ICMIEE-PI-140169

Artificial Bee Colony, Firefly and Bat Algorithm in Unconstrained Optimization

*Md. Tajmiruzzaman, Md. Asadujjaman**
Department of Industrial & Production Engineering, Rajshahi University of Engineering & Technology, Rajshahi, BANGLADESH

ABSTRACT
Meta-heuristic algorithms have been proven to outperform deterministic algorithms in real world optimization problems. Artificial Bee Colony (ABC) Algorithm is a meta-heuristic optimization algorithm based on the intelligent behavior of honeybee swarm. Firefly algorithm is a recently developed algorithm inspired by the flashing behavior of fireflies. And a most recently developed algorithm is Bat algorithm which exploits the so-called echolocation of bats. In this paper, Artificial Bee Colony, Firefly and Bat algorithms are tested on some standard well-known bench-mark problems of unconstrained optimization to compare performance of these algorithms. The result indicates that, Artificial Bee Colony algorithm outperforms the other algorithms with Firefly algorithm performing better than Bat algorithm, although Bat algorithm scores best in convergence speed. This paper concludes that further in-depth researches for modification purposes and detailed parametric studies are needed for these algorithms to work best.

Keywords: Artificial Bee Colony Algorithm, Firefly Algorithm, Bat Algorithm, Unconstrained Optimization.

1. Introduction

Nature has always been an inspiration for researchers and scientists. Many nature-inspired algorithms have been developed to solve complex problems in optimization and in real world. In general, there are two main concepts developed in bio-inspired computation:

1) Evolutionary algorithms
2) Swarm based algorithms.

Evolutionary algorithms [1] are optimization techniques that base on Darwin's principle of survivor of the fittest. Such kinds of algorithms are Genetic algorithm, Evolution strategies, Genetic programming, Evolutionary programming, Differential evolution etc. Swarm intelligence is the collective behavior of decentralized, self-organized systems, either natural or artificial. Swarm intelligence was introduced by Beny, in 1989 [2]. The most well-known classes of swarm intelligence algorithms are as follows: Particle swarm optimization, Ant Colony Optimization, Artificial Bee Colony, Firefly algorithm, Cuckoo Search, Bat algorithm etc. In this paper, Artificial Bee Colony, Firefly and Bat Algorithms are checked for their performance in terms of convergence speed and precision in solving unconstrained optimization problem for single objective function.

The structure of the paper is as follows. In Section-2, Artificial Bee Colony, Firefly and Bat Algorithms are introduced. In section-3, benchmark test functions used in this paper are described briefly. Section-4 shows experimental settings of the algorithms and experimental analysis on the three algorithms. And finally, Section-5 concludes the work done.

2. Overview of the Algorithms

2.1 Artificial Bee Colony (ABC)

In 2005, D. Karaboga introduced a bee swarm algorithm called artificial bee colony algorithm for numerical optimization problems [4]; and B. Basturk and D. Karaboga compared the performance of ABC with that of some other well-known population based optimization algorithms [5]. The artificial bee colony contains three groups: scouts, onlooker bees and employed bees. The bee carrying out random search is known as scout. The bee which is going to the food source which is visited by it previously is employed bee. The bee waiting on the dance area is an onlooker bee. The bees search for the rich food sources around the hive. The employed bees store the food source information and share the information with onlooker bees. The number of food sources is equal to the number of employed bees and also equal to the number of onlooker bees. Employed bees whose solutions cannot be improved through a predetermined number of trials (that is "limit") become scouts and their solutions are abandoned [4]. in the optimization context, the number of food sources in ABC algorithm represents the number of solutions in the population. The ABC consists of four main phases:

Initialization Phase:
The food sources, whose population size is SN, are randomly generated by scout bees. Each food source, represented by x_m is an input vector to the optimization problem, x_m has D variables and D is the dimension of searching space of the objective function to be optimized. The initial food sources are randomly produced via the Eq.(1).

$$x_m = x_{min} + rand(0,1) \times (x_{max} - x_{min}) \quad (1)$$

Where x_{max} and x_{min} are the upper and lower bound of the solution space of objective function, rand (0, 1) is a random number within the range [0, 1].

* Corresponding author. Tel.: +88-01913599639
E-mail address: jonikhan007@yahoo.com

Employed Bee Phase:
Employed bee flies to a food source and finds a new food source within the neighborhood of the food source. The higher quantity food source is memorized by the employed bees. The food source information stored by employed bee will be shared with onlooker bees. A neighbor food source v_{mi} is determined and calculated by the following Eq.(2).

$$v_{mi} = x_{mi} + rand(-1,1) \times (x_{mi} - x_{ki}) \quad (2)$$

Where i is a randomly selected parameter index, x_k is a randomly selected food source, rand(-1,1) is a random number within the range [-1, 1]. The range of this parameter can make an appropriate adjustment on specific issues. The fitness of food sources is essential in order to find the global optimal. The fitness is calculated by the following Eq.(3), after that a greedy selection is applied between x_m and v_m.

$$fit_m(x_m) = \begin{pmatrix} \frac{1}{1+f_m(x_m)}, & f_m(x_m) > 0 \\ 1 + |f_m(x_m)|, & f_m(x_m) < 0 \end{pmatrix} \quad (3)$$

Where $f_m(x_m)$ is the objective function value of x_m.

Onlooker Bee Phase:
Onlooker bees calculates the profitability of food sources by observing the waggle dance in the dance area and then select a higher food source randomly. After that onlooker bees carry out randomly search in the neighborhood of food source. The quantity of a food source is evaluated by its profitability and the profitability of all food sources. Pm is determined by the Eq.(4).

$$p_m = \frac{fit_m(x_m)}{\sum_{m=1}^{SN} fit_m(x_m)} \quad (4)$$

Where $fit_m(x_m)$ is the fitness of x_m. Onlooker bees search the neighborhood of food source according to Eq.(5).

$$v_{mi} = x_{mi} + rand(-1,1) \times (x_{mi} - x_{ki}) \quad (5)$$

Scout Phase:
if the profitability of food source cannot be improved and the times of unchanged greater than the predetermined number of trials, which called "limit", the solutions will be abandoned by scout bees. Then, the new solutions are randomly searched by the scout bees. The new solution x_m will be discovered by the scout by using Eq.(6).

$$x_m = x_{min} + rand(0,1) \times (x_{max} - x_{min}) \quad (6)$$

Table 1 Pseudo code for ABC algorithm

1) **Begin**	9) **Generate** new solutions v_{mi} for the Onlooker bees using Eq.(5) and evaluate them
2) **Initialize** the solution population, i = 1, . . . , SN	10) **Keep** the best solution between current and candidate
3) **Evaluate** population	11) **Determine** if exist an abandoned food Source and replace it using a scout bee
4) cycle = 1	12) **Save** in memory the best solution so far
5) **Repeat**	13) cycle = cycle + 1
6) **Generate** new solutions v_{mi} for the employed bees using Eq.(2) and evaluate them.	14) Until cycle = M C N
7) **Keep** the best solution between current and candidate	
8) **Select** the visited solution for onlooker bees by their fitness	

2.2 Firefly Algorithm
The Firefly algorithm was introduced by Dr. Xin She yang [6,7] at Cambridge University in 2007 which was inspired by the mating or flashing behavior of fireflies. The FA is assumed as follows: 1) All fireflies are unisex, so that one firefly will be attracted to all other fireflies. 2) Attractiveness is proportional to their brightness, and for any two fireflies, the less brighter one will be attracted by the brighter one. However, the brightness can decrease as their distance increases. If there are no fireflies brighter than a given firefly, it will move randomly. 3) The brightness of a firefly is affected or determined by the landscape of the objective function. For a minimum optimization problem f(x), the light intensity I_i of a firefly i is determined by Eq.(7).

$$I_i = f(x_i) \quad (7)$$

Based on these three rules, the basic steps of the FA can be summarized as the pseudo code shown in Table 2. The initial positions of fireflies are generated at random ($x_i \in [x_{min}, x_{max}]^D$). The movement of a firefly i is attracted by another more attractive firefly j, which has better solution, is determined by Eq.(8).

Table 2 Pseudo code for Firefly algorithm

Objective function f(x), $x=(x_1,\ldots\ldots,x_D)^T$
Initialize positions of fireflies x_i (i=1,2,……,M)
Calculate Light intensities by $I_i=f(x_i)$
while (t< MaxGeneration t_{max}) **do**
  for i = 1 to M, all M fireflies **do**
    for j = 1 to M, all M fireflies **do**
      if $I_i > I_i$ **then**
        Move firefly i toward j by Eq.(8)
      end if
    end for j
  end for i
  Evaluate new solutions $f(x_i)$
  Rank the fireflies and find the current global best g_*
end while

$$x_i^{new} = x_i^{old} + \beta_{i,j}\left(x_j(t) - x_i^{old}\right) + \alpha(t)(rand(0,1) - 0.5)L \quad (8)$$

Where, the second term is due to the attraction. The attractive-ness β is determined by Eq.(9)

$$\beta_{i,j} = (\beta_o - \beta_{min})e^{-\gamma r_{i,j}^2} + \beta_{min} \quad (9)$$

Where, β_0 is the attractiveness at r = 0, β_{min} is the minimum value of β, and an absorption coefficient γ is crucially important in determining the speed of the convergence. Thus, the attractiveness will vary with the distance $r_{i,j}$ between firefly i and j.

$$r_{i,j} = \left\|x_i^{old} - x_j(t)\right\| = \sqrt{\sum_{d=1}^{D}(x_{i,d} - x_{j,d})^2} \quad (10)$$

The third term of Eq.(8) is randomization with α(t) being the randomization parameter;

$$\alpha(t) = \alpha(0)\left(1 - \left(\frac{10^{-4}}{0.9}\right)^{1/t_{max}}\right) \quad (11)$$

Where, Rand(0,1) is a random number generator uniformly distributed in [0, 1], and L is the average scale of the problem $|x_{max} - x_{min}|$. The brightest firefly k moves randomly according to Eq.(12)

$$x_k(t+1) = x_k(t) + \alpha(t)(rand(o,1) - 0.5)L \quad (12)$$

2.3 Bat Algorithm

Bat Algorithm, proposed by Yang, is inspired by echolocation characteristic of bats which they use to detect prey and to avoid obstacles [8]. These bats emit very loud sound and listen for the echo that bounces back from the surrounding objects [9]. Thus a bat can compute how far they are from an object. In order to transform these behaviors of bats to algorithm, Yang idealized some rules:

1) All bats use echolocation to sense distance, and they also know the difference between food/prey and background barriers in some magical way;
2) Bats fly randomly with velocity v_i at position x_i with a frequency f_{min} varying wavelength and loudness A_0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission r ϵ [0,1], depending on the proximity of their target;
3) Although the loudness can vary in many ways, we assume that the loudness varies from a large (positive) A_0 to a minimum constant value A_{min};

Initialization of Bat Population:

Initial population is randomly generated from real-valued vectors with dimension d and number of bats n, by taking into account lower and upper boundaries.

$$x_{ij} = x_{minj} + rand(0,1)(x_{maxj} - x_{minj}) \quad (13)$$

where i=1,2,…n, j=1, 2,….d. x_{minj} and x_{maxj} are lower and upper boundaries for dimension j respectively.

Update Process of Frequency, Velocity and Solution:

The frequency factor controls step size of a solution in BA. This factor is assigned to random value for each bat (solution) between upper and lower boundaries [f_{min}, f_{max}]. Velocity of a solution is proportional to frequency and new solution depends on its new velocity.

$$f_i = f_{min} + (f_{max} - f_{min})\beta \quad (14)$$

$$v_i^t = v_i^{t-1} + (x_i^t - x^*)f_i \quad (15)$$

$$x_i^t = x_i^{t-1} + v_i^t \quad (16)$$

Where β ϵ [0, 1] indicates randomly generated number, x^* represents current global best solutions. For local search part of algorithm (exploitation) one solution is selected among the selected best solutions and random walk is applied .

$$x_{new} = x_{old} + \varepsilon\overline{A^t} \quad (17)$$

Where $\overline{A^t}$ is average loudness of all bats, Ɛ ϵ [0,1] is random number and represents direction and intensity of random-walk.

Update Process of Loudness and Pulse Emission Rate:

Loudness and pulse emission rate must be updated as iterations proceed. As a bat gets closer to its prey then loudness *A* usually decreases and pulse emission rate also increases. Loudness *A* and pulse emission rate *r* are updated by the following equations Eq.(18) and Eq.(19):

$$A_i^{t+1} = \alpha A_i^t \quad (18)$$

$$r_i^{t+1} = r_i^0[1 - e^{(-\gamma t)}] \quad (19)$$

Where α and γ are constants. r_i^0 and A_i are factors which consist of random values and A_i^0 can typically be [1, 2], while r_i^0 can typically be [0,1].

Table 3 Pseudo code for Bat algorithm

Objective function f(x), $x = (x_1,...,x_d)^T$
Initialize the bat population x_i (i = 1, 2,...,n) and v_i
Define pulse frequency f_i at x_i
Initialize pulse rates r_i and the loudness A_i

while (t <Max number of iterations)
Generate new solutions by adjusting frequency, and updating velocities and solutions by Eq.(14) to Eq.(16) .
if (rand > r_i)
Select a solution among the best solutions
Generate a local solution around the selected best solution
end if
Generate a new solution by flying randomly
if (rand < A_i & $f(x_i) < f(x_*)$)
Accept the new solutions
Increase r_i and reduce A_i
end if
Rank the bats and find the current best x_*
end while

3. Benchmark Test Functions

Seven well-known Benchmark functions are used in our experiment to test the performance of the three algorithms. These functions are useful to evaluate characteristics of any optimization algorithms [10]. Both the unimodal and multimodal functions are used. The separability and dimensionality of these functions are worth being studied carefully [3]. In Table 4, U= unimodal, S=separable, M=multimodal, N=non-separable

Table 4 Benchmark test functions

S. NO.	***Function***	***Formulation***	***Characteristics***	***Range***	***Global minimum point***
f_1	Sphere	$f(x) = \sum_{i=1}^{D} x_i^2$	U/S	[-5.12, 5.12]	x_i =0,.,0
f_2	Step	$f(x) = \sum_{i=1}^{D} (\lvert x_i + 0.5 \rvert)^2$	U/S	[-10, 10]	x_i =0,.,0
f_3	Rosenbrock	$f(x) = \sum_{i=1}^{D} [100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2]$	M/N	[-5, 5]	x_i =1,.,1
f_4	Rastrigin	$f(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	M/N	[-15, 15]	x_i =0,.,0
f_5	Griewank	$f(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	M/N	[-600, 600]	x_i =0,.,0
f_6	Schwefel	$f(x) = D * 418.9829 + \sum_{i=1}^{D} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	M/N	[-500,500]	x_i =420,.,420
f_7	Ackley	$f(x) = 20 + e - 20e^{\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right)} - e^{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)}$	M/N	[-32, 32]	x_i =0,.,0

4. Experiments and Results

All the experiments are executed in a Fujitsu LH531 computer and the configuration of PC was Intel(R) Pentium(R) CPU B960 @ 2.20 GHz and 2 GB RAM. Each algorithm is tested with 30 independent runs for each test function and the population size (no. of bees, fireflies or bats) is fixed to 50 for each run. Three different sets of dimensions (variables) were taken into account viz: D=10, D=20, D=30 and maximum cycle numbers for each dimension were taken as 2000, 4000 and 6000 respectively. Parameter settings for three algorithms are given in Table 5. The experimental results for objective function values and processing times are shown in Table 6 and Table 7 respectively.

Table 5 Parameter settings

ABC	FA	BA
Limit: 100	α (randomness): 0.5	A_0 (loudness): 1.8
	γ(absorption): 1	α: 0.9
	β_0: 1	r_0 (pulse rate): 0.9
	β_{min}: 0.2	γ: 0.9
		Q_{min} (minimum frequency) : 0
		Q_{max} (maximum frequency): 2

Table 6 Objective function values for three algorithms

Functions	D	ABC			FA			BA		
		Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean
f_1	10	2.5511e-17	1.0062e-016	7.6965e-017	9.0434e-008	2.5190e-007	1.7624e-007	8.2939e-007	1.2646	0.1244
	20	2.0769e-016	4.9192e-016	3.3509e-016	4.7583e-007	9.8592e-007	7.0852e-007	3.8730e-006	0.2416	0.0129
	30	4.1396e-016	9.4409e-016	6.6442e-016	1.1955e-006	2.3205e-006	1.6189e-006	1.2419e-005	2.2329e-005	1.6328e-005
f_2	10	3.7136e-017	1.7645e-016	8.5316e-017	3.1041e-007	1.1208e-006	6.9179e-007	1.0309	53.8202	17.7914
	20	2.5010e-016	4.6896e-016	3.3612e-016	1.5421e-006	4.1627e-006	2.9305e-006	2.4155	93.1103	36.3792
	30	4.1167e-016	7.7174e-016	6.2551e-016	5.2488e-006	9.5321e-006	6.8868e-006	1.1551	71.2719	23.4291
f_3	10	0.0017	0.2392	0.0290	4.2452	9.3845	6.3135	1.4919	246.829	43.1519
	20	0.0038	0.7961	0.0957	14.8867	18.4818	17.0887	9.0993	204.319	54.2309
	30	2.5162e-004	0.4781	0.1112	26.1359	27.9784	26.9346	17.0327	151.151	35.0552
f_4	10	0	0	0	0.9953	8.9548	5.2404	43.7783	336.286	152.193
	20	0	5.6843e-014	1.6106e-014	7.9605	47.7589	19.9003	205.954	637.753	374.197
	30	0	1.1937e-012	2.3874e-013	18.9078	30.8471	23.8821	424.844	919.313	630.294
f_5	10	0	0.0123	0.0025	1.9674e-004	0.1382	0.0210	29.5652	86.3512	62.4397
	20	0	4.3881e-011	1.9047e-012	7.2438e-004	0.0011	8.8518e-004	47.9587	297.395	160.523
	30	1.1102e-016	5.4412e-012	5.2344e-013	0.0012	0.0087	0.0030	146.711	417.807	282.642
f_6	10	-3.4e+3	-1.6e+3	-2.4e+3	769.857	1.7372e+003	1.2673e+003	-Inf	-Inf	-Inf
	20	-4.7e+3	-2.3e+3	-3.6e+3	2.2504e+003	3.4151e+003	2.7340e+003	-Inf	-Inf	-Inf
	30	-6.3e+3	-3.9e+3	-4.8e+3	3.8494e+003	6.1592e+003	5.1997e+003	-Inf	-Inf	-Inf
f_7	10	2.6645e-015	9.7700e-015	6.2172e-015	0.0022	0.0040	0.0032	11.7922	17.8919	15.6894
	20	2.0428e-014	3.1086e-014	2.6586e-014	0.0038	0.0055	0.0048	14.1646	18.2478	16.8652
	30	4.1744e-014	6.3061e-014	4.9679e-014	0.0054	0.0066	0.0060	15.7225	18.4957	17.2802

In Table 6, we can see ABC algorithm outperforms both Firefly and Bat algorithm with Firefly algorithm performing better than Bat algorithm in all the functions. For unimodal separable functions (sphere, step), ABC algorithm outperforms FA by a factor 10^{-10} and FA outperforms BA by 10^{-7}. All the algorithms showed some difficulties to reach global minima for schwefel function. Except this, other functions reach global minima or require only a few extra iterations to reach global minima. It can also be noticed that, as dimension increases, the difficulty with finding global minima also increases. In Table 7, we can see that BA converges much faster than FA (or even than ABC algorithm) and FA performs worst of the three algorithms.

Table 7 Processing time for three algorithms

Function	D	ABC			FA			BA		
		Best	**Worst**	**Mean**	**Best**	**Worst**	**Mean**	**Best**	**Worst**	**Mean**
	10	1.7811	1.9407	1.7933	18.142	18.442	18.261	1.6009	1.658	1.6135
f_1	20	3.6818	4.0251	3.7087	37.128	37.851	37.286	3.222	3.381	3.249
	30	5.7592	6.1313	5.7857	58.554	62.314	59.367	4.9783	5.1387	5.0023
	10	1.8109	2.0030	1.8220	17.674	18.353	17.934	1.6158	1.6645	1.63
f_2	20	3.7641	4.0917	3.7940	36.896	37.151	37.025	3.3071	3.5034	3.3350
	30	5.9447	6.3332	5.9920	58.622	59.387	58.872	5.0219	5.2680	5.0610
	10	2.1526	2.3195	2.1732	18.294	18.787	18.632	2.6183	2.6819	2.6286
f_3	20	4.7530	5.0703	4.7844	39.871	40.211	39.934	5.3558	5.4489	5.3852
	30	7.8199	8.1974	7.8453	60.486	60.834	60.635	8.3112	9.3287	8.4250
	10	2.2073	2.4376	2.2449	18.270	18.720	18.559	2.4014	2.4585	2.4167
f_4	20	5.2398	5.5808	5.2816	38.067	38.439	38.147	4.9231	4.9976	4.9444
	30	9.1117	9.6187	9.1856	59.336	60.222	59.623	7.5976	7.7640	7.6367
	10	4.6223	5.0921	4.6473	20.291	21.081	20.389	5.3466	5.5116	5.3897
f_5	20	10.978	11.365	11.046	42.540	45.628	43.292	10.963	11.142	11.020
	30	18.952	19.527	19.111	67.587	68.456	67.966	16.836	17.379	16.946
	10	7.3361	7.5628	7.4105	17.390	18.976	17.625	2.8161	3.1559	2.9904
f_6	20	16.313	18.324	16.969	36.411	36.921	36.571	7.5515	9.2466	8.3519
	30	26.438	27.345	26.775	57.383	58.159	57.659	15.299	16.866	16.062
	10	7.2599	7.5470	7.3308	18.130	18.249	18.173	3.0408	3.0941	3.0561
f_7	20	15.388	15.848	15.520	38.187	38.331	38.248	6.2228	6.3651	6.2526
	30	24.111	24.676	24.213	60.330	61.086	60.650	9.4868	9.6724	9.5449

5. Conclusion

This paper compared the performance of the three algorithms in terms of accuracy and convergence speed. We have used basic versions of these algorithms without finely tuning the parameters to compare the results. From simulation results, it is turned out that, ABC algorithm gives the best result and firefly algorithm has a faster convergence speed. Although these algorithms have some difficulties with higher multimodality, it can be concluded that, these difficulties can be overcome by some modification or improvement of the algorithms and some extensive parametric studies.

References

[1] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computing. *Springer-Verlag, Berlin*, (2003).

[2] G. Beni, J. Wang, Swarm Intelligence in Cellular Robotic Systems, *Proceed. NATO Advanced Workshop on Robots and Biological Systems, Tuscany, Italy*, June 26–30, (1989).

[3] G. Hadley, Nonlinear and Dynamics Programming, *Addison Wesley, Reading, MA* (1964).

[4] D. Karaboga, An idea based on honey bee swarm for numerical optimization, *Technical Report-TR06,Erciyes University, Engineering Faculty, Computer Engineering Department*, (2005).

[5] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm, *Journal of Global Optimization*, 39-459-471,(2007).

[6] X. S. Yang, Firefly Algorithm, Stochastic Test Functions and Design Optimization, *Int. J. Bio-Inspired Computation*, Vol. 2, No. 2, pp.78–84 (2010).

[7] X. S. Yang, Nature-Inspired Metaheuristic Algorithms, *Luniver Press*,(2008).

[8] X. S. Yang, J. R. Gonzalez et al., A New Metaheuristic Bat-Inspired Algorithm, Nature Inspired Cooperative Strategies for Optimization (NISCO 2010), *Eds., Springer Press*, vol. 284, pp. 65-74, (2010).

[9] X. S. Yang and A. H. Gandomi, Bat Algorithm: A Novel Approach for Global Engineering Optimization, *Engineering Computations*, Vol. 29, Issue 5, pp.464 – 483,(2012).

[10] Marcin Molga, Czesław Smutnicki, Test functions for optimization needs, *3 kwietnia* (2005).