Information Sciences 453 (2018) 463–485

Contents lists available at ScienceDirect

Information Sciences

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

A neighbor-based learning particle swarm optimizer with short-term and long-term memory for dynamic optimization problems

Leilei Cao[a], Lihong Xu[a,*], Erik D. Goodman[b]

[a] Department of Control Science and Engineering, Tongji University, Shanghai 201804, China
[b] BEACON Center for the Study of Evolution in Action, Michigan State University, East Lansing, MI 48824 USA

ARTICLE INFO

Article history:
Received 17 July 2017
Revised 8 March 2018
Accepted 15 April 2018
Available online 21 April 2018

Keywords:
Neighbor-based learning
Particle swarm optimization
Worst-replacement
Short-term and long-term memory
Dynamic optimization problems

ABSTRACT

This paper presents a novel Particle Swarm Optimization algorithm to address Dynamic Optimization Problems. The algorithm incorporates a neighbor-based learning strategy into the velocity update of Particle Swarm Optimization, in order to enhance the exploration and exploitation capabilities of particles. Unlike the traditional swarm update scheme, a “worst replacement” strategy is used to update the swarm, whereby the position of the worst particle in the swarm is replaced by a better newly generated position. The short-term memory is employed to store solutions with intermediate fitnesses from the most recent environment, and the long-term memory is to store the historical best solutions found in all previous environments. After an environmental change is detected, some particles' positions in the swarm are replaced by the members of the short-term memory, and the best member in the long-term memory under the current environment is re-introduced to the active swarm along with its Gaussian neighborhood, then the remaining particles' positions are re-initialized. The performance of the proposed algorithm is compared with six state-of-the-art dynamic algorithms over the Moving Peaks Benchmark problems and Dynamic Rotation Peak Benchmark Generator. Experimental results indicate that out algorithm obtains superior performance compared with the competitors.

1. Introduction

In general, real-world optimization problems can be categorized into two groups: design/static problems, and operation/dynamic problems. Static problems include engineering structural design, travelling salesman problems, knapsack problems, job shop scheduling problems, vehicle routing problems, etc. Dynamic problems include online-control optimization problems, online path planning problems, investment portfolio problems, etc. [13]. Problems of this sort are called Dynamic Optimization Problems (DOPs) or optimization problems in dynamic environments, in which fitness functions, design variables, or constraints may change over time [37]. In real-world dynamic optimization problems, the problem may change in various ways. In this paper, we only focus on those problems with changeable fitness functions and sometimes, a changeable number of design variables, both of which change only intermittently.

* Corresponding author.
E-mail addresses: mcaoleilei@sina.com (L. Cao), xulhk@163.com (L. Xu), goodman@egr.msu.edu (E.D. Goodman).

https://doi.org/10.1016/j.ins.2018.04.056
0020-0255/

Evolutionary Algorithms (EAs) have been widely and successfully employed on static optimization problems because of their excellent performance. Most heuristic algorithms including EAs, have the capability to locate optimal or near-optimal solutions, although sometimes not a global optimum. However, for dynamic optimization problems, locating a global optimum accurately is not the only aim—continuing to track the global optimum in a dynamic environment is a more important task [9,18]. For traditional Evolutionary Algorithms, the population may have converged on a global optimum when solving an optimization problem, which means that all individuals are the same or close in Euclidean distance, and it is difficult to track a new optimum due to a lack of diversity. Therefore, traditional EAs must be modified to adapt to dynamic environments. Several approaches have been combined with EAs to address DOPs, including hyper-mutation [8], immigrant schemes [46], memory [45] or archive [44], prediction [41,48] and multi-population schemes [21].

Differential Evolution (DE) [42] and Particle Swarm Optimization (PSO) [16] are two typical EAs, which have been widely employed for addressing optimization problems since 1995. Unlike traditional EAs, DE variants are generated from scaled differences of randomly selected individuals [12]. The moving velocities of particles in PSO are influenced by their individual past experiences and the global best particle in the current swarm [16]. With appropriate parameters, both of them have excellent optimizing capabilities on some optimization problems. When addressing high-dimensional or multi-modal optimization problems, DE typically converges slowly or be trapped in a local optimum, while PSO also converges prematurely. In dynamic environments, a converged population of DE or PSO cannot readily evolve to a new global optimum. In most DOPs, high dimensionality or multi-modality and unknown changes are typical characteristics. An ideal algorithm should keep diversity in the population and have a fast convergence speed when solving DOPs. Yet more ideal for DOPs that often undergo limited magnitudes of change or change among a set of recurring environments would be an algorithm that learned useful information about those environments and was able to draw upon that information when detecting an environmental change.

This research to develop a novel algorithm is strongly motivated by some new learning strategies of modified particle swarm optimization [7,23,28] and the effectiveness of memory for DOPs [37]. When applied to DOPs, most PSO variants did not change the learning strategy of particles [2,25,29,39]. Each particle still learns from its personal best position and global (or local) best position. Various authors have realized that involving of other neighbors could enhance the exploration of particles, but also involving the best performer was abandoned by most of them [7,28]. Consequently, the search efficiency was reduced. The principle of DE tells us that involving different stochastically chosen individuals could generate widely distributed offspring, which increase the diversity of the population [12,42]. Therefore, on the basis of still retaining the involvement of the best performer, we can introduce some neighboring stochastic particles into the velocity update of PSO. Instead of learning from personal historical best position and global best position, a target particle learns from another randomly chosen particle among its neighbors and the global best one in the swarm, in which all other members are defined as neighbors of the target particle. This learning strategy tends to enhance the exploration and exploitation of particles and to decrease the risk of falling into a local optimum. Since each particle has the same opportunity to be selected as a leader for a given step, particles with poor fitnesses might adversely affect the search process of the swarm. That is to say the performance of the whole swarm could be strongly degraded by the worst particle under the proposed learning strategy. The worst particle is therefore given priority to be updated, which we call “worst replacement.” Also, many of researchers have used a memory or archive scheme to store the best solution of the population before an environmental change is detected, which has proven effective for those changing environments with small severity. Motivated by this idea, we introduce more solutions (not only the best solution) to be stored in a memory set.

Drawing this together, this paper proposes a novel dynamic evolutionary algorithm called neighbor-based learning particle swarm optimization with short-term and long-term memory (NLPSO) to address dynamic optimization problems. We incorporate three new strategies into the velocity update of PSO: neighbor-based learning, learning probability, and worst-replacement. For each particle update, two particles are selected to be its leaders—a randomly chosen one from its neighbors and another one with global best fitness in the swarm. Learning probability means that a particle’s velocity update can either be influenced by its leaders or retain its previous velocity components. After a new better position is generated, the worst particle in the swarm moves to that new better position. To the best of our knowledge, this learning strategy has not been previously proposed. In order to re-use information regarding locations of optima in the previous environments, a short-term and long-term memory scheme is designed to store some solutions with intermediate fitnesses in the most recent environment (short-term memory), and the historical best solutions in all previous environments are also stored (long-term memory). The solutions with intermediate fitnesses remain until replaced by new superior ones, and the historical best solutions in all previous environments remain forever to be re-evaluated in later new environment. Members in the memory are selectively re-introduced to the active swarm after an environmental change is detected. When solving those dynamic optimization problems with multiple optima, we employ a multi-swarm method in which several swarms search independently. In addition, an exclusion rule is employed to prevent different swarms from converging on the same peak. The performance of the proposed algorithm is compared with six state-of-the-art dynamic EAs that have been proposed to solve DOPs, and the comparison is on the moving peaks benchmark (MPB) problems and the dynamic rotation peak benchmark generator (DRPBG).

The rest of this paper is organized as follows. Section 2 provides a brief description of PSO, along with the performance of PSO and DE variants, as modified for dynamic problems, when applied in dynamic environments introduced in the literature. Section 3 details the proposed algorithm: a neighbor-based learning particle swarm optimization with short-term and long-term memory. Section 4 describes the benchmark, the performance metrics and the experimental settings. In Section 5, the