Implementation of a swarm intelligence algorithm to a mobile device

L.E. Gomez[1], J.F. Jimenez[1], J.H. Sossa[1], F.J. Cuevas[2] , O. Pogrebnyak[1], R. Barron[1]

[1] Centro de Investigación en Computación-IPN, Unidad Profesional Adolfo-López Mateos, Av. Juan de Dios Bátiz s/n and M. Othón de Mendizábal, Zacatenco, México, DF. 07738, Mexico
[2] Centro de Investigaciones en Óptica A.C. Loma del Bosque #115, Col. Lomas del Campestre C.P. 37150, León Gto. México
sgomezb08@sagitario.cic.ipn.mx, jfvielma@cio.mx, hsossa@cic.ipn.mx, fjcuevas@cio.mx, olek@cic.ipn.mx, rbarron@cic.ipn.mx

Abstract. In this paper, we propose the implementation of the algorithm of particle swarm optimization to the mobile platform, with its limitations even in our day is a new way to launch applications with bio-inspired techniques for this type of platform. The particle swarm optimization (PSO) proposed by Kennedy and Eberhart in 1995, is discussed in a numerical optimization of a three benchmark functions taken from the literature. PSO is motivated by social behavior to organisms, such as meetings among birds or fish. Each particle consists of three main parts, its speed, cognitive knowledge and social knowledge.

Keywords: Particle Swarm Optimization; Mobile device; JAVA.

1 Introduction

The optimization in the sense of finding the best solution, or at least a good enough solution for a problem is a field of vital importance in real life. We are constantly solving small problems of optimization, such as the shortest way to get from one place to another, the organization of a book, etc.. In general they are small enough and can be resolved without recourse to external elements to our brain. But as they get larger and more complex, the use of computers to its resolution is unavoidable.

Due the great importance optimization problems over the history computing, have developed multiple methods try to solve them. Around the seventies came a class of algorithms are not accurate, whose basic idea was combine different heuristics to a higher level to get an exploration of the search space efficiently and effectively. These techniques have been called metaheuristics.

From the different descriptions of metaheuristics which are found in the literature can be rankings were certain fundamental properties that characterize this type method:

- The metaheuristic are templates by general strategies or "guide" the search process.

Received 30/04/10
Final version 19/05/10
Accepted 14/05/10

- The goal is an exploration of the search space efficiently to find solutions (almost) optimal.
- The metaheuristic algorithms are not exact and are generally nondeterministic.
- They may incorporate mechanisms to avoid areas of non-optimal search space.
- The basic layout any metaheuristic is general and not dependent on the problem to be solved.
- The functions used in metaheuristics is goodness (fitness functions) to quantify the appropriateness of a particular solution

Summarizing these points, a metaheuristic is a high-level strategy that uses different methods to explore the search space.

With the dramatic increase and sophistication of mobile devices such as cell phones, also comes the demand for applications that run on them, corporations want to expand consumer devices for mobile communications devices for voice communications applications traditionally found on laptops and PC's.

Developers, handset manufacturers, are eager to fill this need, however there is a serious difficulty, mobile communications devices use different application platforms and operating systems, in addition to the various virtual machines which do not have all the libraries in J2SE (Java 2 Standard Edition).

An application that runs on a device hardly runs in another. Mobile devices lack a standard application platform and the same operating system, which causes the development of applications for mobile devices, is a financial risk for developers.

The lack of standards is nothing new to the area of computer technology or any new development. Traditionally, hardware device manufacturers try to force the market to accept its standards. Owners of as standards in the industry, other times, industry leaders formed a consortium, such as Java Community Process Program, to collectively define a standard [1].

The J2ME architecture is oriented to small devices and embedded systems such as mobile phones, PDAs, Pockets, etc. In order have a J2ME runtime environment that meets the requirements of a wide range of devices and target markets is required to be made to Figure 1:

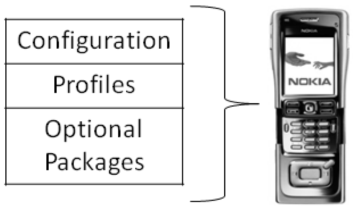


Fig. 1. Mobile Device Architecture.

The settings consist of a virtual machine and a minimal set of function libraries. Provide basic functionality for a set of devices that share similar characteristics, such as memory management or network connectivity. There are two types of configurations such as CLDC devices aimed at processing and memory constraints, and the CDC focused on devices with more resources.

To form a complete runtime environment targeted to a category of devices, the settings have to be combined with a set of APIs to a higher level, called profiles, which go a step further in defining the life cycle model of applications, user interface and access to specific properties of the devices. At present there are four profiles: MIDP (Mobile Information Device Profile), FP (Foundation Profile), PP (Personal Profile) and PBP (Personal Basis Profile) [2].

Regarding J2ME optional packages can be extended by combining various optional packages with CLDC and CDC along with their profiles. These packages were created to meet specific requirements and offer a set of standard APIs for using both existing and emerging technologies such as Bluetooth, Web services, wireless messaging, multimedia capabilities and connectivity to databases. Because they are modular, manufacturers can incorporate according needed to improve the features supported.

There are two versions of MIDP: 1.0 and 2.0. In MIDP, often uses the term to refer a Midlet application that supports the profile. For distribution, one or more MIDlets are grouped in bundles called suites, which consist of a file format Jar and an optional descriptor Jad [3], this is shown in Figure 2.

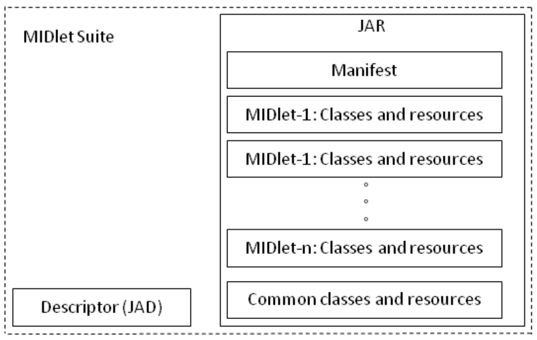

MIDlet Suite
JAR
Manifest
MIDlet-1: Classes and resources
MIDlet-1: Classes and resources
MIDlet-n: Classes and resources
Common classes and resources
Descriptor (JAD)

Fig. 2. Structure of a MIDlet suite.

The active state of a MIDlet is when the application is running. State is destroyed when the application terminates and frees the memory required. The paused state occurs when the application stops for a particular event or has not been activated yet, when it created [4], this is shown in Figure 3.

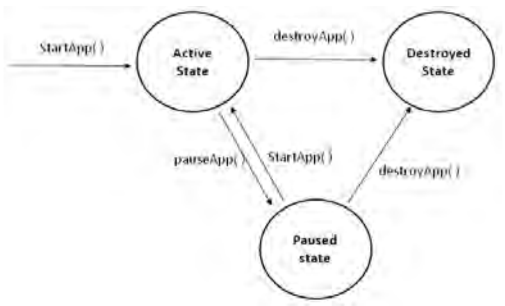


Fig. 3. Life cycle of a MIDlet.

2 Particle Swarm Optimization (PSO)

PSO is a stochastic global optimization method which is based on simulation of social behavior. As in GA and ES, PSO exploits a population of potential solutions to probe the search space. In contrast to the aforementioned methods in PSO no operators inspired by natural evolution are applied to extract a new generation of candidate solutions. Instead of mutation PSO relies on the exchange of information between individuals, called *particles*, of the population, called *swarm*. In effect, each particle adjusts its trajectory towards its own previous best position, and towards the best previous position attained by any member of its neighborhood [5]. In the global variant of PSO, the whole swarm is considered as the neighborhood. Thus, global sharing of information takes place and particles profit from the discoveries and previous experience of all other companions during the search for promising regions of the landscape. To visualize the operation of the method consider the case of the single objective minimization case; promising regions in this case possess lower function values compared to others, visited previously.

Several variants of PSO have been proposed up to date, following Eberhart and Kennedy who were the first to introduce this method [6], [7], [8]. The variants which were applied in our experiments are exposed in the following paragraphs.

Table 1 shows a number of terms used in the traditional PSO algorithm.

Table 1. Terms of PSO algorithm.

Term	Description
Particle / Agent	An individual of the swarm
Location / Position	Coordinates of an agent in N-dimensional space represents a solution to the problem
Swarm	A whole collection of agents / A population of individuals
Fitness	A number that indicates the quality of a given solution (represented by a location in the solution space)
pbest (Personal best)	It is the best location obtained by a given agent during the process
gbest (Global best)	It is the best location in all the particle swarm

Initially, let us de ne the notation adopted in this paper: assuming that the search space is D dimensional, the *i*-th particle of the swarm is represented by a D dimensional vector $X_i = (x_{i1}, x_{i2}, ..., x_{iD})$ and the best particle of the swarm, i.e. the particle with the lowest function value, is denoted by index g. The best previous position (i.e. the position corresponding to the best function value) of the *i*-th particle is recorded and represented as $P_i = (p_{i1}, p_{i2}, ..., p_{iD})$, and the position change (velocity) of the *i*-th particle is $V_i = (v_{i1}, v_{i2}, ..., v_{iD})$.

The particles are manipulated according to the following equations (the superscripts denote the iteration):

$$V_i^{k+1} = \gamma\left(\omega V_i^k + c_1 rand_{i1}^k(\ )\left(P_i^k - X_i^k\right) + c_2 rand_{i2}^k(\ )\left(P_g^k - X_i^k\right)\right) \quad \textbf{(1)}$$

$$X_i^{k+1} = X_i^k + V_i^{k+1}, \tag{2}$$

where $i = 1, 2, ..., N$, and N is the size of the population; γ is a *constriction factor* which is used to control and constrict velocities; ω is the *inertia weight*; c_1 and c_2 are two positive constants, called the cognitive and social parameter respectively; $rand_{i1}()$ ri1 and $rand_{i2}()$ are random numbers uniformly distributed within the range $[0,1]$. Eq. (1) is used to determine the *i*-th particle's new velocity, at each iteration, while Eq. (2) provides the new position of the *i*-th particle, adding its new velocity, to its current position. The performance of each particle is measured according to a fitness function, which is problem dependent. In optimization problems, the fitness function is usually identical with the objective function under consideration.

The update of a particle in general is illustrated in Figure 4.

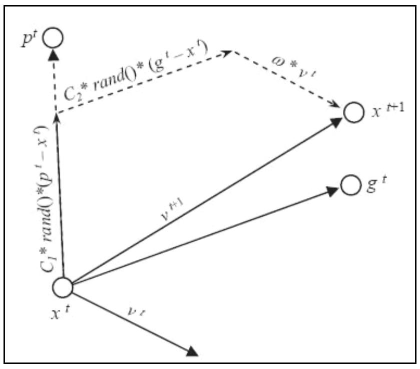


Fig. 4. Update particle.

The role of the inertia weight ω is considered important for the PSO's convergence behavior. The inertia weight is employed to control the impact of the previous history of velocities on the current velocity. Thus, the parameter ω regulates the tradeoff between the global (wide ranging) and the local (nearby) exploration abilities of the swarm. A large inertia weight facilitates exploration (searching new areas), while a small one tends to facilitate exploitation, i.e. fine tuning the current search area. A proper value for the inertia weight ω provides balance between the global and local exploration ability of the swarm, and, thus results in better solutions. Experimental results imply that it is preferable to initially set the inertia to a large value, to promote global exploration of the search space, and gradually decrease it to obtain refined solutions [9]. The initial population can be generated either randomly or by using a Sobol sequence generator [10], which ensures that the *D*-dimensional vectors will be uniformly distributed within the search space.

The PSO technique has proven to be very efficient for solving real valued global unconstrained optimization problems [11],[12]. In the next section experimental results of the performance of PSO in mobile device.

3 Experiment

Three numerical optimization problems were chosen to compare the relative performance of PSO algorithm in a mobile device. These functions are standard functions of all test patterns and minimization problems.

3.1 Functions

The functions are unimodal. All functions are designed to have global minimum near the origin.

The first test function is the function given by equation Sphere:

$$\begin{aligned} &f_1(x) = \sum_{i=1}^{n} x_i^2 \\ &-100 \le x_i \le 100 \\ &\min(f_1) = f_1(0,..,0) = 0 \end{aligned} \tag{3}$$

x is a real vector of dimension n and x_i is the i-th element in the vector. The results of optimizing the function (3), PSO heuristics are shown in Table 2.

Table 2 Results from the function (3).

Iterations	*Particles*	Velocity	Variables	Fitness
500	50	4	10	6.15
400	100	4	10	2.88
300	50	4	10	2.03
300	30	4	10	3.19
200	100	4	10	5.43

The second function is Schwefel's problem, given by the equation:

$$\begin{aligned} &f_2(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i| \\ &-10 \le x_i \le 10 \\ &\min(f_2) = f_1(0,..,0) = 0 \end{aligned} \tag{4}$$

The results of optimizing the function (4), PSO heuristics are shown in Table 3.

Table 3 Results from the function (4).

Iterations	*Particles*	Velocity	Variables	Fitness
400	30	1	10	1.26
300	50	1	10	1.15
300	30	1	10	1.27
250	50	1	10	1.25
200	30	1	10	1.49

The third function is Step, given by equation:

$$\begin{aligned} & f_3(x) = \sum_{i=1}^{n} \left(\lfloor x_i + 0.5 \rfloor \right)^2 \\ & -100 \le x_i \le 100 \\ & \min(f_3) = f_1(0,..,0) = 0 \end{aligned} \qquad (5)$$

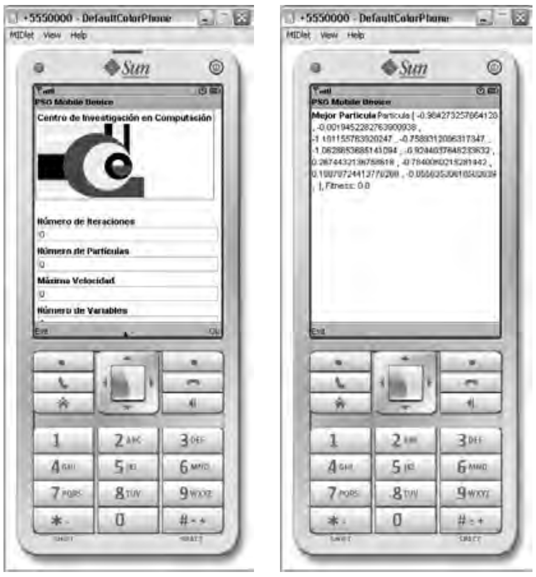
The results of optimizing the function (5), PSO heuristics are shown in Table 4.

Table 4 Results from the function (5).

Iterations	*Particles*	Velocity	Variables	Fitness
600	20	4	10	0.0
500	20	4	10	0.0
450	20	4	10	0.0
400	20	4	10	3.0
350	20	4	10	3.0

3.2 PSO Simulator

The PSO was programmed in the Java language in its version J2ME for programming with IDE I use Netbeans 6.8 and Sun Java Wireless Toolkit 2.5.2 for CLDC, the first tests were in a simulator on a PC, these performances are shown in Figure 5.



Número de Iteraciones
0
Número de Partículas
0
Máxima Velocidad
0
Número de Variables

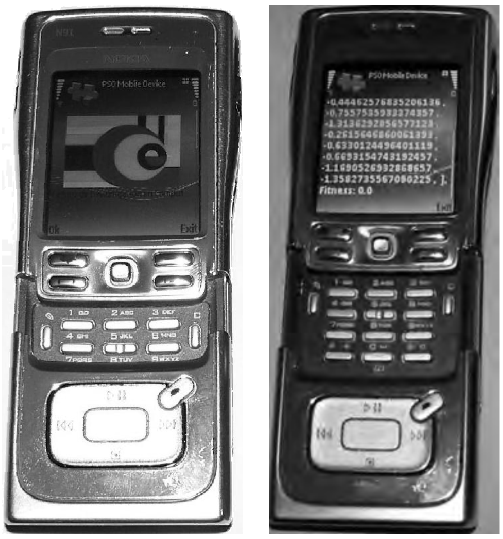
Mejor Particula Particula [-0.984273257664120 , -0.0019452282763900938 , -1.101155783920247 , -0.7588312096317347 , -1.0628853885141094 , -0.9244037848233832 , 0.2874432136758618 , -0.7840080215281442 , 0.10070724413770208 , -0.0556253061[illegible]], Fitness: 0.0

a) b)

Fig. 5. a) Home Screen Simulator, b) Results Screen Simulator.

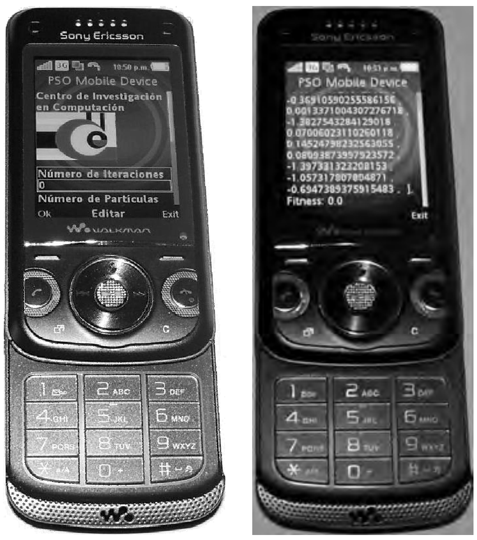
2.1 PSO Mobile Devices

To run on mobile devices requires that the device has JAVA support, the tests presented here in two phones, a Nokia brand, model N91, the other brand W760i SONY ERIKSSON, executions shown in Figures 6 and 7.



```
-0.44462576835206136 ,
-0.7557535993274357 ,
-1.3136292856577123 ,
-0.2615646860061393 ,
-0.6330124496401119 ,
-0.6693154743192457 ,
-1.1690526932868657 ,
-1.3582735567090225 , ].
Fitness: 0.0
```

a) b)
Fig. 6. a) Start Screen in Nokia N91, b) Results Screen in Nokia N91.



Centro de Investigación en Computación
Número de Iteraciones
0
Número de Particulas

```
-0.36910590255586156 ,
0.0013371004307276718 ,
-1.3827543284129018 ,
0.07006023110260118 ,
0.14524798232563055 ,
0.08093873997923572 ,
-1.397331323208153 ,
-1.0573178070604871 ,
-0.6947389375915483 , ].
Fitness: 0.0
```

b) b)
Fig. 7. a) Start Screen in Sony Ericsson w760i, b) Results Screen in Sony Ericsson w760i.

4 Conclusions

In this paper, it was possible the analysis of three well-known problems in the literature. The results showed that with PSO optimized mobile devices programmatically gives very good results to minimize such functions, which have been tested with other techniques such as bio-inspired Differential Evolution.

The implementation of bio-inspired techniques to a new platform such as mobile devices opens a further step in the investigation. For many years the technology has advanced so amazing, just look around us to realize, before he took on a Palm, pocket, and mobile phone was to please a taste, now cover human needs. Support multiple applications, ranging from taking a picture, listening to music, to do complex numerical computations, important for the realization of this project.

Programming for mobile now has endless barriers, since they do not have all the benefits of programming language when it is for PC's, talking specifically about the mobile libraries included. However, there are ways to get the results from its own library of programming for that platform. The program was tested on different mobile devices.

Acknowledgements. We wish to thank the Centro de Investigación en Computación of the I.P.N. by the support to accomplish this project. L.E. Gomez and J.F. Jimenez thanks CONACYT by the scholarship received to complete his doctoral studies. H. Sossa thanks the SIP-IPN under grant 20091421 for the support. R. Barron thanks the SIP-IPN under grant 20100379 for the support. O. Probegnyak thanks the SIP-IPN for the support. H. Sossa also thanks CINVESTAV-GDL for the support to do a sabbatical stay from December 1, 2009 to May 31, 2010. Authors thank the European Union, the European Commission and CONACYT for the economical support. This paper has been prepared by economical support of the European Commission under grant FONCICYT 93829. The content of this paper is an exclusive responsibility of the CIC-IPN and it cannot be considered that it reflects the position of the European Union. Finally, authors thank the reviewers for their comments for the improvement of this paper.

References

[1] Microjava.: Introducción a J2ME y KVM, Tutorial, http://microjava.com.
[2] Keogh James.: The complete reference, Edit. McGraw-Hill, 2003.
[3] Qusay Mahmoud.: Learming wireless Java, Edit. O'Really, December 2001.
[4] Gálvez Rojas Sergio., Ortega Díaz Lucas.: Java a tope: J2ME, Depto de lenguajes y ciencias de la computación.
[5] Kennedy, J.: The Behavior of Particles. Evol. Progr. VII (1998) 581-587.
[6] Eberhart, R.C., Simpson, P.K., Dobbins, R.W.: Computational Intelligence PC Tools. Academic Press Professional, Boston (1996).
[7] Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. Proc. IEEE Int. Conf. Neural Networks. Piscataway, NJ (1995) 1942-1948.
[8] Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann (2001).

[9] Shi, Y., Eberhart, R.C.: Parameter Selection in Particle Swarm Optimization. Evolutionary Programming VII (1998) 591-600.

[10] Press, W.H., Vetterling, W.T., Teukolsky, S.A., Flannery, B.P.: Numerical Recipes in Fortran 77. Cambridge University Press, Cambridge (1992).

[11] Parsopoulos, K.E., Plagianakos, V.P., Magoulas, G.D., Vrahatis, M.N.: Objective Function "Stretching" to Alleviate Convergence to Local Minima. Nonlinear Analysis TMA 47(5) (2001) 3419-3424.

[12] Parsopoulos, K.E., Vrahatis, M.N.: Initializing the Particle Swarm Optimizer Using the Nonlinear Simplex Method. A. Grmela, N.E. Mastorakis (eds.), Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation. WSEAS Press(2002) 216-221.