Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 121782, 10 pages
http://dx.doi.org/10.1155/2014/121782

Research Article

Towards the Novel Reasoning among Particles in PSO by the Use of RDF and SPARQL

Iztok Fister Jr.,[1] Xin-She Yang,[2] Karin Ljubič,[3] Dušan Fister,[1] Janez Brest,[1] and Iztok Fister[1]

[1] *University of Maribor, Faculty of Electrical Engineering and Computer Science, Smetanova 17, 2000 Maribor, Slovenia*
[2] *Middlesex University Hendon Campus, London NW4 4BT, UK*
[3] *University of Maribor Faculty of Medicine, Taborska 8, 2000 Maribor, Slovenia*

Correspondence should be addressed to Iztok Fister Jr.; iztok.fister2@uni-mb.si

Received 15 January 2014; Accepted 26 February 2014; Published 27 March 2014

Academic Editors: N. Chakraborti, P. Melin, and F. Neri

The significant development of the Internet has posed some new challenges and many new programming tools have been developed to address such challenges. Today, semantic web is a modern paradigm for representing and accessing knowledge data on the Internet. This paper tries to use the semantic tools such as resource definition framework (RDF) and RDF query language (SPARQL) for the optimization purpose. These tools are combined with particle swarm optimization (PSO) and the selection of the best solutions depends on its fitness. Instead of the local best solution, a neighborhood of solutions for each particle can be defined and used for the calculation of the new position, based on the key ideas from semantic web domain. The preliminary results by optimizing ten benchmark functions showed the promising results and thus this method should be investigated further.

1. Introduction

Searching for the optimal solutions of the hardest real-world problems is an active field especially in computer science. An eternal desire of computer scientists is to develop a general problem solver that will be able to cope with all classes of real-world problems. Unfortunately, the most of the so-called clever algorithms are subject of the No Free Lunch Theorem [1]. Regarding this theorem, if one algorithm is good on one class of problems, it does not mean that it will also be good on the other classes of problems. Especially, three domains of algorithms have recently been appeared in the role of general problem solver, as follows: Artificial Intelligence (AI) [2], evolutionary algorithms (EA) [3], and Swarm Intelligence (SI) [4]. While the former mimics operating a human brain, the latter domains are inspired by nature. Evolutionary algorithms are inspired by Darwinian principles of natural evolution [5] according to which the fittest individuals have the greater possibilities for survival and pass on their characteristics to their offspring during a process of reproduction.

Nowadays, evolutionary computation (AC) [6] captures the algorithms involved in evolutionary domain and it considers genetic algorithms (GA) [7], genetic programming [8], evolution strategies (ES) [9], evolutionary programming [10], and differential evolution (DE) [11–13]. The mentioned algorithms differ between each other according to representation of individual. As a result, these kinds of algorithms have been applied to various optimization, modeling, and simulation problems.

However, this paper concentrates on the SI domain that is concerned with the design of multiagent systems with applications, for example, in optimization and in robotics [4]. Inspiration for the design of these systems is taken from the collective behavior of social insects, like ants, termites, and bees, as well as from the behavior of other animal societies, like flocks of birds or schools of fish. Recently, there exist a lot of different algorithms from this domain that is still being developed. Let us mention only the most important members of the SI algorithms, as follows: the particle swarm optimization (PSO) [14], the firefly algorithm (FA) [15, 16], cuckoo search [17], the bat algorithm (BA) [18, 19], and so forth.

The PSO is population-based algorithm that mimics movement of the swarm of particles (e.g., birds) by flying across a landscape, thus searching for food. Each particle in PSO represents the candidate solution of the problem to be solved. Position of the particle consists of the problem parameters that are modified when the virtual particle is moved in the search space. The motion depends on the current particle position and the current position of local best and global best solutions, respectively. The local best solutions denote the best solutions that are whenever arisen on the definite location in the population, while the current best is the best solution whenever found in the whole population. This solution has the main impact on the direction of moving the swarm towards the optimal solution. When this solution is not improved anymore, the population gets stuck into a local optimum.

Mainly, we focused on the new definition of the neighborhood within the PSO algorithm. In place of the local best solutions, the neighborhood is defined using the predefined radius of fitness values around each candidate solution, thus capturing all candidate solutions with the fitness value inside the predefined virtual radius. The size of this neighborhood can be variable. Therefore, at least one but maximum three candidate solutions can be permitted to form this neighborhood. Although this is not the first try how to define the variable neighborhood within the PSO algorithm [20–22], in this paper, such neighborhood is defined using the Resource Description Framework (RDF), SPARQL Protocol, and RDF query language (SPARQL) tools taken from semantic web domain. As a result, the modified RDF-PSO algorithm was developed. Both web tools are appropriate for describing and manipulating decentralized and distributed data. On the other hand, the original PSO algorithm maintains a population of particles that are also decentralized in their nature. An aim of using these web tools was to simulate a distributed population of particles, where each particle is placed on the different location in the Internet.

The remainder of this paper is structured as follows. In Section 2, we outline a short description of the PSO algorithm. The section is finished by introducing the semantic web tools, that is, RDF and SPARQL. Section 3 concentrates on development of the modified RDF-PSO algorithm. Section 4 presents the conducted experiments and results obtained with the RDF-PSO algorithms. Finally, Section 5 summarizes our work and potential future directions for the future work are outlined.

2. Background

2.1. Particle Swarm Optimization. Particle swarm optimization (PSO) was one of the first SI algorithms to be presented at an International Conference on Neural Networks by Kennedy and Eberhart in 1995 [14]. PSO is inspired by the social foraging behavior of some animals such as flocking behavior of birds (Figure 1) and schooling behavior of fish [23]. In nature, there are some individuals with better developed instinct for finding food. According to these individuals, the whole swarm is directed into more promising regions in the landscape.

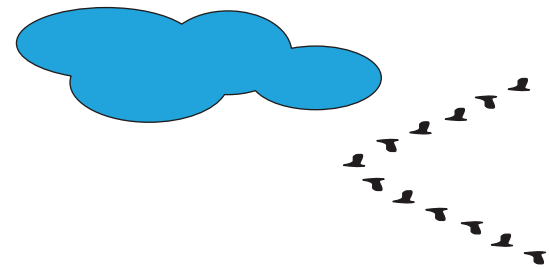

Figure 1: PSO.

The PSO is a population-based algorithm that consists of N particles $\mathbf{x}_i^{(t)} = (x_{i1}, \ldots, x_{iD})^T$ representing their position in a D-dimensional search space. These particles move across this space with velocity $\mathbf{v}_i^{(t)} = (v_{i1}, \ldots, v_{iD})^T$ according to the position of the best particle $\mathbf{x}_{\text{best}}^{(t)}$ towards the more promising regions of the search space. However, this movement is also dependent on the local best position of each particle $\mathbf{p}_i^{(t)}$ and is mathematically expressed, as follows:

$$\begin{aligned}\mathbf{v}_i^{(t+1)} = \mathbf{v}_i^{(t)} &+ c_1 r_1^{(t)} \left(\mathbf{p}_i^{(t)} - \mathbf{x}_i^{(t)}\right) \\ &+ c_2 r_2^{(t)} \left(\mathbf{x}_{\text{best}}^{(t)} - \mathbf{x}_i^{(t)}\right), \quad \text{for } i = 1, \ldots, N,\end{aligned} \tag{1}$$

where $r_1^{(t)}$, $r_2^{(t)}$ denote the random numbers drawn from the interval $[0, 1]$, and c_1, c_2 are constriction coefficients that determine the proportion, with which the local and global best solutions influence the current solution. Then, the new particle position is calculated according to the following expression:

$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)} + \mathbf{v}_i^{(t)}, \quad \text{for } i = 1, \ldots, N. \tag{2}$$

Pseudocode of the PSO algorithm is illustrated in Algorithm 1.

After finishing the initialization in function "init_particles" (Algorithm 1), the PSO algorithm optimizes a problem by iteratively improving the candidate solution [24, 25]. Thus, two functions are applied. The function "evaluate_the_new_solution" calculates the fitness value of particles obtained after initialization or movement. The movement according to (1) and (2) is implemented in the function "generate_new_solution."

2.2. RDF. The RDF is an XML application devoted to encoding, exchanging, and reusing structural metadata [26]. It enables the knowledge to be represented in symbolical form. Fortunately, this knowledge is human readable. On the other hand, it is understandable to machines. The main characteristic of this framework is that RDF data can be manipulated on decentralized manner and distributed among various servers on the Internet. Resources identified by Uniform Resource Identifier (URI) are described in RDF graphs, where each resource representing the node has many properties that are associated with the resource using the

Input: PSO population of particles $\mathbf{x_i} = (x_{i1}, \ldots, x_{iD})^T$ for $i = 1, \ldots, N$.
Output: The best solution $\mathbf{x}_{\text{best}}$ and its corresponding value $f_{\min} = \min(f(\mathbf{x}))$.

```
(1)  init_particles;
(2)  eval = 0;
(3)  while termination_condition_not_meet do
(4)     for i = 1 to N do
(5)        f_i = evaluate_the_new_solution(x_i);
(6)        eval = eval + 1;
(7)        if f_i ≤ pBest_i then
(8)           p_i = x_i;   pBest_i = f_i; // save the local best solution
(9)        end if
(10)       if f_i ≤ f_min then
(11)          x_best = x_i;   f_min = f_i; // save the global best solution
(12)       end if
(13)       x_i = generate_new_solution(x_i);
(14)    end for
(15) end while
```

Algorithm 1: Pseudocode of the classic PSO algorithm.

```
(1) <rdf:RDF>
(2) <rdf:Description rdf:about="http://www.example.org/Person">
(3) <ns1:Name>John</ns1:Name>
(4) <ns1:Surname>Smith</ns1:Surname>
(5) </rdf:Description>
(6) </rdf:RDF>
```

Algorithm 2: Pseudocode of the Person description in RDF.

property-type relationship. This relationship represents an edge in RDF graph. Thus, attributes may be atomic in nature (e.g., numbers, text strings, etc.) or represent other resources with their own properties [27].

The resource, property-type relation, and attribute present a triplet suitable for presentation in RDF graph. A sample of this graph is presented in Figure 2, from which two triples (also 2-triples) can be translated from the diagram. These 2-triples are written into a RDF database. In general, the format of this RDF data is serialization of N-triples obtained from the RDF graphs. For instance, the description of RDF database obtained from the RDF graph in Figure 2 is presented in Algorithm 2.

2.3. SPARQL. RDF enables data to be decentralized and distributed across the Internet. On the other hand, the SPARQL Protocol has been developed for accessing and discovering RDF data. SPARQL is an RDF query language that has its own syntax very similar to SQL queries. The SPARQL query consists of two parts [28]. The former SELECT clause identifies the variables that appear in the query results, while the latter WHERE clause provides the basic patterns that match against the RDF graph. Usually, these query patterns consist of three parts denoting the resource name, property-type relation, and attribute. As a result, the matched patterns are returned by the query. A sample of SPARQL query is illustrated in Algorithm 3.

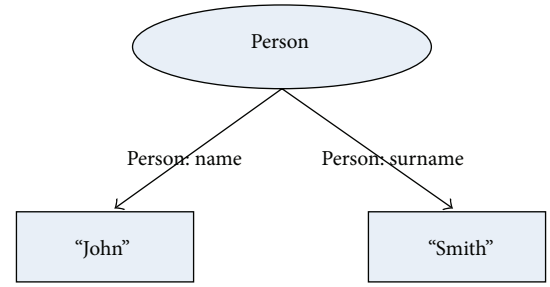

Figure 2: Diagram illustrates a resource Person in RDF graph. The resource consists of two atomic attributes "John" and "Smith" that are assigned to it with relations "PERSON: Name" and "PERSON: Surname". In other words, the name of the person is "John" and the surname of the same person is "Smith." Note that the word "PERSON:" denotes a location (URI), where a name space containing the description of semantics for this relation is located.

As a result of query presented in Algorithm 3, the name "John" and surname "Smith" are returned.

3. The Modified RDF-PSO Algorithm

The modified RDF-PSO algorithm implements two features:

(i) using the variable neighborhood of candidate solutions in place of the local best solutions,

```
(1) SELECT ?name ?surname
(2) WHERE {
(3)     <http://www.example.org/Person> PERSON:Name ?name.
(4)     <http://www.example.org/Person> PERSON:Surname ?surname.
(5) }
```

Algorithm 3: Example of SPARQL query.

Input: PSO population of particles $\mathbf{x_i} = (x_{i1}, \ldots, x_{iD})^T$ for $i = 1, \ldots, N$.
Output: The best solution $\mathbf{x}_{\text{best}}$ and its corresponding value $f_{\min} = \min(f(\mathbf{x}))$.

```
(1)  init_particles;
(2)  eval = 0;
(3)  while termination_condition_not_meet do
(4)     for i = 1 to N do
(5)        f_i = evaluate_the_new_solution(x_i);
(6)        eval = eval + 1;
(7)        if f_i ≤ f_min then
(8)           x_best = x_i;   f_min = f_i; // save the global best solution
(9)        end if
(10)       N(x_i) = {p_j | p_j is_neighbor_of x_i};
(11)       x_i = generate_new_solution(x_i, N(x_i));
(12)    end for
(13) end while
```

Algorithm 4: The proposed RDF-PSO algorithm.

(ii) using the RDF for describing and SPARQL for manipulating this neighborhood.

The main reason for applying these well-known tools from the semantic web domain was to develop a distributed population model that could later be used in other SI algorithms. On the other hand, we try to use the semantic web tools for optimization purposes as well. Fortunately, RDF is suitable tool for describing the distributed population models, in general. In the PSO algorithm, it is applied for describing the relations between particles in population. For our purposes, a relation "is_neigbour_of" is important that each particle determines its neighborhood. Furthermore, SPARQL is used for determining the particles in its neighborhood. As a result, the RDF-PSO algorithm has been established, whose pseudocode is presented in Algorithm 4.

The three main differences distinguish the proposed RDF-PSO with the original PSO algorithm, as follows:

(i) no local best solutions that are maintained by the RDF-PSO (lines 10–12 omitted in the Algorithm 1),

(ii) defining the neighborhood of candidate solution (line 10 in Algorithm 4),

(iii) generating the new solution according to the defined variable neighborhood relation (line 11 in Algorithm 4).

The relation $\mathcal{N}(\mathbf{x}_i) = \{\mathbf{x}_j \mid \mathbf{x}_j \text{ is\_neighbor\_of } \mathbf{x}_j\}$ (line 10 in Algorithm 4) is defined according to the following relation:

$$\textbf{if} \quad \text{abs}\left(f\left(x_j\right) - f\left(x_i\right)\right) \leq R \quad \textbf{then } \mathbf{x}_j \in \mathcal{N}\left(\mathbf{x}_i\right), \tag{3}$$

where radius R defines the necessary maximum fitness distance of two candidate solutions that can be in neighborhood. In fact, this parameter regulates the number of candidate solutions in the neighborhood.

Here, the radius is expressed as $R = \sum_{i=1}^{N} |f(\mathbf{x}_i)|/\sqrt{N}$. Indeed, the neighborhood captures all solutions with the fitness differences less than the radius R. Typically, when the radius R is small, the size of neighborhood can also be small. However, this assertion holds if the population diversity is higher enough. When the particles are scattered across the search space, no particles are located in the vicinity of each other. Consequently, the size of neighborhood becomes zero. On the other hand, when the particles are crowded around some fitter individuals, the number of its neighbors can be increased enormously. In order to prevent this undersizing and oversizing, the neighborhood size is defined in such a manner that it cannot exceed the value of three and cannot be zero; in other words, $|\mathcal{N}(\mathbf{x}_i) \in [1, 3]|$.

For each observed particle $\mathbf{x}_i$, the new solution is generated according to the number of neighbors $|\mathcal{N}(\mathbf{x}_i)|$ in "generate_new_solution" function. The following modified equation is used in RDF-PSO for calculating the velocity:

$$\mathbf{v}_i^{(t+1)} = w \cdot \mathbf{v}_i^{(t)} + c_1 r_1^{(t)} \left(\mathbf{x}_{\text{best}}^{(t)} - \mathbf{x}_i^{(t)}\right) + \left[\frac{\sum_{j=1}^{|\mathcal{N}(\mathbf{x}_j)|} c_{j+1} r_{j+1}^{(t)} \left(\mathbf{p}_j^{(t)} - \mathbf{x}_i^{(t)}\right)}{\left|\mathcal{N}\left(\mathbf{x}_i\right)\right|}\right], \tag{4}$$

where, r_1, and r_2 are the real numbers randomly drawn from the interval $[0, 1]$, c_1 and c_2 denote the constriction

```
(1)  <rdf:RDF>
(2)     <rdf:Description rdf:about="http://www.example.org/population">
(3)        <ns1:member_of rdf:resource="http://www.example.org/particle1"/>
(4)        <ns1:member_of rdf:resource="http://www.example.org/particle2"/>
(5)        ...
(6)        <ns1:member_of rdf:resource="http://www.example.org/particlen"/>
(7)     </rdf:Description>
(8)     <rdf:Description rdf:about="http://www.example.org/particle1">
(9)        <ns1:is_neighbor_of rdf:HREF="http://www.example.org/particle2"/>
(10)       <ns1:is_neighbor_of rdf:HREF="http://www.example.org/particle5"/>
(11)       <ns1:id>1</ns1:id>
(12)    </rdf:Description>
(13)    ...
(14) </rdf:RDF>
```

Algorithm 5: Pseudocode of the PSO population in RDF.

coefficients, $\mathbf{p}_j^{(t)} = \{\mathbf{x}_k \mid \mathbf{x}_k \text{ is\_neighbor\_of } \mathbf{x}_i \wedge 1 \leq k \leq N\}$, $j \in [1, |\mathcal{N}(\mathbf{x}_i)|]$, and $\sum_{j=1}^{|\mathcal{N}(\mathbf{x}_i)|} c_{j+1} = 1$. Thus, it is expected that the movement of more crowded neighborhood depends on more neighbors. Furthermore, the term between square parenthesis ensures that the proportion of each neighbor as determined by constriction coefficients $\{c_2, c_3, c_4\}$ never exceeded the value of one.

3.1. Representation of a Distributed Population. The rapid growth of the Internet means that new kinds of application architectures have been emerged. The Internet applications are suitable to exploit enormous power of the computers connected to this huge network. Typically, these applications search for data distributed on many servers. These data need to be accessed easily, securely, and efficiently.

This paper proposes the first steps of developing the distributed population model within the PSO algorithm. In line with this, the RDF tool is applied that introduces a description of relations between particles in the population. These relations make us possible to manipulate population members on a higher abstraction level. At the moment, only the relation "is_neighbor_of" is implemented that determines the neighborhood of a specific particle in the population.

For this purpose, RDF is devoted for defining the various resources on different Internet servers. In our case, each particle in the population represents the resource that is defined with corresponding property-type relation (e.g., "is_neighbor_of") and attributes. The RDF graph of the distributed population is illustrated in Figure 3.

The definition of a distributed population in RDF is presented in Algorithm 5, from which it can be seen that two kinds of attributes are encountered in this definition, that is, the references to neighbors of specific particle and its sequence number. Some details are omitted in this algorithm because of the space limitation of this paper. The missing parts of code are denoted by punctuation marks.

3.2. Accessing the Distributed Population. The distributed population in RDF can be accessed using the SPARQL query language, whose syntax is similar to the standard SQL syntax. An example of SPARQL query for returning the neighborhood of fourth particle is represented in Algorithm 6. Note that the SPARQL query from the mentioned algorithm will return all attributes that are related to the "resource4" with the relation "is_neighbor_of."

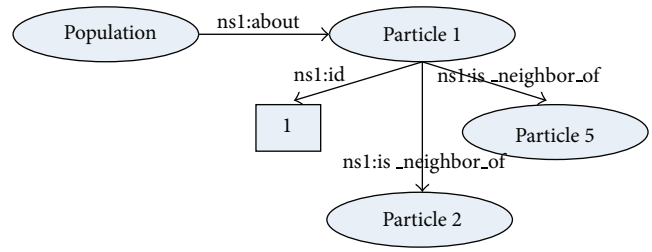

Figure 3: This PSO distributed population model contains the definitions of resources *population* and *particle*. Thus, each *particle* can relate to one or more neighbors and has an identification number. The former represents a reference to other *particles* in a swarm, while the latter is an atomic value.

```
(1) SELECT ?particle
(2) WHERE {
(3) <http://www.example.org/particle4>
(4) <http://www.example.org/is_neighbor_of>
(5) ?particle
(6) }
```

Algorithm 6: SPAQL query that returns particles in the neighborhood of particle4.

3.3. Implementation Details. The proposed RDF-PSO algorithm was implemented in Python programming language

Table 1: Definitions of benchmark functions.

f	Function name	Definition
f_1	Griewangk's function	$f_1(\mathbf{x}) = -\prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + \sum_{i=1}^{n} \frac{x_i^2}{4000} + 1$
f_2	Rastrigin's function	$f_2(\mathbf{x}) = n * 10 + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i)\right)$
f_3	Rosenbrock's function	$f_3(\mathbf{x}) = \sum_{i=1}^{n-1} 100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2$
f_4	Ackley's function	$f_4(\mathbf{x}) = \sum_{i=1}^{n-1} \left(20 + e^{-20} e^{-0.2\sqrt{0.5(x_{i+1}^2 + x_i^2)}} - e^{0.5(\cos(2\pi x_{i+1}) + \cos(2\pi x_i))}\right)$
f_5	Schwefel's function	$f_5(\mathbf{x}) = 418.9829 * D - \sum_{i=1}^{D} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$
f_6	De Jong's sphere function	$f_6(\mathbf{x}) = \sum_{i=1}^{D} x_i^2$
f_7	Easom's function	$f_7(\mathbf{x}) = -(-1)^D \left(\prod_{i=1}^{D} \cos^2(x_i)\right) \exp\left[-\sum_{i=1}^{D} (x_i - \pi)^2\right]$
f_8	Michalewicz's function	$f_8(\mathbf{x}) = -\sum_{i=1}^{D} \sin(x_i) \left[\sin\left(\frac{iA \cdot x_i^2}{\pi}\right)\right]^{2 \cdot 10}$
f_9	Xin-She Yang's function	$f_9(\mathbf{x}) = \left(\sum_{i=1}^{D} \lvert x_i \rvert\right) \exp\left[-\sum_{i=1}^{D} \sin\left(x_i^2\right)\right]$
f_{10}	Zakharov's function	$f_{10}(\mathbf{x}) = \sum_{i=1}^{D} x_i^2 + \left(\frac{1}{2}\sum_{i=1}^{D} i x_i\right)^2 + \left(\frac{1}{2}\sum_{i=1}^{D} i x_i\right)^4$

and executed on Linux operating system. Additionally, the following libraries were used:

(i) *rdflib* which is a python library for working with RDF [29],

(ii) *NumPy* that is the fundamental package for scientific computing with Python [30]

(iii) *matplotlib* that is a python 2D plotting library [31].

The decision for using Python has been taken because there already existed a lot of the PSO implementation. Furthermore, the RDF and SPARQL semantic tools are also supported in this language and ultimately, programming in Python is easy.

4. Experiments and Results

The goal of our experimental work was to show that the semantic web tools, that is, RDF and SPARQL can be useful for the optimization purposes as well. Moreover, we want to show that using the variable neighborhood in RDF-PSO can also improve the results of the original PSO.

In line with this, the RDF-PSO algorithm was applied to the optimization of ten benchmark functions taken from literature. The function optimization belongs to a class of continuous optimization problems, where the objective function $f(\mathbf{x})$ is given and $\mathbf{x} = \{x_1, \ldots, x_D\}$ is a vector of D design variables in a decision space S. Each design variable $x_i \in [Lb_i, Ub_i]$ is limited by its lower $Lb_i \in \mathbb{R}$ and upper $Ub_i \in \mathbb{R}$ bounds. The task of optimization is to find the minimum of the objective functions.

In the remainder of this section, the benchmark suite is described; then, the experimental setup is presented and finally, the results of experiments are illustrated in detail.

4.1. Test Suite. The test suite consisted of ten functions, which were selected from the literature. However, the primary reference is the paper by Yang [32] that proposed a set of optimization functions suitable for testing the newly developed algorithms. The definitions of the benchmark functions are represented in Table 1, while their properties are illustrated in Table 2.

Table 2 consists of five columns that contain the function identifications (tag f), their global optimum (tag f^*), the values of optimal design variables (tag x^*), the lower and upper bounds of the design variables (tag *Bound*), and their characteristics (tag *Characteristics*). The lower and upper bounds of the design variables determine intervals that limit the size of the search space. The wider is the interval, the wider is the search space. Note that the intervals were selected, so that the search space was wider than those proposed in the standard literature. The functions within the benchmark suite can be divided into *unimodal* and *multimodal*. The multimodal functions have two or more local optima. Typically, the multimodal functions are more difficult to solve. The most complex functions are those that have an exponential number of local optima randomly distributed within the search space.

Table 2: Properties of benchmark functions.

f	f^*	x^*	Bounds	Characteristics
f_1	0.0000	$(0, 0, \ldots, 0)$	$[-600, 600]$	Highly multi-modal
f_2	0.0000	$(0, 0, \ldots, 0)$	$[-15, 15]$	Highly multi-modal
f_3	0.0000	$(1, 1, \ldots, 1)$	$[-15, 15]$	Multiple local optima
f_4	0.0000	$(0, 0, \ldots, 0)$	$[-32.768, 32.768]$	Highly multi-modal
f_5	0.0000	$(0, 0, \ldots, 0)$	$[-500, 500]$	Highly multi-modal
f_6	0.0000	$(0, 0, \ldots, 0)$	$[-600, 600]$	Uni-modal, convex
f_7	−1.0000	$(\pi, \pi, \ldots, \pi)$	$[-2\pi, 2\pi]$	Multiple local optima
f_8	−1.8013[1]	$(2.20319, 1.57049)$[1]	$[0, \pi]$	Multiple local optima
f_9	0.0000	$(0, 0, \ldots, 0)$	$[-2\pi, 2\pi]$	Multiple local optima
f_{10}	0.0000	$(0, 0, \ldots, 0)$	$[-5, 10]$	Uni-modal

[1]These values are valid for dimensions $D = 2$.

4.2. Experimental Setup. This experimental study compares the results of the RDF-PSO using different kind of distributed populations within the original PSO algorithm. All PSO algorithms used the following setup. The parameter w was randomly drawn from the interval $[0.4, 0.9]$, while the constriction coefficients were set as $c_1 = c_2 = 1.0$. As a termination condition, the number of fitness function evaluations was considered. It was set to FEs $= 1000 \cdot D$, where D denotes dimension of the problem. In this study, three different dimensions of functions were applied; that is, $D = 10$, $D = 30$, and $D = 50$. However, the population size is a crucial parameter for all population-based algorithms that have a great influence on their performance. In line with this, extensive experiments had been run in order to determine the most appropriate setting of this parameter by all algorithms in the test. As a result, the most appropriate setting of this parameter $N = 100$ was considered for the experiments. Parameters, like the termination condition, dimensions of the observed functions, and the population size were also used by the other algorithms in experiments.

The PSO algorithms are stochastic in nature. Therefore, statistical measures, like minimum, maximum, average, standard deviation, and median, were accumulated after 25 runs of the algorithms in order to fairly estimate the quality of solutions.

4.3. Results. The comparative study was conducted in which we would like to show, firstly, that the semantic web tools can be successfully applied to the optimization purposes as well and, secondly, that using the distributed population affects the results of the original PSO algorithm. In the remainder of this section, a detailed analysis of RDF-PSO algorithms is presented.

4.3.1. Analysis of the RDF-PSO Algorithms. In this experiment, the characteristics of the RDF-PSO algorithm were analyzed. In line with this, the RDF-PSO with neighborhood size of one (RDF1), the RDF-PSO with neighborhood size of two (RDF2), and the RDF-PSO with neighborhood size of tree (RDF3) were compared with the original PSO algorithm (PSO) by optimizing ten benchmark functions with dimensions $D = 10$, $D = 30$, and $D = 50$. The obtained results by the optimization of functions with dimension $D = 30$ are aggregated in Table 3. Note that the best average values are for each function presented bold in the table.

From Table 3, it can be seen that the best average values were obtained by the RDF-1 algorithm eight times, that is, by $f_1 - f_4$, f_6, f_8, and f_{10}. The best results were two times observed also by the original PSO algorithm, that is, f_5 and f_9. On average, the results of the other two RDF-PSO algorithms, that are, RDF-2 and RDF-3, were better than the results of the original PSO algorithm.

In order to statistically estimate the quality of solution, the Friedman nonparametric test was conducted. Each algorithm enters this test with five statistical measures for each of observed functions. As a result, each statistical classifier (i.e., various algorithms) consists of $5 \cdot 10 = 50$ different variables. The Friedman test [33, 34] compares the average ranks of the algorithms. The closer the rank to one, the better is the algorithm in this application. A null hypothesis states that two algorithms are equivalent and, therefore, their ranks should be equal. If the null hypothesis is rejected, that is, the performance of the algorithms is statistically different, the Bonferroni-Dunn test [35] is performed that calculates the critical difference between the average ranks of those two algorithms. When the statistical difference is higher than the critical difference, the algorithms are significantly different. The equation for the calculation of critical difference can be found in [35].

Friedman tests were performed using the significance level 0.05. The results of the Friedman nonparametric test are presented in Figure 4 where the three diagrams show the ranks and confidence intervals (critical differences) for the algorithms under consideration. The diagrams are organized according to the dimensions of functions. Two algorithms are significantly different if their intervals do not overlap.

The first diagram in Figure 4 shows that the RDF-1 algorithm significantly outperforms the RDF-3 algorithm. Interestingly, the results of the original PSO are also better than the results of the RDF-2 and RDF-3 algorithm. The situation is changed in the second (by $D = 30$) and third diagram (by $D = 50$), where RDF-3 improves the results

Table 3: Comparing the results of different PSO algorithms ($D = 30$).

Alg.	Meas.	f_1	f_2	f_3	f_4	f_5
PSO	Best	$7.40E-001$	$5.83E+002$	$7.08E+004$	$2.00E+001$	$1.55E+003$
	Worst	$1.41E+000$	$5.65E+003$	$1.10E+007$	$2.06E+001$	$9.77E+003$
	Mean	$1.06E+000$	$1.44E+003$	$3.22E+006$	$2.03E+001$	**$5.16E+003$**
	StDev	$1.06E+000$	$1.13E+003$	$2.04E+006$	$2.04E+001$	$7.44E+003$
	Mean	$1.37E-001$	$1.02E+003$	$3.14E+006$	$1.90E-001$	$5.97E+003$
RDF-1	Best	$8.87E-014$	$4.55E-010$	$2.85E+001$	$5.31E-005$	$8.77E+003$
	Worst	$3.33E-010$	$1.95E-006$	$2.88E+001$	$9.64E-003$	$1.05E+004$
	Mean	**$3.40E-011$**	**$2.18E-007$**	**$2.87E+001$**	**$2.61E-003$**	$9.88E+003$
	StDev	$1.27E-011$	$4.81E-008$	$2.87E+001$	$1.53E-003$	$9.95E+003$
	Median	$6.91E-011$	$4.90E-007$	$5.85E-002$	$2.68E-003$	$3.99E+002$
RDF-2	Best	$1.10E-005$	$3.37E+000$	$3.95E+001$	$2.78E-001$	$8.77E+003$
	Worst	$2.42E-001$	$7.03E+001$	$3.42E+002$	$3.64E+000$	$1.04E+004$
	Mean	$6.03E-002$	$3.08E+001$	$1.64E+002$	$1.95E+000$	$9.73E+003$
	StDev	$4.12E-002$	$2.52E+001$	$1.36E+002$	$1.72E+000$	$9.65E+003$
	Mean	$5.80E-002$	$2.12E+001$	$9.62E+001$	$1.04E+000$	$4.43E+002$
RDF-3	Best	$3.07E-005$	$1.85E+001$	$5.49E+001$	$9.11E-001$	$8.81E+003$
	Worst	$2.52E-001$	$1.56E+002$	$4.03E+002$	$4.56E+000$	$1.03E+004$
	Mean	$8.94E-002$	$7.75E+001$	$1.77E+002$	$2.63E+000$	$9.70E+003$
	StDev	$7.62E-002$	$7.60E+001$	$1.64E+002$	$2.66E+000$	$9.78E+003$
	Mean	$5.51E-002$	$3.46E+001$	$8.38E+001$	$1.11E+000$	$4.19E+002$
Evals	Meas.	f_6	f_7	f_8	f_9	f_{10}
PSO	Best	$1.10E-003$	$0.00E+000$	$-1.77E+001$	$6.94E-012$	$5.39E-001$
	Worst	$3.75E-001$	$0.00E+000$	$-8.45E+000$	$5.38E-011$	$2.42E+001$
	Mean	$1.13E-001$	$0.00E+000$	$-1.42E+001$	**$1.30E-011$**	$5.26E+000$
	StDev	$6.89E-002$	$0.00E+000$	$-1.41E+001$	$8.60E-012$	$3.36E+000$
	Mean	$1.24E-001$	$0.00E+000$	$1.87E+000$	$1.05E-011$	$5.35E+000$
RDF-1	Best	$7.80E-014$	$0.00E+000$	$-6.45E+000$	$1.83E-007$	$2.47E-012$
	Worst	$3.55E-009$	$0.00E+000$	$-3.85E+000$	$1.39E-005$	$5.70E-007$
	Mean	**$3.83E-010$**	$0.00E+000$	**$-4.81E+000$**	$3.68E-006$	**$2.59E-008$**
	StDev	$2.97E-011$	$0.00E+000$	$-4.74E+000$	$3.35E-006$	$1.34E-009$
	Mean	$8.15E-010$	$0.00E+000$	$6.12E-001$	$3.29E-006$	$1.13E-007$
RDF-2	Best	$6.63E-004$	$0.00E+000$	$-6.47E+000$	$3.57E-009$	$2.55E-001$
	Worst	$2.03E+000$	$0.00E+000$	$-4.06E+000$	$1.73E-007$	$1.99E+002$
	Mean	$4.36E-001$	$0.00E+000$	$-4.93E+000$	$6.03E-008$	$2.54E+001$
	StDev	$1.48E-001$	$0.00E+000$	$-4.81E+000$	$3.83E-008$	$5.92E+000$
	Mean	$5.58E-001$	$0.00E+000$	$5.92E-001$	$4.64E-008$	$4.90E+001$
RDF-3	Best	$5.21E-003$	$0.00E+000$	$-6.27E+000$	$2.42E-009$	$5.94E+000$
	Worst	$2.09E+000$	$0.00E+000$	$-4.09E+000$	$1.31E-007$	$4.53E+002$
	Mean	$8.47E-001$	$0.00E+000$	$-5.03E+000$	$3.82E-008$	$1.09E+002$
	StDev	$8.14E-001$	$0.00E+000$	$-5.05E+000$	$3.35E-008$	$6.65E+001$
	Mean	$5.73E-001$	$0.00E+000$	$5.92E-001$	$3.08E-008$	$1.15E+002$

of the RDF-3 and the original PSO, but not the RDF-2 algorithm. Additionally, the RDF-2 is significantly better than the original PSO also by $D = 50$.

In summary, the RDF-1 exposes the best results between all the other algorithms in tests by all observed dimensions of functions. On the other hand, the original PSO algorithm is only comparable with the modified PSO algorithms by optimizing the low dimensional functions ($D = 10$). The question why the RDF-PSO with neighborhood size of one outperformed the other RDF-PSO algorithms remains open for the future work. At this moment, it seems that here the primary role plays the constriction coefficients that determine an influence of specific neighbors.

5. Conclusion

The aim of this paper was twofold. First is to prove that the semantic web tools, like RDF and SPARQL, can also

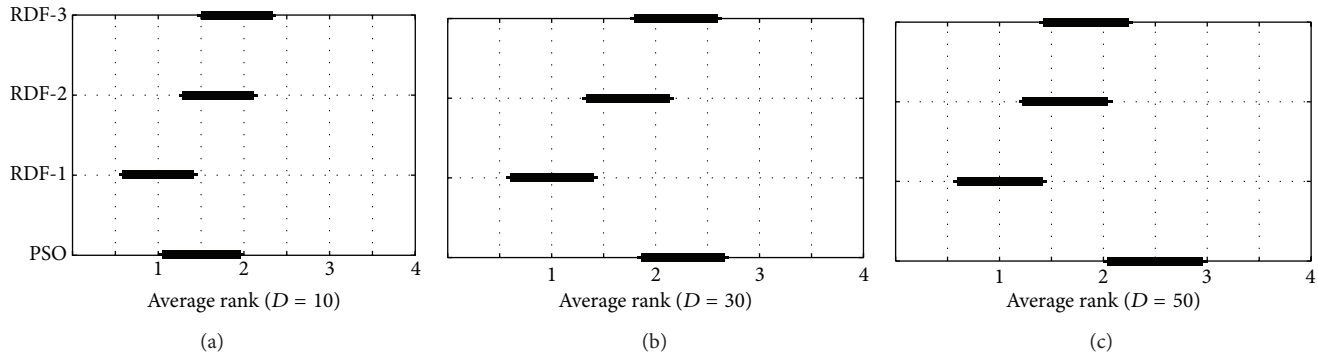

Figure 4: Results of the Friedman nonparametric test.

be used for the optimization purposes. Second is to show that the results of the modified RDF-PSO using the variable neighborhood are comparable with the results of the original PSO algorithm.

In line with the first hypothesis, a distributed population model was developed within the PSO algorithm that is suitable for describing the variable neighborhood of particles in the population. Furthermore, moving particles across the search space depends on all the particles in the neighborhood in place of the local best solutions as proposed in the original PSO algorithm.

In order to confirm the second hypothesis, the benchmark suite of ten well-known functions from the literature was defined. The results of extensive experiments by optimization of benchmark functions showed that the optimal neighborhood size within the RDF-PSO algorithm is one (RDF1). This variant of the RDF-PSO also outperformed the original PSO algorithm.

The distributed population model extends the concept of population in SI. This means that the population is no longer a passive data structure for storing particles. Not only can the particles now be distributed, but also some relations can be placed between the population members. In this proof of concept, only one relation was defined, that is, "is_neighbor_of." Additionally, not the whole definition of the distributed population was put onto Internet at this moment. Although we are at the beginning of the path of how to make an intelligent particle in swarm intelligence algorithms, the preliminary results are encouraging and future researches would investigate this idea of distributed population models in greater detail.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[2] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, New York, NY, USA, 2009.

[3] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Springer, Berlin, Germany, 2003.

[4] C. Blum and D. Merkle, *Swarm Intelligence: Introduction and Applications*, Springer, Berlin, Germany, 2008.

[5] C. Darwin, *The Origin of Species*, John Murray, London, UK, 1859.

[6] W. Paszkowicz, "Genetic algorithms, a nature-inspired tool: survey of applications in materials science and related fields," *Materials and Manufacturing Processes*, vol. 24, no. 2, pp. 174–197, 2009.

[7] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Massachusetts, Mass, USA, 1996.

[8] J. Koza, *Genetic Programming 2—Automatic Discovery of Reusable Programs*, The MIT Press, Cambridge, Mass, USA, 1994.

[9] T. Bck, *Evolutionary Algorithms in Theory and Practice—Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, University Press, Oxford, UK, 1996.

[10] L. Fogel, A. Owens, and M. Walsh, *Artificial Intelligence through Simulated Evolution*, John Wiley & Sons, New York, NY, USA, 1996.

[11] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[12] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[13] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[14] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, IEEE, Perth, Australia, December 1995.

[15] I. Fister, I. Fister Jr., X. -S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," *Swarm and Evolutionary Computation*, vol. 13, pp. 34–46, 2013.

[16] I. Fister, X. -S. Yang, J. Brest, and I. Fister Jr., "Modified firefly algorithm using quaternion representation," *Expert Systems with Applications*, vol. 40, no. 18, pp. 7220–7230, 2013.

[17] I. Fister Jr., D. Fister, and I. Fister, "A comprehensive review of cuckoo search: variants and hybrids," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, no. 4, pp. 387–409, 2013.

[18] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, pp. 65–74, Springer, New York, NY, USA, 2010.

[19] I. Fister Jr., D. Fister, and I. Fister, "Differential evolution strategies with random forest regression in the bat algorithm," in *Proceeding of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, pp. 1703–1706, ACM, 2013.

[20] P. N. Suganthan, "Particle swarm optimiser with neighbourhood operator," in *Proceedings of the Congress on Evolutionary Computation (CEC '99)*, vol. 3, IEEE, Washington, Wash, USA, July 1999.

[21] H. Liu, A. Abraham, O. Choi, and S. H. Moon, "Variable neighborhood particle swarm optimization for multi-objective flexible job-shop scheduling problems," in *Simulated Evolution and Learning*, vol. 4247 of *Lecture Notes in Computer Science*, pp. 197–204, Springer, New York, NY, USA, 2006.

[22] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, pp. 760–766, Springer, New York, NY, USA, 2010.

[23] N. Chakraborti, R. Jayakanth, S. Das, E. D. Çalişir, and Ş. Erkoç, "Evolutionary and genetic algorithms applied to Li+-C system: calculations using differential evolution and particle swarm algorithm," *Journal of Phase Equilibria and Diffusion*, vol. 28, no. 2, pp. 140–149, 2007.

[24] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proceedings of the Congress on Evolutionary Computation*, vol. 1, pp. 81–86, IEEE, May 2001.

[25] Y. Shi and R. Eberhart, "Modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 69–73, IEEE, May 1998.

[26] E. Miller, "An introduction to the resource description framework," *D-Lib Magazine*, vol. 4, no. 5, pp. 14–25, 1998.

[27] D. Allemang and J. Hendler, *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*, Morgan Kaufmann, Amsterdam, The Netherlands, 2nd edition, 2011.

[28] S. Harris and A. Seaborne, *Sparql 1.1 Query Language*, 2013.

[29] rdflib: A python library for working with RDF, 2013, http://code.google.com/p/rdflib/.

[30] Numpy, 2013, http://www.numpy.org/.

[31] Matplotlib, 2013, http://matplotlib.org/.

[32] X.-S. Yang, "Appendix A: test problems in optimization," in *Engineering Optimization*, X.-S. Yang, Ed., pp. 261–266, John Wiley & Sons, Hoboken, NJ, USA, 2010.

[33] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937.

[34] M. Friedman, "A comparison of alternative tests of significance for the problem of *m* rankings," *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940.

[35] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.