Fully Informed Swarm Optimization Algorithms: Basic Concepts, Variants and Experimental Evaluation

Szymon Łukasik[1,2]

[1]Department of Automatic Control and IT
Cracow University of Technology
ul. Warszawska 24, 31-155 Krakow, Poland
Email: {szymonl,pkowal}@pk.edu.pl

Piotr A. Kowalski[1,2]

[2]Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6, 01-447 Warsaw, Poland
Email: {slukasik,pakowal}@ibspan.waw.pl

***Abstract*—Particle swarm optimization constitutes currently one of the most important nature-inspired metaheuristics, used successfully for both combinatorial and continuous problems. Its popularity has stimulated the emergence of various variants of swarm-inspired techniques, based in part on the concept of pairwise communication of numerous swarm members solving optimization problem in hand. This paper overviews some examples of such techniques, namely Fully Informed Particle Swarm Optimization (FIPSO), Firefly Algorithm (FA) and Glowworm Swarm Optimization (GSO). It underlines similarities and differences among them and studies their practical features. Performance of those algorithms is also evaluated over a set of benchmark instances. Finally, some concluding remarks regarding the choice of suitable problem-oriented optimization technique along with areas of possible improvements are given as well.**

I. Introduction

PARTICLE Swarm Optimization introduced by Kennedy, Eberhart and Shi in 1995 [1] is at the moment one of the most noteworthy nature-inspired metaheuristics used for variety of tasks, both in science and engineering. It was induced by the observation of flocking and schooling patterns of birds and fish. An idea to represent each solution of the optimization problem at-hand as a member of the virtual swarm – communicating with others and modifying its position under the influence of best individuals – proved to be extremely successful. The degree of this success can be represented by the significant amount of contributions employing PSO in real-world problems e.g. in data analysis [2], resource allocation [3] etc. It can be also quantified through a number of related algorithms, based on the idea of intelligent swarms. One of recent examples of such include: Quantum-behaved Particle Swarm Optimization [4] and Multi-Swarm PSO [5].

The general goal of continuous optimization is to find x^* which satisfies:

$$f(x^*) = \min_{x \in S} f(x), \tag{1}$$

where $S \subset R^N$, and $f(x)$ constitutes solution's x cost function value. Therefore actual task of the optimizer is to find argument minimizing f.

Initial PSO algorithm's behavior was built on the assumption that each individual member of the swarm, i.e. solution of the optimization problem (1), changes its velocity vector in the consecutive algorithm's iteration as a result of the influence of two specific solutions: the best one found so far by the swarm and the top solution identified by this individual. Fully Informed Particle Swarm Optimization (FIPSO) presented first by Mendes, Kennedy and Neves [6] constitute a modification of this approach. In the most general variant of FIPSO velocity update is constructed using weighted average position of all swarm members. Creators of the algorithm considered however alternative communication topologies, e.g. ring or cluster, as well as different schemes of assigning weights to prioritize individuals' inputs. Firefly Algorithm (FA) created by Xin She Yang in 2008 is constructed on similar assumptions [7]. The position of swarm member x_m within feasible solution space S is determined by all other individuals' fitness – better solutions will attract those which are worse, in the sense of selected cost function f value [8]. Glowworm Swarm Optimization (GSO) developed by Krishnanand and Ghose [9] exhibits similar behavior however swarm member is attracted here by its better-performing neighbors found only within given radius. Considering those similarities FA and GSO like FIPSO can be perceived as most representative members of broader family of techniques named here Fully Informed Particle Swarm algorithms. It can be characterized by building solution space exploration process on exchanging information between swarm members regarding local fitness landscape and modifying their position accordingly. For other examples of such methods one can refer to [10], [11], [12].

The goal of this contribution is to provide synthetic comparative perspective on major Fully Informed Particle Swarm algorithms, introduced above, both on conceptual and performance-based grounds. It is organized as follows. First the description of all techniques studied here in their basic variants is provided, along with similarities between them. It also contains brief discussion of selected technical aspects, examples of applications and possible modifications. Then

the results of performed comparative experimental studies are given, both in the context of optimization performance, algorithms' convergence and computational demands. Finally general remarks concerning the choice of suitable problem-oriented optimization techniques and planned further studies are under consideration.

II. Fully-Informed Swarm Algorithms

First let us introduce the notation which will be used in the following subsections. To solve the optimization problem (1) the swarm, consisting of M members will be used. It will be represented by a set of N-dimensional vectors – equivalent to individuals' positions – within the iteration k denoted by:

$$x_1(k), x_2(k), ..., x_M(k). \tag{2}$$

Euclidean distance between two swarm members, indexed p and q is denoted here by $d(x_p, x_q)$. The best position found by given swarm member m prior to iteration k is given by $x_m(k)^*$ with cost function value $f(x_m(k)^*)$. At the same time:

$$x(k)^* = \arg \min_{m=1,...,M} f(x_m(k)), \tag{3}$$

corresponds to the best solution found by the algorithm in its k iterations, with $f(x(k)^*)$ representing its related cost function value. Swarm optimization algorithms, based in particular on PSO paradigm, employ frequently a concept of individual's m velocity, denoted here in iteration k as $v_m(k)$. It is used to update particles' positions and can be initialized randomly within given bounds.

The following part of the paper will provide comprehensive description of Fully Informed Particle Swarm algorithms, referring to the notation given above.

A. Fully-Informed Particle Swarm Optimization (FIPSO)

Fully Informed Particle Swarm Optimization constitutes one of heuristic algorithms derived from the basic PSO paradigm. The idea of using information from a group of particle's K_m neighbors, rather than just the best one – as in traditional canonical PSO – was first proposed by Suganthan in 1999 [13]. It was also included in complete Fully Informed PSO procedure suggested by Mendes, Kennedy and Neves. In each iteration of the algorithm particle's position $x_m(k)$ is updated by moving it iteratively along vector $v_m(k)$ with the coordinates $n = 1, ..., N$ adjusted as follows:

$$v_{mn}(k+1) = \chi \left[v_{mn}(k) + \frac{1}{K_m} \sum_{j=1}^{K_m} U(0,\varphi)(x_{N_m(j)_n}(k)^* - x_{mn}(k)) \right], \tag{4}$$

with χ known as a constriction factor, whereas $U(0,p)$ corresponds to the uniformly distributed random number in $(0,p)$, φ constitutes an acceleration coefficient, and finally, $N_m(j)$ is a function which returns the index of j-th nearest neighbor of particle m. Complete FIPSO procedure was provided below in the form of pseudocode (Algorithm 1).

Algorithm 1 Fully Informed Particle Swarm Optimization algorithm

$k \leftarrow 1$ {initialization}
for $m = 1$ to M **do**
  Generate_Solution($x_m(k)$)
  Initialize_Velocity($v_m(0)$)
  $f(x_m(0)^*) \leftarrow \infty$
end for
{main loop}
repeat
  {evaluate and update best solutions}
  for $m = 1$ to M **do**
    $f(x_m(k)) \leftarrow$ Evaluate_quality($x_m(k)$)
    if $f(x_m(k)) < f(x_m(k-1)^*)$ **then**
      $x_m(k)^* \leftarrow x_m(k)$
    else
      $x_m(k)^* \leftarrow x_m(k-1)^*$
    end if
    if $f(x_m(k)) < f(x(k)^*)$ **then**
      $x(k)^* \leftarrow x_m(k)$
    else
      $x(k)^* \leftarrow x(k-1)^*$
    end if
  end for
  for $m = 1$ to M **do**
    for $n = 1$ to N **do**
      $c_{mn}(k) \leftarrow 0$
      for all K_m nearest neighbors (index p) of m **do**
        $c_{mn}(k) \leftarrow c_{mn}(k) + (U(0,\varphi)(x_{pn}(k)^* - x_{mn}(k))$
      end for
      $v_{mn}(k) \leftarrow \chi * [v_{mn}(k-1) + 1/K_m * c_{mn}(k)]$
      $x_{mn}(k+1) \leftarrow x_{mn}(k) + v_{mn}(k)$
    end for
  end for
  $stop\_condition \leftarrow Check\_stop\_condition()$
  $k \leftarrow k + 1$
until $stop\_condition$ = **false**
return $f(x(k)^*)$, $x(k)^*$, k

One of the findings of initial study on FIPSO was that increasing the size of the "informing" neighborhood seems to deteriorate the performance of the swarm. FIPSO with a fully connected topology, i.e., when each particle has all the particles in the swarm as its neighbors, shows a particularly bad performance in comparison with the one attained with other topologies, e.g. ring or square. In addition to that FIPSO convergence was thoroughly studied in [14]. Authors observe there that for highly connected topologies, the particles explore a region close to the centroid of the swarm. It may bring positive results for some specific functions however the algorithm in that case is prone to becoming trapped in local minima. The algorithm in the form introduced above was successfully applied for engineering problems like power

systems optimization [15]. It was also used as a starting point for other similar approaches [16], [17] as well as a component of hybrid algorithms [18]. Here we consider most general FIPSO with fully-connected particles to study its performance when referencing it to two other more recent approaches.

B. Firefly Algorithm (FA)

Firefly Algorithm developed by Xin-She Yang [7] is inspired by mechanisms of firefly communication via luminescent flashes. This swarm intelligence optimization technique is based on the assumption that solution of an optimization problem can be perceived as agent (firefly) which "glows" proportionally to its quality in a considered problem setting. Consequently each brighter firefly attracts its partners (regardless of their sex), which makes the search space being explored more efficiently [8].

Each firefly has its distinctive attractiveness β which implies how strong it attracts other members of the swarm. For attractiveness in FA an exponential function of the distance $r_j = d(x_m, x_j)$ to the chosen firefly j is used:

$$\beta = \beta_0 e^{-\gamma r_j} \tag{5}$$

where β_0 and γ are predetermined algorithm parameters: maximum attractiveness value and absorption coefficient, respectively. Every member of the swarm is also characterized by its light intensity I_m which can be directly expressed as a inverse of a cost function $f(x_m)$.

To effectively explore considered search space S it is assumed that each firefly m is changing its position iteratively taking into account two factors: attractiveness of other swarm members with higher light intensity i.e. $I_j > I_m, \forall j = 1, ...M, j \neq m$ – which is varying across distance – and a fixed random step vector $U(min, max)$. It should be noted as well that if no brighter firefly can be found only such randomized step is being used [8].

Algorithm 2 presents generic Firefly Algorithm which includes all aforementioned elements. For a recent overview of FA modifications, variants and applications one can refer to [19]. We employ here standard FA algorithm with uniform random number generator and scaling factor related to search space size S [8].

Algorithm 2 Firefly Algorithm

$k \leftarrow 1$ {initialization}
for $m = 1$ to M **do**
  Generate_Solution($x_m(k)$)
end for
$f(x(0)^*) \leftarrow \infty$
{main loop}
repeat
  {evaluate and update best solution}
  for $m = 1$ to M **do**
    $f(x_m(k)) \leftarrow$ Evaluate_quality($x_m(k)$)
    if $f(x_m(k)) < f(x(k-1)^*)$ **then**
      $x(k)^* \leftarrow x_m(k)$
    else
      $x(k)^* \leftarrow x(k-1)^*$
    end if
  end for
  for $m = 1$ to M **do**
    for $p = 1$ to M **do**
      if $f(x_m(k)) < f(x_p(k))$ **then**
        $r_p \leftarrow Calculate\_Distance(x_m(k), x_p(k))$
        $\beta \leftarrow \beta_0 e^{-\gamma r_j}$
        for $n = 1$ to N **do**
          $x_{mn}(k) \leftarrow (1-\beta)x_{mn}(k) + \beta x_{pn}(k) + U_n(min, max)$
        end for
      end if
    end for
  end for
  {best moves randomly}
  for $m = 1$ to M **do**
    if $Was\_Moved(x_m(k))$ = **false then**
      for $n = 1$ to N **do**
        $x_{mn}(k) \leftarrow x_{mn}(k) + U_n(min, max)$
      end for
    end if
  end for
  $stop\_condition \leftarrow Check\_stop\_condition()$
  $k \leftarrow k + 1$
until $stop\_condition$ = **false**
return $f(x(k)^*)$, $x(k)^*$, k

C. Glowworm Swarm Optimization (GSO)

Glowworm Swarm Optimization is another optimization strategy which was stimulated by the observation of fireflies' social behavior. In contrast to Firefly Algorithm agents in GSO depend only on information available in their strict neighborhood to make decisions [9]. What is more GSO uses an adaptive neighborhood range in order to successfully deal with multimodal functions landscapes. Both luciferin quantity $\iota_m(k)$, which predetermines the probability of individual's movement, and neighborhood radius $r_m(k)$ are updated on per-iteration basis. It is realized using the following formulas:

$$\iota_m(k) = (1-\rho)\iota_m(k-1) + \gamma f(x_m(k))^{-1}, \tag{6}$$

$$r_m(k+1) = \min\{r_s, \max\{0, r_m(k) + \beta(N_{set} - |N_m(k)|)\}\}, \tag{7}$$

with ρ representing lucifirin decay parameter, γ constituting luciferin enhancement constant, r_s - maximum sensor range, N_{set} - parameter controlling number of neighbors and finally, $N_m(k)$ denoting a set of neighbors of $x_m(k)$ located within radius $r_m(k)$:

$$N_m(k) = \{x_j(k) : d(x_m(k), x_j(k)) < r_m(k) : \iota_m(k) < \iota_j(k)\}. \tag{8}$$

Probability of glowworm movement towards one of other individuals in the neighborhood $N_m(k)$ is proportional to its

luciferin quantity, related to the sum of luciferin values for all neighbors found in $N_m(k)$. It is denoted for all neighbors by a vector $p_m(k)$. The Algorithm 3 presents plain description of GSO procedure including most important technical details.

Algorithm 3 Glowworm Swarm Optimization algorithm

$k \leftarrow 1$ {initialization}
$f(x(k)^*) \leftarrow \infty$
for $m = 1$ to M **do**
  Generate_Solution($x_m(k)$)
  $f(x_m(k)) \leftarrow$ Evaluate_quality($x_m(k)$)
  $\iota_m(0) \leftarrow \iota_0$
  $r_m(0) \leftarrow r_0$
end for
{main loop}
repeat
  {update luciferin quantity}
  for $m = 1$ to M **do**
    $\iota_m(k) \leftarrow (1-\rho)\iota_m(k-1) + \gamma f(x_m(k))^{-1}$
  end for
  {move glowworms}
  for $m = 1$ to M **do**
    $N_m(k) \leftarrow Find\_Neighborhood(x_m(k))$
    {sum selection probalities for all p neighbors in $N_m(k)$}
    $P_{sum} \leftarrow sum(\iota_p(k) - \iota_m(k))$
    for all $x_j(k)$ in ($N_m(k)$) **do**
      $p_{mj}(k) \leftarrow (\iota_j(k) - \iota_m(k))/P_{sum}$
    end for
    $q \leftarrow Select\_neighbor\_index(p_m(k))$
    {move selected}
    $x_m(k+1) \leftarrow x_m(k) + s(x_q(k) - x_m(k))$
      $/(\|x_q(k) - x_m(k)\|)$
    $r_m(k+1) \leftarrow \min\{$
    $r_s, \max\{0, r_m(k) + \beta(N_{set} - |N_m(k)|)\}\}$
  end for
  for $m = 1$ to M **do**
    if $Was\_Moved(x_m(k)$ = **false then**
      $x_m(k+1) \leftarrow x_m(k)$
    end if
    $f(x_m(k)) \leftarrow$ Evaluate_quality($x_m(k)$)
    if $f(x_m(k)) < f(x(k)^*)$ **then**
      $x(k)^* \leftarrow x_m(k)$
    else
      $x(k)^* \leftarrow x(k-1)^*$
    end if
  end for
  $stop\_condition \leftarrow Check\_stop\_condition()$
  $k \leftarrow k + 1$
until $stop\_condition$ = **false**
return $f(x(k)^*)$, $x(k)^*$, k

GSO like FA attracted much attention resulting in several contributions improving the general scheme of the algorithm [20], studying theoretical properties [21] or employing GSO for real-life problems [22]. Here, for comparative studies we utilize standard GSO developed by Krishnanand and Ghose.

D. Summary

Techniques covered in this Section are employing mutual information exchange between all members of the swarm or its selected groups (depending on precise variant and parameter values). It is used for modifying swarm member position (GSO and FA) or its velocity vector (FIPSO).

In case of FIPSO individual's movement is influenced by a set of best solutions obtained by other swarm members. For GSO and FA latest position of other individuals can be used, however it must be better than the one of solution currently under consideration.

Algorithms studied here employ a randomization component, either in the form of implicit randomized movement (FA), by random selection of informing agent (GSO) or determining strength of each neighbor's influence (FIPSO). All mechanisms tend to improve algorithms' abilities to escape local minima. In this aspect additional dynamics contained within FIPSO technique could be extremely beneficial.

Every technique studied here possesses significant number of parameters, with GSO being most parameter-rich and FIPSO parameter-free one. All required parameters are listed in Table I. For most of them some guidelines have been already worked out in the related contributions - they are listed in the table as well.

TABLE I
ALGORITHMS' PARAMETERS AND THEIR SUGGESTED VALUES

Algorithm	Parameter	Suggested value/range	Source
FIPSO	M	[20,50]	[23]
	χ	0.72984	[24]
	φ	4.1	[24]
	K_m	[3,5]	[6]
FA	M	[20,50]	[25]
	α	[0.1,0.2]	[25]
	β_0	1	[8]
	γ	[1,30]	[8], [25]
GSO	M	[10,500]	[21]
	ρ	0.4	[21]
	γ	0.6	[21]
	β	0.08	[21]
	N_{set}	5	[21]
	s	0.03	[21]
	ι_0	5	[21]
	r_s	use pilot runs	[21]
	r_0	$r_0 = r_s$	[21]

As for computational complexity of algorithms studied here it is in all cases significant. With regards to swarm size M and iteration number K it can be expressed by notation $O(KM^2)$. The actual relative time needed for execution of all algorithms as other performance measures will be studied in the following Section.

III. EXPERIMENTAL STUDIES

One of the main goals of conducted experiments was to examine dynamics of swarm's performance for all considered techniques during the optimization process. Running times

TABLE II
BENCHMARK FUNCTIONS USED FOR EXPERIMENTAL STUDIES

f	Name	Expression	Feasible bounds	N	f^*
f_1	Sphere	$f_1(x) = \sum_{i=1}^{M} z_i^2 + f_1^*$ $z = x - o$	$[-100, 100]^N$	10	-1400
f_2	Different Powers	$f_2 = \sqrt{\sum_{i=1}^{N} \lvert z_i \rvert^{2+4\frac{i-1}{N-1}}} + f_2^*$ $z = x - o$	$[-100, 100]^N$	10	-1000
f_3	Rotated Rastrigin	$f_3(x) = \sum_{i=1}^{N} (z_i^2 - 10\cos(2\pi z_i) + 10) + f_3^*$ $z = M_1 \Lambda^{10} M_2 T_{asy}^{0.2}(T_{osz}(M_1 \frac{5.12(x-o)}{100}))$	$[-100, 100]^N$	10	-300
f_4	Schwefel	$f_4(x) = 418.9829N \sum_{i=1}^{N} g(z_i) + f_4^*$ $z = \Lambda^{10}(\frac{1000(x-o)}{100}) + 4.209687462275036e + 002$ $g(z_i) = z_i \sin(\lvert z_i \rvert^{1/2})$	$[-100, 100]^N$	10	-100
f_5	Rotated Katsuura	$f_5(x) = \frac{10}{N^2} \prod_{i=1}^{N} (1 + i \sum_{j=1}^{32} \frac{\lvert 2^j z_i - round(2^j z_i) \rvert}{2^j})^{\frac{10}{N^{1.2}}} - \frac{10}{N^2} + f_5^*$ $z = M_2 \Lambda^{100}(M_1 \frac{5(x-o)}{100})$	$[-100, 100]^N$	10	200

Symbols:
$o = [o_1, o_2, ..., o_N]$ – shifted global optimum, randomly distributed in $[-80, 80]^N$,
M_1, M_2 – orthogonal (rotation) matrix generated from standard normally distributed entries by Gram-Schmidt orthonormalization.

Λ^{α} – diagonal matrix in N dimensions with the i^{th} diagonal element $\lambda_{ii} = \alpha^{\frac{i-1}{2(N-1)}}$ for $i = 1, 2, ..., N$.
T_{asy}^{β} – if $x_i > 0, x_i = x_i^{1+\beta\frac{i-1}{N-1}\sqrt{x_i}}$ for $i = 1, 2, ..., N$.
T_{osz} – for $x_i = sign(x_i)\exp(\hat{x}_i + 0.049(sin(c_1\hat{x}_i) + sin(c_2\hat{x}_i)))$ for $i = 1, 2, ..., N$.
where:
$\hat{x}_i = log(\lvert x_i \rvert)$ for $x_i \neq 0$, otherwise $\hat{x}_i = 0$,
$c_1 = 10$ if $x_i > 0$, otherwise $c_1 = 5.5$,
$c_2 = 7.9$ if $x_i > 0$, otherwise $c_2 = 3.1$.

were also carefully studied as well as final cost function values, which were additionally compared by means of statistical tests. The following subsections are covering the details of algorithms' numerical evaluation.

A. Problems and Experimental Setting

For computational experiments set of benchmark problems considered in CEC'13 competition was used [26]. Table II lists those functions along with their mathematical expressions, dimensionality and optimum values.

The experiments were conducted for fixed number of iterations $K = 10000$ $(1000 * N)$, and 30 trials for each function. Population size $M = 40$ was used for all algorithms. FIPSO was configured with fully-connected topology and parameter values as suggested in Table I. For FA we used $\alpha = 0.15$ and $\gamma = 10$ following the suggestions found in related literature. Random step size was also scaled to the size of search space S. GSO was configured with parameters given in Table I, with modified step size $s = 0.8$. For r_s we used half of maximum distance in S (that is $r_s = 315$) and for r_0 the value of 90 was selected. Both settings were established during a set of pilot runs to adopt neighborhood size properly to the domain of given optimization task.

As a performance measure mean optimization error $\overline{E}(k)$ was used (with $E(k) = |f(x(k)^*) - f^*|$) along with its standard deviation $\sigma_{E(k)}$. We have also studied mean execution time $\overline{t}$ in seconds needed for one algorithm's run.

B. Algorithms' Search Process Dynamics

First set of experiments was aimed at establishing dynamics of swarm performance in the function of execution time (iterations). For all algorithms mean optimization errors in 30 trials during 10000 iterations were reported. The results of this study for all investigated techniques are shown on Figures 1-5.

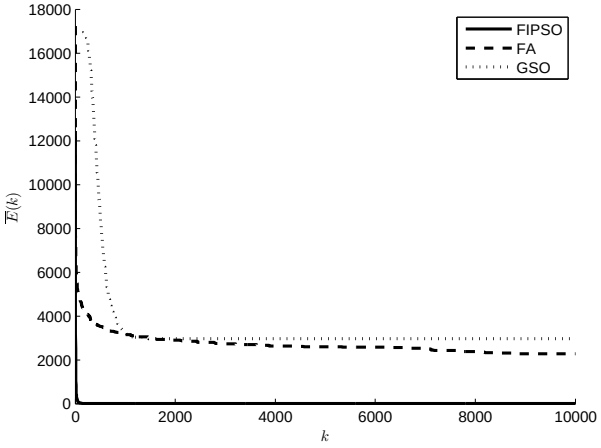

Fig. 1. Mean error values obtained within 10000 iteration of optimization process for f_1 (Sphere) function

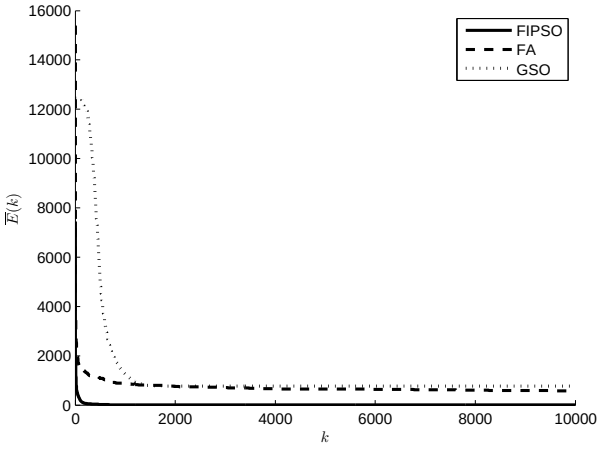

Fig. 2. Mean error values obtained within 10000 iteration of optimization process for f_2 (Different Powers) function

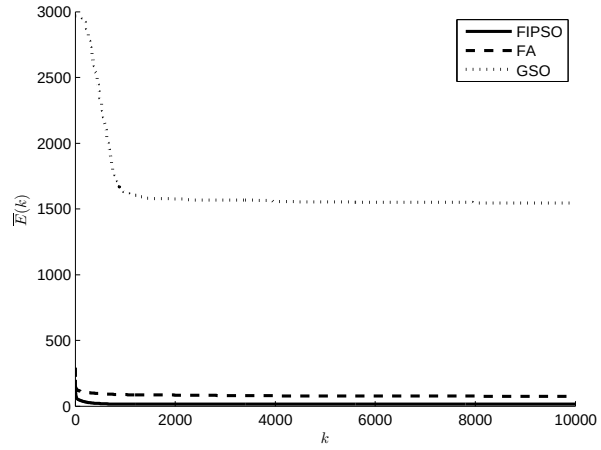

Fig. 3. Mean error values obtained within 10000 iteration of optimization process for f_3 (Rotated Rastrigin) function

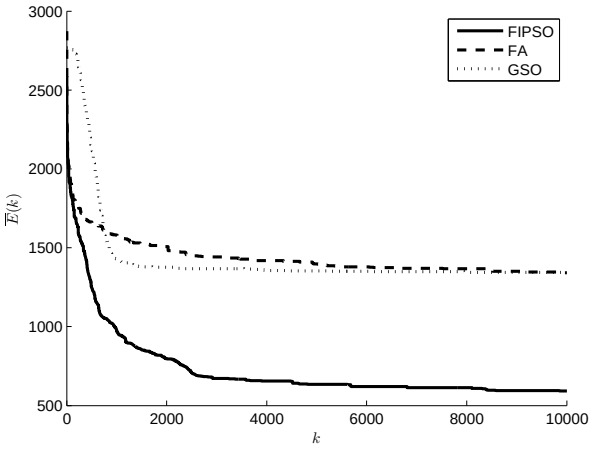

Fig. 4. Mean error values obtained within 10000 iteration of optimization process for f_4 (Schwefel) function

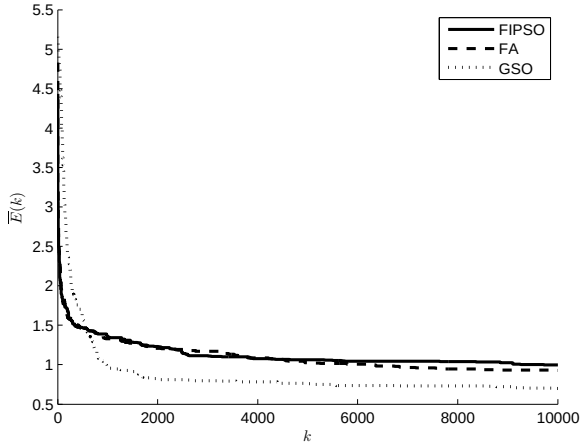

Fig. 5. Mean error values obtained within 10000 iteration of optimization process for f_5 (Rotated Katsuura) function

It can be seen that in the framework of GSO swarm members tend to find local minima and to stop exploring problem's search space afterwards. This tendency will demonstrate itself again in the next analysis. As for dynamics of FA it is also prone for being trapped in local minima. Most intense progress of error values' minimization during first optimization phase was observed for FIPSO, for all but one (f_5) function.

C. Algorithms' Performance

To compare performance of all considered algorithms we examined final cost function values obtained through a set of runs described above. In this quantitative comparison, both means, standard deviations and best obtained cost values were reported. Relation between algorithm's performance indicators was studied by means of pairwise T-tests. In most cases differences proved to be statistically meaningful at 0.95 significance level. We ranked all algorithms according to conclusive results of those tests.

Among studied algorithms FIPSO was found to be best-performing one. What is more its execution for fully-connected topology requires the least computational effort. Among two other algorithms FA was found to perform better than GSO, however here the difference – both in terms of final cost function values and required execution time – proved to be not very substantial. The most difficult problem for all the algorithms to tackle was minimization of Schwefel function. At the same time surprisingly Sphere function proved to be problematic for FA and GSO algorithms.

IV. Conclusion

The paper examined practical features of a set of important nature-inspired metaheuristics: Fully Informed Particle Swarm Optimization, Firefly Algorithm and Glowwworm Swarm Optimization. They were for the first time considered within the same methodological context, with their performance being examined on a set of benchmark functions.

We found that when using the most basic variants of aforementioned techniques Fully Informed Particle Swarm Optimization proved to be most effective and least-computationally expensive. At the same time Firefly Algorithm and Glowworm Swarm Optimization were found to be similar, both in terms of performance and computational resources' requirements. From our observations GSO is a very powerful technique for discovering local minima especially when it employs large swarm population. To use it as a tool of global optimization however one should complement it by randomized local search algorithm (e.g. Simulated Annealing [27]) or other procedure which would allow swarm members to escape from cost function valleys. In case of FA we see a suitable choice of random step generation method (e.g. Lévy Flight [28]) as a crucial element for successful applications in the field of continuous optimization.

Finally it should be noted that performed experiments do not include recent modifications of all analyzed optimization strategies, described briefly in Section 2, and more specific benchmarks. Their results however can be used as a baseline for enriching this study by including those aspects. It is planned within forthcoming follow-up contribution [29].

Acknowledgment

First author is thankful to Marco A. Montes de Oca for helpful suggestions and inspiration provided during his stay at Université Libre de Bruxelles.

References

[1] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Proceedings of IEEE International Conference on Neural Networks, vol. IV*, 1995. doi: 10.1109/ICNN.1995.488968 pp. 1942–1948. [Online]. Available: http://dx.doi.org/10.1109/ICNN.1995.488968

[2] T. Cura, "A particle swarm optimization approach to clustering," *Expert Systems with Applications*, vol. 39, no. 1, pp. 1582–1588, 2012. doi: 10.1016/j.eswa.2011.07.123. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2011.07.123

[3] Y. Morsly, N. Aouf, M. Djouadi, and M. Richardson, "Particle Swarm Optimization Inspired Probability Algorithm for Optimal Camera Network Placement," *IEEE Sensors Journal*, vol. 12, no. 5, pp. 1402–1412, 2012. doi: 10.1109/JSEN.2011.2170833. [Online]. Available: http://dx.doi.org/10.1109/JSEN.2011.2170833

TABLE III
ALGORITHMS' PERFORMANCE COMPARISON

Function	FIPSO				FA				GSO				Pairwise T-test results (order)
	$\overline{E}(K)$	$\sigma_{E(K)}$	$\overline{t}$	$\min E(K)$	$\overline{E}(K)$	$\sigma_{E(K)}$	$\overline{t}$	$\min E(K)$	$\overline{E}(K)$	$\sigma_{E(K)}$	$\overline{t}$	$\min E(K)$	
f_1	0.00	0.00	26.33	0.00	2273.98	362.74	194.71	1516.96	2972.20	1715.90	233.95	946.33	1.FIPSO, 2.FA, 3.GSO
f_2	0.01	0.01	28.48	0.00	575.11	105.46	193.26	341.09	774.01	501.62	217.58	178.25	1.FIPSO, 2.FA, 3.GSO
f_3	14.80	5.20	30.84	4.00	73.02	8.20	196.82	51.98	80.04	26.55	238.45	21.24	1.FIPSO. 2.FA,GSO
f_4	592.85	220.73	31.87	261.83	1341.30	124.44	234.44	937.65	1343.50	339.56	194.29	641.79	1.FIPSO, 2.FA,GSO
f_5	1.00	0.14	32.55	0.68	0.93	0.14	249.83	0.67	0.70	0.22	212.90	0.31	1.GSO, 2.FIPSO, 3.FA

[4] W. Fang, J. Sun, Y. Ding, X. X. Wu, and W. Xu, "A Review of Quantum-behaved Particle Swarm Optimization," *IETE Technical Review*, vol. 27, pp. 336–348, 2010.

[5] A. Röhler and S. Chen, "Multi-swarm hybrid for multi-modal optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2012. doi: 10.1109/CEC.2012.6256566 pp. 1759–1766. [Online]. Available: http://dx.doi.org/10.1109/CEC.2012.6256566

[6] R. Mendes, J. Kennedy, and J. Neves, "The Fully Informed Particle Swarm: Simpler, Maybe Better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 204–210, 2004. doi: 10.1109/TEVC.2004.826074. [Online]. Available: http://dx.doi.org/10.1109/TEVC.2004.826074

[7] X. Yang, *Nature-Inspired Metaheuristic Algorithms*. Frome: Luniver Press, 2008.

[8] S. Łukasik and S. Żak, "Firefly Algorithm for Continuous Constrained Optimization," *Lecture Notes in Artificial Intelligence*, vol. 5796, pp. 97–106, 2009. doi: 10.1007/978-3-642-04441-0_8. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04441-0_8

[9] K. Krishnanand and D. Ghose, "Glowworm Swarm Optimization for Simultaneous Capture of Multiple Local Optima of Multimodal Functions," *Swarm Intelligence*, vol. 3, no. 2, pp. 87–124, 2008. doi: 10.1007/s11721-008-0021-5. [Online]. Available: http://dx.doi.org/10.1007/s11721-008-0021-5

[10] T. Havens, C. Spain, N. Salmon, and J. Keller, "Roach infestation optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium 2008*, 2008. doi: 10.1109/SIS.2008.4668317 pp. 1–7. [Online]. Available: http://dx.doi.org/10.1109/SIS.2008.4668317

[11] D. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim, and M. Zaidi, "The Bees Algorithm - A Novel Tool for Complex Optimisation Problems," in *Proceedings of IPROMS 2006 Conference*, 2006, pp. 454–461.

[12] X. Yang and A. Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Engineering Computations*, vol. 29, no. 5, pp. 464–483, 2012. doi: 10.1108/02644401211235834. [Online]. Available: http://dx.doi.org/10.1108/02644401211235834

[13] P. Suganthan, "Particle swarm optimiser with neighbourhood operator," in *Proceedings of the 1999 Congress on Evolutionary Computation, Vol. 3*, vol. 3, 1999. doi: 10.1109/CEC.1999.785514 pp. 1958–1962. [Online]. Available: http://dx.doi.org/10.1109/CEC.1999.785514

[14] M. A. M. de Oca and T. Stutzle, "Convergence Behavior of the Fully Informed Particle Swarm Optimization Algorithm," in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, 2008. doi: 10.1145/1389095.1389106 pp. 71–78. [Online]. Available: http://dx.doi.org/10.1145/1389095.1389106

[15] H. Tehzeeb-Ul-Hassan, R. Zafar, S. Mohsin, and O. Lateef, "Reduction in power transmission loss using fully informed particle swarm optimization," *International Journal of Electrical Power & Energy Systems*, vol. 43, no. 1, pp. 364 – 368, 2012. doi: 10.1016/j.ijepes.2012.05.028. [Online]. Available: http://dx.doi.org/10.1016/j.ijepes.2012.05.028

[16] L. Hongliang, E. Howely, and J. Duggan, "Particle Swarm Optimisation with Gradually Increasing Directed Neighbourhoods," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, 2011. doi: 10.1145/2001576.2001582 pp. 29–36. [Online]. Available: http://dx.doi.org/10.1145/2001576.2001582

[17] M. Setayesh, M. Zhang, and M. Johnston, "Effects of Static and Dynamic Topologies in Particle Swarm Optimisation for Edge Detection in Noisy Images," in *Proceedings of the 2012 IEEE Congress on Evolutionary Computation*, 2012. doi: 10.1109/CEC.2012.6256104 pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/CEC.2012.6256104

[18] M. A. M. de Oca, T. Stutzle, M. Birattari, and M. Dorigo, "Frankenstein's PSO: A composite particle swarm optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1120–1132, 2009. doi: 10.1109/TEVC.2009.2021465. [Online]. Available: http://dx.doi.org/10.1109/TEVC.2009.2021465

[19] X. Yang and X. He, "Firefly algorithm: recent advances and applications," *International Journal of Swarm Intelligence*, vol. 1, no. 1, pp. 36 – 50, 2013. doi: 10.1504/IJSI.2013.055801. [Online]. Available: http://dx.doi.org/10.1504/IJSI.2013.055801

[20] P. Oramus, "Improvements to Glowworm Swarm Optimization algorithm," *Computer Science*, vol. 11, pp. 7–20, 2010. doi: 10.7494/csci.2010.11.0.7. [Online]. Available: http://dx.doi.org/10.7494/csci.2010.11.0.7

[21] K. Krishnanand and D. Ghose, "Theoretical Foundations for Rendezvous of Glowworm-inspired Agent Swarms at Multiple Locations," *Robotics and Autonomous Systems*, vol. 56, no. 7, pp. 549–569, 2008. doi: 10.1016/j.robot.2007.11.003. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2007.11.003

[22] A. Karegowda and M. Prasad, "A Survey of Applications of Glowworm Swarm Optimization Algorithm," in *IJCA Proceedings of the International Conference on Computing and Information Technology 2013*, 2013, pp. 39–42.

[23] L. Zhang, H. Yu, and S. Hu, "Optimal choice of parameters for particle swarm optimization," *Journal of Zhejiang University Science A*, vol. 6, no. 6, pp. 528–534, 2005. doi: 10.1007/BF02841760. [Online]. Available: http://dx.doi.org/10.1007/BF02841760

[24] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002. doi: 10.1109/4235.985692. [Online]. Available: http://dx.doi.org/10.1109/4235.985692

[25] M. Yuan-bin, M. Yan-zhui, and Z. Qiao-yan, "Optimal Choice of Parameters for Firefly Algorithm," in *Proceedings of the Fourth International Conference on Digital Manufacturing and Automation (ICDMA)*, 2013. doi: 10.1109/ICDMA.2013.210 pp. 887–892. [Online]. Available: http://dx.doi.org/10.1109/ICDMA.2013.210

[26] J. Liang, B. Qu, P. Suganthan, and A. Hernandez-Diaz, "Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session and Competition on Real-Parameter Optimization," 2013, technical Report 201212, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore.

[27] Y. Zhou, G. Zhou, and J. Zhang, "A Hybrid Glowworm Swarm Optimization Algorithm for Constrained Engineering Design Problems," *Applied Mathematics and Information Sciences*, vol. 7, no. 1, pp. 379–388, 2013.

[28] X. Yang, "Firefly Algorithm, Lévy Flights and Global Optimization," in *Research and Development in Intelligent Systems XXVI*, M. Bramer, R. Ellis, and M. Petridis, Eds. Springer London, 2010, pp. 209–218. [Online]. Available: http://dx.doi.org/10.1007/978-1-84882-983-1_15

[29] S. Łukasik and P. Kowalski, "Reviewing Fully Informed Swarm Optimization Algorithms," 2014, in preparation.