J. Parallel Distrib. Comput. 72 (2012) 564–578

Contents lists available at SciVerse ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

A parallel and distributed meta-heuristic framework based on partially ordered knowledge sharing

Jinwoo Kim[a], Minyoung Kim[b], Mark-Oliver Stehr[b], Hyunok Oh[c], Soonhoi Ha[a,*]

[a] School of Computer Engineering, Seoul National University, Seoul, 151-744, South Korea
[b] Computer Science Laboratory, SRI international, Menlo Park, CA 94025, USA
[c] Department of information Systems, Hanyang University, Seoul, South Korea

ARTICLE INFO

Article history:
Received 1 June 2011
Received in revised form
3 December 2011
Accepted 21 January 2012
Available online 28 January 2012

Keywords:
Parallel and distributed optimization framework
Meta-heuristic
Knowledge sharing
Design space exploration

ABSTRACT

We propose a new distributed and parallel meta-heuristic framework to address the issues of scalability and robustness in the optimization problem. The proposed framework, named PADO (Parallel And Distributed Optimization framework), can utilize heterogeneous computing and communication resources to achieve scalable speedup while maintaining high solution quality. Specifically, we combine an existing meta-heuristic framework with a loosely coupled distributed island model for scalable parallelization. Based on a mature sequential optimization framework, we implement a population-based meta-heuristic algorithm with an island model for parallelization. The coordination overhead of previous approaches is significantly reduced by using a partially ordered knowledge sharing (POKS) model as an underlying model for distributed computing. The resulting framework can encompass many meta-heuristic algorithms and can solve a wide variety of problems with minimal configuration. We demonstrate the applicability and the performance of the framework with a traveling salesman problem (TSP), multi-objective design space exploration (DSE) problem of an embedded multimedia system, and a drug scheduling problem of cancer chemotherapy.

1. Introduction

Meta-heuristics [27] are a computational method that optimizes a problem by iteratively improving candidate solution(s) with a given measure of solution quality. Meta-heuristic algorithms make few or no assumption about the problem being optimized and can search numerous candidate solutions by tight incorporation of both random and probabilistic methods. Meta-heuristic algorithms aim for a near optimal solution within tractable time for NP-complete combinatorial optimization problems.

Meta-heuristic optimization algorithms can be classified into two types: population-based and single-state. In population-based optimization, solution candidates (i.e., members of population) keep evolving into better ones via interaction of individuals within a population. Genetic algorithm (GA) [12], particle swarm optimization (PSO) [19], and differential evolution (DE) [37] are examples of population-based optimization algorithms. In contrast, the single-state (also known as local search or solution-based) optimization starts with an initial candidate solution and modifies it to achieve a better solution. Hill climbing (HC) [33], simulated annealing (SA) [23], and Tabu search (TS) [15] are examples of single-state optimization algorithms. Note that population-based optimization is exploration oriented due to its capability to broadly cover the solution space. In contrast, single-state optimization is exploitation oriented, because it mainly improves the quality of a local solution.

In this paper, we are concerned with population-based optimization to solve multi-objective optimization problems. Because a multi-objective problem usually has a set of Pareto-optimal solutions, we need to broadly explore the solution space. The current implementation of the proposed technique is based on the Opt4J [6] framework, which supports various meta-heuristic optimization algorithms applicable to a wide variety of optimization problems.

In population-based optimization algorithms, the computation is often very demanding because of a large population size, a complex fitness evaluation, and/or a stringent convergence threshold. Thus, several parallel techniques for meta-heuristic algorithms have been proposed to enhance the convergence speed with acceptable solution quality. The taxonomy of the existent parallel techniques and their basic concepts can be found in [27,7,38].

The well-known parallelization techniques for population-based meta-heuristic algorithms are of two types: a master–slave model and an Island model. The master–slave model includes a master and one or more corresponding slaves being assigned to

* Corresponding author.
E-mail addresses: jwkim@iris.snu.ac.kr (J. Kim), mkim@csl.sri.com (M. Kim), stehr@csl.sri.com (M.-O. Stehr), hoh@hanyang.ac.kr (H. Oh), sha@snu.ac.kr (S. Ha).

0743-7315/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2012.01.007

evaluate an individual when a master needs to assess the fitness of an individual. Unlike the master–slave model, the Island model divides the entire population into separate sub-populations, permitting populations to overlap. Each Island proceeds with the local evolution of its sub-population on potentially heterogeneous computational resources. Occasionally, the Island model asynchronously performs migration steps (i.e., exchange of individuals) to disseminate new solution candidates to improve the overall solution quality in a cooperative manner.

Both models have advantages and limitations. In the master–slave model, a robust and powerful master node can serve as a centralized repository of the population, which simplifies data collection, analysis, and recovery from failures of slave nodes. However, the master node can be both a performance bottleneck and a single source of failure. The Island model offers high evolution speed due to decreased sub-population size while compensating low solution diversity by migration. The distributed nature of the Island model also provides robustness against node failures. Motivated by the need for a scalable and robust solution, we adopt the Island model for parallelization of meta-heuristic algorithms.

In this approach, multiple Islands (i.e., sub-populations) independently evolve and asynchronously exchange their individuals. To achieve good solution quality, the likelihood of being trapped in a local optimum needs to be reduced by increasing population diversity. The population diversity can be improved by increasing the migration ratio, which has been the main approach in previous work. Increasing the migration ratio, however, can adversely affect the convergence speed, because it increases the communication and synchronization overhead. We must also account for the Island topology, as it restricts the migrations. We propose to adopt the loosely coupled distributed computational model of partially ordered knowledge sharing (POKS) [36] for minimizing the communication overhead and improving the scalability of distributed execution.

In the POKS model, processors disseminate knowledge, a locally optimal solution in our context, to the other processors. Each processor receives multiple units of knowledge from other processors. We define an ordering on knowledge by means of the solution quality in our context. Each processor maintains the knowledge of the highest order among which it receives. Different from those in a synchronized approach, our system's processors may see different versions of the knowledge at a given point in time during the process of evolution. The in-network replacement of inferior knowledge decreases the communication overhead by propagating only winning individuals. By not relying on the existence of other nodes or persistent connectivity, the POKS model offers robustness against node/communication failures.

In this paper, we propose a novel methodology to parallelize population-based meta-heuristic algorithms, which is realized in our framework, named PADO (Parallel And Distributed Optimization framework).[1] PADO consists of two parts: The frontend is based on the Opt4J framework that provides user interfaces for algorithm and problem specification. The backend is the cyber-application framework [22] (in short, cyber-framework) that provides a programming environment for representing, manipulating, and sharing knowledge across the network under minimal assumptions on connectivity based on the POKS model. The cyber-framework supports shared-memory *parallelism* as well as *distributed* modes of operation. PADO bridges Opt4J and the cyber-framework via a parallel processing parameter configuration file. Both Opt4J and the cyber-framework provide configuration parameters that are easily adjusted depending on the optimization problem. As a result, a sequential version of the algorithm and problem specified in Opt4J is converted and executed in a parallel and distributed Island model on the cyber-framework with minimal configuration effort in the PADO framework.

To prove effectiveness of our approach, we address the design space exploration (DSE) problem in an embedded system design, one of the numerous application areas of meta-heuristic algorithms, as a real-life case study. In an embedded system design, system performance can greatly benefit from optimization of key design parameters. To find optimal parameters with acceptable overhead, DSE is often performed by using meta-heuristic algorithms. Examples of this DSE problem include optimization of task scheduling [42]; memory architecture [20,18]; bus architecture [21]; automotive communication architecture [34,31]; and energy consumption [17]. In many cases, the design-optimization process must deal with multiple objective functions, which requires Pareto optimality in the solution. In this paper, the DSE problem is to find an optimal mapping of tasks to the processors, considering the trade-offs between throughput performance and resource requirements. For this, we must determine the number of processors, the buffer sizes for data communication, and the static mapping of the tasks to the processors. Additionally, we use the TSP (traveling salesman problem) as well as a drug scheduling problem to show the viability of the proposed framework.

The paper is organized as follows: in Section 2, we introduce a DSE problem that is solved in this paper as an illustrative optimization example. In Section 3, we present some background on the POKS model, the cyber-framework, and Opt4J. Section 4 presents the detailed architecture of PADO. In Section 5, we demonstrate the experimental results from three case studies. The first tackles a TSP that is a well-studied NP-complete problem. The second case study tries to solve the DSE problem presented in Section 2 and the last case study solves a drug scheduling problem of cancer chemotherapy [40]. In Section 6, we review existing parallel meta-heuristic frameworks for comparison. We present our main contributions and conclude our paper in Section 7, along with a description of future work.

2. A motivational example

The DSE problem used as a motivational optimization example is an extended version of [35]. We briefly recast the problem in this section. We assume that an application is specified by a synchronous dataflow (SDF) graph [25]. SDF specification is widely used for stream-based signal-processing applications, in which throughput is usually more important than latency. Also SDF specification is suitable for multicore embedded system design, because the graph explicitly specifies the parallelism of an application. In an SDF graph, a node represents a computation task and an arc indicates a FIFO communication channel between two adjacent nodes. Fig. 1(a) shows a simple SDF graph. The number of data units (called sample rate) to be produced or consumed per node execution is an integer that is greater than or equal to one. A node is executable when it has a sufficient number of samples accumulated on all its input arcs. The sample rate is specified on each arc in Fig. 1(a). From the sample rates, we can determine the ratio between the execution rates of two adjacent nodes. For example, Node B will be executed twice as fast as Node C. If we denote $r(X)$ as the execution rate of Node X, the ratio between the execution rates becomes the following: $r(A) : r(B) : r(C) = 3 : 2 : 1$. The minimum number of executions that satisfy this ratio defines an *iteration* of the SDF graph.

For parallel execution of a given SDF graph, we assume that nodes are mapped to the processors at compile-time (static mapping), and the execution order of nodes on each processor is determined at run-time (dynamic scheduling). Fig. 1(b) shows

[1] PADO means the "wave" in Korean. In our approach the nodes cooperate by emitting waves of knowledge, which interfere until all local solutions asynchronously converge to a global solution.