MMAS implementation [35], average tour length, best tour length and relative closeness of the average tour length to the optimal solution.

REFERENCES

- Delisle, P. (2013). Parallel Ant Colony Optimization: Algorithmic Models and Hardware Implementations. In *Ant Colony Optimization-Techniques and Applications*. InTech.

- Stützle, T., & Hoos, H. (1998). Improvements on the ant-system: Introducing the MAX-MIN ant system. In *Artificial Neural Nets and Genetic Algorithms* (pp. 245-249). Springer, Vienna.

- Stützle, T. (1998, September). Parallelization strategies for ant colony optimization. In *International Conference on Parallel Problem Solving from Nature* (pp. 722-731). Springer, Berlin, Heidelberg.