



Parallel Ant Colony Optimization on Graphics Processing Units

Audrey Delévacq^a, Pierre Delisle^{a,*}, Marc Gravel^b, Michaël Krajecki^a

^a CRESTIC, Université de Reims Champagne-Ardenne, Reims, 51687, France

^b Département d'Informatique et de Mathématique, Université du Québec à Chicoutimi, Saguenay, Canada

ARTICLE INFO

Article history:

Received 9 June 2011

Received in revised form

15 December 2011

Accepted 10 January 2012

Available online 25 January 2012

Keywords:

Ant colony optimization

Parallel metaheuristics

GPU

CUDA

MMAS

Parallel ants

Multiple colonies

ABSTRACT

The purpose of this paper is to propose effective parallelization strategies for the Ant Colony Optimization (ACO) metaheuristic on Graphics Processing Units (GPUs). The Max–Min Ant System (MMAS) algorithm augmented with 3-opt local search is used as a framework for the implementation of the *parallel ants* and *multiple ant colonies* general parallelization approaches. The four resulting GPU algorithms are extensively evaluated and compared on both speedup and solution quality on a state-of-the-art Fermi GPU architecture. A rigorous effort is made to keep parallel algorithms true to the original MMAS applied to the Traveling Salesman Problem. We report speedups of up to 23.60 with solution quality similar to the original sequential implementation. With the intent of providing a parallelization framework for ACO on GPUs, a comparative experimental study highlights the performance impact of ACO parameters, GPU technical configuration, memory structures and parallelization granularity.

© 2012 Elsevier Inc. All rights reserved.

1. Introduction

The Ant Colony Optimization (ACO) metaheuristic [17] is a constructive, population-based approach based on the social behavior of ants. As it is acknowledged as a powerful method to solve combinatorial optimization problems, a considerable amount of work is dedicated to improving its performance. Among the proposed solutions, we find the use of parallel computing to reduce computation time, improve solution quality or both.

Most parallel ACO implementations can be classified into two general approaches. The first one is the parallel execution of the ants construction phase in a single colony. Initiated by Bullnheimer et al. [5], it aims to accelerate computations by distributing ants to computing elements. The second one, introduced by Stützle [27], is the execution of multiple ant colonies. In this case, entire ant colonies are attributed to processors in order to speedup computations as well as to potentially improve solution quality by introducing cooperation schemes between colonies. These implementations usually follow the *message-passing* and *shared-memory* computing paradigms. The relatively high-level abstraction model they provide facilitates the development of effective and portable

optimization software on conventional CPU-based parallel architectures.

However, as research on parallel architectures is rapidly evolving, new types of hardware have recently become available for high performance computing. Among them, we find Graphics Processing Units (GPUs) which provide great computing power at an affordable cost but are difficult to program. In fact, it is not clear that conventional paradigms are suitable for expressing parallelism in a way that is efficiently implementable on GPU architectures. As academic and industrial combinatorial optimization problems always increase in size and complexity, the field of parallel metaheuristics has to follow this evolution of high performance computing.

The purpose of this paper is to propose parallel implementations of ACO that are suitable for GPU computing environments. For both *parallel ants* and *multiple colonies* general approaches, two parallelization strategies are designed and experimentally compared on speedup and solution quality. Important algorithmic, technical and programming issues are also addressed in this context.

This paper is organized as follows. First, we present the ACO metaheuristic, the Max–Min Ant System (MMAS) algorithm and its application to the Traveling Salesman Problem (TSP). We choose MMAS and TSP to focus on algorithmic aspects of ACO and technical issues of GPU computing that are not problem dependent, as well as to strictly compare our results to the original works of Stützle and Hoos [28]. After a fairly complete review of the literature on parallel ACO, the proposed GPU parallelization strategies for

* Corresponding author.

E-mail addresses: audrey.delevacq@univ-reims.fr (A. Delévacq), pierre.delisle@univ-reims.fr (P. Delisle), mgravel@uqac.ca (M. Gravel), michael.krajecki@univ-reims.fr (M. Krajecki).

MMAS are explained. Finally, extensive experimental results are presented to evaluate and compare their performance.

2. Ant Colony Optimization for the traveling salesman problem

The Traveling Salesman Problem (TSP) is well-known in combinatorial optimization. It may be defined as a complete weighted directed graph $G = (V, A, d)$ where $V = \{1, 2, \dots, n\}$ is a set of vertices (cities), $A = \{(i, j) | (i, j) \in V \times V\}$ is the set of arcs, and $d : A \rightarrow \mathbb{N}$ is a function assigning a weight or distance (positive integer) d_{ij} to every arc (i, j) . The objective is to find a minimum weight Hamilton cycle in G , which is a path of minimal length visiting each city exactly once.

In most ACO algorithms, a given number of ants gradually and concurrently build tours using heuristic and pheromone information. Ants update pheromone values in the process to guide other ants to potentially better tours. In many cases, updates are performed according to the tours built and to various general rules. Following a given number of iterations where ants have built many – hopefully – improved solutions, the best one is chosen as the solution of the problem. A complete description of ACO can be found in Dorigo [16,17].

The Max–Min Ant System (MMAS) [28] is generally recognized as one of the most effective ACO algorithms at the present time. It also incorporates the main mechanisms and memory structures that are common to most algorithmic versions of this metaheuristic. Fig. 1 illustrates a simplified pseudo-code of the MMAS. In this algorithm, the number of ants m is set to the number of cities n . The ants tour construction process is performed in each of the ni iterations. To that end, each ant ant_k is initially placed on a randomly chosen city. Then, at each solution construction step, ant_k builds its tour T_k by repeatedly applying a state transition rule to choose the cities that will be added to its tour among the unvisited cities. After all ants have built their tour, pheromone τ is updated according to some rule which follows two objectives: to increase the desirability of arcs associated to the global best solution found so far T_{gl} or to the best solution of the current iteration T_{it} and to reduce the ones that have not been used. To avoid search stagnation, τ is kept between minimal and maximal values τ_{min} and τ_{max} on each arc. τ is also initialized at τ_{max} in order to facilitate exploration of the search space in the beginning of the algorithm. Moreover, MMAS uses a trail-smoothing mechanism which also promotes exploration by increasing the probability of choosing arcs with low pheromone values.

For better readability, the state transition and pheromone update rules are not explained in this paper. More information on these subjects may be found in the original works of Stützle and Hoos [28]. However, it is important to mention that the state transition rule computes the probability for each unvisited city to be chosen by the ant according to distance and pheromone values. These are stored in two $n \times n$ matrices that need to be available to each ant. Consequently, when implementing MMAS (as well as most ACO algorithms) on a real computer architecture, memory must be large enough to accommodate these data structures and fast enough to keep up with the numerous requests from processing elements. This requirement becomes more prohibitive as problem size increases.

When faced with large problems, MMAS uses candidate lists to reduce the possible cities to be chosen by ants during the tour construction phase. These lists contain, for each city, a given number of its cl nearest neighbors sorted in increasing order. Ants choose cities exclusively in candidate lists until all candidates are visited. Only in that case is an ant allowed to pick a city outside the lists.

Finally, MMAS may be augmented with a local search procedure such as 3-opt [21] to improve the solutions found by the ants. This

```

Initialize colony parameters
Initialize pheromone matrix  $\tau$  with  $\tau_{max}$  for each pair of cities
for  $i = 1$  to  $ni$  do
  for  $k = 1$  to  $m$  do {TOUR CONSTRUCTION}
    Place  $ant_k$  on a randomly chosen city
    while the  $n$  cities are not all visited do
      Move  $ant_k$  to next city according to state transition rule
      Compute length  $L_k$  of the tour  $T_k$  produced by  $ant_k$ 
  for  $k = 1$  to  $m$  do {LOCAL SEARCH}
    while  $T_k$  is improved do
      for all  $0 < a < n$  and  $0 < b < n$  and  $0 < c < n$  do
        Delete arcs  $(a, a + 1), (b, b + 1), (c, c + 1)$ 
        Produce  $T'$  by reconnecting partial tours with other arcs
        Compute  $L'$ 
        if  $L' < L_k$  then
          Update  $T_k$  with  $T'$ 
  for  $k = 1$  to  $m$  do
    if  $L_k < L_{it}$  then
      Update  $T_{it}$  with  $T_k$ 
  if  $L_{it} < L_{gl}$  then
    Update  $T_{gl}$  with  $T_{it}$ 
  Evaporate  $\tau$  and update  $\tau$  for each pair of cities of  $T_{it}$  or  $T_{gl}$ 
  Control  $\tau_{min} < \tau < \tau_{max}$  and update  $\tau$  with trail smoothing mechanism

```

Fig. 1. MMAS pseudo-code with local search.

method aims to improve a current solution by replacing at most three of its arcs. The process of replacing the current solution with an improved one is then iterated until no better solution is found.

ACO algorithms have proven to be successful in solving many academic and industrial combinatorial optimization problems [17]. However, faced with large and hard problems, they need a considerable amount of computing time and memory space to be effective in their exploration of the search space. Consequently, some interest in their parallelization has been raised in the recent years. The following section presents a literature review on parallel ACO.

3. Literature review on parallel Ant Colony Optimization

The concurrent nature of both tour construction and global search of the solution space makes the ACO metaheuristic a good candidate for parallelization. However, this potential comes with important challenges mainly due to pheromone management and to the size of the data structures that have to be maintained. Works on traditional, CPU-based parallel ACO can be classified into two general approaches: *parallel ants* and *multiple ant colonies*. These approaches are briefly explained in Sections 3.1 and 3.2. On the other hand, few authors have proposed parallel implementations dedicated to specific architectures. Section 3.3 is dedicated to these *hardware-oriented* approaches. In all cases, a survey of related works is also provided.

3.1. Parallel ants

Works related to the parallel ants approach, which aims to execute the ants tour construction phase on many processing elements, were initiated by Bullnheimer et al. [5]. They proposed two parallelization strategies for the Ant System on a message passing and distributed-memory architecture. The first one is a low-level and synchronous strategy that aims to accelerate computations by distributing ants to processors in a master-slave fashion. At each iteration, the master broadcasts the pheromone structure to slaves, which then compute their tours in parallel and send them back to the master. The time needed for these global communications and synchronizations implies a considerable overhead. The second strategy aims to reduce it by letting the algorithm perform a given number of iterations without exchanging information. The authors conclude that this partially asynchronous strategy is preferable due to the considerable reduction of the communication overhead.

متن کامل مقاله

دریافت فوری ←

ISIArticles

مرجع مقالات تخصصی ایران

- ✓ امکان دانلود نسخه تمام متن مقالات انگلیسی
- ✓ امکان دانلود نسخه ترجمه شده مقالات
- ✓ پذیرش سفارش ترجمه تخصصی
- ✓ امکان جستجو در آرشیو جامعی از صدها موضوع و هزاران مقاله
- ✓ امکان دانلود رایگان ۲ صفحه اول هر مقاله
- ✓ امکان پرداخت اینترنتی با کلیه کارت های عضو شتاب
- ✓ دانلود فوری مقاله پس از پرداخت آنلاین
- ✓ پشتیبانی کامل خرید با بهره مندی از سیستم هوشمند رهگیری سفارشات