Computer Science • Vol. 11 • 2010

Piotr Oramus[*]

# IMPROVEMENTS TO GLOWWORM SWARM OPTIMIZATION ALGORITHM

*Glowworm Swarm Optimization algorithm is applied for the simultaneous capture of multiple optima of multimodal functions. The algorithm uses an ensemble of agents, which scan the search space and exchange information concerning a fitness of their current position. The fitness is represented by a level of a luminescent quantity called* luciferin. *An agent moves in direction of randomly chosen neighbour, which broadcasts higher value of the* luciferin. *Unfortunately, in the absence of neighbours, the agent does not move at all. This is an unwelcome feature, because it diminishes the performance of the algorithm. Additionally, in the case of parallel processing, this feature can lead to unbalanced loads. This paper presents simple modifications of the original algorithm, which improve performance of the algorithm by limiting situations, in which the agent cannot move. The paper provides results of comparison of an original and modified algorithms calculated for several multimodal test functions.*

**Keywords:** *Swarm intelligence, Glowworm Swarm Optimization, Multimodal function optimization*

## ULEPSZENIA ALGORYTMU GLOWWORM SWARM OPTIMIZATION

*Algorytm Glowworm Swarm Optimization jest stosowany do równoczesnego odnajdywania wielu optimów funkcji multimodalnych. Algorytm używa zespołu agentów przeszukujących przestrzeń poszukiwań i wymieniających się informacjami o wartości funkcji przystosowania w danym położeniu. Funkcja przystosowania jest reprezentowana przez poziom emitującego światło pigmentu – lucyferyny. Agenci poruszają się w kierunku losowo wybranego sąsiada, który rozgłasza wyższą wartość poziomu lucyferyny. Niestety w przypadku braku sąsiadów agent nie porusza się wcale. Stanowi to niepożądaną cechę algorytmu ograniczającą jego wydajność. W przypadku przetwarzania równoległego cecha ta może prowadzić do niezrównoważenia obciążenia. Praca ta przedstawia proste modyfikacje oryginalnego algorytmu zwiększające jego wydajność poprzez ograniczanie liczby takich sytuacji, w których agent nie może się poruszyć. Przedstawione zostały wyniki porównania pracy oryginalnego i zmodyfikowanych algorytmów dla kilku funkcji testowych.*

**Słowa kluczowe:** *inteligencja roju, optymalizacja*

[*] Department for Information Technology, Faculty of Physics, Astronomy and Applied Computer Science, Jagiellonian University in Krakow, ul. Reymonta 4, 30-059 Krakow, Poland `piotr.oramus@uj.edu.pl`

## 1. Introduction

Glowworm Optimization Algorithm (GSO) [1, 2] is a novel algorithm, which belongs to the group of algorithms based on swarm intelligence [3].

Observation of behaviour of ants, honeybee swarms, fish schools, flocking of birds proves that complex goals can be achieved by interactions of relatively simple individuals. Such observations help to construct multi agents algorithms, where decisions are not taken centrally.

The development of the optimization algorithm called Glowworm Optimization Algorithm has been based on the behaviour of glowworms (insects, which are able to modify their light emission and use the bioluminescence glow for different purposes).

GSO algorithm is especially useful for a simultaneous search of multiple optima, usually having different objective function values. To achieve this goal, a swarm must have an ability to split into disjoint groups. Otherwise, only one (local or global) optimum will be found. In GSO agents exchange information locally. Moreover, their movements are non deterministic.

It should be pointed out, that GSO algorithm computes multiple optima in parallel during one program run, what provides a set of alternative solutions to the user. This is especially beneficial when search space represents parameters of e.g. a real production process, where some conditions are easer to set up or simply financially cheaper.

Another algorithm, which can be used for locating multiple alternative solutions is a Niching Particle Swarm Optimization (NichePSO) algorithm [4], which is a variant of Particle Swarm Optimization (PSO) algorithm [5]. NichePSO divides particles into subswarms and for each subswarm uses Guaranteed Convergence Particle Swarm Optimization algorithm [6, 7] to move particles. A direct comparison of NichePSO and GSO [1] on three different test functions has shown that GSO performs better than NichePSO in terms of the number of successfully located solutions.

A comparison of GSO with gradient-ascent approach, which has been performed under noise conditions, can be found in [8]. The advantage of GSO has been indicated.

The scientific aim of this paper is to further improve performance of GSO algorithm. An improved algorithms should locate more optimum, effectively use small swarms and require less iterations i.e. it should consume less computer resources both CPU and memory.

The paper is organized as follows. Original and modified GSO algorithms are presented in Section 2. Test functions, performance measure and model constants are given in Section 3. Section 4 contains result of simulations. Conclusions are enclosed in the Section 5.

## 2. Algorithms

### 2.1. Glowworm Swarm Optimization

In GSO a swarm is composed of $N$ agents called glowworms. A state of a glowworm $i$ at time $t$ can be described by the following set of variables: a position in the search space ($\boldsymbol{x}^i(t)$), a *luciferin* level ($l^i(t)$) and a neighbourhood range ($r^i(t)$). GSO algorithm describes how these variables change over time.

Initially, agents are randomly distributed in the search space. Other parameters are initialized by predefined constants. Each, next iteration is composed of three phases: *luciferin* level update, glowworm movement and neighbourhood range update.

To encode in the *luciferin* level the fitness of the current position of a glowworm $i$, the following formula is used:

$$l^i(t) = (1-\rho)l^i(t-1) + \gamma J\left(\boldsymbol{x}^i(t)\right)$$

where: $\rho$ is the *luciferin* decay constant, $\gamma$ is the *luciferin* enhancement constant and $J$ is an objective function.

Then, each glowworm tries to find neighbours. In GSO a glowworm $j$ is a neighbour of a glowworm $i$ only if the distance between glowworms $i$ and $j$ is shorter than the neighbourhood range $r^i(t)$ and additionally glowworm $j$ has to shine brighter than $i$ ($l^j(t) > l^i(t)$). If one glowworm has multiple neighbours, chooses one by random with probability proportional to the *luciferin* level of this neighbour. Finally, glowworm moves one step in direction of the chosen neighbour. Step size is constant and equals $s$.

In the last phase, the neighbourhood range $r^i(t)$ is updated in order to limit the range of the communication in an ensemble of agents. The following formula is used:

$$r^i(t+1) = \min\left\{r_s, \max\left[0, r^i(t) + \beta\left(n_d - |n^i(t)|\right)\right]\right\}$$

where: $r_s$ is a sensor range (a constant, which limits the size of the neighbourhood range), $n_d$ is a desired number of neighbours, $|n^i(t)|$ is a number of neighbours of a glowworm $i$ at time $t$, and $\beta$ is a model constant.

### 2.2. Modified GSO

All proposed modifications aim to eliminate situations, in which the glowworm does not change its position.

#### 2.2.1. GSO v1

In the first version of modified GSO, in the case of no neighbours an agent *tries* to jump to a new location. The new coordinate $j$ of the glowworm $i$ is randomly generated on the basis of previous location by means of the formula:

$$\chi^i_j(t+1) = x^i_j(t) + (0.5 - R)s$$

where: $R$ is a random number having a uniform distribution in the range [0, 1) and $s$ is the step size.

New, *trial* location ($\chi$) is inside a hyper-cube whose side length equals to the step size and a centre is at old glowworm position. The jump is accepted only if the current solution will not be worsen. Otherwise the position of the glowworm remains not changed.

### 2.2.2. GSO v2

In the second version of modified GSO the concept of trial movements from GSO v1 is used and additionally the sensor range ($r_s$) is extended by a factor of 5% when number of glowworms inside a sphere of radius $r_s$ is less then $n_d$. This modification helps the agent to use information gained from other agents and weakens the effect of random movements introduced by first modification.

### 2.2.3. GSO group

In the original algorithm agents spontaneously divide into disjoint groups, but some of them are not able to join to any group or even to use information from other agents.

In accordance with presented in this chapter modification, agents always move and collaborate in predefined groups.

In each group there is one chosen agent (called *master*), which determines the movement of all agents from the group. The master is placed in the middle of its group and is surrounded by several *slave* agents, which are almost uniformly distributed on a hyper-sphere centred at master position. Distance between master and slave agents is denoted by $r_d$ and initially equals to 10% of the sensor range. The master-slave distance for a given group is controlled by the master agent and varies in the range from 33% of the step size to 10% of the sensor range. Only for some special cases, described later in this chapter, the master-slave distance equals to the sensor range.

In each iteration the master checks the *luciferin* value of own slave agents and in most cases uses standard GSO algorithm to designate the direction of a common group step. Agents from other groups are interrogated only in the case when all group members have the same value of the *luciferin*.

The whole GSO group algorithm is shown in the form of flowchart in Figure 1. It must be emphasised, that a core of the algorithm is relatively simple and consists of boxes marked with thick lines. At the same time, most of the flowchart area represents a code, which is seldom used in cases, where all agents manifest the same *luciferin* level.

If all agents have the same value of the *luciferin* the standard algorithm does not work at all, because all agents stay still. However, here, the master increases the $r_d$ to the value of the sensor range and allows slaves to search for other agents and copy the best found *luciferin* value. If this move does not help, the master allows slaves to randomly increase the *luciferin* quantity. The last strategy allows the group to explore the search space even if the objective function is locally constant.

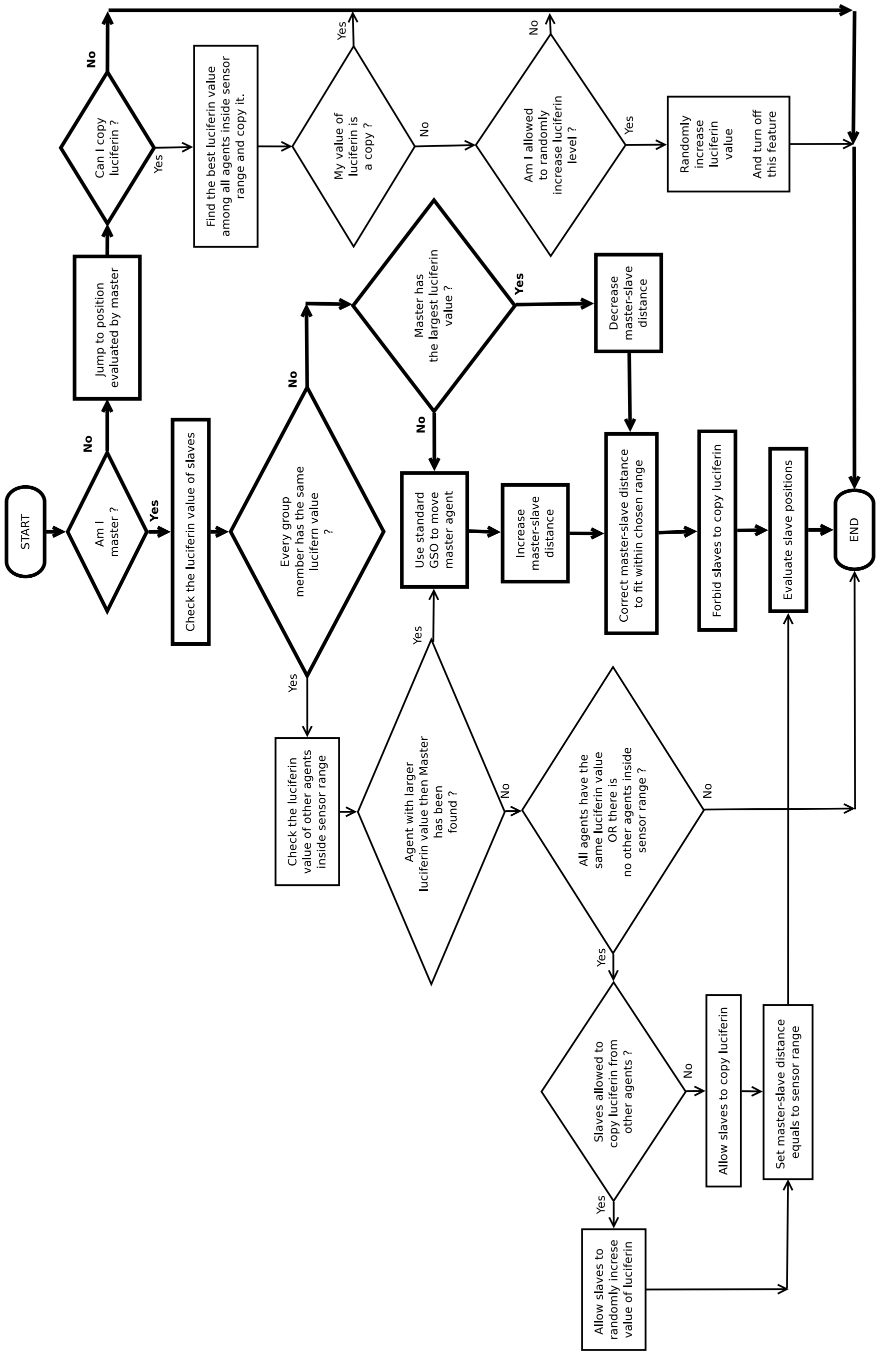
START
Am I master ?
No
Yes
Jump to position evaluated by master
Can I copy luciferin ?
No
Yes
Find the best luciferin value among all agents inside sensor range and copy it.
My value of luciferin is a copy ?
Yes
No
Am I allowed to randomly increase luciferin level ?
No
Yes
Randomly increase luciferin value
And turn off this feature
Check the luciferin value of slaves
Every group member has the same lucifern value ?
No
Yes
Master has the largest luciferin value ?
Yes
No
Decrease master-slave distance
Use standard GSO to move master agent
Increase master-slave distance
Correct master-slave distance to fit within chosen range
Forbid slaves to copy luciferin
Evaluate slave positions
END
Check the luciferin value of other agents inside sensor range
Agent with larger luciferin value then Master has been found ?
Yes
No
All agents have the same luciferin value OR there is no other agents inside sensor range ?
No
Yes
Slaves allowed to copy luciferin from other agents ?
No
Yes
Allow slaves to copy luciferin
Set master-slave distance equals to sensor range
Allow slaves to randomly increse value of luciferin

**Fig. 1.** Flowchart of single iteration of GSO group algorithm

**Table 1**
Summary of test functions

| Test function | Search space size | Dimensions | Number of maximums |
|---|---|---|---|
| Peaks $J_1(x,y)$ | [-3,3]×[-3,3] | 2 | 3+1 |
| Rastrigin's $J_2(x,y)$ | [-5,5]×[-5,5] | 2 | 100 |
| Himmelblau's $J_3(x,y)$ | [-5,5]×[-5,5] | 2 | 4 |
| $J_4(x,y)$ | [-5,5]×[-5,5] | 2 | 5 |
| Random peaks $J_5(\boldsymbol{x})$ | [-5,5]×$\cdots$×[-5,5] | 2-25 | 25 |

The last issue that must be explained is how positions of slaves are evaluated on the basis of the master-slave distance and a position of the master agent: a simple parabolic potential is used to bind slaves with their master – this potential has spherical symmetry and a minimum at $r_d$. Additionally, several orders of magnitudes weaker repulsive potential pushes the glowworms apart. Finally, master agent computes new positions of the slaves by a potential energy minimization.

## 3. Test procedure

### 3.1. Multimodal test functions

Four different algorithms (standard GSO and three proposed modifications) were tested on several test functions, which maxima were known. The search space size, the number of dimensions and the number of optimum solutions (here all points where function has a maximum, even a local one) for each test function are given in Table 1. The function equations are written down below:

Peaks function [9]:

$$\begin{aligned}J_1(x,y) =& 3(1-x)^2\exp\left(-x^2-(y+1)^2\right)-10\left(\frac{x}{5}-x^3-y^5\right)\exp\left(-x^2-y^2\right)-\\&\frac{1}{3}\exp\left(-(x+1)^2-y^2)\right)\end{aligned} \tag{1}$$

This function requires one comment: because search space is limited, the function at a point (3,-3) takes the value greater than in the allowed surrounding so this point was also treated as optimum.

Rastrigin's function [10]:

$$J_2(x,y) = 20 + \left(x^2 - 10\cos(2\pi x) + y^2 - 10\cos(2\pi y)\right)$$

Himmelblau's function [4]:

$$J_3(x,y) = 200 - (x^2+y-11)^2 - (x+y^2-7)^2$$

Test function introduced by the author:

$$\begin{aligned} J_4(x,y) =& H(\frac{7}{4} - d_1(x,y)) \cos(2\pi d_1(x,y)) \left(\exp(-d_2(x,y)) + \exp(-d_3(x,y))\right) + \\ & H(\frac{7}{6} - d_4(x,y)) \cos(3\pi d_4(x,y)) + H(\frac{7}{8} - d_5(x,y)) \cos(4\pi d_5(x,y)) \end{aligned} \tag{2}$$

where:

$$\begin{aligned} d_1(x,y) &= \sqrt{x^2 + (y-2.5)^2} \\ d_2(x,y) &= \sqrt{x^2 + (y-3.5)^2} \\ d_3(x,y) &= \sqrt{x^2 + (y-1.5)^2} \\ d_4(x,y) &= \sqrt{(x+2.5)^2 + (y+2.5)^2} \\ d_5(x,y) &= \sqrt{(x-2.5)^2 + (y+2.5)^2} \end{aligned}$$

and $H(x)$ is Heaviside unit step function.

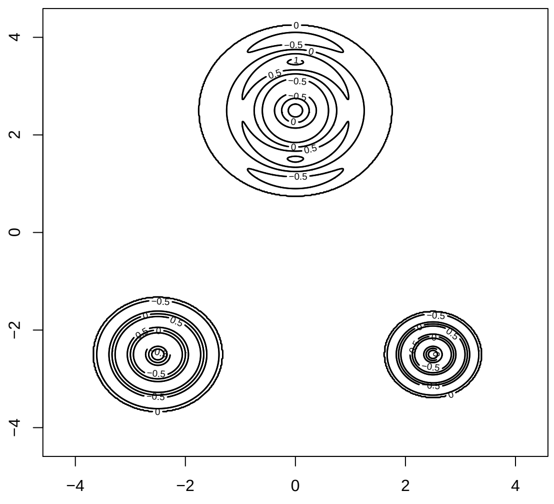

**Fig. 2.** Contour plot of the test function introduced by the author

The contour plot of this test function is given in Figure 2. It can be easily seen, that for the most of the search area this function assumes constant value. Moreover, each maximum is surrounded be a circle area, where the function is negative. Finally, the optimal solutions are difficult to detect, because most of GSO agents reveal the same or worse value of the *luciferin* and standard algorithm stops.

**Table 2**
GSO model constants

| $\rho$ | $\gamma$ | $\beta$ | $n_d$ | $s$ | initial *luciferin* value |
|---|---|---|---|---|---|
| 0.4 | 0.6 | 0.08 | 5 | 0.03 | 5.0 |

Random peaks multi-dimentional test function:

$$J_5(\boldsymbol{x}) = \sum_{i=1}^{25} a_i \exp(-b_i||\boldsymbol{x} - \boldsymbol{v}_i||)$$

where: $||\,||$ denotes Euclidean distance, and $a_i$, $b_i$, $\boldsymbol{v}_i$ are randomly generated constants. The positions of $\boldsymbol{v}_i$ points were chosen from 25-dimensional hyper space, but program allows to decrease dimensionality of this function by simply omitting excess information.

## 3.2. Performance measure

The single point maximum was considered to be found, when at least 3 glowworms were localized in its vicinity [1] – the distance limit 0.05 was used for all experiments. A normalized number of correctly found solutions was used as performance measure of the algorithm.

## 3.3. Common parameters

The model constants (see Table 2) were taken from paper [1]. The sensor range was set to 1.0 for all calculations and used an initial value for the neighbourhood range.

## 3.4. Test procedure, software and hardware used

All algorithms were tested by means of all presented test functions. Calculations for one set of input parameters (version of optimization algorithm, test function, the number of iterations and a swarm size) were repeated at least 100 times. The initial positions of agents were always generated by random. The system pseudo-random generator (`rand()` function from `stdlib` library) was applied for all experiments.

The group size used in the case of GSO group modification, was equal to 4 for two-dimensional test functions (from $J_1$ to $J_4$) and 25 for the random peaks multi-dimensional test function ($J_5$).

For two-dimensional test functions one result was evaluated during several seconds of CPU time of Intel(R) Pentium(R) D running at 3.4 GHz. Similar calculations made for multi-dimensional cases required up to 15 hours of CPU time per one optimization and were performed on IBM BladeCenter HS21 (`mars.cyfronet.pl`) cluster.

All input and output parameters of self-made software were stored in the relational database (MySQL). The results were elaborated and plotted by means of

R-language scripts, which were able to download the averaged data directly from the database.

## 4. Results

In order to ease analysis of the results, the same symbol is used to represent one algorithm in all Figures. The standard GSO is represented by a filled circle (•). For GSO modifications version 1 and version 2, open triangles are used (GSOv1 - △, GSOv2 - ▽). Data points for GSO group algorithm are piloted by means of an open square symbol (□). The error bars shown in Figures are equal to the standard deviation of the mean.

### 4.1. Two dimensional test functions

The calculation were performed for two different swarm sizes: 25 and 100 glowworms. For a single swarm size four different numbers of iterations were used: 250, 500, 750 and 1000. In all Figures normalized average number of peaks captured are plotted versus number of iterations. Results for 25 glowworms are plotted in Figures a) on the left and for 100 glowworms in Figures b) on the right.

#### 4.1.1. Peaks function $J_1$

The results of calculations are presented in Figures 3a and b. The optima of the peaks function are relatively easy to locate. Even in the worst case (25 agents and only 250 iterations) more than 40% of them were correctly found. When 100 agents were used all algorithms were able to locate more than 80% of peaks.

For a swarm composed of 25 members, all proposed by the author algorithms were better than the original one. In the case of calculations with 100 glowworms, only group algorithm was significantly better in comparison to the standard GSO. The performance of GSO v2 was worse than the standard GSO.

Bad performance of GSO v2 can be explained by a following observation: this modification adds an ability to increase value of the sensor range parameter, but by using this feature agent increases its own area of an information exchange and is attracted in direction of a global optimum. Finally, the number of agents, which can explore the search space looking for local optimum decreases.

#### 4.1.2. Rastrigin's function $J_2$

The calculation results are presented in Figures 4a and b. Because of the large number of possible solutions (100) in comparison to the number of agents, the performance measure did not take large values. The best behaviour was revealed by an algorithm, where glowworms travelled in predefined groups (GSO group).

This phenomena can be easily understood, when one takes into account a definition of a success criteria used here as a performance measure: a peak is considered as captured when arbitrary chosen number of glowworms can be found in its neighbourhood. In GSO group algorithm agents always have neighbours and moreover this algorithm allows the master agent to decrease the master-slave distance. Finally, all group members meet together at an optimum location.

a) b)

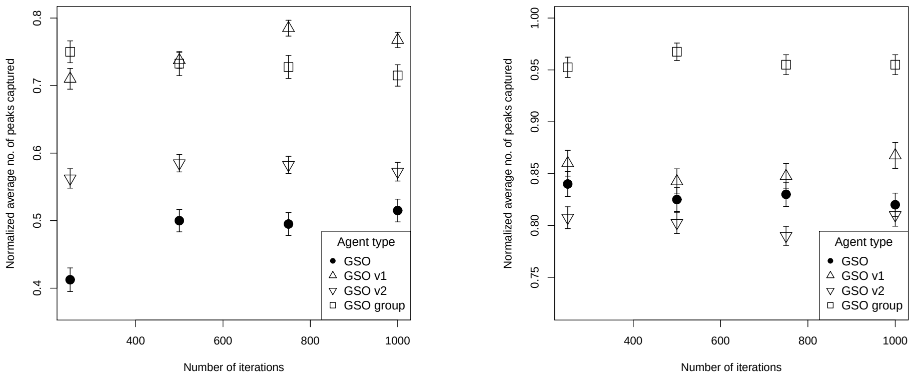

**Fig. 3.** Peaks test function

a) b)

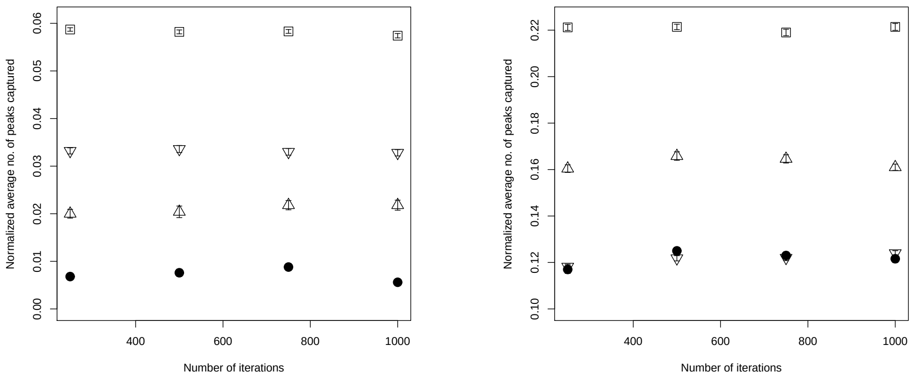

**Fig. 4.** Rastrigin's test function

Nonetheless, presented results prove, that GSO group algorithm works correctly and efficiently steers group towards the optimum.

In the case of the GSO v2 algorithm, the increase of the sensor range helped only in the case of a small number of agents (25). When number of agents was relatively large (100) the performance of GSO v2 and standard GSO was comparable.

### 4.1.3. Himmelblau's function $J_3$

The experimental results are plotted in Figures 5a and b. In the case of small number of agents (25) results reveal the superiority of the improved algorithms. When number of iterations was large enough, the GSO v1 algorithm was able to locate all solutions. For 100 agents all algorithms were almost equally good.

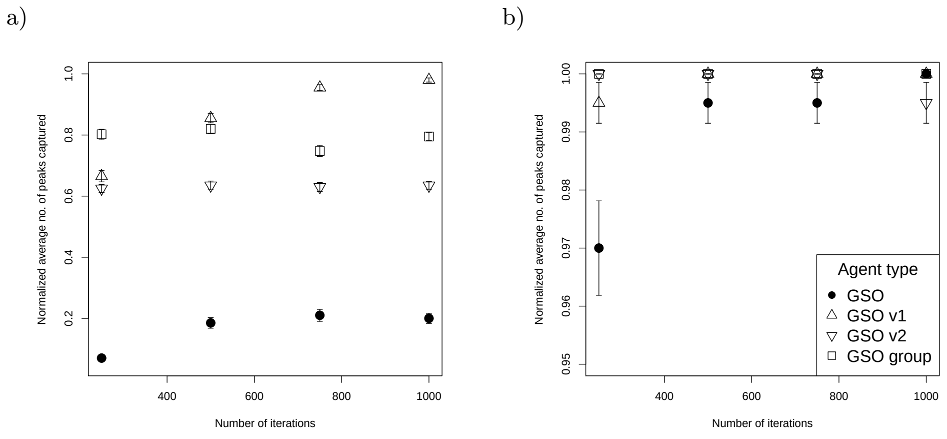

**Fig. 5.** Himmelblau's test function

### 4.1.4. Test function introduced by the author $J_4$

The results of calculations performed for the test function introduced by the author are shown in Figures 6a and b.

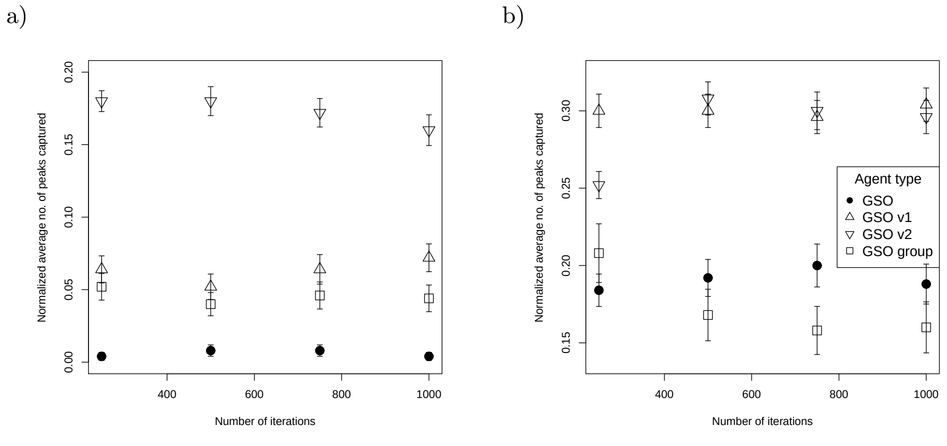

**Fig. 6.** $J_4$ test function

As it was mentioned before, optima of this function are difficult to find, because most of the agents broadcast the same *luciferin* level and the standard GSO algorithm stops. Therefore, for the swarm composed of 25 glowworms, the performance of the standard GSO was almost zero. The improved algorithms behaved better and reported from around 5% (GSO v1 and GSO group) to even more than 15% (GSO v2) of peaks as successfully located. One should note, that all proposed modifications allow search space exploration even in lack of an information coming from the difference in the *luciferin* level.

The performance of the bigger swarms (100 members) was much better and at least 15% of peaks were localized. The best results were obtained in the case of GSO v1 and GSO v2.

## 4.2. Multi-dimensional test function $J_5$

The calculation were performed for a swarm consisting in 1000 glowworms and took 10000 iterations. Experiments were done for the following number of dimensions: 2, 5, 10, 15, 20 and 25. The results are shown in Figure 7.

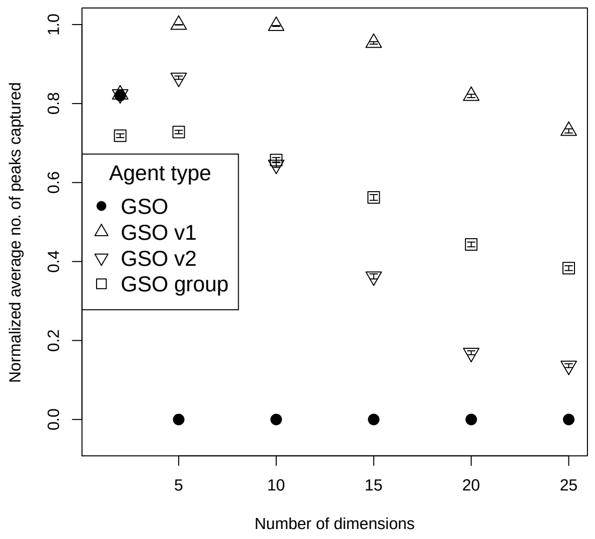

**Fig. 7.** Random peaks multi-dimensional test function. Normalized average number of peaks captured plotted versus number of dimensions

It can be easily seen, that the standard GSO algorithm completely lost its ability to locate optima for 5 dimensions and more. It took place, because the agents density rapidly dropped with the increasing number of dimensions and a lot of glowworms could not find any neighbour, so they were unable to move, again!

All algorithms proposed by the author achieved better performance in comparison to the standard GSO, when the number of dimensions was equal to 5 or larger.

## 5. Conclusions

Simple modifications of the standard GSO algorithm were able to substantially increase the algorithm performance. The improvement was especially visible in the case of swarms composed of relatively small number of agents, because more agents were able to move and thus were involved in the search for solutions.

Probably it is possible to further improve the family of GSO algorithms. In the case of GSO v2 it is possible to add an additional mechanism, which will decrement the sensor range to its initial level. This modification aims to restore the local information exchange.

### Acknowledgements

*The research reported here was partially supported by the grant no. MNiSW/ SGI4700/UJ/126/2007.*

## References

[1] Krishnanand K. N., Ghose D.: *Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions*, Swarm Intelligence, vol. 3, 2009, No. 2, pp. 87–124.

[2] Krishnanand K. N., Amruth P., Guruprasad M. H., Bidargaddi S. V., Ghose D.: *Rendezvous of glowworm-inspired robot swarms at multiple source locations: a sound source based real-robot implementation.* Ant Colony Optimization and Swarm Intelligence, Lecture Notes in Computer Science, LNCS 4150, Springer, Berlin, Germany, 2006, pp. 259–269.

[3] Bonabeau E., Dorigo M., Theraulaz: *G. Swarm intelligence: from natural to artificial systems.* New York, Oxford University Press 1999.

[4] Brits R., Engelbrecht A. P., van den Bergh F.: *A niching particle swarm optimizer.* Proc. of the 4th Asia-Pacific Conference on Simulated Evolution and Learning, Singapore, 2002, pp. 692–696.

[5] Kennedy J., Eberhart R.: *Particle Swarm Optimization.* Proc. IEEE International Conference on Neural Networks, 1995, pp. IV: 1942–1948.

[6] van den Bergh F.: *An analysis of Particle Swarm Optimizers.* PhD Thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

[7] van den Bergh F., Engelbrecht A. P.: *A new locally convergent particle swarm optimizer*. Proc. of IEEE International Conference on Systems, Man, and Cybernetics 2002, 2002, pp. 96–101.

[8] Krishnanand N. K., Ghose D.: *Glowworm swarm based optimization algorithm form multimodal functions with collective robotic applications*. Multiagent and Grid Systems, 2(3), 2006, pp. 209–222.

[9] Reutskiy S. Y., Chen C. S.; *Approximation of multivariate functions and evaluation of particular solutions using Chebyshev polynomial and trigonometric basis functions*. International Journal for Numerical Methods in Engineering, 67(13), 2006, pp. 1811–1829.

[10] Törn A., Zilinskas A.: *Global optimization*. New York, Springer 1989.