ZESZYTY NAUKOWE POLITECHNIKI ŁÓDZKIEJ
Nr 1045 ELEKTRYKA, z. 117 2009

JAROSŁAW KACERKA
Technical University of Lodz
Institute of Automatic Control

APPLICATION OF SWARM INTELLIGENCE ALGORITHMS IN CONTROL PROBLEMS[1]

Reviewer: **Professor Jacek Kabziński, Ph.D., D.Sc.**

Manuscript received 27.03.2009

An application of swarm intelligence algorithms to control problems which may be described as an optimisation task is analysed in the paper. A multi-purpose implementation of PSO algorithm allowing for optimisation requiring a reduced number of fitness function evaluations is presented. The conducted experiments show the effectiveness and efficiency of the proposed algorithms.

1. INTRODUCTION

Algorithms based on Swarm Intelligence (SI) are global search and optimisation techniques which operate analogously to the behaviour of large groups of various species existing in nature. The behaviour is influenced by the external stimuli and leads to obtaining a certain goal which is optimal for the survival of the whole population (e.g. a flock of birds or a fish school). In a SI algorithm a collection of particles is created as an analogy to a swarm of living organisms. Their position is modified according to specific rules which are based on the fitness function describing the search space, connections between particles and the search history. Areas or points of swarm concentration are the solution of the chosen optimisation problem.

Numerous problems of control may be formulated as optimisation tasks. They may include identification, modelling, extremal control, optimal tuning of controller parameters, tuning of fuzzy and neuronal models, distributed control

[1] The research was supported by Ministry of Science and Higher Education under grant number N N514 0918 33.

in robotics etc. The evaluation of the fitness function value in such cases often requires complicated calculations since it is a result of long-lasting simulation or measurements in a real system. The search for global optimum is performed at the expense of evaluation of the fitness function value in a large number of locations and it constitutes a serious disadvantage in control applications. Therefore, a method to reduce the number of required fitness function evaluations is needed.

2. PARTICLE SWARM OPTIMISATION

Particle swarm optimisation (PSO) is a computation method developed by Kennedy and Eberhart [1], [2]. It is based on the imitation of social behaviour rather than evolution as in other evolutionary algorithms, e.g. genetic algorithms. It is a population-based evolutionary algorithm, similar to the other population-based evolutionary algorithms, and it is initialised with a population of random solutions. Potential solutions, called particles, are characterised by their position in the D-dimensional problem space $x_i=(x_{i1},\ldots,x_{id},\ldots,x_{iD})$ where $x_{id}\in[l_d,u_d]$ (l_d, u_d are the lower and upper bounds for the d-th dimension of the problem space respectively; $L=(l_1,\ldots,l_d,\ldots,l_D)$, $U=(u_1,\ldots,u_d,\ldots,u_D)$ are vectors describing lower and upper bound for all dimensions) and velocity $v_i=(v_{i1},\ldots,v_{id},\ldots,v_{iD})$ (limited to $V_{max}=(v_{max,1},\ldots,v_{max,d},\ldots,v_{max,D})$). Both positions and velocities of particles are initially randomised.

Every particle retains the location $P_i=(p_{i1},\ldots,p_{id},\ldots,p_{iD})$ and fitness function value associated with the best solution (the lowest value of the fitness function) it has achieved. This value is called a *pbest.* The overall best value called a *gbest* and its location $P_g=(p_{g1},\ldots,p_{gd},\ldots,p_{gD})$ obtained by any of the particles in the population is tracked by the particle swarm optimiser. In every iteration of the optimisation algorithm velocity of each particle is changed towards its *pbest* and *gbest* locations with separate random weights according to the equations:

$$v_{id} = w\cdot v_{id} + c_1\cdot R_1\cdot(p_{id} - x_{id}) + c_2\cdot R_2\cdot(p_{gd} - x_{id}) \tag{1}$$

$$x_{id} = x_{id} + v_{id} \tag{2}$$

The left side of these equations represents the value of velocity and location in the current iteration, while v_{id} and x_{id} on the right side are the values from the previous iteration, w is an inertia weight, c_1 and c_2 are acceleration constants, and R_1, R_2 are random numbers in the range of $[0,1]$. In equation (1) the first component represents the inertia of the previous velocity, the second part is the "cognition" part, which represents the private thinking of the particle; the third part is the "social" part, which represents the co-operation among

particles. If the sum of accelerations causes the velocity v_{id} on that dimension to exceed $v_{max,d}$, then v_{id} is limited to $v_{max,d}$. V_{max} determines the resolution with which regions between the present position and the target position are searched [2].

The process of PSO is as follows:

1 – Initialise a population (array) of N particles with random positions and velocities in the D-dimensional problem space;
2 – Evaluate fitness function value for every particle;
3 – Compare each particle fitness evaluation with its *pbest*. If current value is better than *pbest*, then set the *pbest* value equal to the current value and the *pbest* location equal to the current location in space;
4 – Compare each particle's fitness evaluation with the population's overall previous best. If current value is better than *gbest*, then set *gbest* to the current particle's location and value;
5 – Change the velocity and position of the particle according to equations (1) and (2), respectively;
6 – Loop to step (2) until a stopping criterion is met, usually a sufficiently good fitness or a maximum number of iterations.

3. REDUCTION OF FITNESS FUNCTION EVALUATIONS

The adaptation of the Particle Swarm Algorithm to optimisation problems which require the evaluation of a costly fitness function is a multi-stage task. Therefore, the performance of PSO as an optimisation method should be effective and the evaluations of fitness function which are not necessary should be eliminated.

The right choice of swarm parameter values and their adaptation according to the information about the progress of the optimisation process allows acceleration of the algorithm performance and may as well prevent stagnation i.e. stopping the whole swarm in a local optimum [3][4][5]. Faster convergence of the algorithm results in fewer iterations being made and in fewer required fitness function values.

Based on the analysis of the solutions described in literature several rules were formulated. The swarm parameters should be tuned according to those rules to ensure an effective optimisation. The proposed Enhanced Fuzzy-Adaptive PSO (EFAPSO) algorithm, which is based on the ideas described in [3], follows those rules. The values of two swarm parameters i.e. inertia w and c_1/c_2 ratio are tuned by a fuzzy inference system (figure 1) based on three proposed indicators evaluating progress of swarm optimisation: normalised best found value of the fitness function, normalised average distance between particles, number of algorithm iterations without improvement of fitness function value.

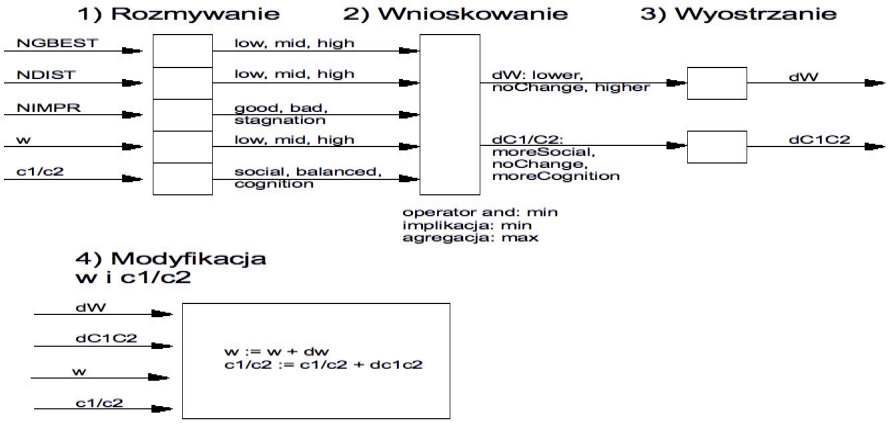

Fig. 1. Diagram of the fuzzy inference system in EFAPSO algorithm

An analysis of the PSO algorithm shows, that only these evaluated values of the fitness function are used which are better fitted than those found previously. The information about all other values is discarded and lost. Nevertheless, it may be used to construct a model of the fitness function, which – with some accuracy – allows to approximate the function value and thus to avoid some unnecessary computation. The difficulty in modelling a function based only on the points obtained during the swarm optimisation is that they are spaced irregularly – the majority of modelling methods described in literature require a definite, regular grid of points. In the proposed Mapping PSO (MPSO) algorithm the model of the fitness function is built in two stages. The values obtained during previous iterations of the optimisation algorithm are used to calculate a local model of the function (figure 2). The values of the local model, which approximate the fitness function in a small area, are calculated by multiquadric interpolation (MQI) [6]. Additionally, the reliability of each approximated value is calculated – depending on the predicted accuracy of the approximation. The value approximated by the whole model, i.e. the value of the global model, is calculated as a weighted average of local models.

The further possibility of reducing the cost of fitness function evaluation depends on the specifics of the optimised problem, for example the value calculated as a result of simulation may be extrapolated after performing only a small part of the simulation.

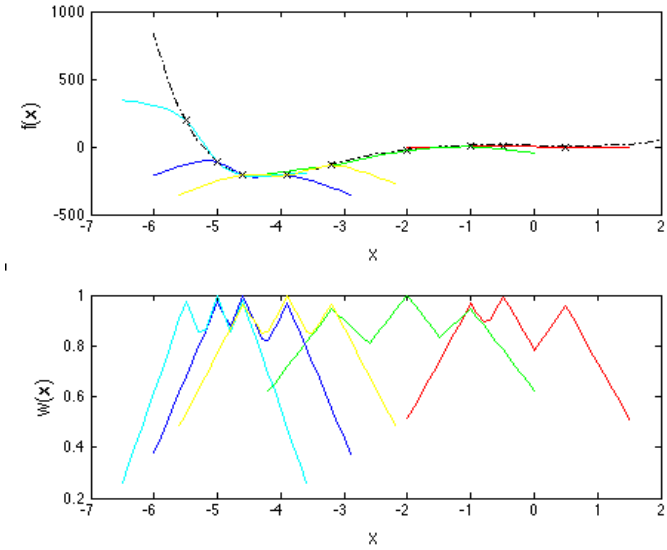

Fig. 2. Local models approximating the fitness function and their reliability

In the PSO algorithm the value of the fitness function $f(x_i)$ in position x_i is used to:

- be compared with the best value found by the particle – p_i. If $f(x_i)$ is lower than p_i ,then the value of $f(x_i)$ and position x_i are stored in memory,
- be compared with the best value found by the whole swarm (or the whole neighbourhood) – if it is better, then it is stored together with the position x_i.

If the calculated fitness function $f(x_i)$ is higher than p_i (value of g_i is always lower or equal to any of p_i's), it is – from the swarm point of view – useless: the only information used is that it is not optimal. This points to a possibility to reduce necessary calculations by stopping the simulation when the calculated value exceeds p_i with no side effects on the performance of the PSO algorithm. Further acceleration of the optimisation process is possible if the fitness function obtained after the whole simulation - a high value in particular – is approximated after performing only the beginning of the simulation. In the problem of model parameter tuning the value of fitness function results from the comparison of a model response and a signal measured in the identified system. The shape of the fitness function in time depends on the differences between those two signals and may be different depending on the type of model and input signal. Several methods of approximation of the value of fitness function after the whole simulation were proposed, examined and utilised to tune a Sugeno type fuzzy inference system using ANFIS. The shape of the function described by the fuzzy system is shown in figure 3. Its goal is to evaluate the accuracy of the fitness function approximation based on a beginning fragment of

a simulation depending on the fitness value found by the swarm (p_{in}) and the length of the performed simulation (T_{pn}).

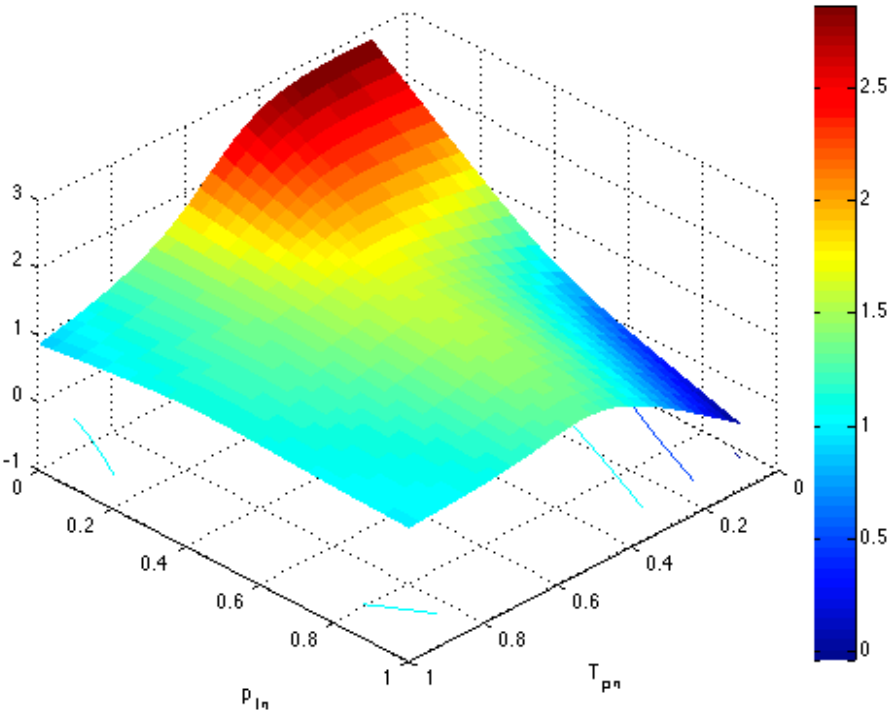

Fig. 3. Function described by the fuzzy inference system in PS-PSO algorithm

4. CONTROLLER TUNING AND MODEL IDENTIFICATION

The effectiveness of the PS-MEFAPSO algorithm (a combination of Partial Simulation PSO with fuzzy adaptation and mapping) in the problems of model parameter estimation and controller tuning was verified on a laboratory stand equipped with a DS1102 DSP Controller Board and Matlab Real Time Workshop. The DPS board allowed the collection of data necessary for a model identification and for a practical verification of the performance of the control system tuned by PSO. A diagram of the laboratory stand is shown in figure 4.

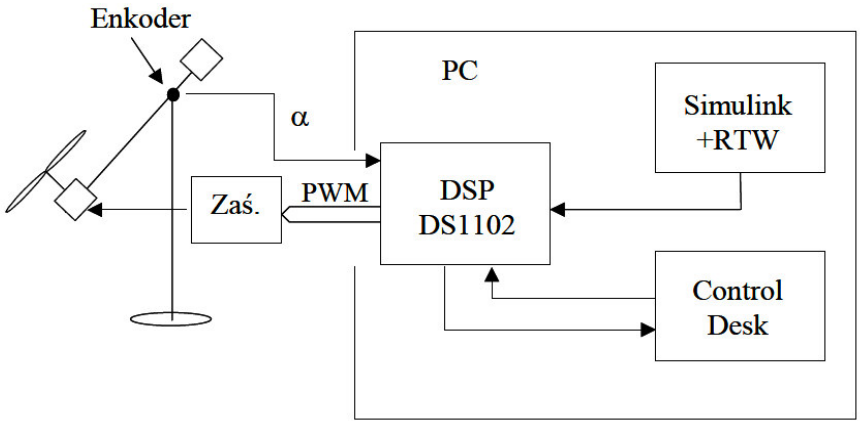

Fig. 4. Diagram of the laboratory stand

The goal of the control system is to establish a given angular displacement of the pendulum by regulating the rotational speed of the propeller motor. The motor is controlled by a pulse width modulation (PWM). Based on physical equations a model of the pendulum is proposed (figure 5). The model includes a nonlinear characteristic of the relation between torque and the PWM duty cycle $M_n(\mu)$.

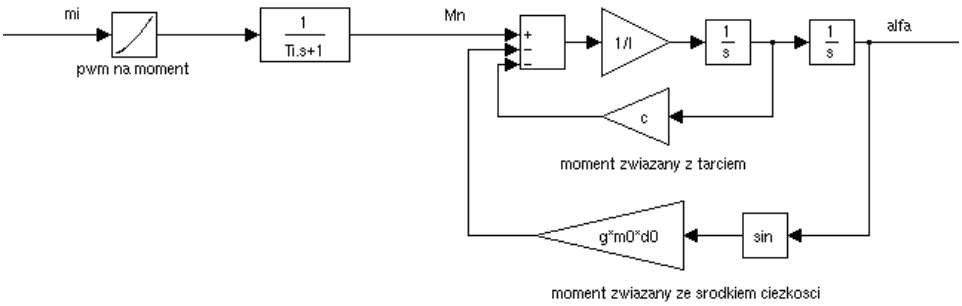

Fig. 5. Simulink model of the pendulum system used during identification and controller tuning

The values of the model parameters were identified by the proposed PS-MEFAPSO algorithm (which included three methods of fitness function evaluation and cost reduction) with the use of a measured response of the real control system. In order to verify the accuracy of the identified parameter values the modelled and measured responses to a test signal were compared. The comparison proved high accuracy of the obtained parameters. In the angular displacement control loop a PID controller with anti-windup loop was used. A controller tuning was performed by PS-MEFAPSO with the requirements of a given maximum control time and overshoot. The application of the proposed

algorithm resulted in a tenfold reduction of the total performed simulation time as compared to a standard PSO. The time is comparable to required by Simulink Response Optimisation toolbox (which uses a genetic algorithm), however the swarm obtained more favourable controller parameters.

5. CONCLUSIONS

The paper presents the results of the research into an application of a swarm intelligence algorithm PSO to solving chosen control problems, which may be described as an optimisation task: parametric identification of dynamic systems and controller tuning. In order to improve the efficiency of the optimisation process, several modifications of the algorithm, allowing an effective search for optimum while requiring a reduced number of fitness function evaluations, are proposed. They are as follows:

- a dynamic adaptation of swarm parameter values (inertia and c_1, c_2 coefficients) by a fuzzy inference system based on the swarm performance,
- an approximation of the fitness function based on the values obtained during previous iterations,
- an intelligent extrapolation of the fitness function value which is calculated as a result of a simulation – this method uses fuzzy inference and allows to skip the unnecessary part of a simulation or measurement, consequently to shorten the optimisation time.

The efficiency of the proposed algorithms was verified in the simulated and real control systems.

REFERENCES

[1] **Beni G., Wang J.**: Swarm Intelligence, Proc. of the 17th Annual Meeting of the Robotics Society of Japan, 1989.

[2] **Bonabeau E., Dorigo M., Theraulaz G.:** Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, Oxford, 1999.

[3] **Shi Y., Eberhart R.**: Fuzzy adaptive particle swarm optimization, Proc. of the Congress on Evolutionary Computation, 2001.

[4] **Saber A.Y. et al.**: Unit Commitment Computation - A Novel Fuzzy Adaptive Particle Swarm Optimization Approach, Proc. of the Power Systems Conference and Exposition, 2006.

[5] **Breaban M., Luchian H.**: PSO under an adaptive scheme, Proc. of the IEEE Congress on Evolutionary Computation, 2005.

[6] **Alotto P.**: A Multiquadrics-based Algorithm for the Acceleration of Simulated Annealing Optimization Procedures, IEEE Transactions on Magnetics nr 3, str. 1198-1201, 1996.

ZASTOSOWANIE ALGORYTMÓW WYKORZYSTUJĄCYCH INTELIGENCJĘ ROJU W PROBLEMACH STEROWANIA

Streszczenie

W pracy przedstawiono analizę możliwości zastosowania algorytmów wykorzystujących inteligencję roju w problemach sterowania, które sprowadzić można do zadania optymalizacji. Opracowano uniwersalną implementację algorytmu PSO pozwalającą na przeprowadzenie skutecznej optymalizacji przy zmniejszonej ilości wykonywanych obliczeń funkcji celu. Proponowane rozwiązania obejmują adaptacyjną zmianę parametrów roju przy pomocy układu logiki rozmytej oraz budowę modelu funkcji celu na podstawie punktów uzyskanych we wcześniejszych iteracjach. Przedstawiono sposoby przystosowania algorytmu PSO do rozwiązania zadania identyfikacji parametrycznej modeli nieliniowych oraz strojenia regulatorów (gdzie zaproponowano ekstrapolację optymalizowanego kryterium po wykonaniu niepełnej symulacji lub pomiaru). Skuteczność proponowanego rozwiązania zbadano na stanowisku badawczym wyposażonym w procesor sygnałowy.

Promotor: dr hab. inż. Jacek Kabziński, prof. PŁ
Recenzenci pracy doktorskiej:
prof. dr hab. inż. Zdzisław Kowalczuk, Politechnika Gdańska
dr hab. inż. Andrzej Bartoszewicz, prof. PŁ, Politechnika Łódzka