Chaos, Solitons & Fractals 73 (2015) 29–35

Contents lists available at ScienceDirect

Chaos, Solitons & Fractals

Nonlinear Science, and Nonequilibrium and Complex Phenomena

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

Particle swarm optimization for automatic creation of complex graphic characters

Iztok Fister Jr. [a], Matjaž Perc [b,c,*], Karin Ljubič [d], Salahuddin M. Kamal [c], Andres Iglesias [e,f], Iztok Fister [a]

[a] Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova 17, SI-2000 Maribor, Slovenia
[b] Faculty of Natural Sciences and Mathematics, University of Maribor, Koroška cesta 160, SI-2000 Maribor, Slovenia
[c] Faculty of Science, King Abdulaziz University, Jeddah, Saudi Arabia
[d] Faculty of Medicine, University of Maribor, Taborska 8, SI-2000 Maribor, Slovenia
[e] Department of Applied Mathematics and Computational Sciences, University of Cantabria, Avda. de los Castros, 39005 Santander, Spain
[f] Department of Information Science, Faculty of Sciences, Toho University, 2-2-1 Miyama, 274-8510 Funabashi, Japan

ARTICLE INFO

Article history:
Received 25 November 2014
Accepted 22 December 2014
Available online 21 January 2015

ABSTRACT

Nature-inspired algorithms are a very promising tool for solving the hardest problems in computer sciences and mathematics. These algorithms are typically inspired by the fascinating behavior at display in biological systems, such as bee swarms or fish schools. So far, these algorithms have been applied in many practical applications. In this paper, we present a simple particle swarm optimization, which allows automatic creation of complex two-dimensional graphic characters. The method involves constructing the base characters, optimizing the modifications of the base characters with the particle swarm optimization algorithm, and finally generating the graphic characters from the solution. We demonstrate the effectiveness of our approach with the creation of simple snowman, but we also outline in detail how more complex characters can be created.

1. Introduction

Computer games are one of the most profitable products of the interactive entertainment industry. Worldwide, the number of active users is staggering, and this largely regardless of age, gender and ethnicity. The positive trend in the number of user since the 80's has been spurred on further with the advent of massive online multi-player games. The demand on the technology to deliver ever more authentic and exhilarating user experience is of course also increasing, and this creates a continuous need for innovation and improvement at many different levels of the industry. In addition to the obvious importance of hardware improvement, the development of new software, new interaction possibilities, as well as a firm positioning in the market all demand expertise and skill. At the juncture of these different aspects of development, theoretical research in computer science, applied mathematics and complex systems also plays a key role. And it is to the latter subject in particular that we aim to contribute with this paper.

In the past decades, the methods and techniques employed first in computer games have been applied with notable success in computational agent-based system research [21]. Many studies have also looked at video games as an art form on its own right [8]. At some universities and research institutes, computer games have become the subject on which one could do a masters or a PhD. Most notably, however, video games have had the most significant impact on the development of personal computers. Due to popular demand, personal computers had to improve steadily in the amount of RAM, processor

* Corresponding author at: Faculty of Natural Sciences and Mathematics, University of Maribor, Koroška cesta 160, SI-2000 Maribor, Slovenia.
E-mail address: matjaz.perc@um.si (M. Perc).

http://dx.doi.org/10.1016/j.chaos.2014.12.019
0960-0779/

speed, display resolution and graphic card capabilities to keep with the pace of more and more sophisticated games.

Simultaneously with the development of hardware, software methods have also arisen for solving the hardest real-world problems. As many of these problems are NP-hard [19], many heuristical algorithms have been proposed for solving such problems approximately. The majority of these heuristics take inspiration from nature. Two branches of algorithms are particularly known for their close ties with nature, namely the evolutionary algorithms (EA) [9] and swarm intelligence algorithms (SI) [4]. The former are inspired by the Darwinian principle of "only the fittest survive" [26], while the later take from social insects, swarms of birds and flocks of fish to execute complex actions based on elementary and indeed very simple interactions among individuals [38]. An intersection of both is the emergence of collective behavior in evolutionary games [28].

In general, it has been established that using advanced nature-inspired algorithms in computer graphics can be very useful. Applications to this domain include the EA in [33,34], where genetic algorithms were introduced to computer graphics. Later on, other evolutionary algorithms were also applied [14,2], while the most recent studies include also genetic algorithms [18] for polynomial B-spline surface reconstruction, particle swarm optimization [16] for the same problem and particle swarm optimization for Bézier surface reconstruction [15]. For example, Gálvez and Iglesias [17] applied the firefly algorithm [10,11] for polynomial Bézier surface parameterization, which is a well known problem in computer graphics. Further to these advances, creative 3D shape modeling has been proposed, where an initial population of 3D models is evolved to produce novel shapes in the next generation [40]. A new emerging field termed search-based procedural content generation is also growing rapidly, the aim of which is to use metaheuristics in automatic generation of content for games [36].

Since many games operate with a large number of different characters, we here address a very basic problem in a new way. In particular, we explore the usefulness of particle swarm optimization (PSO) for the automatic creation of complex graphic characters. We show how a modified particle swarm optimization algorithm can be used to create a gallery of 2D characters with different colors and shapes using a predefined set of parts. For the approach to work, the problem is defined as constraint satisfaction problem (CSP). Each candidate solution that satisfies all constraints is placed in a gallery from which the best characters can be selected by the game designer. As an example, we show the generation of 2D snowman, but we also outline the approach for the creation of 3D and more complex characters.

The remainder of this paper is structured as follows. Section 2 reviews the basics of the constraint satisfaction problem. In Section 3, we present particle swarm optimization algorithm for the automatic creation of graphic characters. The experiments and results are presented in Section 4. We conclude with a summary and directions for future development, as well as with a broader discussion about the relevance of nature-inspired algorithms for the emergence of intelligent design and cognition [29].

2. Constraint satisfaction problems

The majority of intractable (NP-hard) real-world problems are constrained [19]. That means, not all of candidate solutions obtained in EA and SI are valid after performing various variation operators. Usually, these operators act blindly according to constraints, i.e., these do not provide valid solutions. The constraint satisfaction problems (CSP) consist of three components $\langle \mathbf{x}, \mathbf{D}, \mathbf{C} \rangle$ [30], where.

- $\mathbf{x}$ is a vector of variables $\mathbf{x} = (x_1, \ldots, x_n)$, where n denotes a dimension of the problem,
- $\mathbf{D}$ is a set of domains $\mathbf{D} = \{D_1, \ldots, D_n\}$ for each variable of $\mathbf{x}$,
- $\mathbf{C}$ is a set of constraints $\mathbf{C} = \{C_1, \ldots, C_m\}$ that specify allowable combinations of values.

Each domain is determined by its lower and upper bounds $D_i \in [lb_i, ub_i]$. Each constraint C_i consists of a pair $\langle scope, relation \rangle$, where *scope* determines variables that participate in the constraint and *relation* defines an relation that needs to be valid in order to satisfy the so named feasibility condition ϕ. The feasibility condition for each candidate solution is composed of constraints and can be expressed as conjunction of constraints $\phi(\mathbf{x}) = \wedge_{i=1\ldots m} C_i(\mathbf{x})$. In fact, the feasibility condition is true if and only if all constraints are satisfied.

There are two main types of constraints handling in EA and SI [9]:

- Indirect, where constraints are transformed into optimization objectives,
- Direct, where the population based algorithms keep alone account about constraints during the run.

In this paper, indirect constraint handling was considered, where the proper solution is found, when the feasibility condition is satisfied. In this case, the value of fitness function is zero, i.e., $\phi(\mathbf{x}) = true$ if and only if $f(\mathbf{x}) = 0$.

3. Automatic creation of graphic characters

Particle swarm optimization (PSO) was one of the oldest swarm intelligence algorithms that was introduced at an International conference on Neural Networks by Kennedy and Eberhart in 1995 [23]. PSO is inspired by the social foraging behavior of some animals such as flocking behavior of birds and schooling behavior of fish. There are some individuals with better developed instinct for finding food, in both animal species. According these individuals, the whole swarm is directed into more promising regions in the fitness landscape.

The PSO is a population-based algorithm that consists of particles $\mathbf{x}_i^{(t)} = (x_{i,1}, \ldots, x_{i,n})^T$ for $i = 1 \ldots Np$ representing their position in a n-dimensional search space. Thus, the variable Np limits the number of particles in the population. The particles move across the search space