Plagiarism Detector v. 2129 - Originality Report 14/04/2023 08:16:57

Analyzed document: elb-13-01.pdf Licensed to: Lindung Manik

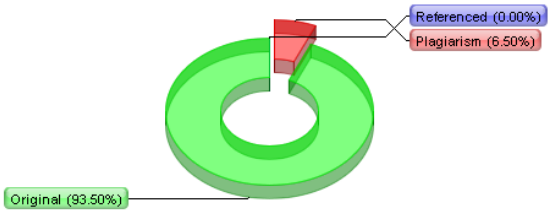
Comparison Preset: Rewrite Detected language: En

Check type: Internet Check

TEE and encoding: ifilter

Detailed document body analysis:

Relation chart:



Distribution graph:

Top sources of plagiarism: 33

4%	ABC	214	1. https://shsfeapi1.pdc-gate2.com/get_doc.php?id=6215/MTAuMTAwNy85NzgtMy0zMTktNTEyODEtNQ==.txt
3%	ABC	354	2. https://shsfeapi1.pdc-gate2.com/get_doc.php?id=5844/MTAuMTAwMi9hcnQuMzk0NDg=.txt
2%	ABC	54	3. https://shsfeapi1.pdc-gate2.com/get_doc.php?id=5282/MTAuMTEwOS9JTUNDQy4yMDE1LjMyMw==.txt

Processed resources details: 222 - Ok / 16 - Failed

Important notes:

Wikipedia:	Google Books:	Ghostwriting services:	Anti-cheating:
Wiki Detected!	[not detected]	[not detected]	[not detected]

UACE: UniCode Anti-Cheat Engine report:

1. Status: Analyzer **On** Normalizer **On** character similarity set to **100%**
2. Detected UniCode contamination percent:**0.5%** with limit of: 4%
3. Document not normalized: percent not reached 5%
4. All suspicious symbols will be marked in purple color:Abcd...
5. Invisible symbols found: 0

Assessment recommendation:

No special action is required. Document is Ok.

Alphabet stats and symbol analyzes:

Active References (Urls Extracted from the Document):

No URLs detected

Excluded Urls:

1. **https://www.academia.edu/69630355/Features_Engineering_with_Generating_Genetic_Algorithm_for_Software_Effort_Estimation**
2. **https://www.researchgate.net/profile/Lindung-ParningotanManik/publication/358137604_Features_Engineering_with_Generating_Genetic_Algorithm_for_Software_Effort_Estimation/links/61f20eec5779d3**
Engineering-withGenerating-Genetic-Algorithm-for-Software-Effort-Estimation.pdf
3. **http://www.icicelb.org/ellb/contents/2022/1/elb-13-01-01.pdf**

Included Urls:

No URLs detected

Detailed document analysis:

ICIC Express Letters Part B: Applications ICIC International c 2022 ISSN 2185-2766 Volume 13, Number 1, January 2022 pp. 1-9 FEATURES ENGINEERING WITH GENERATING GENETIC ALGORITHM FOR SOFTWARE EFFORT ESTIMATION LINDUNG PARNINGOTAN MANIK1,2 1Research Center for Informatics National Research and Innovation Agency Jl. Sangkuriang/Cisitu No. 21/154D, Bandung 40135, Indonesia 2Faculty of Information Technology University of Nusa Mandiri Jl. Kramat Raya No. 18, Jakarta 10450, Indonesia lind004@lipi.go.id; lindung.lpm@nusamandiri.ac.id Received March 2021; accepted June 2021 ABSTRACT. In software engineering, effort estimation is the process of predicting the most realistic amount of effort, which could be expressed in terms of person-hours or person-months, required to develop software. The software effort estimation is a difficult task since different factors could have different effects on the efforts in the various envi- ronments. Thus, mining the project repositories based on historical data in an efficient manner is a critical issue to make an accurate estimation. Machine learning algorithms are used in this case as the solutions to predict the effort by considering the software development factors and environments as the machine learning features. Over the years, most of the research works in this field dealt with the attributes selection and features weighting to improve the estimations accuracy. This paper investigates generating ge- netic algorithm as the feature engineering approach for the software development effort estimation. This method may select some features, and it also may create new features from the original set of features. The experiment results show that the proposed method makes impressive performance improvement. Keywords: Software effort, Machine learning, Feature engineering, Genetic algorithm 1. Introduction. A software development project's success is influenced by many fac- tors, including executive support, user involvement in the process, team experience, clear business objectives, and software infrastructure. Other factors are related to the project's timing and scope, including the minimal scope and reliable estimation [1]. The more accurate the estimation is, the more likely the software project is to be successful. The estimation at the early stage of software development is also very beneficial to make a project schedule, financial budgeting, and resource management plan. Software development effort estimation (SDEE) predicts how many resources are need- ed to complete a project plan in terms of person-hours or person-months. To make an accurate estimation, all software development factors and its environments must be con- sidered, such as software physical size estimation (as expressed in the size of lines of codes), and the functionalities or the requirements of the software (as conveyed in the function points or use-case points). There are various techniques, estimation models, and tools used for software estimation. The most common approach and easy to implement is by using expert judgment [2]. In this approach, a project estimator tends to use their expertise based on historical data and similar projects to estimate the software effort. This method is very subjective and lacks standardization; thus, it cannot be reusable. DOI: 10.24507/icicelb.13.01.1 1 2 L. P. MANIK Another approach to estimating the software effort is using algorithmic models such as the constructive cost model (COCOMO) or the system evaluation and estimation of resource-software estimation model [3]. The main driver of these models is the software size estimation, usually the source of lines of code (SLOC). To estimate the SLOC, func- tion points [4] or use-case points [5] analysis is used. Besides the physical software size, the software effort is also computed by considering another set of cost drivers that in- clude subjective assessment factors related to the project's environmental and technical complexity, such as hardware, product, project, and personnel attributes. Recently, data mining usage with machine learning techniques has been an active re- search area to estimate software development effort [6] in conjunction or as an alternative to the algorithmic models. The analogy-based estimation (ABE) approach uses the un- derlying algorithmic model principle, which characterizes the project in features. It can be done by collecting the completed projects, then training the machine learning algo- rithms with the historical data to predict new projects' software effort. Various machine learning algorithms have been applied for software effort estimation, including gradient boosting machine and deep learning [7]. There are various obstacles encountered when using the machine learning approach to estimate the software development effort. One of them is to find the most relevant features that represent the resulting effort value. In addition to that, noisy features of the data set also affect the accuracy of the estimation. Combining the machine learning algorithms with features selection and weighting process which choose the best set of features and give relevant weights has been proven as a better solution to improve estimation accuracy [8]. It also reduces the complexity of a model and makes it easier to interpret. Popular methods that are frequently used in searching for the best solution automatical- ly using attributes selection and feature weighting are stepwise regression like forward or backward selection and bio-inspired metaheuristic approaches such as particle swarm op- timization [9] or genetic algorithm [10]. These techniques are classified as a wrapper-based approach that involves training a learner during the searching process. This approach's primary disadvantage is that it is computationally slow, but it gives the best feature selection and weights in terms of accuracy. Nevertheless, the attributes selection and feature weighting only are often not enough. Transforming original features to create new ones could provide superior performance compared to the model developed on the original features in predicting the outcomes [11]. This paper's major contribution is a novel feature engineering approach in the application of software effort estimation domain. The proposed method may select features and may also create new features from the original feature set. It is derived from the genetic algorithm since the algorithm can find a high-quality solution in the full search within a reasonable period of time [12]. As the main advantage of this study, the proposed approach is designed to improve the estimator's accuracy in existing studies. The rest of this paper is structured as follows. In Section 2, the research methods are presented. Meanwhile, the results and discussions are described in Section 3. Lastly, the conclusion is given in Section 4. 2. Research Methods. Like the particle swarm optimization (PSO) algorithm, the ge- netic algorithm (GA) is also a heuristic technique that is used to construct valuable solutions to optimization and search problems. This heuristic starts with a group of ran- domly generated populations, evaluates, updates the population with a new generation, and performs a random search for the

optimum. The GA imitates the natural evolution process, reflecting

the process of natural selection where the fittest individuals from a population are chosen for reproduction to generate offspring of the next generation.

A population is a set of chromosomes or, in this context, a set of individual solutions where each individual is composed of genes or feature vectors. ICIC EXPRESS LETTERS, PART B: APPLICATIONS, VOL.13, NO.1, 2022 3 n n xi To generate an optimum solution within the possible solution set, the GA uses tech- niques inspired by natural evolution, such as selection, crossover, and mutation. In the proposed approach, transformations of the feature vectors are performed to boost the al- gorithm's performance. Unlike the simple GA, the proposed method, generating genetic algorithm (GGA), creates new features and thus may alter the individual's length. It is also known to be a blending process of selecting and generating attributes. Thus, a specialized mutation technique, namely generating mutation, is introduced. The GGA does not modify the original features until it achieves accuracy improvement by adding or deleting (or both) features. The pseudo-code of the GGA is shown in Algorithm 1. At the first step, an initial population is created with a predetermined size, s. Then, the crossover operation is performed (lines 10-15). The operation recombines the genetic information (features) of two individuals. Then, the generating mutation operation (lines 16-23) performs one of the followings with different probabilities: - Add a randomly selected original attribute to the feature vector. - Add a newly generated attribute to the feature vector. The new features are cre- ated by applying feature generators such as basic arithmetic operations, and power functions, to the feature vector's random subset. - Remove a selected attribute randomly from the feature vector. The selection operation is performed to generate a new generation of the population (lines 24-41). It is composed of the best individual in the current population with the maximum fitness value and others that win the selection tournament. The searching stops when it meets one of two criteria, either the number of generations exceeds the agreed maximum number of generations, maxgen, or the determined maximum fitness value, maxfit, is less than or equals to the best individual's fitness value. In the GA, the fitness function is used as the performance metric to evaluate how close the solution is to the problem's optimal solution. In this proposed method, the accuracy of the effort estimation is selected as the fitness criterion. The accuracy is calculated by computing the prediction error, which is the difference between actual and estimated effort. The smaller the error, the more accurate the prediction. In this research, the magnitude of relative error (MRE) is chosen as the performance metric. It is the absolute deviation of prediction from the actual value divided by the actual value. Equation (1) computes the error where yi is the predicted value and xi is the actual value at the ith sample. Moreover, the mean magnitude of relative error (MMRE) is also considered as the sample average of the MRE 's. It is computed by Equation (2), where n is the number of the samples, y1, y2, . . . , yn are the predicted values, and x1, x2, . . . , xn are the actual values. Fitness value in the GA is used for optimization purposes and must always be maximized. Since the error is better, the smaller the value is, the fitness is computed by negating the error, shown by Equation (3). As the best solution's error value is zero, then the maximum fitness value, maxfit, should also be zero. MRE = |yi − xi| xi (1) MMRE = 1 ∑ |yi − xi| (2) fitness = −1 × MMRE (3) The data sets are shown in Table 1. Three machine learning algorithms are selected in this experiment as the base regressors, that is, the regression tree (RT) using least square as the split criterion and a maximum depth of 10, the k-nearest neighbor (KNN) using Euclidean distance as similarity function and k of 5, and the generalized linear model (GLM) using Gaussian family. Shuffled five-fold cross-validation is used for learning and i=1 4 L. P. MANIK } { Data: population size s, maximum fitness maxfit, maximum number of generations maxgen, feature vector f , number of iterations i = 0, feature generator vector g = basic arithmetic operation, square roots, power functions, trigonometric function, exp value, log value, signum value, absolute value, floor ceil , tournament fraction tf = 0.25 Result: the best individual from population p 1 p ← {p1, p2, . . . , ps} where pi ∈ p has feature vector of a random subset of f ; 2 maxfp ← −100; 3 foreach individual pi ∈ p do 4 maxfp ← max(fitness(pi), maxfp); 5 end 6 while i maxgen and maxfit ≤ maxfp do 7 i ← i + 1; 8 m ← 0; 9 p' ← p; 10 while length(p') 1 do 11 {px, py} ← p' random selection; 12 remove {px, py} from p'; 13 crossover(px, py); 14 insert(px, py) into p; 15 end 16 foreach individual pi ∈ p do 17 p'i ← pi; 18 add an original feature randomly to p'i; 19 gi ← g random selection; 20 generate features mutation by applying gi to a random subset of features vector of p'i; 21 remove features randomly from p'i; 22 insert p'i into p; 23 end 24 foreach j ← 0: length(p) do 25 if fitness(pj) maxfp then 26 maxfp ← fitness(pj); 27 m ← j; 28 end 29 end 30 newgen ← {pm}; 31 ts ← length(p) × tf ; 32 while length(newgen) s do 33 winner ← ∅; 34 foreach k ← 0 : ts do 35 challenger ← p random selection; 36 if fitness(challenger) fitness(winner) then 37 winner ← challenger; 38 insert winner into newgen; 39 end 40 end 41 end 42 p ← newgen; 43 end 44 return p1 Algorithm 1: Generating genetic algorithm ICIC EXPRESS LETTERS, PART B: APPLICATIONS, VOL.13, NO.1, 2022 5 TABLE 1. Data sets No Dataset Records Features Effort (unit of measures) 1 Albrecht [13] 24 8 Person-hours 2 China [14] 499 19 Person-hours 3 Cocomo81 [15] 63 17 Person-months 4 Deshernais [16] 81 12 Person-hours 5 Finnish [17] 38 9 Person-hours 6 Kemerer [18] 15 8 Person-months 7 Kitchenham [19] 145 10 Person-hours 8 Maxwell [20] 62 27 Person-hours 9 Miyazaki94 [21] 48 9 Person-months 10 NASA60 [22] 60 17 Person-months 11 NASA93 [23] 93 24 Person-months 12 SDR [24] 12 25 Person-months testing data. Finally, a null hypothesis significance testing is used to compare the models with a predetermined significant level. It is a test designed to determine if two related treatments are statistically significantly different. A null hypothesis proposes that there is no significant difference between the models. 3. Results and Discussions. In this experiment, the PSO algorithm is compared with the proposed features engineering method. The comparison is performed with the pro- posed GGA without generating mutation (GGA1) and with generating mutation (GGA2). All features along with their weight that was greater than zero were utilized by the machine learning algorithms. In all experiments,

the maximum number of generations, maxgen, is limited to 30, and the value of the

initial population size, s, is set to 5. At first, the experiments on 12 data sets are conducted using

the base regressors to estimate the effort label in the data sets. Not all attributes in the data sets are used as machine learning features since some do not exist in the early software development stage. For instance, attributes of "duration" and "productivity" can be only obtained after the development process. Thus, these attributes are not included in the base features. The number of initial features is shown in Table 2 in the 'Base' column. The table also reports the number of used features in the other columns as the results of experimenting with the implementation of the PSO, the GGA1, and the GGA2 using the base regressors. Meanwhile, the number in the brackets represents the number of generated features that are utilized by the GGA2. From the results, it can be highlighted that the GGA-based models utilized fewer features than the PSO-based models. It indicates that the models generated by the GGA were simpler than the PSO. The relative errors of implementing the different approaches using the RT, the KNN, and the GLM as the base regressors are shown in Table 3 where the lowest error for each data set is boldly printed. On average, the base performance of the KNN was better than the RT and the GLM. This result is also confirmed by [25] which states the KNN is the most suitable single technique after the support vector regressor (SVR). It might be due to the KNN's simplicity that makes no assumption with the correlations (linear relationship) between the features and the label. Surprisingly, the GLM has the worst performance out of the RT and the KNN even though it is supported by [26] which reports that linear models like neural networks and simple linear regression perform much worse than other learners. If the hyperparameters of the GLM learning algorithm were tuned during the experiments, the model's performance could be better. Moreover, on average, the attributes selection and features weighting using the PSO increased the base performance by 20% for the RT, 22% for the KNN, and only 2% for 6 L. P. MANIK TABLE 2. The number of used features in the RT, the KNN, and the GLM as base regressors No Data set Base RT + RT + RT + KNN + KNN + KNN + GLM + GLM + GLM + PSO GGA1 GGA2 PSO GGA1 GGA2 PSO GGA1 GGA2 1 Albrecht 7

2 2 2 (1) 5 2 2 (1) 6 3 2 (6) 2 China 6 5 1 1 (1) 4 3 3 (3) 6 3 3 (4) 3 Cocomo81 16 12 7 5 (2) 5 9 7 (3) 9 13 9 (5) 4 Deshernais 8 5 2 2 (4) 6 4 2 (2) 3 2 1 (2) 5 Finnish 4 3 1 1 (2) 3 1 2 (4) 2 1 1 (5) 6 Kemerer 5 5 1 1 (2) 4 2 1 (1) 4 2 2 (5) 7 Kitchenham 3 2 1 1 (5) 3 2 2 (4) 1 1 1 (4) 8 Maxwell 24 13 3 6 (1) 17 1 2 (3) 19 2 2 (0) 9 Miyazaki94 7 3 2 1 (3) 3 1 1 (3) 7 1 1 (0) 10 NASA60 16 11 3 2 (1) 15 12 1 (2) 10 5 2 (2) 11 NASA93 22 14 3 1 (1) 16 6 1 (2) 18 8 1 (1) 12 SDR 23 20 8 1 (2) 12 9 1 (2) 13 4 1 (5) Average 67% 25% 36% 69% 40% 41% 69% 34% 42% TABLE 3. The relative errors using the RT, the KNN, and the GLM as base regressors RT KNN GLM Data set Base PSO GGA1 GGA2 Base PSO GGA1 GGA2 Base PSO GGA1 GGA2 Albrecht 1.025 0.754 0.754 0.754 0.859 0.740 0.816 0.845 1.411 1.406 1.784 1.165 China 2.075 1.913 1.683 1.649 1.780 1.478 1.581 1.555 3.025 3.025 3.390 3.154 Cocomo81 1.789 1.302 1.072 1.132 1.920 1.234 1.919 1.909 18.18 18.17 18.20 18.04 Deshernais 0.762 0.667 0.699 0.684 0.685 0.682 0.657 0.639 1.216 0.957 0.999 0.909 Finnish 1.005 0.937 1.114 0.881 1.114 0.939 1.114 1.065 2.640 2.639 2.639 2.636 Kemerer 1.692 1.692 1.692 0.558 1.052 0.872 0.896 0.881 1.496 1.495 1.568 1.210 Kitchenham 2.011 2.001 2.033 1.020 0.901 0.805 0.901 0.831 2.095 1.917 1.917 1.599 Maxwell 0.762 0.511 0.615 0.597 0.615 0.509 0.608 0.570 2.021 2.021 1.948 1.837 Miyazaki94 2.472 1.524 1.499 1.066 1.001 0.513 0.520 0.463 2.103 2.103 1.196 1.410 NASA60 0.518 0.455 0.445 0.467 0.442 0.363 0.456 0.430 5.036 4.954 5.001 0.715 NASA93 1.425 1.361 1.221 0.826 1.460 0.952 1.175 1.329 7.150 6.948 7.158 2.243 SDR 1.751 0.705 0.670 0.670 0.571 0.571 0.571 0.537 1.409 1.263 1.041 0.730 Average 1.441 1.152 1.125 0.859 1.033 0.805 0.935 0.921 3.982 3.908 3.904 2.970 the GLM. On the other hand, the GGA1 was no better than the PSO since it improved the base regressors' performance by 22% for the RT, only 9% for the KNN, and 2% for the GLM on average. Nevertheless, the features engineering with the proposed method, the GGA2, made a further improvement on the base performance by 40% for the RT, 11% for the KNN, and 25% for the GLM on average. With fewer features, the performance of GGA1 is comparable to the PSO using the RT and the GLM as the base regressors. However, the performance of GGA2 is superior to the PSO for those regressors. On the other hand, the PSO makes impressive improvements using the KNN regressor. It might be due to proper weights assigned by the PSO to the KNN. It is supported by [27], which states that the PSO with the ABE method like the KNN leads to a high-performance model. Since the distribution of the variance between the means of relative errors cannot be considered to be normally distributed, the paired two-tailed Wilcoxon signed-rank test is chosen as a non-parametric statistical hypothesis test [28] to compare the model's performance by computing the pvalue. It is the likelihood of achieving test outcomes

at least as extreme as the results currently obtained, assuming that the null hypothesis is correct [29]. A low pvalue indicates that such an unusual observed result will be improbable under the null hypothesis. If the value is greater than the predetermined significance level of 0.05, then the null hypothesis cannot be rejected. ICIC EXPRESS LETTERS, PART B: APPLICATIONS, VOL.13, NO.1, 2022 7 The results of pvalue for each approach using the RT, the KNN, and the GLM as the base regressors are shown in Tables 4, 5, and 6, respectively. As can be inferred from the test results, the PSO implementation and the proposed method, GGA2, have significant differences with all base regressors. It means that the PSO and the GGA2 make an impressive performance improvement in estimating the software effort. Furthermore, there are insignificant differences between the PSO and the GGA1 using the RT and the GLM as the base regressors. As can be seen from Tables 4 and 6, the pvalue amounted 0.50926 and 0.54186 respectively. This result confirms [30] that a simple genetic algorithm does not have a significant distinction with the PSO when used as a feature selection method. Nevertheless, the PSO elevates the performance essentially when used as feature selection and weighting for the KNN regressor since it has notable differences with all approaches. As

also can be seen from Table 5, there is no significant distinction between GGA1 and GGA2 for the KNN since the pvalue is greater than 0.05. Moreover, from Table 6, it can be seen that

GGA1 failed to improve the GLM base performance since the pvalue is much higher than 0.05. However, the proposed method, GGA2, produces a further gain on the performance since it has notable differences, even with the PSO, using the RT and the GLM as the base regressors. TABLE 4. Significant test results using the RT as the base regressor RT RT + PSO RT + GGA1 RT + GGA2 RT 0.00338 (sig.) 0.01278 (sig.) 0.00222 (sig.) RT + PSO 0.00338 (sig.) 0.50926 (not sig.) 0.02642 (sig.) RT + GGA1 0.01278 (sig.) 0.50926 (not sig.) 0.04660 (sig.) RT + GGA2 0.00222 (sig.) 0.02642 (sig.) 0.04660 (sig.) TABLE 5. Significant test results using the KNN as the base regressor KNN KNN + PSO KNN + GGA1 KNN + GGA2 KNN 0.00338 (sig.) 0.02780 (sig.) 0.00222 (sig.) KNN + PSO 0.00338 (sig.) 0.00758 (sig.) 0.03400 (sig.) KNN + GGA1 0.02780 (sig.) 0.00758 (sig.) 0.09894 (not sig.) KNN + GGA2 0.00222 (sig.) 0.03400 (sig.) 0.09894 (not sig.) TABLE 6. Significant test results using the GLM as the base regressor GLM GLM + PSO GLM + GGA1 GLM + GGA2 GLM 0.00512 (sig.) 0.47770 (not sig.) 0.00374 (sig.) GLM + PSO 0.00512 (sig.) 0.54186 (not sig.) 0.00480 (sig.) GLM + GGA1 0.47770 (not sig.) 0.54186 (not sig.) 0.00758 (sig.) GLM + GGA2 0.00374 (sig.) 0.00480 (sig.) 0.00758 (sig.) 4. Conclusions. A novel feature engineering approach for software effort estimation is proposed in this paper.

Plagiarism detected: **0.61%** https://shsfeapi1.pdc-gate2.com/get_doc.php?... id: **6**

The proposed method is applied to selecting features and generat- ing new features from the original feature subset. The algorithm is employed to deal with the

noise attributes problem and improve the estimation accuracy. A comparative study of the

Plagiarism detected: **0.96%** https://shsfeapi1.pdc-gate2.com/get_doc.php?... + 5 id: **7**

resources!

implementation of the proposed approach and the particle swarm optimization as the state-of-the-art using three machine learning algorithms applied to 12 data sets with the context of estimating software development effort was conducted. The experimental results show that the proposed method

made an impressive prediction improvement to the base performances. Furthermore, it achieved the lowest relative error 8 L. P. MANIK with fewer features using the regression tree and the generalized linear model as the base regressors.

Plagiarism detected: **0.65%** https://shsfeapi1.pdc-gate2.com/get_doc.php?... id: **8**

On the other hand, the particle swarm optimization approach obtained the highest accuracy using the k-nearest neighbor regressor. From the comparison results, it also can be concluded that

there is no significant diff- erence between the particle swarm optimization and the generating genetic algorithm without generating mutation when utilized as feature selection using the regression tree and the generalized linear model as the base regressors. However, the last-mentioned method tends to utilize fewer features. Moreover, with generating mutation, the algorithm makes a further significant difference. Blending feature selection with generating new features works better than combining the feature selection and weighting in this case. More research will be conducted for investigating and benchmarking the metaheuristic methods to optimize the hyperparameters of the machine learning algorithms. Acknowledgment. The computation in this work has been done using the facilities of HPC LIPI, the Indonesian Institute of Sciences (LIPI).

Plagiarism detected: **0.42%** https://shsfeapi1.pdc-gate2.com/get_doc.php?... + 2 id: **9**

resources!

The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.