REVISTA DE MATEMÁTICA: TEORÍA Y APLICACIONES 2015 **22**(2) : 341–367
CIMPA – UCR ISSN: 1409-2433 (PRINT), 2215-3373 (ONLINE)

SC – SYSTEM OF CONVERGENCE: THEORY AND FOUNDATIONS

SC – SISTEMA DE CONVERGENCIA: TEORÍA Y FUNDAMENTOS

SERGIO G. DE-LOS-COBOS-SILVA [*]

Received: 25/Feb/2014; Revised: 11/May/2015;
Accepted: 20/May/2015

[*]Universidad Autónoma Metropolitana-Iztapalapa, Departamento de Ingeniería Eléctrica, Av. San Rafael Atlixco 186, Col. Vicentina, Del. Iztapalapa, México D.F., C.P. 09340, México. E-Mail: cobos@xanum.uam.mx

Abstract

In this paper a novel system of convergence (SC) is presented as well as its fundamentals and computing experience. An implementation using a novel mono-objetive particle swarm optimization (PSO) algorithm with three phases (PSO-3P): stabilization, generation with broad-ranging exploration and generation with in-depth exploration, is presented and tested in a diverse benchmark problems. Evidence shows that the three-phase PSO algoritm along with the SC criterion (SC-PSO-3P)can converge to the global optimum in several difficult test functions for multiobjective optimization problems, constrained optimization problems and unconstrained optimization problems with 2 until 120,000 variables.

Keywords: particle swarm optimization; unconstrained optimization; constrained optimization; multiobjective optimization; fuzzy numbers.

Resumen

En este trabajo se presenta un novedoso sistema de convergencia (SC), sus fundamentos y la experiencia computacional. Se implementó en un algoritmo PSO monoobjetivo de tres fases (PSO-3P): Estabilización, generación y búsqueda en amplitud, generación y búsqueda a profundidad, el cual se probó con diversos problemas benchmark. La evidencia muestra que el algoritmo PSO de 3 fases junto con el criterio SC (SC-PSO-3P) convergen al óptimo global para diversas funciones consideradas como difíciles para problemas de optimización multiobjetivo, para problemas de optimización con restricciones y para problemas de optimización sin restricciones que van desde 2 hasta 120,000 variables.

Palabras clave: optimización por enjambres de partículas; optimización sin restricciones; optimización con restricciones; optimización multiobjetivo.

Mathematics Subject Classification: 90C26, 90C29, 90C59.

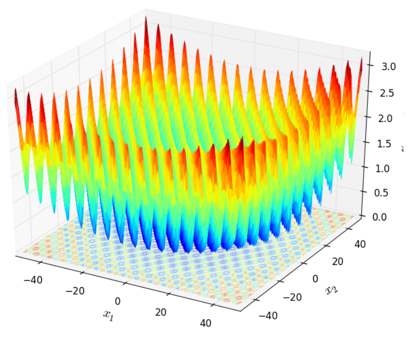
1 Introduction

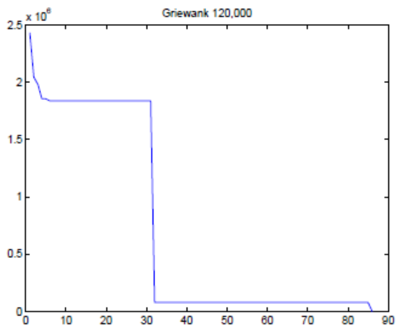
Generally, an optimization problem can be defined as:

$$\min_{x\in X} h(x).$$

There are several heuristic methods to solve optimization problems, but they have at least one of the following inconveniences:

1. Except under certain conditions, its convergence to global optima is not guaranteed.





(a) Griewank function ([5] hardness 6.08). **(b)** Convergence with SC for Griewank function (dimension=120,000).

Figure 1: Griewank function.

2. They do not have a consistent ability to jump deep valleys using only one criterion.

3. To date, there is not a general optimization problem solver: multi-objective problems, mono-objective problems with and without constraints, both discrete and continuous cases must be solved using different optimization algorithms.

In recent decades several heuristics in optimization methods have been developed. These methods are able to find solutions close to the optimum, where exact or analytical methods cannot produce optimal solutions within reasonable computation time. This is especially accurate when a global optimum is surrounded by many local optima, a situation known as deep valleys or black holes. This paper presents a novel approach that guarantees the solution of the above points using only one algorithm. In practice, it can be observed that the proposed system of convergence (SC) along with the three-phase PSO algorithm, allows to escape from suboptimal entrapments in many difficult instances. Moreover, SC is an alternative to the classical criteria, that allows the use of a weighted algorithm to find different solutions that those extreme points for non-convex problems.

In particular, evidence of convergence in Griewank function [5] with 120,000 variables $x_i \in [-600, 600]$ is presented (see Figure 1a and 1b) using a PSO based algorithm with only 3 particles can reach global optimum in 90 runs and 40 seconds on average using a Matlab program running on a Notebook with an Intel Atom N280 processor at 1.66 Ghz.

The work is divided as follows: background on fuzzy numbers is presented in the second section; in the third section the proposed Convergence System SC, as well as their definitions and fundamentals are presented. The general guidelines of novel particle swarm optimization (PSO) with 3 phases is presented in the fourth section. Numerical examples are presented in the fifth section. Finally, conclusions and future research are presented in the sixth section.

2 Fuzzy numbers

In this section, we introduce the basic concepts of fuzzy numbers based on [4].

A fuzzy number $A = (a, b, c, d; w)$, where $0 \leq w \leq 1$, and $a, b, c, d \in \mathbb{R}$, and $a \leq b \leq c \leq d$, is defined as a fuzzy subset of the real line $\mathbb{R}$ with membership function h_A such that:

1. h_A is a continuous mapping from $\mathbb{R}$ to the closed interval $[0, w]$.
2. $h_A = 0, \forall x \in (-\infty, a]$.
3. h_A is a strictly increasing function on $[a, b]$.
4. $h_A = w, \forall x \in [b, c]$, where w is a constant and $0 \leq w \leq 1$.
5. h_A is strictly decreasing on $[c, d]$.
6. $h_A = 0, \forall x \in [d, +\infty)$.

If $w = 1$, the generalized fuzzy number A is called a normal trapezoidal fuzzy number (see Fig. 2) denoted as $A = (a, b, c, d)$. If $a = b$ and $c = d$, then A is a crisp interval. If $b = c$, then A is a generalized triangular fuzzy number. If $a = b = c = d$, then A is a real number.

The membership function h_A of A can be expressed as:

$$h_A = \begin{cases} h_A^L(x), & a \leq x \leq b, \\ w & b \leq x \leq c, \\ h_A^R(x), & c \leq x \leq d, \\ 0, & \text{otherwise}, \end{cases}$$

where: $h_A^L(x) : [a, b] \longrightarrow [0, w]$ and $h_A^R(x) : [c, d] \longrightarrow [0, w]$ are continuous, $h_A^L(x)$ is strictly increasing and $h_A^R(x)$ strictly decreasing. The inverse functions of $h_A^L(x)$ and $h_A^R(x)$ are denoted by $g_A^L(x)$ and $g_A^R(x)$, respectively. These functions are continuous on $[0, w]$, this means both $\int_0^w g_A^L(x)$ and $\int_0^w g_A^R(x)$ exists [10].

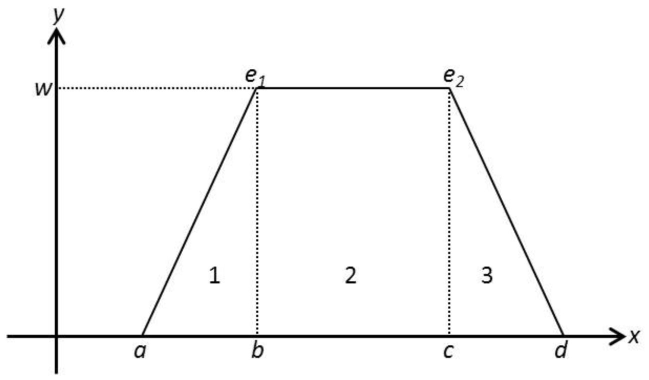


Figure 2: Trapezoidal fuzzy number.

Let be two trapezoidal fuzzy numbers. $\tilde{B}_1 = (a_1, b_1, c_1, d_1; w)$, $\tilde{B}_2 = (a_2, b_2, c_2, d_2; w)$ and $c \in \mathbb{R}$, then:

1. $c\tilde{B}_1 = (ca_1, cb_1, cc_1, cd_1; w)$
2. $\tilde{B}_1 + \tilde{B}_2 = (a_1 + a_2, b_1 + b_2, c_1 + c_2, d_1 + d_2; w)$

3 System of convergence SC

This section is based on [2]. Let:

1. $0 \leq f_1, f_2, f_3 \leq 1$,
2. $f_1 + f_2 + f_3 = 1$,
3. $0 < w \leq 1$,
4. $a_i \in \mathbb{R}, \alpha_i \in \mathbb{R}$.

Consider the following class of fuzzy numbers:

$C(\tilde{\mathbf{B}}) = \{\tilde{B}_i = (a_i, f_1, f_2, f_3, \alpha_i; w) |$ satisfying the preceding four conditions$\}$,

where:

1. $b_i = a_i + f_1\alpha_i$;
2. $c_i = a_i + (f_1 + f_2)\alpha_i$;
3. $d_i = a_i + (f_1 + f_2 + f_3)\alpha_i = a_i + \alpha_i$.

Property 1. For any two fuzzy numbers in $\tilde{B}_1, \tilde{B}_2 \in C(\tilde{\mathbf{B}})$, such that $\tilde{B}_1 = (a_1, f_1, f_2, f_3, \alpha_1; w)$ and $\tilde{B}_2 = (a_2, f_1, f_2, f_3, \alpha_2; w)$ and for all $c \in \mathbb{R}$, the following equations are satisfied:

1. $c\tilde{B}_1 = (ca_1, f_1, f_2, f_3, c\alpha_1; w)$,
2. $\tilde{B}_1 + \tilde{B}_2 = (a_1 + a_2, f_1, f_2, f_3, \alpha_1 + \alpha_2; w)$.

Definition 1. Given a function $G : C(\tilde{\mathbf{B}}) \longrightarrow \mathbb{R}$, and $\tilde{B}_1, \tilde{B}_2 \in C(\tilde{\mathbf{B}})$ it is said that they are SC-equivalent if and only if:

$$G(\tilde{B}_1) = G(\tilde{B}_2).$$

Definition 2. Given a function $G : C(\tilde{\mathbf{B}}) \longrightarrow \mathbb{R}$, and a $g_G \in \mathbb{R}$, in the codomain of G, the following SC-equivalence class is defined $\tilde{\mathbf{B}}_{g_G} \subset C(\tilde{\mathbf{B}})$ as:

$$\tilde{\mathbf{B}}_{g_G} = \left\{ \tilde{B} \in C(\tilde{\mathbf{B}}) \mid G(\tilde{B}) = g_G \right\}.$$

Remarks:

1. $\tilde{\mathbf{B}}_{g_G}$ is an equivalence class.
2. $\tilde{\mathbf{B}}_{g_1} \bigcap \tilde{\mathbf{B}}_{g_2} = \emptyset$, for all $g_1 \neq g_2$.
3. $\bigcup_{g \in \mathbb{R}} \tilde{\mathbf{B}}_g = C(\tilde{\mathbf{B}})$.

Definition 3. Let f_1, f_2, f_3, w be eral values that satisfy:

1. $0 \leq f_1, f_2, f_3 \leq 1$,
2. $f_1 + f_2 + f_3 = 1$,
3. $0 < w \leq 1$;

we define:

1. $F_1 = (1 + f_2)$,
2. $F_2 = (2f_1^2 + 6f_1f_2 + 3f_2^2 + 3f_3 - 2f_3^2)$,
3. $F_3 = (3f_1^3 + 4f_2^3 + 3f_3^3 + 12f_1f_2^2 + 12f_2f_1^2 + 6f_3 - 8f_3^2)$, it can be proven that $F_3 > 0$,
4. $A = \frac{12}{w^3(1+3f_2)}$.

Theorem 1. Let:

$$G : C(\tilde{\mathbf{B}}) \longrightarrow \mathbb{R}$$

given by:

$$G(\tilde{B}) = A\left[\alpha^2 F_3 - 4\alpha F_2 + 6aF_1\right].$$

Then G is bijective on each SC-equivalence class.

Proof. G **is injective** by construction.

G **is surjective**:
Let $g \in \mathbb{R}$, to prove that there is $\tilde{\mathbf{B}}_g \subset C(\tilde{\mathbf{B}})$ such that:

$$G(\tilde{B}) = g, \forall \tilde{B} \in \tilde{\mathbf{B}}_g$$

it will be proven by construction.

Let $\tilde{\mathbf{B}}_g = \{\tilde{B} = (a, f_1, f_2, f_3, \alpha; w)\}$ which satisfy the conditions of theorem 1, and let:

$$\begin{aligned} \alpha &= \frac{2F_2}{F_3}, \\ a &= \frac{4F_2^2 + A^{-1}F_3 g}{6F_1F_3}. \end{aligned}$$

By definition we have:

$$\begin{aligned} G(\tilde{B}) &= A\left[\alpha^2 F_3 - 4\alpha F_2 + 6aF_1\right] \\ &= A\left[\left(\frac{2F_2}{F_3}\right)^2 F_3 - 4\frac{2F_2}{F_3}F_2 + 6\frac{4F_2^2 + A^{-1}F_3 g}{6F_1F_3}F_1\right] \\ &= A\left[\frac{4F_2^2}{F_3} - \frac{8F_2^2}{F_3} + \frac{4F_2^2}{F_3} + A^{-1}g\right] = g. \end{aligned}$$

Therefore, function G is bijective on each class. □

Proposition 1. Let $\tilde{B} \in C(\tilde{\mathbf{B}})$, $s, s^*, \epsilon \in \mathbb{R}$ such that $s = s^* + \epsilon$, then:

$$\lim_{\epsilon \to 0} G(\tilde{Bs}) = G(\tilde{Bs^*}).$$

Proof. Let $\tilde{B} \in C(\tilde{\mathbf{B}})$ such that $\tilde{B} = (a, f_1, f_2, f_3, \alpha; w)$, and $s \in \mathbb{R}$ then:

$$\tilde{B}s = (as, f_1, f_2, f_3, \alpha s; w)$$

Using the definition of G we have that:

$$\begin{aligned}
G(\tilde{B}s) &= A\left[(\alpha s)^2F_3 - 4(\alpha s)F_2 + 6(as)F_1\right]\\
&= A\left[((\alpha(s^*+\epsilon))^2F_3 - 4\alpha(s^*+\epsilon)F_2 + 6a(s^*+\epsilon)F_1\right]\\
&= A[(\alpha s^*)^2F_3 + 2\alpha^2 s^*\epsilon F_3 + (\alpha\epsilon)^2F_3 - 4\alpha s^*F_2 - 4\alpha\epsilon F_2 +\\
&\quad 6as^*F_1 + 6a\epsilon F_1]\\
&= A[(\alpha s^*)^2F_3 - 4\alpha s^*F_2 + 6as^*F_1] + A[2\alpha^2 s^*\epsilon F_3 + (\alpha\epsilon)^2F_3\\
&\quad -4\alpha\epsilon F_2 + 6a\epsilon F_1].
\end{aligned}$$

Therefore:

$$\begin{aligned}
\lim_{\epsilon\to 0} G(\tilde{B}s) &= \lim_{\epsilon\to 0} A\left[(\alpha s^*)^2F_3 - 4\alpha s^*F_2 + 6as^*F_1\right]\\
&\quad + \lim_{\epsilon\to 0} A\left[2\alpha^2 s^*\epsilon F_3 + (\alpha\epsilon)^2F_3 - 4\alpha\epsilon F_2 + 6a\epsilon F_1\right]\\
&= G(\tilde{B}s^*)
\end{aligned}$$

□

Proposition 2. Let $\tilde{B}_i \in C(\tilde{\mathbf{B}}), i = 1,\ldots,n$, and $s_i, \epsilon_i \in \mathbb{R}$ such that $s_i = s_i^* + \epsilon_i$, $i = 1,2,\ldots,n$, then:

$$\lim_{\{\epsilon_i\}_{i=1}^n\to 0} G\left(\sum_{i=1}^n \tilde{B}_i s_i\right) = G\left(\sum_{i=1}^n \tilde{B}_i s_i^*\right)$$

Proof. It will be demonstrated by mathematical induction.
i.- First it will be proven for $k = 2$:

$$\begin{aligned}
G(\tilde{B}_1s_1 + \tilde{B}_2s_2) &= A[(s_1\alpha_1 + s_2\alpha_2)^2F_3 - 4(s_1\alpha_1 + s_2\alpha_2)F_2\\
&\quad +6(s_1a_1 + s_2a_2)F_1]\\
&= A\left[((s_1^*\alpha_1)^2 + 2s_1^*\alpha_1 s_2^*\alpha_2 + (s_2^*\alpha_2)^2)F_3 +\right.\\
&\quad \left. -\, 4(s_1^*\alpha_1 + s_2^*\alpha_2)F_2 + 6(s_1^*a_1 + s_2^*a_2)\right]\\
&= A[(s_1^*\alpha_1 + s_2^*\alpha_2)^2F_3 - 4(s_1^*\alpha_1 + s_2^*\alpha_2)F_2 +\\
&\quad 6(s_1^*a_1 + s_2^*a_2)F_1] + A[H(\epsilon_1,\epsilon_2)]
\end{aligned}$$

where:

$$\begin{aligned}
H(\epsilon_1,\epsilon_2) &= (2\epsilon_1 s_1^*\alpha_1 + (\epsilon_1\alpha_1)^2 +\\
&\quad 2\alpha_1\alpha_2(s_1^*\epsilon_2 + s_2^*\epsilon_1 + \epsilon_1\epsilon_2) + 2\epsilon_2 s_2^*\alpha_2 + (\epsilon_2\alpha_2)^2)F_3 +\\
&\quad -4(\epsilon_1\alpha_1 + \epsilon_2\alpha_2)F_2 + 6(\epsilon_1 a_1 + \epsilon_2 a_2)F_1.
\end{aligned}$$

Therefore:

$$\begin{aligned}\lim_{\{\epsilon_1,\epsilon_2\}\to 0} G(\tilde{B}_1 s_1 + \tilde{B}_2 s_2) &= \lim_{\{\epsilon_1,\epsilon_2\}\to 0} A[(s_1^*\alpha_1 + s_2^*\alpha_2)^2 F_3 - 4(s_1^*\alpha_1 + s_2^*\alpha_2)F_2 \\ &\quad +6(s_1^* a_1 + s_2^* a_2)F_1] + \lim_{\{\epsilon_1,\epsilon_2\}\to 0} A[H(\epsilon_1,\epsilon_2)] \\ &= G(\tilde{B}_1 s_1^* + \tilde{B}_2 s_2^*).\end{aligned}$$

ii.- Using mathematical induction, it is assumed true for $k = n-1$ and will be proven for $k = n$.

Note that $\forall i = 1, 2, \dots, k \leq n-1$ it is true that:

$$G\left(\sum_{i=1}^{k} (\tilde{B}_i s_i)^2\right) = G\left(\sum_{i=1}^{k} (\tilde{B}_i s_i^*)^2\right) + A\left[H(\{\epsilon_i\}_{i=1}^{k})\right]$$

such that: $\lim_{\{\epsilon_i\}\to 0} H\left(\{\epsilon_i\}_{i=1}^{k}\right) = 0$.

iii.- For $k = n$, we have:

$$\text{Let } \tilde{B} = \sum_{i=1}^{n} \tilde{B}_i s_i, \text{ such that } \tilde{B}_i \in C(\mathbf{\tilde{B}}), i = 1, 2, \dots, n.$$

It can be seen that:

1. $\tilde{B} = \sum_{i=1}^{n} \tilde{B}_i s_i = \sum_{i=1}^{n-1} \tilde{B}_i s_i + \tilde{B}_n s_n = \tilde{B}_{n-1}' + \tilde{B}_n s_n$;

2. $\alpha = \sum_{i=1}^{n-1} (\alpha_i s_i) + \alpha_n s_n = (\alpha_{n-1}') + \alpha_n s_n$;

3. $a = \sum_{i=1}^{n-1} (a_i s_i) + a_n s_n = (a_{n-1}') + a_n s_n$.

Therefore:

$$\begin{aligned}
G\left(\sum_{i=1}^{n}\tilde{B}_i s_i\right) &= A\Big[(\alpha'_{n-1}+s_n\alpha_n)^2F_3-4(\alpha'_{n-1}+s_n\alpha_n)F_2+ \\
&\quad +6(a'_{n-1}+s_na_n)F_1+H(\{\epsilon_i\}_{i=1}^{n-1})\Big] \\
&= A\Big[(\alpha'_{n-1}+(s_n^*+\epsilon_n)\alpha_n)^2F_3 \\
&\quad -4(\alpha'_{n-1}+(s_n^*+\epsilon_n)\alpha_n)F_2++6(a'_{n-1}+s_na_n)F_1 \\
&\quad +H(\{\epsilon_i\}_{i=1}^{n-1})] \\
&= A\Big[\Big((\alpha'_{n-1})^2+(s_n^*\alpha_n)^2+2\alpha'_{n-1}s_n^*\alpha_n\Big)F_3 \\
&\quad -4\left(\alpha'_{n-1}+s_n^*\alpha_n\right)F_2+ \\
&\quad +6\left((a'_{n-1}+s_n^*a_n)F_1+H(\{\epsilon_i\}_{i=1}^{n-1}\right)+H(\epsilon_n)\Big] \\
&= A\left[\left(\sum_{i=1}^{n}(\alpha_i s_i^*)^2\right)F3-4\left(\sum_{i=1}^{n}(\alpha_i s_i^*)\right)F_2+\right. \\
&\quad \left.+6\left(\sum_{i=1}^{n}a_i s_i^*\right)F_1+H(\{\epsilon_i\}_{i=1}^{n-1})+H(\epsilon_n)\right].
\end{aligned}$$

where: $H(\epsilon_n)=\epsilon_n(2s_n^*\alpha_n^2+2\alpha'_{n-1}\alpha_n+\alpha_n^2\epsilon_n+\alpha_nF_2+a_nF_1)$.
Hence:

$$\begin{aligned}
\lim_{\{\epsilon_i\}_{i=1}^{n}\to 0}G\left(\sum_{i=1}^{n}\tilde{B}_i s_i\right)&=\lim_{\{\epsilon_i\}_{i=1}^{n}\to 0}A\left[\left(\sum_{i=1}^{n}(\alpha_i s_i^*)^2\right)F3-4\left(\sum_{i=1}^{n}(\alpha_i s_i^*)\right)F_2+\right. \\
&\quad \left.+6\left(\sum_{i=1}^{n}a_i s_i^*\right)F_1+H(\{\epsilon_i\}_{i=1}^{n-1})+H(\epsilon_n)\right] \\
&=G\left(\sum_{i=1}^{n}\tilde{B}_i s_i^*\right).
\end{aligned}$$

□

Definition 4. Consider the following multiobjective optimization problem $h(x)$, $x \in X \subseteq \mathbb{R}^n$, where:

$$h(x) = (h_1(x), h_2(x), \ldots, h_k(x)); \;\; h_i : \mathbb{R}^n \to \mathbb{R}, i = 1, 2, \ldots, k.$$

Then:

1. Given $y = (y_1, y_2, \ldots, y_k)$ is said that it dominates $z = (z_1, z_2, \ldots, z_k)$ if and only if $\forall i \in \{1, 2, \ldots, k\} \;\; y_i \leq z_i$ and $\exists i_0 \in \{1, 2, \ldots, k\}$ such that $y_{i_0} < z_{i_0}$.

2. Pareto Frontier, (PF) = $\{ \; h(x) |$ are non dominated.$\}$

3. A solution vector x^* is said to be Pareto Optimal if there is no other vector x such that $h(x)$ dominates $h(x^*)$.

Definition 5. The SC-Frontier (SC-F) is defined as:

SC-F $= \{h(x) | \forall \epsilon > 0 \exists y \in$ Codomain$(h) -$Image(h) such that $||y - h(x)|| < \epsilon\}$.

Proposition 3. PF $\subseteq$ SC-F.

Proof. The proof will be done by contradiction: Suppose:

1. $y^* = (y_1^*, y_2^*, \ldots, y_k^*) \in$ PF;

2. $y^* \notin$ SC-F.

Then we have:

1. $y^* \in$ Image(h), and y^* is non dominated.

2. $\exists \; \epsilon > 0$, such that if $||y^* - y|| \leq \epsilon \Rightarrow y \in$ Image(h).

Consider a point: $y_0 = (y_1^* - \frac{\epsilon}{\sqrt{k}}, y_2^* - \frac{\epsilon}{\sqrt{k}}, \ldots, y_k^* - \frac{\epsilon}{\sqrt{k}})$. Note that:

$$||y^* - y_0|| = \sqrt{(\frac{\epsilon}{\sqrt{k}})^2 + \ldots + (\frac{\epsilon}{\sqrt{k}})^2} = \epsilon,$$

for that reason $y_0 \in$ Image(h) and y_0 dominates y^*, which is a contradiction, therefore: PF $\subseteq$ SC-F. $\square$

Definition 6. Criterion SC: Given a multiobjective optimization problem $h(x)$, $x \in X \subseteq \mathbb{R}^n$, where:

$$h(x) = (h_1(x), h_2(x), \ldots, h_k(x)); \;\; h_i : \mathbb{R}^n \to \mathbb{R}, i = 1, 2, \ldots, k,$$

and

$$\tilde{\mathbf{B}} = \left(\tilde{B_1}, \tilde{B_2}, \ldots, \tilde{B_k}\right),$$

where $\tilde{B_i} \in C(\tilde{\mathbf{B}}), \;\; i = 1, \ldots, k$.
Define:

$$G\left(\tilde{\mathbf{B}}h(x)\right) = G\left(\sum_{i=1}^{k} \tilde{B_i} h_i(x)\right).$$

Definition 7. Given a multiobjective optimization problem and a $\tilde{\mathbf{B}}$, define:

$$x^*_{\tilde{B}} = \text{ argmin } \{G(\tilde{\mathbf{B}}h(x)) \mid h(x) \in \text{SC-F}\}.$$

Definition 8. The optimal set of SC solutions is defined as:

$$\text{SC-O } = \{x^*_{\tilde{B}}, \;\; \tilde{B} \in C(\tilde{\mathbf{B}})\}.$$

3.1 Empirical boundary conditions

Given an optimization problem with k objective functions, the following values have been used empirically:

1. $f_1 = f_3 = 0.25, f_2 = 0.5, w = 1,$
2. $\alpha_i \in [-12K_1, 12K_1], i = 1, 2, \ldots, k,$
3. $a_i = K_2 + K_3 g_i,$

where: $K_1 = \frac{2F_2}{F_3}$, $K_2 = \frac{2F_2}{3kF_1}$, $K_3 = \frac{A^{-1}F_3}{6kF_1}$, $g_i = \begin{cases} 1 & \text{if} \quad h_i(x^*) > 0 \\ 0 & \text{if} \quad h_i(x^*) = 0 \\ -1 & \text{if} \quad h_i(x^*) < 0. \end{cases}$

3.2 SC algorithm

The main idea is to use a kind of fitness function instead of the original objective function, this fitness function is obtained through defuzzification (transformation by a set of trapezoidal fuzzy numbers). This defuzzification (SC criterion) has interesting properties, among which are the quadratic topology (see definition for G in Theorem 1) and its convergence to the same point as the original function (see Propositions 1 and 2).

Step 0:

1. Provide stopping criteria.
2. Set f_1, f_2, f_3, w such that they satisfy the conditions of Theorem 1.
3. Given the function to optimize $h = (h_1, h_2, \dots h_k)$, find the boundary conditions (or use the suggested empirical conditions), take $\tilde{B}_i \in C(\tilde{\mathbf{B}})$, $i =$ 1,2,$\dots, k$.
4. Set values for a_i and α_i.
5. Take any value x in the domain of h and evaluate:

$$G\left(\tilde{B}h(x)\right) = G\left(\sum_{i=1}^{k} \tilde{B}_i h_i(x)\right).$$

Set:

$$g_G(x) \leftarrow G\left(\tilde{B}h(x)\right).$$

Step 1: Through m neighborhoods, find $x_j \in \text{Phase}^j(x)$ and evaluate $G\left(\tilde{B}h(x_j)\right), j = 1, 2, \dots, m$

$$g_G(x_1) = \min_j \{G(\tilde{B}h(x_j)), G(\tilde{B}h(x))\},$$

$$x \leftarrow x_1.$$

Step 2: Stop in the following conditions:

1. if for a certain number of iterations $g_G(x_1) = g_G(x)$, then it is considered that an optimum has been reached,
2. a stop criterion was met.

In any other case go to **Step 1**.

Remark: When g_G is very close to zero, it can be used, for example: $(1 - g_G)^2$.

4 Particle swarm optimization

The particle swarm optimization (PSO) [8] is a subset of what is known as swarm intelligence and has its roots in artificial life, social psychology, engineering and computer science. PSO is based on the use of a set of particles or agents that correspond to states of an optimization problem, where each particle moves across the solution space in search of an optimal position or at least a good solution. In PSO the agents communicate with each other, and the agent with a better position (measured according to an objective function) influences the others by attracting them to it.

The population is started by assigning an initial random position and velocity for each element. At each iteration, the velocity of each particle is randomly accelerated towards its best position (where the value of the fitness function or objective function improves) and also considering the best positions of their neighbors.

To solve a problem, PSO uses a dynamic management of particles; this approach allows breaking cycles and diversifying the search. In this work, a r-particle swarm is represented at time t under the form:

$$\theta_{1,t}, \theta_{2,t}, \ldots, \theta_{r,t}$$

with $\theta_{j,t}$ in a domain, $j = 1, 2, \ldots, r$, then a movement of the swarm is defined according to equation (1):

$$\theta_{j,t+1} = \theta_{j,t} + V_{j,t+1}, \tag{1}$$

where the velocity $V_{j,t+1}$ is given in equation (2):

$$V_{j,t+1} = \alpha V_{j,t} + rand(0, \varphi_1)[\theta'_{j,t} - \theta_{j,t}] + rand(0, \varphi_2)[\theta'_{g,t} - \theta_{j,t}], \tag{2}$$

where:

D: space of feasible solutions,

$V_{j,t}$: speed at time t of the j-th particle,

$V_{j,t+1}$: speed at time $t+1$ of the j-th particle,

$\theta_{j,t}$: j-th particle at time t,

$\theta'_{g,t}$: the particle with the best value for all time t,

$\theta'_{j,t}$: j-th particle with the best value to the time t,

rand(0,φ): random value uniformly distributed on the interval [0,φ],

α: parameter of scale.

The PSO algorithm is described in Table 1.

Table 1: PSO algorithm.

1. Create a population of particles distributed in the feasible space.
2. Evaluate each position of the particles according to the objective function (fitness function).
3. If the current position of a particle is better than the previous update it.
4. Determine the best particle (according to the best previous positions).
5. Update the particle velocities $j = 1, 2, \ldots, r$ according to: $V_{j,t+1} = \alpha V_{j,t} + rand(0, \varphi_1)[\theta'_{j,t} - \theta_{j,t}] + rand(0, \varphi_2)[\theta'_{g,t} - \theta_{j,t}]$.
6. Move the particles to new positions according to: $\theta_{j,t+1} = \theta_{j,t} + V_{j,t+1}$.
7. Go to Step 2 until the termination criterion is satisfied.

4.1 SC-PSO-3P

The three-phase PSO algorithm (PSO-3P) along with the SC criterion is named SC-PSO-3P. The SC-PSO-3P algorithm is described in Table 2.

As can be seen in the algorithm in table 2, the key modification is to consider the G function given in Theorem 1, the SC criterion, and the 3 phases, the rest of the algorithm remains the similar.

However, the position of the particles can be modified using different strategies, which are sequentially applied in three phases of the searching process.

In phase 1, called stabilization, according to the description presented in section 4, the PSO-3P algorithm generates randomly a set of particles in the solution space. Then, during itF_1 iterations the position of the particles is modified using equations (1) and (2). Thus, at the end of this phase the particles are concentrated, or stabilized, in a promising region.

When phase 1 is completed, a breadth-first search strategy, called phase 2, is incorporated. In this phase, if the global best solution is not improved after three consecutive iterations, the position of M_2 particles is modified randomly. However, the particle with the best known position is preserved. Thus, the population is dispersed in the solution space, but it can be attracted to the best region visited so far. This diversification strategy is considered during itF_2 iterations.

Finally, phase 3 is initialized. During itF_3 iterations the following depth-first search strategy is applied. If the global best solution is not improved after three consecutive iterations, M_3 particles are randomly positioned in a neighborhood of the best known solution. Thus, phase 3 includes an intensification process in a promising region.

Table 2: SC-PSO algorithm.

```
1. Set variables c, itF1, itF2, itF3, MaxIt and prop.

2. Create a population of nPop particles distributed in the feasible space.

3. Set cont=0 and it=1. Evaluate each position of the particles according to the
   function given in Definition 6 (fitness function).

4. If the current position of a particle is better (with respect to SC) than the previ-
   ous update it.

5. Determine the best particle (according to the best previous positions against the
   criterion SC). If a better particle cannot be found, let cont=cont+1.

6. Update the particle velocities $j = 1, 2, \dots,$ nPop according to:
   $V_{j,t+1} = \alpha V_{j,t} + rand(0, \varphi_1)[\theta'_{j,t} - \theta_{j,t}] + rand(0, \varphi_2)[\theta'_{g,t} - \theta_{j,t}]$.

7. Move the particles to new positions according to: $\theta_{j,t+1} = \theta_{j,t} + V_{j,t+1}$.

8. (Phase 1: Stabilization)

       if it $\leq$ itF1: go to Step 11.

9. (Phase 2: Generation with broad-range exploration).

       if itF1 $<$ it $\leq$ itF2
           If cont=c
               Set n=1. While n $\leq$ nPop*prop
                   Create a random particle and, with a probability bigger than
                   0.5 a 0.5 substitute randomly a particle in the swarm.
                   Set n=n+1.
           Set cont=0.
       go to Step 11.

10. (Phase 3: Generation with in-depth exploration).

       if itF2 $<$ it $\leq$ itF3
           If cont=c
               Set n=1. While n $\leq$ nPop*prop
                   Create a random particle in a variable neighborhood of $\theta'_{g,t}$
                   and substitute randomly a particle in the swarm.
                   Set n=n+1.
           Set cont=0.
       go to Step 11.

11. Select the best nPop particles according to SC criterion.

12. Set it=it+1. Go to Step 3 until the termination criterion is satisfied.
```

Any version of standard PSO can be used, per example in order to be recreated by anyone interested, a version of PSO found in [7] was taken, which, although multiobjective, was implemented and modified, making it mono-objective using the SC criterion and 3 phases neighborhoods to solve all the reported problems. Additional modifications to the code are listed below:

1. Set the number of particles in the repository (nRep) to 1.
2. Store the solution that improves according to the SC criterion.
3. In the cleaning routine the worst solutions regarding SC are removed.
4. Implement the Phase 2: generation with broad-ranging exploration.
5. Implement the Phase 3: generation with in-depth exploration.

5 Computational results

The algorithm was implemented in Matlab R2008a and was run in an Intel Core i5-3210M processor computer at 2.5 GHz, running on Windows 8.

In this work, a novel criterion called System of Convergence (SC) is proposed, which was implemented using a novel neighborhood search, PSO-based algorithm with three phases: stabilization, generation with broad-ranging exploration and generation with in-depth exploration.

In this paper, the functions for unconstrained optimization were taken from [6], [12] and [5]. For continuous optimization without constraints the graphs were taken from [5]. For multi-objective optimization references [13], [3] and [1] were used. Finally, restricted optimization functions were taken from [9].

5.1 Unconstrained optimization

In this section, the efficiency was defined as:

$$\text{efficiency} = \begin{cases} 1 - abs(\frac{h(x)-h(x^*)}{h(x^*)}) & \text{if} \quad h(x^*) \neq 0 \\ \\ 1 - abs(h(x)) & \text{if} \quad h(x^*) = 0. \end{cases}$$

When the efficiency was greater than .999999, it is rounded to 1.

Reference [5] is quoted verbatim: “It can be easily seen that the DeVilliers-Glasser02, Damavandi and CrossLegTable were always hard problems for all the algorithms (see table 3).” It can be remarked that the success rate is very high when SC is used.

Table 3 shows the percentage of success for some difficult problems presented in [5].

At Table 4 results when using 3 particles (Part) to at least one success in 24 runs of 500 iterations each are presented. The average time (time (sec)) in seconds per run, the percentage of success (Perc) and the average number of iterations per run (Mean Iter) are displayed in subsequent columns.

Benchmark test functions		(hardness)
Optimization Method	N	Overall Success (%)
DeVilliersGlasser02	5	0
Damavandi	2	0.25
CrossLegTable	2	0.83
XinSheYang03	2	1.08
Griewank	2	6.08
XinSheYang02	2	31.33

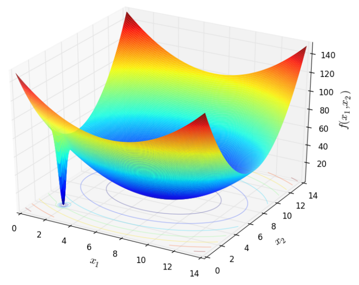
Table 3: Hardness according to [5].

Function(dim) [[5]]	Figure	Part.	time (sec)	Perc.	Mean Iter
Cross Leg table(2) [0.83]	3b	3	0.8	75.0	223.6
Devillier Glasser 02(5) [0.00]	-	3	0.6	100.0	151.9
Griewank(2) [6.08]	1a	3	0.7	100.0	209.5
Griewank(120000)	1b	3	3.5	100.0	87.4
Xin She Yang02(2) [31.33]	3c	3	0.6	95.8	170.1
Xin She Yang03(2) [1.08]	3d	3	0.5	100.0	155.3

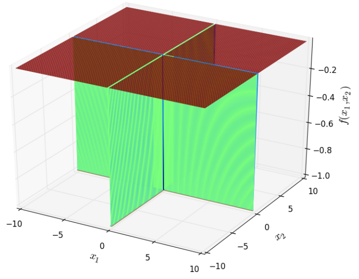
Table 4: Results with 3 particles, 500 iterations per run.

Function(dim.) [hardness]	Figure	Part.	Mean Eff	time (sec)	Mean Iter
Cross Leg table(2) [0.83]	3b	60	0.833426	8.07	198.42
Damavandi(2) [0.25]	3a	60	0.989719	12.42	490.25
Devillier Glasser 02(5) [0.00]	-	12	1.000000	2.60	499.00
Griewank(2) [6.08]	1a	24	1.000000	1.53	191.04
Griewank(120000)	1b	12	1.000000	13.79	87.00
Xin She Yang02(2) [31.33]	3c	48	1.000000	4.83	148.83
Xin She Yang03(2) [1.08]	3d	3	1.000000	0.55	160.33

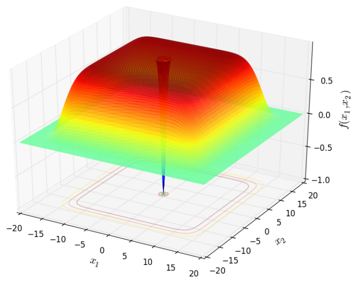
Table 5: Time and efficiency table.



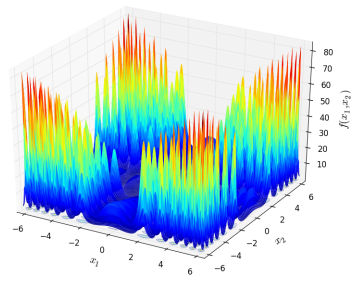
(a) Damavandi function (0.25*).



(b) CrossLegTable function (0.83*).



(c) XinSheYang03 function (1.08*).



(d) XinSheYang02 function (31.33*).

Figure 3: Different Functions and its *hardness [5].

In Table 5 are presented, for each function, the number of particles (Part), the average efficiency (Mean Eff), the average time per run in seconds (time (sec)) and the average number of iterations per run (Mean It), all data considers 24 runs and 500 as maximum of iterations.

5.2 Multiobjetive optimization

5.2.1 ZDT test functions [13]

Each of the test functions defined below is structured in the same manner and consists itself of three functions f_1, g, h:

$$\begin{array}{ll} \text{Minimize} & T(X) = (f_1(x_1), f_2(X)) \\ \text{subject to} & f_2(X) = g(x_2, x_3, \ldots, x_m)h(f_1(x_1), g(x_1, x_2, \ldots, x_m)), \\ \text{where} & X = (x_1, x_2, \ldots, x_m). \end{array}$$

The test function ZDT6 includes two difficulties caused by the nonuniformity of the search space: first, the Pareto-optimal solutions are nonuniformily distributed along the global Pareto front (the front is biased for solutions for which $f_1(x)$ is near one); second, the density of the solutions is lower near the Pareto-optimal front and bigger further from the front:

$$\begin{aligned} lf_1 &= 1 - \exp(-4x_1)\sin^6(6\pi x_1) \\ g(x_2, \ldots, x_m) &= 1 + 9(\sum_{i=2}^{m} \frac{x_i^2}{(m-1)})^{0.25} \\ h(f_1, g) &= 1 - (\frac{f_1}{g})^2 \end{aligned}$$

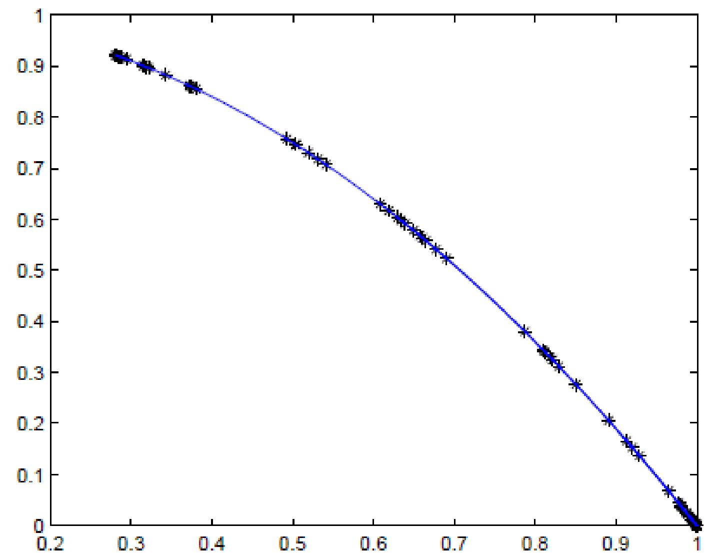
where $m = 10$ and $x_i \in [0, 1]$. The global Pareto-optimal front is formed with $g(X) = 1$ and is nonconvex.

For ZDT6 problems (see Figure 4), efficiency was defined as:

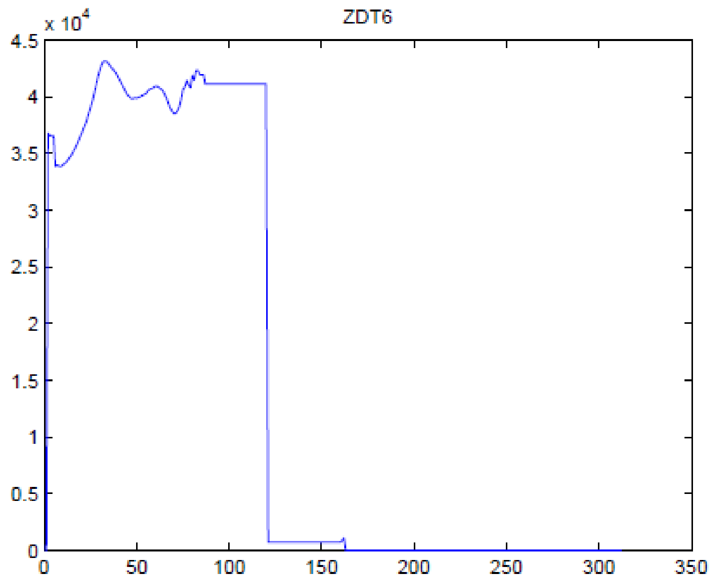
$$\text{efficiency} = 1 - \sum_{i=2}^{m} x_i^2.$$

Function(dim)	Part.	Mean It.	Perc.	Time (sec)
ZDT6(10)	3	357.1	73	1.2

Table 6: Results with 3 particles and 100 runs.



(a) 100 SC points for ZDT6(10).



(b) Convergence of ZDT6(120,000).

Figure 4: Function ZDT6.

The points found on the Pareto frontier for ZDT6 of dimension 10, are presented in Figure 4a, in Figure 4b is presented the convergence of a random run for ZDT6 of dimension 120,000.

In Table 6 the results of 100 runs using 3 particles, are presented with a maximum of 500 iterations each. The number of times the optimum was reached (Perc), the average iterations per run (Mean It) as well as the average time per run (Time (sec)) are also presented.

In Table 7 the average time per run (Time (sec)) given in seconds, the average efficiency obtained and the number of runs in which the optimal (Perc / Runs) was reached for ZDT6 problems, 100 runs were performed.

Function(dim)	Time(sec)	Mean Eff.	Mean It	Per/Runs
ZDT6(10)	4.15	1.00000000	105.88	100/100

Table 7: Efficiency with 120 particles and 500 iterations.

In Table 8 the average time per run (Time (sec)) given in seconds, the number of runs that reach the optimum (Perc / runs) and the average number of iterations (Mean It) to reach the optimum are presented. 120 particles and a maximum of 1000 iterations per run was used.

Function(dim)	Time(sec)	Per/runs	Mean It.
ZDT6(120,000)	290.22	17/24	524.88

Table 8: Results Table using SC-PSO-3P (120,000 variables).

5.2.2 DTLZ test problems

In DTLZ problems [3], the total number of variables is $n = M + k - 1$, where M is the number of objectives and k is the number of variables of the functional g. All of them are minimization problems.

The DTLZ1 problem is defined as:

$$\begin{aligned}
f_1 &= \frac{1}{2}(1+g)\prod_{i=1}^{M-1} x_i \\
f_{m=2:M-1} &= \frac{1}{2}(1+g)\prod_{i=1}^{M-m} x_i(1-x_{M-m+1}) \\
f_M &= \frac{1}{2}(1+g)(1-x_1) \\
g &= 100\left[k+\sum_{i=1}^{k}((z_i-0.5)^2-\cos(20\pi(z_i-0.5)))\right].
\end{aligned}$$

For DTLZ1, the Pareto-optimal solutions correspond to $x_i^* = 0.5, x_i \in X_k$ and the objective function values lie on the linear hyperplane: $\sum_{m=1}^{M} f_m^* = 0.5$. The search space contains $(11^k - 1)$ local Pareto-optimal fronts. Efficiency is measured as:

$$\text{Efficiency} = 1 - abs\left(\frac{0.5 - \sum_{m=1}^{M} f_m}{0.5}\right)$$

and distance as:

$$\text{distance} = \sum_{x_i \in X_k} (0.5 - x_i)^2.$$

For the function DTLZ1(5.15), in 23 out of 24 runs the Pareto Optimal Frontier was reached, but the run where optimum was not reached, the solution was far from it, for this reason the percentage of efficiency and the distances to the points was greatly affected.

Stop criteria were (number of iterations) or (distance $< 10^{-12}$ and efficiency > 0.99999).

In Table 9 the second column shows the average time per run (Time) in seconds, third column shows the average iterations (Mean It) per run. Last column shows the percentage by which the optimum was reached in 24 runs, using only three particles a maximum of 1500 iterations per run.

In Table 10 the second column shows the average time per run (Time) in seconds, third column shows the number of particles used (Part), fourth column shows the maximum number of iterations, fifth and sixth columns show the average iterations (Mean It) per run and the average efficiency (Mean Eff). Last two columns show the average distance (Mean Dist) and the percentage by which the optimum was reached in 24 runs.

Function (M,n)	Time (sec)	Mean It.	Perc.
DTLZ1(3,8)	0.88	194.67	95.83%
DTLZ1(5,15)	0.75	167.46	100.00%
DTLZ1(50,100)	4.89	1031.50	37.50%
DTLZ1(100,175)	7.93	1438.96	4.17%

Table 9: Percentage with three particles, 1500 iterations and 24 runs.

Function(M,n)	Time	Part	Iter	Mean It	Mean Eff.	Mean Dist.	Per/runs
DTLZ1(3,8)	22.83	120	500	168.33	0.9583544	4.16E-04	95.83%
DTLZ1(5,15)	22.33	120	500	170.58	0.2048734	6.29E-03	95.83%
DTLZ1(50,100)	41.33	120	1500	417.13	1.0000000	4.03E-02	91.67%
DTLZ1(100,175)	128.32	120	1500	1500.00	1.0000000	7.27E+00	0.00%

Table 10: Results using SC-PSO-3P.

5.3 Constrained optimization

In this section we consider the global optimization problem with restrictions:

$$\begin{array}{lll} \text{Minimize:} & h(x) & \\ \text{subject to:} & f_i(x) \leq 0 & i = 1,2,\ldots,m \\ & t_j(x) = 0 & j = 1,2,\ldots,k \\ & x \in S = [L,U], & \end{array} \tag{3}$$

where:

- $[L,U] = \{x = (x_1, x_2, \ldots, x_n) | l_i \leq x_i \leq u_i\} \subset \mathbb{R}^n$,
- the feasible region (D) is defined as:

$$D = \{x \in S, f_i(x) \leq 0,\ i = 1,2,\ldots,m,\ t_j(x) = 0,\ j = 1,2,\ldots,k\}.$$

The problem given in (3) can be transformed into the problem (4) given by:

$$\begin{array}{rrl} \text{Minimize}_{x\in D}: & F(x) = & (h_1(x), h_2(x), h_3(x)) \\ \text{where:} & h_1(x) = & h(x) \\ & h_2(x) = & P_1 \sum_{i=1}^{m} \frac{c_i(x)}{c(x)+\epsilon} \\ & h_3(x) = & P_2 \sum_{j=1}^{k} d_j \end{array} \tag{4}$$

and:

- $c_i(x) = \max\{0, f_i(x)\}, i = 1, 2, \ldots, m.$
- $c(x) = \max_{i=1,2,\ldots,m}\{c_i(x)\}.$
- $d_j(x) = \max\{0, |t_j(x)| - \delta\}.j = 1, 2, \ldots, k.$
- $\epsilon, \delta > 0, P_i \gg 0, i = 1, 2.$

Problem(dim)	Mean Time	Max Eff.	Mean Eff.
G3(12)	32.45	0.999982	0.995865
G5(4)	32.03	0.999971	0.989064

Table 11: Results with 120 particles, 500 iterations and 24 runs.

Table 11 shows the average time (Mean Time) given in seconds per run, the maximum efficiency obtained (Eff Max) and the average efficiency (Mean Eff) in 24 runs using 120 particles and 500 iterations per run.

Function(n)	Type of h	Ratio* ρ %
G3(12)	Polynomial	0.0000
G5(4)	Cubic	0.0000

Table 12: Summary ([11]), $*\rho = \frac{\text{number of solution}_{\in D}}{\text{number of solution}_{\in S}}$.

Table 12 shows the ratio ρ for problems G3 and G5 . [11] presents a summary of some characteristics of these problems.

In Table 13 information on the average time as per run in seconds (Time) and the maximum efficiency (Max Eff) obtained in 24 runs using 3 particles and 500 iterations per run is presented.

Problem	Time(sec)	Max Eff.
G3	2.22	0.995847429
G5	2.43	0.997725422

Table 13: Maximum efficiency achieved with 3 particles and 500 iterations, for 24 runs.

6 Conclusions and further research

In this work, a novel criterion called System of Convergence (SC) is proposed, which was implemented using a novel PSO based algorithm with three phases: stabilization, generation with broad-ranging exploration and generation with in-depth exploration. This algorithm was tested in a set of benchmark instances, monoobjective as well as multiobjective, with and without restrictions. The empirical evidence shows that SC-PSO-3P is very efficient in both small and big instances. In very big instances we are reporting results never found before in the literature.

As an important remark, a deeper study concerning the boundary conditions of the parameters a and α of SC, as well as their implementation with an algorithm different from PSO must be performed.

It is also necessary to conduct a further study about the topology generated by SC. It was observed that in all the cases studied, SC can reach the global optimum or being very close to it with shorter iteration times and less iterations, as well as the ability to jump deep valleys. Worth mentioning that in a later work a more extensive study will be conducted for each types of optimization models.

Acknowledgements

The author would like to thank to D.Sto., and to P.V.Gpe. for their inspiration, and his family: Ma, Ser, Mon, Chema, and to his Flaquita for all their support.

References

[1] Coello Coello, C.A.; Lamont, G.B.; Veldhuizen, D.A. van (2007) *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2d. Ed. Springer, New York.

[2] de-los-Cobos-Silva, S.G. (2014) "SC: sistema de convergencia: Condiciones y propiedades", Titular: Universidad Autónoma Metropolitana, Registro (copyright) 03-2014-030511365700-01, 07 de marzo de 2014, Instituto Nacional del Derecho de Autor, Secretaria de Educación Publica, México.

[3] Deb, K.; Thiele, L.; Laumanns, M.; Zitzler, E. (2002) "Scalable multi-objective optimization test problems", in: *Proceedings of the Congress on Evolutionary Computation* (CEC 2002), IEEE Press, Honolulu: 825–830.

[4] Dubois, D.; Prade, H. (1978) "Operations on fuzzy numbers", *International Journal of Systems Science* **9**(6): 613–626.

[5] Gavana, A. (2007?) "Global optimization benchmarks and AMPGO. Test functions", in: `http://infinity77.net/global_optimization/`

[6] Hedar, A.R. (2007?) "Global optimization test problems", in: `http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/go.htm`

[7] Kalami Heris, S.M. (s.f.) `http://www.kalami.ir`

[8] Kennedy, J.; Eberhart, R.C.; Shi, Y. (2001) *Swarm Intelligence*. Morgan Kaufmann, San Francisco.

[9] Liang, J.J.; Runarsson, T.P.; Mezura-Montes, E.; Clerc, M.; Suganthan, P.N.; Coello Coello, C.A.; Deb, K. (2006) "Problem definitions and evaluation criteria for the CEC 2006, special session on constrained real-parameter optimization", Technical Report, *IEEE Congress on Evolutionary Computation*, 24 pp.

[10] Liou, T.S.; Wang, M.J.J. (1992) "Ranking fuzzy numbers with integral value", *Fuzzy Sets and Systems* **50**(3): 247–255..

[11] Michalewicz, Z.; Fogel, D.B. (1998) *How to Solve It: Modern Heuristics*, 2nd. Edition. Springer, Berlin.

[12] Surjanovic, S.; Bingham, D. (2013) "Virtual library of simulation experiments: test functions and datasets", Simon Fraser University, in: `http://www.sfu.ca/~ssurjano/optimization.html`

[13] Zitzler, E.; Deb, K.; Thiele, L. (2000) "Comparison of multiobjective evolutionary algorithm: Empirical results", *Evolutionary Computation* **8**(2): 173–195.