Engineering Applications of Artificial Intelligence 23 (2010) 177–187

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Parallel scalable hardware implementation of asynchronous discrete particle swarm optimization

Amin Farmahini-Farahani *, Shervin Vakili, Sied Mehdi Fakhraie, Saeed Safari, Caro Lucas

School of Electrical and Computer Engineering, University of Tehran, North Kargar Ave., Tehran 14395-515, Iran

ARTICLE INFO

Article history:
Received 13 June 2007
Received in revised form
17 August 2009
Accepted 1 December 2009
Available online 12 January 2010

Keywords:
Evolutionary algorithms
Particle swarm optimization
Parallel architecture
Multiprocessing
Field programmable gate array (FPGA)
System-on-a-programmable-chip
Real-time applications

ABSTRACT

This paper presents a novel hardware framework of particle swarm optimization (PSO) for various kinds of discrete optimization problems based on the system-on-a-programmable-chip (SOPC) concept. PSO is a new optimization algorithm with a growing field of applications. Nevertheless, similar to the other evolutionary algorithms, PSO is generally a computationally intensive method which suffers from long execution time. Hence, it is difficult to use PSO in real-time applications in which reaching a proper solution in a limited time is essential. SOPC offers a platform to effectively design flexible systems with a high degree of complexity. A hardware pipelined PSO (PPSO) Core is applied with which the required computational operations of the algorithm are performed. Embedded processors have also been employed to evaluate the fitness values by running programmed software codes. Applying the subparticle method brings the benefit of full scalability to the framework and makes it independent of the particle length. Therefore, more complex and larger problems can be addressed without modifying the architecture of the framework. To speed up the computations, the optimization architecture is implemented on a single chip master–slave multiprocessor structure. Moreover, the asynchronous model of PSO gains parallel efficacy and provides an approach to update particles continuously. Five benchmarks are exploited to evaluate the effectiveness and robustness of the system. The results indicate a speed-up of up to 98 times over the software implementation in the elapsed computation time. Besides, the PPSO Core has been employed for neural network training in an SOPC-based embedded system which approves the system applicability for real-world applications.

1. Introduction

Evolutionary algorithms (EAs) are general-purpose search algorithms used to solve difficult numerical optimization problems by simulating natural evolution over populations of candidate solutions (Schwefel, 1981; Holland, 1975; Fogel, 1994; Bäck et al., 1997). Numerical optimization has been widely used in engineering to solve a variety of NP-complete problems in areas such as structural optimization, neural network training, layout and scheduling problems, and control system analysis and design (Fogel, 1991; Bäck et al., 1997; Zitzler et al., 2000; Freitas, 2002; Bäck, 1996; Deb, 2001; Dasgupta and Michalewicz, 1997). Particle swarm optimization (PSO) is one of the emerging computation techniques that was developed in 1995 (Kennedy and Eberhart, 1995) as an evolutionary optimization methodology over a complex solution space. PSO deals with the concept of social interaction. It was inspired by the social behavior of bird flocking or fish schooling. The PSO algorithm exploits the gathered information of the particles in a swarm during their food-searching activities and affects the trajectory of particles. Each particle flies through the search space with a velocity which is dynamically adjusted according to its own experience as well as the experiences of its neighbors. Therefore, the particles have a tendency to fly towards the better and better search area over the progress of search process.

Like the other evolutionary computational techniques, PSO is a derivative-free, stochastic and population-based search algorithm which is initialized with a population (swarm) of random solutions, called particles. Unlike the other evolutionary computation techniques, each particle in PSO is also associated with a velocity.

The PSO-based approaches converge faster than genetic-algorithm-(GA)-based techniques, and require less computational complexity (Song and Gu, 2004). PSO has few parameters to adjust and general values for these parameters are not devastative (Carlisle and Dozier, 2001). Moreover, PSO is well suited to large-scale

* Corresponding author. Silicon Intelligence and VLSI Signal Processing Laboratory of School of Electrical and Computer Engineering, University of Tehran, North Kargar Ave., Tehran 14395-515, Iran. Tel.: +98 21 88013196; fax: +98 21 88778690.

E-mail addresses: a.farmahini@ece.ut.ac.ir (A. Farmahini-Farahani), s.vakili@ece.ut.ac.ir (S. Vakili), fakhraie@ut.ac.ir (S.M. Fakhraie), saeed@ut.ac.ir (S. Safari), lucas@ipm.ir (C. Lucas).

0952-1976/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2009.12.001

and complex optimization problems and is mainly used in NP-complete problems (Hu et al., 2004). PSO is inherently parallel since each particle can be considered as an independent agent (Schutte et al., 2004). However, the iterative evolution process of PSO like other heuristic algorithms is time consuming. For many real-world applications, PSO can run for days, even when it is executed on a high performance workstation. Computational parallelism is an applicable approach to alleviate the problem of the prolonged execution time of PSO.

Particle swarm engines are one instance of the bio-inspired computing systems that is employed in the embedded systems. Considering the real-time requirements for embedded applications, most embedded processor cores lack the performance to run particle swarm computations or to emulate other bio-inspired subsystems. Therefore, employing particle swarm in embedded applications requires efficient custom hardware intellectual property (IP) cores implemented for them (Mathew et al., 2004). On the other hand, the competitive market of embedded systems requires solutions that take shorter time in design, are cost-efficient in development, have flexibility in utilization, expose simplicity in integration, and exhibit reusability (Zhang et al., 2001). A particle swarm IP core for embedded systems should offer these capabilities as an off-the-shelf component. Nevertheless, obtaining an optimal solution (if exists) in real time for large-scale problems is difficult. Since making a decision in a limited time is vital for many practical problems, obtaining a suitable solution in real time is much better than finding the optimal solution off-line. Thus, our objective is to find appropriate solution in a limited time.

The complexity of the modern chips is rising and fundamental changes in system design are being more essential. The system-on-a-programmable-chip (SOPC) concept is bringing a major revolution in the design of integrated circuits, due to the flexibility it provides and the complexity it caters to. The SOPC embedded systems refer to the packaging of all the necessary electronic functions, memory blocks, interfaces, microprocessors, and so forth of different functions onto a single chip to form a complete electronic system. SOPC brings the combination of programmable logic and embedded processors, mixing software and hardware. Thus, SOPC technology allows all of the various components to be integrated together on a chip rather than connecting those components on a circuit board to construct an electronic system.

In this paper, an on-chip multiprocessing SOPC-based architecture has been proposed for high performance realization of the particle swarm algorithm in embedded systems to speed up its calculations. The proposed architecture can obtain high computational power due to its parallel processing architecture. In addition, the software fitness evaluator along with the parameterized hardware PSO Core provide a scalable framework for a range of discrete PSO applications. Hence, the architecture is intended to facilitate the use of PSO-based techniques in embedded systems. The implemented system performs fitness evaluation in software and all other PSO operations in hardware. In addition, the employed master–slave parallel system uses an asynchronous and discrete model of PSO. The proposed system has been implemented on an Altera Stratix Development Kit, and its performance has been compared with that of the corresponding software implementation. Test beds are MaxOne problem and optimizing four classic arithmetic functions.

The remaining sections are organized as follows: Section 2 reviews the previous work on applications and hardware implementations of the PSO algorithm. Section 3 gives an overview of the basis of PSO, different ways to deal with particles, and parallel characteristics of PSO. The proposed framework is described in Section 4. Also, the implementation of the system and details of the employed architecture are described in this section. Section 5 explains the experimental results over five benchmarks and compares the results with the software-based implementation as well as pure-hardware implementation. Section 6 presents an adoption of our PSO implementation for neural network training as a case study of a real-world application. Section 7 concludes the paper and explains our future work.

2. Previous work

Particle swarm optimization is a new population-based stochastic optimization technique. More and more researchers are interested in this new algorithm and it has been investigated from various perspectives (Song and Gu, 2004; Hu et al., 2004). To reduce the execution time of heuristic algorithms, several methods have been offered, including parallel and/or distributed processing of these algorithms along with their hardware implementation (Konfrst, 2004; Chen et al., 2005; Calaor et al., 2002; Hidalgo et al., 2003).

Different types of parallel implementations of PSO have been introduced in literature. Early parallel PSO implementations have employed synchronous evolution methods (Schutte et al., 2004; Chang et al., 2005; Cui and Weile, 2005; Jin and Rahmat-Samii, 2005), while asynchronous evolution has been emphasized recently (Koh et al., 2006; Venter and Sobieszczanski-Sobieski, 2005). Parallel implementations of PSO are mostly based on cluster computing and message passing interface protocol (Koh et al., 2006; Schutte et al., 2004, 2003; Venter and Sobieszczanski-Sobieski, 2005; Gies and Rahmat-Samii, 2003; Jin and Rahmat-Samii, 2005).

Although different hardware approaches have been proposed for GA so far (Zhu et al., 2006), a few hardware-based implementations of PSO have been reported. In Kokai et al. (2006), the authors have used hardware implementation of PSO in an FPGA for the employment with blind adaptation of the directional characteristic of array antennas. They have introduced multi-swarm architecture in which each single swarm optimizes only a single parameter of the application. However, they have not mentioned the hardware implementation cost, achieved frequency, and also performance.

In Reynolds et al. (2005), the authors have implemented a modified particle swarm optimizer and neural network in FPGA. In their architecture, many of the computations are preformed in parallel to reduce computation time as compared to software implementation. They have employed two Xilinx XC2V6000 FPGAs. One FPGA was used for fitness function calculations, and the other was used for the particle swarm operations. At 100 MHz, their implementation was about 60 times faster than software implementation. Also, they have not reported the hardware cost.

In Pena et al. (2006), a hybrid swarm optimization technique has been offered to accommodate embedded or hardware dedicated applications. Their approach does not require multiplication and consists of a population of neural networks in an FPGA that are evaluated in an embedded processor and are trained by the proposed algorithm. In Pena and Upegui (2007), they have presented an architecture for a hardware-friendly version of PSO. The architecture is composed of a number of particle computation blocks connected to each other to shape a ring topology. Each computation block consists of memory units, a 6-stage pipelined particle update unit, and a personal best update unit. Each computation block updates a particle, performs particle computation, and is connected to a fitness evaluation block. All the particles in the swarm are evaluated and updated in parallel, increasing hardware cost linearly with the population size. The architecture has been implemented on a Xilinx Virtex-4