Computers & Industrial Engineering 65 (2013) 322–330

Contents lists available at SciVerse ScienceDirect

Computers & Industrial Engineering

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

# PSO with path relinking for resource allocation using simulation optimization

M.S.R. Martins, S.C. Fuchs, L.U. Pando, R. Lüders *, M.R. Delgado

*Federal University of Technology – Paraná, Av. Sete de Setembro 3165, 80230-901 Curitiba, PR, Brazil*

ARTICLE INFO

*Article history:*
Received 21 March 2012
Received in revised form 30 January 2013
Accepted 5 February 2013
Available online 19 February 2013

*Keywords:*
Particle swarm optimization
Discrete event simulation
Task assignment
Oil industry

ABSTRACT

This paper proposes a PSO-based optimization approach with a particular path relinking technique for moving particles. PSO is evaluated for two combinatorial problems. One under uncertainty, which represents a new application of PSO with path relinking in a stochastic scenario. PSO is considered first in a deterministic scenario for solving the Task Assignment Problem (TAP) and hereafter for a resource allocation problem in a petroleum terminal. This is considered for evaluating PSO in a problem subject to uncertainty whose performance can only be evaluated by simulation. In this case, a discrete event simulation is built for modeling a real-world facility whose typical operations of receiving and transferring oil from tankers to a refinery are made through intermediary storage tanks. The simulation incorporates uncertain data and operational details for optimization that are not considered in other mathematical optimization models. Experiments have been carried out considering issues that affect the choice of parameters for both optimization and simulation. The results show advantages of the proposed approach when compared with Genetic Algorithm and OptQuest (a commercial optimization package).



## 1. Introduction

Particle Swarm Optimization (PSO) is a technique developed by Engelbrecht (2007), Kennedy and Eberhart (1995). It is inspired on the behavior of birds in flocks where solutions to a given optimization problem, called particles, "fly" (like birds) through a multidimensional search space. Similarly to Genetic Algorithms (GAs), PSO can be classified as a bio-inspired paradigm. PSO was first developed for continuous optimization problems. However, research of PSO applications for combinatorial optimization problems can be found in the recent literature (Wang, Cai, Zhou, Wang, & Li, 2011; Kennedy & Eberhart, 1997; Rosendo & Pozo, 2010; Souza, Goldbarg, & Goldbarg, 2006; Hu, 2011). In discrete spaces a technique named path relinking (Glover et al., 1999) can be used to move a particle (solution) toward another one, generating a path between the two solutions.

This paper proposes a PSO-based optimization approach where a particular path relinking technique is adopted for moving particles. Additionally, the PSO algorithm is modified to deal with randomness. In this case, simulation is used whenever the fitness of a particle cannot be captured by a simple computation. PSO has been chosen mainly due the fact that it is easy to implement with few parameters to adjust, and interesting results have also been obtained with PSO for combinatorial optimization (literature has demonstrated it gets better results in a faster and cheaper way when compared with other methods (Poli, 2008)). Two applications are considered in this paper to emphasize these PSO properties. Results for both applications are compared with those obtained by a Genetic Algorithm-based approach. In the first part of experiments, PSO is considered in a simple scenario (the Task Assignment Problem – TAP) where simulation and randomness are not present. In the second part, PSO is applied in a more complex scenario involving simulation whose results are also compared with those obtained by OptQuest (Kleijnen & Wan, 2007) (a well-known and fully integrated optimization package of ARENA simulation framework (Kelton, Sadowski, & Sturrock, 2007)). Then, PSO should be adapted to solve a complex combinatorial problem subject to uncertain data. The proposed approach computes the fitness according to an average performance measure obtained over a set of simulation replications. In this case, PSO is used to optimize the utilization of piers and tanks in a complex petroleum terminal. Operational details and uncertain information are incorporated into optimization.

There are two main contributions of this paper. The first is that the PSO algorithm was modified to deal with combinatorial problems in scenarios of randomness whose results have only been recently reported in the literature (Mukhef, Farhan, & Jassim, 2008; Jiao, Chen, & Yan, 2011). These modifications are related to: (i) introduction of the path relinking technique where a particle

* Corresponding author. Tel.: +55 41 33104688; fax: +55 41 33104683.
*E-mail addresses:* marcella_engcomp@yahoo.com.br (M.S.R. Martins), stefan1234@gmail.com (S.C. Fuchs), lucianourgal@hotmail.com (L.U. Pando), luders@utfpr.edu.br (R. Lüders), myriamdelg@utfpr.edu.br (M.R. Delgado).

0360-8352/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cie.2013.02.004

encodes sets of same type resources of the application (previous approaches adopt this technique in other application contexts (Rosendo & Pozo, 2010; Souza et al., 2006)) and (ii) analysis of path relinking for PSO subject to uncertain information (path relinking has not been previously used in scenarios of randomness). The second contribution was in the modeling. This includes (i) formalizing the description of the purpose of the approach in a path relinking context, firstly by adapting the notation of combinatorial PSO approaches when they use particles with permutation-based encoding and secondly, by redefining the velocity equation, especially when the terms associated with inertial, cognitive and social components are presented and (ii) a real-world facility that incorporates uncertain data is modeled and operational details which are not considered in other mathematical optimization models are taken into account for optimization.

The paper is organized as follows. Section 2 presents details about PSO and its implementation. The addressed problems are described in Section 3. Section 4 presents our proposed simulation model and optimization approach. Results are shown in Section 5 followed by conclusions in Section 6.

## 2. Particle Swarm Optimization (PSO)

The particle swarm algorithm adjusts trajectories of a population of particles through a problem space using information about each particle's best performance and the best performance of its neighbors.

Consider a particle (candidate solution) positioned in the search space $\mathbb{R}^D$ at current discrete time $t$ represented by a vector $\mathbf{p}^t = (p_1^t, p_2^t, \cdots, p_D^t)$. The particle $\mathbf{p}^t$ is moved, by a velocity vector $\mathbf{v}^{t+1} = (v_1^{t+1}, \cdots, v_D^{t+1})$, to a new position $\mathbf{p}^{t+1}$ according to the following equation:

$$\mathbf{p}^{t+1} = \mathbf{p}^t + \mathbf{v}^{t+1}. \quad (1)$$

Usually, each element of the velocity vector is updated by inertial, cognitive and social components as defined by the following equation:

$$v_j^{t+1} = w \cdot v_j^t + (c_1 r_1) \cdot v_{j(cogn)}^t + (c_2 r_2) \cdot v_{j(social)}^t, \quad (2)$$

where $v_j^t$ is taken from the current velocity (or inertial component), $v_{j(cogn)}^t = \left(best_j - p_j^t\right)$ corresponds to the cognitive component with $best_j$ taken from the current best position achieved so far by the particle, and $v_{j(social)}^t = \left(best_{j(global)} - p_j^t\right)$ corresponds to the social component with $best_{j(global)}$ taken from the current best position achieved so far by the whole swarm. The coefficients $w$, $c_1$ and $c_2$ define how much a particle trust in its previous movement (inertial component), own history (cognitive component) and in the whole set of particles (social component), respectively (Engelbrecht, 2007). The terms $r_1$ and $r_2$ are random numbers drawn from an uniform distribution. The PSO algorithm initiates with a randomly generated swarm (position and velocity's particle) which is then updated for each iteration $t$ ($best_j$ is updated for each particle and $best_{j(global)}$ for the whole swarm). When a maximum of $T$ iterations is reached, the best solution is $\mathbf{best}_{global} = (best_{1(global)}, \ldots, best_{D(global)})$.

Compared to other Evolutionary Algorithms (EAs), PSO presents some advantages such as easy implementation and few parameters to adjust. PSO has been successfully applied in many areas: nonlinear optimization, artificial neural network training, fuzzy control, and other areas where EA can be applied. Mostly PSO gets better results with less computational effort when compared with other methods (Hu, 2011). Although PSO was first developed for continuous optimization problems, interesting results have also been obtained for combinatorial optimization. In most cases, the original equations are maintained.

The first version of PSO for discrete problems was presented in Kennedy and Eberhart (1997). In this work, Kennedy and Eberhart proposed an array of binary values to encode each particle. The velocity is defined as an array of probabilities of changing from 1 to 0 and vice versa. In other words, if $v_j^{t+1} = 0.20$, there is twenty percent of chance that $p_j^{t+1}$ will be one, and eighty percent of chance that it will be zero. If previous best positions have had a zero in the $j$th component (i.e. $best_j = 0$), then $\left(best_j - p_j^t\right)$ can be reasonably calculated as $-1$ or 0 and used to weight a change in the probability of $v_j^{t+1}$ for the next step. Eq. (2) remains unchanged, except that now each value considered in the $j$th component is an element of $\{0,1\}$ and $v_j^{t+1}$ must be constrained into the interval of probability $[0.0, 1.0]$. This way, the dimension of a particle movement can be seen as the number of bits changed from one iteration to another in a binary search space. A particle does not move if no bit is toggled, whereas it executes a full movement if all bits are toggled.

In Rosendo and Pozo (2010) and Souza et al. (2006), a technique named path relinking is used to move a particle toward another one. Path relinking is an intensification technique whose idea was originally proposed by Glover in the context of scheduling methods with better decision rules for job shop problems (Glover et al., 1999). Generally speaking, this strategy generates a path between two solutions. Assuming a source solution $\mathbf{p}^t$ and a destination solution $\mathbf{p}^{t+1}$, a path is a sequence of changes (steps) $\mathbf{p}^{t0}$, $\mathbf{p}^{t1}, \ldots, \mathbf{p}^{tm}$ with $\mathbf{p}^t = \mathbf{p}^{t0}$ and $\mathbf{p}^{tm} = \mathbf{p}^{t+1}$ such that $\mathbf{p}^{t(k+1)}$ is obtained from $\mathbf{p}^{tk}$ by using movements that reduce the distance between source and destination. This idea is used in some of the literature with discrete spaces to redefine velocity in terms of two particles $p_a$ and $p_b$ such that $v = p_a - p_b$ or $p_a = p_b + v$, i.e., by applying $v$ to $p_b$ results in $p_a$. This technique is used in our proposed optimization approach according to Rosendo and Pozo (2010), but using a particular notation suitable to represent permutations. The main contribution is a well-defined procedure for applying a list of permutations in the elements of a particle in terms of its inertial, cognitive and social components as described in the next section.

### 2.1. The proposed PSO-based approach

In our proposed approach each particle in the swarm is represented by a list of values indicating a resource allocation for the problem being optimized, and velocity is represented by a list of permutations of elements necessary to move $\mathbf{p}^t$ toward $\mathbf{p}^{t+1}$. The movement is based on the following equation:

$$\mathbf{p}^{t+1} = v^{t+1}(\mathbf{p}^t), \quad (3)$$

where the velocity $v^{t+1}$ is an operator represented by a list of pairs of indexes indicating which elements of $\mathbf{p}^t$ should be swapped. For example, assuming $\mathbf{p}^t = (11,22,33,44,55)$ and $v^{t+1} = \{(1,2), (2,3)\}$, then $\mathbf{p}^{t+1} = v^{t+1}((11,22,33,44,55)) = (22,33,11,44,55)$.

By considering the path relinking technique previously discussed, lets define how a particle $\mathbf{p}^t = (22,33,11,55,44)$ is moved to $\mathbf{p}^{t+1} = (11,22,33,44,55)$ using a list of three permutations. Consider $v^{t+1} = \left\{v_1^{t+1}, v_2^{t+1}, v_3^{t+1}\right\}$ and a path $\mathbf{p}^{t0}$, $\mathbf{p}^{t1}$, $\mathbf{p}^{t2}$, $\mathbf{p}^{t3}$ such that $\mathbf{p}^{t(k+1)} = v_{k+1}^{t+1}(\mathbf{p}^{tk})$, $k = 0, 1, 2$. As $p_1^{t+1} = p_3^t = 11$, $v_1^{t+1} = (1,3)$ and $\mathbf{p}^{t1} = (\mathbf{11}, 33, \mathbf{22}, 55, 44)$. Similarly, $v_2^{t+1} = (2,3)$, $\mathbf{p}^{t2} = (11, \mathbf{22}, \mathbf{33}, 55, 44)$, $v_3^{t+1} = (4,5)$ and $\mathbf{p}^{t3} = (11,22,33,\mathbf{44},\mathbf{55})$. Hence, $v^{t+1} = \{(1,3), (2,3), (4,5)\}$. Although using an integer encoding, a binary encoding is straightforward.

This procedure can be generalized by considering the inertial, cognitive and social components in the following order ($\lfloor \cdot \rfloor$ stands for floor and $|\cdot|$ for cardinality):

1. $u = \lfloor w \cdot |v^t| \rfloor$ permutations for the inertial component $v^t$;